Technische Universität München
TUM School of Computation, Information and Technology

Computational Complexity of Verifying Parameterized Systems

Balasubramanian Ayikudi Ramachandrakumar

Vollständiger Abdruck der von der TUM School of Computation, Information and Technology der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Die Dissertation wurde am 10.07.2023 bei der Technischen Universität München eingereicht und durch die TUM School of Computation, Information and Technology am 26.03.2024 angenommen.

Abstract

Parameterized systems are distributed systems with an arbitrary number of participating agents. The verification of such systems amounts to proving that some property holds for any population of agents. Its dual problem is to find the existence of some population of agents which violates a given property. Almost all verification problems are undecidable for parameterized systems in which the agents are allowed to have identities. Even in the case of systems with anonymous agents, the state space is infinite, which makes the automated verification of such systems a challenging task. Various general techniques have been discovered to obtain decidability results for parameterized systems with anonymous agents; two such techniques are the theory of well-quasi-orders and decision procedures for linear arithmetic theories. In this thesis, we focus on the computational complexity of verifying parameterized systems by refining and applying these two families of techniques.

In the first part of the thesis, we focus on transition systems with an underlying compatible well-quasi-order on their configuration space, called Well-Structured Transition Systems (WSTS). Many safety properties for WSTS can be reduced to the so-called coverability problem. The complexity of the coverability problem primarily depends upon the maximum length of certain sequences over the underlying well-quasi-order called controlled bad sequences. As our first contribution, we provide upper bounds on the length of controlled bad sequences for various families of well-quasi-orders. We then apply these results to obtain upper bounds on the complexity of the coverability problem for some parameterized systems. In addition to these upper bounds, we also provide complexity lower bounds for the coverability problem for some parameterized systems.

In the second part of the thesis, we focus on the application of decision procedures for linear arithmetic theories to parameterized verification. We combine results from linear arithmetic theories over the natural numbers, integers and rationals to provide efficient algorithms for verifying properties about two different classes of parameterized systems. Our algorithms can be used to analyze fault-tolerant distributed protocols from the literature. Empirical evaluations indicate that they perform well in practice.

Zusammenfassung

Parametrisierte Systeme sind verteilte Systeme mit einer beliebigen Anzahl von teilnehmenden Agenten. Die Verifikation solcher Systeme läuft darauf hinaus, zu beweisen, dass eine bestimmte Eigenschaft für eine beliebige Population von Agenten gilt. Das duale Problem besteht darin, die Existenz einer Population von Agenten zu finden, die eine bestimmte Eigenschaft verletzt. Fast alle Verifikationsprobleme sind unentscheidbar für parametrisierte Systeme, in denen die Agenten Identitäten haben dürfen. Selbst im Fall von Systemen mit anonymen Agenten ist der Zustandsraum unendlich, was die automatische Verifikation solcher Systeme zu einer schwierigen Aufgabe macht. Es wurden verschiedene allgemeine Techniken entdeckt, um Ergebnisse zur Entscheidbarkeit von parametrisierten Systemen mit anonymen Agenten zu erhalten; zwei dieser Techniken sind die Theorie der Wohl-Quasi-Ordnungen und Entscheidungsverfahren für lineare arithmetische Theorien. In dieser Arbeit konzentrieren wir uns auf die Komplexität der Verifikation parametrisierter Systeme durch Verfeinerung und Anwendung dieser beiden Familien von Techniken.

Im ersten Teil der Arbeit konzentrieren wir uns auf Übergangssysteme mit einer zugrundeliegenden kompatiblen Wohl-Quasi-Ordnung auf ihrem Konfigurationsraum, genannt Well-Structured Transition Systems (WSTS). Viele Sicherheitseigenschaften für WSTS lassen sich auf das sogenannte Überdeckbarkeitsproblem reduzieren. Die Komplexität des Überdeckbarkeitsproblems hängt in erster Linie von der maximalen Länge bestimmter Sequenzen über der zugrundeliegenden Wohl-Quasi-Ordnung ab, die als kontrollierte schlechte Sequenzen bezeichnet werden. Als ersten Beitrag liefern wir obere Schranken für die Länge von kontrollierten schlechten Sequenzen für verschiedene Familien von Wohl-Quasi-Ordnungen. Anschließend wenden wir diese Ergebnisse an, um obere Schranken für die Komplexität des Überdeckbarkeitsproblems für einige parametrisierte Systeme zu erhalten. Zusätzlich zu diesen oberen Schranken liefern wir auch untere Schranken für die Komplexität des Überdeckbarkeitsproblems für einige parametrisierte Systeme.

Im zweiten Teil der Arbeit konzentrieren wir uns auf die Anwendung von Entscheidungsverfahren für lineare arithmetische Theorien auf parametrisierte Verifikation. Wir kombinieren Ergebnisse aus linearen arithmetischen Theorien über die natürlichen Zahlen, ganzen Zahlen und rationalen Zahlen, um effiziente Algorithmen für die Verifikation von Eigenschaften über zwei verschiedene Klassen parametrisierter Systeme zu entwickeln. Unsere Algorithmen können für die Analyse von fehlertoleranten verteilten Protokollen aus der Literatur verwendet werden. Empirische Auswertungen zeigen, dass sie in der Praxis gut funktionieren.

Acknowledgments

First, I would like to thank my advisor Javier Esparza, for accepting me as a doctoral student and giving me the freedom to pursue my research. He has been a wonderful guide and a great person to collaborate with. I have learned a lot from him, both academically and otherwise. I will forever be grateful for his mentorship.

I want to thank Igor Walukiewicz, for his mentorship during my Master's education and Ahmed Bouajjani, for agreeing to review this thesis. I would also like to thank Nathalie Bertrand, Nicolas Markey, Sylvain Schmitz, Philippe Schnoebelen, Rupak Majumdar and Georg Zetzsche, whose guidance has been integral to my research.

I am indebted to all the professors at Chennai Mathematical Institute who have taught me everything about computer science. Without their teaching and counsel, this thesis would not have been possible.

I would like to thank all my (other) co-authors: Marijana Lazić, Mikhail Raskin, Timo Lang, Revantha Ramanayake, Chana Weil-Kennedy, K. S. Thejaswini and Lucie Guillou. I am glad that I was able to publish a paper with them.

I am grateful to all my colleagues at the Informatics Chair 7 at TUM who made it fun to be there. Special thanks to Chana, who was always a great person to talk to about anything, and Christoph and Muqsit, who were amazing companions, both to work with and hang out with. I am also grateful to my colleagues and friends at MPI-SWS - especially Ram, Ashwani and Satya - for a fantastic time during my brief visit there.

My gratitude to my friends from CMI - particularly Ahad, Thejaswini, Siva Tej and Bishal is immeasurable. They have been with me during some of the lowest points of my life so far and have always been kind to me and supported me despite all my mistakes. They are an integral part of my life, and my memories with them are something that I will cherish forever.

I want to thank my family for being a constant pillar of support throughout my life. My parents, my aunt and my brother ensured that I had a fabulous childhood. I am particularly grateful to my cousin, Rajesh, for instilling in me a love for science and math. It was because of his constant encouragement and support that I was able to attend CMI and pursue a degree in computer science. I dedicate this thesis to him.

Finally, I would like to thank my wife, Soundarya, who is one of the most amazing women that I know. Her mere presence in my life has made it infinitely more beautiful, and I am grateful for each and every day that I spend with her.

Contents

Chapter 1

Introduction

Distributed systems, in the most general sense, comprise a collection of agents or processes performing some actions and interacting with each other by some communication mechanism. *Parameterized systems* are distributed systems in which the number of participating agents is not fixed a priori. Such systems are designed to work for any population of agents. They form a particularly relevant class of distributed systems, as in many applications, it might be infeasible or impossible to know the amount of participating agents. Common examples include mobile networks, robotic swarms, web services, blockchains and molecular systems. The ubiquity of such systems then justifies the need for studying and developing techniques to analyze them and verify their correctness for *any number* of agents, often called the *parameterized verification* problem. For example, one possible property that we would like to prove in a safety-critical parameterized system is to show that in any population, no agent ever enters a faulty state during the course of a computation.

Since parameterized systems can be extremely complex, it would be desirable to have algorithmic techniques which can be deployed to *automatically* verify their correctness. Unfortunately, it is known that this is impossible, even for simple systems in which the agents are allowed to have unique identities [11]. Consequently, much of the bulk of research on automatic verification of parameterized systems has concentrated on systems with anonymous agents, i.e., systems without identities [35]. Such systems also prove to be extremely important from a modeling perspective. For example, molecular systems, both natural and computational, might simply just be molecules in a solution with no identifiers. Further, in various fault-tolerant distributed algorithms, processes only care about *how many* other processes have sent a message, rather than *which* processes have sent a message. In such cases, it becomes extremely useful to abstract away the identities and only reason about the number of different types of participating processes.

Even in the absence of identities, reasoning about parameterized systems require us to deal with an unbounded number of processes, thereby giving rise to *infinite-state* systems which makes the underlying verification problem a challenging task. Further, depending on the precise communication mechanism, computational power of the agents and the underlying property that needs to be verified, we obtain a massive number of models. Each such model might be better suited to model some particular application and can either offer more expressive power or be more efficient with respect to some measure when compared with other systems. For this reason, there exists a plethora of well-studied parameterized models in the literature.

Various general techniques have been proposed in the literature for analyzing and verifying these different classes of parameterized systems [35]. Among this array of techniques, two are well-known: the first is the theory of well-quasi-orders and well-structured transition systems, and the second is the collection of efficient decision procedures for linear arithmetic theories. In this thesis, we refine, enhance and apply these techniques from a *complexity-theoretic perspective* to various parameterized systems to obtain new, and in some cases, surprising results.

In the next two subsections, we give a "big picture" outline of this thesis in the framework of above mentioned two techniques. In each of these subsections, we present our contributions using a high-level overview. We go into more depth on each of these contributions in the main chapters of our thesis.

1.1 Well-structured transition systems

Well-structured transition systems (WSTS) are transition systems equipped with a well-quasi-order (wqo) over its space of configurations. Intuitively, a well-quasi-order $\preceq$ over a set S is an order on the elements of S such that it is impossible to have an "infinite descent" in the set S with respect to $\preceq$. For example, the set of natural numbers with the usual ordering is a well-quasi-order because if we start a sequence at some element i, then we can only go at most $i-1$ more steps before we encounter something which is at least i. On the other hand, the set of integers with the usual ordering is not a well-quasi-order because

$$-1, -2, -3, -4, \ldots$$

is an example of infinite descent.

A well-structured transition system is a system whose configuration space comes with an underlying well-quasi-order $\preceq$. Further, the transitions of the system respect this order, in the sense that, if a smaller configuration can execute some transition, then so can any larger configuration. This means that if a configuration c can move in one step to c' and $c \preceq d$, then d can move in one step to d' where $c' \preceq d'$. Hence, any transition from a smaller configuration can be "simulated" by a transition from a larger configuration, leading to similar results. This property is called the *compatibility* property. Any system that satisfies the compatibility property is a WSTS. It turns out that for WSTS, some verification problems become decidable under certain assumptions.

The main problem that is of interest to us regarding WSTS is the *coverability* problem described as follows: Given a WSTS $(S, \rightarrow)$ over a wqo $\preceq$ and two configurations s and t, is there a run of the system starting from s which reaches some t' with $t \preceq t'$. Intuitively, in this case, the set of configurations that are bigger than or equal to t is a set of *error configurations* and we want to check that such configurations are never reached from the starting configuration s. In the context of parameterized systems, this problem is particularly interesting because it can be used to phrase the question of whether some agent of the given parameterized system can ever reach an error state.

Under some assumptions on the underlying class of WSTS, it turns out that the coverability problem is decidable [1, 73]. The algorithm for the coverability problem essentially starts with the set $U_0 := \{t' : t \preceq t'\}$ and then iteratively constructs a sequence of sets $U_0, U_1, \dots$, such that $U_{i+1} := U_i \cup pre(U_i)$, where $pre(U_i)$ is the set of all configurations which can reach some configuration in U_i in one step. Using the compatibility property, one can show that this computation will reach a fix-point U_m. Once that happens, it is sufficient to check if $s \in U_m$ in order to answer the coverability problem.

The framework of WSTS has proved to be a powerful tool for proving decidability results [73, 1]. Consequently, it is an important tool in the toolkit of techniques for verifying parameterized systems (Chapter 3 of [35]). The analysis of parameterized systems from the lens of WSTS has utilized a wide range of wqos. For instance, existing decidability results in the literature on parameterized verification using the theory of WSTS employ wqos over vectors [63], sets [4], multisets [3], trees [94] and even graphs [54].

In the first part of this dissertation, we use the framework of well-structured transition systems to derive results on the safety verification of some classes of parameterized systems. Our contributions in this part can be categorized into two sub-parts, which we now briefly explain in the next two subsections.

1.1.1 Contribution I: Upper bounds for coverability

While the argument given in the previous part proves that the algorithm for deciding coverability always terminates, it does not give a bound on the running time of the coverability algorithm. For many classes of WSTS, the "predecessor" computations and deciding the underlying well-quasi-order relation can be done rather efficiently. Hence, the running time of the algorithm is primarily bounded by the number of steps taken until a fix-point is achieved. Under some assumptions, a bound on the running time taken to achieve saturation can be given in terms of the maximum length of *controlled bad sequences* of the underlying wqo.

The central idea behind controlled bad sequences is that we first assign a measure to each element of the wqo, formalized by a norm function $|\cdot|$. Then, a controlled bad sequence of a wqo is a sequence $s_0, s_1, \dots$ such that for each i, s_i is not bigger than s_j for any $j > i$ and the norm of s_i grows with respect to i, g and n for some function g called the control function and some number n called the initial norm. For many classes of WSTS, we can bound the number of steps until the coverability algorithm reaches a fix-point by the length of the longest controlled bad sequence for some control function g and some initial norm based on the input. Hence, if we define the *length function* of a wqo $(A, \preceq)$ with norm $|\cdot|$ and a control function g as

$L_{A,g}(n)$ is the length of a longest (g, n)-controlled bad sequence over $(A, \preceq, |\cdot|)$

then upper bounds on $L_{A,g}$ translate to upper bounds on the running time of the coverability algorithm. This motivates the study of the following class of so-called *length function questions*.

Given a wqo $(A, \preceq)$, a norm $|\cdot|$ and a function g, provide upper bounds for the length function $L_{A,g}$.

As mentioned in the previous part, the analysis of parameterized systems requires a variety of wqos. Any upper bounds on the length function proved for any one of these wqos can then be translated to any parameterized system which uses that wqo.

With this motivation, in Chapter 4, we consider three different families of wqos and prove upper bounds for length functions over them. The first two families of wqos are orders over finite subsets of $\mathbb{N}^d$, respectively called the majoring and the minoring ordering. The last family is about the induced subgraph ordering over graphs. We also use these results on the length functions to provide upper bounds on the coverability problem for some classes of parameterized systems, thereby illustrating the applicability of our results in the context of parameterized verification.

1.1.2 Contribution II: Lower bounds for coverability

As a next natural question, we can ask if the upper bounds obtained for the models considered in Chapter 4 are optimal, that is

Are there lower bounds for these models that match the upper bounds obtained from the length functions?

In Chapter 5, we answer this question in the affirmative for two classes of models. We do this by giving polynomial-time reductions from known hard problems to the coverability problem for both these classes. Our lower bounds complement the upper bounds provided in Chapter 5, thereby leading to *completeness* results for both these classes.

In summary, the results in Chapters 4 and 5 provide complexity-theoretic upper bounds and completeness results for safety verification of classes of parameterized systems by utilizing the tools and framework of well-structured transition systems.

1.2 Linear arithmetic theories

A linear arithmetic theory is any logical theory about number systems equipped with an order relation $<$ and an addition function $+$. Common examples include the theory of addition over natural numbers $\mathbb{N}$ (also called Presburger arithmetic), the integers $\mathbb{Z}$ and the rational numbers $\mathbb{Q}$. Linear arithmetic theories are important in the context of verification of parameterized systems. For instance, results from the linear arithmetic theory of the rationals are used for deriving algorithms for verifying specifications of rendez-vous protocols [76] and also timed networks [8]. Further, in some cases, linear arithmetic theories can serve as *over-approximations* of the reachability relation of the given parameterized system. Given a parameterized system, one can abstract away certain features of the system and keep track of only the number of agents satisfying a certain property. Then we can apply linear arithmetic theories to reason about the counter system and derive sufficient conditions for the safety of the original system. This approach is mentioned in Chapter 9 of [35]. Linear arithmetic theories can also be used to derive incomplete algorithms for parameterized verification [113, 36].

In the second part of this thesis, we apply various results about linear arithmetic theories to derive complete and efficient algorithms for parameterized systems. Our contributions in this part can once again be categorized into two sub-parts, which we now briefly explain in the next two subsections.

1.2.1 Contribution III: The cut-off problem

For our first contribution using linear arithmetic theories, we consider the model of *rendez-vous protocols*. These are systems in which communication between agents happens by means of a rendez-vous. At any point in time, two agents meet, exchange messages and update their states according to a specified transition relation.

Various papers on parameterized verification for rendez-vous protocols have established the decidability and complexity landscape for many specifications, i.e., checking whether every initial configuration satisfies a given property (for e.g., see [76]). A natural relaxation of this type of question is asking if infinitely many initial configurations satisfy a given property or if all but finitely many initial configurations satisfy a given property. Affirmative answers to questions of the second type imply the existence of *cut-offs* in the underlying protocol; cut-offs state the existence of an initial configuration such that a property holds for all larger initial configurations.

In this line of work, Horn and Sangnier studied the *cut-off problem* for rendez-vous protocols [81]. The cut-off problem takes as input a rendez-vous protocol, and asks if there exists a number B such that for all $n \geq B$, the initial configuration of the protocol with n agents in an initial state can reach the final configuration with all of the n agents in a final state. Such a number B is called a *cut-off* for the system. Horn and Sangnier showed that the problem is in $\mathsf{EXPSPACE}$, but did not provide any lower bound for the problem. This leads to the following question.

What is the precise complexity of the cut-off problem for rendez-vous protocols?

Our contribution to the theory of rendez-vous protocols is to show that this problem is actually P-complete, which significantly improves upon the existing bound of EXPSPACE. We also show better bounds for some other special cases considered in [81]. All of our results are obtained by using techniques from various linear arithmetic theories.

1.2.2 Contribution IV: Verification of threshold automata

For our second contribution using linear arithmetic theories, we consider another class of parameterized systems called *threshold automata* [89]. These systems can be used to analyze fault-tolerant distributed algorithms, i.e., distributed algorithms which must work even in the presence of some faulty agents [88]. Extensions of thresold automata have also been used to model and verify randomized distributed algorithms and blockchain protocols [33, 32].

Despite the interest that this formalism and its extensions have received, results on the computational complexity of the main verification problems for threshold automata have been missing from the literature. This leads to the following question concerning the analysis of threshold automata.

What is the complexity of verifying threshold automata?

We address this question and analyze the complexity of verifying threshold automata. By using results from Presburger arithmetic, we prove NP-completeness results for some verification problems concerning threshold automata. Moreover, we also analyze some of these problems from the perspective of *parameterized complexity*, which allows for a more fine-grained analysis of the computational complexity of problems by studying them in terms of the various parameters of the given input. Finally, we also present an implementation of our algorithms and test it against a set of standard benchmarks from the literature.

In summary, we have used a variety of results on linear arithmetic theories to provide sound, complete, efficient and complexity-theoretic optimal algorithms for classes of parameterized systems.

1.3 Outline and Publications

This thesis is divided into two parts; the first part is focused on well-structured transition systems and the second part is focused on linear arithmetic theories.

Both these parts begin with a chapter that introduces the necessary background and recalls the essential results and techniques for our purposes. Every other chapter in each of these parts describes one of our four main contributions. Each chapter begins with a short introduction giving a short motivation for the problems that we consider, and ends with a section on related work and a short summary. In between these sections, we present the models that we consider along with our contributions. The proofs of all our results are only presented at a high level, focusing only on intuition. For formal details, we refer the reader to the papers attached in the appendix.

Outline.

Chapter 2 introduces basic notations and definitions that we will use in this thesis. After that, the thesis is divided into two parts.

- Part I contains the following chapters.
  - Chapter 3 introduces Part I on well-structured transition systems and sets up the basic definitions regarding controlled bad sequences and their relation to the coverability problem.
  - Chapter 4 covers Contribution I (Subsection 1.1.1) and describes our results on upper bounds for length functions and their applications to parameterized systems. The results of this chapter are based on [13, 15, 18].
  - Chapter 5 covers Contribution II (Subsection 1.1.2) and describes our results on lower bounds for the coverability problem for parameterized systems. The results of this chapter are based on [15, 18].
- Part II contains the following chapters.
  - Chapter 6 introduces Part II on linear arithmetic theories and recalls the results that we will use.
  - Chapter 7 covers Contribution III (Subsection 1.2.1) and describes our results on the cut-off problem for rendez-vous protocols. The results of this chapter are based on [21].
  - Chapter 8 covers the first part of Contribution IV (Subsection 1.2.2) and describes our results on the complexity of verification of threshold automata. The results of this chapter are based on [20].
  - Chapter 9 covers the second part of Contribution IV and describes our results on the parameterized complexity of threshold automata. The results of this chapter are based on [14].
- Chapter 10 concludes with a summary of the thesis.

Publications.

The following is the list of my entire publications during my Ph.D. studies at TUM.

Core publications included in the thesis. The following is the list of core publications included in the thesis. All of these papers are included in the Appendix.

- *Complexity of Coverability in Bounded Path Broadcast Networks.* A. R. Balasubramanian. In conference proceedings of FSTTCS 2021 [15]. This paper is reprinted in Appendix B.
- *Complexity of Coverability in Depth-Bounded Processes.* A. R. Balasubramanian. In conference proceedings of CONCUR 2022 [18]. This paper is reprinted in Appendix C.
- *Complexity of Verification and Synthesis of Threshold Automata.* A. R. Balasubramanian, J. Esparza, M. Lazić. In conference proceedings of ATVA 2020 [20]. This paper is reprinted in Appendix E.
- *Parameterized Complexity of Safety of Threshold Automata.* A. R. Balasubramanian. In conference proceedings of FSTTCS 2020 [14]. This paper is reprinted in Appendix F.

Non-core publications included in the thesis. The following is the list of non-core publications included in the thesis. All of these papers are included in the Appendix.

- *Complexity of Controlled Bad Sequences over Finite Sets of* $\mathbb{N}^d$*.* A. R. Balasubramanian. In conference proceedings of LICS 2020 [13]. This paper is reprinted in Appendix A.
- *Finding Cut-Offs in Leaderless Rendez-Vous Protocols is Easy.* A. R. Balasubramanian, J. Esparza, M. Raskin. In conference proceedings of FoSSaCS 2021. [21]. This paper is reprinted in Appendix D.

A part of the work of the paper [13] was done before I became a Ph.D. student at TUM. However, I continued to work on the results of this paper, refining it, submitting it to the LICS 2020 conference and publishing it after I became a Ph.D. student at TUM. For this reason, the paper [13] is listed as a non-core publication.

Other publications. The following is a list of my other publications during my Ph.D. studies at TUM. These papers are not included in the thesis and they are listed here for the sake of completeness.

- *Characterizing Consensus in the Heard-Of Model.* A. R. Balasubramanian, I. Walukiewicz. In conference proceedings of CONCUR 2020 [25].
- *Decidability and Complexity in Weakening and Contraction Hypersequent Substructural Logics.* A. R. Balasubramanian, T. Lang, R. Ramanayake. In conference proceedings of LICS 2021 [23].

- *Adaptive Synchronisation of Pushdown Automata.* A. R. Balasubramanian, K. S. Thejaswini. In conference proceedings of CONCUR 2021 [24].
- *Reconfigurable Broadcast Networks and Asynchronous Shared-Memory Systems are Equivalent.* A. R. Balasubramanian, C. Weil-Kennedy. In conference proceedings of GandALF 2021 [26].
- *Parameterized Verification of Coverability in Infinite State Broadcast Networks.* A. R. Balasubramanian. In Information and Computation journal, 2021 [16].
- *Parameterized Analysis of Reconfigurable Broadcast Networks.* A. R. Balasubramanian, L. Guillou, C. Weil-Kennedy. In conference proceedings of FoSSaCS 2022 [22].
- *Coefficient Synthesis for Threshold Automata.* A. R. Balasubramanian. In conference proceedings of RP 2022 [17].

Chapter 2

Preliminaries

In this chapter, we recall and set up basic definitions, notations and results that we will use throughout the thesis. This includes topics such as multisets, well-quasi-orders, and fast-growing hierarchies.

2.1 Basic notations

Graphs, ordinals, numbers, functions and sequences

We assume basic familiarity with graphs and ordinals. Usually, we will work with labeled graphs, which are graphs along with a labeling function that maps each vertex to some element in a given set. If the image of the labeling function is a set A, then the graph is called an A-labeled graph. To denote ordinals, we will always use Greek letters $\alpha, \beta, \ldots$.

Throughout the dissertation, we shall use $\mathbb{N}, \mathbb{N}_{>0}, \mathbb{Z}, \mathbb{Q}_{\geq 0}$ and $\mathbb{Q}$ to denote the set of non-negative integers, positive integers, integers, non-negative rational numbers and rational numbers, respectively. Unless specified otherwise, we will use $+, -, \cdot$ to denote the usual addition, subtraction and multiplication operations and $\leq$ to denote the usual order relation on rational numbers.

A function $g : \mathbb{N} \to \mathbb{N}$ is increasing (resp. strictly increasing) if whenever $x < y$ we have $g(x) \leq g(y)$ (resp. $g(x) < g(y)$). It is *inflationary* if $g(x) \geq x$.

Transition systems

A *transition system* is a tuple $(S, \to)$ where S is the set of *configurations* and $\to \subseteq S \times S$ is the *transition relation*. If A is some set, an *A-labeled transition system* (or simply labeled transition system) is a tuple $(S, \to, L)$ such that $(S, \to)$ is a transition system and $L : S \to A$ is the *labeling function*.

If $\to$ is some relation over a set S, we will often use $\xrightarrow{*}$ to denote its reflexive and transitive closure. Often, we will employ the notation $s \to s'$ (resp. $s \xrightarrow{*} s'$) to mean that $(s, s') \in \to$ (resp. $(s, s') \in \xrightarrow{*}$); we will extensively use this notation for transition relations of a transition system. If the underlying relation $\rightarrow$ is unambiguous, we will often say that s can reach t if $s \xrightarrow{*} t$.

Vectors, multisets and matrices

Let E and S be some sets. A vector from E to S is a function $v : E \rightarrow S$. The set of all vectors from E to S will be denoted by S^E. If $v \in S^E$, then v is referred to as a vector over S. When $S = \mathbb{N}$, then each element of S^E will be alternately referred to as a discrete multiset of E (or just a multiset).

Suppose $S \in \{\mathbb{N}, \mathbb{Z}, \mathbb{Q}\}$ and E is some set. Given a vector $v \in S^E$ and an element $\alpha \in S$, we let $\alpha \cdot v$ (or αv) be the vector given by $(\alpha \cdot v)(e) = \alpha \cdot v(e)$ for all $e \in E$. The *support* of a vector v is the set $[\![v]\!] := \{e : v(e) \neq 0\}$. We sometimes use the notation $x \in v$ to denote that $x \in [\![v]\!]$.

Given two vectors $v, v' \in S^E$ (where $S \in \{\mathbb{N}, \mathbb{Z}, \mathbb{Q}\}$) we say that $v \leq v'$ if $v(e) \leq v'(e)$ for all $e \in E$ and we let $v + v'$ be the vector given by $(v + v')(e) = v(e) + v'(e)$ for all $e \in E$. Further, if $S \in \{\mathbb{Z}, \mathbb{Q}\}$, then we define $v - v'$ as the vector given by $(v - v')(e) = v(e) - v'(e)$ for all e. On the other hand, if v and v' are multisets (i.e., vectors over $\mathbb{N}$) *such that* $v' \leq v$, then we define $(v - v')(e) = v(e) - v'(e)$ for all e.

The vector which maps every element of E to 0 (resp. 1) is denoted by $\mathbf{0}$ (resp. $\mathbf{1}$). A finite multiset is a multiset $v : E \rightarrow \mathbb{N}$ such that $[\![v]\!]$ is finite. We use $\mathbb{M}_{\mathrm{f}}(E)$ to denote the set of all finite multisets of E. We sometimes denote multisets using a set-like notation, e.g. $\wr a, 2 \cdot b, c \int$ denotes the multiset given by $M(a) = 1, M(b) = 2, M(c) = 1$ and $M(e) = 0$ for all $e \notin \{a, b, c\}$.

A matrix is a function $F : A \times B \rightarrow S$ for some sets A, B and S. We use the notation $F[a, b]$ to denote the value of F at (a, b). If $F : A \times B \rightarrow S$ is a matrix, then we say that F is a matrix over S whose rows are indexed by A and columns are indexed by B. If $F : A \times B \rightarrow \mathbb{Q}$ and $G : A \times B \rightarrow \mathbb{Q}$ are matrices, then $F + G$ and $F - G$ are matrices defined by pointwise addition and subtraction, i.e., $(F+G)[a, b] = F[a, b]+G[a, b]$ and $(F-G)[a, b] = F[a, b]-G[a, b]$ for every $a \in A$ and $b \in B$. Further, if $F : A \times B \rightarrow \mathbb{Q}$ is a matrix and $\mathbf{v} : B \rightarrow \mathbb{Q}$ is a vector, then the product vector of F and $\mathbf{v}$, denoted by $F \cdot \mathbf{v}$ or $F\mathbf{v}$, is defined as the vector $(F\mathbf{v})(a) = \sum_{b \in B} F[a, b] \cdot \mathbf{v}(b)$ for every $a \in A$.

Sequences

A sequence of a set A is a function $f : S \rightarrow A$ for some subset S which is either empty or is of the form $\{1, 2, \dots, k\}$ for some k. If S is empty, then f is called the empty sequence, and its length is set to 0. On the other hand, if $S = \{1, \dots, k\}$, then the length of f is set to k. If f is a non-empty sequence, then we often denote f as $a_1, a_2, \dots, a_k$ where k is the length of f and each $a_i = f(i)$. We let A^* denote the set of all sequences of A. Concatenation of sequences by the operator $\cdot$ is defined as usual, i.e., $\epsilon \cdot f = f \cdot \epsilon = f$ for any sequence f and $f \cdot g$ for any two non-empty sequences $f = a_1, \dots, a_k$ and $g = a'_1, \dots, a'_m$, is the sequence given by $a_1, \dots, a_k, a'_1, \dots, a'_m$. Sometimes we use the notation fg

to denote the concatenation of the sequences f and g. For a sequence f and a number n, we let f^n denote the concatenation of f with itself n times.

2.2 Well-quasi-orders

We now introduce the necessary definitions and notations for well-quasi-orders. Most of the presentation in this section and the next one is taken from [107, 106, 13].

A *quasi ordering* (qo) over a set A is a relation $\leq_A$ such that $\leq_A$ is reflexive and transitive. We write $x <_A y$ if $x \leq_A y$ and $y \not\leq_A x$. An element $x \in A$ is said to be *minimal* if there is no y such that $y \leq_A x$.

Definition 1. A *well-quasi-ordering* (wqo) over a set A is a qo $\leq_A$ such that for every infinite sequence $x_0, x_1, x_2, \ldots$, there exists $i < j$ such that $x_i \leq_A x_j$.

As is standard, whenever we have a wqo $\leq_A$ over a set A, we shall often abuse notation and call the pair $(A, \leq_A)$ a well-quasi-order. If there is no scope for confusion, we often drop the A subscript from $\leq_A$. If $(A, \leq)$ is a wqo and $S \subseteq A$, then the wqo induced by S is the wqo $(S, \leq_S)$ where $\leq_S$ is the restriction of $\leq$ to the set S.

Example 2. (*Some basic wqos*). The set of natural numbers $\mathbb{N}$ with the usual ordering $\leq$, i.e., $(\mathbb{N}, \leq)$ is a wqo. Throughout the dissertation, if we do not explicitly specify the underlying order for $\mathbb{N}$, then it is to be assumed that it is the usual ordering. Another wqo is any finite set $A := \{a_0, a_1, \ldots, a_{k-1}\}$ such that distinct elements are unordered, i.e., $a_i \leq_A a_j$ if and only if $a_i = a_j$. We shall denote this wqo by $(A, =)$.

Having seen basic examples of wqos, we shall now provide constructions to produce "complex" wqos from simpler ones.

Definition 3. (*Sums and products*). Let $(A_1, \leq_{A_1})$ and $(A_2, \leq_{A_2})$ be two wqos. The *disjoint sum* (or simply the sum) of $(A_1, \leq_{A_1})$ and $(A_2, \leq_{A_2})$ is the wqo $(A_1 + A_2, \leq_{A_1+A_2})$ where

$$A_1 + A_2 := \{(i, x) : i \in \{1, 2\} \text{ and } x \in A_i\}$$
$$(i, x) \leq_{A_1+A_2} (j, y) \quad \Leftrightarrow \quad i = j \text{ and } x \leq_{A_i} y$$

The *cartesian product* (or simply the product) of $(A_1, \leq_{A_1})$ and $(A_2, \leq_{A_2})$ is the wqo $(A_1 \times A_2, \leq_{A_1 \times A_2})$ where

$$A_1 \times A_2 := \{(x_1, x_2) : x_1 \in A_1, x_2 \in A_2\}$$
$$(x_1, x_2) \leq_{A_1 \times A_2} (y_1, y_2) \quad \Leftrightarrow \quad x_1 \leq_{A_1} y_1 \text{ and } x_2 \leq_{A_2} y_2$$

It can be easily verified that both $(A_1 + A_2, \leq_{A_1+A_2})$ and $(A_1 \times A_2, \leq_{A_1 \times A_2})$ are wqos, when $(A_1, \leq_{A_1})$ and $(A_2, \leq_{A_2})$ are. When the product operation is applied to a wqo $(A, \leq_A)$ with itself repeatedly for some d number of times, we denote the resulting wqo by $(A^d, \leq_{A^d})$.

Remark 4. The product $(\mathbb{N}^d, \leq_{\mathbb{N}^d})$ obtained from $(\mathbb{N}, \leq)$ will be of special interest to us. From now on, whenever we refer to the underlying order of $\mathbb{N}^d$, we will always mean this cartesian product ordering.

Basis, good and bad sequences

Let $(A, \leq_A)$ be some wqo. The definition of a wqo naturally lends itself to some definitions which we shall routinely use throughout this dissertation. We recall a few such notions here, beginning with the notion of an upward closed set.

Definition 5. For any subset $S \subseteq A$, we denote by $\uparrow S$ the set $\{y : \exists x \in S, x \leq_A y\}$. A set S is said to be *upward closed* if $S = \uparrow S$.

Intuitively, $\uparrow S$ is the set of all elements that are "bigger" than or equal to elements in S. S is upward closed if whenever an element x is present in S, so are all the elements "bigger" than x.

Upward-closed sets enjoy certain nice properties; in particular, it is possible to finitely represent them by means of a *basis*, which we now define.

Definition 6. Let S be some upward closed set of A. A set U is a *basis* for S if $\uparrow U = S$.

By definition of a wqo, we can immediately deduce that every upward closed set admits a *finite basis*. This particular property of upward closed sets is very useful, as it will allow us to effectively store and compare even infinite upward closed sets.

Another important notion concerning wqos is the notion of *bad sequences*. We define it along with its counterpart, the *good sequences*.

Definition 7. A sequence $x_0, x_1, \dots$ is called *good* if there exists $i < j$ such that $x_i \leq_A x_j$. A sequence that is not good is called *bad*.

By definition, every bad sequence of a wqo is necessarily finite. Bad sequences will prove to be useful in the analysis of algorithms in the first part of our thesis.

2.3 Complexity classes

We assume basic familiarity with Turing machines. For a function $f : \mathbb{N} \to \mathbb{N}$, we let $\mathrm{DTIME}(f)$ denote the class of decision problems which can be solved by a deterministic Turing machine whose running time is upper bounded by the function f. We assume that the reader is familiar with the usual complexity classes like P, NP, EXP, NEXP and NC [100]. A somewhat esoteric complexity class is $\mathsf{ELEMENTARY}$ which is defined as

$$\mathsf{ELEMENTARY} = \bigcup_{k \geq 1} \mathrm{DTIME}(\mathtt{exp}^k(n))$$

where $\texttt{exp}^1(n) = 2^n$ and $\texttt{exp}^{k+1}(n) = 2^{\texttt{exp}^k(n)}$. Intuitively, ELEMENTARY contains the set of all problems which can be solved by a deterministic Turing machine running in time which is a fixed tower of exponentials of the input size.

A substantial part of this thesis deals with complexity classes that are even beyond ELEMENTARY and are not very standard. For this reason, we introduce here a hierarchy of complexity classes, the so-called *fast-growing hierarchy* of complexity classes. We present here only the basic definitions and details behind these classes. The reader is referred to [106, 79] for more details on these complexity classes.

Ordinals

To introduce the fast-growing hierarchy, we first need to define basic operations on *ordinals* (see [105, 79]). For the purposes of this dissertation, we will mostly only deal with ordinals that can be syntactically denoted as terms in *Cantor Normal Form* (CNF) as follows: $\alpha = \omega^{\beta_1} + \omega^{\beta_2} + \cdots + \omega^{\beta_m}$ where $\beta_1, \ldots, \beta_m$ are ordinals such that $\alpha > \beta_1 \geq \beta_2 \geq \cdots \geq \beta_m$. Note that if $\beta_m = 0$, then α is of the form $\alpha = \alpha' + 1$ for some ordinal α'; in such cases, α is called a *successor ordinal.* On the other hand, if $\beta_m \neq 0$, then α is called a *limit ordinal.* We will usually use λ to denote limit ordinals.

For $c \in \mathbb{N}$, let $\omega^\beta \cdot c$ denote $\overbrace{\omega^\beta + \cdots + \omega^\beta}^{c \text{ times}}$. We sometimes write ordinals in a *strict form* as $\alpha = \omega^{\beta_1} \cdot c_1 + \omega^{\beta_2} \cdot c_2 + \cdots + \omega^{\beta_m} \cdot c_m$ where $\beta_1 > \beta_2 > \cdots > \beta_m$ and each c_i is strictly bigger than 0.

A *fundamental sequence* for a *limit ordinal* λ is a sequence $(\lambda(x))_{x<\omega}$ with supremum λ, which we fix to be,

$$(\gamma + \omega^{\beta+1})(x) := \gamma + \omega^\beta \cdot (x+1), \qquad (\gamma + \omega^\lambda)(x) := \gamma + \omega^{\lambda(x)}$$

Along with the set of terms in CNF, we will also sometimes consider the ordinal ϵ_0, which is the supremum of all the ordinals which can be denoted by terms in CNF. In particular, $\alpha < \epsilon_0$ for any α which is expressible in CNF. ϵ_0 will be treated as a limit ordinal with fundamental sequence defined by $\epsilon_0(0) = \omega$ and $\epsilon_0(x+1) = w^{\epsilon_0(x)}$, i.e., $\epsilon_0(x+1)$ is a tower of ω's of height $x+1$.

Subrecursive hierarchies

Given an increasing, inflationary function h, we now define three different hierarchies of functions based on h. These hierarchies will be useful later on to define fast-growing complexity classes.

Definition 8. Let $h : \mathbb{N} \to \mathbb{N}$ be an increasing, inflationary function. The *Hardy hierarchy* for the function h is given by $(h^\alpha)_\alpha$ where

$$h^0(x) := x, \qquad h^{\alpha+1}(x) := h^\alpha(h(x)) \qquad h^\lambda(x) := h^{\lambda(x)}(x)$$

The *Cichoń hierarchy* $(h_\alpha)_\alpha$ for the function h is defined as

$$h_0(x) := 0, \qquad h_{\alpha+1}(x) := 1 + h_\alpha(h(x)) \qquad h_\lambda(x) := h_{\lambda(x)}(x)$$

Finally, we have the *fast-growing hierarchy*, which is defined as follows:

$$f_{h,0}(x) = h(x), \qquad f_{h,\alpha+1}(x) = f_{h,\alpha}^{x+1}(x) \qquad f_{h,\lambda}(x) = f_{h,\lambda(x)}(x)$$

Here $f_{h,\alpha}^i$ denotes the i-fold composition of $f_{h,\alpha}$ with itself.

Example 9. Let $S : \mathbb{N} \to \mathbb{N}$ be the successor function $S(x) = x + 1$. Notice that $S^i(x) = x + i$ and $S_i(x) = i$ for any $i \in \mathbb{N}$. Moreover, $S^\omega(x) = S^{\omega(x)}(x) = S^{x+1}(x) = 2x + 1$ and $S_\omega(x) = S_{x+1}(x) = x + 1$.

Further, notice that $f_{S,0}(x) = x + 1, f_{S,1}(x) = f_{S,0}^{x+1}(x) = 2x + 1, f_{S,2}(x) = f_{S,1}^{x+1}(x) = 2^{x+1}x + 2^{x+1} - 1$. Also, $f_{S,3}(x) = f_{S,2}^{x+1}(x)$ grows faster than the $\texttt{exp}^k$ function for any fixed k and hence an algorithm with running time $f_{S,3}(x)$ does not have elementary running time complexity.

Fast-growing complexity classes

Let $S : \mathbb{N} \to \mathbb{N}$ be the standard successor function. Let $\{S^\alpha\}, \{S_\alpha\}$ and $\{F_\alpha\}$ denote respectively the Hardy, Cichoń and fast-growing hierarchies for the successor function. Using these hierarchies, we define the *extended Grzegorczyk hierarchy* of fast-growing function classes $\{\mathscr{F}_\alpha\}_{\alpha<\epsilon_0}$ as follows:

$$\mathscr{F}_\alpha := \bigcup_{c<\omega} \text{FDTIME}(F_\alpha^c(n))$$

Here FDTIME$(F_\alpha^c(n))$ denotes the set of all functions $f : \mathbb{N}^d \to \mathbb{N}$ for some d that can be computed by a deterministic Turing machine in time $F_\alpha^c(n)$ where F_α^c denotes the c-fold composition of F_α with itself. Note that these are complexity classes for *functions*. We now define a similar hierarchy called the *fast-growing complexity classes* $\{\mathbf{F}_\alpha\}_\alpha$ for *decision problems*.

For any ordinal α, let $\mathscr{F}_{<\alpha} := \bigcup_{\beta<\alpha} \mathscr{F}_\beta$. We now define $\mathbf{F}_\alpha$ as

$$\mathbf{F}_\alpha := \bigcup_{p\in\mathscr{F}_{<\alpha}} \text{DTIME}(F_\alpha(p(n)))$$

Note that in contrast to $\mathscr{F}_\alpha$, the definition of $\mathbf{F}_\alpha$ allows for only one application of the function F_α (composed with a "lower" function p). One milestone along this hierarchy is the class $\mathbf{F}_3 :=$ TOWER, which already contains non-elementary problems.

The collection $\bigcup_{k<\omega} \mathscr{F}_k$ and $\bigcup_{k<\omega} \mathbf{F}_k$ will be referred to as the class of primitive recursive functions and primitive recursive problems, respectively.

Relativized fast-growing hierarchies

Let $h : \mathbb{N} \to \mathbb{N}$ be a strictly increasing, inflationary function. We can define a *relativized* hierarchy of fast-growing complexity classes $\{\mathbf{F}_{h,\alpha}\}_\alpha$ with respect to the function h as follows

$$\mathbf{F}_{h,\alpha} := \bigcup_{p \in \mathscr{F}_{<\alpha}} \mathrm{DTIME}(f_{h,\alpha}(p(n)))$$

Note that if h is the successor function, then we get the usual fast-growing complexity classes. For most of our applications, it will be convenient to work with this relativized hierarchy rather than the original hierarchy. However, as the following theorem indicates, for almost all practical purposes, these two hierarchies coincide.

Theorem 10. ([106, Theorem 4.2, Corollary 4.3]). If h is a strictly increasing, inflationary, primitive recursive function and $\alpha \geq \omega$, then $\mathbf{F}_{h,\alpha} = \mathbf{F}_{\alpha}$.

Part I

Well-Structured Transition Systems

Chapter 3

Introduction and Background

The first part of this thesis is regarding the applications of the theory of well-structured transition systems (WSTS) to parameterized verification. We show that by means of the theory of well-quasi-orders and WSTS, we can obtain complexity results for verification problems concerning parameterized systems.

We use this chapter to set up the necessary notations and definitions and provide a background of the existing results regarding WSTS. In the next two chapters, we explain our contributions to parameterized verification using the framework of WSTS.

3.1 Well-structured transition systems

A WSTS is a transition system whose configurations are equipped with a wqo $\preceq$. Further, the transitions of a WSTS respect the underlying wqo, in the sense that, if a transition is enabled at some configuration s, then it is also enabled at any configuration s' such that $s \preceq s'$. We now proceed to define this formally by roughly following the notions from [107].

Definition 11. A well-structured transition system (WSTS) is a tuple $\mathcal{S} = (S, \rightarrow, \preceq)$ where S is a set of *configurations*, $\rightarrow \subseteq S \times S$ is a set of *transitions* and $\preceq$ is a well-quasi-order on the set S satisfying the following *compatibility* property

$$\forall s_1, s_2, t_1 \in S, \text{ If } s_1 \rightarrow s_2 \text{ and } s_1 \preceq t_1, \text{ then } \exists t_2 \in S \text{ such that } t_1 \rightarrow t_2 \text{ and } s_2 \preceq t_2$$

Intuitively, the compatibility property states that if s_1 can reach s_2 by a transition and t_1 is any configuration bigger than s_1 (with respect to $\preceq$) then, t_1 can, in a step, reach a configuration t_2 which is bigger than s_2.

It is known that a variety of systems fall under the scope of WSTS [73, 1]. As an example, here we consider *lossy counter machines*, whose presentation and associated results are taken from [107].

Example 12. (*Lossy counter machines*). A *d-lossy counter machine* (d-LCM or LCM) is a tuple $\mathcal{M} = (Q, \delta)$ where Q is a finite set of *states* and $\delta \subseteq Q \times \{1, \ldots, d\} \times \{= 0?, \texttt{++}, \texttt{--}\} \times Q$ is a finite set of *rules*.

Intuitively, a d-LCM has access to d counters, each of which can hold a non-negative integer. The rules of an LCM allow it to move from one state to another and also allow it to increment, decrement or zero-test the value of a counter. Further, at any given point in time, the values of the counters can non-deterministically decrease, hence giving it the name *lossy counter machines*. We now formalize these notions.

A configuration of an LCM is an element of $Q \times \mathbb{N}^d$, denoted as tuples $(q, \mathbf{v})$ with $q \in Q$ and $\mathbf{v} \in \mathbb{N}^d$. For a rule $r = (q, i, t, q') \in \delta$ and configurations $C = (q, \mathbf{v})$ and $C' = (q', \mathbf{v}')$ we say that C can reach C' by firing r (denoted by $C \xrightarrow{r} C'$) if $\mathbf{v}'(j) = \mathbf{v}(j)$ for all $j \neq i$ and the following conditions are satisfied.

- If $t = \texttt{++}$, then $\mathbf{v}'(i) = \mathbf{v}(i) + 1$.
- If $t = \texttt{--}$, then $\mathbf{v}(i) > 0$ and $\mathbf{v}'(i) = \mathbf{v}(i) - 1$.
- If $t = \, = 0?$, then $\mathbf{v}'(i) = \mathbf{v}(i) = 0$.

We say that $C \rightarrow C'$ if $C \xrightarrow{r} C'$ for some rule r. Note that the transition relation $\rightarrow$ does not allow for "lossy" behaviors of the counters. To take care of this, we define a new transition relation $\rightarrow_\ell$ as $(q, \mathbf{v}) \rightarrow_\ell (q', \mathbf{v}')$ if and only if there exists $\mathbf{w} \leq_{\mathbb{N}^d} \mathbf{v}$ and $\mathbf{v}' \leq_{\mathbb{N}^d} \mathbf{w}'$ such that $(q, \mathbf{w}) \rightarrow (q', \mathbf{w}')$. Intuitively, we first allow the counter values to go down from $\mathbf{v}$ to $\mathbf{w}$, then we fire some rule to reach the counter value $\mathbf{w}'$, and then we again allow the counter values to go down from $\mathbf{w}'$ to $\mathbf{v}'$. It can now be verified that $\rightarrow_\ell$ is compatible with the product ordering obtained from $(Q, =_Q)$ and $(\mathbb{N}^d, \leq_{\mathbb{N}^d})$ and so lossy counter machines are well-structured.

3.2 The coverability problem

Despite its broad definition and scope, there are interesting problems that are solvable for WSTS under certain assumptions on the underlying configuration space and wqo. In this thesis, we will be concerned with the *coverability* problem for WSTS, which we define as follows.

Definition 13. The coverability problem for WSTS is defined as the following decision problem:

Input:	A WSTS $\mathcal{S} = (S, \rightarrow, \preceq)$ and two configurations s, t
Decide:	If t can be *covered* from s in $\mathcal{S}$, i.e., if there is a configuration t' such that $t \preceq t'$ and $s \xrightarrow{*} t'$

The coverability problem intuitively corresponds to verifying a *safety* property for the underlying WSTS. Indeed, if any configuration that is bigger than or equal to t is an *error* configuration of $\mathcal{S}$, then the coverability problem asks if we can reach some error configuration starting from s.

It is known that the coverability problem is decidable when certain assumptions are met by the underlying order and WSTS. To state these assumptions formally, we first set up some basic notation. Let $\mathcal{S} = (S, \rightarrow, \preceq)$ be a WSTS. For any subset of configurations U, define $pre(U) := \{t \in S : \exists s \in U, t \rightarrow s\}$. Intuitively, $pre(U)$ is the set of configurations that can reach some configuration in U by a single transition. Note that by the compatibility property of $\mathcal{S}$, if U is an upward-closed set, then $pre(U)$ is also an upward-closed set and hence always has a finite basis.

Having set up this notation, we assume that the following properties are true regarding the WSTS $\mathcal{S} = (S, \rightarrow, \preceq)$, in order to prove the decidability of the coverability problem. Namely, we assume that

- The set S and the order $\preceq$ are decidable and
- There is an algorithm that takes as input some configuration $s \in S$ and outputs a finite basis for the set $pre(\uparrow s)$.

These assumptions will henceforth be collectively referred to as the *effective computation* assumptions. From [73, 1], it is known that

Theorem 14. The coverability problem is decidable for well-structured transition systems which satisfy the effective computation assumptions.

Example 15. (*Coverability for lossy counter machines*). Let us consider the class of WSTS given by lossy counter machines. Let $\mathcal{M} = (Q, \delta)$ be a d-LCM. Note that the configuration space of $\mathcal{M}$ and its underlying order are decidable. Further, given a configuration $C = (q, \mathbf{v})$, we can compute a finite basis for $pre(\uparrow C)$ as follows. Let $\delta_C \in \delta$ be the set of rules such that $r = (q, i, t, q')$ for some i, t and q' and $\mathbf{v}(i) = 0$ if $t = {=}\,0?$. We can take the basis for $pre(\uparrow C)$ to be the minimal configurations of the set $\{C_r : r \in \delta_C\}$ where

- If $r = (q, i, t, q')$ with $t =$ ++, then $C_r = (q', \mathbf{v}')$ where $\mathbf{v}'(j) = \mathbf{v}(j)$ for all $j \neq i$ and $\mathbf{v}'(i) = \max(0, \mathbf{v}(i) - 1)$.
- If $r = (q, i, t, q')$ with $t =$ --, then $C_r = (q', \mathbf{v}')$ where $\mathbf{v}'(j) = \mathbf{v}(j)$ for all $j \neq i$ and $\mathbf{v}'(i) = \mathbf{v}(i) + 1$.
- If $r = (q, i, t, q')$ with $t = {=}\,0?$, then $C_r = (q', \mathbf{v}')$ where $\mathbf{v}'(j) = \mathbf{v}(j)$ for all $j \neq i$ and $\mathbf{v}'(i) = 0$.

By Theorem 14, it then follows that the coverability problem is decidable for LCMs.

The coverability algorithm

Let us now prove Theorem 14 by giving an algorithm for coverability. Let $\mathcal{S} = (S, \rightarrow, \preceq)$ be a WSTS satisfying the effective computation assumptions and let

s, t be two configurations. Suppose we want to decide if s can cover t. Consider the sequence of subsets $U_0 \subseteq U_1 \subseteq U_2 \dots$ defined as $U_0 := \uparrow \{t\}$, $U_{i+1} := \min(U_i \cup pre(U_i))$, where $\min(U_i \cup pre(U_i))$ is the set of minimal elements of $U_i \cup pre(U_i)$. By the compatibility property of $\mathcal{S}$, it follows that each U_i is an upward-closed set. We claim that this sequence eventually stabilizes, i.e., there exists an index m such that $\bigcup_{i \in \mathbb{N}} U_i = U_m$. For the sake of contradiction, suppose for every i, there exists $t_i \in U_i \setminus U_{i-1}$. Then, by definition of well-quasi-orders, there must be two indices $i < j$ such that $t_i \preceq t_j$. Since each U_i is upward-closed, this means that $t_j \in U_i$, which is a contradiction. It follows that the sequence eventually stabilizes to some U_m, and so t can be covered from s if and only if there exists $s' \in U_m$ such that $s' \preceq s$. Hence, if we can effectively store each set U_i, we can decide if s can cover t. The former can be accomplished by storing a finite basis for each U_i. Note that a basis for U_0 is simply $\{t\}$ and if we know a finite basis for U_i, then we can compute a finite basis for U_{i+1}, thanks to the effective computation assumptions. Hence, we have a decision procedure for coverability.

Note that from the sequence of sets $U_0 \subseteq U_1 \subseteq \cdots \subseteq U_m$, we can derive a sequence of elements $t_0, t_1, \dots, t_m$ as follows: We let t_0 be t and for any $1 \leq i \leq m$, we let t_i be *any minimal element* in the set $U_i \setminus U_{i-1}$. Any such sequence will be called a *pseudo-witness* for the coverability algorithm on the instance $(\mathcal{S}, s, t)$. Note that by construction, any pseudo-witness is a bad sequence (Definition 7). The notion of a pseudo-witness will be useful for us in the next section, which will talk about the *complexity* of the coverability algorithm.

3.3 Complexity of coverability

While the coverability algorithm allows us to prove the decidability of safety properties for WSTS, the arguments behind its proof do not immediately lend themselves to *complexity-theoretic upper bounds.* In this section, we shall show that under some assumptions on the underlying wqo and WSTS, we can prove upper bounds on the running time for the coverability algorithm. To this end, we define the notion of a normed wqo.

Normed wqos

A *normed qo* (or nqo) is a tuple $(A, \preceq, |\cdot|)$ where $(A, \preceq)$ is a qo and $|\cdot| : A \to \mathbb{N}$ is a function called the *norm* that satisfies the property that for every n, the set $\{a : |a| = n\}$ is finite. The norm function can be thought of as assigning a measure to every element of A, with the constraint that only finitely many elements are assigned the same measure. We let $A_{\leq n}$ denote the set of elements of A whose norm is at most n, i.e., $A_{\leq n} := \{a : |a| \leq n\}$. A *normed wqo* is a normed qo $(A, \preceq, |\cdot|)$ such that $(A, \preceq)$ is also a wqo.

Example 16. We can equip the wqo $(\mathbb{N}, \leq)$ with the identity function `id` as the norm to obtain a nwqo $(\mathbb{N}, \leq, \texttt{id})$. As another example, for any finite set A, we have a nwqo $(A, =, \mathbf{0})$ where $\mathbf{0}$ is the zero function that maps any element to 0.

Similar to sums and products of wqos (Definition 3), we can construct sums and products of nwqos, as illustrated in the following definition.

Definition 17. Suppose $(A_1, \leq_{A_1}, |\cdot|_{A_1})$ and $(A_2, \leq_{A_2}, |\cdot|_{A_2})$ are two nwqos. Their sum is the nwqo $(A_1 + A_2, \leq_{A_1+A_2}, |\cdot|_{A_1+A_2})$ where $(A_1 + A_2, \leq_{A_1+A_2})$ is the usual sum and $|(i, x)|_{A_1+A_2} = |x|_{A_i}$.

Similarly the product of $(A_1, \leq_{A_1}, |\cdot|_{A_1})$ and $(A_2, \leq_{A_2}, |\cdot|_{A_2})$ is the nwqo $(A_1 \times A_2, \leq_{A_1\times A_2}, |\cdot|_{A_1\times A_2})$ where $(A_1 \times A_2, \leq_{A_1\times A_2})$ is the usual product and $|(x_1, x_2)|_{A_1\times A_2} = \max(|x_1|_{A_1}, |x_2|_{A_2})$.

Remark 18. We note that by the above definition, the norm for the product nwqo $(\mathbb{N}^d, \leq_{\mathbb{N}^d}, |\cdot|_{\mathbb{N}^d})$ is simply the function that maps a vector $\mathbf{v}$ to the highest value in $\{\mathbf{v}(1), \mathbf{v}(2), \dots, \mathbf{v}(d)\}$. We shall call this norm the *max* norm. When there is no scope for confusion, we will often drop the $\mathbb{N}^d$ subscript in the notation for this nwqo.

The introduction of the norm function to wqos allows us to introduce a new type of sequence, different from the good and bad sequences as follows.

Definition 19. Let $n \in \mathbb{N}$ and let g be an increasing, inflationary function. A sequence of elements $x_0, x_1, \dots$ from an nwqo $(A, \preceq, |\cdot|)$ is said to be (g, n)-controlled if for every i, $|x_i| \leq g^i(n)$, where g^i is the i-fold composition of g with itself. The function g is called the *control function*, and the number n is called the *initial norm*.

It is known that for every control function g and every number n, there is always a longest (g, n)-controlled bad sequence (Section 3.1.3 of [107]). Hence, for every nwqo $(A, \preceq, |\cdot|)$ and every control function g, we can define a *length function* $L_{A,g}(n)$ as

$$L_{A,g}(n) := \text{ Length of a longest } (g, n)\text{-controlled bad sequence in } (A, \preceq, |\cdot|)$$

We will now use this notion of length functions to bound the running time of the coverability algorithm for certain classes of WSTS.

Normed WSTS

A *normed WSTS* is simply a WSTS $(S, \rightarrow, \preceq)$ equipped with a norm function $|\cdot|$ on the wqo $(S, \preceq)$. For an increasing, inflationary function $g : \mathbb{N} \rightarrow \mathbb{N}$, we say that the normed WSTS $(S, \rightarrow, \preceq, |\cdot|)$ is *g-controlled* if it satisfies the property that for any $s \in S$, the norm of the minimal elements in the set $\mathit{pre}(\uparrow s)$ is bounded by $g(|s|)$.

Example 20. Let us consider our running example, namely lossy counter machines. Let $\mathcal{M} = (Q, \delta)$ be a d-LCM. From the construction given in Example 15, it follows that if C' is any minimal configuration of $\mathit{pre}(\uparrow C)$ for some configuration C, then $|C'| \leq |C| + 1$. Hence, lossy counter machines are g-controlled where $g(n) = n + 1$.

Let us now assume that we have a class $\mathcal{C}$ of normed g-controlled WSTS satisfying all the effective computation assumptions where g is some increasing, inflationary, primitive recursive function. Assume that each instance $I = (\mathcal{S}, s, t)$ of the coverability problem in the class $\mathcal{C}$ of WSTS is encoded by some string $\texttt{input}_I$ whose length is $\texttt{len}_I$. Suppose there are strictly increasing, inflationary, primitive recursive functions h and N satisfying the following criteria, called the *running time assumptions*: For every input instance $I = (\mathcal{S}, s, t)$,

- Given a minimal basis B for an upward closed set U, we can compute a minimal basis for the set $pre(U) \cup U$ and check for membership in U in time $h(\texttt{len}_I, \texttt{len}_B)$ where $\texttt{len}_B$ is the size of the input representation of B.
- Given a number n, there are at most $N(\texttt{len}_I, n)$ elements of $\mathcal{S}$ whose norm is at most n.

Let $t_0, t_1, \ldots, t_m$ be any pseudo-witness of the coverability algorithm on the instance I. As mentioned in Subsection 3.2, $t_0, \ldots, t_m$ is a bad sequence. Moreover, since $\mathcal{S}$ is g-controlled, it follows that $t_0, \ldots, t_m$ is also a $(g, |t|)$-controlled sequence. Hence $m \leq L_{A,g}(|t|)$.

Now notice that the running time of the coverability algorithm is at most $m \cdot h(\texttt{len}_I, N(\texttt{len}_I, g^m(|t|)))$. This expression is a primitive recursive function in m and $\texttt{len}_I$ ([48, Definition 9], [106, Section 5.3.1]). Now, suppose $L_{A,g}(|t|) \leq f_{p,\alpha}(q(\texttt{len}_I))$ for some primitive recursive functions p, q and some ordinal $\alpha \geq \omega$. Then, by using existing results regarding the fast-growing hierarchy ([106, Lemma 4.6]) we can show that the running time of the algorithm is dominated by some function of the form $f_{t,\alpha}(w(\texttt{len}_I))$ where t is some primitive recursive function and w is some function in $\mathscr{F}_{<\alpha}$. By Theorem 10, it then follows that the coverability problem for this class $\mathcal{C}$ of WSTS is in the complexity class $\mathbf{F}_\alpha$.

Hence, if we find a way to bound the length functions of the underlying nwqo of a class of normed WSTS satisfying the running time assumptions, we can translate those to get running time bounds for the coverability algorithm. This strand of research has been quite successful and has resulted in a plethora of results for various classes of nwqos, such as the product ordering over $\mathbb{N}^d$, the subword ordering over words of a finite alphabet, the priority ordering over words of a finite priority alphabet, the multiset ordering over finite multisets of $\mathbb{N}^d$ and the linear ordering over ordinals, to name a few. (See [106, 105]).

Example 21. Note that the class of lossy counter machines satisfies the running time assumptions. By using length functions for the product ordering over tuples of natural numbers, it is possible to derive an $\mathbf{F}_\omega$ upper bound for the coverability problem for lossy counter machines [106, Section 6.1.2].

Our contributions

Having covered the necessary results that we shall use, we make a small remark on our contributions in the first part of the thesis.

The next two chapters cover our contributions I and II that were mentioned in Subsections 1.1.1 and 1.1.2, respectively. Chapter 4 contains results pertaining to upper bounds on length functions and their applications to parameterized systems. Chapter 5 contains lower bound results for some classes of parameterized systems, complementing the upper bounds obtained in Chapter 4. These results are discussed in depth in the respective chapters.

Chapter 4

Upper bounds for the coverability problem

We saw in the previous chapter that upper bounds on length functions for a normed wqo can be translated to upper bounds on the running time of the coverability algorithm for any WSTS over that normed wqo which satisfies some basic assumptions. As mentioned in Section 1.1, various parameterized systems can be analyzed under the lens of WSTS using a variety of nwqos. This diverse collection of nwqos suggests the need to compute bounds on length functions for different families of nwqos. Note that once such a bound has been computed for a family, it can be applied to any parameterized system which uses that family for its set of configurations.

With this motivation, in this chapter, we prove upper bounds on the length functions for three different families of nwqos. The first two are over finite sets of $\mathbb{N}^d$ for some d and the third family is over graphs. We also present applications of these results by providing upper bounds for the coverability problem for some classes of parameterized systems.

The rest of this chapter is structured as follows. In the first three sections, we introduce a nwqo in each section, state our main results regarding that nwqo and briefly sketch the idea behind our results. In the fourth section, we describe applications of our results to parameterized systems. Finally, we discuss related work and conclude with a short summary of our results.

The results of this chapter are taken from the papers [13, 15] and [18], except for Subsection 4.4.1, which is a new application of the results from [13]. These papers [13, 15] and [18] are reprinted in Appendices A, B and C, respectively.

We fix an increasing, inflationary and primitive recursive function $g : \mathbb{N} \to \mathbb{N}$ for the rest of this chapter.

4.1 The majoring ordering

The first nwqo that we will consider is the *majoring ordering*, which is a nwqo over *finite subsets* of tuples of natural numbers. We begin by formally defining the majoring ordering.

For any d, let us consider the nwqo $(\mathbb{N}^d, \leq, |\cdot|)$ where $\leq$ is the product ordering and $|\cdot|$ is the max norm as stated in Definition 17 and Remark 18. Let $\mathbb{P}_{\mathrm{f}}(\mathbb{N}^d)$ denote the set of all *finite subsets* of $\mathbb{N}^d$. We can now define the *majoring* ordering as follows.

Definition 22. The majoring nwqo over $\mathbb{P}_{\mathrm{f}}(\mathbb{N}^d)$ is given by the tuple $(\mathbb{P}_{\mathrm{f}}(\mathbb{N}^d), \sqsubseteq^{\mathtt{maj}}, |\cdot|_{\mathbb{P}_{\mathrm{f}}(\mathbb{N}^d)})$ where

$$\begin{aligned} X \sqsubseteq^{\mathtt{maj}} Y &\quad\Leftrightarrow\quad \forall x \in X,\ \exists y \in Y \text{ such that } x \leq y \\ |X|_{\mathbb{P}_{\mathrm{f}}(\mathbb{N}^d)} &\quad:=\quad \max(\{|x| : x \in X\} \cup \{\mathtt{card}(X)\}) \end{aligned}$$

Here max is the maximum function, and $\mathtt{card}(X)$ is the cardinality of X.

By Higman's lemma [80], it is known that the majoring ordering is indeed a nwqo. This nwqo has been used in the analysis of some WSTS in the literature; for instance, it has been used to prove decidability results for incrementing tree counter automata in [82].

Example 23. Let $X = \{(0,1),(7,3)\}$ and let $Y = \{(7,5),(10,0)\}$. Since $(0,1) \leq (7,5)$ and $(7,3) \leq (7,5)$, we have that $X \sqsubseteq^{\mathtt{maj}} Y$. We see that the norm of X is 7, and the norm of Y is 10.

On the other hand, if we let $Z = \{(6,5),(10,1)\}$ then $X \not\sqsubseteq^{\mathtt{maj}} Z$, since there is no element in Z which is at least as big as $(7,3)$.

Our contribution

Since the majoring ordering is a nwqo, length functions for it are well-defined. Our main result for the majoring nwqo is to give upper bounds for its length functions. We present this result in a general fashion so that it becomes applicable later on for parameterized systems.

Let $d, k \in \mathbb{N}$. By $(\mathbb{P}_{\mathrm{f}}(\mathbb{N}^d)^k, \sqsubseteq^{\mathtt{maj}}, |\cdot|_{\mathbb{P}_{\mathrm{f}}(\mathbb{N}^d)^k})$ we mean the nwqo obtained by taking the product of $(\mathbb{P}_{\mathrm{f}}(\mathbb{N}^d), \sqsubseteq^{\mathtt{maj}}, |\cdot|_{\mathbb{P}_{\mathrm{f}}(\mathbb{N}^d)})$ with itself k times. Recall that for a function h and an ordinal $\alpha \leq \epsilon_0$, the functions h^{α}, h_{α} and $f_{h,\alpha}$ are the functions at the α^{th} level of the Hardy, Cichoń and fast-growing hierarchies for the function h, respectively (See Definition 8). We now state our upper bounds for length functions over the majoring ordering.

Theorem 24. (*Majoring upper bounds*). Let $A := (\mathbb{P}_{\mathrm{f}}(\mathbb{N}^d)^k, \sqsubseteq^{\mathtt{maj}}, |\cdot|_{\mathbb{P}_{\mathrm{f}}(\mathbb{N}^d)^k})$ and let $\alpha = \omega^{\omega^{d-1}\cdot k}$. For $n > 0$, the length function $L_{A,g}$ satisfies

$$L_{A,g}(n) \leq h_{\alpha}(4dkn)$$

where $h(x) = 4x \cdot g(x)$.

By using this theorem and exploiting relationships between the three hierarchies, we can show that

Corollary 25. Let $A := (\mathbb{P}_{\mathrm{f}}(\mathbb{N}^d)^k, \sqsubseteq^{\mathtt{maj}}, |\cdot|_{\mathbb{P}_{\mathrm{f}}(\mathbb{N}^d)^k})$. For $n > 0$, we have

$$L_{A,g}(n) \leq f_{h,\omega^{d-1}\cdot k}(p(d,k,n))$$

where h and p are some primitive recursive functions over d, k and n.

Main ideas: Reflections, Descent equation and Derivatives

We now sketch the main ideas behind obtaining the upper bounds for the majoring nwqo. The concrete details can be found in Section 4 of [13]. The first main idea is the notion of *reflections*, which is a major tool to prove upper bounds for length functions.

Definition 26. Let $(A, \leq_A, |\cdot|_A)$ be a nqo and $(B, \leq_B, |\cdot|_B)$ be a nwqo. A *polynomial reflection* is a mapping $r : A \to B$ such that there exists a polynomial $q : \mathbb{N} \to \mathbb{N}$ satisfying

$$\forall x, y \in A : r(x) \leq_B r(y) \text{ implies } x \leq_A y$$

$$\forall x \in A : |r(x)|_B \ \leq \ q(|x|_A)$$

In such a case, we say that r is a polynomial reflection with polynomial q and denote it by $r : A \overset{q}{\hookrightarrow} B$.

The most important result about polynomial reflections is that once we have established a reflection from A to B, we can use existing upper bounds for B to prove upper bounds for A.

Theorem 27. (*Transfer theorem*). Let $r : A \overset{p}{\hookrightarrow} B$ be a polynomial reflection. Then there is a polynomial q such that

$$L_{A,g}(n) \leq L_{B,(q\circ g)}(q(n))$$

Apart from polynomial reflections, we need a few more tools to obtain the required upper bounds, which we now briefly explain, starting with the notion of residuals.

Definition 28. Let $(A, \leq_A, |\cdot|_A)$ be a nwqo and let $x \in A$. Let $A/x = \{y : x \not\leq_A y\}$. The *residual* of x with respect to A is the nwqo induced by the subset A/x.

Residuals give rise to the following notion of *descent equation*, which equates the length function for a nwqo in terms of the length function of its residuals.

Theorem 29. (*Descent equation*). Let $(A, \leq_A, |\cdot|_A)$ be a nwqo. We have

$$L_{A,g}(n) = \max_{x \in A_{\leq n}} \{1 + L_{A/x,g}(g(n))\}$$

Hence, a natural way to obtain bounds on the length function for the majoring ordering is to first obtain bounds on the length function of its residuals and then use the descent equation. The former can be obtained by once again taking residuals and then again applying the descent equation and so on and so forth till we reach "simple" nwqos, like $\mathbb{N}$ for which the length functions become trivial. The problem with this approach is that the residuals can become extremely complex as we unravel the descent equation. To overcome this, we use the framework established by Schmitz, Schnoebelen and others (For instance, see [71, 108]). The high-level idea behind this framework is that to each residual, we first associate an ordinal (called its order type) and then we also define a "derivative" operator for each order type. Then, by constructing reflections and using the Transfer theorem, we show that the residuals can be replaced with derivative operators of the corresponding order types in the descent equation. This allows us to use existing results regarding ordinals to derive the required upper bounds, which then lets us prove Theorem 24.

4.2 The minoring ordering

The second nwqo that is of interest to us is the minoring ordering, which is also defined over finite subsets of tuples of natural numbers. Intuitively, the minoring ordering is obtained by flipping the order of the sets X and Y in the definition of the majoring ordering. The formal definition of the minoring ordering is as follows.

Definition 30. The minoring nwqo over $\mathbb{P}_{\mathrm{f}}(\mathbb{N}^d)$ is given by the tuple $(\mathbb{P}_{\mathrm{f}}(\mathbb{N}^d), \sqsubseteq^{\mathtt{min}}, |\cdot|_{\mathbb{P}_{\mathrm{f}}(\mathbb{N}^d)})$ where $|\cdot|_{\mathbb{P}_{\mathrm{f}}(\mathbb{N}^d)}$ is the same as for the majoring nwqo and $\sqsubseteq^{\mathtt{min}}$ is defined as

$$X \sqsubseteq^{\mathtt{min}} Y \quad \Leftrightarrow \quad \forall y \in Y,\ \exists x \in X \text{ such that } x \leq y$$

It is known that the minoring ordering is indeed a nwqo [5, Theorems IV.3, IV.4]. The minoring ordering is useful in the analysis of some classes of automata with registers [72].

Example 31. Similar to Example 23, let us once again consider $X = \{(0,1),(7,3)\}$, $Y = \{(7,5),(10,0)\}$ and $Z = \{(6,5),(10,1)\}$. Note that there is no element in X which is smaller than or equal to $(10,0)$ and so $X \not\sqsubseteq^{\mathtt{min}} Y$. On the other hand, since $(0,1) \leq (6,5)$ and $(0,1) \leq (10,1)$, we have that $X \sqsubseteq^{\mathtt{min}} Z$. Hence, this example illustrates the difference between the majoring and the minoring ordering.

Our contribution

Similar to the majoring ordering, we now present the results for the minoring ordering in a general fashion. Let $d, k \in \mathbb{N}$. By $(\mathbb{P}_{\mathrm{f}}(\mathbb{N}^d)^k, \sqsubseteq^{\mathtt{min}}, |\cdot|_{\mathbb{P}_{\mathrm{f}}(\mathbb{N}^d)^k})$ we mean the nwqo obtained by taking the product of $(\mathbb{P}_{\mathrm{f}}(\mathbb{N}^d), \sqsubseteq^{\mathtt{min}}, |\cdot|_{\mathbb{P}_{\mathrm{f}}(\mathbb{N}^d)})$ with itself k times. The following theorem provides upper bounds for this nwqo.

Theorem 32. (*Minoring upper bounds*). Let $A := (\mathbb{P}_\mathrm{f}(\mathbb{N}^d)^k, \sqsubseteq^{\mathtt{min}}, |\cdot|_{\mathbb{P}_\mathrm{f}(\mathbb{N}^d)^k})$ and let $\alpha = \omega^{\omega^{d-1}\cdot(2^d k)}$. If we set $c = 4dk^2 2^d$ then for all $n \geq dk2^d$,

$$L_{A,g}(n) \leq h_\alpha(c \cdot g(n)^{2d})$$

where $h(x) = 4kx \cdot (g(x)+1)^d$.

The main idea behind this theorem is to give a polynomial reflection from the minoring ordering to a product of different majoring nwqos, for which we have already shown upper bounds. More details behind the proof of this theorem can be found in Section 6 of [13]. Similar to the majoring ordering, we can convert the bounds given in Theorem 32 in terms of the fast-growing hierarchy to obtain the following result.

Corollary 33. Let $A := (\mathbb{P}_\mathrm{f}(\mathbb{N}^d)^k, \sqsubseteq^{\mathtt{min}}, |\cdot|_{\mathbb{P}_\mathrm{f}(\mathbb{N}^d)^k})$. For $n > 0$, we have

$$L_{A,g}(n) \leq f_{h,\omega^{d-1}\cdot(2^d k)}(p(d,k,n))$$

where h and p are some primitive recursive functions over d, k and n.

4.3 The induced subgraph ordering

The final nwqo that is of interest to us is the *induced subgraph ordering*, which, as the title indicates, is over (labeled) graphs.

For any graph G, let its *vertex norm* be the number of vertices of G. Over this norm, we define the induced subgraph ordering as follows.

Definition 34. Let $G_1 = (V_1, E_1, L_1)$ and $G_2 = (V_2, E_2, L_2)$ be two S-labeled graphs for some finite set S. We say that G_1 is an induced subgraph of G_2, denoted by $G_1 \preceq_{in} G_2$, if there is an injection $h : V_1 \to V_2$ such that

- For every vertex $v \in V_1$, $L_1(v) = L_2(h(v))$.
- For every pair $(u,v) \in V_1 \times V_1$, $(u,v) \in E_1$ if and only if $(h(u), h(v)) \in E_2$.

Intuitively, if $G_1 \preceq_{in} G_2$ then G_2 contains a "copy" of G_1 as a subgraph. We stress that the induced subgraph ordering is not a wqo when the domain is the set of all graphs. However, there is an interesting class of graphs for which $\preceq_{in}$ turns out to be a wqo, which we now define.

Definition 35. Let G be a labeled graph. The *depth* of G is defined to be the length of the *longest simple path* in G. We say that G is *k-path bounded* or *k-depth bounded* if the depth of G is at most k.

For any $k \in \mathbb{N}$, it turns out that if we restrict ourselves to k-path bounded graphs, then the induced subgraph ordering over the vertex norm is a nwqo.

Theorem 36. ([58, Theorem 2.2]). Let $k \in \mathbb{N}$, S be some finite set and let $\mathcal{G}_k$ be the set of all S-labeled k-path bounded graphs. Then the induced subgraph ordering over $\mathcal{G}_k$ with the vertex norm is a nwqo.

Our contribution

Our contribution to the induced subgraph nwqo is to present upper bounds for its length functions in terms of the Hardy hierarchy (See Definition 8).

Theorem 37. (*Induced subgraph upper bounds*). Let $k \in \mathbb{N}$ and let S be some finite set whose size is d. Let A be the induced subgraph nwqo among all S-labeled k-path bounded graphs. Then, for all $n > 0$, we have

$$L_{A,g}(n) \leq h^{\epsilon_0}(p(k, d, n))$$

where h and p are some primitive recursive functions over k, d and n.

The main idea behind this result is to provide a polynomial reflection from this nwqo to another well-studied nwqo called *generalized priority alphabets*, for which upper bounds are already known [79, Proposition 4.1, Corollary 4.2]. The desired polynomial reflection roughly follows a similar mapping given in [79, Section 8.1.2] for bounded-depth trees. More details behind the proof of this theorem can be found in [15, Sections 6 and A.1].

4.4 Applications to parameterized systems

We now apply the results from the previous sections to obtain upper bounds on the time complexity of deciding safety properties for three classes of parameterized systems. The result for the first system is new and is not mentioned in any of the papers in the appendix. The results for the other two models can be found in [15, Section 6] and [18, Section 5].

4.4.1 Parameterized phaser programs

A (non-atomic) parameterized phaser program is a model of distributed computation in which synchronization between agents happens primarily by means of *phasers*. Intuitively, in this model, we have an arbitrary but finite number of agents, a set of shared Boolean variables and some number of phasers. Each agent is registered to a subset of phasers, and for each phaser that it is registered to, it has a signal value and a wait value. Apart from updating the global Boolean variables, agents are allowed to de-register from phasers, spawn more agents and issue signal and wait commands. A signal command allows an agent t to increment its signal value corresponding to some phaser p that it is registered to. Intuitively, this means that the agent has moved on to the next phase of its computation. On the other hand, a wait command to a phaser p allows an agent t to make a move provided that for every agent that is registered to p, the signal value of that agent with respect to p is strictly larger than the wait value of t with respect to p. Once an agent makes a wait transition, the wait value of t with respect to p is incremented by one. Intuitively, a wait transition for a phaser p acts as a synchronization barrier to the subset of agents that are registered to p and allows an agent t to pass only when all other agents are ahead of the phase dictated by the wait value of t.

The paper [75] mentions some possible applications of phasers, which we briefly outline here. Phasers have been implemented in variants of the Java programming language (See [75, Section 1]) and might be useful in applications that need dynamic load balancing. Further, it is also mentioned that phasers can implement many different synchronization barriers and so the generality of this construct makes it interesting from a theoretical perspective.

The authors of [75] prove that the coverability problem for phaser programs is decidable under some restrictions, among which one of them is that the model has only a fixed finite number of phasers. This result is proved by reducing this problem to a problem regarding constraints over the configurations of the phaser program, which is then solved by using tools from the theory of WSTS. To describe the underlying wqo that they use, we need some notation.

Given two multisets $X, Y \in \mathbb{M}_f(\mathbb{N}^d)$ for some d, we let $X \sqsubseteq Y$ if there is a surjection h from Y to X such that $h(y) \leq y$ for every $y \in Y$. Further, given $X \in \mathbb{M}_f(\mathbb{N}^d)$, we let $|X|_M := \{|x| : x \in \mathbb{N}^d\} \cup \{\texttt{card}(X)\}$. The underlying well-quasi order that the paper [75] uses for proving the decidability of the coverability problem can be reflected into a number of products of the wqo $\sqsubseteq$ with norm $|\cdot|_M$ over the domain $\mathbb{M}_f(\mathbb{N}^i)^j$ where i depends on the number of phasers and j depends on the input. Hence, it suffices to compute upper bounds for length functions over this nwqo.

To that end, consider two multisets $X, Y \in \mathbb{M}_f(\mathbb{N}^d)$ for some d. Notice that $X \sqsubseteq Y$ if and only if there is an injection f from X to Y such that $x \leq f(x)$ and $[\![X]\!] \sqsubseteq^{\texttt{min}} [\![Y]\!]$. The former condition is essentially the *multiset ordering* over $\mathbb{N}^d$ for which upper bounds are already known ([104, Theorem 1]). Further, we had already mentioned that the upper bounds for the minoring nwqo were obtained by giving a polynomial reflection from the minoring nwqo to a product of majoring nwqos. Since the majoring ordering can be easily reflected into the multiset ordering by the identity function, by using results about reflections ([13, Proposition 2.13]), we can show that the minoring ordering can be polynomially reflected into the multiset ordering. This then allows us to use results about the multiset ordering ([104, Theorem 1]) to prove that

Theorem 38. The coverability problem for phaser programs with a fixed number of phasers is in $\mathbf{F}_{\omega^\omega}$.

4.4.2 Bounded-path broadcast networks

The second class of parameterized systems that we consider is the formalism of broadcast networks. Intuitively, a broadcast network consists of a collection of finite-state, anonymous agents situated on the nodes of some graph called the communication topology. All of these agents execute the same underlying protocol. Initially, all of the agents start in one of the initial states of the given protocol. A transition of the protocol allows an agent to broadcast a message (from a finite alphabet) which is then received by *all of its neighbors* on the graph. The communication topology remains fixed throughout, i.e., the set of neighbors of an agent cannot change during an execution of the network.

The coverability problem for broadcast networks is to decide, given a protocol and a state q of the protocol, whether there is an execution of the network from some initial communication topology that results in some agent reaching the state q. Note that this is a parameterized verification problem, since we are parameterizing over the space of all graphs and therefore, also over the number of agents. It is known that this problem is undecidable ([54, Theorem 1]). However, when we only parameterize over the set of all k-path bounded graphs (for some constant k), then the broadcast network (comprising only labeled k-path bounded graphs) becomes a WSTS under the induced subgraph ordering. We call such networks bounded-path broadcast networks. A minor modification of the coverability algorithm (to work with infinitely many initial configurations) lets us show that the coverability problem for bounded-path broadcast networks is decidable. Once again, the running time of the algorithm depends primarily on length functions for the induced subgraph ordering. Hence, by using our result for bounded-depth graphs (Theorem 37) and some properties of the fast-growing hierarchy, we can show that

Theorem 39. The coverability problem for bounded-path broadcast networks is in $\mathbf{F}_{\epsilon_0}$.

In the next chapter, we will show that this bound is tight for bounded-depth broadcast networks.

4.4.3 Depth-bounded π-calculus processes

The π-calculus [95, 96] is a well-known formalism for describing concurrent message-passing systems that admit unbounded process creation and mobility of agents. Intuitively speaking, a configuration of such a system is a graph in which each vertex is a process labeled by its current state, and there is an edge between two processes if they share a channel. Due to its immense expressive power, all interesting problems quickly become undecidable for π-calculus processes.

Consequently, research on π-calculus has been focused on finding fragments for which certain problems are decidable. The most expressive fragment of π-calculus for which some verification problems still remain decidable is the class of depth-bounded processes [94]. Intuitively, depth-bounded processes are those in which the length of simple paths in the set of reachable configurations is bounded by a constant. It is known that depth-bounded processes can be viewed as WSTS [94]. This implies that the coverability problem for such systems is decidable [94, 114]. The underlying WSTS uses the induced subgraph ordering over the set of all K-path bounded trees for some suitable K depending on the depth of the (reachable) configuration space. Hence, we can use Theorem 37 to obtain upper bounds on the running time of the coverability problem. However, we note that upper bounds on length functions for K-path bounded *trees* were already known prior to our contribution (See [79, Section 8.1.2]), and so the following upper bound follows immediately from those results.

Theorem 40. The coverability problem for depth-bounded processes is in $\mathbf{F}_{\epsilon_0}$.

Hence, in this particular case, the results were already there in some sense; however, we are not aware of any published paper (apart from our own contribution [18]) which mentions this result. Furthermore, in the next chapter, we will prove that this upper bound is tight for depth-bounded processes, and so we have mainly included this result here for the sake of completeness.

4.4.4 Other applications

We briefly mention a few more applications of our results beyond the realm of parameterized verification. The results for the majoring and the minoring ordering have been used to show upper bounds for other classes of WSTS, such as tree counter automata with incrementing errors and two different models of register automata [13, Section 8]. Further, it has also been used in the context of hypersequent logics [23, Section 5] to provide upper bounds for substructural logics.

4.5 Related work

As mentioned in Section 3.3, a lot of work has been invested in computing upper bounds for length functions for different families of nwqos [71, 108, 105, 104, 79, 106]. Our work contributes to this line of research.

The majoring ordering is also sometimes known as the *Hoare* ordering in the powerdomain literature and has been considered for the analysis of some classes of WSTS [82]. To the best of our knowledge, the paper which explicitly considered computing upper bounds for the majoring ordering was [6] and then later the journal version [7]. Over $\mathbb{P}_\mathrm{f}(\mathbb{N}^d)$, the bound given in [7] is $f_{h,\omega^d}(p(d,n))$ for some primitive recursive functions h and p. Our result improves upon that upper bound by reducing the exponent in the ω term by 1.

The minoring ordering is also sometimes known as the *Smyth* ordering in the powerdomain literature and has been used in the analysis of data automata with registers [72]. The paper [5] proves some facts regarding the minoring ordering, but it ends with an open problem about the length of controlled bad sequences for the minoring ordering. Our upper bounds are the first bounds for length functions over the minoring ordering.

To the best of our knowledge, the paper [58] was the first paper that proved that bounded-depth graphs are well-quasi ordered under the induced subgraph ordering. Regarding upper bounds on length functions, as mentioned before, they were already known for the class of bounded-depth trees [79], which are a special class of bounded-depth graphs.

4.6 Conclusion

We have provided upper bounds for the length of controlled bad sequences over three different classes of nwqos. We have also used these results to bound the running time of the coverability algorithm for various classes of parameterized systems.

In this chapter, we have only provided upper bounds for length functions over classes of nwqos. One can also study the dual question and ask for lower bounds on length functions. Note that if we were to obtain a lower bound f for a length function over some nwqo A, then this means that the naive analysis of the running time of coverability for a WSTS $\mathcal{S}$ over A cannot be made shorter than the function f; (This however does not imply that the coverability problem for $\mathcal{S}$ cannot have an algorithm faster than the function f). We have some results on the lower bounds for length functions for the majoring and the minoring ordering in the paper [13]. For the majoring ordering, this bound is tight. However, we do not yet have a tight lower bound for the minoring ordering, which might be an interesting problem for future work.

Chapter 5

Lower bounds for the coverability problem

In the previous chapter, we provided upper bounds on the coverability problem for three classes of parameterized systems. The bounds that we had obtained grew faster than even primitive recursive functions. This leads to a natural question for these models: Are these bounds really the best that we can do?

In this chapter, we will show that this is indeed the case for two of the models considered in the previous chapter, namely bounded-path broadcast networks and depth-bounded (π-calculus) processes. For both these models, we will provide lower bounds for the coverability problem, which will match the upper bounds ($\mathbf{F}_{\epsilon_0}$) that we obtained in the previous chapter. The results that we obtain from these lower bounds solve open problems mentioned in [79, 114].

While the obtained lower bounds are negative from a tractability perspective, understanding the precise complexity of a particular problem is important because it may allow us to solve it in practice by reducing it to various other well-studied problems for which tools and heuristics have been developed. Our results also contribute to the program of Schmitz and Schnoebelen [109], whose focus is on populating the classes of the fast-growing hierarchy with more and more complete problems. The addition of these new complete problems allow future hardness results for WSTS to use these problems as intermediate problems rather than beginning from Turing machines or counter machines.

The rest of this chapter is structured as follows. We begin by introducing a known $\mathbf{F}_{\epsilon_0}$-hard problem, called the *coverability problem for nested counter systems*, from which we shall give reductions to both the coverability problem for bounded-path broadcast networks and the coverability problem for depth-bounded processes. Then, in the next two sections, we formally define both these models, state our results and give a sketch of the proofs. Then we discuss some related work and conclude with possible avenues for future work.

The results of this chapter are taken from the papers [15] and [18], which are reprinted in Appendices B and C, respectively.

5.1 Nested counter systems

Intuitively, a k-nested counter system (k-NCS) is a generalization of a usual counter system with *higher order counters*. A 1-dimensional counter is a normal counter which holds a natural number and which we can either increment or decrement by 1. A 2-dimensional counter is a counter which can add or subtract 1-dimensional counters, and a 3-dimensional counter can add or subtract 2-dimensional counters and so on. A k-NCS has access to a certain amount of k-dimensional counters, and it also has rules which allow it to manipulate these counters in a specific way. We now define this model in a formal manner, with slight alterations from the definition given in [51].

A k-nested counter system (k-NCS or simply NCS) is a tuple $\mathcal{N} = (Q, \delta)$ where Q is a finite set of *states* and $\delta \subseteq \bigcup_{1 \leq i,j \leq k+1}(Q^i \times Q^j)$ is a set of *rules*. The set $\mathcal{C}_{\mathcal{N}}$ of *configurations* of $\mathcal{N}$ is defined to be the set of all labeled rooted trees of height at most k, with labels from the set Q. When the underlying NCS is clear from context, we drop the $\mathcal{N}$ subscript in the notation for configurations.

The operational semantics of $\mathcal{N}$ is defined in terms of the following transition relation $\rightarrow \subseteq \mathcal{C} \times \mathcal{C}$ on configurations: Let $r := ((q_0, \dots, q_i), (q'_0, \dots, q'_j)) \in \delta$ be a rule with $i \leq j \leq k$. We say that a configuration C can move to the configuration C' using the rule r (denoted by $C \xrightarrow{r} C'$), if there is a path $v_0, v_1 \dots, v_i$ in C starting at the root such that for every $0 \leq l \leq i$, the label of v_l is q_l and, C' is obtained from C by 1) for every $0 \leq l \leq i$, changing the label of each v_l to q'_l and 2) for every $i+1 \leq l \leq j$, creating a new vertex v_l with label q'_l and adding it as a child to v_{l-1}.

Similarly, suppose $r := ((q_0, \dots, q_i), (q'_0, \dots, q'_j)) \in \delta$ is a rule with $j < i \leq k$. Then $C \xrightarrow{r} C'$ if there is a path $v_0, v_1, \dots, v_i$ in C starting at the root such that for every $0 \leq l \leq i$, the label of v_l is q_l and C' is obtained from C by 1) for every $0 \leq l \leq j$, changing the label of each v_l to q'_l and 2) removing the subtree rooted at the vertex v_{j+1}.

We use $C \rightarrow C'$ to denote that there is some rule r for which $C \xrightarrow{r} C'$ and we use $\xrightarrow{*}$ to denote the resulting reachability relation between two configurations.

Example 41. Let us consider the NCS $\mathcal{N}$ given by the states $Q = \{p_i, p'_i, q_i, q'_i : 0 \leq i \leq 4\}$ and consisting of the following rules: $r_1 = ((q_0, q_1), (q'_0, q'_1, q'_2))$, $r_2 = ((q'_0, q_3, q_2), (p_0))$, $r_3 = ((p_0), (p'_0))$. In Figure 5.1, we illustrate the application of these rules to a configuration of $\mathcal{N}$.

Let $\mathcal{N} = (Q, \delta)$ be some k-NCS and let $init, fin \in Q$. We say that the state *init* can cover the state *fin* if the (unique) configuration consisting of the single root vertex labeled by *init* (also called the initial configuration of $\mathcal{N}$) can reach *some* configuration where the root is labeled by *fin*. This notion of coverability leads to the following decision problem.

Definition 42. The coverability problem for NCS is defined as the following problem:

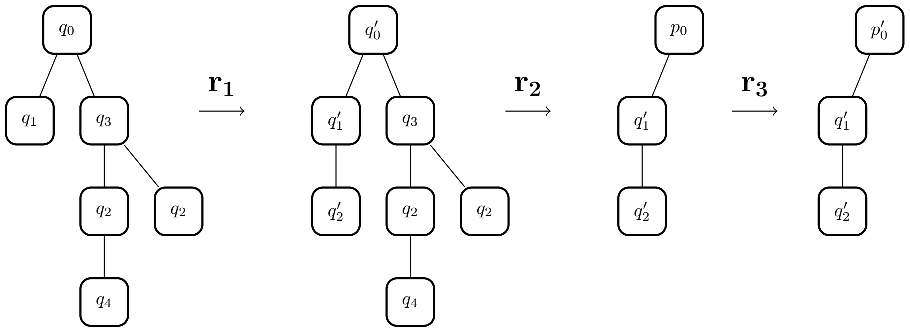

Figure 5.1: Application of the rules r_1, r_2 and r_3 to a configuration of the NCS $\mathcal{N}$ described in Example 41.

Input:	An NCS $\mathcal{N} = (Q, \delta)$ and two states *init* and *fin*
Decide:	If *init* can cover *fin*

By an already existing result from the literature ([51, Theorem 7]), we know that

Theorem 43. The coverability problem for NCS is $\mathbf{F}_{\epsilon_0}$-hard.

By Theorem 43, it follows that if we give polynomial-time reductions from the coverability problem for NCS to the coverability problems for bounded-path broadcast networks and depth-bounded π-calculus processes, then the latter two problems are $\mathbf{F}_{\epsilon_0}$-hard. In the next two sections, we will present the main ideas behind both these polynomial-time reductions.

Remark 44. We can also show that NCS are well-structured transition systems under a suitably chosen well-quasi-ordering. It follows then that the usual coverability problem for WSTS can also be defined for NCS. The coverability problem for NCS as defined in Definition 42 will then be a special case of this general coverability problem. However, for our purposes, it is sufficient to work with the special case defined in Definition 42.

5.2 Lower bound for bounded-path broadcast networks

Our first result in this chapter is regarding the model of broadcast networks. The intuitive idea behind the notion of a broadcast network was already discussed in Subsection 4.4.2. Hence, here we begin our discussion on broadcast networks by formally presenting its syntax and semantics. The definitions here are taken from the ones given in [28].

Definition and semantics

Each agent in a broadcast network executes a given finite-state protocol. Before we state the formal semantics of a broadcast network, we formalize the notion of a protocol.

Definition 45. A broadcast protocol is a tuple $\mathcal{P} = (Q, I, \Sigma, \Delta)$ where Q is a finite set of states, $I \subseteq Q$ is the set of initial states, Σ is a finite set of messages and $\Delta \subseteq Q \times \{!a, ?a, : a \in \Sigma\} \times Q$ is the transition relation.

We write $q \xrightarrow{!a} q'$ (resp. $q \xrightarrow{?a} q'$) for $(q, !a, q') \in \Delta$ (resp. $(q, ?a, q') \in \Delta$). A transition $q \xrightarrow{!a} q'$ (resp. $q \xrightarrow{?a} q'$) intuitively corresponds to broadcasting (resp. receiving) the message a.

Each agent of a broadcast network is labeled by some state of the protocol and is situated on a node of a graph called the communication topology. To formalize this, given a broadcast protocol $\mathcal{P} = (Q, I, \Sigma, \Delta)$, a *configuration* of $\mathcal{P}$ is a labeled graph $\gamma = (\mathsf{N}, \mathsf{E}, \mathsf{L})$ where N is a finite set of nodes, $\mathsf{E} \subseteq \mathsf{N} \times \mathsf{N}$ is a finite set of (undirected) edges specifying for every pair of nodes whether or not there is a communication link between them and $\mathsf{L} : \mathsf{N} \to Q$ is a labeling function that specifies the current state of each agent at each node. A configuration is *initial* if the state of each agent in the configuration belongs to I. Further, for any k, we let $\mathcal{T}_k(\mathcal{P})$ denote the set of all k-path bounded configurations of the network.

The semantics of the broadcast network of a protocol $\mathcal{P}$ is given by means of transitions between its configurations. There is a step from the configuration $\gamma = (\mathsf{N}, \mathsf{E}, \mathsf{L})$ to the configuration $\gamma' = (\mathsf{N}', \mathsf{E}', \mathsf{L}')$ (denoted by $\gamma \to \gamma'$) if $\mathsf{N}' = \mathsf{N}$, $\mathsf{E}' = \mathsf{E}$ and there exists a node n and a message $a \in \Sigma$ such that $(\mathsf{L}(\mathsf{n}), !a, \mathsf{L}'(\mathsf{n})) \in \Delta$, and for every other node n', if $(\mathsf{n}, \mathsf{n}') \in \mathsf{E}$, then $(\mathsf{L}(\mathsf{n}), ?a, \mathsf{L}'(\mathsf{n}')) \in \Delta$; otherwise $\mathsf{L}(\mathsf{n}') = \mathsf{L}'(\mathsf{n}')$. Intuitively, a step consists of a node n broadcasting some message a, which is then received by *all* of its neighbors; all the other nodes do nothing. This step relation naturally induces a notion of reachability between configurations of a broadcast network.

Given a state f and a configuration γ_0, we say that γ_0 can cover f if there is a run from γ_0 to some configuration γ such that $f \in \mathsf{L}(\gamma)$. Intuitively, this means that starting from γ_0 we can reach a configuration in which some agent is in the state f. The *coverability* problem for broadcast networks is to decide, given a broadcast protocol $\mathcal{P}$ and a state f, whether there is some initial configuration that can cover f.

Example 46. We consider the broadcast protocol given in Figure 5.2. Figure 5.3 shows an execution in this protocol covering the state $(e, 0)$.

Our contribution

It is known that the coverability problem for broadcast networks is undecidable ([54, Theorem 1]). To overcome this undecidability result, we look at the coverability problem for *bounded-path* broadcast networks, which is defined as follows.

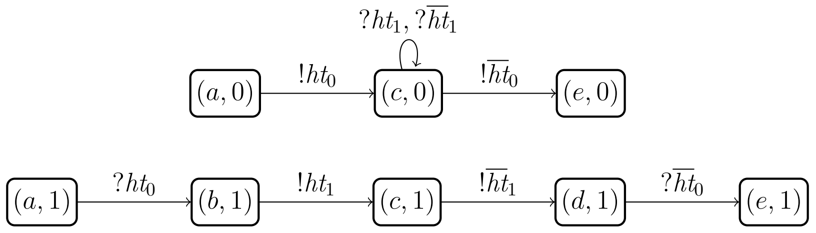

Figure 5.2: Example of a broadcast protocol where we set $I = \{(a,0),(a,1)\}$ and $\Sigma = \{ht_i, \overline{ht}_i : 0 \le i \le 1\}$. If for a state (f,i), we have not depicted what happens when message m is received at (f,i), we assume that $(f,i) \xrightarrow{?m} (\perp, i)$. Here $(\perp,0)$ and $(\perp,1)$ are new *sink* states, i.e., states with no outgoing transition.

Definition 47. The coverability problem for bounded-path broadcast networks is defined as the following problem:

Input:	A protocol $\mathcal{P} = (Q, I, \Sigma, \Delta)$, a state $f \in Q$ and a number k
Decide:	If there is some initial configuration in $\mathcal{T}_k(\mathcal{P})$ which can cover f

Note that in this problem, we are only interested in configurations of depth at most k. As discussed in the previous chapter, this problem is in $\mathbf{F}_{\epsilon_0}$. We complement this upper bound by means of the following lower bound.

Theorem 48. The coverability problem for bounded-path broadcast networks is $\mathbf{F}_{\epsilon_0}$-hard.

Together with the upper bound, this establishes the $\mathbf{F}_{\epsilon_0}$-completeness of the coverability problem for bounded-path broadcast networks. We now give an overview of the proof of Theorem 48. Formal details behind the proof can be found in [15, Sections 4 and 5].

The lower bound is achieved by giving a polynomial-time reduction from the coverability problem for NCS to the coverability problem for bounded-path broadcast networks. Roughly speaking, there are three main ideas behind this reduction. Let $\mathcal{N} = (Q, \delta)$ be a k-NCS. The first idea is that we can interpret each configuration C of $\mathcal{N}$ as a network whose communication topology is given by the graph of C. Intuitively, we have an agent situated at each node of C, whose state is the current state of that node in C, along with its distance from the root. The second idea is that, once we adopt this lens of viewing configurations as topologies, a step in the NCS $\mathcal{N}$ from C is essentially a sequence of communication exchanges between some i agents where $i \le k$, starting at the root and going down the tree. We then show that these communication exchanges can be simulated by gadgets of a broadcast protocol, called the *simulator protocol.* This essentially allows us to show that starting from large enough

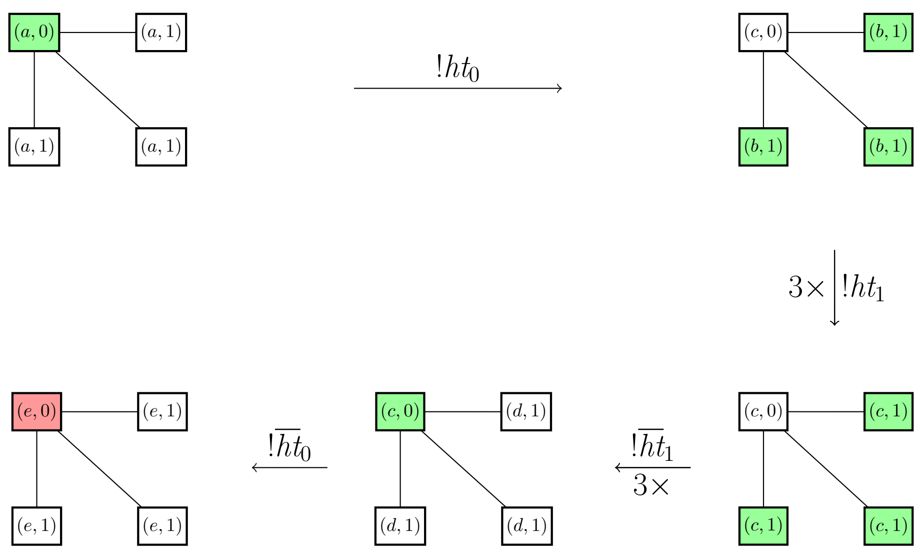

Figure 5.3: Example of an execution covering $(e, 0)$ in the broadcast protocol given in Figure 5.2. The nodes marked in green make the broadcasts, i.e., first the node on the topmost left broadcasts ht_0, then all the other nodes broadcast ht_1 in some order, and then $\overline{ht}_1$ in some order, and then the node on the topmost left broadcasts $\overline{ht}_0$.

initial configurations which resemble a "tree"-like structure in our broadcast network, we can simulate runs of the NCS $\mathcal{N}$. The third and final idea is that we can come up with a gadget called the *seeker protocol*, which starting from any topology, searches for a tree-like sub-structure in that topology. Hence, first deploying the seeker protocol and then attaching it with the simulator protocol allows us to simulate computations of the NCS $\mathcal{N}$, thereby giving the desired hardness result.

5.3 Lower bound for depth-bounded π-calculus processes

We now move on to the formalism of π-calculus. We have given an informal description of π-calculus processes in Subsection 4.4.3. Here, we focus on formally defining π-calculus and presenting its semantics.

Definition and semantics

We begin by presenting the syntax and the semantics of the version of π-calculus that we will use. The definitions here are taken from the ones given in [114].

We assume that there is a countable collection of *names* (denoted by $x, y, \dots$) and a countable collection of *process identifiers* (denoted by $A, B, \dots$). Each name and identifier has an associated *arity* in $\mathbb{N}$. We use boldface letters like $\mathbf{x}, \mathbf{y}$ to denote (possibly empty) vectors over names and denote substitution of names by $[\mathbf{x}/\mathbf{y}]$, i.e., if $\mathbf{x} = x_1, \dots, x_n$ and $\mathbf{y} = y_1, \dots, y_n$, then $[\mathbf{x}/\mathbf{y}]$ denotes a mapping in which each y_i is mapped to x_i, and every other name is mapped to itself. Intuitively, the current state of an agent or a *thread* is given by its process identifier and channels are described by names.

A *process term* (or simply a term) P is either the unit process 0, or a parameterized process identifier $A(\mathbf{x})$, or any term obtained by the standard operations of parallel composition $P_1 \mid P_2$, external choice $\pi_1 \cdot P_1 + \pi_2 \cdot P_2$ and name restriction $(\nu x)P_1$. Here P_1 and P_2 are themselves terms, and π_1 and π_2 are prefixes which can either be an *input prefix* $x(\mathbf{y})$ or an *output prefix* $\bar{x}(\mathbf{y})$ or the empty string. A *thread* is a term of the form $A(\mathbf{x})$. We use Π and Σ to denote (indexed) parallel composition and external choice. We further use $(\nu\mathbf{x})$ to denote $(\nu x_1)(\nu x_2)\dots(\nu x_n)$ where $\mathbf{x} = x_1, \dots, x_n$. The application of a substitution of names σ to a term P, denoted by $\sigma(P)$, is defined in the usual way.

An occurrence of a name x in a term P is called *free* if it is not below a (νx) or an input prefix $y(x)$. We let $\mathtt{fn}(P)$ denote the set of free names of P. A *bound name* of P is a name of P which is not free. We say that P is *closed* if $\mathtt{fn}(P) = \emptyset$. We use the usual *structural congruence relation* $P \equiv Q$ on process terms, i.e., $P \equiv Q$ if P is syntactically equal to Q upto renaming and reordering of bound names, associativity and commutativity of parallel composition and external choice, elimination of units ($(P \mid 0) \equiv P, (\nu x)0 \equiv 0$) and *scope extrusion* ($(\nu x)(P \mid Q) \equiv (\nu x)P \mid Q$ if $x \notin \mathtt{fn}(Q)$).

A *configuration* is a closed term of the form $(\nu\mathbf{x})\,(\Pi_{i\in I} A_i(\mathbf{x}_i))$. A *process* $\mathcal{P}$ is a pair $(I, \mathcal{E})$ where I is an *initial configuration* and $\mathcal{E}$ is a set of *parametric equations* of the form $A(\mathbf{x}) = P$ where A is an identifier and P is a term such that 1) every identifier in $\mathcal{P}$ is defined by exactly one equation in $\mathcal{E}$ and 2) if $A(\mathbf{x}) = P$ is an equation, then $\mathtt{fn}(P) \subseteq \{\mathbf{x}\}$. We assume that all the equations are given in the following form:

$$A(\mathbf{x}) = \sum_{i\in I} \pi_i.(\nu\mathbf{x}_i)\left(\prod_{j\in J_i} A_j(\mathbf{x}_j)\right)$$

Operational semantics. Let $\mathcal{P} = (I, \mathcal{E})$ be a process. We define a transition relation on the set of configurations using $\mathcal{E}$ as follows. Let P and Q be configurations. Then $P \to Q$ if and only if the following conditions are satisfied:

- $P \equiv (\nu\mathbf{u})(A(\mathbf{v}) \mid B(\mathbf{w}) \mid P')$,
- The defining equation of A in $\mathcal{E}$ is of the form $A(\mathbf{x}) = x(\mathbf{x}').(\nu\mathbf{x}'')(M) + M'$,
- The defining equation of B in $\mathcal{E}$ is of the form $B(\mathbf{y}) = \bar{y}(\mathbf{y}').(\nu\mathbf{y}'')(N) + N'$,
- $\sigma = [\mathbf{v}/\mathbf{x}, \mathbf{w}/\mathbf{y}, \mathbf{w}'/\mathbf{x}', \mathbf{z}_A/\mathbf{x}'', \mathbf{z}_B/\mathbf{y}'']$ where $\mathbf{z}_A, \mathbf{z}_B$ are fresh names and $\mathbf{w}'$ is the set of names assigned to $\mathbf{y}'$ under the mapping $[\mathbf{w}/\mathbf{y}]$.

- $\sigma(x) = \sigma(y)$ and
- $Q \equiv (\nu \mathbf{u}, \mathbf{z}_A, \mathbf{z}_B)(\sigma(M) \mid \sigma(N) \mid P')$

We denote such a step by $P \to Q$. The intuitive idea behind this relation is that there is a channel $\sigma(y)$ through which a thread with identifier B sends the names corresponding to $\mathbf{w}'$, and these names are received by a thread with identifier A along the same channel $\sigma(x) = \sigma(y)$. Once that happens, the thread A creates new channels $\mathbf{z}_A$ and then works according to the description of the term M. Similarly, the thread B creates new channels $\mathbf{z}_B$ and then works according to the description of the term N. We say that a configuration P is coverable in $\mathcal{P}$ if $P \equiv (\nu \mathbf{x})P'$ and there exists $Q \equiv (\nu \mathbf{x})(P' \mid R)$ such that $I \xrightarrow{*} Q$.

Depth-bounded processes. We now define the class of depth-bounded processes. The nesting of restrictions *nest* of a term P is defined inductively as follows: $nest(0) = nest(A(\mathbf{x})) = nest(\pi_1 \cdot P_1 + \pi_2 \cdot P_2) = 0$, $nest((\nu x)P) = 1 + nest(P)$ and $nest(P_1 \mid P_2) = \max\{nest(P_1), nest(P_2)\}$. The *depth* of a term P is the minimal nesting of restrictions of terms in the congruence class of P:

$$depth(P) := \min\{nest(Q) : Q \equiv P\}$$

Definition 49. A set of configurations $\mathcal{C}$ is called k-depth-bounded if the depth of all configurations in $\mathcal{C}$ is at most k. $\mathcal{C}$ is called depth-bounded if there is some k such that it is k-depth-bounded. A process $\mathcal{P}$ is called (k) depth-bounded if the set of reachable configurations from the initial configuration of $\mathcal{P}$ is (k) depth-bounded.

Example 50. The following example intuitively demonstrates a system in which there is one "Level0" thread which can spawn "Level1" threads by using a "New1" thread. The intuition is that once a New1 thread receives a message from a Level0 thread by some channel (corresponding to the name x in the equations of Level0 and New1), it creates a fresh name (corresponding to the name y in the equation of New1), creates two new threads with identifiers Level1 and New2, gives Level1 the access to the channels corresponding to x and y and gives New1 the access to the channel corresponding to y. Then, each Level1 thread can itself spawn "Level2" threads by using their own "New2" threads.

$$\begin{aligned}
Level0(x) &= \bar{x}().Level0(x)\\
New1(x) &= x().((\nu y)(New1(x) \mid Level1(x,y) \mid New2(y)))\\
Level1(x,y) &= \bar{y}().Level1(x,y)\\
New2(y) &= y().((\nu z)(New2(y) \mid Level2(y,z) \mid New3(z)))\\
Level2(y,z) &= \bar{z}().Level2(y,z)\\
New3(z) &= z().New3(z)
\end{aligned}$$

Suppose we set $I = (\nu x)(Level0(x) \mid New1(x))$. Then the following is a valid run:

$$\begin{aligned} I &\to (\nu x)(Level0(x) \mid New1(x) \mid (\nu y)(Level1(x,y) \mid New2(y))) \\ &\to (\nu x)(Level0(x) \mid New1(x) \mid (\nu y)(Level1(x,y) \mid New2(y) \mid \\ &\qquad\qquad\qquad\qquad (\nu z)(Level2(y,z) \mid New3(z)))) \end{aligned}$$

We note that the depth of the last configuration in this run is 3. Indeed, we can show that the depth of any reachable configuration from I is at most 3.

Our contribution

The main problem that we consider regarding depth-bounded π-calculus processes is the coverability problem, which is defined as follows:

Definition 51. The coverability problem for depth-bounded π-calculus processes is defined as the following problem:

Input:	A k-depth-bounded process $\mathcal{P} = (I, \mathcal{E})$ and a configuration P
Decide:	If P is coverable in $\mathcal{P}$

We have already seen in the previous chapter that this problem is in $\mathbf{F}_{\epsilon_0}$. We complement this upper bound by means of the following result.

Theorem 52. The coverability problem for depth-bounded processes is $\mathbf{F}_{\epsilon_0}$-hard.

Here, we assume that the input consists of a process $\mathcal{P}$ and a number k such that $\mathcal{P}$ is k-depth-bounded. Together with the upper bound given in the previous chapter, this proves that the coverability problem is $\mathbf{F}_{\epsilon_0}$-complete.

We now sketch the main details behind the proof of the lower bound. The formal details behind the proof can be found in [18, Sections 3 and 4]. The lower bound is obtained in two stages. First, we introduce a model called *k-nested counter system with levels* (k-NCSL or simply NCSL). Intuitively, an NCSL is the same as an NCS, except that steps between configurations involve changes to at most 2 nodes in the configuration, a parent node and a child node. In some sense, NCS can do "global" steps by means of changing an entire path from the root; in contrast, NCSL can only do "local" steps. We show that given an NCSL $\mathcal{N}$, it is possible to simulate each step of $\mathcal{N}$ by means of a depth-bounded π-calculus process. The intuition behind this simulation is that each step in an NCSL corresponds to a communication between two nodes, and this communication can be captured by a π-calculus process. Having proven this simulation, we then show that the coverability problem for NCSL is $\mathbf{F}_{\epsilon_0}$-hard by giving a reduction from the coverability problem for NCS. The intuitive idea behind this reduction is that we can simulate one step of a given k-NCS by a series of ck steps of an NCSL for some constant c. The simulation first picks some rule of the given NCS to work with and then simulates this rule "locally" along a path of a configuration, starting from the root node.

Putting these two simulations together, we then get the required $\mathbf{F}_{\epsilon_0}$-hardness result for depth-bounded π-calculus processes.

5.4 Related work

There has been a great deal of work in proving lower bounds for different WSTS models. We refer the interested reader to [106] for a catalog of upper and lower bounds for various well-structured transition systems.

The coverability problem for broadcast networks was first considered in [54], where it was shown that it is undecidable for the general case. The same paper also showed that the problem becomes decidable when restricted to bounded-path graphs by using the framework of WSTS. However, the precise complexity of the problem has been open since then, and we settle it by our results. Ever since [54] was published, there has been a flurry of papers related to the verification of broadcast networks and its extensions with probabilities, registers and clocks [57, 55, 56, 31, 30, 29, 52, 2, 16]. A particularly interesting variant of broadcast networks is the model where messages are allowed to be lost; this has also been studied under the name of reconfigurable broadcast networks [19, 53, 28, 47]. For this variant, coverability is decidable and in polynomial time. Further, some parameterized verification problems beyond coverability, such as synchronization and repeated coverability, are also decidable in polynomial time.

π-calculus is an expressive formalism of distributed computation first considered in the papers [95, 96]. Due to its immense power, all interesting problems regarding it become undecidable. The depth-bounded fragment is the most expressive fragment known till now, for which some problems are decidable. The decidability of coverability for this fragment was shown in [94] by using the theory of WSTS. The paper [114] proposes another algorithm for the coverability problem, which works even when the bound on the depth of the process is not known a priori. However, the precise complexity of the problem was open prior to the publication of our result.

5.5 Conclusion

We have shown that the coverability problems for bounded-path broadcast networks and depth-bounded π-calculus processes are $\mathbf{F}_{\epsilon_0}$-hard. Combined with the upper bounds from the previous chapter, this proves that both these problems are $\mathbf{F}_{\epsilon_0}$-complete. These two lower bound results solve open problems from [79, 114].

The common thread between these two models is the notion of *bounded depth*. This intuitive notion of bounded depth has also been explored in other contexts

(For instance, see the models in [42, Section 8.3]). Depth-bounded systems have been mentioned as having the ability to model a wide variety of distributed systems [27]. It is possible that the central idea behind our reductions can also be used to prove hardness results for other bounded depth models mentioned in [42, Section 8.3].

Finally, our results for both these models do not give tight complexity bounds when the underlying bound on the depth is a fixed constant. Resolving these two problems in such a case is an intriguing direction of research for future work. We believe that the models of NCS and NCSL will also be useful for such questions.

Part II

Linear Arithmetic Theories

Chapter 6

Introduction and Background

The second part of this thesis deals with the applications of linear arithmetic theories to provide efficient algorithms for the verification of parameterized systems. A linear arithmetic theory is a logical theory where the variables are allowed to take values from some underlying number system equipped with an addition operation and an order relation. We show that by combining linear arithmetic theories over various domains, we can obtain simple, complete and efficient algorithms for analyzing interesting classes of parameterized systems.

We use this chapter to set up the necessary definitions and results that we shall use in this part of the thesis. In the next three chapters, we explain our contributions to parameterized verification using results from linear arithmetic theories.

In the following, we will mostly use the notations from [101, 17] to describe the linear arithmetic theories that are of interest to us. For a domain $\mathbb{A} \in \{\mathbb{N}, \mathbb{Q}, \mathbb{Q}_{\geq 0}, \mathbb{Z}\}$, its underlying linear arithmetic theory is the first-order theory over the structure $\langle \mathbb{A}, 0, 1, +, < \rangle$ where $+$ and $<$ are the standard addition and order operations over $\mathbb{A}$. In a straightforward manner, we can extend our syntax with the abbreviations: $\leq, =, \geq, >$ and $ax = \sum_{1 \leq i \leq a} x$ where $a \in \mathbb{N}$ and x is a variable. A *linear polynomial* of such a first-order theory is a term of the form $\sum_{1 \leq i \leq n} a_i \cdot x_i + b$ where each x_i is a variable, each $a_i \in \mathbb{N}$ and $b \in \mathbb{A}$. An *atomic formula* of the underlying theory is a term of the form $p(\mathbf{x}) \bowtie q(\mathbf{x})$ where p and q are linear polynomials over the variables $\mathbf{x}$ and $\bowtie \in \{<, \leq, =, \geq, >\}$. Formulas of the theory are built from atomic formulas by the standard negation $\neg$, conjunction $\wedge$, disjunction $\vee$, existential quantification $\exists x$ and universal quantification $\forall x$ operations. A sentence is a formula where each variable is under the scope of some quantifier.

Of particular interest to us are the *existential fragments* of these theories, which consist only of existential formulas, i.e., formulas where no variable is universally quantified. Another special class is the class of *linear equations*, which are formulas constructed by taking conjunction of atomic formulas of the form $p(\mathbf{x}) = q(\mathbf{x})$ for some linear polynomials p and q.

Having defined the theories of interest to us, we now state the results that we shall use regarding these theories. The first one is regarding the existential fragment of the theory of natural numbers, also called *existential Presburger arithmetic.*

Theorem 53. (Section 5.1 of [78]). Deciding the truth of sentences in existential Presburger arithmetic is in NP.

The second one is about finding solutions to linear equations over various domains (See [83, 84]).

Theorem 54. Solving linear equations over $\mathbb{Q}_{\geq 0}$ and $\mathbb{Z}$ is in P.

Our contributions

Having covered the necessary results that we shall use, we make a small remark on our contributions in the second part of the thesis.

The next three chapters cover contributions III and IV, which were mentioned in Subsections 1.2.1 and 1.2.2, respectively. Chapter 7 contains results on the cut-off problem for rendez-vous protocols. Chapter 8 covers our results on the complexity of verification of threshold automata, and Chapter 9 contains our main results regarding the parameterized complexity of the safety verification problem for threshold automata. These results are discussed in depth in the respective chapters.

Chapter 7

The cut-off problem for rendez-vous protocols

A rendez-vous protocol is a parameterized system in which agents communicate in pairs. At any point in time, two agents meet, exchange a message and then update their states according to some transition relation. The simplicity of this model has lent itself to many papers dedicated to understanding the decidability and complexity of the main verification problems of this model and its variants [76, 62, 63]. However, almost all of the existing works in the literature have concentrated only on questions that ask if every initial configuration satisfies a given property (or the dual question of whether some initial configuration violates a given property). While this is a powerful framework for asking interesting questions, in some cases, it might happen that some properties hold only beyond a certain population size in a protocol. (We will present an example of this phenomenon in Section 7.1). The usual universal quantification framework cannot be used to identify such cases.

Motivated by this, the authors of [81] considered questions that go beyond the usual existential/universal quantification framework and looked at determining the existence of *cut-offs* in the system, i.e., a bound on the number of agents such that a given property holds for all population sizes beyond this bound. Concretely they studied the following problem called the *cut-off problem*: Given a rendez-vous protocol with an initial state and a final state, does there exists a number B such that for all $n \geq B$, the initial configuration with n agents in the initial state can reach the final configuration with n agents in the final state. Essentially, the cut-off problem asks if some reachability property is satisfied by the protocol *asymptotically in the limit*, i.e., for all but finitely many population sizes. The authors prove that the problem is in EXPSPACE, but leave open the precise complexity of the problem.

Our main contribution is to improve upon this bound and prove that the problem is actually P-complete. More specifically, we show that by leveraging results about linear arithmetic theories, it is possible to obtain efficient algorithms for the cut-off problem. Using the framework of linear arithmetic theories, we also manage to improve upon some of the other algorithms given in [81] for special cases of the cut-off problem.

Horn and Sangnier also considered a more general version of rendez-vous protocols in which one distinguished agent, called a leader, is allowed to execute its own protocol. They proved that the cut-off problem for this general version is decidable and $\mathbf{F}_\omega$-hard. We will not consider that version of the problem in this chapter, i.e., the model of rendez-vous protocols that we will consider in this chapter are *leaderless*.

The results of this chapter are taken from [21], which is reprinted in Appendix D. That paper also has results pertaining to some special cases of rendez-vous protocols (with a leader). However, since the focus here is on presenting the main results of [21] with intuitive ideas, we have not considered those special cases in this chapter. For more details on our results on these special cases, we refer the reader to Sections 6 and 7 of [21].

We note that our polynomial-time algorithm for the cut-off problem for (leaderless) rendez-vous protocols is in some sense surprising, as the addition of a single leader makes the problem non-primitive recursive. This demonstrates the power of a leader in the context of rendez-vous protocols and also the usefulness of linear arithmetic theories in identifying and isolating tractable islands of problems concerning parameterized systems.

The rest of this chapter is structured as follows. We begin by formally defining rendez-vous protocols and stating our main result. We then reformulate the notion of a rendez-vous protocol as a Petri net, which allows us to use results from the theory of Petri nets in order to solve the cut-off problem. Then, we discuss a polynomial-time algorithm for the cut-off problem and conclude with a section on related work and a short summary.

7.1 Rendez-vous protocols

Having discussed the intuitive notion of a rendez-vous protocol at the beginning of this chapter, we proceed to formally define the model and state its semantics. The definitions given here are slight modifications of the ones given in [81].

Definition and semantics

We begin by formalizing the notion of a protocol that all agents in our system will execute.

Definition 55. A *rendez-vous protocol* $\mathcal{P}$ is a tuple $(Q, \Sigma, \mathit{init}, \mathit{fin}, R)$ where Q is a finite set of *states*, Σ is the communication alphabet consisting of a finite set of *messages*, $\mathit{init}, \mathit{fin} \in Q$ are the *initial* and *final* states respectively and $R \subseteq Q \times \{!a, ?a : a \in \Sigma\} \times Q$ is the set of *rules*.

A configuration C of $\mathcal{P}$ is a multiset of states, where $C(q)$ should be interpreted as the number of agents in state q. An initial (resp. final) configuration

C is a configuration such that $C(q) = 0$ if $q \neq \mathit{init}$ (resp. $C(q) = 0$ if $q \neq \mathit{fin}$). We use C^n_{init} (C^n_{fin}) to denote the initial (resp. final) configuration such that $C^n_{\mathit{init}}(\mathit{init}) = n$ (resp. $C^n_{\mathit{fin}}(\mathit{fin}) = n$).

The operational semantics of a rendez-vous protocol $\mathcal{P}$ is given by means of a transition system between the configurations of $\mathcal{P}$. We say that there is a transition between C and C', denoted by $C \Rightarrow C'$, if there exists a message a in Σ and rules $(p, !a, p'), (q, ?a, q')$ in R such that $C \geq \wr p, q \int$ and $C' = C - \wr p, q \int + \wr p', q' \int$. Intuitively, the configuration C has agents at states p and q, and the agent at state p sends the message a and moves to p', and the agent at state q receives this message and moves to q'. As usual, $\xRightarrow{*}$ denotes the reflexive and transitive closure of $\Rightarrow$.

Example 56. Let us consider the rendez-vous protocol in Figure 7.1, which is taken from a slightly modified version of the family of protocols described in Figure 5 of [81]. The protocol has five rules, two from *init* to q_1 labeled by $!a$ and $?a$ respectively, two from *init* and q_1 to *fin* labeled by $?b$ and $!b$ respectively and finally, a self-loop at *fin* labeled by $!b$.

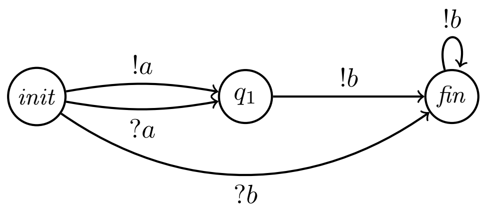

Figure 7.1: An example of a rendez-vous protocol

The cut-off problem. The main focus of this chapter is the *cut-off problem* which is defined as follows.

Definition 57. The cut-off problem for rendez-vous protocols is defined as the following decision problem:

Input:	A rendez-vous protocol $\mathcal{P}$
Decide:	If there is $B \in \mathbb{N}$ such that $C^n_{\mathit{init}} \xRightarrow{*} C^n_{\mathit{fin}}$ for every $n \geq B$

If such a B exists, then we say that $\mathcal{P}$ admits a cut-off and that B is a cut-off for $\mathcal{P}$. The intuitive idea behind a cut-off is that it ensures that all but finitely many initial configurations satisfy the property that they can reach a final configuration. As we shall see in the next example, it might be the case that protocols sometimes need a minimum number of agents in order to satisfy a property, and such instances cannot be identified with the usual type of questions which quantifies over every initial configuration.

Example 58. Consider the protocol given in Figure 7.1. We can show that 4 is a cut-off for this protocol. Indeed, if $n \geq 4$, then we have the run $C^n_{init} \Rightarrow \wr(n-2) \cdot init + 2 \cdot q_1 \wr \Rightarrow \wr(n-3) \cdot init + q_1 + 2 \cdot fin \wr \Rightarrow \wr(n-4) \cdot init + 4 \cdot fin \wr$. The first transition involves sending and receiving the message a from the state *init*, and the other two involve sending the message b from q_1 and receiving it from *init*. Once we reach $\wr(n-4) \cdot init + 4 \cdot fin \wr$, we can reach C^n_{fin} by repeatedly using the rules $(fin, !b, fin)$ and $(init, ?b, fin)$.

Further, we can show that no number strictly less than 4 can be a cut-off for this protocol. Indeed, suppose $C^n_{init} \xRightarrow{*} C^n_{fin}$. Since we need at least two agents for a transition to occur, it follows that $n \geq 2$. By construction of the protocol, the first transition along this run must be $C^n_{init} \Rightarrow \wr(n-2) \cdot init + 2 \cdot q_1 \wr$. If $n = 2$, then the run gets stuck at this configuration because no agent is at a state capable of receiving a message. If $n = 3$, then the only transition that is possible is $\wr init + 2 \cdot q_1 \wr \Rightarrow \wr q_1 + 2 \cdot fin \wr$, at which point we reach a configuration where no agent is capable of receiving a message. This implies that $n \geq 4$ and so no number strictly less than 4 can be a cut-off for this protocol.

Our contribution

Our main contribution to the theory of rendez-vous protocols is the following result.

Theorem 59. The cut-off problem for rendez-vous protocols is P-complete.

In particular, our result improves upon the previously known upper bound of EXPSPACE [81, Theorem 27]. The polynomial-time algorithm for the cut-off problem is obtained by actually considering a generalized model of rendez-vous protocols called *Petri nets* and then proving that the cut-off problem for Petri nets is in P. We now proceed to formally define Petri nets.

7.2 Petri Nets

We now show that rendez-vous protocols can be cast in terms of one of the most well-studied concurrency models, namely Petri nets [103, 68, 61]. The advantage of doing this is that we can now leverage results from Petri net theory to solve problems for rendez-vous protocols.

Definition 60. A *Petri net* is a tuple $\mathcal{N} = (P, T, Pre, Post)$ where P is a finite set of *places*, T is a finite set of *transitions*, Pre and $Post$ are matrices over $\mathbb{N}$ whose rows and columns are indexed by P and T respectively. The *incidence matrix* $\mathcal{A}$ of $\mathcal{N}$ is defined as $\mathcal{A} = Post - Pre$.

A *marking* of $\mathcal{N}$ is a multiset $M \in \mathbb{N}^P$, which intuitively denotes the number of *tokens* that are present in every place of the net. The transitions of a Petri net are responsible for creating, destroying and moving around the tokens in the places of the Petri net.

For $t \in T$ and markings M and M', we say that M' is reached from M by firing t, denoted by $M \xrightarrow{t} M'$, if for every place p, $M(p) \geq Pre[p,t]$ and $M'(p) = M(p) + \mathcal{A}[p,t]$. Intuitively, when t is fired, it first removes $Pre[p,t]$ tokens from each place p and then puts $Post[p,t]$ tokens in each place p. We use $M \rightarrow M'$ to mean that $M \xrightarrow{t} M'$ for some t and we use $\xrightarrow{*}$ to mean the reflexive and transitive closure of $\rightarrow$.

A *firing sequence* is any sequence of transitions $\sigma \in T^*$. The support of σ, denoted by $[\![\sigma]\!]$, is the set of all transitions which appear in σ. Given a firing sequence $\sigma = t_1, t_2, \dots, t_k$, we let $M \xrightarrow{\sigma} M'$ denote that there are markings $M_1, \dots, M_{k-1}$ such that $M \xrightarrow{t_1} M_1 \xrightarrow{t_2} M_2 \dots M_{k-1} \xrightarrow{t_k} M'$.

Marking equation of a Petri net system. A *Petri net system* is a triple $(\mathcal{N}, M, M')$ where $\mathcal{N}$ is a Petri net and M and M' are markings. The *marking equation* for $(\mathcal{N}, M, M')$ is the equation

$$M' = M + \mathcal{A}\mathbf{v}$$

over the variables $\mathbf{v}$. It is well known that $M \xrightarrow{\sigma} M'$ implies $M' = M + \mathcal{A}\overrightarrow{\sigma}$, where $\overrightarrow{\sigma} \in \mathbb{N}^T$ is the the *Parikh image* of σ, defined as the vector whose component $\overrightarrow{\sigma}[t]$ for a transition t is equal to the number of times t appears in σ. Therefore, if $M \xrightarrow{\sigma} M'$, then $\overrightarrow{\sigma}$ is a nonnegative integer solution of the marking equation [99, Section V. B]. The converse does not hold.

From rendez-vous protocols to Petri nets.

Rendez-vous protocols can be seen as a special class of Petri nets in which no tokens are created or destroyed during a run. Indeed suppose $\mathcal{P} = (Q, \Sigma, \mathit{init}, \mathit{fin}, R)$ is a rendez-vous protocol. Then we can construct a Petri net $\mathcal{N}_\mathcal{P}$ whose set of places is Q and whose set of transitions is obtained as follows: For every $a \in \Sigma$ and every pair of rules $r = (q, !a, s)$ and $r' = (q', ?a, s')$ in R, we have a transition $t_{r,r'}$ in $\mathcal{N}_\mathcal{P}$ which removes tokens according to the multiset $\wr q, q' \int$ and puts tokens according to the multiset $\wr s, s' \int$. We demonstrate this construction by means of an example.

Example 61. Let us consider the rendez-vous protocol $\mathcal{P}$ from Figure 7.1. Its associated Petri net $\mathcal{N}_\mathcal{P}$ is given in Figure 7.2. The three places of the Petri net correspond to the three states of the protocol $\mathcal{P}$. We also have three transitions: t_1 corresponds to the pair $(\mathit{init}, !a, q_1), (\mathit{init}, ?a, q_1)$, t_2 corresponds to the pair $(q_1, !b, \mathit{fin}), (\mathit{init}, ?b, \mathit{fin})$ and t_3 corresponds to the pair $(\mathit{fin}, !b, \mathit{fin}), (\mathit{init}, ?b, \mathit{fin})$.

Our construction of $\mathcal{N}_\mathcal{P}$ ensures that any marking of $\mathcal{N}_\mathcal{P}$ is also a configuration of $\mathcal{P}$ and vice versa. Moreover, it preserves the reachability relation of the protocol $\mathcal{P}$. Hence, if we define and efficiently solve a version of the cut-off problem for Petri nets that conservatively extends the cut-off problem for

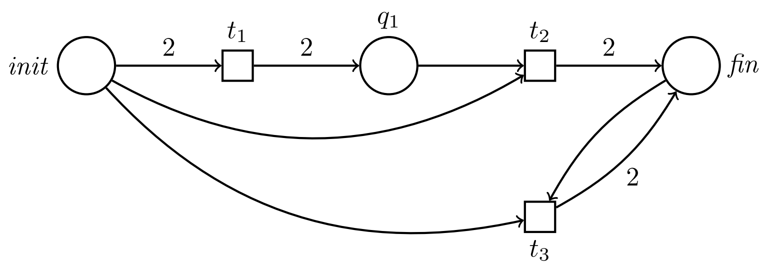

Figure 7.2: Petri net corresponding to the protocol from Figure 7.1

rendez-vous protocols, then we can also efficiently solve the latter. This is what we do now by defining the cut-off problem for Petri nets in the following way.

Definition 62. The cut-off problem for Petri nets is defined as the following problem.

Input:	A Petri net system $(\mathcal{N}, M, M')$
Decide:	If there is $B \in \mathbb{N}$ such that $nM \xrightarrow{*} nM'$ for every $n \geq B$

If such a B exists, then we say that $(\mathcal{N}, M, M')$ admits a cut-off and that B is a cut-off for $\mathcal{N}$. Since reachability is preserved by moving from a protocol $\mathcal{P}$ to its net $\mathcal{N}_\mathcal{P}$ and vice versa, it follows that B is a cut-off for $\mathcal{P}$ if and only if B is a cut-off for the Petri net system $(\mathcal{N}_\mathcal{P}, \wr \mathit{init} \int, \wr \mathit{fin} \int)$.

Example 63. Let us consider the Petri net $\mathcal{N}_\mathcal{P}$ given in Figure 7.2. By the argument given in Example 58, we can show that C^n_{init} can reach C^n_{fin} in the Petri net $\mathcal{N}_\mathcal{P}$ if and only if $n \geq 4$. Hence, 4 is a cut-off for $(\mathcal{N}_\mathcal{P}, \wr \mathit{init} \int, \wr \mathit{fin} \int)$ and no number less than 4 can be a cut-off.

Our main result regarding the cut-off problem is the following theorem.

Theorem 64. The cut-off problem for Petri nets is in P.

Note that this theorem proves that the cut-off problem for rendez-vous protocols is in P. We now proceed to give a sketch of the proof of Theorem 64.

7.3 The cut-off problem is in P

We now sketch the main ideas behind the proof of Theorem 64. More details regarding the proof can be found in [21, Sections 3, 4 and 5].

The proof of Theorem 64 is achieved by giving a characterization of all net systems which admit a cut-off. Moreover, this characterization can be efficiently checked in polynomial time, thanks to results from linear arithmetic theories. To state this characterization, we first need to recall the notion of a *continuous Petri net*.

Continuous Petri nets. Let $\mathcal{N} = (P, T, Pre, Post)$ be a Petri net. In addition to the usual semantics, we can equip $\mathcal{N}$ with an alternative semantics called the *continuous semantics*, in which markings are *continuous multisets*, i.e., a marking is now a function $M : P \to \mathbb{Q}_{\geq 0}$, which assigns a non-negative rational number to each place. Such markings are called *continuous markings*. The intuitive idea behind continuous semantics is that we are now allowed to split tokens and fire transitions fractionally by whichever non-zero fraction we want. More precisely, a continuous marking M enables a transition t *with factor* $\lambda \in (0, 1]$ if $M(p) \geq \lambda \cdot Pre[p, t]$ for every place p; we also say that M enables λt. If M enables λt, then λt can be fired from M, leading to a new marking M' given by $M'(p) = M(p) + \lambda \cdot \mathcal{A}[p, t]$ for every $p \in P$. We denote this by $M \xrightarrow[\mathbb{Q}]{\lambda t} M'$, and say that M' is reached from M by firing λt. A *continuous firing sequence* is any sequence of the form $\sigma = \lambda_1 t_1, \lambda_2 t_2, \ldots, \lambda_k t_k \in ((0, 1] \times T)^*$. Similar to the usual semantics, we can define the notions of $M \xrightarrow[\mathbb{Q}]{\sigma} M', M \xrightarrow[\mathbb{Q}]{} M'$ and $M \xrightarrow[\mathbb{Q}]{*} M'$.

From the definition of the continuous semantics, it follows that there exists a number $n \geq 1$ such that $nM \xrightarrow{*} nM'$ if and only if $M \xrightarrow[\mathbb{Q}]{*} M'$. Indeed, if $M \xrightarrow[\mathbb{Q}]{\sigma} M'$ for some continuous firing sequence $\sigma = \lambda_1 t_1, \lambda_2 t_2, \ldots, \lambda_k t_k$, then we can *scale* σ to a sequence $nM \xrightarrow{n\sigma} nM'$ where n is least common multiple of the denominators of all the λ_i and $n\sigma = t_1^{n\lambda_1} t_2^{n\lambda_2} t_k^{n\lambda_k}$, with each $t_i^{n\lambda_i}$ representing the sequence $\underbrace{t_i, t_i, \ldots, t_i}_{n\lambda_i \text{ times}}$. For the other direction if $nM \xrightarrow{\sigma} nM'$ holds for some $n \geq 1$ and some $\sigma = t_1, \ldots, t_k$, then $M \xrightarrow[\mathbb{Q}]{\sigma/n} M'$ is true where σ/n is the continuous firing sequence obtained by *scaling down* σ *by* n, i.e., $\sigma/n = t_1/n, t_2/n, \ldots, t_k/n$.

Example 65. Let us consider the Petri net from Figure 7.2 and equip it with the continuous semantics. We saw in Example 63, that it is not possible for the marking $\wr init \int$ to reach the marking $\wr fin \int$ under the usual semantics. However, we have $\wr 4 \cdot init \int \xrightarrow{t_1} \wr 2 \cdot init, 2 \cdot q_1 \int \xrightarrow{t_2, t_2} \wr 4 \cdot fin \int$. This run can be scaled down to get a continuous run of the form $\wr init \int \xrightarrow[\mathbb{Q}]{1/4\ t_1} \wr 1/2 \cdot init, 1/2 \cdot q_1 \int \xrightarrow[\mathbb{Q}]{1/4\ t_2,\ 1/4\ t_2} \wr fin \int$. By scaling this down even further by the fraction 1/8, we have $\wr init \int \xrightarrow[\mathbb{Q}]{*} \wr 1/2 \cdot init, 1/2 \cdot fin \int$ from which we can fire $1/2\ t_3$ to reach $\wr fin \int$. Hence, we have a continuous run from $\wr init \int$ to $\wr fin \int$, which uses all the transitions.

Unlike the case of reachability over the usual semantics, which is known to be $\mathbf{F}_\omega$-complete for Petri nets [50, 93, 92], it is known that reachability over the continuous semantics can be solved in P (Proposition 27 of [74], Theorems 3.3 and 3.6 of [39]). A crucial ingredient in the polynomial time algorithm behind reachability for continuous Petri nets is the existence of a polynomial time algorithm for linear programming, i.e., solving linear equations over $\mathbb{Q}_{\geq 0}$ (Theorem 54). Further, it is also known that there are polynomial-time fragments of the existential linear arithmetic theory of non-negative rationals in which the reachability relation of a Petri net over the continuous semantics can be defined [39, Theorem 3.6]. Hence, linear arithmetic theories play an important role in deciding the reachability relation over the continuous semantics, which in turn, plays an important role in deciding the cut-off problem for Petri nets, as we shall see now.

With the notion of a continuous Petri net defined, we are now ready to give a characterization of net systems that admit a cut-off.

Theorem 66. A Petri net system $(\mathcal{N}, M, M')$ admits a cut-off if and only if there is a continuous firing sequence σ such that $M \xrightarrow[\mathbb{Q}]{\sigma} M'$ and the marking equation has a solution $\mathbf{y}$ over the integers with the property that $[\![\mathbf{y}]\!] \subseteq [\![\sigma]\!]$.

The main ideas behind the proof of this characterization are the following.

- First, if the system admits a cut-off, then we can show that there is a number n and firing sequences τ, ζ such that $nM \xrightarrow{\tau} nM'$, $(n+1)M \xrightarrow{\zeta} (n+1)M'$ and $[\![\zeta]\!] \subseteq [\![\tau]\!]$. We can then take $\mathbf{y}$ to be $\overrightarrow{\tau} - \overrightarrow{\zeta}$ and σ to be the continuous firing sequence obtained by scaling down τ by n.

- For the other direction, first note that if $M \xrightarrow[\mathbb{Q}]{\sigma} M'$, then we can scale σ by some number n to get a firing sequence $n\sigma$ such that $nM \xrightarrow{n\sigma} nM'$. This implies that $knM \xrightarrow{*} knM'$ for every natural number $k \geq 1$.

- Then, we prove a lemma which, when given a run $M_1 \xrightarrow{\tau} M_2$ and an integral solution $\mathbf{x}$ to the marking equation for markings L_1, L_2 such that $[\![\mathbf{x}]\!] \subseteq [\![\tau]\!]$, allows us to derive a run $\mu M_1 + L_1 \xrightarrow{*} \mu M_2 + L_2$, for some number μ. We call this the *Insertion Lemma.*

- By applying the Insertion lemma to our context, we get a run of the form $(\mu n + 1)M \xrightarrow{*} (\mu n + 1)M'$. This, combined with the second step, gives us a number μ such that $\mu n M \xrightarrow{*} \mu n M'$ and $(\mu n + 1)M \xrightarrow{*} (\mu n + 1)M'$. Let $\lambda = \mu n$.

- Since every number bigger than λ^2 can be written as a linear combination of λ and $\lambda + 1$, by using the previous step, we can derive a run between kM and kM' for any $k \geq \lambda^2$. Hence λ^2 is a cut-off, which completes the proof.

Example 67. Let us consider the Petri net system from Figure 7.2 where the initial and final markings are $M := \lbag \mathit{init} \rbag$ and $M' := \lbag \mathit{fin} \rbag$ respectively. We have already seen in Example 65 that there is a continuous run from M and M' which uses all the transitions of the net. Also, notice that the vector which assigns the values 0, 0 and 1 to the transitions t_1, t_2 and t_3, respectively, is a solution to the marking equation. By Theorem 66, this system admits a cut-off.

The characterization offered by Theorem 66 almost immediately leads to a polynomial-time algorithm, obtained by invoking polynomial-time algorithms for solving the reachability relation over the continuous semantics (Proposition 27 of [74], Theorems 3.3 and 3.6 of [39]) and linear equations over the integers (Theorem 54). Hence, we can show that the cut-off problem is in P. By giving a logarithmic-space reduction from the circuit-value problem [91], we can prove that the cut-off problem is P-hard for rendez-vous protocols (and hence also for net systems). This then proves that the cut-off problems for Petri nets and rendez-vous protocols are both P-complete.

7.4 Related work

Parameterized verification of rendez-vous protocols was first considered in [76], where the authors showed a variety of results both in the case of protocols with and without a leader. By casting this model in terms of Petri nets (or alternatively vector addition system with states), the authors of [76] provide decision procedures for checking if all the executions of a process satisfy a given specification in a logic called PTL. They also show that decidability is retained even when fairness constraints are added to the model. Finally, they also present more efficient algorithms for some scenarios in the case without a leader. Since the publication of [76], this model and its extensions have been studied by other papers. (See Chapter 5 of [35] for references regarding the main results for this model and its extensions).

Population protocols [9, 10] are a model of distributed computation that is closely related to rendez-vous protocols. Intuitively, population protocols are like rendez-vous protocols, except that they also assume the existence of a fair scheduler to schedule runs of the system. This model has been analyzed using tools from formal methods and Petri nets [64, 65, 66, 67, 69, 37, 38].

7.5 Conclusion

We have shown that the cut-off problem for Petri nets is P-complete. This proves that the cut-off problem for rendez-vous protocols is in P and improves upon the previous bound of EXPSPACE. Our algorithm for the cut-off problem relied heavily on efficiently solving linear equations over $\mathbb{Q}_{\geq 0}$ and $\mathbb{Z}$.

As mentioned before, problems that were traditionally studied for rendez-vous protocols were usually concerned either about the existence of some population which satisfies some property or whether every population satisfies a given property. The cut-off problem is one of the few problems which ask if a given property is satisfied by all populations *in the limit*, i.e., whether or not the property is satisfied for all large enough populations.

It is possible to generalize the cut-off problem to deal with specifications that go beyond reachability and such "generalized" cut-off problems can be worth studying as part of future work. It would be interesting to see if linear arithmetic theories play an important role in solving such generalized cut-off problems as well.

Chapter 8

The complexity of verification of threshold automata

Threshold automata are a formalism of distributed computation that can be used to model fault-tolerant distributed algorithms. In the setting of such algorithms, we have a collection of agents executing some given protocol. Agents are allowed to be faulty; for instance, agents can either crash, or they can also behave antagonistically. However, only a specified fraction of the total number of agents is allowed to be faulty. Further, transitions between states of such protocols are labeled by *threshold guards*, which allow an agent to only make a move when it has received enough messages from some specified fraction of the total number of participating agents. For instance, a guard of the form $x \geq n/3 - t$ (where x counts the number of messages of a certain type, n is the total number of participating agents and t is the maximum number of faulty agents), specifies that the number of messages received should be bigger than $n/3 - t$ in order for an agent to proceed.

We note that in many distributed models, whether or not a step is enabled at a configuration depends only on a fixed number of processes. For instance, given two rules in a rendez-vous protocol, which respectively send and receive a message a, whether or not this pair of rules can be fired at a configuration, depends only on the existence of two processes. However, threshold guards separate threshold automata from this type of protocols because a guard might dictate that an agent can proceed only when it has received a message from a strict majority of the total number of agents. In essence, the notion of a threshold guard allows transitions to impose a *global constraint* on the configuration of the system, as opposed to other models, which impose only a local constraint.

This ability to impose global constraints makes threshold automata a popular formalism, and many papers [89, 88, 86, 87] have been dedicated to developing algorithms and implementations for its analysis. However, none of these papers have fully explored the complexity of verifying threshold automata. We fill in this gap and contribute to the study of the complexity of the main verification problems for threshold automata. We show that threshold automata are well-behaved in the sense that their reachability relation can be defined in existential Presburger arithmetic. This allows us to prove NP upper bounds for the primary verification problems related to threshold automata. Further, we also provide an implementation of our algorithms by using an SMT solver.

The rest of this chapter is structured as follows. We begin by formally defining threshold automata. Then in the next two sections, we state our main contributions regarding model-checking various classes of properties for threshold automata. Then we briefly comment on an experimental evaluation of our algorithms, discuss related work and finally conclude with a short summary.

The results of this chapter are taken from [20], which is reprinted in Appendix E.

8.1 Threshold automata

We now formally define threshold automata, mostly following the notations used in [14]. Along the way, we also illustrate the definitions on an example from [88], which is presented in Figure 8.2. This example automaton is a model of the Byzantine agreement protocol given in Figure 8.1.

Environments

Threshold automata are defined relative to an *environment* $Env = (\Pi, RC, N)$, where Π is a set of *environment variables or parameter variables* ranging over $\mathbb{N}$, $RC \subseteq \mathbb{N}^{\Pi}$ is a *resilience condition* over the environment variables, expressible as a linear formula, and $N \colon RC \to \mathbb{N}$ is a linear function called the *number function.* Intuitively, a valuation of Π determines the number of processes of different kinds (e.g., faulty) executing the protocol, and RC describes the admissible combinations of values of the environment variables. Finally, N associates to each admissible combination, the number of processes explicitly modeled. In a Byzantine setting, faulty processes behave arbitrarily, so we do not model them explicitly; in this case, the system consists of one copy of the automaton for every correct process. In the crash fault model, processes behave correctly until they crash and they must be modeled explicitly.

Example 68. In the threshold automaton of Figure 8.2, the environment variables are n, f, and t, describing the number of processes, the number of (Byzantine) faulty processes, and the maximum possible number of faulty processes, respectively. The resilience condition is the constraint $n/3 > t \geq f$. The function N is given by $N(n, t, f) = n - f$, which is the number of correct processes.

Threshold automata

A *threshold automaton* over an environment Env is a tuple $\mathsf{TA} = (\mathcal{L}, \mathcal{I}, \Gamma, \mathcal{R})$, where $\mathcal{L}$ is a finite set of *local states* (or *locations*), $\mathcal{I} \subseteq \mathcal{L}$ is the set of *initial*

```
1  var $myval_i \in \{0,1\}$
2  var $accept_i \in \{false, true\} \leftarrow false$
3
4  while true do (in one atomic step)
5    if $myval_i = 1$
6       and not sent ECHO before
7    then send ECHO to all
8
9    if received ECHO from at least
10      $t + 1$ distinct processes
11      and not sent ECHO before
12   then send ECHO to all
13
14   if received ECHO from at least
15      $n - t$ distinct processes
16   then $accept_i \leftarrow true$
17 od
```

Figure 8.1: Pseudocode of a reliable broadcast protocol from [111] for a correct process i, where n and t denote the number of processes, and an upper bound on the number of faulty processes. If $t < n/3$, the protocol satisfies its specification (if $myval_i = 0$ for every correct process i, then no correct process sets its *accept* variable to *true*).

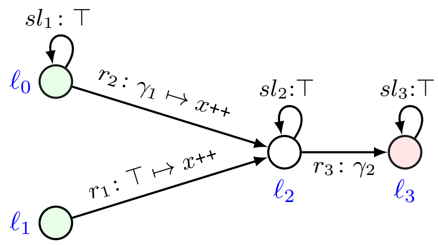

Figure 8.2: Threshold automaton from [88] modeling the body of the loop in the protocol from Fig. 8.1. Symbols γ_1, γ_2 stand for the threshold guards $x \geq (t+1) - f$ and $x \geq (n-t) - f$, where n and t are as in Fig. 8.1, and f is the actual number of faulty processes. The shared variable x models the number of ECHO messages sent by correct processes. Processes with $myval_i = b$ (line 1) start in location ℓ_b (in green). Rules r_1 and r_2 model sending ECHO at lines 7 and 12. The self-loop rules $sl_1, \ldots, sl_3$ are stuttering steps.

locations, Γ is a set of *shared variables* ranging over $\mathbb{N}$, and $\mathcal{R}$ is a set of *rules*, formally described below.

A *rule* is a tuple $r = (from, to, \varphi, \vec{u})$, where *from* and *to* are the *source* and *target* locations, $\varphi \subseteq \mathbb{N}^{\Pi \cup \Gamma}$ is a conjunction of *threshold guards* (described below), and $\vec{u} \colon \Gamma \to \{0,1\}$ is an *update*. Intuitively, r states that a process can move from *from* to *to* if the current values of Π and Γ satisfy φ, and when it moves, it updates the current valuation $\vec{g}$ of Γ by performing the update $\vec{g} := \vec{g} + \vec{u}$. Since all components of $\vec{u}$ are nonnegative, the values of shared variables never decrease. A *threshold guard* φ is a term of the following form: $b \cdot x \bowtie a_0 + a_1 \cdot p_1 + \ldots + a_{|\Pi|} \cdot p_{|\Pi|}$ where $\bowtie \in \{\geq, <\}$, $x \in \Gamma$ is a shared variable, $p_1, \ldots, p_{|\Pi|} \in \Pi$ are the environment variables, $b \in \mathbb{N}_{>0}$ and $a_0, a_1, \ldots, a_{|\Pi|} \in \mathbb{Z}$ are integer coefficients. If $b = 1$, then the guard is called a simple guard. Additionally, if $\bowtie = \geq$, then the guard is called a *rise guard*, and otherwise, the guard is called a *fall guard*.

Example 69. The rule r_2 of Figure 8.2 has ℓ_0 and ℓ_2 as source and target locations, $x \geq (t+1) - f$ as guard, and increments the value of the shared variable x by 1.

Configurations and transition relation

A *configuration* of TA is a triple $\sigma = (\vec{\boldsymbol{\kappa}}, \vec{g}, \mathbf{p})$ where $\vec{\boldsymbol{\kappa}} \colon \mathcal{L} \to \mathbb{N}$ describes the number of processes at each location, and $\vec{g} \in \mathbb{N}^{\Gamma}$ and $\mathbf{p} \in RC$ are valuations of the shared variables and the environment variables. In particular, $\sum_{\ell \in \mathcal{L}} \vec{\boldsymbol{\kappa}}(\ell) = N(\mathbf{p})$ always holds. A configuration is *initial* if $\vec{\boldsymbol{\kappa}}(\ell) = 0$ for every $\ell \notin \mathcal{I}$, and $\vec{g} = \vec{0}$. We often let $\sigma.\vec{\boldsymbol{\kappa}}, \sigma.\vec{g}, \sigma.\mathbf{p}$ denote the components of σ.

A configuration $\sigma = (\vec{\boldsymbol{\kappa}}, \vec{g}, \mathbf{p})$ *enables* a rule $r = (\mathit{from}, \mathit{to}, \varphi, \vec{u})$ if $\vec{\boldsymbol{\kappa}}(\mathit{from}) > 0$, and $(\vec{g}, \mathbf{p})$ satisfies the guard φ. If σ enables r, then TA can *move* from σ to the configuration $r(\sigma) = (\vec{\boldsymbol{\kappa}}', \vec{g}', \mathbf{p}')$ defined as follows: (i) $\mathbf{p}' = \mathbf{p}$, (ii) $\vec{g}' = \vec{g} + \vec{u}$, and (iii) $\vec{\boldsymbol{\kappa}}' = \vec{\boldsymbol{\kappa}} + \vec{v}_r$, where $\vec{v}_r = \vec{0}$ if $\mathit{from} = \mathit{to}$ and otherwise, $\vec{v}_r(\mathit{from}) = -1$, $\vec{v}_r(\mathit{to}) = +1$, and $\vec{v}_r(\ell) = 0$ for all other locations ℓ. As usual, we use $\xrightarrow{*}$ to denote the reachability relation between configurations.

8.2 Parameterized reachability and coverability

Having stated the main definitions concerning threshold automata in the previous section, we now proceed to state our contributions to the complexity of verifying threshold automata. One of our first results in this regard is concerned with the parameterized reachability problem for threshold automata, defined as follows.

Definition 70. The parameterized reachability problem for threshold automata is defined as the following decision problem.

Input:	An environment *Env*, a threshold automaton TA and two sets of locations $\mathcal{L}_{>0}, \mathcal{L}_{=0}$
Decide:	If there is a configuration σ such that σ is reachable from some initial configuration and $\sigma(\ell) > 0$ for every $\ell \in \mathcal{L}_{>0}$ and $\sigma(\ell) = 0$ for every $\ell \in \mathcal{L}_{=0}$

A special case of the parameterized reachability problem is the *parameterized coverability* problem, where $\mathcal{L}_{>0}$ is a singleton and $\mathcal{L}_{=0}$ is empty. By a reduction from the 3-SAT problem, we show that the parameterized coverability problem is NP-hard, even under various restrictions ([20, Theorem 1]). Hence, it is highly unlikely that the parameterized reachability problem admits a polynomial-time algorithm. However, we complement the hardness result by giving a tight upper bound, which enables us to prove the following theorem.

Theorem 71. Parameterized reachability is NP-complete.

This theorem is obtained by means of our following main result concerning threshold automata.

Theorem 72. Given a threshold automaton TA, there is an existential Presburger formula ϕ_{reach} such that $\phi_{reach}(\sigma, \sigma')$ holds if and only if $\sigma \xrightarrow{*} \sigma'$.

We now present the main ideas behind this theorem. More details can be found in [20, Section 4].

As a first step to proving this theorem, we show that there is an existential Presburger formula which can capture the *steady reachability relation* of a given threshold automaton, which is defined as follows. We say that there is a steady run between two configurations σ and σ' if there is a run between σ and σ' such that along any configuration in this run, the set of rise guards that evaluate to true and the set of fall guards that evaluate to false remain the same.

The formula $\phi_{steady}(\sigma, \sigma')$ capturing the steady reachability relation is constructed as follows. For every rule $r \in \mathcal{R}$, we let x_r be an existential variable ranging over $\mathbb{N}$. Intuitively, the value of x_r will represent the number of times r is fired during the (supposed) steady run from σ to σ'. ϕ_{steady} then expresses a certain number of conditions that must be necessarily satisfied in order for there to be a steady run between σ and σ'. For example, one of the conditions expressed by ϕ_{steady} is that the values of σ and σ' are consistent with the net effect of firing each rule r exactly x_r many times. This condition is, in some sense, similar to the marking equation for Petri nets.

We show that all the conditions expressed by ϕ_{steady} are necessary and sufficient for the existence of a steady run between two configurations. Having proved this, we use an existing result of the theory of threshold automata to construct a formula for the general reachability relation. More specifically, it is known that a run of a threshold automaton can always be decomposed into a polynomial number of single steps and steady runs. This result, combined with the formula for the steady reachability relation, allows us to derive a formula for the general reachability relation.

8.3 Parameterized safety and liveness

We now use the result that the reachability relation of threshold automata is definable in existential Presburger arithmetic to provide algorithms for model-checking certain safety and liveness properties for threshold automata. To this end, we recall the definition of *Fault-Tolerant Temporal Logic* ($\mathsf{ELTL_{FT}}$), which is the fragment of LTL used in [88] to specify and verify properties of a large number of fault-tolerant distributed algorithms. Given a threshold automaton $\mathsf{TA} = (\mathcal{L}, \mathcal{I}, \Gamma, \mathcal{R})$ whose set of guards is Φ, formulas of $\mathsf{ELTL_{FT}}$ over TA have the following syntax, where $S \subseteq \mathcal{L}$ is a set of locations and $guard \in \Phi$ is a guard:

$$\begin{array}{ll} \psi ::= pf \mid \mathsf{G}\,\psi \mid \mathsf{F}\,\psi \mid \psi \wedge \psi & \qquad cf ::= S = 0 \mid \neg(S = 0) \mid cf \wedge cf \\ pf ::= cf \mid gf \Rightarrow cf & \qquad gf ::= guard \mid gf \wedge gf \mid gf \vee gf \end{array}$$

A configuration σ satisfies $S = 0$, if no agent in σ is present in any of the locations in S and it satisfies *guard*, if the values of the shared and environment variables of σ satisfy *guard*. The rest of the semantics is standard. The negations of specifications of the benchmarks [43, 111, 40, 97, 102, 77, 59, 41, 110] can be expressed in $\mathsf{ELTL_{FT}}$, as we are interested in finding possible violations.

We note that $\mathsf{ELTL_{FT}}$ can be thought of as a fragment of LTL with F and G operators and limited negation and disjunction; for example, it cannot express the property "It is always the case that either there is no agent at ℓ_1 or there is no agent at ℓ_2". Nevertheless, it is a useful fragment which can express some interesting properties.

Example 73. One specification of the algorithm from Figure 8.1 is that if $myval_i = 1$ for every correct process i, then eventually $accept_j = true$ for some correct process j. In the words of the automaton from Figure 8.2, a violation of this property would mean that initially, all correct processes are in location ℓ_1, but no correct process ever reaches location ℓ_3. In $\mathsf{ELTL_{FT}}$ we write this as

$$\{\ell_0, \ell_2, \ell_3\} = 0 \ \wedge\ \mathsf{G}\,(\{\ell_3\} = 0)$$

This has to hold under the following fairness constraint, which ensures that if an outgoing rule for a process from some location is enabled infinitely often, then that process fires that rule at some point and moves out of that location.

$$\mathsf{G}\,\mathsf{F}\left((x \geq t+1 \Rightarrow \{\ell_0\}{=}0) \ \wedge\ \{\ell_1\}{=}0 \ \wedge\ (x \geq n-t \Rightarrow \{\ell_2\}{=}0) \right)$$

It is known that model-checking LTL with F and G operators for threshold automata is undecidable [88]. Hence, it makes sense to study the model-checking problem of threshold automata against $\mathsf{ELTL_{FT}}$ specifications, formalized as the following problem.

Definition 74. The model-checking problem for threshold automata against $\mathsf{ELTL_{FT}}$ specifications is defined as the following decision problem.

Input:	An environment *Env*, a threshold automaton TA and a formula φ in $\mathsf{ELTL_{FT}}$
Decide:	If there is an initial configuration σ_0 and an infinite path τ from σ_0 which satisfies φ

Note that since the model-checking problem encompasses the parameterized coverability problem, it is NP-hard. We show that for threshold automata with *multplicative environments*, the model-checking problem is in NP. We now proceed to define the notion of a multiplicative environment.

Definition 75. An environment $Env = (\Pi, RC, N)$ is *multiplicative* for a threshold automaton TA if every fall guard is simple and for every $\mu \in \mathbb{N}_{>0}$ (i) for every rational valuation $\mathbf{p} \in RC$ we have $\mu \cdot \mathbf{p} \in RC$ and $N(\mu \cdot \mathbf{p}) = \mu \cdot N(\mathbf{p})$, and (ii) for every guard $\varphi := b \cdot x \bowtie a_0 + a_1 p_1 + a_2 p_2 + \dots a_k p_k$ in TA, if $(y, q_1, q_2, \dots, q_k)$ is a rational solution to φ then $(\mu \cdot y, \mu \cdot q_1, \dots, \mu \cdot q_k)$ is also a solution to φ.

Example 76. In Figure 8.2, if the resilience condition $t < n/3$ holds for a pair (n, t), then it also holds for $(\mu \cdot n, \mu \cdot t)$; similarly, the function $N(n, t, f) = n - f$ also satisfies $N(\mu \cdot n, \mu \cdot t, \mu \cdot f) = \mu \cdot n - \mu \cdot f = \mu \cdot N(n, t, f)$. Moreover, if $x \geq t + 1 - f$ holds in σ, then we also have $\mu \cdot x \geq \mu \cdot t + 1 - \mu \cdot f$ in $\mu \cdot \sigma$. A similar claim also holds for the other guard $x \geq n - t - f$.

The multiplicative property allows us to reason about multiplied paths in large systems. Namely, condition (ii) from Definition 75 yields that if a rule is enabled in $\sigma = (\vec{\boldsymbol{\kappa}}, \vec{g}, \mathbf{p})$, it is also enabled in $\mu \cdot \sigma = (\mu \cdot \vec{\boldsymbol{\kappa}}, \mu \cdot \vec{g}, \mu \cdot \mathbf{p})$. We exploit this property to show that if a counterexample exists in a small system, then a counterexample also exists in a large system. This enables us to prove that

Theorem 77. Model-checking $\mathsf{ELTL}_{\mathsf{FT}}$ specifications is in NP for threshold automata over multiplicative environments.

8.4 Experiments

We implemented the techniques presented in this chapter to verify a number of fault-tolerant distributed algorithms modeled as threshold automata. Our benchmarks include automata derived from various distributed algorithms in the literature [111, 43, 40, 102, 77, 97, 59, 41, 110]. We used Z3 as a back-end SMT solver for solving constraints given in existential Presburger arithmetic. A practical evaluation of our algorithm and its comparison with an existing tool can be found in [20, Section 7].

8.5 Related work

As mentioned at the start of this chapter, various algorithms have been developed for analyzing and verifying threshold automata [89, 88, 86, 87]. The papers [89, 86] developed algorithms for solving the reachability problem using bounded model-checking and SAT/SMT solvers. These papers show that bounded model-checking is complete for reachability properties and also provide some experimental results. An algorithm for model-checking against $\mathsf{ELTL}_{\mathsf{FT}}$ specifications was proposed in [88], along with experiments on a set of benchmarks. The paper [87] discusses the ByMC tool for model-checking threshold automata. We refer the interested reader to [85] for a survey of the main results concerning threshold automata.

There are some extensions and variants of threshold automata that have been proposed in the literature. The paper [90] extends threshold automata in various ways (for instance, by allowing non-linear guards) and proves decidability and undecidability results for different extensions. Threshold automata extended with probabilities have been used to study randomized distributed algorithms [33, 34]. Synchronous threshold automata are considered in [113, 112] to model synchronous distributed algorithms.

Finally, as an explicit application, threshold automata have also been used to model blockchains [32], where the authors verified a blockchain consensus algorithm using tools developed for threshold automata.

8.6 Conclusion

We have studied the complexity of the fundamental verification problems for threshold automata. Our results have established upper bounds for some problems pertaining to this model. A crucial tool for our results is the fact that the reachability relation for threshold automata is definable in existential Presburger arithmetic.

One of the results that we have shown is that the model-checking problem for $\mathsf{ELTL_{FT}}$ specifications against multiplicative threshold automata is in NP. As part of future work, it might be interesting to extend this result to the entire class of threshold automata.

Chapter 9

The parameterized complexity of safety of threshold automata

Complexity theory is primarily concerned with studying some measure of complexity of a problem which depends *solely* on the input size n. In some cases, this can be too restrictive, as it does not consider other measures of the input. *Parameterized complexity* is a field of complexity theory that attempts to study decision problems that come along with a *parameter* [49]. In parameterized complexity, apart from the size of the input n, one considers further parameters k that capture the structure of the input and one looks for algorithms that run in time $f(k) \cdot n^{O(1)}$, where f is some function dependent on k alone. The hope is to find parameters that are quite small in practice and to base the dominant running time of the algorithm on this parameter alone. Problems solvable in such a manner are called fixed-parameter tractable (FPT).

In this chapter, we focus on studying the safety verification problem for threshold automata, or dually the reachability problem for threshold automata, from the lens of parameterized complexity. In Section 8.2 of the previous chapter, we showed that this problem is NP-complete. Here, we refine this result by considering the same problem with parameters that are usually small in practice. Our contributions include both hardness and tractability results, as well as an implementation of our techniques.

The results of this chapter are taken from [14], which is reprinted in Appendix F. That paper also has results on a special case of threshold automata called acyclic threshold automata. Since the focus here is on presenting the main results of [14], we have not considered that special case here. For more details on this special case, we refer the reader to Section 4 of [14].

The rest of this chapter is structured as follows. We begin by formally defining the concepts of parameterized complexity that we shall use. Then, in the next two sections, we state our main results and give a sketch of our ideas. Finally, we briefly comment on some experiments, discuss related work and conclude with a short summary.

9.1 Preliminaries

Parameterized complexity

We now formally define the notion of fixed-parameter tractable algorithms. We refer the reader to [49] for more information on parameterized complexity and only give the necessary definitions here. A *parameterized problem* L is a subset of $\Sigma^* \times \mathbb{N}$ for some alphabet Σ. A parameterized problem L is said to be *fixed-parameter tractable* (FPT) if there exists an algorithm A such that $(x, k) \in L$ if and only if $A(x, k)$ is true and A runs in time $f(k) \cdot n^{O(1)}$ where $n = |x|$ is the length of x and f is some computable function, depending only on the *parameter* k. FPT is roughly the analog of P in parameterized complexity.

Given parameterized problems $L, L' \subseteq \Sigma^* \times \mathbb{N}$ we say that L is reducible to L' if there is an algorithm that, given an input (x, k), produces another input (x', k') in time $f(k) \cdot |x|^{O(1)}$ such that $(x, k) \in L \iff (x', k') \in L'$ and $k' \leq g(k)$ for some functions f and g depending only on k.

The parameterized clique problem is the set of all pairs (G, k) such that the graph G has a clique of size k, i.e., there is a subset S of vertices of G of size k such that there is an edge between every two vertices in S. A parameterized problem L is said to be W[1]-hard if there is a parameterized reduction from the parameterized clique problem to L. If L is W[1]-hard and there is a parameterized reduction from L to L' then L' is W[1]-hard as well. W[1]-hardness is roughly the analog of NP-hardness in parameterized complexity, and if a problem is W[1]-hard, it is usually taken to be evidence that the problem does not have an FPT algorithm.

Coverability and reachability

In this chapter, we shall concern ourselves with the parameterized coverability and parameterized reachability problems for threshold automata, which we shall refer to as the coverability and the reachability problems here, in order to avoid confusion with the different notion of parameterized complexity. We study both these problems with some combinations of parameters.

Let $\mathsf{TA} = (\mathcal{L}, \mathcal{I}, \Gamma, \mathcal{R})$ be a threshold automaton over an environment $Env = (\Pi, RC, N)$ and let $\mathcal{L}_{>0}$ and $\mathcal{L}_{=0}$ be two sets of locations. In the sequel, we let $\mathcal{L}_{\mathsf{spec}}$ denote the set $\mathcal{L}_{>0} \cup \mathcal{L}_{=0}$. Recall that the reachability problem is to decide if there is a configuration σ reachable from some initial configuration such that $\sigma(\ell) > 0$ for every $\ell \in \mathcal{L}_{>0}$ and $\sigma(\ell) = 0$ for every $\ell \in \mathcal{L}_{=0}$. The coverability problem is the special case of the reachability problem where $\mathcal{L}_{>0}$ is a singleton set and $\mathcal{L}_{=0}$ is empty. The next two sections describe our main contributions to these two problems from the perspective of parameterized complexity.

9.2 Hardness of coverability

Our first result regarding the parameterized complexity of verifying threshold automata is a hardness result, which states that the coverability and reachability problems are hard even when parameterized by parameters that are usually small in practice. More specifically, we consider the coverability problem parameterized by the following parameters: $|\Phi|$ (the number of distinct guards), $|\mathcal{L}_{\mathsf{spec}}|$ (the size of the specification), $|RC|$ (the number of constraints in the resilience condition) and C (the maximum constant appearing in any of the guards of TA). In practice, all these values are quite small, roughly in the range of 10 to 25. Hence it would be desirable to obtain an FPT algorithm for coverability or reachability when parameterized by $|\Phi| + |\mathcal{L}_{\mathsf{spec}}| + |RC| + C$. However, we prove the following hardness result regarding the coverability problem (See [14, Theorem 1]).

Theorem 78. Coverability (and hence reachability) for threshold automata parameterized by $|\Phi| + |\mathcal{L}_{\mathsf{spec}}| + |RC| + C$ is $W[1]$-hard, even when $|\Phi^{\text{fall}}|$ is a constant.

This hardness result shows that even basic verification problems for threshold automata are hard in the parameterized complexity landscape. However, as we explain in the next section, it turns out that with some restrictions on the underlying automaton, we can obtain a tractability result.

9.3 Multiplicative threshold automata with constantly many fall guards

For our main tractability result, we consider threshold automata in which the number of fall guards is a constant. It turns out that for many of the automata occurring in practice, the number of fall guards is at most one. Notice that our hardness result already applies to automata with constantly many fall guards. Hence, to get tractable results, we need to constrain this class further. To this end, we consider *multiplicative* threshold automata (See Definition 75) with constantly many fall guards. Our main result regarding this class is the following theorem.

Theorem 79. The reachability problem for multiplicative threshold automata with constantly many fall guards is fixed-parameter tractable when parameterized by the total number of guards $|\Phi|$.

We now provide a sketch of the proof of this theorem. More details behind the proof can be found in [14, Section 5].

The proof of this theorem is established in four stages. In the first stage, we decompose every run of a threshold automaton into a concatenation of at most $|\Phi| + 1$ *steady runs* and single steps. We have already used a similar decomposition in Section 8.2 to establish the NP upper bound for reachability of threshold automata, but we modify it slightly here to work with sets of rules instead of sets of guards. In the second and most important stage, we forge a connection between multiplicative threshold automata and *continuous Petri nets*. Namely, we construct a (continuous) Petri net that correctly simulates the firing of every rule, but it does not check that the guards are true. We remedy this situation by only considering executions in the Petri net corresponding to steady runs of the automaton. We show that if there is a steady run between σ and σ' in the automaton, then it also has a corresponding run in the continuous Petri net. Conversely, if there is a run between two continuous markings M and M' of the Petri net which satisfies a few constraints, then this run can be lifted to a steady run between configurations μM and $\mu M'$ in the automaton where μ is some natural number such that μM and $\mu M'$ are integral.

In the third stage, we use the fact that there is a logic characterizing reachability in continuous Petri nets, which can be tested for satisfiability in polynomial time [39, Theorems 3 and 6] to show that for steady runs and single steps, we can efficiently decide reachability between configurations of a multiplicative threshold automaton. In the final stage, we use the fact that the number of possible decompositions of a path into steady runs and single steps in a threshold automaton with constantly many fall guards is of the form $f(|\Phi|) \cdot n^{O(1)}$, where n is the size of the input. Hence, once we have guessed such a decomposition, we can efficiently decide reachability, thanks to the third stage. This then proves the desired result.

9.4 Experiments

We have implemented the algorithms given in this chapter on threshold automata derived from various distributed algorithms in the literature. Similar to the results in the previous chapter, we once again used Z3 to solve constraints. An experimental evaluation of our algorithm can be found in [14, Section 7].

9.5 Related work

There has been a lot of work on studying the parameterized complexity of graph problems, which has resulted in a wide array of techniques for proving both algorithmic as well as hardness results. We refer the interested reader to [49] for more details on this subject.

Over the years, the analysis of the parameterized complexity of problems in verification has gathered some attention [46, 45, 60, 70, 44]. To the best of our knowledge, no results on the parameterized complexity of threshold automata was known before our work. Our results contribute to a step in this direction.

9.6 Conclusion

We have studied the parameterized complexity of reachability for threshold automata and established both hardness and tractability results. One of the main ingredients that we used for our tractability results is the notion of a continuous Petri net, whose reachability problem is efficiently decidable, thanks to its reliance on linear programming over the rationals.

In this work, we have only concentrated on some of the parameters of a threshold automaton which are usually small in practice. A more systematic classification of hardness and tractability results based on different combinations of parameters might be an interesting avenue for future work and might help to pinpoint the precise factors responsible for the hardness of verifying threshold automata.

Chapter 10

Summary and Outlook

The first half of our thesis covered two of our contributions to the theory of parameterized systems. These contributions were focused on using the framework of well-structured transition systems to prove results for parameterized verification. As part of our first contribution, we provided new tools for proving upper bounds for the coverability algorithm for well-structured systems, and we showed their applicability on some classes of parameterized systems. As part of our second contribution, we proved that some problems related to parameterized verification are complete for classes in the fast-growing hierarchy.

We believe that the techniques presented for proving upper bounds can be potentially applied to areas beyond the ones considered in this thesis. We justified this claim by already mentioning applications of our results in logic and automata theory. We also hope that the techniques used for proving lower bounds can be transferred to other well-structured systems which have an underlying notion of *bounded depth*.

The second half of our thesis covered our remaining two contributions. These contributions were focused on using the framework of linear arithmetic theories to prove results for parameterized verification. Our results in this part firmly establish that linear arithmetic theories can prove to be a useful tool to obtain both *complete and efficient* algorithms for the analysis of parameterized systems.

An important ingredient that we used to provide tractable algorithms in the second part is the *continuous semantics* for Petri nets. It is possible to define an appropriate notion of this semantics for any class of counter systems working over the natural numbers. Similar to the case of Petri nets, it can happen that this semantics becomes a tractable overapproximation for many classes of counter systems. We believe that the continuous semantics will prove to be a useful tool in the analysis of parameterized systems.

In summary, we have used two tools, namely well-structured transition systems and linear arithmetic theories, to answer complexity-theoretic questions for a variety of parameterized systems. We hope that some of the techniques from this thesis will help provide better algorithms in the future for analyzing parameterized systems.

Bibliography

[1] Parosh Aziz Abdulla, Karlis Cerans, Bengt Jonsson, and Yih-Kuen Tsay. General decidability theorems for infinite-state systems. In *Proceedings, 11th Annual IEEE Symposium on Logic in Computer Science, New Brunswick, New Jersey, USA, July 27-30, 1996*, pages 313–321. IEEE Computer Society, 1996. `doi:10.1109/LICS.1996.561359`.

[2] Parosh Aziz Abdulla, Giorgio Delzanno, Othmane Rezine, Arnaud Sangnier, and Riccardo Traverso. On the verification of timed ad hoc networks. In Uli Fahrenberg and Stavros Tripakis, editors, *Formal Modeling and Analysis of Timed Systems - 9th International Conference, FORMATS 2011, Aalborg, Denmark, September 21-23, 2011. Proceedings*, volume 6919 of *Lecture Notes in Computer Science*, pages 256–270. Springer, 2011. `doi:10.1007/978-3-642-24310-3\_18`.

[3] Parosh Aziz Abdulla and Bengt Jonsson. Model checking of systems with many identical timed processes. *Theor. Comput. Sci.*, 290(1):241–264, 2003. `doi:10.1016/S0304-3975(01)00330-9`.

[4] Parosh Aziz Abdulla and Aletta Nylén. Timed Petri nets and BQOs. In José Manuel Colom and Maciej Koutny, editors, *Application and Theory of Petri Nets 2001, 22nd International Conference, ICATPN 2001, Newcastle upon Tyne, UK, June 25-29, 2001, Proceedings*, volume 2075 of *Lecture Notes in Computer Science*, pages 53–70. Springer, 2001. `doi:10.1007/3-540-45740-2\_5`.

[5] Sergio Abriola and Santiago Figueira. A note on the order type of minoring orderings and some algebraic properties of ω^2-well quasi-orderings. In *XL Latin American Computing Conference, CLEI 2014, Montevideo, Uruguay, September 15-19, 2014*, pages 1–9. IEEE, 2014. `doi:10.1109/CLEI.2014.6965188`.

[6] Sergio Abriola, Santiago Figueira, and Gabriel Senno. Linearizing bad sequences: Upper bounds for the product and majoring well quasi-orders. In C.-H. Luke Ong and Ruy J. G. B. de Queiroz, editors, *Logic, Language, Information and Computation - 19th International Workshop, WoLLIC 2012, Buenos Aires, Argentina, September 3-6, 2012. Proceedings*, volume

7456 of *Lecture Notes in Computer Science*, pages 110–126. Springer, 2012. doi:10.1007/978-3-642-32621-9_9.

[7] Sergio Abriola, Santiago Figueira, and Gabriel Senno. Linearizing well quasi-orders and bounding the length of bad sequences. *Theor. Comput. Sci.*, 603:3–22, 2015. doi:10.1016/j.tcs.2015.07.012.

[8] Benjamin Aminof, Sasha Rubin, Florian Zuleger, and Francesco Spegni. Liveness of parameterized timed networks. In Magnús M. Halldórsson, Kazuo Iwama, Naoki Kobayashi, and Bettina Speckmann, editors, *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part II*, volume 9135 of *Lecture Notes in Computer Science*, pages 375–387. Springer, 2015. doi:10.1007/978-3-662-47666-6_30.

[9] Dana Angluin, James Aspnes, Zoë Diamadi, Michael J. Fischer, and René Peralta. Computation in networks of passively mobile finite-state sensors. *Distributed Comput.*, 18(4):235–253, 2006. doi:10.1007/s00446-005-0138-3.

[10] Dana Angluin, James Aspnes, David Eisenstat, and Eric Ruppert. The computational power of population protocols. *Distributed Comput.*, 20(4):279–304, 2007. doi:10.1007/s00446-007-0040-2.

[11] Krzysztof R. Apt and Dexter Kozen. Limits for automatic verification of finite-state concurrent systems. *Inf. Process. Lett.*, 22(6):307–309, 1986. doi:10.1016/0020-0190(86)90071-2.

[12] A. R. Balasubramanian. Parameterized verification of coverability in well-structured broadcast networks. In Andrea Orlandini and Martin Zimmermann, editors, *Proceedings Ninth International Symposium on Games, Automata, Logics, and Formal Verification, GandALF 2018, Saarbrücken, Germany, 26-28th September 2018*, volume 277 of *EPTCS*, pages 133–146, 2018. doi:10.4204/EPTCS.277.10.

[13] A. R. Balasubramanian. Complexity of controlled bad sequences over finite sets of $\mathbb{N}^d$. In Holger Hermanns, Lijun Zhang, Naoki Kobayashi, and Dale Miller, editors, *LICS '20: 35th Annual ACM/IEEE Symposium on Logic in Computer Science, Saarbrücken, Germany, July 8-11, 2020*, pages 130–140. ACM, 2020. doi:10.1145/3373718.3394753.

[14] A. R. Balasubramanian. Parameterized complexity of safety of threshold automata. In Nitin Saxena and Sunil Simon, editors, *40th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2020, December 14-18, 2020, BITS Pilani, K K Birla Goa Campus, Goa, India (Virtual Conference)*, volume 182 of *LIPIcs*, pages 37:1–37:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.FSTTCS.2020.37.

[15] A. R. Balasubramanian. Complexity of coverability in bounded path broadcast networks. In Mikolaj Bojanczyk and Chandra Chekuri, editors, *41st IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2021, December 15-17, 2021, Virtual Conference*, volume 213 of *LIPIcs*, pages 35:1–35:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. `doi:10.4230/LIPIcs.FSTTCS.2021.35`.

[16] A. R. Balasubramanian. Parameterized verification of coverability in infinite state broadcast networks. *Inf. Comput.*, 278:104592, 2021. `doi:10.1016/j.ic.2020.104592`.

[17] A. R. Balasubramanian. Coefficient synthesis for threshold automata. In Anthony W. Lin, Georg Zetzsche, and Igor Potapov, editors, *Reachability Problems - 16th International Conference, RP 2022, Kaiserslautern, Germany, October 17-21, 2022, Proceedings*, volume 13608 of *Lecture Notes in Computer Science*, pages 125–139. Springer, 2022. `doi:10.1007/978-3-031-19135-0\_9`.

[18] A. R. Balasubramanian. Complexity of coverability in depth-bounded processes. In Bartek Klin, Slawomir Lasota, and Anca Muscholl, editors, *33rd International Conference on Concurrency Theory, CONCUR 2022, September 12-16, 2022, Warsaw, Poland*, volume 243 of *LIPIcs*, pages 17:1–17:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. `doi:10.4230/LIPIcs.CONCUR.2022.17`.

[19] A. R. Balasubramanian, Nathalie Bertrand, and Nicolas Markey. Parameterized verification of synchronization in constrained reconfigurable broadcast networks. In Dirk Beyer and Marieke Huisman, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 24th International Conference, TACAS 2018, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2018, Thessaloniki, Greece, April 14-20, 2018, Proceedings, Part II*, volume 10806 of *Lecture Notes in Computer Science*, pages 38–54. Springer, 2018. `doi:10.1007/978-3-319-89963-3\_3`.

[20] A. R. Balasubramanian, Javier Esparza, and Marijana Lazić. Complexity of verification and synthesis of threshold automata. In Dang Van Hung and Oleg Sokolsky, editors, *Automated Technology for Verification and Analysis - 18th International Symposium, ATVA 2020, Hanoi, Vietnam, October 19-23, 2020, Proceedings*, volume 12302 of *Lecture Notes in Computer Science*, pages 144–160. Springer, 2020. `doi:10.1007/978-3-030-59152-6\_8`.

[21] A. R. Balasubramanian, Javier Esparza, and Mikhail A. Raskin. Finding cut-offs in leaderless rendez-vous protocols is easy. In Stefan Kiefer and Christine Tasson, editors, *Foundations of Software Science and Computation Structures - 24th International Conference, FOSSACS 2021, Held as*

Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2021, Luxembourg City, Luxembourg, March 27 - April 1, 2021, Proceedings, volume 12650 of *Lecture Notes in Computer Science*, pages 42–61. Springer, 2021. `doi:10.1007/978-3-030-71995-1\_3`.

[22] A. R. Balasubramanian, Lucie Guillou, and Chana Weil-Kennedy. Parameterized analysis of reconfigurable broadcast networks. In Patricia Bouyer and Lutz Schröder, editors, *Foundations of Software Science and Computation Structures - 25th International Conference, FOSSACS 2022, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2022, Munich, Germany, April 2-7, 2022, Proceedings*, volume 13242 of *Lecture Notes in Computer Science*, pages 61–80. Springer, 2022. `doi:10.1007/978-3-030-99253-8\_4`.

[23] A. R. Balasubramanian, Timo Lang, and Revantha Ramanayake. Decidability and complexity in weakening and contraction hypersequent substructural logics. In *36th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2021, Rome, Italy, June 29 - July 2, 2021*, pages 1–13. IEEE, 2021. `doi:10.1109/LICS52264.2021.9470733`.

[24] A. R. Balasubramanian and K. S. Thejaswini. Adaptive synchronisation of pushdown automata. In Serge Haddad and Daniele Varacca, editors, *32nd International Conference on Concurrency Theory, CONCUR 2021, August 24-27, 2021, Virtual Conference*, volume 203 of *LIPIcs*, pages 17:1–17:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. `doi:10.4230/LIPIcs.CONCUR.2021.17`.

[25] A. R. Balasubramanian and Igor Walukiewicz. Characterizing consensus in the Heard-Of model. In Igor Konnov and Laura Kovács, editors, *31st International Conference on Concurrency Theory, CONCUR 2020, September 1-4, 2020, Vienna, Austria (Virtual Conference)*, volume 171 of *LIPIcs*, pages 9:1–9:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. `doi:10.4230/LIPIcs.CONCUR.2020.9`.

[26] A. R. Balasubramanian and Chana Weil-Kennedy. Reconfigurable broadcast networks and asynchronous shared-memory systems are equivalent. In Pierre Ganty and Davide Bresolin, editors, *Proceedings 12th International Symposium on Games, Automata, Logics, and Formal Verification, GandALF 2021, Padua, Italy, 20-22 September 2021*, volume 346 of *EPTCS*, pages 18–34, 2021. `doi:10.4204/EPTCS.346.2`.

[27] Kshitij Bansal, Eric Koskinen, Thomas Wies, and Damien Zufferey. Structural counter abstraction. In Nir Piterman and Scott A. Smolka, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 19th International Conference, TACAS 2013, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2013, Rome, Italy, March 16-24, 2013. Proceedings*, volume

7795 of *Lecture Notes in Computer Science*, pages 62–77. Springer, 2013. doi:10.1007/978-3-642-36742-7_5.

[28] Nathalie Bertrand, Patricia Bouyer, and Anirban Majumdar. Reconfiguration and message losses in parameterized broadcast networks. *Log. Methods Comput. Sci.*, 17(1), 2021. URL: https://lmcs.episciences.org/7280.

[29] Nathalie Bertrand and Paulin Fournier. Parameterized verification of many identical probabilistic timed processes. In Anil Seth and Nisheeth K. Vishnoi, editors, *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2013, December 12-14, 2013, Guwahati, India*, volume 24 of *LIPIcs*, pages 501–513. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2013. doi:10.4230/LIPIcs.FSTTCS.2013.501.

[30] Nathalie Bertrand, Paulin Fournier, and Arnaud Sangnier. Playing with probabilities in reconfigurable broadcast networks. In Anca Muscholl, editor, *Foundations of Software Science and Computation Structures - 17th International Conference, FOSSACS 2014, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2014, Grenoble, France, April 5-13, 2014, Proceedings*, volume 8412 of *Lecture Notes in Computer Science*, pages 134–148. Springer, 2014. doi:10.1007/978-3-642-54830-7_9.

[31] Nathalie Bertrand, Paulin Fournier, and Arnaud Sangnier. Distributed local strategies in broadcast networks. In Luca Aceto and David de Frutos-Escrig, editors, *26th International Conference on Concurrency Theory, CONCUR 2015, Madrid, Spain, September 1.4, 2015*, volume 42 of *LIPIcs*, pages 44–57. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2015. doi:10.4230/LIPIcs.CONCUR.2015.44.

[32] Nathalie Bertrand, Vincent Gramoli, Igor Konnov, Marijana Lazić, Pierre Tholoniat, and Josef Widder. Holistic verification of blockchain consensus. In Christian Scheideler, editor, *36th International Symposium on Distributed Computing, DISC 2022, October 25-27, 2022, Augusta, Georgia, USA*, volume 246 of *LIPIcs*, pages 10:1–10:24. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.DISC.2022.10.

[33] Nathalie Bertrand, Igor Konnov, Marijana Lazic, and Josef Widder. Verification of randomized consensus algorithms under round-rigid adversaries. *Int. J. Softw. Tools Technol. Transf.*, 23(5):797–821, 2021. doi:10.1007/s10009-020-00603-x.

[34] Nathalie Bertrand, Marijana Lazic, and Josef Widder. A reduction theorem for randomized distributed algorithms under weak adversaries. In Fritz Henglein, Sharon Shoham, and Yakir Vizel, editors, *Verification,*

Model Checking, and Abstract Interpretation - 22nd International Conference, VMCAI 2021, Copenhagen, Denmark, January 17-19, 2021, Proceedings, volume 12597 of *Lecture Notes in Computer Science*, pages 219–239. Springer, 2021. doi:10.1007/978-3-030-67067-2_11.

[35] Roderick Bloem, Swen Jacobs, Ayrat Khalimov, Igor Konnov, Sasha Rubin, Helmut Veith, and Josef Widder. *Decidability of Parameterized Verification.* Synthesis Lectures on Distributed Computing Theory. Morgan & Claypool Publishers, 2015. doi:10.2200/S00658ED1V01Y201508DCT013.

[36] Michael Blondin, Javier Esparza, Martin Helfrich, Antonín Kucera, and Philipp J. Meyer. Checking qualitative liveness properties of replicated systems with stochastic scheduling. In Shuvendu K. Lahiri and Chao Wang, editors, *Computer Aided Verification - 32nd International Conference, CAV 2020, Los Angeles, CA, USA, July 21-24, 2020, Proceedings, Part II*, volume 12225 of *Lecture Notes in Computer Science*, pages 372–397. Springer, 2020. doi:10.1007/978-3-030-53291-8_20.

[37] Michael Blondin, Javier Esparza, Stefan Jaax, and Antonín Kucera. Black ninjas in the dark: Formal analysis of population protocols. In Anuj Dawar and Erich Grädel, editors, *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018*, pages 1–10. ACM, 2018. doi:10.1145/3209108.3209110.

[38] Michael Blondin, Javier Esparza, Stefan Jaax, and Philipp J. Meyer. Towards efficient verification of population protocols. *Formal Methods Syst. Des.*, 57(3):305–342, 2021. doi:10.1007/s10703-021-00367-3.

[39] Michael Blondin and Christoph Haase. Logics for continuous reachability in Petri nets and vector addition systems with states. In *32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, June 20-23, 2017*, pages 1–12. IEEE Computer Society, 2017. doi:10.1109/LICS.2017.8005068.

[40] Gabriel Bracha and Sam Toueg. Asynchronous consensus and broadcast protocols. *J. ACM*, 32(4):824–840, 1985. doi:10.1145/4221.214134.

[41] Francisco Vilar Brasileiro, Fabíola Greve, Achour Mostéfaoui, and Michel Raynal. Consensus in one communication step. In Victor E. Malyshkin, editor, *Parallel Computing Technologies, 6th International Conference, PaCT 2001, Novosibirsk, Russia, September 3-7, 2001, Proceedings*, volume 2127 of *Lecture Notes in Computer Science*, pages 42–50. Springer, 2001. doi:10.1007/3-540-44743-1_4.

[42] Pierre Chambart and Philippe Schnoebelen. The ordinal recursive complexity of lossy channel systems. In *Proceedings of the Twenty-Third Annual IEEE Symposium on Logic in Computer Science, LICS 2008, 24-27 June 2008, Pittsburgh, PA, USA*, pages 205–216. IEEE Computer Society, 2008. doi:10.1109/LICS.2008.47.

[43] Tushar Deepak Chandra and Sam Toueg. Unreliable failure detectors for reliable distributed systems. *J. ACM*, 43(2):225–267, 1996. `doi:10.1145/226643.226647`.

[44] Peter Chini, Jonathan Kolberg, Andreas Krebs, Roland Meyer, and Prakash Saivasan. On the complexity of bounded context switching. In Kirk Pruhs and Christian Sohler, editors, *25th Annual European Symposium on Algorithms, ESA 2017, September 4-6, 2017, Vienna, Austria*, volume 87 of *LIPIcs*, pages 27:1–27:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. `doi:10.4230/LIPIcs.ESA.2017.27`.

[45] Peter Chini, Roland Meyer, and Prakash Saivasan. Complexity of liveness in parameterized systems. In Arkadev Chattopadhyay and Paul Gastin, editors, *39th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2019, December 11-13, 2019, Bombay, India*, volume 150 of *LIPIcs*, pages 37:1–37:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. `doi:10.4230/LIPIcs.FSTTCS.2019.37`.

[46] Peter Chini, Roland Meyer, and Prakash Saivasan. Fine-grained complexity of safety verification. *J. Autom. Reason.*, 64(7):1419–1444, 2020. `doi:10.1007/s10817-020-09572-x`.

[47] Peter Chini, Roland Meyer, and Prakash Saivasan. Liveness in broadcast networks. *Computing*, 104(10):2203–2223, 2022. `doi:10.1007/s00607-021-00986-y`.

[48] Peter Clote. Computational models and function algebras. In Daniel Leivant, editor, *Logical and Computational Complexity. Selected Papers. Logic and Computational Complexity, International Workshop LCC '94, Indianapolis, Indiana, USA, 13-16 October 1994*, volume 960 of *Lecture Notes in Computer Science*, pages 98–130. Springer, 1994. `doi:10.1007/3-540-60178-3\_81`.

[49] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. `doi:10.1007/978-3-319-21275-3`.

[50] Wojciech Czerwinski and Lukasz Orlikowski. Reachability in vector addition systems is Ackermann-complete. In *FOCS 2021*, pages 1229–1240. IEEE, 2021. `doi:10.1109/FOCS52979.2021.00120`.

[51] Normann Decker and Daniel Thoma. On freeze LTL with ordered attributes. In Bart Jacobs and Christof Löding, editors, *Foundations of Software Science and Computation Structures - 19th International Conference, FOSSACS 2016, Eindhoven, The Netherlands, April 2-8, 2016, Proceedings*, volume 9634 of *Lecture Notes in Computer Science*, pages 269–284. Springer, 2016. `doi:10.1007/978-3-662-49630-5\_16`.

[52] Giorgio Delzanno, Arnaud Sangnier, and Riccardo Traverso. Parameterized verification of broadcast networks of register automata. In Parosh Aziz Abdulla and Igor Potapov, editors, *Reachability Problems - 7th International Workshop, RP 2013, Uppsala, Sweden, September 24-26, 2013 Proceedings*, volume 8169 of *Lecture Notes in Computer Science*, pages 109–121. Springer, 2013. `doi:10.1007/978-3-642-41036-9\_11`.

[53] Giorgio Delzanno, Arnaud Sangnier, Riccardo Traverso, and Gianluigi Zavattaro. On the complexity of parameterized reachability in reconfigurable broadcast networks. In Deepak D'Souza, Telikepalli Kavitha, and Jaikumar Radhakrishnan, editors, *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2012, December 15-17, 2012, Hyderabad, India*, volume 18 of *LIPIcs*, pages 289–300. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2012. `doi:10.4230/LIPIcs.FSTTCS.2012.289`.

[54] Giorgio Delzanno, Arnaud Sangnier, and Gianluigi Zavattaro. Parameterized verification of ad hoc networks. In Paul Gastin and François Laroussinie, editors, *CONCUR 2010 - Concurrency Theory, 21th International Conference, CONCUR 2010, Paris, France, August 31-September 3, 2010. Proceedings*, volume 6269 of *Lecture Notes in Computer Science*, pages 313–327. Springer, 2010. `doi:10.1007/978-3-642-15375-4\_22`.

[55] Giorgio Delzanno, Arnaud Sangnier, and Gianluigi Zavattaro. On the power of cliques in the parameterized verification of ad hoc networks. In Martin Hofmann, editor, *Foundations of Software Science and Computational Structures - 14th International Conference, FOSSACS 2011, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2011, Saarbrücken, Germany, March 26-April 3, 2011. Proceedings*, volume 6604 of *Lecture Notes in Computer Science*, pages 441–455. Springer, 2011. `doi:10.1007/978-3-642-19805-2\_30`.

[56] Giorgio Delzanno, Arnaud Sangnier, and Gianluigi Zavattaro. Verification of ad hoc networks with node and communication failures. In Holger Giese and Grigore Rosu, editors, *Formal Techniques for Distributed Systems - Joint 14th IFIP WG 6.1 International Conference, FMOODS 2012 and 32nd IFIP WG 6.1 International Conference, FORTE 2012, Stockholm, Sweden, June 13-16, 2012. Proceedings*, volume 7273 of *Lecture Notes in Computer Science*, pages 235–250. Springer, 2012. `doi:10.1007/978-3-642-30793-5\_15`.

[57] Giorgio Delzanno and Riccardo Traverso. Decidability and complexity results for verification of asynchronous broadcast networks. In Adrian-Horia Dediu, Carlos Martín-Vide, and Bianca Truthe, editors, *Language and Automata Theory and Applications - 7th International Conference, LATA 2013, Bilbao, Spain, April 2-5, 2013. Proceedings*, volume 7810 of *Lecture Notes in Computer Science*, pages 238–249. Springer, 2013. `doi:10.1007/978-3-642-37064-9\_22`.

[58] Guoli Ding. Subgraphs and well-quasi-ordering. *J. Graph Theory*, 16(5):489–502, 1992. doi:10.1002/jgt.3190160509.

[59] Dan Dobre and Neeraj Suri. One-step consensus with zero-degradation. In *2006 International Conference on Dependable Systems and Networks (DSN 2006), 25-28 June 2006, Philadelphia, Pennsylvania, USA, Proceedings*, pages 137–146. IEEE Computer Society, 2006. doi:10.1109/DSN.2006.55.

[60] Constantin Enea and Azadeh Farzan. On atomicity in presence of non-atomic writes. In Marsha Chechik and Jean-François Raskin, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 22nd International Conference, TACAS 2016, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2016, Eindhoven, The Netherlands, April 2-8, 2016, Proceedings*, volume 9636 of *Lecture Notes in Computer Science*, pages 497–514. Springer, 2016. doi:10.1007/978-3-662-49674-9_29.

[61] Javier Esparza. Decidability and complexity of Petri net problems - an introduction. In Wolfgang Reisig and Grzegorz Rozenberg, editors, *Lectures on Petri Nets I: Basic Models, Advances in Petri Nets, the volumes are based on the Advanced Course on Petri Nets, held in Dagstuhl, September 1996*, volume 1491 of *Lecture Notes in Computer Science*, pages 374–428. Springer, 1996. doi:10.1007/3-540-65306-6_20.

[62] Javier Esparza. Keeping a crowd safe: On the complexity of parameterized verification (Invited talk). In Ernst W. Mayr and Natacha Portier, editors, *31st International Symposium on Theoretical Aspects of Computer Science (STACS 2014), STACS 2014, March 5-8, 2014, Lyon, France*, volume 25 of *LIPIcs*, pages 1–10. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2014. doi:10.4230/LIPIcs.STACS.2014.1.

[63] Javier Esparza, Alain Finkel, and Richard Mayr. On the verification of broadcast protocols. In *14th Annual IEEE Symposium on Logic in Computer Science, Trento, Italy, July 2-5, 1999*, pages 352–359. IEEE Computer Society, 1999. doi:10.1109/LICS.1999.782630.

[64] Javier Esparza, Pierre Ganty, Jérôme Leroux, and Rupak Majumdar. Model checking population protocols. In Akash Lal, S. Akshay, Saket Saurabh, and Sandeep Sen, editors, *36th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2016, December 13-15, 2016, Chennai, India*, volume 65 of *LIPIcs*, pages 27:1–27:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.FSTTCS.2016.27.

[65] Javier Esparza, Pierre Ganty, Jérôme Leroux, and Rupak Majumdar. Verification of population protocols. *Acta Informatica*, 54(2):191–215, 2017. doi:10.1007/s00236-016-0272-3.

[66] Javier Esparza, Martin Helfrich, Stefan Jaax, and Philipp J. Meyer. Peregrine 2.0: Explaining correctness of population protocols through stage graphs. In Dang Van Hung and Oleg Sokolsky, editors, *Automated Technology for Verification and Analysis - 18th International Symposium, ATVA 2020, Hanoi, Vietnam, October 19-23, 2020, Proceedings*, volume 12302 of *Lecture Notes in Computer Science*, pages 550–556. Springer, 2020. doi:10.1007/978-3-030-59152-6_32.

[67] Javier Esparza, Stefan Jaax, Mikhail A. Raskin, and Chana Weil-Kennedy. The complexity of verifying population protocols. *Distributed Comput.*, 34(2):133–177, 2021. doi:10.1007/s00446-021-00390-x.

[68] Javier Esparza and Mogens Nielsen. Decidability issues for Petri nets - a survey. *Bull. EATCS*, 52:244–262, 1994.

[69] Javier Esparza, Mikhail A. Raskin, and Chana Weil-Kennedy. Parameterized analysis of immediate observation Petri nets. In Susanna Donatelli and Stefan Haar, editors, *Application and Theory of Petri Nets and Concurrency - 40th International Conference, PETRI NETS 2019, Aachen, Germany, June 23-28, 2019, Proceedings*, volume 11522 of *Lecture Notes in Computer Science*, pages 365–385. Springer, 2019. doi:10.1007/978-3-030-21571-2_20.

[70] Azadeh Farzan and P. Madhusudan. The complexity of predicting atomicity violations. In Stefan Kowalewski and Anna Philippou, editors, *Tools and Algorithms for the Construction and Analysis of Systems, 15th International Conference, TACAS 2009, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2009, York, UK, March 22-29, 2009. Proceedings*, volume 5505 of *Lecture Notes in Computer Science*, pages 155–169. Springer, 2009. doi:10.1007/978-3-642-00768-2_14.

[71] Diego Figueira, Santiago Figueira, Sylvain Schmitz, and Philippe Schnoebelen. Ackermannian and primitive-recursive bounds with Dickson's lemma. In *Proceedings of the 26th Annual IEEE Symposium on Logic in Computer Science, LICS 2011, June 21-24, 2011, Toronto, Ontario, Canada*, pages 269–278. IEEE Computer Society, 2011. doi:10.1109/LICS.2011.39.

[72] Diego Figueira and Luc Segoufin. Bottom-up automata on data trees and vertical XPath. In Thomas Schwentick and Christoph Dürr, editors, *28th International Symposium on Theoretical Aspects of Computer Science, STACS 2011, March 10-12, 2011, Dortmund, Germany*, volume 9 of *LIPIcs*, pages 93–104. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2011. doi:10.4230/LIPIcs.STACS.2011.93.

[73] Alain Finkel and Philippe Schnoebelen. Well-structured transition systems everywhere! *Theor. Comput. Sci.*, 256(1-2):63–92, 2001. doi:10.1016/S0304-3975(00)00102-X.

[74] Estíbaliz Fraca and Serge Haddad. Complexity analysis of continuous Petri nets. *Fundam. Informaticae*, 137(1):1–28, 2015. doi:10.3233/FI-2015-1168.

[75] Zeinab Ganjei, Ahmed Rezine, Ludovic Henrio, Petru Eles, and Zebo Peng. On reachability in parameterized phaser programs. In Tomás Vojnar and Lijun Zhang, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 25th International Conference, TACAS 2019, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2019, Prague, Czech Republic, April 6-11, 2019, Proceedings, Part I*, volume 11427 of *Lecture Notes in Computer Science*, pages 299–315. Springer, 2019. doi:10.1007/978-3-030-17462-0_17.

[76] Steven M. German and A. Prasad Sistla. Reasoning about systems with many processes. *J. ACM*, 39(3):675–735, 1992. doi:10.1145/146637.146681.

[77] Rachid Guerraoui. Non-blocking atomic commit in asynchronous distributed systems with failure detectors. *Distributed Comput.*, 15(1):17–25, 2002. doi:10.1007/s446-002-8027-4.

[78] Christoph Haase. A survival guide to Presburger arithmetic. *ACM SIGLOG News*, 5(3):67–82, 2018. doi:10.1145/3242953.3242964.

[79] Christoph Haase, Sylvain Schmitz, and Philippe Schnoebelen. The power of priority channel systems. *Log. Methods Comput. Sci.*, 10(4), 2014. doi:10.2168/LMCS-10(4:4)2014.

[80] Graham Higman. Ordering by divisibility in abstract algebras. *Proceedings of the London Mathematical Society*, s3-2(1):326–336, 1952. URL: https://londmathsoc.onlinelibrary.wiley.com/doi/abs/10.1112/plms/s3-2.1.326, arXiv:https://londmathsoc.onlinelibrary.wiley.com/doi/pdf/10.1112/plms/s3-2.1.326, doi:https://doi.org/10.1112/plms/s3-2.1.326.

[81] Florian Horn and Arnaud Sangnier. Deciding the existence of cut-off in parameterized rendez-vous networks. In Igor Konnov and Laura Kovács, editors, *31st International Conference on Concurrency Theory, CONCUR 2020, September 1-4, 2020, Vienna, Austria (Virtual Conference)*, volume 171 of *LIPIcs*, pages 46:1–46:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.CONCUR.2020.46.

[82] Marcin Jurdzinski and Ranko Lazic. Alternating automata on data trees and XPath satisfiability. *ACM Trans. Comput. Log.*, 12(3):19:1–19:21, 2011. doi:10.1145/1929954.1929956.

[83] Ravindran Kannan and Achim Bachem. Polynomial algorithms for computing the Smith and Hermite normal forms of an integer matrix. *SIAM J. Comput.*, 8(4):499–507, 1979. doi:10.1137/0208040.

[84] Narendra Karmarkar. A new polynomial-time algorithm for linear programming. In Richard A. DeMillo, editor, *Proceedings of the 16th Annual ACM Symposium on Theory of Computing, April 30 - May 2, 1984, Washington, DC, USA*, pages 302–311. ACM, 1984. doi:10.1145/800057.808695.

[85] Igor Konnov, Marijana Lazic, Ilina Stoilkovska, and Josef Widder. Survey on parameterized verification with threshold automata and the Byzantine model checker. *Log. Methods Comput. Sci.*, 19(1), 2023. doi:10.46298/lmcs-19(1:5)2023.

[86] Igor Konnov, Marijana Lazic, Helmut Veith, and Josef Widder. Para^2: parameterized path reduction, acceleration, and SMT for reachability in threshold-guarded distributed algorithms. *Formal Methods Syst. Des.*, 51(2):270–307, 2017. doi:10.1007/s10703-017-0297-4.

[87] Igor Konnov and Josef Widder. ByMC: Byzantine Model Checker. In Tiziana Margaria and Bernhard Steffen, editors, *Leveraging Applications of Formal Methods, Verification and Validation. Distributed Systems - 8th International Symposium, ISoLA 2018, Limassol, Cyprus, November 5-9, 2018, Proceedings, Part III*, volume 11246 of *Lecture Notes in Computer Science*, pages 327–342. Springer, 2018. doi:10.1007/978-3-030-03424-5_22.

[88] Igor V. Konnov, Marijana Lazic, Helmut Veith, and Josef Widder. A short counterexample property for safety and liveness verification of fault-tolerant distributed algorithms. In Giuseppe Castagna and Andrew D. Gordon, editors, *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages, POPL 2017, Paris, France, January 18-20, 2017*, pages 719–734. ACM, 2017. doi:10.1145/3009837.3009860.

[89] Igor V. Konnov, Helmut Veith, and Josef Widder. On the completeness of bounded model checking for threshold-based distributed algorithms: Reachability. *Inf. Comput.*, 252:95–109, 2017. doi:10.1016/j.ic.2016.03.006.

[90] Jure Kukovec, Igor Konnov, and Josef Widder. Reachability in parameterized systems: All flavors of threshold automata. In Sven Schewe and Lijun Zhang, editors, *29th International Conference on Concurrency Theory, CONCUR 2018, September 4-7, 2018, Beijing, China*, volume 118 of *LIPIcs*, pages 19:1–19:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.CONCUR.2018.19.

[91] Richard E. Ladner. The circuit value problem is log space complete for P. *SIGACT News*, 7(1):18–20, 1975. doi:10.1145/990518.990519.

[92] Slawomir Lasota. Improved Ackermannian lower bound for the Petri nets reachability problem. In Petra Berenbrink and Benjamin Monmege, editors, *39th International Symposium on Theoretical Aspects of Computer Science, STACS 2022, March 15-18, 2022, Marseille, France (Virtual Conference)*, volume 219 of *LIPIcs*, pages 46:1–46:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. `doi:10.4230/LIPIcs.STACS.2022.46`.

[93] Jérôme Leroux. The reachability problem for Petri nets is not primitive recursive. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 1241–1252. IEEE, 2021. `doi:10.1109/FOCS52979.2021.00121`.

[94] Roland Meyer. On boundedness in depth in the π-calculus. In Giorgio Ausiello, Juhani Karhumäki, Giancarlo Mauri, and C.-H. Luke Ong, editors, *Fifth IFIP International Conference On Theoretical Computer Science - TCS 2008, IFIP 20th World Computer Congress, TC 1, Foundations of Computer Science, September 7-10, 2008, Milano, Italy*, volume 273 of *IFIP*, pages 477–489. Springer, 2008. `doi:10.1007/978-0-387-09680-3\_32`.

[95] Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes, I. *Inf. Comput.*, 100(1):1–40, 1992. `doi:10.1016/0890-5401(92)90008-4`.

[96] Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes, II. *Inf. Comput.*, 100(1):41–77, 1992. `doi:10.1016/0890-5401(92)90009-5`.

[97] Achour Mostéfaoui, Eric Mourgaya, Philippe Raipin Parvédy, and Michel Raynal. Evaluating the condition-based approach to solve consensus. In *2003 International Conference on Dependable Systems and Networks (DSN 2003), 22-25 June 2003, San Francisco, CA, USA, Proceedings*, pages 541–550. IEEE Computer Society, 2003. `doi:10.1109/DSN.2003.1209964`.

[98] Ketan Mulmuley. A fast parallel algorithm to compute the rank of a matrix over an arbitrary field. *Comb.*, 7(1):101–104, 1987. `doi:10.1007/BF02579205`.

[99] Tadao Murata. Petri nets: Properties, analysis and applications. *Proc. IEEE*, 77(4):541–580, 1989. `doi:10.1109/5.24143`.

[100] Christos H. Papadimitriou. *Computational complexity*. Academic Internet Publ., 2007.

[101] Guillermo A. Pérez and Ritam Raha. Revisiting parameter synthesis for one-counter automata. In Florin Manea and Alex Simpson, editors, *30th*

EACSL Annual Conference on Computer Science Logic, CSL 2022, February 14-19, 2022, Göttingen, Germany (Virtual Conference), volume 216 of *LIPIcs*, pages 33:1–33:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. `doi:10.4230/LIPIcs.CSL.2022.33`.

[102] Michel Raynal. A case study of agreement problems in distributed systems: Non-blocking atomic commitment. In *2nd High-Assurance Systems Engineering Workshop (HASE '97), August 11-12, 1997, Washington, DC, USA, Proceedings*, pages 209–214. IEEE Computer Society, 1997. `doi:10.1109/HASE.1997.648067`.

[103] Wolfgang Reisig. *Petri Nets: An Introduction*, volume 4 of *EATCS Monographs on Theoretical Computer Science*. Springer, 1985. `doi:10.1007/978-3-642-69968-9`.

[104] Fernando Rosa-Velardo. Ordinal recursive complexity of unordered data nets. *Inf. Comput.*, 254:41–58, 2017. `doi:10.1016/j.ic.2017.02.002`.

[105] Sylvain Schmitz. Complexity bounds for ordinal-based termination - (Invited talk). In Joël Ouaknine, Igor Potapov, and James Worrell, editors, *Reachability Problems - 8th International Workshop, RP 2014, Oxford, UK, September 22-24, 2014. Proceedings*, volume 8762 of *Lecture Notes in Computer Science*, pages 1–19. Springer, 2014. `doi:10.1007/978-3-319-11439-2\_1`.

[106] Sylvain Schmitz. Complexity hierarchies beyond elementary. *ACM Trans. Comput. Theory*, 8(1):3:1–3:36, 2016. `doi:10.1145/2858784`.

[107] Sylvain Schmitz. *Algorithmic Complexity of Well-Quasi-Orders. (Complexité algorithmique des beaux pré-ordres)*. 2017. URL: `https://tel.archives-ouvertes.fr/tel-01663266`.

[108] Sylvain Schmitz and Philippe Schnoebelen. Multiply-recursive upper bounds with Higman's lemma. In Luca Aceto, Monika Henzinger, and Jirí Sgall, editors, *Automata, Languages and Programming - 38th International Colloquium, ICALP 2011, Zurich, Switzerland, July 4-8, 2011, Proceedings, Part II*, volume 6756 of *Lecture Notes in Computer Science*, pages 441–452. Springer, 2011. `doi:10.1007/978-3-642-22012-8\_35`.

[109] Sylvain Schmitz and Philippe Schnoebelen. The power of well-structured systems. In Pedro R. D'Argenio and Hernán C. Melgratti, editors, *CONCUR 2013 - Concurrency Theory - 24th International Conference, CONCUR 2013, Buenos Aires, Argentina, August 27-30, 2013. Proceedings*, volume 8052 of *Lecture Notes in Computer Science*, pages 5–24. Springer, 2013. `doi:10.1007/978-3-642-40184-8\_2`.

[110] Yee Jiun Song and Robbert van Renesse. Bosco: One-step Byzantine asynchronous consensus. In Gadi Taubenfeld, editor, *Distributed Computing,*

22nd International Symposium, DISC 2008, Arcachon, France, September 22-24, 2008. Proceedings, volume 5218 of *Lecture Notes in Computer Science*, pages 438–450. Springer, 2008. `doi:10.1007/978-3-540-87779-0\_30`.

[111] T. K. Srikanth and Sam Toueg. Simulating authenticated broadcasts to derive simple fault-tolerant algorithms. *Distributed Comput.*, 2(2):80–94, 1987. `doi:10.1007/BF01667080`.

[112] Ilina Stoilkovska, Igor Konnov, Josef Widder, and Florian Zuleger. Eliminating message counters in synchronous threshold automata. In Fritz Henglein, Sharon Shoham, and Yakir Vizel, editors, *Verification, Model Checking, and Abstract Interpretation - 22nd International Conference, VMCAI 2021, Copenhagen, Denmark, January 17-19, 2021, Proceedings*, volume 12597 of *Lecture Notes in Computer Science*, pages 196–218. Springer, 2021. `doi:10.1007/978-3-030-67067-2\_10`.

[113] Ilina Stoilkovska, Igor Konnov, Josef Widder, and Florian Zuleger. Verifying safety of synchronous fault-tolerant algorithms by bounded model checking. *Int. J. Softw. Tools Technol. Transf.*, 24(1):33–48, 2022. `doi:10.1007/s10009-021-00637-9`.

[114] Thomas Wies, Damien Zufferey, and Thomas A. Henzinger. Forward analysis of depth-bounded processes. In C.-H. Luke Ong, editor, *Foundations of Software Science and Computational Structures, 13th International Conference, FOSSACS 2010, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2010, Paphos, Cyprus, March 20-28, 2010. Proceedings*, volume 6014 of *Lecture Notes in Computer Science*, pages 94–108. Springer, 2010. `doi:10.1007/978-3-642-12032-9\_8`.

Appendix

Appendix A

Complexity of Controlled Bad Sequences over Finite Sets of $\mathbb{N}^d$ (LICS 2020)

This section contains a reprinting of the following paper, which has been published as a peer-reviewed conference paper.

> A. R. Balasubramanian. *Complexity of controlled bad sequences over finite sets of* $\mathbb{N}^d$. In conference proceedings of LICS 2020. ACM, 2020. Pages - 130-140. doi: `10.1145/3373718.3394753`

According to the author rights of the Association for Computing Machinery (ACM), the author of this thesis is permitted to include the above paper in this thesis. The relevant excerpt is the following:

For more information, please see `https://authors.acm.org/author-resources/author-rights`, in particular, the section on Reuse.

Summary

We provide upper and lower bounds for the length of controlled bad sequences over the majoring and the minoring orderings of finite sets of $\mathbb{N}^d$. The results are obtained by bounding the length of such sequences by functions from the Cichoń hierarchy. This allows us to translate these results to bounds over the fast-growing complexity classes.

The obtained bounds are proven to be tight for the majoring ordering. Finally, we use the results on controlled bad sequences to prove upper bounds for the emptiness problem of some classes of automata.

Contributions of the author of this thesis

I am the sole author of this paper.

Complexity of controlled bad sequences over finite sets of $\mathbb{N}^d$

A. R. Balasubramanian*
Technical University of Munich
Munich, Germany
bala.ayikudi@tum.de

Abstract

We provide upper and lower bounds for the length of controlled bad sequences over the majoring and the minoring orderings of finite sets of $\mathbb{N}^d$. The results are obtained by bounding the length of such sequences by functions from the Cichon hierarchy. This allows us to translate these results to bounds over the fast-growing complexity classes.

The obtained bounds are proven to be tight for the majoring ordering, which solves a problem left open by Abriola, Figueira and Senno (Theor. Comp. Sci, Vol. 603). Finally, we use the results on controlled bad sequences to prove upper bounds for the emptiness problem of some classes of automata.

CCS Concepts: • **Theory of computation** → **Complexity classes**; *Program verification.*

Keywords: well-quasi orders, controlled bad sequences, majoring and minoring ordering

ACM Reference Format:
A. R. Balasubramanian. 2020. Complexity of controlled bad sequences over finite sets of $\mathbb{N}^d$. In *Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS '20), July 8–11, 2020, Saarbrücken, Germany.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3373718.3394753

1 Introduction

A well-quasi order (wqo) over a set A is a reflexive and transitive relation $\leq_A$ such that every infinite sequence $x_0, x_1, x_2, \ldots$ over A has an increasing pair $x_i \leq_A x_j$ with $i < j$. A normed wqo (nwqo) is a wqo $(A, \leq_A)$ which has a *norm* function $|\cdot| : A \to \mathbb{N}$ such that the pre-image of n under $|\cdot|$ is finite for every n. A sequence over a wqo is called a bad sequence if it contains no increasing pair. Hence, all bad sequences over a well-quasi order are necessarily finite.

Well-quasi orders are an important tool in logic, combinatorics and computer science as evidenced by their applications in term-rewriting systems [8], algorithms [9, 16] and verification of infinite state systems [1, 2, 12]. Indeed, well-quasi orders form the backbone for the ubiquitous *well-structured transition systems* (wsts) [1, 12], whose coverability problem is shown to be decidable thanks to well-quasi orders.

In recent years, significant effort has been put in to understand the complexity of the coverability procedure for various well-structured transition systems (See [4, 6, 11, 17, 20] and also [19] for a catalogue of many problems). The key idea behind proving upper bounds for the coverability algorithm is the following: For a given class of wsts, the running time of the coverability procedure for that class can be bounded by the length of *controlled bad sequences* of the underlying normed well-quasi order (See definition 2.4 for a formal definition of controlled bad sequences). Intuitively, for a function g and a number n, a sequence $x_0, x_1, x_2, \ldots, x_l$ is called a (g, n)-controlled bad sequence if $|x_0| \leq n, |x_1| \leq g(n), |x_2| \leq g(g(n))$ and so on. A simple application of Konig's lemma can be used to show that for every n, there is a (g, n)-controlled bad sequence of maximum length. Hence, for every function g we can define a *length function* which maps a number n to the length of the longest (g, n)-controlled bad sequence.

The main observation made in [4, 6, 11, 17, 19, 20] is that, for various classes of well-structured systems, an upper bound on the running time of the coverability procedure could be obtained by bounding the length function of some specific g over the underlying wqo of that class. Motivated by this, upper bounds on the length of controlled bad sequences have been obtained for various well-quasi orders: The product ordering over $\mathbb{N}^d$ ([11]), the lexicographic ordering over $\mathbb{N}^d$ ([4]), the multiset ordering over multisets of $\mathbb{N}^d$ ([4]), the subword ordering over words [20] and the linear ordering over ordinals [18], to name a few. Using these results, time bounds have been established for the following problems (See [19] for a more detailed overview): coverability of lossy counter machines, coverability and termination of lossy channel systems, coverability of unordered data nets, emptiness of alternating 1-register and 1-clock automata, the

*Work done when the author was an intern at LSV, ENS-Saclay Paris

LICS '20, July 8–11, 2020, Saarbrücken, Germany
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-7104-9/20/07...$15.00
https://doi.org/10.1145/3373718.3394753

regular Post embedding problem, conjunctive relevant implication and 1-dimensional VASS universality. The present work is a contribution in this field of inquiry.

Our contributions: In this paper, we prove lower and upper bounds on the length of controlled bad sequences for the *majoring* and *minoring* ordering (See definition 2.9) over the collection of all finite sets of $\mathbb{N}^d$ (hereafter denoted by $\mathbb{P}_f(\mathbb{N}^d)$). Both orderings have been used to prove the decidability of the emptiness problem for some classes of automata in [14] and [10]. Our main results are the following:

- We show that if the function g is primitive recursive, then the length function of g for the majoring ordering over $\mathbb{P}_f(\mathbb{N}^d)$ is bounded by a function in the complexity class $\mathscr{F}_{\omega^{d-1}}$ (For a definition of $\mathscr{F}_{\omega^{d-1}}$, see section 7). We also show that the length function of g for the minoring ordering over $\mathbb{P}_f(\mathbb{N}^d)$ is bounded by a function in $\mathscr{F}_{\omega^{d-1}.2^d}$
- To complement the upper bounds, we also provide lower bounds on the length functions. We prove the existence of a primitive recursive (in fact, polynomial) function g such that the length function of g over the majoring ordering is bounded from below by a function in $\mathscr{F}_{\omega^{d-1}}$. A similar result is also obtained for the minoring ordering.
- We use the upper bounds on the length functions to provide upper bounds on the running time for the emptiness problem of some classes of automata operating on trees.

Related work: Length functions for the majoring ordering over $\mathbb{P}_f(\mathbb{N}^d)$ was considered in [4], where the authors proved an upper bound of $\mathscr{F}_{\omega^d}$. However no lower bound was provided and the authors left open the question of the tightness of their bound. Our results (theorems 7.1, 7.2) show that their bound is not optimal and gives tight upper and lower bounds. Some results concerning the minoring ordering were presented in [3], but no bounds on the length function were proven. To the best of our knowledge, we provide the first upper bounds for length functions of the minoring ordering over $\mathbb{P}_f(\mathbb{N}^d)$.

Our techniques: Various results regarding length functions have been proven using the notion of a *reflection* from one normed wqo to another (See definition 3.3 of [20] or definition 2.12). A reflection is a map from one nwqo A to another nwqo B, which satisfies some properties on the order and norm. If a reflection exists from A to B it can be easily proven that the length function of g over A is less than the length function of g over B. However, it turns out that reflections are not sufficient for our purposes. To this end, we define a generalization of reflections called *polynomial reflections* (See definition 2.12). We show that if a polynomial reflection exists from A to B, then bounds on the length function for g over A can be easily transferred to bounds on the length function for h over B, where h is a function obtained by composing a polynomial with g.

We then show that there exists a polynomial reflection from the set of ordinals less than $\omega^{\omega^{d-1}}$ (with the usual ordinal ordering) to $\mathbb{P}_f(\mathbb{N}^d)$ with the majoring ordering (Lemma 3.1). This enables us to establish a lower bound for the majoring ordering in terms of lower bounds for the order on ordinals, which are already known ([18]).

The upper bound for the majoring ordering is proved by following the framework established by Schmitz and Schnoebelen in a series of papers ([11, 18, 20]), which we briefly describe here. It is well known that using the descent equation, the length function for a nwqo can be expressed inductively by length functions over its "residuals". However, the residuals of a nwqo can become extremely complex to derive any useful bounds for the length function. To overcome this, we associate an ordinal to each residual (called the order type) and a non-trivial "derivative" operator for each ordinal. We then show that in the descent equation, we can replace the residuals of a nwqo with the derivative operator of the order type of that nwqo, which are much more amenable to analysis. Once this is carried out, we exploit some properties of the derivative operator along with some facts about the Cichon hierarchy and ordinal ordering to establish bounds on the length function.

The lower bound for the minoring ordering is established by giving a simple polynomial reflection from the majoring ordering to the minoring ordering (Lemma 5.1). By using the lower bounds proved for the majoring ordering, we can infer lower bounds for the minoring ordering. Finally, the upper bound for the minoring ordering is established by giving a non-trivial polynomial reflection from the minoring ordering to a *cartesian product* of various majoring orderings (Lemma 6.2). The intuition behind the reflection is discussed in detail in section 6.

Outline of the paper: We recall basic notions of wqos, ordinals and sub-recursive hierarchies in section 2. In sections 3 and 4 we prove lower and upper bounds for the majoring ordering in terms of functions from the Cichon hierarchy. Similar results are proved in sections 5 and 6 for the minoring ordering. We give a classification of these bounds in the fast-growing hierarchy in section 7. Finally, we conclude with providing some applications of our results in 8.

Due to lack of space, some of the proofs can be found in the full version of this paper, which is available at: https://arxiv.org/abs/1909.01667

2 Preliminaries

We recall some basic facts about well-quasi orders (see [5]). A *quasi ordering* (qo) over a set A is a relation $\leq$ such that $\leq$ is reflexive and transitive. We write $x < y$ if $x \leq y$ and $y \not\leq x$. We also say $x \equiv y$ if $x \leq y$ and $y \leq x$. A *well-quasi ordering* (wqo) over a set A is a qo $\leq$ such that for every infinite sequence $x_0, x_1, x_2, \ldots$, there exists $i < j$ such that $x_i \leq x_j$. A *norm function* over a set A is a function $|\cdot| : A \to \mathbb{N}$ such that for every $n \in \mathbb{N}$, the set $\{x \in A : |x| < n\}$ is finite.

Definition 2.1. A *normed wqo* is a wqo $(A, \leq_A, |\cdot|_A)$ equipped with a norm function $|\cdot|_A$.

The set A will be called as the domain of the nwqo. If $(A, \leq_A, |\cdot|_A)$ is a nwqo and $S \subseteq A$, then the nwqo induced by S is the nwqo $(S, \leq_S, |\cdot|_S)$ where $\leq_S$ and $|\cdot|_S$ are the restrictions of $\leq_A$ and $|\cdot|_A$ on S respectively. We use the notation $A_{\leq n}$ to define the set $\{x \in A : |x| \leq n\}$. Whenever the order $\leq_A$ or the norm $|\cdot|_A$ is clear from the context, we will drop those and just refer to the nwqo by the domain A.

Example 2.2. (*Some basic nwqos*) : The set of natural numbers with the usual ordering and the identity norm $(\mathbb{N}, \leq, \mathtt{id})$ is clearly seen to be a nwqo. Another nwqo is any finite set $\{a_0, a_1, \ldots, a_{k-1}\}$ such that distinct letters are unordered and $|a_i| = 0$ for every i. We will denote this nwqo by Γ_k. Notice that Γ_0 is the empty nwqo.

Given two nwqos A and B we write $A \equiv B$ when A and B are *isomorphic* structures. In particular the norm functions must be preserved by the isomorphism.

Good, bad and controlled sequences

Definition 2.3. A sequence $x_0, x_1, \ldots$ over a qo $(A, \leq_A)$ is called *good* if there exists $i < j$ such that $x_i \leq_A x_j$. A sequence which is not good is called *bad*. Notice that every bad sequence in a wqo is necessarily finite.

Definition 2.4. A *control function* is a mapping $g : \mathbb{N} \to \mathbb{N}$. For an $n \in \mathbb{N}$, a sequence $x_0, x_1, \ldots$ over a nwqo A is (g, n)-controlled if

$$\forall i \in \mathbb{N},\ |x_i|_A \leq g^i(n) = \overbrace{g(g(\ldots(g}^{i \text{ times}}(n))))$$

By a straightforward application of Konig's lemma, we have the following proposition: (See proposition 2.5 of [20])

Proposition 2.5. Let A be a nwqo and let g be a control function. For every $n \in \mathbb{N}$, there exists a finite maximum length $L \in \mathbb{N}$ for (g, n)-controlled bad sequences over A.

Therefore the above proposition lets us define a function $L_{A,g} : \mathbb{N} \to \mathbb{N}$ which for every $n \in \mathbb{N}$, assigns the maximum length of a (g, n)-controlled bad sequence over A. We will call this *the length function of A and g*. From now on, we assume that g is a *strictly increasing inflationary* function (Here inflationary means that $g(n) \geq n$ for all $n \in \mathbb{N}$).

Descent equation

We can express the length function by induction over nwqos. To do this we need the notion of *residuals*.

Definition 2.6. Let A be a nwqo and $x \in A$. The *residual* A/x is the nwqo induced by the subset $A/x := \{y \in A : x \not\leq_A y\}$

We have the following proposition: (See proposition 2.8 of [20])

Proposition 2.7.

$$L_{A,g}(n) = \max_{x \in A_{\leq n}} \{1 + L_{A/x,g}(g(n))\}$$

This equation is called the *descent equation*. The descent equation implies that unraveling the length function inductively gives us a way of computing it. If $A \supsetneq A/x_0 \supsetneq A/x_0/x_1 \supsetneq \ldots$, it follows that $x_0, x_1, \ldots$ is a bad sequence and so the inductive unraveling of proposition 2.7 is well founded.

2.1 Constructing Normed Wqo's

In this section, we will see how to construct "complex" nwqos in terms of more simpler nwqos. The constructions we use in this paper are disjoint sums, cartesian products and finite powersets.

Definition 2.8. (Disjoint sum and cartesian product) Let A_1 and A_2 be two nwqos. The *disjoint sum* $A_1 + A_2$ is the nwqo given by

$$A_1 + A_2 := \{(i, x) : i \in \{1, 2\} \text{ and } x \in A_i\}$$
$$(i, x) \leq_{A_1+A_2} (j, y) \Leftrightarrow i = j \text{ and } x \leq_{A_i} y$$
$$|(i, x)|_{A_1+A_2} := |x|_{A_i}$$

The cartesian product $A_1 \times A_2$ is the nwqo given by

$$A_1 \times A_2 := \{(x_1, x_2) : x_1 \in A_1, x_2 \in A_2\}$$
$$(x_1, x_2) \leq_{A_1 \times A_2} (y_1, y_2) \Leftrightarrow x_1 \leq_{A_1} y_1 \text{ and } x_2 \leq_{A_2} y_2$$
$$|(x_1, x_2)|_{A_1 \times A_2} := \max(|x_1|_{A_1}, |x_2|_{A_2})$$

It is well known that both $A_1 + A_2$ and $A_1 \times A_2$ are nwqos when A_1 and A_2 are. Of special interest to us is the cartesian product $(\mathbb{N}^d, \leq_{\mathbb{N}^d}, |\cdot|_{\mathbb{N}^d})$ which is obtained by taking cartesian product of $(\mathbb{N}, \leq, \mathtt{id})$ with itself d times. From now on, whenever we refer to the underlying order of $\mathbb{N}^d$, we will always mean this cartesian product ordering.

Definition 2.9. (Majoring and minoring orderings) Let A be a nwqo. We construct two nwqos whose domain will be the set of all finite subsets of A, which we denote by $\mathbb{P}_f(A)$. The first is called the *majoring ordering* and is defined as

$$\mathbb{P}_f(A) := \{X : X \subseteq A \text{ and } X \text{ is finite}\}$$
$$X \sqsubseteq^{\mathtt{maj}}_{\mathbb{P}_f(A)} Y \Leftrightarrow \forall x \in X, \exists y \in Y \text{ such that } x \leq_A y$$
$$|X|_{\mathbb{P}_f(A)} := \max(\{|x|_A : x \in X\}, \mathtt{card}(X))$$

Here $\mathtt{card}(X)$ denotes the cardinality of the set X.

The second is called the *minoring ordering* and it has the same domain and the norm as that of the majoring ordering. The difference lies in the ordering, which is given by

$$X \sqsubseteq^{\mathtt{min}}_{\mathbb{P}_f(A)} Y \Leftrightarrow \forall y \in Y, \exists x \in X \text{ such that } x \leq_A y$$

The fact that $(\mathbb{P}_f(A), \sqsubseteq^{\mathtt{maj}}_{\mathbb{P}_f(A)}, |\cdot|_{\mathbb{P}_f(A)})$ is a nwqo easily follows from Higman's lemma ([13]). However $(\mathbb{P}_f(A), \sqsubseteq^{\mathtt{min}}_{\mathbb{P}_f(A)}$

$, |\cdot|_{\mathbb{P}_f(A)})$ is *not necessarily* a nwqo whenever A is ([3]). But, it is known that $(\mathbb{P}_f(\mathbb{N}^d), \sqsubseteq^{\mathtt{min}}_{\mathbb{P}_f(\mathbb{N}^d)}, |\cdot|_{\mathbb{P}_f(\mathbb{N}^d)})$ is a nwqo for every d (See [3]). Whenever there is no confusion, we drop the $\mathbb{P}_f(A)$ as a subscript and refer to the majoring (resp. minoring) nwqo as $(\mathbb{P}_f(A), \sqsubseteq^{\mathtt{maj}})$ (resp. $(\mathbb{P}_f(A), \sqsubseteq^{\mathtt{min}})$).

The results that we prove in this paper will only concern the nwqos $(\mathbb{P}_f(\mathbb{N}^d), \sqsubseteq^{\mathtt{maj}}, |\cdot|_{\mathbb{P}_f(\mathbb{N}^d)})$ and $(\mathbb{P}_f(\mathbb{N}^d), \sqsubseteq^{\mathtt{min}}, |\cdot|_{\mathbb{P}_f(\mathbb{N}^d)})$. However, for the purposes of our proofs, we also need the following wqos which can be seen as extensions of the majoring and minoring ordering to the set of *all* subsets of a wqo.

Definition 2.10. (Arbitrary subsets) Let $(A, \leq_A)$ be a wqo and let $\mathbb{P}(A)$ denote the set of all subsets (finite and infinite) of A. Let $X, Y \in \mathbb{P}(A)$. We define,

$$X \sqsubseteq^{\mathtt{maj}} Y \iff \forall x \in X, \exists y \in Y \text{ such that } x \leq_A y$$

$$X \sqsubseteq^{\mathtt{min}} Y \iff \forall y \in Y, \exists x \in X \text{ such that } x \leq_A y$$

Further given $X \in \mathbb{P}(A)$ define,

- $\min(X) := \{x \in X : \forall x' \in X,\ [x' \leq_A x \implies x \equiv x']\}$
- $\uparrow X = \{a : \exists x \in X, x \leq_A a\}$
- $\downarrow X = \{a : \exists x \in X, a \leq_A x\}$

Notice that if $\leq_A$ is also guaranteed to be antisymmetric, then $\min(X)$ is always a finite set, irrespective of whether X is finite or infinite. Also, observe that we do *not* endow $\mathbb{P}(A)$ with a norm.

Proposition 2.11. Let $X, Y \in \mathbb{P}(\mathbb{N}^d)$. The following facts are known about $(\mathbb{P}(\mathbb{N}^d), \sqsubseteq^{\mathtt{maj}})$ and $(\mathbb{P}(\mathbb{N}^d), \sqsubseteq^{\mathtt{min}})$ (see [3]):

1. The ordering $(\mathbb{P}(\mathbb{N}^d), \sqsubseteq^{\mathtt{maj}})$ is a wqo
2. The ordering $(\mathbb{P}(\mathbb{N}^d), \sqsubseteq^{\mathtt{min}})$ is a wqo
3. $X \sqsubseteq^{\mathtt{maj}} Y \iff \downarrow X \sqsubseteq^{\mathtt{maj}} \downarrow Y$
4. $X \sqsubseteq^{\mathtt{min}} Y \iff \uparrow X \sqsubseteq^{\mathtt{min}} \uparrow Y$
5. $X \sqsubseteq^{\mathtt{maj}} Y \iff \mathbb{N}^d \setminus \downarrow X \sqsubseteq^{\mathtt{min}} \mathbb{N}^d \setminus \downarrow Y$
6. $X \sqsubseteq^{\mathtt{min}} Y \iff \mathbb{N}^d \setminus \uparrow X \sqsubseteq^{\mathtt{maj}} \mathbb{N}^d \setminus \uparrow Y$
7. $X \sqsubseteq^{\mathtt{min}} Y \iff \min(X) \sqsubseteq^{\mathtt{min}} \min(Y)$

Reflections

A major tool to prove lower and upper bounds on the length of controlled bad sequences is the notion of a *normed reflection* (See definition 3.3 of [20]). However, for our purposes we require the following notion of a *polynomial normed reflection.*

Definition 2.12. A *polynomial nwqo reflection* is a mapping $r : A \to B$ such that there exists a polynomial $q : \mathbb{N} \to \mathbb{N}$ and

$$\forall x, y \in A : r(x) \leq_B r(y) \text{ implies } x \leq_A y$$

$$\forall x \in A : |r(x)|_B \leq q(|x|_A)$$

If these conditions are satisfied then we say that r is a polynomial nwqo reflection with polynomial q and denote it by $r : A \overset{q}{\hookrightarrow} B$. If the polynomial q is the identity function, we call it a *nwqo reflection* and denote it by $r : A \hookrightarrow B$.

It is easy to see that if $r : A \overset{q}{\hookrightarrow} B$ and $r' : B \overset{q'}{\hookrightarrow} C$ are polynomial nwqo reflections, then $r' \circ r : A \xhookrightarrow{q' \circ q} C$ is also a polynomial nwqo reflection. Further, reflections are also a *precongruence* with respect to disjoint sums and cartesian products, i.e.,

Proposition 2.13. Suppose $r : A \overset{q}{\hookrightarrow} B$ and $r' : A' \overset{q'}{\hookrightarrow} B'$ are polynomial nwqo reflections. Then there exists functions s and p such that $s : A + A' \xhookrightarrow{q+q'} B + B$ and $p : A \times A' \xhookrightarrow{q+q'} B \times B'$.

We have the following important result regarding polynomial nwqo reflections.

Proposition 2.14. Let $r : A \overset{p}{\hookrightarrow} B$ be a polynomial nwqo reflection. Then $L_{A,g}(n) \leq L_{B,(q \circ g)}(q(n))$ for some polynomial q. Further if p is increasing and inflationary, then it suffices to take $q = p$.

2.2 Ordinals and subrecursive hierarchies

Since all our results will be phrased in terms of functions in the Cichon hierarchy, we recall basic facts about ordinals and subrecursive hierarchies in this section.

Ordinal terms

For basic notions about ordinals and its ordering, we refer the reader to [18]. We will use Greek letters $\alpha, \beta, \ldots$ to denote ordinals and $\leq$ to denote the ordering on ordinals. We will always use λ to denote limit ordinals.

An ordinal α has the general form (also called the *Cantor Normal Form*) $\alpha = \omega^{\beta_1} + \omega^{\beta_2} + \cdots + \omega^{\beta_m}$ where $\beta_1, \ldots, \beta_m$ are ordinals such that $\beta_1 \geq \beta_2 \geq \cdots \geq \beta_m$. For an ordinal α, we let $\mathrm{CNF}(\alpha)$ denote the set of all ordinals strictly less than α. For the purposes of this paper, we will restrict ourselves to ordinals in $\mathrm{CNF}(\epsilon_0)$ (where ϵ_0 is the supremum of $\omega, \omega^\omega, \omega^{\omega^\omega}, \cdots$)

For $c \in \mathbb{N}$, let $\omega^\beta \cdot c$ denote $\overbrace{\omega^\beta + \cdots + \omega^\beta}^{c \text{ times}}$. We sometimes write ordinals in a *strict form* as $\alpha = \omega^{\beta_1} \cdot c_1 + \omega^{\beta_2} \cdot c_2 + \cdots + \omega^{\beta_m} \cdot c_m$ where $\beta_1 > \beta_2 > \cdots > \beta_m$ and the *coefficients* c_i must be strictly bigger than 0. Using the strict form, we define a norm N on $\mathrm{CNF}(\epsilon_0)$ as follows: if $\alpha = \omega^{\beta_1} \cdot c_1 + \omega^{\beta_2} \cdot c_2 + \cdots + \omega^{\beta_m} \cdot c_m$ in the strict form then $N\alpha = \max\{c_1, \ldots, c_m, N\beta_1, \ldots, N\beta_m\}$. It is not very hard to notice that for every $\alpha < \epsilon_0$, the set $\mathrm{CNF}(\alpha)_{\leq n}$ is always finite for any n. Hence for every $\alpha < \epsilon_0$, we have a nwqo $(\mathrm{CNF}(\alpha), \leq, N)$.

We finish this sub-section with the definitions of *natural sum* $(\oplus)$ and *natural product* $(\otimes)$ for ordinals in $\mathrm{CNF}(\epsilon_0)$:

$$\sum_{i=1}^{m} \omega^{\beta_i} \oplus \sum_{j=1}^{n} \omega^{\beta'_j} := \sum_{k=1}^{m+n} \omega^{\gamma_k}$$

$$\sum_{i=1}^{m} \omega^{\beta_i} \otimes \sum_{j=1}^{n} \omega^{\beta'_j} := \bigoplus_{i=1}^{m} \bigoplus_{j=1}^{n} \omega^{\beta_i \oplus \beta'_j}$$

where $\gamma_1 \geq \gamma_2 \cdots \geq \gamma_{m+n}$ is a rearrangement of $\beta_1, \ldots, \beta_m$, $\beta'_1, \ldots, \beta'_n$.

As mentioned before, all our results will be obtained by providing reflections to and from the ordinal ordering. Hence, it is important to understand how "fast" the length of controlled bad sequences in the ordinal ordering can grow. For this purpose, we introduce sub-recursive hierarchies.

Sub-recursive hierarchies

For the purposes of describing the length of controlled bad sequences over the ordinal ordering, the hierarchies of Hardy and Cichon are sufficient [7]. However, before we introduce them we need some preliminary definitions.

A *fundamental sequence* for a *limit ordinal* λ is a sequence $(\lambda(x))_{x<\omega}$ with supremum λ, which we fix to be,

$$(\gamma + \omega^{\beta+1})(x) := \gamma + \omega^{\beta} \cdot (x+1), \qquad (\gamma + \omega^{\lambda})(x) := \gamma + \omega^{\lambda(x)}$$

The *predecessor* P_x of an ordinal $\alpha > 0$ at $x \in \mathbb{N}$ is given by

$$P_x(\alpha+1) := \alpha, \qquad P_x(\lambda) := P_x(\lambda(x))$$

Let $h : \mathbb{N} \to \mathbb{N}$ be a function. The *Hardy hierarchy* for the function h is given by $(h^{\alpha})_{\alpha<\epsilon_0}$ where

$$h^0(x) := x, \qquad h^{\alpha}(x) := h^{P_x(\alpha)}(h(x))$$

and the *Cichon hierarchy* $(h_{\alpha})_{\alpha<\epsilon_0}$ is defined as

$$h_0(x) := 0, \qquad h_{\alpha}(x) := 1 + h_{P_x(\alpha)}(h(x))$$

We also define another hierarchy called the *fast growing hierarchy* as follows:

$$f_{h,0}(x) = h(x), \qquad f_{h,\alpha+1}(x) = f_{h,\alpha}^{x+1}(x), \qquad f_{h,\lambda}(x) = f_{h,\lambda_x}(x)$$

Here $f_{h,\alpha}^{i}$ denotes i-fold composition of $f_{h,\alpha}$ with itself.

Let $L_{\alpha,g}(n)$ denote the the length of the longest (g,n)-controlled bad sequence in $\mathrm{CNF}(\alpha)$. The following theorem states that, for large enough n, the length function $L_{\alpha,g}$ and the function g_{α} in the Cichon hierarchy coincide.

Theorem 2.15. (Theorem 3.3 of [18]) Let $\alpha < \epsilon_0$ and $n \geq N\alpha$. Then $L_{\alpha,g}(n) = g_{\alpha}(n)$.

3 Lower bound for majoring ordering

In this section we prove a lower bound for length functions over $(\mathbb{P}_f(\mathbb{N}^d), \sqsubseteq^{\mathtt{maj}}, |\cdot|_{\mathbb{P}_f(\mathbb{N}^d)})$. The lower bound is presented in terms of functions over the Cichon hierarchy.

The following lemma follows an unpublished idea of Abriola, Schmitz and Schnoebelen, which has been adapted to controlled bad sequences here.

Lemma 3.1. There exists a poly. nwqo reflection

$$\mathcal{R} : (\mathrm{CNF}(\omega^{\omega^{d-1}}), \leq, N) \overset{\varphi}{\hookrightarrow} (\mathbb{P}_f(\mathbb{N}^d), \sqsubseteq^{\mathtt{maj}}, |\cdot|_{\mathbb{P}_f(\mathbb{N}^d)})$$

where $\varphi(x) = x(x+1)^d$.

Proof. We decompose the proof into three parts. As a first step, we define the map $\mathcal{R}$ from $\mathrm{CNF}(\omega^{\omega^{d-1}})$ to $\mathbb{P}_f(\mathbb{N}^d)$. In the second step, we show that $\mathcal{R}(\gamma) \sqsubseteq^{\mathtt{maj}} \mathcal{R}(\zeta) \implies \gamma \leq \zeta$. In the third step, we show that $|\mathcal{R}(\gamma)|_{\mathbb{P}_f(\mathbb{N}^d)} \leq \varphi(N\gamma)$ where N is the norm defined on ordinals in section 2.2.

First step. Let $\gamma \in \mathrm{CNF}(\omega^{\omega^{d-1}})$ such that the Cantor normal form of γ is $\omega^{\beta_1} + \omega^{\beta_2} + \cdots + \omega^{\beta_l}$. Notice that each $\beta_i \in \mathrm{CNF}(\omega^{d-1})$ and hence can be written as $\beta_i = \omega^{d-2} \cdot c_{(i,d-2)} + \omega^{d-3} \cdot c_{(i,d-3)} + \cdots + \omega^0 \cdot c_{(i,0)}$ where the coefficients $c_{i,j}$ *can* be 0. The map $\mathcal{R}$ is then defined on γ as

$$\mathcal{R}(\gamma) := \{(i, c_{(i,0)}, c_{(i,1)}, \ldots, c_{(i,d-2)}) : 1 \leq i \leq l\}$$

Second step. We now show that if $\mathcal{R}(\gamma) \sqsubseteq^{\mathtt{maj}} \mathcal{R}(\zeta)$ then $\gamma \leq \zeta$. Instead of proving this we prove the contrapositive, namely: If $\gamma > \zeta$ then $\mathcal{R}(\gamma) \not\sqsubseteq^{\mathtt{maj}} \mathcal{R}(\zeta)$.

Let $\gamma \in \mathrm{CNF}(\omega^{\omega^{d-1}})$ such that $\gamma := \omega^{\beta_1} + \omega^{\beta_2} + \cdots + \omega^{\beta_p}$ and $\beta_1 \geq \beta_2 \geq \cdots \geq \beta_p$. Further let each $\beta_i := \omega^{d-2} \cdot c_{(i,d-2)} + \omega^{d-3} \cdot c_{(i,d-3)} + \cdots + \omega^0 \cdot c_{(i,0)}$. Let $\zeta \in \mathrm{CNF}(\omega^{\omega^{d-1}})$ such that $\zeta := \omega^{\eta_1} + \omega^{\eta_2} + \cdots + \omega^{\eta_q}$ and $\eta_1 \geq \eta_2 \geq \cdots \geq \eta_q$. Further let each $\eta_i := \omega^{d-2} \cdot e_{(i,d-2)} + \omega^{d-3} \cdot e_{(i,d-3)} + \cdots + \omega^0 \cdot e_{(i,0)}$. Suppose $\gamma > \zeta$. Hence, there exists $i \in \{1, \ldots, p\}$ such that

- Either $\beta_i > \eta_i$ (or) $i > q$ and
- $\forall j$ such that $0 \leq j < \min(i,q), \beta_j = \eta_j$

Let $x := (i, c_{(i,0)}, c_{(i,1)}, \ldots, c_{(i,d-2)})$. By construction of the map $\mathcal{R}$ we have that $x \in \mathcal{R}(\gamma)$. For every $j \in \{1, \ldots, q\}$, let $y_j := (j, e_{(j,0)}, e_{(j,1)}, \ldots, e_{(j,d-2)})$. By construction of the map $\mathcal{R}$ we have that $\mathcal{R}(\zeta) = \{y_1, \ldots, y_q\}$. We will now show that $x \not\leq_{\mathbb{N}^d} y_j$ for each j. We consider two cases:

- *Case 1:* $j < i$. Therefore $\eta_j = \beta_j$. Hence $y_j := (j, c_{(j,0)}, \ldots, c_{(j,d-2)})$. Since $j < i$, we have that $x \not\leq_{\mathbb{N}^d} y_j$.
- *Case 2:* $j \geq i$. Therefore $\beta_i > \eta_i \geq \eta_j$. Suppose $x \leq_{\mathbb{N}^d} y_j$. Hence $(i, c_{(i,0)}, \ldots, c_{(i,d-2)}) \leq (j, e_{(j,0)}, \ldots, e_{(j,d-2)})$ and so $(c_{(i,0)}, \ldots, c_{(i,d-2)}) \leq (e_{(j,0)}, \ldots, e_{(j,d-2)})$. But this means that $\beta_i \leq \eta_j$ which leads to a contradiction. Hence we have that $x \not\leq_{\mathbb{N}^d} y_j$.

Therefore $x \not\leq_{\mathbb{N}^d} y_j$ for every j and so we have $\mathcal{R}(\gamma) \not\sqsubseteq^{\mathtt{maj}} \mathcal{R}(\zeta)$.

Third step. We now show that $|\mathcal{R}(\gamma)|_{\mathbb{P}_f(\mathbb{N}^d)} \leq \varphi(N\gamma)$. Let $\gamma \in \mathrm{CNF}(\omega^{\omega^{d-1}})$ such that the Cantor normal form of γ is $\omega^{\beta_1} + \omega^{\beta_2} + \ldots \omega^{\beta_l}$. Further let each $\beta_i := \omega^{d-2} \cdot c_{(i,d-2)} + \omega^{d-3} \cdot c_{(i,d-3)} + \cdots + \omega^0 \cdot c_{(i,0)}$. It is clear that

$$|\mathcal{R}(\gamma)|_{\mathbb{P}_f(\mathbb{N}^d)} = \max(l, \{c_{(i,j)}\}_{0 \leq j \leq d-2}^{1 \leq i \leq l}) \tag{1}$$

Suppose γ in the strict form looks like: $\omega^{\gamma_1} \cdot e_1 + \omega^{\gamma_2} \cdot e_2 + \ldots + \omega^{\gamma_m} \cdot e_m$ where $\gamma_1 > \gamma_2 > \cdots > \gamma_m$ and each $e_i > 0$. Notice that $l = \sum_{i=1}^{m} e_i$. Further it is also easy to observe that for all $i \in \{1, \ldots, m\}$, there exists $j \in \{1, \ldots, l\}$ such that $\gamma_i = \beta_j$. With this observation, just unraveling the definition of the norm function N implies that

$$N\gamma = \max(d-2, \{e_i\}^{1 \leq i \leq m}, \ \{c_{i,j}\}_{0 \leq j \leq d-2}^{1 \leq i \leq l}) \tag{2}$$

Since each $\gamma_i \in \mathrm{CNF}(\omega^{d-1})$, we can write each γ_i as $\omega^{d-2} \cdot c'_{(i,d-2)} + \omega^{d-3} \cdot c'_{(i,d-3)} + \cdots + \omega^0 \cdot c'_{(i,0)}$ where each $c'_{(i,j)} \leq N\gamma_i \leq N\gamma$. Notice that each γ_i is uniquely determined by its coefficients $(c'_{(i,0)}, \ldots, c'_{(i,d-2)})$, i.e., if $\gamma_i \neq \gamma_j$ then $(c'_{(i,0)}, \ldots, c'_{(i,d-2)}) \neq (c'_{(j,0)}, \ldots, c'_{(j,d-2)})$. Therefore we have an injective map from $\{\gamma_i : 1 \leq i \leq m\}$ to the set $\{x : x \in \mathbb{N}^{d-1},\ |x|_{\mathbb{N}^{d-1}} \leq N\gamma\}$. It then follows that $m \leq (N\gamma + 1)^d$. Hence

$$l = \sum_{i=1}^{m} e_i \leq \sum_{i=1}^{m} N\gamma \leq \sum_{i=1}^{(N\gamma+1)^d} N\gamma = \varphi(N\gamma)$$

By equations (1) and (2) this implies that $|\mathcal{R}(\gamma)|_{\mathbb{P}_f(\mathbb{N}^d)} \leq \varphi(N\gamma)$. □

Therefore by applying proposition 2.14 and theorem 2.15 we have,

Theorem 3.2. Let $\alpha = \omega^{\omega^{d-1}}$, $\varphi(x) = x(x+1)^d$ and let $n \geq N(\omega^{\omega^{d-1}})$. Then

$$g_\alpha(n) = L_{\alpha,g}(n) \leq L_{(\mathbb{P}_f(\mathbb{N}^d), \sqsubseteq^{\mathtt{maj}}), (\varphi \circ g)}(\varphi(n))$$

4 Upper bound for majoring ordering

In this section we will prove upper bounds on the length of controlled bad sequences for the majoring ordering over $\mathbb{P}_f(\mathbb{N}^d)$. The upper bounds are proven by following the framework established by Schmitz and Schnoebelen in a series of papers([20],[11],[18]) to prove upper bounds for various well-quasi orders.

We consider the family of nwqos obtained from $\{(\mathbb{P}_f(\mathbb{N}^d), \sqsubseteq^{\mathtt{maj}})\}_{d>0}$ and $\{\Gamma_d\}_{d\in\{0,1\}}$ by taking disjoint sums and cartesian products. We call this family of nwqos the *majoring powerset nwqos*. From now on, we will denote majoring powerset nwqos as a triple $(A, \leq_A^{\mathtt{maj}}, |\cdot|_A)$ where A is the domain of the nwqo, $\leq_A^{\mathtt{maj}}$ is the underlying order and $|\cdot|_A$ is the norm.

Similar to the proof of upper bounds for the subword ordering in [20], we introduce an ordinal notation for each majoring powerset nwqo, called the type of that nwqo. The type of a nwqo will turn out to be useful in bounding the corresponding length function using subrecursive hierarchies.

Notice that if $\alpha \in \mathrm{CNF}(\omega^{\omega^\omega})$ then α can always be decomposed as $\alpha = \bigoplus_{i=1}^{m} \bigotimes_{j=1}^{j_i} \omega^{\omega^{d_{i,j}}}$. (Here the empty product is taken to be 1 and the empty sum is taken to be 0). We now map each majoring powerset nwqo to an ordinal in $\mathrm{CNF}(\omega^{\omega^\omega})$ as follows:

$$o(\Gamma_0) = 0, \qquad o(\Gamma_1) = 1, \qquad o(\mathbb{P}_f(\mathbb{N}^d)) = \omega^{\omega^{d-1}}$$

$$o(A + B) = o(A) \oplus o(B), \qquad o(A \times B) = o(A) \otimes o(B)$$

Also with each ordinal $\alpha \in \mathrm{CNF}(\omega^{\omega^\omega})$ we can associate a canonical majoring powerset nwqo, which we will denote by $C(\alpha)$.

$$C(0) = \Gamma_0, \qquad C(1) = \Gamma_1, \qquad C(\omega^{\omega^d}) = \mathbb{P}_f(\mathbb{N}^{d+1})$$

$$C(\alpha \oplus \beta) = C(\alpha) + C(\beta), \qquad C(\alpha \otimes \beta) = C(\alpha) \times C(\beta)$$

It can be easily seen that the operators o and C are bijective inverses of each other (modulo isomorphism of nwqos).

Derivatives

The next step is to define a *derivative* operator for ordinals. To this end, for each $n \in \mathbb{N}$, we define a D_n operator as follows:

$$D_n(k) = k-1, \qquad D_n(\omega) = n+1, \qquad D_n(\omega^{\omega^d}) = \omega^{\omega^{d-1}\cdot(d+1)n}$$

$$D_n(\omega^{\omega^{p_1}+\omega^{p_2}+\cdots+\omega^{p_k}}) = \bigoplus_{i=1}^{k} \left(D_n(\omega^{\omega^{p_i}}) \otimes \bigotimes_{j\neq i} \omega^{\omega^{p_j}} \right)$$

Using this operator, we define a ∂_n operator as follows:

$$\partial_n\left(\sum_{i=1}^{m} \omega^{\beta_i}\right) = \left\{ D_n(\omega^{\beta_i}) \oplus \bigoplus_{j\neq i} \omega^{\beta_j} \mid i = 1, \ldots, m \right\}$$

Notice that if $\alpha = \omega^\beta$ then $\partial_n(\alpha) = \{D_n(\alpha)\}$.

Proposition 4.1. If $\beta \in \partial_n(\alpha)$ then $\beta < \alpha$

The following theorem lets us forget the actual underlying nwqo and remember only its type.

Theorem 4.2. Let A be a majoring powerset nwqo and let $\alpha = o(A)$. If $X \in A_{\leq n}$, then there exists $\alpha' \in \partial_n(\alpha)$ such that there exists a nwqo reflection $r : A/X \hookrightarrow C(\alpha')$.

Since o and C are inverses of each other, by combining the descent equation and theorem 4.2 we get,

Lemma 4.3.

$$L_{C(\alpha),g}(n) \leq \max_{\alpha' \in \partial_n(\alpha)} \{1 + L_{C(\alpha'),g}(g(n))\}$$

Upper bound using subrecursive hierarchies

Given $\alpha \in \mathrm{CNF}(\omega^{\omega^\omega})$ define

$$M_{\alpha,g}(n) = \max_{\alpha' \in \partial_n(\alpha)} \{1 + M_{\alpha',g}(g(n))\}$$

From the definition of $M_\alpha(n)$ and lemma 4.3, it is clear that $L_{C(\alpha),g}(n) \leq M_{\alpha,g}(n)$ or in other words, $L_{A,g}(n) \leq M_{o(A),g}(n)$ for any majoring powerset nwqo A. Therefore, in what follows, we will concentrate on proving upper bounds for $M_{\alpha,g}(n)$.

Let $\alpha \in \mathrm{CNF}(\omega^{\omega^\omega})$. We will say that α is k-lean if $N\alpha \leq k$. Let $h(x) = 4x \cdot g(x)$ where g is the control function. We have the following important theorem:

Theorem 4.4. If α is k-lean and $n > 0$ then $M_{\alpha,g}(n) \leq h_\alpha(4kn)$

Using theorem 4.4 and the fact that $L_{A,g}(n) \leq M_{o(A),g}(n)$, we have the following:

Theorem 4.5. Let A be any majoring powerset nwqo such that $o(A)$ is k-lean. Then for $n > 0$, we have $L_{A,g}(n) \leq M_{o(A),g}(n) \leq h_{o(A)}(4kn)$ where $h(x) = 4x \cdot g(x)$.

In particular,

Corollary 4.6. Let $\alpha = \omega^{\omega^{d-1}}$ and let $n > 0$. Then

$$L_{(\mathbb{P}_f(\mathbb{N}^d),\sqsubseteq^{\mathtt{maj}}),g}(n) \leq h_\alpha(4dn)$$

where $h(x) = 4x \cdot g(x)$.

5 Lower bound for minoring ordering

We give a lower bound on the length of controlled bad sequences for the nwqo $(\mathbb{P}_f(\mathbb{N}^d), \sqsubseteq^{\mathtt{min}}, |\cdot|_{\mathbb{P}_f(\mathbb{N}^d)})$ by giving a polynomial nwqo reflection from $(\mathbb{P}_f(\mathbb{N}^d), \sqsubseteq^{\mathtt{maj}}, |\cdot|_{\mathbb{P}_f(\mathbb{N}^d)})$ to $(\mathbb{P}_f(\mathbb{N}^d), \sqsubseteq^{\mathtt{min}}, |\cdot|_{\mathbb{P}_f(\mathbb{N}^d)})$.

Lemma 5.1. There exists a poly. nwqo reflection

$$\mathcal{R} : (\mathbb{P}_f(\mathbb{N}^d), \sqsubseteq^{\mathtt{maj}}, |\cdot|_{\mathbb{P}_f(\mathbb{N}^d)}) \overset{p}{\hookrightarrow} (\mathbb{P}_f(\mathbb{N}^d), \sqsubseteq^{\mathtt{min}}, |\cdot|_{\mathbb{P}_f(\mathbb{N}^d)})$$

where $p(x) = d(x+1)$.

Proof. Similar to lemma 3.1, we split the proof into three parts. In the first part, we define the reflection $\mathcal{R}$. In the second part we show that $\mathcal{R}(X) \sqsubseteq^{\mathtt{min}} \mathcal{R}(Y) \implies X \sqsubseteq^{\mathtt{maj}} Y$. Finally, we prove that $|\mathcal{R}(X)|_{\mathbb{P}_f(\mathbb{N}^d)} \leq p(|X|_{\mathbb{P}_f(\mathbb{N}^d)})$.

First part. The reflection $\mathcal{R}$ is defined as the following simple map: Given a set $X \in \mathbb{P}_f(\mathbb{N}^d)$, let $\mathcal{R}(X) := \min(\mathbb{N}^d \setminus \downarrow X)$.

Second part. Suppose $\mathcal{R}(X) \sqsubseteq^{\mathtt{min}} \mathcal{R}(Y)$. By definition this means that $\min(\mathbb{N}^d \setminus \downarrow X) \sqsubseteq^{\mathtt{min}} \min(\mathbb{N}^d \setminus \downarrow Y)$. By the last point of proposition 2.11 we have that $\mathbb{N}^d \setminus \downarrow X \sqsubseteq^{\mathtt{min}} \mathbb{N}^d \setminus \downarrow Y$. By the fifth point of proposition 2.11 it follows that $X \sqsubseteq^{\mathtt{maj}} Y$.

Third part. First, we set up some notation. Let $\mathbf{0}_d$ denote the zero vector in $\mathbb{N}^d$. Given an $x = (x_1, x_2, \ldots, x_d) \in \mathbb{N}^d$ and $i \in \{1, \ldots, d\}$, define $x_i^+ := (x_1, x_2, \ldots, x_{i-1}, x_i + 1, x_{i+1}, \ldots, x_d)$. Further, if $x_i > 0$ define $x_i^- := (x_1, x_2, \ldots, x_{i-1}, x_i - 1, x_{i+1}, \ldots, x_d)$.

We further split this part into two subparts. In the first subpart we prove something about the $\mathcal{R}$ mapping. In the second part, we use the proposition proven in the first subpart to show that $|\mathcal{R}(X)|_{\mathbb{P}_f(\mathbb{N}^d)} \leq p(|X|_{\mathbb{P}_f(\mathbb{N}^d)})$.

First subpart: Let $X \in \mathbb{P}_f(\mathbb{N}^d)$. We first claim that

$$\text{If } y \in \mathcal{R}(X) \text{ then } y = x_i^+ \text{ for some } x \in X \text{ and some } i \quad (3)$$

Let $y \in \mathcal{R}(X) = \min(\mathbb{N}^d \setminus \downarrow X)$. Therefore, $y \not\leq_{\mathbb{N}^d} x$ for any $x \in X$. In particular $y \neq \mathbf{0}_d$ and so there exists i such that $y_i \neq 0$. Suppose $y_i^- \in \mathbb{N}^d \setminus \downarrow X$. Since $y_i^- \leq_{\mathbb{N}^d} y$, it then follows that $y \notin \min(\mathbb{N}^d \setminus \downarrow X) = \mathcal{R}(X)$, leading to a contradiction.

Hence, if $y \in \mathcal{R}(X)$ then there exists i such that $y_i > 0$ and $y_i^- \notin \mathbb{N}^d \setminus \downarrow X$. Therefore $\exists x \in X$ such that $y_i^- \leq_{\mathbb{N}^d} x$. Since $y \in \mathcal{R}(X) = \min(\mathbb{N}^d \setminus \downarrow X)$ it follows that $y \not\leq_{\mathbb{N}^d} x$. The only way in which we can have $y_i^- \leq_{\mathbb{N}^d} x$ but $y \not\leq_{\mathbb{N}^d} x$ is when $y = x_i^+$, which proves that (3) is true.

Second subpart: Let $X^+ := \{x_i^+ : x \in X,\ 1 \leq i \leq d\}$. By (3) it is clear that $\mathcal{R}(X) \subseteq X^+$ and so

$$|\mathcal{R}(X)|_{\mathbb{P}_f(\mathbb{N}^d)} \leq |X^+|_{\mathbb{P}_f(\mathbb{N}^d)} \quad (4)$$

We proceed to bound $|X^+|_{\mathbb{P}_f(\mathbb{N}^d)}$. To do so, we only need to bound the norm of each element in X^+ and the cardinality of X^+. By construction, it is easy to see that if $y \in X^+$, then $|y|_{\mathbb{N}^d} \leq |X|_{\mathbb{P}_f(\mathbb{N}^d)} + 1$. Further, by definition of X^+, we have $\mathtt{card}(X^+) \leq d(\mathtt{card}(X)) \leq d(|X|_{\mathbb{P}_f(\mathbb{N}^d)})$. It then follows that

$$|X^+|_{\mathbb{P}_f(\mathbb{N}^d)} \leq d(|X|_{\mathbb{P}_f(\mathbb{N}^d)} + 1) \quad (5)$$

By equations 4 and 5 it follows that $|\mathcal{R}(X)|_{\mathbb{P}_f(\mathbb{N}^d)} \leq p(|X|_{\mathbb{P}_f(\mathbb{N}^d)})$ which proves the lemma. ☐

Let $\varphi(x) = x(x+1)^d$ and let $g_\varphi = \varphi \circ g$. Since $\mathcal{R}$ is a polynomial nwqo reflection, by proposition 2.14 and theorem 3.2 we have

Theorem 5.2. Let $\alpha = \omega^{\omega^{d-1}}$ and let $n \geq N(\omega^{\omega^{d-1}})$. Then

$$g_\alpha(n) \leq L_{(\mathbb{P}_f(\mathbb{N}^d),\sqsubseteq^{\mathtt{maj}}),g_\varphi}(\varphi(n)) \leq L_{(\mathbb{P}_f(\mathbb{N}^d),\sqsubseteq^{\mathtt{min}}),(p\circ g_\varphi)}(p(\varphi(n)))$$

6 Upper bound for minoring ordering

For the rest of this section, we assume that $d \geq 1$ is fixed. For any $i \leq d$, let $(P_i, \leq_{P_i}^{\mathtt{maj}}, |\cdot|_{P_i})$ be the *majoring powerset nwqo* obtained by taking cartesian product of $(\mathbb{P}_f(\mathbb{N}^i), \sqsubseteq^{\mathtt{maj}}, |\cdot|_{\mathbb{P}_f(\mathbb{N}^i)})$ with itself $\binom{d}{i}$ times, i.e., $P_i = \mathbb{P}_f(\mathbb{N}^i)^{\binom{d}{i}}$.

Let $(A_d, \leq_{A_d}^{\mathtt{maj}}, |\cdot|_{A_d})$ be the *majoring powerset nwqo* formed by taking cartesian product of $P_1, P_2, \ldots, P_d$, i.e., $A_d = P_1 \times P_2 \times \cdots \times P_d = \prod_{i=1}^d \mathbb{P}_f(\mathbb{N}^i)^{\binom{d}{i}}$. Since A_d is a majoring powerset nwqo, it has an associated *order type* $o(A_d)$ which can be easily seen to be $\bigotimes_{i=1}^d \omega^{(\omega^{i-1})\cdot\binom{d}{i}}$. Further it is easy to notice that $o(A_d)$ is $d2^d$-lean.

Having introduced A_d, we prove upper bounds on the length of controlled bad sequences for the minoring ordering on $\mathbb{P}_f(\mathbb{N}^d)$ by providing a polynomial nwqo reflection to A_d. The reflection that we provide will be a map from $(\mathbb{P}_f(\mathbb{N}^d) \setminus \emptyset, \sqsubseteq^{\mathtt{min}})$ to A_d. However, this can be easily converted to an upper bound for $(\mathbb{P}_f(\mathbb{N}^d), \sqsubseteq^{\mathtt{min}})$, thanks to the following proposition:

Proposition 6.1.

$$\begin{aligned} L_{(\mathbb{P}_f(\mathbb{N}^d),\sqsubseteq^{\mathtt{min}}),g}(n) &= 1 + L_{(\mathbb{P}_f(\mathbb{N}^d)\setminus\emptyset,\sqsubseteq^{\mathtt{min}}),g}(g(n)) \\ &\leq L_{(\mathbb{P}_f(\mathbb{N}^d)\setminus\emptyset,\sqsubseteq^{\mathtt{min}}),g}(g(n)+1) \end{aligned}$$

Proof. Notice that for any subset $X \in \mathbb{P}_f(\mathbb{N}^d)$, $X \sqsubseteq^{\mathtt{min}} \emptyset$ and so $\mathbb{P}_f(\mathbb{N}^d)/X \subseteq \mathbb{P}_f(\mathbb{N}^d)/\emptyset$. Since $X \sqsubseteq^{\mathtt{min}} \emptyset$ for any subset X, it follows that $\mathbb{P}_f(\mathbb{N}^d)/\emptyset = \mathbb{P}_f(\mathbb{N}^d) \setminus \emptyset$. Combining these two and applying the descent equation we get,

$$\begin{aligned} L_{(\mathbb{P}_f(\mathbb{N}^d),\sqsubseteq^{\mathtt{min}}),g}(n) &= \max_{|X|_{\mathbb{P}_f(\mathbb{N}^d)} \leq n} \{1 + L_{(\mathbb{P}_f(\mathbb{N}^d)/X,\sqsubseteq^{\mathtt{min}}),g}(g(n))\} \\ &= 1 + L_{(\mathbb{P}_f(\mathbb{N}^d)\setminus\emptyset,\sqsubseteq^{\mathtt{min}}),g}(g(n)) \end{aligned}$$

This proves the first equality.

The second inequality is true for the following reason: Let $X_0, X_1, \ldots, X_l$ be a $(g, g(n))$ controlled bad sequence in $\mathbb{P}_f(\mathbb{N}^d) \setminus \emptyset$. By the last point of proposition 2.11, we can assume that $X_i = \min(X_i)$ for each i. Let $x := (a_1, a_2, \ldots, a_d) \in X_0$. Construct $x' := (a_1 + 1, a_2, \ldots, a_d)$ and let $X_0' := (X_0 \setminus$

$\{x\}) \cup \{x'\}$. It can be easily verified that $X_0', X_0, X_1, \ldots, X_l$ is a $(g, g(n)+1)$ controlled bad sequence. □

Therefore, in what follows, it suffices to focus on $(\mathbb{P}_f(\mathbb{N}^d) \setminus \emptyset, \sqsubseteq^{\mathtt{min}})$. We have the following lemma:

Lemma 6.2. There exists a poly. nwqo reflection

$$\mathcal{R} : (\mathbb{P}_f(\mathbb{N}^d) \setminus \emptyset, \sqsubseteq^{\mathtt{min}}, |\cdot|_{\mathbb{P}_f(\mathbb{N}^d)}) \overset{q}{\hookrightarrow} (A_d, \leq^{\mathtt{maj}}_{A_d}, |\cdot|_{A_d})$$

where $q(x) = (x+1)^d$.

Proof sketch. We present the proof for the case when $d = 2$ and then sketch how the proof can be generalised to higher dimensions.

Let us consider $(\mathbb{P}_f(\mathbb{N}^2) \setminus \emptyset, \sqsubseteq^{\mathtt{min}})$ and let $X, Y \in \mathbb{P}_f(\mathbb{N}^2) \setminus \emptyset$. By proposition 2.11 $X \sqsubseteq^{\mathtt{min}} Y$ iff $\mathbb{N}^2 \setminus \uparrow X \sqsubseteq^{\mathtt{maj}} \mathbb{N}^2 \setminus \uparrow Y$. Let $\mathtt{comp}(X) := \mathbb{N}^2 \setminus \uparrow X$ and $\mathtt{comp}(Y) := \mathbb{N}^2 \setminus \uparrow Y$. Notice that since $X \neq \emptyset$ and $Y \neq \emptyset$, it follows that $\downarrow \mathtt{comp}(X) \neq \mathbb{N}^2$ and $\downarrow \mathtt{comp}(Y) \neq \mathbb{N}^2$. Therefore there exists n_X and n_Y such that $(n_X, n_X) \notin \downarrow \mathtt{comp}(X)$ and $(n_Y, n_Y) \notin \downarrow \mathtt{comp}(Y)$.

Unfortunately $\mathtt{comp}(X)$ and $\mathtt{comp}(Y)$ might be infinite and so we cannot use the results proved in section 4. However, we will see that we can "compress" the sets $\mathtt{comp}(X)$ and $\mathtt{comp}(Y)$ such that the compressed finite sets preserve the order between $\mathtt{comp}(X)$ and $\mathtt{comp}(Y)$.

Suppose, for some $x \in \mathbb{N}$, there are infinitely many elements of the form $(x, n_1), (x, n_2), (x, n_3), \ldots$ in the set $\mathtt{comp}(X)$. We need not store all these elements, but rather only store that there are infinitely many elements in $\mathtt{comp}(X)$ such that their first co-ordinate is x. In accordance with this intuition, we define $S_1^X := \{x :$ there exists infinitely many n such that $(x, n) \in \mathtt{comp}(X)\}$. Notice that S_1^X is a subset of $\mathbb{N}$. Similarly, we define $S_2^X := \{x' :$ there exists infinitely many n such that $(n, x') \in \mathtt{comp}(X)\}$. To complement these two sets, we now define $S_3^X := \{(x, x') \in \mathtt{comp}(X) : x \notin S_1^X$ and $x' \notin S_2^X\}$. We then consider the tuple (S_1^X, S_2^X, S_3^X). Notice that if $(x, x') \in \mathtt{comp}(X)$ then either $x \in S_1^X$ or $x' \in S_2^X$ or $(x, x') \in S_3^X$. It is then quite easy to see that if $S_1^X \sqsubseteq^{\mathtt{maj}}_{\mathbb{P}_f(\mathbb{N})} S_1^Y$ and $S_2^X \sqsubseteq^{\mathtt{maj}}_{\mathbb{P}_f(\mathbb{N})} S_2^Y$ and $S_3^X \sqsubseteq^{\mathtt{maj}}_{\mathbb{P}_f(\mathbb{N}^2)} S_3^Y$ then $\mathtt{comp}(X) \sqsubseteq^{\mathtt{maj}} \mathtt{comp}(Y)$ and so $X \sqsubseteq^{\mathtt{min}} Y$.

However it is not clear that each of the sets S_1^X, S_2^X and S_3^X are indeed finite. To prove this, first recall that there exists $n_X \in \mathbb{N}$ such that $(n_X, n_X) \notin \downarrow \mathtt{comp}(X)$.

Suppose S_1^X is infinite. By definition this means that there are infinitely many numbers $x_1, x_2, \ldots$ such that for each x_i there are infinitely many elements in $\mathtt{comp}(X)$ with first co-ordinate x_i. Pick an x_i such that $x_i \geq n_X$. Now, by definition of S_1^X we can pick a $n_i \geq n_X$ such that $(x_i, n_i) \in \mathtt{comp}(X)$. However this means that $(n_X, n_X) \in \downarrow \mathtt{comp}(X)$ which leads to a contradiction. Similar arguments also show that S_2^X is infinite.

Suppose S_3^X is infinite. Since $(n_X, n_X) \notin \downarrow \mathtt{comp}(X)$ it follows that $(n_X, n_X) \notin \downarrow S_3^X$ as well. Hence for every element $(x, y) \in S_3^X$ either $x < n_X$ or $y < n_X$. This indicates that if there are infinitely many elements in S_3^X, then there exists $x \in \mathbb{N}$ such that either there are infinitely many elements in S_3^X with their first co-ordinate as x or there are infinitely many elements with their second co-ordinate as x. In either case, by definition of S_3^X we will reach a contradiction.

Finally, we also have to show that $|(S_1^X, S_2^X, S_3^X)|_{A_2} \leq (|X|_{\mathbb{P}_f(\mathbb{N}^2)} + 1)^2$. First we show that if an element belongs to S_1^X or S_2^X or S_3^X then its norm is bounded by $|X|_{\mathbb{P}_f(\mathbb{N}^2)}$.

Suppose $x \in S_1^X$ and $x > |X|_{\mathbb{P}_f(\mathbb{N}^2)}$. Since $x \in S_1^X$ it follows that there exists $n \geq n_X$ such that $(x, n) \in \mathtt{comp}(X)$. Since $(x, n) \in \mathtt{comp}(X) = \mathbb{N}^2 \setminus \uparrow X$ it follows that for all $(y, m) \in X$ it is the case that $(y, m) \not\leq_{\mathbb{N}^2} (x, n)$. Since $x > |X|_{\mathbb{P}_f(\mathbb{N}^2)} \geq y$ it follows that $m > n$. Hence $(y, m) \not\leq_{\mathbb{N}^2} (x + n + 1, n)$ as well. Since this is true for every $(y, m) \in X$ it follows that $(x + n + 1, n) \notin \uparrow X$ and so $(x + n + 1, n) \in \mathtt{comp}(X)$. Since $(x + n + 1, n) \geq_{\mathbb{N}^2} (n_X, n_X)$ it follows that $(n_X, n_X) \in \downarrow \mathtt{comp}(X)$ which leads to a contradiction. Hence if $x \in S_1^X$ then $x \leq |X|_{\mathbb{P}_f(\mathbb{N}^2)}$. A similar argument holds for S_2^X as well.

Suppose $(x, y) \in S_3^X$ and $x > |X|_{\mathbb{P}_f(\mathbb{N}^2)}$. Since $(x, y) \in S_3^X$ there are only finitely many elements in $\mathtt{comp}(X)$ with y as their second co-ordinate. Hence we can find a n_y such that if $n \geq n_y$ then $(n, y) \notin \mathtt{comp}(X)$. Now similar to the case of S_1^X we can now show that $(x + n_y + 1, y) \in \mathtt{comp}(X)$, thus leading to a contradiction. A similar argument is employed when $y > |X|_{\mathbb{P}_f(\mathbb{N}^2)}$.

Since the norms of the elements of S_1^X, S_2^X and S_3^X are bounded by $|X|_{\mathbb{P}_f(\mathbb{N}^2)}$, it follows that their cardinalities are bounded by $(|X|_{\mathbb{P}_f(\mathbb{N}^2)} + 1)^2$. Hence the norms of S_1^X, S_2^X and S_3^X are each bounded by $(|X|_{\mathbb{P}_f(\mathbb{N}^2)} + 1)^2$, which proves our claim.

We now sketch the construction for the general case of higher dimensions, i.e, when the dimension $d \geq 2$. Notice that the set S_1^X, as defined for the case of $d = 2$, can be stated in the following manner as well: It is the set of all x such that if we fix the first co-ordinate to be x and then project $\mathtt{comp}(X)$ to the second axis, the downward closure of the projection is $\mathbb{N}$. Hence if we want to prove the lemma for $d = 3$, one way to define S_1^X would be: The set of all x such that if we fix the first co-ordinate to be x and then project $\mathtt{comp}(X)$ on the other two axes, the downward closure of the projection is $\mathbb{N}^2$. In a similar fashion, we can fill in S_2^X and S_3^X by fixing the second co-ordinate and the third co-ordinate. For S_4^X we fix the first and the second co-ordinates and check if the downward closure of the resulting projection is $\mathbb{N}$ and so on. Then we define the reflection to be $(S_1^X, \ldots, S_7^X)$. The reflection for the general case also follows a similar pattern.

Using lemma 6.2, we can now state upper bounds for the minoring ordering. Let $(\mathbb{P}_f(\mathbb{N}^d)^k, \leq^{\mathtt{min}}_{\mathbb{P}_f(\mathbb{N}^d)^k})$ be the nwqo obtained by taking the cartesian product of $(\mathbb{P}_f(\mathbb{N}^d), \sqsubseteq^{\mathtt{min}})$ with itself k times. Let $(A_d^k, \leq^{\mathtt{maj}}_{A_d^k})$ be the majoring powerset nwqo obtained by taking cartesian product of $(A_d, \leq^{\mathtt{maj}}_{A_d})$

with itself k times. The following theorem is stated in a way such that it is useful for our applications.

Theorem 6.3. Let $\alpha = \omega^{\omega^{d-1}\cdot(2^d\cdot k)}$ and let n be sufficiently large. There exists a constant c (depending only on d and k) such that

$$L_{(\mathbb{P}_f(\mathbb{N}^d)^k,\ \leq^{\mathtt{min}}_{\mathbb{P}_f(\mathbb{N}^d)^k}),g}(n) \leq t_\alpha(c \cdot g(n)^{2d})$$

where $t(x) = 4kx \cdot q(g(x))$ and $q(x) = (x+1)^d$.

Proof. Let $t(x) := 4kx \cdot q(g(x))$. Notice that if g is a strictly increasing inflationary function, then the same is true for t.

Let $\emptyset_k$ denote the tuple $(\overbrace{\emptyset, \ldots, \emptyset}^{\text{k times}})$. The proof of proposition 6.1 can be easily modified to prove that

$$L_{(\mathbb{P}_f(\mathbb{N}^d)^k,\ \leq^{\mathtt{min}}_{\mathbb{P}_f(\mathbb{N}^d)^k}),g}(n) \leq L_{(\mathbb{P}_f(\mathbb{N}^d)^k\setminus\emptyset_k,\ \leq^{\mathtt{min}}_{\mathbb{P}_f(\mathbb{N}^d)^k}),g}(g(n)+1)$$

By proposition 2.13 and lemma 6.2 we have a reflection $(\mathbb{P}_f(\mathbb{N}^d)^k\setminus\emptyset,\ \leq^{\mathtt{min}}_{\mathbb{P}_f(\mathbb{N}^d)^k}) \xhookrightarrow{k\cdot q} (A^k_d, \leq^{\mathtt{maj}}_{A^k_d})$. Using proposition 2.14 and noticing that for large enough n, we have $q(g(n)+1) \leq g(n)^{2d}$, we get,

$$L_{(\mathbb{P}_f(\mathbb{N}^d)^k\setminus\emptyset_k,\ \leq^{\mathtt{min}}_{\mathbb{P}_f(\mathbb{N}^d)^k}),g}(g(n)+1) \leq L_{(A^k_d, \leq^{\mathtt{maj}}_{A^k_d}),((k\cdot q)\circ g)}(k\cdot g(n)^{2d})$$

Notice that $o(A^k_d) = \bigotimes_{j=1}^k \left(\bigotimes_{i=1}^d \omega^{\omega^{i-1}\cdot\binom{d}{i}}\right)$ is $dk2^d$-lean. Hence by theorem 4.5 we have

$$L_{(A^k_d, \leq^{\mathtt{maj}}_{A^k_d}),((k\cdot q)\circ g)}(k \cdot g(n)^{2d}) \leq t_{o(A_d)}(4dk^2 2^d g(n)^{2d})$$

Now $o(A_d) < \omega^{\omega^{d-1}\cdot(2^d\cdot k)}$. It is known that, if $\alpha < \alpha'$ then $h_\alpha(n) \leq h_{\alpha'}(n)$ for sufficiently large n (See Lemma C.9 of [20]). Hence for sufficiently large n,

$$t_{o(A_d)}(4dk^2 2^d g(n)^{2d}) \leq t_{\omega^{\omega^{d-1}\cdot(2^d\cdot k)}}(4dk^2 2^d g(n)^{2d})$$

Hence letting $c := 4dk^2 2^d$ and $\alpha := \omega^{\omega^{d-1}\cdot(2^d\cdot k)}$ and combining all the equations, we have,

$$L_{(\mathbb{P}_f(\mathbb{N}^d)^k,\leq^{\mathtt{min}}_{\mathbb{P}_f(\mathbb{N}^d)^k}),g}(n) \leq t_\alpha(c \cdot g(n)^{2d})$$

□

7 Complexity classification

In this section, we will use the results proved in the previous sections to classify length functions for the majoring and minoring ordering based on *fast-growing complexity classes.* Let $S : \mathbb{N} \to \mathbb{N}$ denote the successor function. Let $\{S_\alpha\}, \{S^\alpha\}, \{F_\alpha\}$ denote the Hardy, Cichon and fast-growing hierarchies for the successor function respectively. Notice that $S^\alpha(x) = S_\alpha(x) + x$ for all x and for all $\alpha < \epsilon_0$.

Using these hierarchies, we define *fast growing function classes* $(\mathscr{F}_\alpha)_\alpha$ (See [15], [19]).

$$\mathscr{F}_\alpha := \bigcup_{c<\omega} \mathrm{FD}(F^c_\alpha(n))$$

Here $\mathrm{FD}(F^c_\alpha(n))$ denotes the set of all functions that can be computed by a deterministic Turing machine in time $F^c_\alpha(n)$ where F^c_α denotes the function that results when F_α is applied to itself c times. We remark in passing that $\bigcup_{\alpha<\omega} \mathscr{F}_\alpha$ already constitutes the set of all primitive recursive functions (See section 2.2.4 of [19]).

For the rest of this section, let g be a fixed strictly increasing and inflationary control function such that $g(x) \geq S(x)$.

Majoring ordering

Fix a $d > 1$ and let $\varphi(x) = x(x+1)^d$. Our lower bound for the majoring ordering can be readily translated into a complexity lower bound as follows:

Theorem 7.1. For sufficiently large n,

$$F_{\omega^{d-1}}(n) - n \leq L_{(\mathbb{P}_f(\mathbb{N}^d), \sqsubseteq^{\mathtt{maj}}),\varphi\circ g}(\varphi(n))$$

Also $L_{(\mathbb{P}_f(\mathbb{N}^d), \sqsubseteq^{\mathtt{maj}}),\varphi\circ g} \notin \mathscr{F}_\alpha$ for any $\alpha < \omega^{d-1}$.

For upper bounds, we state a general result which will be useful for our applications.

Theorem 7.2. Let g be a primitive recursive function and let $A = \mathbb{P}_f(\mathbb{N}^d)^k$ for some numbers d and k. Then $L_{(A, \leq^{\mathtt{maj}}_A),g}$ is eventually bounded by a function in $\mathscr{F}_{(\omega^{d-1})\cdot k}$

Minoring ordering

Let $p(x) = d(x+1)$. The following is a lower bound for the minoring ordering.

Theorem 7.3. For sufficiently large n,

$$F_{\omega^{d-1}}(n) - n \leq L_{(\mathbb{P}_f(\mathbb{N}^d), \sqsubseteq^{\mathtt{min}}),p\circ\varphi\circ g}(p(\varphi(n)))$$

Also $L_{(\mathbb{P}_f(\mathbb{N}^d), \sqsubseteq^{\mathtt{min}}),p\circ\varphi\circ g} \notin \mathscr{F}_\alpha$ for any $\alpha < \omega^{d-1}$.

We also have the following upper bound.

Theorem 7.4. Let g be primitive recursive and let $A = \mathbb{P}_f(\mathbb{N}^d)^k$ for some numbers d and k. Then $L_{(A, \leq^{\mathtt{min}}_A),g}$ is eventually bounded by a function in $\mathscr{F}_{\omega^{d-1}\cdot(2^d\cdot k)}$

8 Applications

We use the bounds proven in this paper to provide upper bounds for some problems in automata theory. As a first application, we consider the emptiness problem of incrementing tree counter automata (ITCA) over finite labelled trees [14]. We only provide an informal sketch of the model here. (The reader is referred to [14] for the technical details). Incrementing tree counter automata are finite state automata which operate over trees and have access to counters which it can increment, decrement or test for zero. To avoid undecidability, the counters are also allowed to have *incrementation errors*, i.e., the values of the counters can increase erroneously at any time. Based on the theory of well-structured transition systems, the paper [14] gives a decision procedure for the emptiness problem for ITCA from a given initial configuration. In [4], the authors argue that if the number of states q

and the number of counters k of the given ITCA are fixed and the running time is measured as a function of the inital configuration v_0 of the ITCA, then the running time of this decision procedure could be upper bounded by a function from $\mathscr{F}_{(\omega^k)\cdot q}$. Since the paper [4] uses a different notion of controlled bad sequences compared to ours (and a different well-quasi order than the one constructed in this paper), we first revisit and adapt their analysis to our setting. Then we apply our results to obtain better bounds for the running time.

Let q, k be fixed natural numbers. Recall that Γ_q is the well-quasi order where the domain has q elements such that distinct elements are unordered and the norm of every element is 0. By taking cartesian product of Γ_q with $\mathbb{N}^k$ and then taking the majoring ordering of this resulting construction we get a well-quasi order which we will denote by $(A, \leq_A, |\cdot|_A)$ where $A = \mathbb{P}_f(\Gamma_q \times \mathbb{N}^k)$. Notice that there is a reflection from $(A, \leq_A, |\cdot|_A)$ to the majoring powerset nwqo $(\mathbb{P}_f(\mathbb{N}^k)^q, \leq^{\mathtt{maj}}_{\mathbb{P}_f(\mathbb{N}^k)^q}, |\cdot|_{\mathbb{P}_f(\mathbb{N}^k)^q})$. Indeed suppose $S \in A$. For every $a \in \Gamma_q$, let $S_a := \{v : (a, v) \in S\}$. It is then clear that the mapping $S \to (S_a)_{a\in\Gamma_q}$ is an reflection from $(A, \leq_A, |\cdot|_A)$ to $(\mathbb{P}_f(\mathbb{N}^k)^q, \leq^{\mathtt{maj}}_{\mathbb{P}_f(\mathbb{N}^k)^q}, |\cdot|_{\mathbb{P}_f(\mathbb{N}^k)^q})$. We will use this fact later on.

We now analyse the algorithm given in [14] for testing the emptiness of an ITCA. Let q and k be the number of states and the number of counters of the ITCA respectively. Let $v_0 \in \Gamma_q \times \mathbb{N}^k$ be the given initial configuration. Let $(A, \leq_A, |\cdot|_A)$ be the nwqo on the domain $A = \mathbb{P}_f(\Gamma_q \times \mathbb{N}^k)$ as described above. The algorithm proceeds by constructing a sequence of finite sets $K_0, K_1, \ldots$ where each $K_i \subseteq A$, K_0 is the initial configuration $\{v_0\}$ and $K_{i+1} = K_i \cup Succ(K_i)$ where *Succ* is the successor function between sets of configurations as described in [14]. The algorithm then finds the first m such that $\uparrow K_m = \uparrow K_{m+1}$ and checks if there is an accepting configuration in $\uparrow K_m$. The complexity of the algorithm is mainly dominated by the length of the sequence $K_0, K_1, \ldots, K_m$. Since m is the first index such that $\uparrow K_m = \uparrow K_{m+1}$, we can find a minimal element $x_i \in \uparrow K_{i+1} \setminus \uparrow K_i$ for each $i < m$. Consider the sequence $x_0, \ldots, x_{m-1}$ over A. Noticing that $x_j \not\geq_A x_i$ if $j > i$, we can conclude that $x_0, \ldots, x_{m-1}$ is a bad sequence over A. Further by a careful inspection of the *Succ* relation (as described in [14]) one can easily establish that $x_0, \ldots, x_{m-1}$ is a $(g, |v_0|_A)$-controlled sequence where g is some primitive recursive function depending on q and k. Now since the nwqo $(A, \leq_A, |\cdot|_A)$ has a reflection into $(\mathbb{P}_f(\mathbb{N}^k)^q, \leq^{\mathtt{maj}}_{\mathbb{P}_f(\mathbb{N}^k)^q}, |\cdot|_{\mathbb{P}_f(\mathbb{N}^k)^q})$, we can apply Theorem 7.2 and Proposition 2.14 to get,

Proposition 8.1. The time complexity of the emptiness problem for an ITCA with q states and k counters is bounded by a function in $\mathscr{F}_{(\omega^{k-1})\cdot q}$

As noticed in [4], the authors of [14] also prove the decidability of emptiness for a class of tree automata operating on finite *data trees* called the *alternating top-down tree one register automata* (ATRA), by providing a PSPACE-reduction to the emptiness problem for ITCA. If the original ATRA had q states, then the constructed ITCA has $k(q) = 2^q - 1 + 2^{4q}$ many counters and $f(q) \in O(2^q)$ many states. Hence, we have

Proposition 8.2. The time complexity of the emptiness problem for an ATRA with q states is bounded by a function in $\mathscr{F}_{(\omega^{k(q)-1})\cdot f(q)}$

As a second application, we consider the emptiness problem for another class of finite data tree automata called the *bottom-up alternating one register data tree automata* (BUDA) (See [10] for a complete description of the model). Apart from having a finite number of states Q, the transitions of a BUDA are also defined by a specified finite semigroup S. In [10], the authors prove the decidability of the emptiness problem for BUDA using the theory of well-structured transition systems. Let q and s be the number of states and the size of the finite semigroup of the given BUDA respectively. Let $k = 2^{q+s}$ and $l = 2q^2s^2 + 1$. The authors construct a wsts corresponding to the given BUDA whose set of configurations can be taken to be $(\mathbb{P}_f(\mathbb{N}^k)^{f(k)}$ (where $f(k)$ is some function in $O(2^k)$) with the underlying order being $\leq^{\mathtt{min}}_{\mathbb{P}_f(\mathbb{N}^k)^{f(k)}}$.

A careful analysis of the decision procedure they describe over this wsts reveals that the algorithm constructs a sequence of finite sets $K_0, K_1, \ldots,$ where each $K_i \subseteq (\mathbb{P}_f(\mathbb{N}^k)^{f(k)}$, K_0 is the initial configuration v_0 and $K_{i+1} = K_i \cup Succ(K_i)$ where *Succ* is the successor function between sets of configurations as described by the wsts. The algorithm then finds the first m such that $\uparrow K_m = \uparrow K_{m+l}$ and checks if there is an accepting configuration in $\uparrow K_m$. The complexity of the algorithm is mainly dominated by the length of the sequence $K_0, \ldots, K_m, \ldots, K_{m+l}$. Since m is the first index such that $\uparrow K_m = \uparrow K_{m+l}$, we can find a minimal element $x_i \in \uparrow K_{i+l} \setminus \uparrow K_i$ for each $i < m$. Let p be the largest number such that $pl \leq m+l$. Similar to the analysis performed for the ITCA model, we can conclude that $x_0, x_l, x_{2l}, \ldots, x_{(p-1)l}$ is a $(g, |v_0|_{\mathbb{P}_f(\mathbb{N}^k)^{f(k)}})$-controlled sequence where g is a primitive recursive function depending on k. Applying theorem 7.4 we then get,

Proposition 8.3. The time complexity of the emptiness problem for a BUDA with q states and s elements in the semigroup, is bounded by a function in $\mathscr{F}_{\omega^{k-1}\cdot(2^k\cdot f(k))}$ where $k = 2^{q+s}$.

9 Conclusion

In this paper, we have proved lower and upper bounds for the length of controlled bad sequences for the majoring and minoring ordering over finite sets of $\mathbb{N}^k$. The results were obtained by giving the bounds in terms of functions from Cichon hierarchy and using known complexity results, were translated into bounds over the fast-growing hierarchy. To the best of our knowledge, this is the first upper bound result for length functions over the minoring ordering of $\mathbb{P}_f(\mathbb{N}^d)$. As an application, we used the results to establish upper bounds for the emptiness problems of three types of automata working on trees.

The bounds on the length function for the majoring ordering on $\mathbb{P}_f(\mathbb{N}^k)$ is easily seen to be tight, which solves a problem left open in [4]. However this is not the case with the bounds for minoring ordering and it might be an interesting question in the future to bridge this gap.

Acknowledgments

The author is extremely grateful to Prof. Philippe Schnoebelen and Prof. Sylvain Schmitz of LSV, ENS-Saclay Paris for their mentorship and useful discussions regarding the paper. This work was done when the author was an intern at LSV, ENS-Saclay Paris.

This material is based upon work supported by the French National Research Agency (ANR) grant BRAVAS under grant number ANR-17-CE40-0028 and also by the Indo-French research unit UMI Relax. The author also acknowledges the support provided by the ERC advanced grant PaVeS under grant number 787367.

References

[1] Parosh Aziz Abdulla, Karlis Cerans, Bengt Jonsson, and Yih-Kuen Tsay. 1996. General Decidability Theorems for Infinite-State Systems. In *Proceedings, 11th Annual IEEE Symposium on Logic in Computer Science*. 313–321. https://doi.org/10.1109/LICS.1996.561359

[2] Parosh Aziz Abdulla and Bengt Jonsson. 2001. Ensuring completeness of symbolic verification methods for infinite-state systems. *Theor. Comput. Sci.* 256, 1-2 (2001), 145–167. https://doi.org/10.1016/S0304-3975(00)00105-5

[3] Sergio Abriola and Santiago Figueira. 2014. A note on the order type of minoring orderings and some algebraic properties of ω^2-well quasi-orderings. In *XL Latin American Computing Conference, CLEI 2014*. 1–9. https://doi.org/10.1109/CLEI.2014.6965188

[4] Sergio Abriola, Santiago Figueira, and Gabriel Senno. 2015. Linearizing well quasi-orders and bounding the length of bad sequences. *Theor. Comput. Sci.* 603 (2015), 3–22. https://doi.org/10.1016/j.tcs.2015.07.012

[5] Joseph B Kruskal. 1972. The Theory of Well-Quasi-Ordering: A Frequently Discovered Concept. *Journal of Combinatorial Theory, Series A* 13 (11 1972), 297–305. https://doi.org/10.1016/0097-3165(72)90063-5

[6] Pierre Chambart and Philippe Schnoebelen. 2008. The Ordinal Recursive Complexity of Lossy Channel Systems. In *Proceedings of the Twenty-Third Annual IEEE Symposium on Logic in Computer Science, LICS 2008, 24-27 June 2008, Pittsburgh, PA, USA*. 205–216. https://doi.org/10.1109/LICS.2008.47

[7] E.A. Cichon and E.Tahhan Bittar. 1998. Ordinal recursive bounds for Higman's theorem. *Theoretical Computer Science* 201, 1 (1998), 63 – 84. https://doi.org/10.1016/S0304-3975(97)00009-1

[8] Nachum Dershowitz and Zohar Manna. 1979. Proving Termination with Multiset Orderings. *Commun. ACM* 22, 8 (1979), 465–476. https://doi.org/10.1145/359138.359142

[9] Michael R. Fellows and Michael A. Langston. 1988. Nonconstructive tools for proving polynomial-time decidability. *J. ACM* 35, 3 (1988), 727–739. https://doi.org/10.1145/44483.44491

[10] Diego Figueira. 2010. *Reasoning on words and trees with data*. Ph.D. Dissertation. École normale supérieure de Cachan, France. https://tel.archives-ouvertes.fr/tel-00718605

[11] Diego Figueira, Santiago Figueira, Sylvain Schmitz, and Philippe Schnoebelen. 2011. Ackermannian and Primitive-Recursive Bounds with Dickson's Lemma. In *Proceedings of the 26th Annual IEEE Symposium on Logic in Computer Science*. 269–278. https://doi.org/10.1109/LICS.2011.39

[12] Alain Finkel and Philippe Schnoebelen. 2001. Well-structured transition systems everywhere! *Theor. Comput. Sci.* 256, 1-2 (2001), 63–92. https://doi.org/10.1016/S0304-3975(00)00102-X

[13] Graham Higman. 1952. Ordering by Divisibility in Abstract Algebras. *Proceedings of the London Mathematical Society* s3-2, 1 (1952), 326–336. https://doi.org/10.1112/plms/s3-2.1.326

[14] Marcin Jurdzinski and Ranko Lazic. 2011. Alternating automata on data trees and XPath satisfiability. *ACM Trans. Comput. Log.* 12, 3 (2011), 19:1–19:21. https://doi.org/10.1145/1929954.1929956

[15] M. H. Löb and S. S. Wainer. 1970. Hierarchies of number-theoretic functions. I. *Archiv für mathematische Logik und Grundlagenforschung* 13, 1 (Mar 1970), 39–51. https://doi.org/10.1007/BF01967649

[16] Bojan Mohar. 1996. Embedding Graphs in an Arbitrary Surface in Linear Time. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing*. 392–397. https://doi.org/10.1145/237814.237986

[17] Fernando Rosa-Velardo. 2017. Ordinal recursive complexity of Unordered Data Nets. *Information and Computation* 254 (2017), 41 – 58. https://doi.org/10.1016/j.ic.2017.02.002

[18] Sylvain Schmitz. 2014. Complexity Bounds for Ordinal-Based Termination - (Invited Talk). In *Reachability Problems - 8th International Workshop, RP 2014*. 1–19. https://doi.org/10.1007/978-3-319-11439-2_1

[19] Sylvain Schmitz. 2016. Complexity Hierarchies beyond Elementary. *TOCT* 8, 1 (2016), 3:1–3:36. https://doi.org/10.1145/2858784

[20] Sylvain Schmitz and Philippe Schnoebelen. 2011. Multiply-Recursive Upper Bounds with Higman's Lemma. In *Automata, Languages and Programming - 38th International Colloquium, ICALP 2011*. 441–452. https://doi.org/10.1007/978-3-642-22012-8_35

Appendix B

Complexity of Coverability in Bounded Path Broadcast Networks (FSTTCS 2021)

This section contains a reprinting of the following paper, which has been published as a peer-reviewed conference paper.

> A. R. Balasubramanian. *Complexity of Coverability in Bounded Path Broadcast Networks.* In conference proceedings of FSTTCS 2021. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. Vol. 213 of LIPIcs, Pages - 35:1-35:16. doi: `10.4230/LIPIcs.FSTTCS.2021.35`

According to the Open Access Policy of LIPIcs (Leibniz International Proceedings in Informatics) by Schloss Dagstuhl Leibniz-Zentrum für Informatik, the author of this thesis is permitted to include the above paper in this thesis. The relevant excerpt is the following:

> LIPIcs volumes are peer-reviewed and published according to the principle of OpenAccess, i.e., they are available online and free of charge. The authors retain their copyright.

For more information, please see `https://www.dagstuhl.de/en/publishing/series/details/LIPIcs`, in particular, the section on Open Access Policy.

Summary

Broadcast networks are a formalism of distributed computation that allows one to model networks of identical nodes communicating through message broadcasts over a communication topology that does not change over the course of executions. The parameterized verification problem for these networks amounts

to proving the correctness of a property for any number of nodes and on all executions. Dually speaking, this problem asks for the existence of an execution of the broadcast network that violates a given property. One specific instance of parameterized verification is the coverability problem which asks whether there is an execution of the network in which some node reaches a given state of the broadcast protocol. This problem is known to be undecidable. Further, if we additionally assume that the underlying communication topology has a bound on the longest path, then the coverability problem is known to be decidable. In this work, we precisely characterize the complexity of the coverability problem for bounded-path topologies and prove that it is $\mathbf{F}_{\epsilon_0}$-complete, where $\mathbf{F}_{\epsilon_0}$ is a class in the fast-growing hierarchy of complexity classes.

Contributions of the author of this thesis

I am the sole author of this paper.

Complexity of Coverability in Bounded Path Broadcast Networks

A. R. Balasubramanian
Technische Universität München, Germany

Abstract

Broadcast networks are a formalism of distributed computation that allow one to model networks of identical nodes communicating through message broadcasts over a communication topology that does not change over the course of executions. The parameterized verification problem for these networks amounts to proving correctness of a property for any number of nodes, and on all executions. Dually speaking, this problem asks for the existence of an execution of the broadcast network that violates a given property. One specific instance of parameterized verification is the coverability problem which asks whether there is an execution of the network in which some node reaches a given state of the broadcast protocol. This problem was proven to be undecidable by Delzanno, Sangnier and Zavattaro (CONCUR 2010). In the same paper, the authors also prove that, if we additionally assume that the underlying communication topology has a bound on the longest path, then the coverability problem becomes decidable.

In this paper, we provide complexity results for the above problem and prove that the coverability problem for bounded-path topologies is $\mathbf{F}_{\epsilon_0}$-complete, where $\mathbf{F}_{\epsilon_0}$ is a class in the fast-growing hierarchy of complexity classes. This solves an open problem of Hasse, Schmitz and Schnoebelen (LMCS, Vol 10, Issue 4).

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness; Theory of computation → Distributed computing models

Keywords and phrases Parameterized verification, Bounded path networks, Fast-growing complexity classes

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2021.35

Funding *A. R. Balasubramanian*: Supported by funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme under grant agreement No 787367 (PaVeS).

1 Introduction

In recent years, significant effort has been put into understanding the precise computational complexity of problems which are *non-elementary*, i.e., problems whose running times cannot be upper bounded by any fixed tower of exponentials of the input size [13, 6, 20, 19, 1, 18, 8]. A well-known such problem is the satisfiability problem of the weak monadic theory of one successor (WS1S) [17]. A more recent addition to this collection is the reachability problem for Petri nets [7]. We refer the reader to the excellent survey by Schmitz [19] for a collection of various non-elementary problems from logic, automata theory and verification which have been proven to be complete for appropriate complexity classes in the *fast-growing hierarchy*. This hierarchy allows for a finer classification of problems lying beyond the elementary regime.

From a tractability perspective, these results are of course negative. However, there are non-elementary problems for which tools have been developed, for e.g. MONA for WS1S [11]; and considerable effort has been put into the development of fast heuristics to solve some non-elementary problems on realistic inputs, for e.g., there is a huge wealth of heuristics and special cases which have been studied for the Petri net reachability problem [3, 12, 14, 4, 5, 15]. Hence, understanding the precise complexity of a non-elementary problem can help us to solve it in practice by reducing it to various other well-studied non-elementary problems.

The fast-growing hierarchy mentioned above can help us in this goal of understanding the computational complexity of non-elementary problems. Proving a problem to be hard for one of these classes implies that that problem cannot have an efficient encoding into any of the non-elementary problems which lie strictly below this class. In their invited paper for CONCUR 2013 [21], Schmitz and Schnoebelen explicity state the program of populating the catalog of hard problems for classes in the fast-growing hierarchy, so that hardness proofs do not have to begin from Turing machines, but can instead rely on simpler reductions.

In this paper, we contribute to this program by considering a problem from the parameterized verification of broadcast networks and proving that it is $\mathbf{F}_{\epsilon_0}$-complete, where $\mathbf{F}_{\epsilon_0}$ is a complexity class in the fast-growing hierarchy. We now offer a brief overview of broadcast networks [9, 2]. Broadcast networks are a formalism of distributed computation that allow one to model networks of *identical nodes* communicating through message broadcasts. Each node runs the same protocol and an underlying communication topology specifies for each node, the set of neighbors that it can broadcast messages to. This topology remains invariant over the course of executions of the network. At any point, a node can broadcast a message which is received by all of its neighbors.

The parameterized verification problem for these networks amounts to proving correctness of a property for any number of nodes and over any communication topology. Dually, we ask for the existence of an execution of the network that violates a given property. One specific instance of parameterized verification is the coverability problem which asks whether there is an execution of the network in which some node reaches a given state of the broadcast protocol. This problem was proven to be undecidable by Delzanno, Sangnier and Zavattaro (Theorem 1 of [9]). In the same paper, the authors also prove that, if we additionally assume that the underlying communication topology has a bound on the longest path (bounded-path topologies), then the coverability problem becomes decidable (Theorem 5 of [9]). Our main result in this paper is that the coverability problem for bounded-path topologies is $\mathbf{F}_{\epsilon_0}$-complete, where $\mathbf{F}_{\epsilon_0}$ is a class in the aforementioned fast-growing hierarchy of complexity classes.

Our result settles a conjecture raised by Hasse, Schmitz and Schnoebelen (Section 8.3 of [16]) and also settles the complexity of the last remaining question from the original paper that initiated the study of parameterized verification problems for broadcast networks [9]. Moreover, we provide a new and rather natural problem to the list of $\mathbf{F}_{\epsilon_0}$-complete problems, which when compared to the list of $\mathbf{F}_{\omega}$-complete and $\mathbf{F}_{\omega^\omega}$-complete problems, is rather small currently (Section 6.4 of [19]). (Both $\mathbf{F}_{\omega}$ and $\mathbf{F}_{\omega^\omega}$ are classes in the fast-growing hierarchy which are much smaller than $\mathbf{F}_{\epsilon_0}$). Hence, in this sense, we contribute to the above-mentioned program of finding hard problems for classes in the fast-growing hierarchy. Further, we hope that the present work might prove to be useful in settling the complexity of other problems conjectured to be $\mathbf{F}_{\epsilon_0}$-complete (Section 8.3 of [16]), since all the problems mentioned there are concerned with infinite-state systems regarding bounded-path trees and graphs, and so those problems are in some sense “close” to the problem that we consider here.

2 Preliminaries

In this section, we recall the model of broadcast networks as defined in [2]. Intuitively, a broadcast network consists of several nodes, each executing the same finite-state *broadcast protocol*. A communication topology assigns to each node, a finite set of neighbors, to which it can communicate. At any point, some node can broadcast a message which is received by all of its neighbors. We now proceed to formalize this intuition.

Broadcast networks

▶ **Definition 1.** *A broadcast protocol is a tuple $\mathcal{P} = (Q, I, \Sigma, \Delta)$ where Q is a finite set of states, $I \subseteq Q$ is the set of initial states, Σ is a finite set of messages and $\Delta \subseteq Q \times \{!a, ?a, : a \in \Sigma\} \times Q$ is the transition relation.*

For ease of notation, we will write $q \xrightarrow{!a} q'$ (resp. $q \xrightarrow{?a} q'$) for $(q, !a, q') \in \Delta$ (resp. $(q, ?a, q') \in \Delta$). A transition $q \xrightarrow{!a} q'$ (resp. $q \xrightarrow{?a} q'$) intuitively corresponds to broadcasting (resp. receiving) the message a. We will assume that broadcast protocols are complete, i.e. for every state q and every message a there exists q' such that $q \xrightarrow{?a} q'$.

As mentioned before, a broadcast network consists of several identical *nodes* running a broadcast protocol and each node has a finite set of neighbors. To formalize this, given a broadcast protocol $\mathcal{P} = (Q, I, \Sigma, \Delta)$, a *configuration* of $\mathcal{P}$ is a labelled graph $\gamma = (\mathsf{N}, \mathsf{E}, \mathsf{L})$ where N is a finite set of nodes, $\mathsf{E} \subseteq \mathsf{N} \times \mathsf{N}$ is a finite set of (undirected) edges specifying for every pair of nodes whether or not there is a communication link between them and $\mathsf{L} : \mathsf{N} \to Q$ is a labelling function that specifies the current state of each node. We let $\mathsf{L}(\gamma) = \{\mathsf{L}(\mathsf{n}) : \mathsf{n} \in \mathsf{N}\}$ be the set of labels appearing in the nodes of γ. We say that γ is *initial* if $\mathsf{L}(\gamma) \subseteq I$.

The semantics of the broadcast network of a protocol $\mathcal{P}$ is given by means of an infinite-state transition system $\mathcal{T}(\mathcal{P})$ which consists of all the configurations of the protocol $\mathcal{P}$. There is a step from the configuration $\gamma = (\mathsf{N}, \mathsf{E}, \mathsf{L})$ to the configuration $\gamma' = (\mathsf{N}', \mathsf{E}', \mathsf{L}')$ if $\mathsf{N}' = \mathsf{N}$, $\mathsf{E}' = \mathsf{E}$ and there exists a node n and a message $a \in \Sigma$ such that $(\mathsf{L}(\mathsf{n}), !a, \mathsf{L}'(\mathsf{n})) \in \Delta$, and for every other node n', if $(\mathsf{n}, \mathsf{n}') \in \mathsf{E}$, then $(\mathsf{L}(\mathsf{n}), ?a, \mathsf{L}'(\mathsf{n}')) \in \Delta$; otherwise $\mathsf{L}(\mathsf{n}') = \mathsf{L}'(\mathsf{n}')$. In this case, we write $\gamma \xrightarrow{\mathsf{n},a} \gamma'$ or simply $\gamma \to \gamma'$. Intuitively, a step consists of a node n broadcasting some message a which is then received by *all* of its neighbors; all the other nodes do nothing. Notice that between steps, the set of nodes and edges do not change.

A *run* from the configuration γ to the configuation γ' is a sequence of steps $\gamma \to \gamma_1 \to \gamma_2 \to \dots \gamma_{k-1} \to \gamma'$. If a run exists between configurations γ and γ' we denote it by $\gamma \xrightarrow{*} \gamma'$. An *execution* is a run starting from an initial configuration.

Given a state f and a configuration γ we say that γ covers f if $f \in \mathsf{L}(\gamma)$, i.e., if the state of some node in γ is f. We say that an execution $\gamma_0 \xrightarrow{*} \gamma$ covers f, if γ covers f. The *coverability* problem for broadcast protocols is to decide, given a broadcast protocol $\mathcal{P}$ and a state f, whether there is an execution from some initial configuration that covers f. It is known that the coverability problem is undecidable (Theorem 1 of [9]).

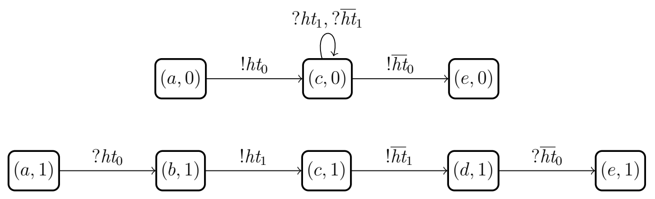

Figure 1 Example of a broadcast protocol where we set $I = \{(a,0),(a,1)\}$ and $\Sigma = \{ht_i, \overline{ht}_i : 0 \le i \le 1\}$. If for a state (f,i), we have not depicted what happens when message m is received at (f,i), we assume that $(f,i) \xrightarrow{?m} (\bot, i)$. Here $(\bot,0)$ and $(\bot,1)$ are new *sink* states, i.e., states with no outgoing transition.

▶ **Example 2.** We consider the broadcast protocol given in Figure 1. Figure 2 shows an execution in this protocol covering the state $(e,0)$. Moreover, let $\gamma = (\mathsf{N}, \mathsf{E}, \mathsf{L})$ be *any* initial configuration and $\gamma' = (\mathsf{N}', \mathsf{E}', \mathsf{L}')$ be *any* configuration covering $(e,0)$ such that $\gamma \xrightarrow{*} \gamma'$.

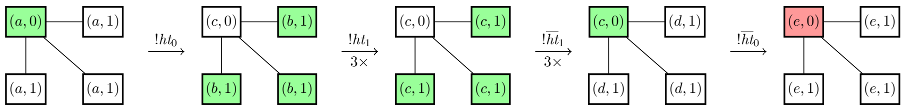

Figure 2 Example of an execution covering $(e, 0)$ in the broadcast protocol given in Figure 1. The nodes marked in green make the broadcasts, i.e., first the node on the topmost left broadcasts ht_0, then all the other nodes broadcast ht_1 in some order, and then $\overline{ht}_1$ in some order, and then the node on the topmost left broadcasts $\overline{ht}_0$.

Hence, there is a node n such that $\mathsf{L}'(\mathsf{n}) = (e, 0)$. Note that $\mathsf{L}(\mathsf{n})$ must be $(a, 0)$. Hence n must have broadcasted both ht_0 and $\overline{ht}_0$ to move into the states $(c, 0)$ and $(e, 0)$ at different points during the run. This means that all of the neighbors of n received $\overline{ht}_0$ at some point, and so the labels of all of its neighbors in γ' must be either $(e, 1)$ or $(\perp, 0)$ or $(\perp, 1)$.

Suppose n' is a neighbor of n such that $\mathsf{L}'(\mathsf{n}') = (e, 1)$. Notice that if there is a neighbor $\mathsf{n}'' \neq \mathsf{n}$ of n' which was at $(c, 0)$ during some point in the run, then n'' must have broadcasted ht_0 during the run. However, then n' would have received two ht_0 messages, which would have caused it to move into either $(\perp, 0)$ or $(\perp, 1)$. Hence, there is exactly one neighbor of n' which was labelled by $(c, 0)$ at some point during the run.

This protocol along with the above discussion will prove useful later on for the lower bound reductions in section 5.

Bounded-path broadcast networks

Motivated by the undecidability of the coverability problem, the authors of [9] also study a different variant of the problem, which we now describe.

Let $\mathcal{P}$ be a broadcast protocol and let $k \geq 1$ be some number. Let γ be a configuration of $\mathcal{P}$. We say that γ is k-path bounded if the length of the *longest* simple path in γ is at most k. Now, let $\mathcal{T}_k(\mathcal{P})$ be the restriction of the transition system $\mathcal{T}(\mathcal{P})$ to only k-path bounded configurations. Notice that since the set of nodes and edges do not change during a run, $\mathcal{T}_k(\mathcal{P})$ is closed under the step relation. The *path bounded* coverability problem (Bounded-Path-Cover) is then defined as follows:

Given: A broadcast protocol $\mathcal{P} = (Q, I, \Sigma, \Delta)$, a state $f \in Q$ and a number k.
Decide: If there is an execution in $\mathcal{T}_k(\mathcal{P})$ which covers f.

The authors of [9] prove that this problem is decidable (Theorem 5 of [9]). The main result that we prove in this paper is that

▶ **Theorem 3.** *Bounded-Path-Cover is* $\mathbf{F}_{\epsilon_0}$*-complete.*

Here $\mathbf{F}_{\epsilon_0}$ is a member of the *fast-growing* complexity class hierarchy. We refer the reader to Section 2.3 of [19] for a description of the fast-growing hierarchy and the class $\mathbf{F}_{\epsilon_0}$. To prove the upper bound for our problem, we will consider the algorithm given in [9] and analyze its running time by means of *controlled-bad* sequences of a suitable well-quasi order, whose upper bounds will allow us to place Bounded-Path-Cover in the complexity class $\mathbf{F}_{\epsilon_0}$. The lower bound is proved by giving a logspace reduction from a known $\mathbf{F}_{\epsilon_0}$-hard problem, which we now proceed to describe.

3 Nested counter systems (NCS)

A nested counter system is a generalisation of a usual counter system with *higher-order counters*, i.e., counters which can themselves contain other (lower-order) counters. Intuitively, an one-dimensional counter is a usual counter, which can either add or subtract 1. A two-dimensional counter can either add or remove an one-dimensional counter, a three-dimensional counter can either add or remove a two-dimensional counter and so on. Here, we slightly alter the definition of nested counter systems as given in [8] so that it better suits our purposes. It can be easily verified that our altered definition does not affect the semantics of the system as given in [8].

A k-nested counter system (k-NCS) is a tuple $\mathcal{N} = (Q, \delta)$ where Q is a finite set of *states* and $\delta \subseteq \bigcup_{1 \le i,j \le k+1} (Q^i \times Q^j)$ is a set of *rules*. The set $\mathcal{C}_\mathcal{N}$ of *configurations* of $\mathcal{N}$ is defined to be the set of all labelled rooted trees of height atmost k, with labels from the set Q.

The operational semantics of $\mathcal{N}$ is defined in terms of the following transition relation $\rightarrow \subseteq \mathcal{C}_\mathcal{N} \times \mathcal{C}_\mathcal{N}$ on configurations: Let $r := ((q_0, \dots, q_i), (q'_0, \dots, q'_j)) \in \delta$ be a rule with $i \le j \le k$. We say that a configuration C can move to the configuration C' using the rule r (denoted by $C \xrightarrow{r} C'$), if there is a path $v_0, v_1 \dots, v_i$ in C starting at the root such that for every $0 \le l \le i$, the label of v_l is q_l and, C' is obtained from C by 1) for every $0 \le l \le i$, changing the label of each v_l to q'_l and 2) for every $i+1 \le l \le j$, creating a new vertex v_l with label q'_l and adding it as a child to v_{l-1}.

Similarly, suppose $r := ((q_0, \dots, q_i), (q'_0, \dots, q'_j)) \in \delta$ is a rule with $j < i \le k$. Then $C \xrightarrow{r} C'$ if there is a path $v_0, v_1, \dots, v_i$ in C starting at the root such that for every $0 \le l \le i$, the label of v_l is q_l and, C' is obtained from C by 1) for every $0 \le l \le j$, changing the label of each v_l to q'_l and 2) removing the subtree rooted at the node v_{j+1}.

▶ **Example 4.** Let us consider the NCS $\mathcal{N}$ given by the states $Q = \{p_i, p'_i, q_i, q'_i : 0 \le i \le 4\}$ and consisting of the following rules: $r_1 = ((q_0, q_1), (q'_0, q'_1, q'_2)), r_2 = ((q'_0, q_3, q_2), (p_0)), r_3 = ((p_0), (p'_0))$. In Figure 3, we illustrate the application of these rules to a configuration of $\mathcal{N}$.

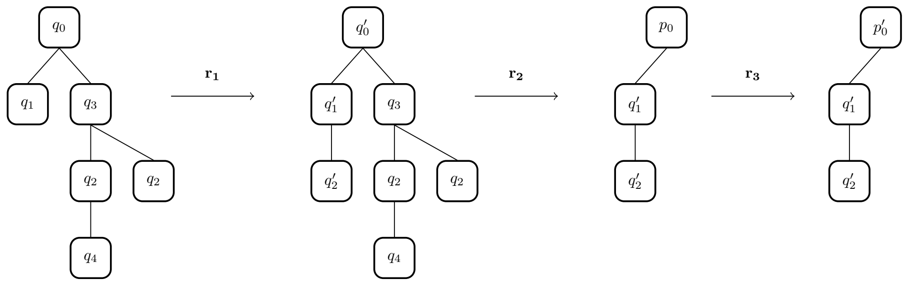

Figure 3 Application of the rules r_1, r_2 and r_3 to a configuration of $\mathcal{N}$, which is described in Example 4.

We say that $C \rightarrow C'$ if $C \xrightarrow{r} C'$ for some rule r. We let $\xrightarrow{*}$ denote the reflexive and transitive closure of $\rightarrow$ and we say that a configuration C reaches C' if $C \xrightarrow{*} C'$. Given two states $q_{in}, q_f \in Q$, we say that q_{in} can cover q_f if the (unique) configuration consisting of the single root vertex labelled with q_{in} can reach *some* configuration where the root is labelled by q_f. The coverability problem for an NCS is then the following: Given an NCS $\mathcal{N}$ and two states q_{in}, q_f, can q_{in} cover q_f? It is known that the coverability problem is $\mathbf{F}_{\epsilon_0}$-hard (Theorem 7 of [8]).

Lossy semantics. In addition to the "usual" semantics of an NCS that we have described in the previous section, we also need a *lossy semantics* which we now define here. Let $\mathcal{N} = (Q, \delta)$ be a k-NCS and let $q_{in}, q_f \in Q$. We say that there is a *lossy step* between configurations C and C', if C' can be obtained from C by deleting the subtree rooted at some vertex v in C. We let $C \dashrightarrow C'$ if either there is a lossy step between C and C' or $C \xrightarrow{r} C'$ for some rule r. As usual, we let $\overset{*}{\dashrightarrow}$ denote the reflexive and transitive closure of $\dashrightarrow$ and we say that C can reach C' in a lossy manner if $C \overset{*}{\dashrightarrow} C'$. We can then define the notion of the state q_{in} covering the state q_f in a straightforward manner.

For configurations C, C', we say that $C \geq C'$ iff C' can be obtained from C by a sequence of lossy steps. Since NCS do not have any zero tests, from the definition of the transition relation, we can easily infer the following proposition.

▶ **Proposition 5.** *If $C_1 \geq C_1'$ and $C_1' \overset{*}{\dashrightarrow} C_2'$ then there exists $C_2 \geq C_2'$ such that $C_1 \xrightarrow{*} C_2$.*

Hence, we get the following corollary.

▶ **Corollary 6.** *q_f can be covered from q_{in} in a lossy manner iff q_f can be covered from q_{in} under the usual semantics.*

This corollary will be useful later on in order to prove our hardness result.

4 A simulator protocol $\mathcal{P}_{\text{sim}}$

Throughout this section, let $\mathcal{N} = (Q, \delta)$ be a fixed k-NCS with two fixed states q_{in} and q_f. In this section, we will construct a broadcast protocol $\mathcal{P}_{\text{sim}} = (Q_{\text{sim}}, I_{\text{sim}}, \Sigma_{\text{sim}}, \delta_{\text{sim}})$, a state p of $\mathcal{P}_{\text{sim}}$, and define a notion of *good initial configurations* of $\mathcal{P}_{\text{sim}}$ such that the following property is satisfied: q_f can be covered from q_{in} in the NCS $\mathcal{N}$ iff p can be covered in $\mathcal{T}_{2k}(\mathcal{P}_{\text{sim}})$ by some execution starting at a *good initial configuration*. Intuitively, the protocol $\mathcal{P}_{\text{sim}}$ will *simulate* the NCS $\mathcal{N}$, *provided* that the initial configuration that it begins with is a good initial configuration.

States, alphabet and good configurations. For each $0 \leq i \leq k$, $\mathcal{P}_{\text{sim}}$ will have two states $(\mathit{start}, i), (\mathit{finish}, i)$. For each $0 \leq i \leq k$ and each $r \in \delta$, we will have five states $(\texttt{req-rec}[r], i), (\texttt{req-fwd}[r], i), (\texttt{wait}[r], i), (\texttt{ack-rec}[r], i)$, $(\texttt{ack-fwd}[r], i)$. Finally, for each $0 \leq i \leq k$ and each $q \in Q$, $\mathcal{P}_{\text{sim}}$ will have a state (q, i). Notice that each state of $\mathcal{P}_{\text{sim}}$ is of the form (a, b) where $a \in Q \cup \{\mathit{start}, \mathit{finish}\} \cup \{\texttt{req-rec}[r], \texttt{req-fwd}[r], \texttt{wait}[r], \texttt{ack-rec}[r], \texttt{ack-fwd}[r]\}$ and $0 \leq b \leq k$. The first part "a" will be called the *base* of the state and the second part "b" will be called the *grade*. Sometimes we will abuse notation and refer to the base (resp. grade) of a node in a configuration to mean the base (resp. grade) of the label of that node.

The initial set of states I_{sim} will be the set $\{(q_{in}, 0)\} \cup \{(\mathit{start}, i) : 1 \leq i \leq k\}$. (The asymmetry in the initial set of states between the case of 0 and others will be discussed in the following paragraphs). The alphabet Σ_{sim} will be the set $\{\mathit{begin}_i^r, \mathit{end}_i^r : r \in \delta, 0 \leq i \leq k\}$.

A configuration γ of $\mathcal{P}_{\text{sim}}$ is called *good* if γ is a tree of height at most k such that 1) the base of the label of every node is in the set $Q \cup \{\mathit{start}, \mathit{finish}\}$, 2) there is exactly one node n labelled by a state of grade 0, which will be called the root of γ and, 3) every node at distance i from n is labelled by a state of grade i. Notice that if γ is a good initial configuration then $\gamma \in \mathcal{T}_{2k}(\mathcal{P})$. Further, notice that in a good initial configuration, the root must be labelled by $(q_{in}, 0)$ and every node at distance i from the root is labelled by (start, i).

Intuition behind good configurations of $\mathcal{P}_{\text{sim}}$. Before we describe the transition relation of $\mathcal{P}_{\text{sim}}$, we describe some intuition behind the notion of a good configuration.

Let γ be a good configuration of $\mathcal{P}_{\text{sim}}$. Notice that there is a way to map γ to a configuration of $\mathcal{N}$: First, forget all the grades from the labels of each node in γ and just keep the bases. Next, remove all nodes whose label is either *start* or *finish* and from the resulting forest, pick the tree T containing the root. In this way, to every good configuration γ of $\mathcal{P}_{\text{sim}}$ we can define a configuration $\mathbb{E}(\gamma)$ of $\mathcal{N}$. Hence, we can use good configurations of $\mathcal{P}_{\text{sim}}$ to encode configurations of $\mathcal{N}$ and this is the reason behind defining good configurations of $\mathcal{P}_{\text{sim}}$. An example of this mapping is given in Figure 4.

Further, notice that if γ is any good initial configuration, then $\mathbb{E}(\gamma)$ is the initial configuration of $\mathcal{N}$. This is the reason behind the asymmetry in the definition of the initial set of states between the case of 0 and others.

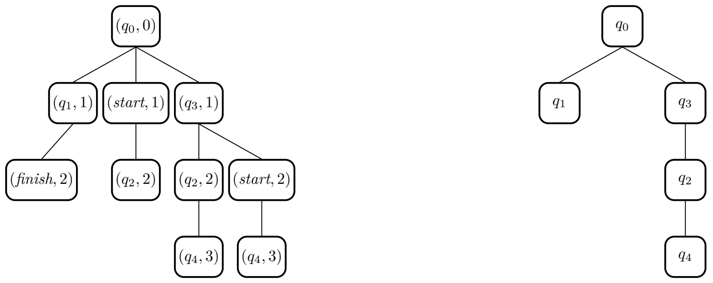

Figure 4 An example of the map $\mathbb{E}$ between good configurations of $\mathcal{P}$ and configurations of $\mathcal{N}$. On the left is a good configuration γ of $\mathcal{P}$ and on the right is its corresponding mapped configuration $\mathbb{E}(\gamma)$ of $\mathcal{N}$.

4.1 Transitions involving the letters $begin_i^r$ and end_i^r

For the rest of this section, let us fix a rule $r = ((q_0, \ldots, q_i), (q'_0, \ldots, q'_j)) \in \delta$ where $i, j \leq k$ and let $w = \max(i, j)$. For the sake of uniformity, if $i < j$, then let $q_l = start$ for every $i < l \leq j$. If $i > j$, then let $q'_l = finish$ for every $j < l \leq i$.

Intuitively, the gadget that we will demonstrate will use the messages $begin_i^r$ and end_i^r to find a path $\mathsf{n}_0, \ldots, \mathsf{n}_w$ labelled by $(q_0, 0), (q_1, 1), \ldots, (q_w, w)$ and then change the labels along this path to $(q'_0, 0), (q'_1, 1), \ldots, (q'_w, w)$. Notice that if $i \leq j$, this means that a path of the form $(q_0, 0), \ldots, (q_i, i), (start, i+1), \ldots, (start, j)$ becomes $(q'_0, 0), \ldots, (q'_i, i), (q'_{i+1}, i+1), \ldots, (q'_j, j)$. Similarly, if $i > j$ then a path of the form $(q_0, 0), \ldots, (q_j, j), \ldots (q_i, i)$ becomes $(q'_0, 0), \ldots, (q'_j, j), (finish, j+1), \ldots, (finish, i)$. This would then allow us to simulate the rule r on good configurations of $\mathcal{P}_{\text{sim}}$.

Formally, we now describe the transitions involving the letters $\{begin_i^r, end_i^r : 0 \leq i \leq k\}$. First, we make a small remark:

▶ Remark 7. In the following, if we do not specify what happens upon receiving a message m from a state with base a and grade b, then it is to be assumed that $(a, b) \xrightarrow{?m} (finish, b)$.

The "gadget" for "simulating" the rule r. We now present the main transitions involving the messages $begin_i^r$ and end_i^r.

- First, we have four transitions

$$(q_0,0) \xrightarrow{!begin_0^r} (\texttt{req-fwd}[r],0) \xrightarrow{?begin_1^r} (\texttt{wait}[r],0) \xrightarrow{?end_1^r} (\texttt{ack-rec}[r],0) \xrightarrow{!end_0^r} (q_0',0)$$

- Then, for every $1 \le l \le w-1$, we have

$$(q_l,l) \xrightarrow{?begin_{l-1}^r} (\texttt{req-rec}[r],l) \xrightarrow{!begin_l^r} (\texttt{req-fwd}[r],l) \xrightarrow{?begin_{l+1}^r} (\texttt{wait}[r],l) \xrightarrow{?end_{l+1}^r}$$
$$\longrightarrow (\texttt{ack-rec}[r],l) \xrightarrow{!end_l^r} (\texttt{ack-fwd}[r],l) \xrightarrow{?end_{l-1}^r} (q_l',l)$$

- Finally, we have four transitions

$$(q_w,w) \xrightarrow{?begin_{w-1}^r} (\texttt{req-rec}[r],w) \xrightarrow{!begin_w^r} (\texttt{wait}[r],w) \xrightarrow{!end_w^r} (\texttt{ack-fwd}[r],w) \xrightarrow{?end_{w-1}^r} (q_w',w)$$

Self-loops. While the previous gadget comprised the main transitions involving $begin_i^r$ and end_i^r, for technical reasons we need the following self-loop transitions as well: For every state with base $a \in Q \cup \{start, finish\}$ and grade $1 \le i \le k$, there are two transitions $(a,i) \xrightarrow{?begin_{i-1}^r} (a,i)$ and $(a,i) \xrightarrow{?end_{i-1}^r} (a,i)$.

This finishes our description of the transition relation of $\mathcal{P}_{\text{sim}}$.

Intuition behind the transitions. We now give a brief intuition behind the gadget in the case of $w = 2$. Notice that only the root n_0 in a good configuration can be labelled by $(q_0,0)$. Hence if n_0 broadcasts $begin_0^r$, it is *forwarding* its request of wanting to simulate the rule r to its children. The children have two choices: either stay where they are by means of the self-loops or *receive* the request and move to $(\texttt{req-rec}[r],1)$. Atleast one child n_1 has to receive the request and move, otherwise the configuration enters into a deadlock. From $(\texttt{req-rec}[r],1)$ n_1 can *forward* this request to its children by broadcasting $begin_1^r$ (and also let n_0 know that is has received its request, whereby it enters a waiting mode). Notice that if two children of n_0 forward the request, then n_0 will enter $(finish,0)$ and the simulation of the rule r cannot happen. Similarly, some child n_2 of n_1 must receive the request of n_1, move to $(\texttt{req-rec}[r],2)$, then broadcast $begin_2^r$. At this point, the base of each n_i is $\texttt{wait}[r]$.

Now n_2 can broadcast end_2^r, forwarding an *acknowledgment* to the request made by n_1. n_1 can receive this acknowledgment and broadcast end_1^r, forwarding an acknowledgment to n_0 which can broadcast end_0^r and move to $(q_0',0)$. At this point, the labels of $\mathsf{n}_0, \mathsf{n}_1$ and n_2 are $(q_0',0),(q_1',1)$ and $(q_2',2)$ respectively, which means that we have changed the labels along a path from $(q_0,0),(q_1,1)$ and $(q_2,2)$ to $(q_0',0),(q_1',1)$ and $(q_2',2)$.

4.2 Proof of correctness

The following lemma tells us that we can use good configurations of $\mathcal{P}_{\text{sim}}$ along with the gadget for the rule r described in the previous section to simulate steps of $\mathcal{N}$.

▶ **Lemma 8** ($\mathcal{P}_{\mathsf{sim}}$ simulates $\mathcal{N}$). *Suppose $C \xrightarrow{r} C'$ is a step in the NCS $\mathcal{N}$. Suppose γ is a good configuration such that 1) $\mathbb{E}(\gamma) = C$ and, 2) there is a path $\mathsf{n}_0, \ldots, \mathsf{n}_w$ in γ where the label of each n_l is (q_l,l). Then there is a good configuration γ' with $\gamma \xrightarrow{*} \gamma'$ such that 1) $\mathbb{E}(\gamma') = C'$ and, 2) γ' is the same as γ except the label of each n_l is (q_l',l).*

Proof sketch. For ease of presentation, we provide the proof in the case of $w = 2$. This proof can be generalized to any w in a straightforward manner.

The proof for $w = 2$ is essentially the same argument that is given in the intuition paragraph. Throughout the run that we are going to describe, if a node $\mathsf{n} \notin \{\mathsf{n}_0, \mathsf{n}_1, \mathsf{n}_2\}$ receives a message, then it will always take the self-loop transitions that we have constructed in the gadget for the rule r.

From γ, n_0 broadcasts $begin_0^r$ and moves to $(\texttt{req-fwd}[r], 0)$ and n_1 receives it and moves to $(\texttt{req-rec}[r], 1)$. Then, n_1 broadcasts $begin_1^r$ and moves to $(\texttt{req-fwd}[r], 1)$ and n_0 and n_2 receive it and move to $(\texttt{wait}[r], 0)$ and $(\texttt{req-rec}[r], 2)$ respectively. Then, n_2 broadcasts $begin_2^r$ and moves to $(\texttt{wait}[r], 2)$ and n_1 receives it and moves to $(\texttt{wait}[r], 1)$. Notice that at this point, the base of each n_i is $\texttt{wait}[r]$ and the labels of all the other nodes are unchanged, i.e., the same as the labels in γ.

Now, we proceed in the reverse direction. n_2 broadcasts end_2^r and moves to $(\texttt{ack-fwd}[r], 2)$ and n_1 receives it and moves to $(\texttt{ack-rec}[r], 1)$. Then, n_1 broadcasts end_1^r and moves to $(\texttt{ack-fwd}[r], 1)$ and n_0 and n_2 receive it and move to $(\texttt{ack-rec}[r], 0)$ and $(q_2', 2)$ respectively. Then, n_0 broadcasts end_0^r and moves to $(q_0', 0)$ and n_1 receives it and moves to $(q_1', 1)$. It is clear that the configuration reached at the end of this run satisfies the required properties. ◀

We now show a partial converse to the above lemma. It says that if there is a run of good configurations which uses only the transitions given in the gadget for the rule r and begins and ends with the root being in $(q_0, 0)$ and $(q_0', 0)$, then it is possible to "lift" that run back to the corresponding configurations in the NCS $\mathcal{N}$.

▶ **Lemma 9** ($\mathcal{N}$ simulates $\mathcal{P}_{\mathsf{sim}}$)**.** *Suppose $\gamma \xrightarrow{*} \gamma'$ where 1) γ is a good configuration, 2) the labels of the root in γ and γ' are $(q_0, 0)$ and $(q_0', 0)$ and 3) in all the configurations between γ and γ', the base of the root is in the set $\{\texttt{req-fwd}[r], \texttt{wait}[r], \texttt{ack-rec}[r]\}$. Then, 1) γ' is a good configuration and, 2) $\mathbb{E}(\gamma) \overset{*}{\dashrightarrow} \mathbb{E}(\gamma')$.*

Proof sketch. Let the run $\gamma \xrightarrow{*} \gamma'$ be of the form $\gamma = \gamma_0 \rightarrow \gamma_1 \rightarrow \dots \gamma_{m-1} \rightarrow \gamma_m = \gamma'$. By means of induction and some extensive case analysis on the gadget that we have constructed, we can first prove that there exists a path $\mathsf{n}_0, \mathsf{n}_1, \dots, \mathsf{n}_w$ in γ with the following properties:

- For each $0 \leq l \leq w$, the label of n_l is (q_l, l) in γ and (q_l', l) in γ'.
- For each $0 \leq l \leq w$, n_l broadcasts exactly two messages: $begin_l^r$ and end_l^r.
- For each $0 \leq l < w$, the only child of n_l that broadcasts a message in the run is n_{l+1}.

We then let $Ch(\mathsf{n}_l)$ denote the set of children of n_l. Notice that the only node which could broadcast a message in γ_0 is n_0 and so it must be the case that $\gamma_0 \xrightarrow{\mathsf{n}_0, begin_0^r} \gamma_1$. Now, suppose, for some $0 \leq l < w$, we have shown that it must be the case that $\gamma_0 \xrightarrow{\mathsf{n}_0, begin_0^r} \gamma_1 \dots \gamma_l \xrightarrow{\mathsf{n}_l, begin_l^r} \gamma_{l+1}$. Then, notice that the only nodes whose labels in γ_{l+1} could have an outgoing broadcast transition are the nodes in $\bigcup_{0 \leq l' < l} (Ch(\mathsf{n}_{l'}) \setminus \{\mathsf{n}_{l'+1}\}) \cup Ch(\mathsf{n}_l)$. By our claim, among these only n_{l+1} broadcasts a message and so we must have that $\gamma_{l+1} \xrightarrow{\mathsf{n}_{l+1}, begin_{l+1}^r} \gamma_{l+2}$. Hence, in this way we get that $\gamma_0 \xrightarrow{\mathsf{n}_0, begin_0^r} \dots \gamma_w \xrightarrow{\mathsf{n}_w, begin_w^r} \gamma_{w+1}$. In exactly the same way, we can show that it must be the case that $\gamma_{w+1} \xrightarrow{\mathsf{n}_w, end_w^r} \gamma_{w+2} \xrightarrow{\mathsf{n}_{w-1}, end_{w-1}^r} \gamma_{w+3} \dots \gamma_{2w+1} \xrightarrow{\mathsf{n}_0, end_0^r} \gamma_{2w+2} = \gamma_m$.

Let S be the set of all nodes whose base in γ belonged to $Q \cup \{start\}$ and whose base in γ' is *finish*. (Notice that $S \subseteq \bigcup_{0 \leq l < w} Ch(\mathsf{n}_l)$ and $S \cup \{\mathsf{n}_0, \dots, \mathsf{n}_w\}$ are exactly the set of nodes whose labels have changed during the run). It is then easy to see that, by firing the rule r from $\mathbb{E}(\gamma)$ and then deleting all the subtrees whose roots are in S, we get $\mathbb{E}(\gamma) \overset{*}{\dashrightarrow} \mathbb{E}(\gamma')$. ◀

With these two "simulation" lemmas, we have the following main result.

▶ **Theorem 10.** *The state q_{in} can cover the state q_f in the NCS $\mathcal{N}$ iff $(q_f, 0)$ can be covered by some execution in $\mathcal{P}$ starting at a good initial configuration.*

5 A seeker protocol $\mathcal{P}_{\text{seek}}$

In the previous section, we have shown that given a k-NCS $\mathcal{N} = (Q, \delta)$ along with two states $q_{in}, q_f \in Q$, we can construct a simulator protocol $\mathcal{P}_{\text{sim}}$, such that q_{in} can cover q_f in $\mathcal{N}$ iff $(q_f, 0)$ can be covered in $\mathcal{P}_{\text{sim}}$ by an execution starting at a *good initial configuration*. In this section, we will construct a *seeker* protocol $\mathcal{P}_{\text{seek}}$ and "attach" it to $\mathcal{P}_{\text{sim}}$ which will let us get rid of the *goodness* assumption. The seeker protocol $\mathcal{P}_{\text{seek}}$ will begin at an arbitrary initial communication topology and *seek* for a subgraph to act as a good initial configuration for $\mathcal{P}_{\text{sim}}$. Hence, once we have deployed $\mathcal{P}_{\text{seek}}$ to find such a subgraph, we can then use $\mathcal{P}_{\text{sim}}$ to simulate the k-NCS $\mathcal{N}$ on this subgraph.

Formally, the seeker protocol $\mathcal{P}_{\text{seek}} = (Q_{\text{seek}}, I_{\text{seek}}, \Sigma_{\text{seek}}, \delta_{\text{seek}})$ will be a generalization of the protocol given in Figure 1 (with the exception that the (e, i) and $(\perp, i)$ states will be replaced by (start, i) and (finish, i) respectively).

States and alphabet. For each $0 \leq i \leq k$, $\mathcal{P}_{\text{seek}}$ will have six states of the form $(a, i), (b, i), (c, i), (d, i), (\mathit{start}, i)$ and (finish, i). Notice that (start, i) and (finish, i) are also present in $\mathcal{P}_{\text{sim}}$. $\mathcal{P}_{\text{seek}}$ will also have the state $(q_{in}, 0)$, which is a part of $\mathcal{P}_{\text{sim}}$ as well. Similar to $\mathcal{P}_{\text{sim}}$, we can define base and grade of a state.

The initial set of states will be $\{(a, i) : 0 \leq i \leq k\}$. For each $0 \leq i \leq k$, Σ_{seek} will have two letters: ht_i and $\overline{ht}_i$. Σ_{seek} will also have another additional letter: *transfer*.

Transitions. Before we define the set of transitions, we make the same convention for $\mathcal{P}_{\text{seek}}$ that we had made in Remark 7 for $\mathcal{P}_{\text{sim}}$. Having stated this, we now describe the transitions:

- For the case of $i = 0$, we have the following transitions:

$$\begin{tikzcd}
(a,0) \arrow[r, "!ht_0"] & (c,0) \arrow[loop above, "?ht_1{,}\, ?\overline{ht}_1"] \arrow[r, "!\overline{ht}_0"] & (\mathit{start},0) \arrow[r, "!\text{transfer}"] & (q_{in},0)
\end{tikzcd}$$

- For the case of $1 \leq i \leq k$, we have the following transitions: (The self-loops over the state (c, i) are not included when $i = k$).

$$\begin{tikzcd}
(a,i) \arrow[r, "?ht_{i-1}"] & (b,i) \arrow[r, "!ht_i"] & (c,i) \arrow[loop above, "?ht_{i+1}{,}\, ?\overline{ht}_{i+1}"] \arrow[r, "!\overline{ht}_i"] & (d,i) \arrow[r, "?\overline{ht}_{i-1}"] & (\mathit{start},i) \arrow[loop above, "?\text{transfer}"]
\end{tikzcd}$$

Intuition behind the transitions. Let us give a brief intuition behind the transitions in the case of $k = 2$. A node n_0 which is at $(a, 0)$ aims to become the root of the good initial configuration that the seeker protocol should find, and so broadcasts ht_0, letting its neighbors know that it wants to be the root of the good subgraph. If any neighbor of n_0 is not in $(a, 1)$ then it immediately moves to a state with base *finish*. Otherwise, the set of all neighbors in $(a, 1)$ move to $(b, 1)$. From here, all of these nodes can broadcast ht_1, letting their neighbors know that they now want to become a child of the root. All these messages will also be received n_0 which will use the self-loop at $(c, 0)$ to ignore these messages. All the nodes which receive a ht_1 message can either move to a state with base *finish* or move to $(b, 2)$, from where they can broadcast ht_2 and thereby move to $(c, 2)$. At this point, we must have a tree subgraph in which n_0 is labelled by $(c, 0)$, its children are labelled by $(c, 1)$ and its children are labelled by $(c, 2)$.

Now the nodes labelled by $(c, 2)$ can all broadcast $\overline{ht}_2$, then the nodes labelled by $(c, 1)$ can all broadcast $\overline{ht}_1$ and then the node n_0 can broadcast $\overline{ht}_0$. This leads to a tree subgraph where n_0 is labelled by $(\mathit{start}, 0)$, its children are labelled by $(\mathit{start}, 1)$ and its children are labelled by $(\mathit{start}, 2)$. Now, n_0 can broadcast the letter "transfer" and move into $(q_{in}, 0)$, thereby *transferring* the control over to the simulator protocol $\mathcal{P}_{\text{sim}}$. In this manner, $\mathcal{P}_{\text{seek}}$ has found a good initial subgraph in which to run $\mathcal{P}_{\text{sim}}$.

Proof of correctness. Let $\mathcal{P} = (Q_{\text{seek}} \cup Q_{\text{sim}}, I_{\text{seek}}, \Sigma_{\text{seek}} \cup \Sigma_{\text{sim}}, \delta_{\text{seek}} \cup \delta_{\text{sim}})$ be the protocol obtained by taking the union of the seeker and the simulator protocols, such that the initial set of states is the initial set of states of the seeker protocol. Similar to the intuition given above, the protocol $\mathcal{P}$ first runs the seeker protocol till a node with label $(q_{in}, 0)$ is reached, at which point it runs the simulator protocol. The following lemma tells us that if a node gets labelled by $(q_{in}, 0)$ while running $\mathcal{P}$, then with that node as the root, there is a good initial configuration for the simulator protocol $\mathcal{P}_{\text{sim}}$. This then allows us the protocol $\mathcal{P}$ to run the simulator protocol on this good initial configuration.

▶ **Lemma 11.** *Suppose $\gamma \xrightarrow{*} \gamma' \xrightarrow{\mathsf{n}, transfer} \eta$ is an execution of $\mathcal{P}$. After removing all nodes whose label's base is finish in η, the connected component containing the node n is a good initial configuration for the simulator protocol $\mathcal{P}_{sim}$.*

Proof sketch. First, let us focus on the execution $\gamma \xrightarrow{*} \gamma'$. By definition of an execution, γ is an initial configuration for the protocol $\mathcal{P}_{\text{seek}}$ and so all the nodes in γ have their labels in the set $\{a_i : 0 \leq i \leq k\}$.

Let T be the connected component containing the node n in γ' after removing all nodes whose base is *finish*. Let $F := \{(\mathit{start}, i) : 0 \leq i \leq k\}$. First, we show that all nodes in T must have labels from the set F. Suppose there is a node n' in T whose label is not in F. Pick such an n' which is at the shortest distance from n and let $\mathsf{n} = \mathsf{n}_0, \mathsf{n}_1, \mathsf{n}_2, \ldots, \mathsf{n}_l, \mathsf{n}'$ be a shortest path from n to n'.

By a generalization of the argument given in Example 2, we can prove by induction that for each $1 \leq i \leq l$, the label of each n_i in T is (start, i) and the only neighbor of n_i which was labelled by $(c, i-1)$ at some point during the run is n_{i-1}. Using this, we can then show that n' must have moved to $(\mathit{start}, l+1)$ at some point during the run.

By assumption, the label of n' is not $(\mathit{start}, l+1)$ in T, and so it must moved out of $(\mathit{start}, l+1)$ to some state of the simulator protocol. By analysing the constructed protocol $\mathcal{P}$, we can then prove that n' must have received two ht_l messages. But any node that receives two ht_l messages must necessarily move to a state with base *finish*, contradicting the fact that $\mathsf{n}' \in T$.

Having proved that every node in T has its label in F, we can then show by examining the structure of the transitions, that T must be a tree of height atmost k such that n_0 is labelled by $(\mathit{start}, 0)$ and all nodes at distance i from n_0 are labelled by (start, i). This then implies that after removing all nodes with base *finish* in η, the connected component containing the node n is a good initial configuration for the simulator protocol. ◀

▶ **Theorem 12.** *The state q_{in} can cover q_f in the NCS $\mathcal{N}$ iff the state $(q_f, 0)$ can be covered from any initial configuration in $\mathcal{T}_{2k}(\mathcal{P})$.*

Proof sketch. Due to lack of space, we focus only on the right to left implication. Suppose $\gamma \xrightarrow{*} \gamma'$ is an execution of $\mathcal{P}$ such that some node n in γ' is labelled by $(q_f, 0)$. Let γ_0 be the configuration along this run when the node n first got the label $(q_{in}, 0)$. (Notice that such a configuration must exist because of the construction of $\mathcal{P}$). By Lemma 11, in γ_0, if we remove all nodes whose base is *finish*, then we get a good initial configuration T for $\mathcal{P}_{\text{sim}}$ with n as the root. Notice that no node with base *finish* can ever broadcast a message. Hence, in the run $\gamma_0 \xrightarrow{*} \gamma'$, none of the nodes in T ever receive a message from any node outside of T. It follows that we can restrict the run $\gamma_0 \xrightarrow{*} \gamma'$ to only the subtree T, to get a run of $\mathcal{P}_{\text{sim}}$ starting at a good initial configuration and covering $(q_f, 0)$. By Theorem 10, we get that q_f can be covered from q_{in} in $\mathcal{N}$. ◀

Hence, we get,

▶ **Corollary 13.** *Bounded-Path-Cover is $\mathbf{F}_{\epsilon_0}$-hard.*

6 Upper bound for Bounded-Path-Cover

In this section, we give a sketch of the proof that Bounded-Path-Cover is in $\mathbf{F}_{\epsilon_0}$. Let $\mathcal{P} = (Q, I, \Sigma, \Delta)$ be a fixed protocol.

▶ **Definition 14.** *Let $\gamma_1 = (\mathsf{N}_1, \mathsf{E}_1, \mathsf{L}_1)$ and $\gamma_2 = (\mathsf{N}_2, \mathsf{E}_2, \mathsf{L}_2)$ be two configurations of $\mathcal{P}$. We say that γ_1 is an induced subgraph of γ_2 (denoted by $\gamma_1 \preceq_{is} \gamma_2$) if there is a label preserving injection h from N_1 to N_2 such that $(\mathsf{n}, \mathsf{n}') \in \mathsf{E}_1$ if and only if $(h(\mathsf{n}), h(\mathsf{n}')) \in \mathsf{E}_2$.*

It is known that, for any $k \geq 1$, the set of all k-path bounded configurations of $\mathcal{P}$ is a well-quasi ordering under the induced subgraph relation $\preceq_{\text{is}}$ (Theorem 2.2 of [10]). Using this fact, the authors of [9] show that for every k, the transition system $\mathcal{T}_k(\mathcal{P})$ is a *well-structured transition system* (WSTS) and then apply the generic backward exploration algorithm for WSTS (See [20, 13]) to prove that Bounded-Path-Cover is decidable. By using the standard and generic complexity arguments for WSTS (See [20, 13, 21]), an upper bound on the running time of their procedure simply boils down to estimating the length of *controlled bad sequences* of k-path bounded configurations under the induced subgraph relation.

Let $H : \mathbb{N} \to \mathbb{N}$ be the successor function and let $n \in \mathbb{N}$. For each $i \in \mathbb{N}$, let H^i denote the i-fold application of H to itself i times, with H^0 being the identity function.

▶ **Definition 15.** *A sequence $\gamma_0, \gamma_1, \ldots,$ of k-path bounded configurations is (H, n)-controlled bad if the number of nodes in each γ_i is at most $H^i(n)$ and $\gamma_i \not\preceq_{is} \gamma_j$ for any $i < j$.*

Our main result is an upper bound on the length of (H, n)-controlled bad sequences of k-path bounded configurations, by embedding these configurations into the well-quasi ordering of *generalized priority alphabets* (See [16]). This encoding is inspired by a similar encoding given for bounded depth trees in Section 8.1 of [16]. This result then allows us to prove that

▶ **Theorem 16.** *Bounded-Path-Cover is in $\mathbf{F}_{\epsilon_0}$ and hence $\mathbf{F}_{\epsilon_0}$-complete.*

References

1 Sergio Abriola, Santiago Figueira, and Gabriel Senno. Linearizing well quasi-orders and bounding the length of bad sequences. *Theor. Comput. Sci.*, 603:3–22, 2015. doi:10.1016/j.tcs.2015.07.012.

2 Nathalie Bertrand, Patricia Bouyer, and Anirban Majumdar. Reconfiguration and message losses in parameterized broadcast networks. *Log. Methods Comput. Sci.*, 17(1), 2021. URL: https://lmcs.episciences.org/7280.

3 Michael Blondin. The abcs of petri net reachability relaxations. *ACM SIGLOG News*, 7(3):29–43, 2020. doi:10.1145/3436980.3436984.

4 Michael Blondin, Alain Finkel, Christoph Haase, and Serge Haddad. The logical view on continuous petri nets. *ACM Trans. Comput. Log.*, 18(3):24:1–24:28, 2017. doi:10.1145/3105908.

5 Michael Blondin and Christoph Haase. Logics for continuous reachability in petri nets and vector addition systems with states. In *32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, June 20-23, 2017*, pages 1–12. IEEE Computer Society, 2017. doi:10.1109/LICS.2017.8005068.

6 Pierre Chambart and Philippe Schnoebelen. The ordinal recursive complexity of lossy channel systems. In *Proceedings of the Twenty-Third Annual IEEE Symposium on Logic in Computer Science, LICS 2008, 24-27 June 2008, Pittsburgh, PA, USA*, pages 205–216, 2008. doi:10.1109/LICS.2008.47.

7 Wojciech Czerwinski, Slawomir Lasota, Ranko Lazic, Jérôme Leroux, and Filip Mazowiecki. The reachability problem for petri nets is not elementary. In Moses Charikar and Edith Cohen, editors, *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 24–33. ACM, 2019. doi:10.1145/3313276.3316369.

8 Normann Decker and Daniel Thoma. On freeze LTL with ordered attributes. In Bart Jacobs and Christof Löding, editors, *Foundations of Software Science and Computation Structures - 19th International Conference, FOSSACS 2016, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2016, Eindhoven, The Netherlands, April 2-8, 2016, Proceedings*, volume 9634 of *Lecture Notes in Computer Science*, pages 269–284. Springer, 2016. doi:10.1007/978-3-662-49630-5_16.

9 Giorgio Delzanno, Arnaud Sangnier, and Gianluigi Zavattaro. Parameterized verification of ad hoc networks. In *CONCUR 2010 - Concurrency Theory, 21th International Conference,*, pages 313–327, 2010. doi:10.1007/978-3-642-15375-4_22.

10 Guoli Ding. Subgraphs and well-quasi-ordering. *Journal of Graph Theory*, 16(5):489–502, 1992. doi:10.1002/jgt.3190160509.

11 Jacob Elgaard, Nils Klarlund, and Anders Møller. Mona 1.x: New techniques for ws1s and ws2s. In Alan J. Hu and Moshe Y. Vardi, editors, *Computer Aided Verification*, pages 516–520, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

12 Javier Esparza, Ruslán Ledesma-Garza, Rupak Majumdar, Philipp J. Meyer, and Filip Niksic. An smt-based approach to coverability analysis. In Armin Biere and Roderick Bloem, editors, *Computer Aided Verification - 26th International Conference, CAV 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 18-22, 2014. Proceedings*, volume 8559 of *Lecture Notes in Computer Science*, pages 603–619. Springer, 2014. doi:10.1007/978-3-319-08867-9_40.

13 Diego Figueira, Santiago Figueira, Sylvain Schmitz, and Philippe Schnoebelen. Ackermannian and primitive-recursive bounds with dickson's lemma. In *Proceedings of the 26th Annual IEEE Symposium on Logic in Computer Science*, pages 269–278, 2011. doi:10.1109/LICS.2011.39.

14 Estíbaliz Fraca and Serge Haddad. Complexity analysis of continuous petri nets. *Fundam. Informaticae*, 137(1):1–28, 2015. doi:10.3233/FI-2015-1168.

15 Christoph Haase and Simon Halfon. Integer vector addition systems with states. In Joël Ouaknine, Igor Potapov, and James Worrell, editors, *Reachability Problems - 8th International Workshop, RP 2014, Oxford, UK, September 22-24, 2014. Proceedings*, volume 8762 of *Lecture Notes in Computer Science*, pages 112–124. Springer, 2014. doi:10.1007/978-3-319-11439-2_9.

16 Christoph Haase, Sylvain Schmitz, and Philippe Schnoebelen. The power of priority channel systems. *Log. Methods Comput. Sci.*, 10(4), 2014. doi:10.2168/LMCS-10(4:4)2014.

17 Albert R. Meyer. Weak monadic second order theory of succesor is not elementary-recursive. In Rohit Parikh, editor, *Logic Colloquium*, pages 132–154, Berlin, Heidelberg, 1975. Springer Berlin Heidelberg.

18 Sylvain Schmitz. Complexity bounds for ordinal-based termination - (invited talk). In *Reachability Problems - 8th International Workshop, RP 2014*, pages 1–19, 2014. `doi:10.1007/978-3-319-11439-2_1`.

19 Sylvain Schmitz. Complexity hierarchies beyond elementary. *ACM Trans. Comput. Theory*, 8(1):3:1–3:36, 2016. `doi:10.1145/2858784`.

20 Sylvain Schmitz and Philippe Schnoebelen. Multiply-recursive upper bounds with higman's lemma. In *Automata, Languages and Programming - 38th International Colloquium, ICALP 2011*, pages 441–452, 2011. `doi:10.1007/978-3-642-22012-8_35`.

21 Sylvain Schmitz and Philippe Schnoebelen. The power of well-structured systems. In Pedro R. D'Argenio and Hernán C. Melgratti, editors, *CONCUR 2013 - Concurrency Theory - 24th International Conference, CONCUR 2013, Buenos Aires, Argentina, August 27-30, 2013. Proceedings*, volume 8052 of *Lecture Notes in Computer Science*, pages 5–24. Springer, 2013. `doi:10.1007/978-3-642-40184-8_2`.

A Appendix

A.1 Proofs for Section 6

First, let us describe the backward exploration algorithm for solving Bounded-Path-Cover that is given in Section 5 of [9]. Given a protocol $\mathcal{P} = (Q, I, \Sigma, \delta)$, a state f and a number k, we consider the set of all configurations in $\mathcal{T}_k(\mathcal{P})$ with the induced subgraph ordering $\preceq_{\text{is}}$. Given a set S of $\mathcal{T}_k(\mathcal{P})$ we let $\uparrow S := \{\gamma' : \exists \gamma \in S, \gamma \preceq_{\text{is}} \gamma'\}$. A set S is called *upward-closed* if $S = \uparrow S$.

In Section 5 of [9], the following results are proved about $\mathcal{T}_k(\mathcal{P})$:

- If S is upward-closed, then there exists a finite set B such that $\uparrow B = S$. Such a B will be called the basis of S.
- If S is upward-closed and if $Pre(S)$ is the set of all configurations $\gamma' \in \mathcal{T}_k(\mathcal{P})$ such that there is a configuration $\gamma \in S$ with $\gamma' \to \gamma$, then $S \cup Pre(S)$ is upward-closed. Moreover, given a basis B of S, we can compute a basis B' of $S \cup Pre(S)$ such that the number of nodes of each configuration in B' is at most one more than the maximum number of nodes in any configuration of B.

In Theorem 5 of [9] it is shown that the following algorithm terminates and is correct for Bounded-Path-Cover : Construct a sequence of finite sets $B_0, B_1, \ldots,$ such that each $B_i \subseteq \mathcal{T}_k(\mathcal{P})$, B_0 is the single node configuration labelled by f and B_{i+1} is a basis for $\uparrow B_i \cup Pre(\uparrow B_i)$. The algorithm then finds the first m such that $\uparrow B_m = \uparrow B_{m+1}$ and checks if there is an initial configuration in $\uparrow B_m$.

The running time complexity of the algorithm is mainly dominated by the length of the sequence $B_0, B_1, \ldots, B_m$. Since m is the first index such that $\uparrow B_m = \uparrow B_{m+1}$, we can find a minimal element $\gamma_i \in \uparrow B_{i+1} \setminus \uparrow B_i$ for each $i < m$.

Consider the sequence $\gamma_0, \ldots, \gamma_{m-1}$. Notice that $\gamma_i \not\preceq_{\text{is}} \gamma_j$ for any $j > i$ and further the number of nodes in each γ_i is at most $H^i(1)$, where H is the successor function. It follows that $\gamma_0, \ldots, \gamma_{m-1}$ is a controlled bad sequence. Our main result is that

▶ **Lemma 17.** *The length of (H, n)-controlled bad sequences over k-path bounded configurations of $\mathcal{P}$ is upper bounded by the function $F_{\epsilon_0}(p(|Q|, k, n))$.*

Here F_{ϵ_0} is the *fast-growing* function at level ϵ_0 and p is some fixed primitive recursive function. For our purposes, we do not need the actual definition of F_{ϵ_0}, but we only need to know that $\mathbf{F}_{\epsilon_0}$ contains the set of problems whose running time is upper bounded by the function F_{ϵ_0} composed with any primitive recursive function (See [19]). By the lemma above and the fact that the running time complexity of the algorithm for Bounded-Path-Cover is primarily dominated by the length of $(H, 1)$-controlled bad sequences we get,

▶ **Theorem 18.** *BOUNDED-PATH-COVER is in* $\mathbf{F}_{\epsilon_0}$.

All that suffices is to prove Lemma 17. To do so, we will reduce the problem of estimating the length of controlled bad sequences over k-path bounded configurations to the problem of estimating the length of controlled bad sequences over another well-quasi order for which we already know upper bounds. We now proceed to recall this well-quasi order as it is defined in [16].

Generalized priority alphabets

Given a number $k \in \mathbb{N}$ called the priority level and a finite set Γ, a *generalised priority alphabet* is the set $\Sigma_{\Gamma,k} := \{(a,i) : a \in \Gamma, 0 \leq i \leq k\}$. Given $m = (a,i) \in \Sigma_{\Gamma,k}$, we say that i is the priority of m. Then for $x, y \in \Sigma^*_{\Gamma,k}$, we say that $x \sqsubseteq_{\Gamma,k} y$ if $x = (a_1,i_1),(a_2,i_2),\dots,(a_l,i_l)$ where each $(a_j,i_j) \in \Sigma_{\Gamma,k}$ and $y = y_1(a_1,i_1)y_2(a_2,i_2)y_3\dots y_l(a_l,i_l)$ such that $\forall 1 \leq j \leq l$, we have $y_j \in \Sigma^*_{\Gamma,i_j}$, i.e., x can be obtained from y by removing subwords in such a manner so that the priority of each removed subword is not bigger than the first preserved letter to its right. It is known that for every k and Γ, the ordering $\sqsubseteq_{\Gamma,k}$ is a well-quasi ordering. (Theorem 3.6 of [16]). Now, similar to controlled bad sequences for k-path bounded configurations, we can define (a slightly different notion of) controlled bad sequences for words over $\Sigma_{\Gamma,k}$. Let $Sq : \mathbb{N} \to \mathbb{N}$ be the squaring function and let Sq^i denote the squaring function composed with itself i times.

▶ **Definition 19.** *A sequence* $w_0, w_1, \dots,$ *of words over* $\Sigma_{\Gamma,k}$ *is* (Sq,n)*-controlled bad if the length of each* w_i *is at most* $Sq^i(n)$ *and* $w_i \not\sqsubseteq_{\Gamma,k} w_j$ *for any* $i < j$.

Encoding k-path bounded graphs using generalized priority alphabets

A labelled k-path bounded graph is any graph $G = (\mathsf{N},\mathsf{E},\mathsf{L})$ such that there is a labelling function $\mathsf{L} : \mathsf{N} \to A$ for some some finite set A. (Notice that the set of k-path bounded configurations of a protocol is a labelled k-path bounded graph where A is the set of states of the protocol). We have the following theorem regarding labelled k-path bounded graphs.

▶ **Theorem 20** (Lemma 2.1 of [10]). *Suppose* G *is a labelled* k*-path bounded graph for* $k \geq 1$. *Then there is a node* n *such that every connected component of* $G \setminus \{\mathsf{n}\}$ *is a labelled* $(k-1)$*-path bounded graph.*

This theorem suggests the following inductive encoding of labelled k-path bounded graphs as strings over a priority alphabet: Let $G = (\mathsf{N},\mathsf{E},\mathsf{L})$ be any labelled graph with labelling function $\mathsf{L} : \mathsf{N} \to A$ where A is some finite set. Let $\mathsf{e}, \bar{\mathsf{e}}$ be two symbols not in the finite set A and let $A^k := \cup_{0 \leq i \leq k} A \times \{\mathsf{e},\bar{\mathsf{e}}\}^i$. Notice that $A^0 := A$. By induction on k, we will now define a string $\langle G \rangle \in \Sigma_{A^k,k}$.

Base case: If G is a 0-path bounded configuration, then G is a single node n and can be encoded as $(\mathsf{L}(\mathsf{n}),0) \in \Sigma^*_{A^0,0}$.

Induction step: Suppose G is a k-path bounded configuration for some $k \geq 1$ such that G is not $(k-1)$-path bounded. Let n be a vertex such that all the connected components $C_1,\dots,C_l$ of $G \setminus \{\mathsf{n}\}$ are $(k-1)$-path bounded configurations. (Such a node exists by Theorem 20). For every node n' in every C_i, first change its label from $\mathsf{L}(\mathsf{n}')$ to $(\mathsf{L}(\mathsf{n}'),\mathsf{e})$ if n' is a neighbor of n in G and otherwise change its label to $(\mathsf{L}(\mathsf{n}'),\bar{\mathsf{e}})$. Call these new labelled graphs as $C_1^{\mathsf{n}},\dots,C_l^{\mathsf{n}}$.

By induction hypothesis, for each C_i^{n}, we have a string $\langle C_i^{\mathsf{n}} \rangle \in \Sigma^*_{((A\times\{\mathsf{e},\bar{\mathsf{e}}\})^{k-1},k-1)} \subseteq \Sigma^*_{A^k,k-1}$. We now let $\langle G \rangle := \langle C_1^{\mathsf{n}} \rangle (\mathsf{L}(\mathsf{n}),k) \langle C_2^{\mathsf{n}} \rangle (\mathsf{L}(\mathsf{n}),k) \dots \langle C_l^{\mathsf{n}} \rangle (\mathsf{L}(\mathsf{n}),k)$.

Notice that if G is a labelled k-path bounded graph which is not $(k-1)$-path bounded, then $\langle G \rangle$ is of the form $\langle C_1^{\mathsf{n}} \rangle(a,k)\langle C_2^{\mathsf{n}} \rangle(a,k) \dots \langle C_l^{\mathsf{n}} \rangle(a,k)$ where 1) a is the label of some node n in G, 2) $C_1, \dots, C_l$ are connected components of $G \setminus \{\mathsf{n}\}$ which are labelled $(k-1)$-path bounded subgraphs of G. This will be called the *decomposition* of $\langle G \rangle$ and the node n will be called its *crown*.

We then have the following lemma:

▶ **Lemma 21.** *If G and H are such that $\langle G \rangle \sqsubseteq_{A^k,k} \langle H \rangle$ then $G \preceq_{is} H$.*

Proof. Notice that if $\langle G \rangle \sqsubseteq_{A^k,k} \langle H \rangle$, then the highest priority appearing in $\langle G \rangle$ and $\langle H \rangle$ must be the same, which, without loss of generality, we can assume to be k.

We prove the lemma by induction on k. The base case of 0 is clear.

For the induction step, let $\langle C_1^{\mathsf{n}} \rangle(a,k)\langle C_2^{\mathsf{n}} \rangle(a,k) \dots \langle C_m^{\mathsf{n}} \rangle(a,k)$ be the decomposition of $\langle G \rangle$ with crown n and let $\langle D_1^{\mathsf{n}'} \rangle(a',k)\langle D_2^{\mathsf{n}'} \rangle(a',k) \dots \langle D_n^{\mathsf{n}'} \rangle(a',k)$ be the decomposition of $\langle H \rangle$ with crown n'. Since $\langle G \rangle \sqsubseteq_{A^k,k} \langle H \rangle$, it must be the case that $a = a'$.

By definition of the $\sqsubseteq_{A^k,k}$ relation, it must be the case that for every C_j^{n}, there exists i_j such that $\langle C_j^{\mathsf{n}} \rangle \sqsubseteq_{A^k,k-1} \langle D_{i_j}^{\mathsf{n}'} \rangle$. Notice that the priority has reduced and we can apply the induction hypothesis to conclude that for each j, $C_j^{\mathsf{n}} \preceq_{\text{is}} D_{i_j}^{\mathsf{n}'}$ and so there exists a label preserving injection h_j from the nodes of C_j^{n} to the nodes of $D_{i_j}^{\mathsf{n}'}$ such that (u,v) is an edge in C_j^{n} iff $(h_j(u), h_j(v))$ is an edge in $D_{i_j}^{\mathsf{n}'}$.

Now, consider the following label preserving injection h from G to H: Map the crown n to the other crown n' and if n'' is any other node in any one of the connected components C_j, then map n'' to $h_j(\mathsf{n}'')$. Notice that if u and v are nodes in G which belong to the same connected component of $G \setminus \{\mathsf{n}\}$ then (u,v) is an edge in G iff $(h(u), h(v))$ is an edge in H. Similarly, if u and v are nodes in G which belong to different connected components of $G \setminus \{\mathsf{n}\}$ then $h(u)$ and $h(v)$ also belong to different connected components of $H \setminus \{\mathsf{n}'\}$ and so the statement "(u,v) is an edge in G iff $(h(u), h(v))$ is an edge in H" is vacously true.

Finally suppose $u = \mathsf{n}$ and v is some other node of G. Notice that the last field in the label of v is e if (u,v) is an edge in G and $\bar{\mathsf{e}}$ otherwise. By definition of h we have that $h(u) = \mathsf{n}'$ and also that the label of $h(v)$ is the same as v. But by definition of decomposition of $\langle H \rangle$, the last field in the label of $h(v)$ is e if $(\mathsf{n}', h(v))$ is an edge in H and $\bar{\mathsf{e}}$ otherwise. Hence, in this case as well, we have shown that (u,v) is an edge in G iff $(h(u), h(v))$ is an edge in H. This concludes the proof. ◀

Upper bound on the length of controlled bad sequences for k-path bounded configurations

Fix a protocol $\mathcal{P}$ with states Q and a number k and consider the set of configurations in $\mathcal{T}_k(\mathcal{P})$. By the previous lemma, we can infer that the length of the longest (H,n)-controlled bad sequence over the set of configurations of $\mathcal{T}_k(\mathcal{P})$ is at most the length of the longest (Sq,n)-controlled bad sequence over the generalized priority alphabet $\Sigma_{Q^k,k}$, which we know is at most $F_{\epsilon_0}(p(|Q|,k,n))$ where p is some primitive recursive function (Proposition 4.1 and Sections 4.1.1 and 4.1.2 of [16]). This then implies Lemma 17, which is what we wanted to prove.

Appendix C

Complexity of Coverability in Depth-Bounded Processes (CONCUR 2022)

This section contains a reprinting of the following paper, which has been published as a peer-reviewed conference paper.

> A. R. Balasubramanian. *Complexity of Coverability in Depth-Bounded Processes.* In conference proceedings of CONCUR 2022. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. Vol. 243 of LIPIcs, Pages - 17:1-17:19. doi: `10.4230/LIPIcs.CONCUR.2022.17`

Summary

We consider the class of depth-bounded processes in π-calculus. These processes are the most expressive fragment of π-calculus, for which verification problems are known to be decidable. The decidability of the coverability problem for this class has been achieved by means of well-quasi-orders. In this work, we characterize the complexity of coverability for depth-bounded processes and

prove that it is $\mathbf{F}_{\epsilon_0}$-complete, where $\mathbf{F}_{\epsilon_0}$ is a class in the fast-growing hierarchy of complexity classes.

Contributions of the author of this thesis

I am the sole author of this paper.

Complexity of Coverability in Depth-Bounded Processes

A. R. Balasubramanian
Technische Universität München, Germany

Abstract

We consider the class of depth-bounded processes in π-calculus. These processes are the most expressive fragment of π-calculus, for which verification problems are known to be decidable. The decidability of the coverability problem for this class has been achieved by means of well-quasi orders. (Meyer, IFIP TCS 2008; Wies, Zufferey and Henzinger, FoSSaCS 2010). However, the precise complexity of this problem has not been known so far, with only a known EXPSPACE-lower bound.

In this paper, we prove that coverability for depth-bounded processes is $\mathbf{F}_{\epsilon_0}$-complete, where $\mathbf{F}_{\epsilon_0}$ is a class in the fast-growing hierarchy of complexity classes. This solves an open problem mentioned by Haase, Schmitz, and Schnoebelen (LMCS, Vol 10, Issue 4) and also addresses a question raised by Wies, Zufferey and Henzinger (FoSSaCS 2010).

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness; Theory of computation → Distributed computing models

Keywords and phrases π-calculus, Depth-bounded processes, Fast-growing complexity classes

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2022.17

Funding *A. R. Balasubramanian*: Supported by funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme under grant agreement No 787367 (PaVeS).

Acknowledgements I am grateful to the reviewers and Prof. Javier Esparza for their useful comments and suggestions.

1 Introduction

The π-calculus [21, 22] is a well-known formalism for describing concurrent message-passing systems admitting unbounded process creation and mobility of agents. Intuitively speaking, a configuration of such a system is a graph in which each vertex is a process labelled by its current state and there is an edge between two processes if they share a channel using which they can pass messages. The flexibility of π-calculus lies in the fact that processes can transmit the names of channels using channels themselves, allowing reconfiguration of channels using process definitions itself. Due to its immense expressive power, all interesting verification problems quickly become undecidable for π-calculus processes.

Consequently, research on π-calculus has been focused on finding fragments for which certain problems are decidable. The most expressive fragment of π-calculus for which some verification problems still remain decidable is the class of depth-bounded processes [20]. Intuitively, depth-bounded processes are those in which the length of simple paths in the set of reachable configurations is bounded by a constant. It is known that depth-bounded processes can be viewed as well-structured transition systems (WSTS) [20]. This implies that the coverability problem for such systems is decidable [20, 27]. Intuitively, coverability consists of deciding if a given system can reach a configuration where some process is in an error state.

However, despite the positive decidability results known regarding this problem, the exact complexity of this problem has remained open so far. To the best of our knowledge, only an EXPSPACE-hardness result is known for this problem [27]. In this paper, we provide complexity-theoretic completeness results for this problem. More specifically, we prove that the coverability problem for depth-bounded processes is $\mathbf{F}_{\epsilon_0}$-complete, where $\mathbf{F}_{\epsilon_0}$ is a complexity class in the *fast-growing hierarchy* of complexity classes [24]. This is a hierarchy of complexity classes which allows for a finer classification of problems that do not admit any elementary-time algorithms, i.e., problems which do not have algorithms whose running times can be upper bounded by a fixed tower of exponentials in the input size. In particular, our result proves that the coverability problem for depth-bounded processes is not primitive-recursive and indeed is harder than even problems complete for the Ackermann complexity class.

The complexity-theoretic classification of problems which are non-elementary has attracted a lot of attention in the recent years, with various techniques developed for proving both lower and upper bounds [13, 6, 25, 24, 1, 23, 8, 19, 7, 18]. While these results are obviously negative from a tractability perspective, understanding the precise complexity of a problem may help us to solve it in practice by reducing it to other well-studied problems for which tools and heuristics have been developed, like the satisfiability problem for weak S1S or the Petri net reachability problem [3, 12, 15, 4, 5, 16, 10]. The fast-growing hierarchy is of great assistance in this task. Adding new complete problems for classes in this hierarchy can help us prove hardness results for other problems in the future, without having to resort to coming up with reductions from scratch, i.e., from Turing machines or counter machines.

Our result significantly improves upon the existing lower bound of EXPSPACE-hardness, which is inherited from the coverability problem for Petri nets. Further, it settles a conjecture raised by Hasse, Schmitz and Schnoebelen (Section 8.3 of [17]) and also addresses a question raised by Wies, Zufferey and Henzinger (Section 5 of [27]).[1] To prove the lower bound, we introduce a new model of computation called *nested counter systems with levels*, which (in a manner) simplifies the already existing model of *nested counter systems* [8], while preserving the hardness of that model.

The techniques used in this paper are similar to the ones presented in [2], in order to prove $\mathbf{F}_{\epsilon_0}$-completeness for parameterized coverability of *bounded-depth broadcast networks*. While some of the ideas between these two papers are similar, there are some differences between the models considered in these two papers. First, as the name suggests, broadcast networks allow for a process to *broadcast* to its set of neighbors, whereas processes in π-calculus interact in a manner akin to *rendez-vous* communication. One might expect that there is a drop in complexity when the communication mechanism goes from broadcast to rendez-vous. For instance, as mentioned in [11], coverability for networks with (unrestricted) broadcast communication is Ackermann-complete, while the same problem for rendez-vous networks is (only) EXPSPACE-complete. Our result suggests that this drop in complexity need not always be the case. Further, in broadcast networks, there is no process creation nor dynamic reconfiguration of channels, whereas π-calculus has both. Finally, for the lower bound construction in this paper, we also need to prove depth-boundedness of any reachable configuration in the process constructed for the reduction, whereas no such property needs to be proven for the lower bound construction for broadcast networks. We also believe that the newly introduced model of nested counter systems with levels (whose hardness we prove by using ideas from [2]), makes the proof of the lower bound for π-calculus cleaner when compared with giving a direct reduction from nested counter systems as was done in [2].

[1] The version of the problem that the authors of [27] consider does not assume that a bound on the depth of the process is given as part of the input, whereas in our setting we take this to be the case, in order to prove the upper bound. However, our lower bound result does not require this assumption.

2 Preliminaries

We first present the syntax and the semantics of the version of π-calculus that we will use . The definitions here are taken from the ones given in [27].

2.1 The π-calculus

We assume that there is a countable collection of *names* (denoted by $x, y, \dots$) and a countable collection of *process identifiers* (denoted by $A, B, \dots$). Each name and identifier has an associated *arity* in $\mathbb{N}$. We use boldface letters like $\mathbf{x}, \mathbf{y}$ to denote (possibly empty) vectors over names and denote substitution of names by $[\mathbf{x}/\mathbf{y}]$, i.e., if $\mathbf{x} = x_1, \dots, x_n$ and $\mathbf{y} = y_1, \dots, y_n$, then $[\mathbf{x}/\mathbf{y}]$ denotes a mapping in which each y_i is mapped to x_i and every other name is mapped to itself.

A *process term* (or simply a term) P is either the unit process 0, or a parameterized process identifier $A(\mathbf{x})$, or any term obtained by the standard operations of parallel composition $P_1 \mid P_2$, external choice $\pi_1 \cdot P_1 + \pi_2 \cdot P_2$ and name restriction $(\nu x)P_1$. Here P_1 and P_2 are themselves terms and π_1 and π_2 are prefixes which can either be an *input prefix* $x(\mathbf{y})$ or an *output prefix* $\bar{x}(\mathbf{y})$ or the empty string. All parameter vectors occuring in a parameterized process identifier or a prefix must respect the arity of the names and identifiers. A *thread* is a term of the form $A(\mathbf{x})$. We use Π and Σ to denote (indexed) parallel composition and external choice. We further use $(\nu \mathbf{x})$ to denote $(\nu x_1)(\nu x_2) \dots (\nu x_n)$ where $\mathbf{x} = x_1, \dots, x_n$. The application of a substitution of names σ to a term P, denoted by $\sigma(P)$, is defined in the usual way.

An occurrence of a name x in a term P is called *free* if it is not below a (νx) or an input prefix $y(x)$. We let $\mathtt{fn}(P)$ denote the set of free names of P. A *bound name* of P is a name of P which is not free. We say that P is *closed* if $\mathtt{fn}(P) = \emptyset$. We use the usual *structural congruence relation* $P \equiv Q$ on process terms, i.e., $P \equiv Q$ if P is syntactically equal to Q upto renaming and reordering of bound names, associativity and commutativity of parallel composition and external choice, elimination of units ($(P \mid 0) \equiv P, (\nu x)0 \equiv 0$) and *scope extrusion* ($(\nu x)(P \mid Q) \equiv (\nu x)P \mid Q$ if $x \notin \mathtt{fn}(Q)$).

A *configuration* is a closed term of the form $(\nu \mathbf{x})\,(\Pi_{i \in I} A_i(\mathbf{x}_i))$. A *process* $\mathcal{P}$ is a pair $(I, \mathcal{E})$ where I is an *initial configuration* and $\mathcal{E}$ is a set of *parametric equations* of the form $A(\mathbf{x}) = P$ where A is an identifier and P is a term such that 1) every identifier in $\mathcal{P}$ is defined by exactly one equation in $\mathcal{E}$ and 2) if $A(\mathbf{x}) = P$ is an equation, then $\mathtt{fn}(P) \subseteq \{\mathbf{x}\}$. We assume that all the equations are given in the following form:

$$A(\mathbf{x}) = \sum_{i \in I} \pi_i.(\nu \mathbf{x}_i) \left(\prod_{j \in J_i} A_j(\mathbf{x}_j) \right)$$

Operational semantics

Let $\mathcal{P} = (I, \mathcal{E})$ be a process. We define a transition relation on the set of configurations using $\mathcal{E}$ as follows. Let P and Q be configurations. Then $P \to Q$ iff the following conditions are satisfied:

- $P \equiv (\nu \mathbf{u})(A(\mathbf{v}) \mid B(\mathbf{w}) \mid P')$,
- The defining equation of A in $\mathcal{E}$ is of the form $A(\mathbf{x}) = x(\mathbf{x}').(\nu \mathbf{x}'')(M) + M'$,
- The defining equation of B in $\mathcal{E}$ is of the form $B(\mathbf{y}) = \bar{y}(\mathbf{y}').(\nu \mathbf{y}'')(N) + N'$,
- $\sigma = [\mathbf{v}/\mathbf{x}, \mathbf{w}/\mathbf{y}, \mathbf{w}'/\mathbf{x}', \mathbf{z}_A/\mathbf{x}'', \mathbf{z}_B/\mathbf{y}'']$ where $\mathbf{z}_A, \mathbf{z}_B$ are fresh names and $\mathbf{w}'$ is the set of names assigned to $\mathbf{y}'$ under the mapping $[\mathbf{w}/\mathbf{y}]$.
- $\sigma(x) = \sigma(y)$ and
- $Q \equiv (\nu \mathbf{u}, \mathbf{z}_A, \mathbf{z}_B)(\sigma(M) \mid \sigma(N) \mid P')$

We denote such a step by $P \xrightarrow{A(\mathbf{v}),\sigma(x),B(\mathbf{w})} Q$ or simply by $P \to Q$. We can then define the reachability relation $\xrightarrow{*}$ as the reflexive and transitive closure of $\to$. We say that a configuration P is reachable in $\mathcal{P}$ iff $I \xrightarrow{*} P$. We further say that P is coverable if $P \equiv (\nu\mathbf{x})P'$ and there exists $Q \equiv (\nu\mathbf{x})(P' \mid R)$ such that $I \xrightarrow{*} Q$. The coverability problem is to decide if a given configuration P is coverable in a given process $\mathcal{P}$.

Depth-bounded processes

We now define the class of depth-bounded processes. The nesting of restrictions *nest* of a term P is defined inductively as follows: $nest(0) = nest(A(\mathbf{x})) = nest(\pi_1 \cdot P_1 + \pi_2 \cdot P_2) = 0$, $nest((\nu x)P) = 1 + nest(P)$ and $nest(P_1 \mid P_2) = \max\{nest(P_1), nest(P_2)\}$. The *depth* of a term P is the minimal nesting of restrictions of terms in the congruence class of P:

$$depth(P) := \min\{nest(Q) : Q \equiv P\}$$

► **Definition 1.** *A set of configurations $\mathcal{C}$ is called k-depth-bounded if the depth of all configurations in $\mathcal{C}$ is at most k. $\mathcal{C}$ is called depth-bounded if there is some k such that it is k-depth-bounded. A process P is called (k-)depth-bounded if its set of reachable configurations is (k-)depth-bounded.*

► **Example 2.** The following example intuitively demonstrates a system in which there is one "level 0" thread which can spawn "level 1" threads by using a "New1" thread. Then, each level 1 thread can itself spawn "level 2" threads by using their own "New2" threads.

$$Level0(x) = \bar{x}().Level0(x) \quad New1(x) = x().((\nu y)(New1(x) \mid Level1(x,y) \mid New2(y)))$$

$$Level1(x,y) = \bar{y}().Level1(x,y) \quad New2(y) = y().((\nu z)(New2(y) \mid Level2(y,z) \mid New3(z)))$$

$$Level2(y,z) = \bar{z}().Level2(y,z) \quad New3(z) = z().New3(z)$$

Suppose we set $I = (\nu x)(Level0(x) \mid New1(x))$. Then the following is a valid run:

$$\begin{aligned} I &\to (\nu x)(Level0(x) \mid New1(x) \mid (\nu y)(Level1(x,y) \mid New2(y))) \\ &\to (\nu x)(Level0(x) \mid New1(x) \mid (\nu y)(Level1(x,y) \mid New2(y) \mid (\nu z)(Level2(y,z) \mid New3(z)))) \end{aligned}$$

We note that the depth of the last configuration in this run is 3. Indeed, we can show that the depth of any reachable configuration from I is at most 3. Later on, we will see that some of the ideas behind this example are relevant to our lower bound construction.

Our main theorem of the paper is that,

► **Theorem 3.** *The coverability problem for depth-bounded processes is $\mathbf{F}_{\epsilon_0}$-complete.*

Here, we assume that the input consists of a process $\mathcal{P}$ and a number k such that $\mathcal{P}$ is k-depth-bounded. Further, $\mathbf{F}_{\epsilon_0}$ is a complexity class in the *fast-growing hierarchy* of complexity classes [24]. Due to lack of space, we do not define it here. The lower bound behind this theorem is accomplished by giving a log-space reduction from a $\mathbf{F}_{\epsilon_0}$-hard problem. The upper bound is obtained by using results on the length of *controlled bad sequences* over a suitable well-quasi ordering.

We first explain the proof of the lower bound. To do this, we first introduce a model called *nested counter systems with levels* (NCSL) and show that the coverability problem for this model is $\mathbf{F}_{\epsilon_0}$-hard. We then give a reduction from this problem to the coverability problem for depth-bounded processes, thereby proving the lower bound of Theorem 3.

3 Nested counter systems with levels (NCSL)

We now introduce a new model of computation called nested counter systems with levels (NCSL) and prove $\mathbf{F}_{\epsilon_0}$-hardness of coverability for this model. NCSL are closely related to the so-called nested counter systems (NCS) [8]. Indeed, in Section 4, we will recall NCS and prove the hardness result for NCSL by giving a reduction from the coverability problem for NCS.

Before describing NCSL in a formal manner, we give some intuition. A k-NCSL is a generalisation of a usual counter system with *higher-order counters*. Intuitively, a 1-dimensional counter is a usual counter which can add or subtract 1. A 2-dimensional counter can add or subtract 1-dimensional counters, a 3-dimensional counter can add or subtract 2-dimensional counters and so on. A k-NCSL can produce up to k-dimensional counters and then manipulate these counters using "local" rules, i.e., rules which update at most 2 counters at a time. Later on, we will consider the NCS model [8], which allows to update mutliple counters in a single step.

Formally, a k-nested counter system with levels (k-NCSL) is a tuple $\mathcal{N} = (Q, \delta_0, \ldots, \delta_{k-1}, \delta_k)$ where Q is a finite set of *states* and each δ_l is a set of *level-l rules* such that $\delta_l \subseteq \bigcup_{1 \leq i \leq j \leq 2}(Q^i \times Q^j)$. We further enforce that if $l = k$ then $\delta_l \subseteq Q \times Q$. The set $\mathcal{C}_\mathcal{N}$ of *configurations* of $\mathcal{N}$ is defined to be the set of all labelled rooted trees of height at most k, with labels from the set Q.

The operational semantics of $\mathcal{N}$ is defined in terms of the following transition relation $\rightarrow \subseteq \mathcal{C}_\mathcal{N} \times \mathcal{C}_\mathcal{N}$ on configurations: Let $r := ((q_0, \ldots, q_i), (q'_0, \ldots, q'_j)) \in \delta_l$ be a level-l rule with $l \leq k$ and $0 \leq i \leq j \leq 1$. We say that a configuration C can move to the configuration C' using the rule r (denoted by $C \xrightarrow{r} C'$) if *there is a node v_0 at depth l in C with label q_0 and the following holds.*

- **Creation.** Suppose $r = ((q_0), (q'_0, q'_1))$. Then C' is obtained from C by changing the label of v_0 to q'_0, creating a new vertex v_1 with label q'_1 and adding it as child to v_0.
- **1-Preservation.** Suppose $r = ((q_0), (q'_0))$. Then C' is obtained from C by changing the label of v_0 to q'_0.
- **2-Preservation.** Suppose $r = ((q_0, q_1), (q'_0, q'_1))$. Then there is a child v_1 of v_0 in C with label q_1 and C' is obtained from C by changing the labels of v_0 and v_1 to q'_0 and q'_1 respectively.

▶ **Example 4.** Let us consider the 2-NCSL $\mathcal{N}$ given by the states $Q = \{p_i, p'_i, q_i, q'_i : 0 \leq i \leq 4\}$ and consisting of the rules $r_0 \in \delta_0, r_1 \in \delta_1, r_2 \in \delta_2$ where $r_0 = ((q_0, q_1), (q'_0, q'_1)), r_1 = ((p_1), (p'_1, p_2)), r_2 = ((p_2), (p'_2))$. In Figure 1, we illustrate the application of these rules to a configuration of $\mathcal{N}$.

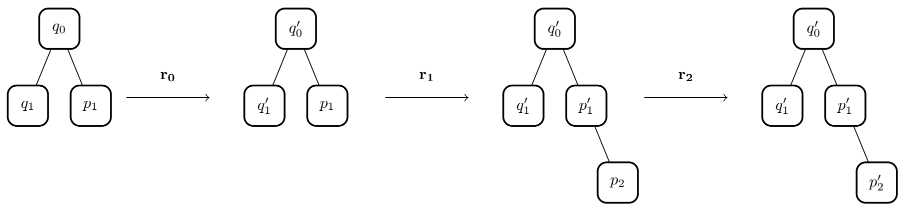

Figure 1 Application of the rules r_0, r_1 and r_2 to a configuration of $\mathcal{N}$, which is described in Example 4.

We say that $C \to C'$ if $C \xrightarrow{r} C'$ for some rule r. We can then define the reachability relation $\xrightarrow{*}$ in a standard manner. Given two states $q_{in}, q_f \in Q$, we say that q_{in} can cover q_f if the (unique) configuration consisting of the single root vertex labelled with q_{in} (also called the initial configuration of $\mathcal{N}$) can reach *some* configuration where the root is labelled by q_f. The coverability problem for an NCSL is then the following: Given an NCSL $\mathcal{N}$ and two states q_{in}, q_f, can q_{in} cover q_f? We prove that

▶ **Theorem 5.** *The coverability problem for NCSL is $\mathbf{F}_{\epsilon_0}$-hard, even when restricted to NCSL which only have creation and 2-preservation rules.*

The proof of Theorem 5 is deferred to Section 4. We shall assume this theorem and first prove the main result of this paper (Theorem 3), i.e., that coverability for depth-bounded π-calculus processes is $\mathbf{F}_{\epsilon_0}$-hard.

3.1 Hardness of coverability for depth-bounded π-calculus processes

Throughout this subsection, we let $\mathcal{N} = (Q, \delta_0, \dots, \delta_{k-1}, \delta_k)$ be a fixed k-NCSL which only has creation and 2-preservation rules. Note that since there are no 1-preservation rules, by definition of a k-NCSL, δ_k is empty and so we will ignore δ_k everywhere in this section. Let q_{in} and q_f be two fixed states of $\mathcal{N}$. We will now construct a depth-bounded process $\mathcal{P}$ and a configuration C of $\mathcal{P}$ such that C can be covered in $\mathcal{P}$ iff q_f can be covered from q_{in} in $\mathcal{N}$.

Process identifiers, names and the initial configuration

To construct $\mathcal{P}$, we have to define an initial configuration and a set of parametric equations. We begin by specifying the set of names and the process identifiers that we shall use in the equations. Based on these names and identifiers, we define the initial configuration and also introduce an injective mapping $\mathbb{B}$ from the set of configurations of $\mathcal{N}$ to the set of configurations of $\mathcal{P}$. This map will be useful to prove the correctness of our reduction.

Process identifiers and names. For each $1 \le i \le k$, we will have a process identifier *start*$[i]$. For each $0 \le i \le k$ and each state q of $\mathcal{N}$, we will have an identifier $q[i]$. Notice that each process identifier is of the form $a[b]$ where $a \in Q \cup \{start\}$ and $0 \le b \le k$. The first part "a" will be called the *base* of the identifier and the second part "b" will be called the *grade* of the identifier. The arities of the identifiers are as follows: The arity of each *start*$[i]$ will be $|\delta_{i-1}|$. For every state q of $\mathcal{N}$, the arity of $q[0]$ will be $|\delta_0|$, the arity of $q[k]$ will be $|\delta_{k-1}|$ and the arity of every other $q[i]$ will be $|\delta_{i-1}| + |\delta_i|$.

The set of names that we will be using in the equations will be the set of rules of $\mathcal{N}$, i.e., $\delta_0 \cup \delta_1 \cup \dots \cup \delta_{k-1}$. For each δ_i, we let $\mathbf{n}_i$ denote some fixed vector comprising all the names from δ_i. We also assume that there is another countably infinite set of names needed to describe the configurations of $\mathcal{P}$. We note that this latter set is not part of the input.

A mapping. We now introduce an injective map from the set of configurations of $\mathcal{N}$ to the set of configurations of $\mathcal{P}$. Let C be a configuration of the NCSL $\mathcal{N}$. To C, we assign a unique configuration of $\mathcal{P}$ (denoted by $\mathbb{B}(C)$) as follows: Let the set of vertices of C be V and let the set of internal vertices of C (the root and the other non-leaf vertices) be IV. $\mathbb{B}(C)$ is then defined as the configuration

$$(\nu \mathbf{z})\ (\Pi_{v \in V} A_v(\mathbf{x}_v, \mathbf{y}_v) \mid \Pi_{v \in IV} B_v(\mathbf{y}_v))$$

where $\{\mathbf{z}\} = \cup_{v \in V}\{\mathbf{x}_v, \mathbf{y}_v\}$ and for each v,

- $\{\mathbf{x}_v\} \cap \{\mathbf{y}_v\} = \emptyset$,
- If the label of v in C is q and v is at depth l, then $A_v = q[l]$ and $B_v = \mathit{start}[l+1]$,
- If v is the root, then $\mathbf{x}_v$ is the empty vector. If v is a leaf, then $\mathbf{y}_v$ is the empty vector. Otherwise, if v is at depth l, then $\mathbf{x}_v$ is of size $|\delta_{l-1}|$ and $\mathbf{y}_v$ is of size $|\delta_l|$.
- For any v', if v' is a child of v, then $\mathbf{x}_{v'} = \mathbf{y}_v$ and $\{\mathbf{y}_{v'}\} \cap \{\mathbf{x}_v\} = \emptyset$; if v' is a sibiling of v, then $\mathbf{x}_{v'} = \mathbf{x}_v$ and $\{\mathbf{y}_{v'}\} \cap \{\mathbf{y}_v\} = \emptyset$; otherwise, $\{\mathbf{x}_v, \mathbf{y}_v\} \cap \{\mathbf{x}_{v'}, \mathbf{y}_{v'}\} = \emptyset$.

To give an intuition behind this mapping, let us look at $\mathbb{B}(C)$ from the perspective of graphs. We construct a graph where there is a vertex for each $A_v(\mathbf{x}_v, \mathbf{y}_v)$ and each $B_v(\mathbf{y}_v)$ and we connect two such vertices by an edge if they share at least one free name and the corresponding identifiers have different grades. By the requirements given above, this would imply that the graph that we get is a tree which has a "copy" of C as a subgraph, along with a new leaf vertex added to every internal vertex of C. Ignoring the new leaf vertices for now, this means that $\mathbb{B}(C)$ can be thought of as a "representation" of C in the process $\mathcal{P}$. The parametric equations that we shall construct will make sure that if $\mathbb{B}(C)$ can move to a new configuration P, then P will be a representation of C' for some C' such that $C \to C'$ in the NCSL $\mathcal{N}$.

We now have the following lemma which proves depth-boundedness of any configuration of the form $\mathbb{B}(C)$. The intuition behind this lemma is that the "graph" of $\mathbb{B}(C)$ contains a copy of C as a subgraph along with some other additional leaf vertices. Hence, since the depth of C is bounded by k, we can expect that the depth of $\mathbb{B}(C)$ is also bounded.

▶ **Lemma 6** (Depth-boundedness). *For any configuration C, the depth of $\mathbb{B}(C)$ is at most $\sum_{l=0}^{k-1} |\delta_l|$.*

Proof. Let V and IV be the set of vertices and internal vertices of C respectively. For any vertex n, let C_n be the (labelled) subtree of C rooted at n and let V_n and IV_n be the set of vertices and internal vertices of C_n respectively.

We know that $\mathbb{B}(C)$ is of the form $(\nu \mathbf{z})\ (\Pi_{v \in V} A_v(\mathbf{x}_v, \mathbf{y}_v) \mid \Pi_{v \in IV} B_v(\mathbf{y}_v))$. Let $\mathbb{B}(C_\mathsf{n})$ be the sub-process term of $\mathbb{B}(C)$ given by $\Pi_{v \in V_\mathsf{n}} A_v(\mathbf{x}_v, \mathbf{y}_v) \mid \Pi_{v \in IV_\mathsf{n}} B_v(\mathbf{y}_v)$ and let $\{\mathbf{z}_\mathsf{n}\} = \cup_{v \in V_\mathsf{n}} \{\mathbf{x}_v, \mathbf{y}_v\}$.

By induction on the height h of the vertex n in the tree C, we will now show that the depth of $(\nu \mathbf{z}_\mathsf{n})\ \mathbb{B}(C_\mathsf{n})$ is at most $\sum_{l=\max\{k-1-h,0\}}^{k-1} |\delta_l|$. For the base case, when n is a leaf and C_n is a tree with a single node, we have that $(\nu \mathbf{z}_\mathsf{n})\ \mathbb{B}(C_\mathsf{n}) \equiv (\nu \mathbf{x}_\mathsf{n})\ q[k](\mathbf{x}_\mathsf{n})$ for some q and some vector $\mathbf{x}_\mathsf{n}$ of size $|\delta_{k-1}|$. This shows that the claim is true for the base case.

For the induction step, let $Ch(\mathsf{n})$ be the children of n. By the requirements imposed upon $\mathbb{B}(C)$, we can use the scope extrusion rule to write $(\nu \mathbf{z}_\mathsf{n})\ \mathbb{B}(C_\mathsf{n})$ as $(\nu \mathbf{x}_\mathsf{n}, \mathbf{y}_\mathsf{n})\ (A_\mathsf{n}(\mathbf{x}_\mathsf{n}, \mathbf{y}_\mathsf{n}) \mid B_\mathsf{n}(\mathbf{y}_\mathsf{n}) \mid \Pi_{v \in Ch(\mathsf{n})} (\nu (\mathbf{z}_v \setminus \mathbf{y}_\mathsf{n}))\ \mathbb{B}(C_v))$. By induction hypothesis, we have that the depth of each $(\nu \mathbf{z}_v)\ \mathbb{B}(C_v))$ is $\sum_{l=k-h}^{k-1} |\delta_l|$. This then implies that the depth of $(\nu \mathbf{x}_\mathsf{n}, \mathbf{y}_\mathsf{n})\ (A_\mathsf{n}(\mathbf{x}_\mathsf{n}, \mathbf{y}_\mathsf{n}) \mid B_\mathsf{n}(\mathbf{y}_\mathsf{n}) \mid \Pi_{v \in Ch(\mathsf{n})} (\nu (\mathbf{z}_v \setminus \mathbf{y}_\mathsf{n}))\ \mathbb{B}(C_v))$ is at most $\sum_{l=k-1-h}^{k-1} |\delta_l|$ if n is not the root. If n is the root, then the depth becomes at most $\sum_{l=0}^{k-1} |\delta_l|$ because $\mathbf{x}_\mathsf{n} = \emptyset$. Hence, the induction step is complete.

Since $\mathbb{B}(C) \equiv (\nu \mathbf{z}_\mathsf{n})\ \mathbb{B}(C_\mathsf{n})$ where n is the root, it follows that the depth of $\mathbb{B}(C)$ is at most $\sum_{l=0}^{k-1} |\delta_l|$. ◀

Initial configuration. Recall that for each $i \in \{0, \dots, k-1\}$, we let $\mathbf{n}_i$ denote some fixed vector comprising all the names from δ_i. We then take the initial configuration of $\mathcal{P}$ to be $(\nu \mathbf{n}_0)(q_{in}[0](\mathbf{n}_0) \mid \mathit{start}[1](\mathbf{n}_0))$. Note that the initial configuration of $\mathcal{P}$ is the image of the initial configuration of $\mathcal{N}$ under the $\mathbb{B}$ mapping.

Parametric equations

Before we describe the parametric equations, we set up some notation. Let $r = ((q_0, \ldots, q_i), (q'_0,$
$\ldots, q'_j))$ be a rule of the NCSL $\mathcal{N}$. By definition of creation and 2-preservation rules, it has to be the case that $i \leq 1$ and $j = 1$. In the sequel, for the sake of uniformity across all rules, we adopt the following nomenclature: If $i = 0$, we let $q_1 = \mathit{start}$. In this way, we can always associate a (unique) tuple $((q_0, q_1), (q'_0, q'_1))$ with any rule r.

Let $r = ((p, q), (p', q'))$ be a rule of $\mathcal{N}$. We say that the tuple (p, q) (resp. (p', q')) is the *precondition* (resp. *postcondition*) of r and we let $\mathtt{pre}^r_{\mathtt{fi}} := p$, $\mathtt{pre}^r_{\mathtt{se}} := q$, $\mathtt{post}^r_{\mathtt{fi}} := p'$ and $\mathtt{post}^r_{\mathtt{se}} := q'$.

We will set up the parametric equations in such a way so that $C \to C'$ is a step in $\mathcal{N}$ iff $\mathbb{B}(C) \to \mathbb{B}(C')$. Intuitively this is accomplished by ensuring that if $r = ((p, q), (p', q')) \in \delta_l$ is a rule of $\mathcal{N}$, then a thread with identifier $p[l]$ can output along a name and go to $p'[l]$ and a thread with identifier $q[l+1]$ can receive along the same name and go to $q'[l+1]$.

Equations for identifiers of grade 0. For any $q \in Q$, the equation for $q[0]$ is,

$$q[0](\mathbf{n}_0) := \sum_{r \in \delta_0,\ \mathtt{pre}^r_{\mathtt{fi}} = q} \overline{r}().\ \mathtt{post}^r_{\mathtt{fi}}[0](\mathbf{n}_0)$$

Intuitively, this equation corresponds to a thread with identifier $q[0]$ trying to execute some rule $r \in \delta_0$ for which $q = \mathtt{pre}^r_{\mathtt{fi}}$ and then becoming $\mathtt{post}^r_{\mathtt{fi}}[0]$.

Equations for identifiers of grade $1 \leq i \leq k-1$. Recall that the arity of any such identifier is $|\delta_{i-1}| + |\delta_i|$, except for identifiers with base *start*, for which it is $|\delta_{i-1}|$.

- For any $q \in Q$, we have

$$q[i](\mathbf{n}_{i-1}, \mathbf{n}_i) := \sum_{r \in \delta_i,\ \mathtt{pre}^r_{\mathtt{fi}} = q} \overline{r}().\ \mathtt{post}^r_{\mathtt{fi}}[i](\mathbf{n}_{i-1}, \mathbf{n}_i) + \sum_{r \in \delta_{i-1},\ \mathtt{pre}^r_{\mathtt{se}} = q} r().\ \mathtt{post}^r_{\mathtt{se}}[i](\mathbf{n}_{i-1}, \mathbf{n}_i)$$

  Intuitively, the first summand of the equation corresponds to to a thread with identifier $q[i]$ trying to execute some rule $r \in \delta_i$ for which $q = \mathtt{pre}^r_{\mathtt{fi}}$ and then becoming $\mathtt{post}^r_{\mathtt{fi}}[i]$. The second summand corresponds to a thread with identifier $q[i]$ trying to execute some rule $r \in \delta_{i-1}$ for which $q = \mathtt{pre}^r_{\mathtt{se}}$ and then becoming $\mathtt{post}^r_{\mathtt{se}}[i]$.
- For the *start* base, we have

$$\mathit{start}[i](\mathbf{n}_{i-1}) := \sum_{r \in \delta_{i-1},\ \mathtt{pre}^r_{\mathtt{se}} = \mathit{start}} r().\ \Big((\nu \mathbf{n}_i)\ \mathit{start}[i](\mathbf{n}_{i-1}) \mid \mathtt{post}^r_{\mathtt{se}}[i](\mathbf{n}_{i-1}, \mathbf{n}_i) \mid \mathit{start}[i{+}1](\mathbf{n}_i) \Big)$$

  Intuitively, this equation is responsible for spawning new threads of grade i with base in Q, when an appropriate output action is taken by some thread of grade $i-1$ with base in Q. First, if a thread with identifier $\mathit{start}[i]$ receives a message along some channel corresponding to some rule $r \in \delta_{i-1}$ with $\mathtt{pre}^r_{\mathtt{se}} = \mathit{start}$, then a fresh set of names (denoted by $\mathbf{n}_i$) are created. After that, the thread retains its identifier and two new threads are spawned, $\mathtt{post}^r_{\mathtt{se}}[i](\mathbf{n}_{i-1}, \mathbf{n}_i)$ and $\mathit{start}[i+1](\mathbf{n}_i)$. We note that these equations have a similar flavor to that of the equations for $New1$ and $New2$ given in Example 2.

Equations for identifiers of grade k. Recall that the arity of any identifier with grade k is $|\delta_{k-1}|$.

- For any $q \in Q$, we have
$$q[k](\mathbf{n}_{k-1}) := \sum_{r \in \delta_{k-1},\ \mathtt{pre}^r_{\mathtt{se}} = q} r().\ \mathtt{post}^r_{\mathtt{se}}[k](\mathbf{n}_{k-1})$$
- For the *start* base, we have
$$\mathit{start}[k](\mathbf{n}_{k-1}) := \sum_{r \in \delta_{k-1},\ \mathtt{pre}^r_{\mathtt{se}} = \mathit{start}} r().\ (\mathtt{post}^r_{\mathtt{se}}[k](\mathbf{n}_{k-1}) \mid \mathit{start}[k](\mathbf{n}_{k-1}))$$

The intuitions behind these equations are the same as the one for the previous case.

3.2 Proof of correctness

We now formally show the proof of correctness of our reduction. We begin with a lemma which shows that the constructed process $\mathcal{P}$ can simulate the NCSL $\mathcal{N}$.

▶ **Lemma 7** ($\mathcal{P}$ simulates $\mathcal{N}$)**.** *Suppose $C \to C'$ is a step in $\mathcal{N}$. Then $\mathbb{B}(C) \to \mathbb{B}(C')$.*

Proof. Let $r = ((p,q),(p',q')) \in \delta_l$ for some $0 \le l \le k-1$ such that $C \xrightarrow{r} C'$. Let V be the set of vertices of C and let IV be the set of internal vertices of C. This means that there is a vertex n in C at depth l such that the label of n in C is p.

Let $\mathbb{B}(C) \equiv (\nu\mathbf{z})\ (\Pi_{v \in V} A_v(\mathbf{x}_v, \mathbf{y}_v) \mid \Pi_{v \in IV} B_v(\mathbf{y}_v))$. By definition of the map $\mathbb{B}$, it has to be the case that $A_{\mathsf{n}} = p[l]$. We have two cases:

- Suppose $q \neq \mathit{start}$. Then there has to be a child n' of n in C such that its label in C is q. Hence, $A_{\mathsf{n}'} = q[l+1]$. Further, $\mathbf{y}_{\mathsf{n}} = \mathbf{x}_{\mathsf{n}'}$. By construction of the parametric equations, this means that $\mathbb{B}(C)$ can reach P where
$$P \equiv (\nu\mathbf{z})\ (\Pi_{v \in V \setminus \{\mathsf{n},\mathsf{n}'\}} A_v(\mathbf{x}_v, \mathbf{y}_v) \mid p'[l](\mathbf{x}_{\mathsf{n}}, \mathbf{y}_{\mathsf{n}}) \mid \ q'[l+1](\mathbf{x}_{\mathsf{n}'}, \mathbf{y}_{\mathsf{n}'}) \mid \Pi_{v \in IV} B_v(\mathbf{y}_v))$$
It is then easy to see that $P \equiv \mathbb{B}(C')$.
- Suppose $q = \mathit{start}$. Then $B_{\mathsf{n}}(\mathbf{y}_{\mathsf{n}}) = \mathit{start}[l+1](\mathbf{y}_{\mathsf{n}})$. By construction of the parametric equations, this means that $\mathbb{B}(C)$ can reach P given by
$$P \equiv (\nu\mathbf{z},\mathbf{z}')\ (\Pi_{v \in V \setminus \{\mathsf{n}\}} A_v(\mathbf{x}_v, \mathbf{y}_v) \mid p'[l](\mathbf{x}_{\mathsf{n}}, \mathbf{y}_{\mathsf{n}}) \mid \Pi_{v \in IV} B_v(\mathbf{y}_v) \mid q'[l+1](\mathbf{y}_{\mathsf{n}}, \mathbf{z}') \mid \mathit{start}[l+2](\mathbf{z}'))$$
where the last term $\mathit{start}[l+2](\mathbf{z}')$ is not present if $l = k-1$. It is then easy to see that $P \equiv \mathbb{B}(C')$. ◀

Next we show that $\mathcal{N}$ can also simulate $\mathcal{P}$.

▶ **Lemma 8** ($\mathcal{N}$ simulates $\mathcal{P}$)**.** *Suppose $\mathbb{B}(C) \to P$. Then there exists a configuration C' of $\mathcal{N}$ such that $C \to C'$ and $P \equiv \mathbb{B}(C')$.*

Proof. Let V be the vertices of C and let IV be the set of internal vertices of C. Let $\mathbb{B}(C) \equiv (\nu\mathbf{z})\ (\Pi_{v \in V} A_v(\mathbf{x}_v, \mathbf{y}_v) \mid \Pi_{v \in IV} B_v(\mathbf{y}_v))$ and let $\mathbb{B}(C) \xrightarrow{T_v(\mathbf{w}_v), c, T_{v'}(\mathbf{w}_{v'})} P$.

By construction of the parametric equations, it must be the case that $T_v(\mathbf{w}_v) = A_{\mathsf{n}}(\mathbf{x}_{\mathsf{n}}, \mathbf{y}_{\mathsf{n}})$ for some node n and c must belong to $\{\mathbf{y}_{\mathsf{n}}\}$. Let $A_{\mathsf{n}} = p[l]$. Since $c \in \{\mathbf{y}_n\}$, by definition of $\mathbb{B}(C)$, c can only be shared among the free names of the threads in $\{A_{\mathsf{n}'}(\mathbf{x}_{\mathsf{n}'}, \mathbf{y}_{\mathsf{n}'}) : \mathsf{n}' \text{ is a child of } \mathsf{n}\} \cup \{B_{\mathsf{n}}(\mathbf{y}_{\mathsf{n}})\}$. We now consider two cases:

- Suppose $T_{v'}(\mathbf{w}_{v'}) = A_{\mathsf{n}'}(\mathbf{x}_{\mathsf{n}'}, \mathbf{y}_{\mathsf{n}'})$ for some n' which is a child of n. Let $A_{\mathsf{n}'} = q[l+1]$. Since we have $\mathbb{B}(C) \xrightarrow{T_v(\mathbf{w}_v), c, T_{v'}(\mathbf{w}_{v'})} P$, by construction of the equations it has to be the case that there is a rule $r \in \delta_l$ of $\mathcal{N}$ such that $\mathtt{pre}^r_{\mathtt{fi}} = p$, $\mathtt{pre}^r_{\mathtt{se}} = q$ and

$$P \equiv (\nu \mathbf{z})\ (\Pi_{v \in V \setminus \{\mathsf{n},\mathsf{n}'\}} A_v(\mathbf{x}_v, \mathbf{y}_v) \mid p'[l](\mathbf{x}_{\mathsf{n}}, \mathbf{y}_{\mathsf{n}}) \mid q'[l+1](\mathbf{x}_{\mathsf{n}'}, \mathbf{y}_{\mathsf{n}'}) \mid \Pi_{v \in IV} B_v(\mathbf{y}_v))$$

  where $p' = \mathtt{post}^r_{\mathtt{fi}}$ and $q' = \mathtt{post}^r_{\mathtt{se}}$ respectively. Since $A_{\mathsf{n}} = p[l]$ and $A_{\mathsf{n}'} = q[l+1]$, it must be the case that the depth of n in C is l and the labels of n and n' in C are p and q respectively. It follows that there exists C' such that $C \xrightarrow{r} C'$. It is then easy to verify that $\mathbb{B}(C') \equiv P$.
- Suppose $T_{v'}(\mathbf{w}_{v'}) = B_{\mathsf{n}}(\mathbf{y}_{\mathsf{n}})$. We know that $B_{\mathsf{n}} = \mathit{start}[l+1]$. Since it is the case that $\mathbb{B}(C) \xrightarrow{T_v(\mathbf{w}_v), c, T_{v'}(\mathbf{w}_{v'})} P$, by construction of the parametric equations it must be that there is a rule $r \in \delta_l$ of $\mathcal{N}$ such that $\mathtt{pre}^r_{\mathtt{fi}} = p$, $\mathtt{pre}^r_{\mathtt{se}} = \mathit{start}$ and

$$P \equiv (\nu \mathbf{z}, \mathbf{z}')\ (\Pi_{v \in V \setminus \{\mathsf{n}\}} A_v(\mathbf{x}_v, \mathbf{y}_v) \mid p'[l](\mathbf{x}_{\mathsf{n}}, \mathbf{y}_{\mathsf{n}}) \mid \Pi_{v \in IV} B_v(\mathbf{y}_v) \mid q'[l+1](\mathbf{y}_{\mathsf{n}}, \mathbf{z}') \mid \mathit{start}[l+2](\mathbf{z}'))$$

  where the last term $\mathit{start}[l+2](\mathbf{z}')$ is not present if $l = k-1$ and $p' = \mathtt{post}^r_{\mathtt{fi}}, q' = \mathtt{post}^r_{\mathtt{se}}$ respectively. Since $A_{\mathsf{n}} = p[l]$, it must be the case that the depth of n in C is l and the label of n in C is p. It follows then that there exists C' such that $C \xrightarrow{r} C'$. It is then easy to verify that $\mathbb{B}(C') \equiv P$. ◀

Note that the initial configuration I of $\mathcal{P}$ is simply the image of the initial configuration of $\mathcal{N}$ under the map $\mathbb{B}$. Hence, using Lemmas 6 and 8, we can conclude that

▶ **Corollary 9.** *The process $\mathcal{P}$ is K-depth-bounded where $K = \sum_{l=0}^{k-1} |\delta_l|$.*

We then get the following theorem, whose proof follows in a straightforward manner by combining Lemmas 7 and 8.

▶ **Theorem 10.** *$C \xrightarrow{*} C'$ is a run in $\mathcal{N}$ iff $\mathbb{B}(C) \xrightarrow{*} \mathbb{B}(C')$ is a run in the process $\mathcal{P}$. Consequently q_{in} can cover q_f in $\mathcal{N}$ iff $(\nu \mathbf{n}_0)\ (q_f[0](\mathbf{n}_0))$ can be covered from the initial configuration I of $\mathcal{P}$.*

Hence, we have

▶ **Corollary 11.** *Coverability of depth-bounded processes is $\mathbf{F}_{\epsilon_0}$-hard.*

4 Nested counter systems (NCS)

We now prove Theorem 5, by giving a reduction from the coverability problem for *nested counter systems* (NCS) which is known to be $\mathbf{F}_{\epsilon_0}$-hard. We first recall the definition of NCS, which we present in a way that is akin to [2].

A k-nested counter system (k-NCS) is a tuple $\mathcal{N} = (Q, \delta)$ where Q is a finite set of *states* and $\delta \subseteq \bigcup_{1 \le i,j \le k+1} (Q^i \times Q^j)$ is a set of *rules.* The set $\mathcal{C}_{\mathcal{N}}$ of *configurations* of $\mathcal{N}$ is defined to be the set of all labelled rooted trees of height atmost k, with labels from the set Q.

The operational semantics of $\mathcal{N}$ is defined in terms of the following transition relation $\rightarrow \subseteq \mathcal{C}_{\mathcal{N}} \times \mathcal{C}_{\mathcal{N}}$ on configurations: Let $r := ((q_0, \ldots, q_i), (q'_0, \ldots, q'_j)) \in \delta$ be a rule with $i \le j \le k$. We say that a configuration C can move to the configuration C' using the rule r (denoted by $C \xrightarrow{r} C'$), if there is a path $v_0, v_1 \ldots, v_i$ in C starting at the root such that for every $0 \le l \le i$, the label of v_l is q_l and, C' is obtained from C by 1) for every $0 \le l \le i$, changing the label of each v_l to q'_l and 2) for every $i+1 \le l \le j$, creating a new vertex v_l with label q'_l and adding it as a child to v_{l-1}.

Similarly, suppose $r := ((q_0, \dots, q_i), (q'_0, \dots, q'_j)) \in \delta$ is a rule with $j < i \leq k$. Then $C \xrightarrow{r} C'$ if there is a path $v_0, v_1, \dots, v_i$ in C starting at the root such that for every $0 \leq l \leq i$, the label of v_l is q_l and, C' is obtained from C by 1) for every $0 \leq l \leq j$, changing the label of each v_l to q'_l and 2) removing the subtree rooted at the node v_{j+1}.

▶ **Example 12** (Example from [2]). Let us consider the NCS $\mathcal{N}$ given by the states $Q = \{p_i, p'_i, q_i, q'_i : 0 \leq i \leq 4\}$ and consisting of the following rules: $r_0 = ((q_0, q_1), (q'_0, q'_1, q'_2)), r_1 = ((q'_0, q_3, q_2), (p_0)), r_2 = ((p_0), (p'_0))$. In Figure 2, we illustrate the application of these rules to a configuration of $\mathcal{N}$.

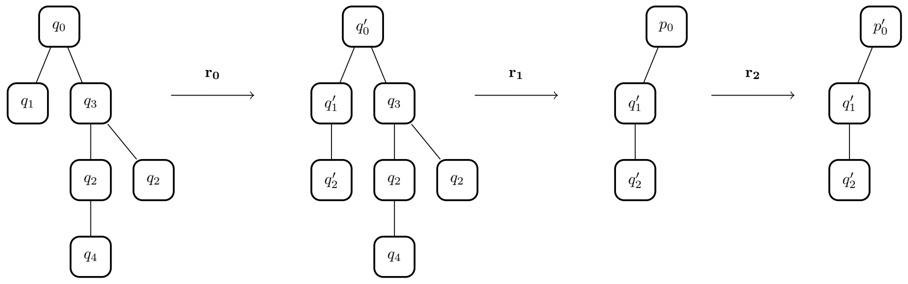

Figure 2 Application of the rules r_0, r_1 and r_2 to a configuration of $\mathcal{N}$, which is described in Example 12.

Similar to NCSL, we can define the notions of $C \to C', C \xrightarrow{*} C'$ and a state q_{in} covering another state q_f. It is known that the coverability problem for NCS is $\mathbf{F}_{\epsilon_0}$-hard (Theorem 7 of [8]).

We note that the rules of an NCS act "globally", in the sense that it allows to update the value of (potentially) k many counters in one step. This is in contrast to NCSL, where we can update the value of at most two counters at a time. While it is not particularly surprising that this "global" update can be replaced by a series of "local" updates (hence giving a reduction from NCS to NCSL), the construction is not entirely trivial and requires some intricate arguments in order to prove its correctness.

A special case of NCS

We make a small remark which will help us simplify our reduction later on. Let $\mathcal{N} = (Q, \delta)$ be a k-NCS and let $q_{in}, q_f \in Q$. From $\mathcal{N}$, we construct a new k-NCS $\mathcal{N}'$ as follows: First we add a new state *end*. Then, if $r = ((q_0, \dots, q_i), (q'_0, \dots, q'_j)) \in \delta$ with $j < i \leq k$, we replace r with the rule $r' := ((q_0, \dots, q_i), (q'_0, \dots, q'_j, \underbrace{end, \dots, end}_{i-j \text{ times}}))$. Intuitively, we are replacing all rules which destroy some counters with corresponding rules that simply convert those counters to the state *end*. It can be easily verified that coverability of q_f from q_{in} is preserved while doing this operation. Hence, from here on, we assume that whenever $\mathcal{N} = (Q, \delta)$ is a k-NCS and $r = ((q_0, \dots, q_i), (q'_0, \dots, q'_j)) \in \delta$ then $i \leq j$.

4.1 Hardness of coverability for NCSL

We shall prove Theorem 5 by giving a reduction from the coverability problem for NCS. Let $k \geq 1$ and let $\mathcal{N} = (Q, \delta)$ be a k-NCS with two fixed states q_{in} and q_f. By the argument given in the previous paragraph, we can assume that if $r = ((q_0, \dots, q_i), (q'_0, \dots, q'_j)) \in \delta$

then $i \leq j$. We shall now construct a k-NCSL $\mathcal{N}' = (Q', \delta'_0, \dots, \delta'_{k-1})$ and two states q'_{in} and q'_f of $\mathcal{N}'$ such that q'_{in} can cover q'_f in $\mathcal{N}'$ iff q_{in} can cover q_f in $\mathcal{N}$. This will then prove Theorem 5. We begin by describing the states of $\mathcal{N}'$.

States of $\mathcal{N}'$. For every state q of $\mathcal{N}$, we will have two states $q[\top]$ and $q[\bot]$. Further, for every rule r of $\mathcal{N}$, we will have four states $\texttt{rec}^r[\top], \texttt{rec}^r[\bot], \texttt{fwd}^r[\top]$ and $\texttt{fwd}^r[\bot]$. Notice that each state of $\mathcal{N}'$ is of the form $a[b]$ where $a \in Q \cup \{\texttt{rec}^r, \texttt{fwd}^r : r \in \delta\}$ and $b \in \{\top, \bot\}$. If a node v in a configuration C has as its label $a[b]$, then 'a' will be called its *base*. Further, if $b = \top$ (resp. $b = \bot$), then v will be called as a *leader node* (resp. *follower node*).

Good configurations of $\mathcal{N}'$. A configuration C is called good if the root of C is a leader, all other nodes are followers and the base of all the nodes of C belong to Q. Notice that there is a straightforward bijection between the set of all configurations of $\mathcal{N}$ and the set of all *good* configurations of $\mathcal{N}'$. This bijection will be denoted by $\mathbb{M}$.

Rules of $\mathcal{N}'$. Before we describe the rules of $\mathcal{N}'$, we will state two invariants that will always be maintained by our construction. The first one is that, in any configuration reachable from a good configuration, exactly one node will be a leader. The second invariant is that, every rule of $\mathcal{N}'$ will have a leader state in its precondition. Combined with the first invariant, this will intuitively ensure that the rules that can be fired from reachable configurations are limited and will help us simplify the proof of correctness of our reduction.

We now describe the rules of $\mathcal{N}'$. Let $r = ((q_0, \dots, q_i), (q'_0, \dots, q'_j))$ be a rule of $\mathcal{N}$. Corresponding to rule r, we will have the following set of rules in $\mathcal{N}'$. (In the following, we adopt the convention that if the name of a rule has a subscript $0 \leq l \leq j$, then that rule belongs to δ'_l).

- $start^r_0 := ((q_0[\top]), (\texttt{rec}^r[\top]))$.
- For every $0 \leq l \leq i-1$, we have a rule $begin^r_l := ((\texttt{rec}^r[\top], q_{l+1}[\bot]), (\texttt{fwd}^r[\bot], \texttt{rec}^r[\top]))$.
- For every $i \leq l \leq j-1$, we have a rule $begin^r_l := ((\texttt{rec}^r[\top]), (\texttt{fwd}^r[\bot], \texttt{rec}^r[\top]))$.
- $middle^r_j := ((\texttt{rec}^r[\top]), (\texttt{fwd}^r[\top]))$.
- For every $0 \leq l \leq j-1$, we have a rule $end^r_l := ((\texttt{fwd}^r[\bot], \texttt{fwd}^r[\top]), (\texttt{fwd}^r[\top], q'_{l+1}[\bot]))$.
- $finish^r_0 := ((\texttt{fwd}^r[\top]), q'_0[\top])$.

4.2 Proof of correctness

The intuitive idea behind the above gadget is given by the run demonstrated in the following lemma.

▶ **Lemma 13** ($\mathcal{N}'$ simulates $\mathcal{N}$). *Suppose $C \xrightarrow{r} C'$ is a step in the NCS $\mathcal{N}$. Then, there is a run $\mathbb{M}(C) \xrightarrow{*} \mathbb{M}(C')$ in the NCSL $\mathcal{N}'$.*

Proof. Let $r = ((q_0, \dots, q_i), (q'_0, \dots, q'_j))$. Since $C \xrightarrow{r} C'$ is a step in $\mathcal{N}$, it follows that there is a path starting at the root of C labelled by $q_0, \dots, q_i$. It follows that in $\mathbb{M}(C)$ there is a path P starting at the root labelled by $q_0[\top], q_1[\bot], q_2[\bot], \dots, q_i[\bot]$. We now execute a sequence of rules according to the gadget for r as follows:

- First, using $start^r_0$, we change the label of the root from $q_0[\top]$ to $\texttt{rec}^r[\top]$.
- Next, by firing $begin^r_0, \dots, begin^r_{i-1}$ in this order, we change the labels of the nodes in the path P to $\underbrace{\texttt{fwd}^r[\bot], \dots, \texttt{fwd}^r[\bot]}_{i \text{ times}}, \texttt{rec}^r[\top]$.

- Then, by firing $begin^r_i, \dots, begin^r_{j-1}$ in this order, we add $j - i$ new nodes to the path P and get a new path P' of length $j+1$ whose labels are $\underbrace{\mathtt{fwd}^r[\bot], \dots, \mathtt{fwd}^r[\bot]}_{j \text{ times}}, \mathtt{rec}^r[\top]$.
- We use $middle^r_j$ to change the label of the last node in P' from $\mathtt{rec}^r[\top]$ to $\mathtt{fwd}^r[\top]$.
- Then, by firing $end^r_{j-1}, \dots, end^r_0$ in this order, we change the labels of the nodes in the path P' to $\mathtt{fwd}^r[\top], q'_1[\bot], \dots, q'_j[\bot]$.
- Finally, we use $finish^r_0$ to change the label of the root from $\mathtt{fwd}^r[\top]$ to $q'_0[\top]$.

It can be easily verified that the resulting configuration D is such that $D = \mathbb{M}(C')$. ◀

We now present a converse to the above lemma which shows that a simulation in the other direction is also possible.

▶ **Lemma 14** ($\mathcal{N}$ simulates $\mathcal{N}'$)**.** *Suppose $C \xrightarrow{*} C'$ is a path of non-zero length in $\mathcal{N}'$ such that 1) C is a good configuration and 2) in all the configurations between C and C', the base of the root is not in Q. Then, C' is a good configuration and there is a rule r such that $\mathbb{M}^{-1}(C) \xrightarrow{r} \mathbb{M}^{-1}(C')$.*

Proof sketch. Let $P := C \to \gamma_0 \to \gamma_1 \dots \to C'$. The essential idea behind this lemma is that since C is a good configuration, the root node is a leader node and by the construction of the rules it must be the case that the first step must be of the form $C \xrightarrow{start^r_0} \gamma_0$ for some rule r. Then, by using the invariant that exactly one node is leader at all times and by using the construction of the rules, we can essentially show that P must be a path of the same form as the one given in the proof of Lemma 13. Having proved that, we can then show that in the NCS $\mathcal{N}$, $\mathbb{M}^{-1}(C) \xrightarrow{r} \mathbb{M}^{-1}(C')$. ◀

Because of these two "simulation" lemmas, we then get

▶ **Theorem 15.** *q_{in} can cover q_f in $\mathcal{N}$ iff $q_{in}[\top]$ can cover $q_f[\top]$ in $\mathcal{N}'$.*

4.3 Wrapping up

The previous theorem implies that coverability for NCSL is $\mathbf{F}_{\epsilon_0}$-hard. To prove Theorem 5, we need to show the same for NCSL with only creation and 2-preservation rules. We now show that 1-preservation rules can be replaced with creation rules in an NCSL while maintaining coverability.

Given a k-NCSL $\mathcal{N}$ with two states q_{in}, q_f, we can remove all 1-preservation rules whilst preserving coverability as follows: We first add a new state *end*. Then if $r = ((q_0), (q'_0))$ is a 1-preservation rule in $\mathcal{N}$, we replace r with $r = ((q_0), (q'_0, end))$. It can be easily seen that doing this procedure gives us a $(k+1)$-NCSL $\mathcal{N}'$ such that q_{in} can cover q_f in $\mathcal{N}'$ iff q_{in} can cover q_f in $\mathcal{N}$. Hence Theorem 5 follows.

5 Upper bound for coverability of depth-bounded processes

We now prove the upper bound claim made in Theorem 3. Let $\mathcal{P} = (I, \mathcal{E})$ be a fixed k-depth-bounded process. By introducing new identifiers and equations if necessary, we can assume that at most one name or thread is created during a step between two configurations of $\mathcal{P}$. Let us consider the following order on the set of configurations: $P \preceq Q$ iff $P \equiv (\nu \mathbf{x})P'$ and $Q \equiv (\nu \mathbf{x})(P' \mid R)$ for some term R. It is known that this is a well-quasi order (wqo) for the set of all k-depth-bounded configurations [20, 27]. Using this fact, we can show that the set of k-depth-bounded configurations of $\mathcal{P}$, forms a well-structured transition system (WSTS) under the $\preceq$ ordering and then apply the generic backward exploration algorithm for

WSTS [13, 25]. Using the standard and generic complexity arguments for WSTS [26, 13, 25], an upper bound on the the running time of this procedure simply boils down to estimating the length of *controlled bad sequences* of k-depth-bounded configurations under the $\preceq$ order.

Let the size of a configuration C be the number of names and threads that appear in C. Let $H : \mathbb{N} \to \mathbb{N}$ be the successor function and let $n \in \mathbb{N}$. For each $i \in \mathbb{N}$, we let H^i denote the i-fold application of H to itself i times, with H^0 being the identity function.

▶ **Definition 16.** *A sequence $C_0, C_1, \dots,$ of configurations is called (H, n)-controlled bad if the size of each C_i is at most $H^i(n)$ and $C_i \not\preceq C_j$ for any $i < j$.*

To estimate an upper bound on the length of controlled bad sequences of configurations, we first recall the *induced subgraph ordering* on bounded-depth trees.

▶ **Definition 17.** *Let $T_1 = (V_1, E_1, L_1)$ and $T_2 = (V_2, E_2, L_2)$ be two labelled trees with labelling functions $L_1 : V_1 \to A$ and $L_2 : V_2 \to A$ for some finite set A. We say that T_1 is an induced subgraph of T_2, if there is a label preserving injection h from V_1 to V_2 such that $(v, v') \in E_1 \iff (h(v), h(v')) \in E_2$.*

It is known that for any $K \geq 1$ and for any finite set A, the set of all labelled trees of depth at most K is well-quasi ordered under the induced subgraph relation (Theorem 2.2 of [9]). Similar to configurations, we can also define controlled bad sequences of labelled bounded-depth trees.

By the arguments given in [20], it follows that the length of controlled bad sequences of k-depth-bounded configurations of $\mathcal{P}$ under the $\preceq$ order can be upper bounded by the length of controlled bad sequences of K-bounded-depth trees with labels from a set A, for some A and K whose sizes are primitive recursive in the size of $\mathcal{P}$. By the known bounds for controlled bad sequences for labelled bounded-depth trees [2, 17], it follows that

▶ **Theorem 18.** *The length of (H, n)-controlled bad sequences for k-depth-bounded configurations of $\mathcal{P}$ is upper bounded by the function $F_{\epsilon_0}(p(|\mathcal{P}|, k, n))$.*

Here F_{ϵ_0} is the *fast-growing function* at level ϵ_0 and p is some primitive recursive function. For our purposes, we do not need the actual definition of F_{ϵ_0}, but we only need to know that $\mathbf{F}_{\epsilon_0}$ consists of problems whose running time is upper bounded by the function F_{ϵ_0} composed with any primitive recursive function (See [24]). It follows that,

▶ **Theorem 19.** *The coverability problem for depth-bounded processes is in $\mathbf{F}_{\epsilon_0}$ and hence $\mathbf{F}_{\epsilon_0}$-complete.*

6 Conclusion

We have shown that the coverability problem for depth-bounded processes in π-calculus is $\mathbf{F}_{\epsilon_0}$-complete. This settles the complexity of the problem and solves an open problem raised in [17] and also in [27]. However, our proof does not give any results regarding the *parameterized complexity* of this problem when the depth k is taken as a parameter, which we plan to investigate as part of future work.

References

1 Sergio Abriola, Santiago Figueira, and Gabriel Senno. Linearizing well quasi-orders and bounding the length of bad sequences. *Theor. Comput. Sci.*, 603:3–22, 2015. doi:10.1016/j.tcs.2015.07.012.

2 A. R. Balasubramanian. Complexity of coverability in bounded path broadcast networks. In Mikolaj Bojanczyk and Chandra Chekuri, editors, *41st IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2021, December 15-17, 2021, Virtual Conference*, volume 213 of *LIPIcs*, pages 35:1–35:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. `doi:10.4230/LIPIcs.FSTTCS.2021.35`.

3 Michael Blondin. The ABCs of petri net reachability relaxations. *ACM SIGLOG News*, 7(3):29–43, 2020. `doi:10.1145/3436980.3436984`.

4 Michael Blondin, Alain Finkel, Christoph Haase, and Serge Haddad. The logical view on continuous Petri nets. *ACM Trans. Comput. Log.*, 18(3):24:1–24:28, 2017. `doi:10.1145/3105908`.

5 Michael Blondin and Christoph Haase. Logics for continuous reachability in Petri nets and vector addition systems with states. In *32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, June 20-23, 2017*, pages 1–12. IEEE Computer Society, 2017. `doi:10.1109/LICS.2017.8005068`.

6 Pierre Chambart and Philippe Schnoebelen. The ordinal recursive complexity of lossy channel systems. In *Proceedings of the Twenty-Third Annual IEEE Symposium on Logic in Computer Science, LICS 2008, 24-27 June 2008, Pittsburgh, PA, USA*, pages 205–216, 2008. `doi:10.1109/LICS.2008.47`.

7 Wojciech Czerwinski and Lukasz Orlikowski. Reachability in vector addition systems is Ackermann-complete. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 1229–1240. IEEE, 2021. `doi:10.1109/FOCS52979.2021.00120`.

8 Normann Decker and Daniel Thoma. On freeze LTL with ordered attributes. In Bart Jacobs and Christof Löding, editors, *Foundations of Software Science and Computation Structures - 19th International Conference, FOSSACS 2016, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2016, Eindhoven, The Netherlands, April 2-8, 2016, Proceedings*, volume 9634 of *Lecture Notes in Computer Science*, pages 269–284. Springer, 2016. `doi:10.1007/978-3-662-49630-5_16`.

9 Guoli Ding. Subgraphs and well-quasi-ordering. *Journal of Graph Theory*, 16(5):489–502, 1992. `doi:10.1002/jgt.3190160509`.

10 Jacob Elgaard, Nils Klarlund, and Anders Møller. MONA 1.x: New techniques for WS1S and WS2S. In Alan J. Hu and Moshe Y. Vardi, editors, *Computer Aided Verification*, pages 516–520, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

11 Javier Esparza. Keeping a crowd safe: On the complexity of parameterized verification (invited talk). In Ernst W. Mayr and Natacha Portier, editors, *31st International Symposium on Theoretical Aspects of Computer Science (STACS 2014), STACS 2014, March 5-8, 2014, Lyon, France*, volume 25 of *LIPIcs*, pages 1–10. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2014. `doi:10.4230/LIPIcs.STACS.2014.1`.

12 Javier Esparza, Ruslán Ledesma-Garza, Rupak Majumdar, Philipp J. Meyer, and Filip Niksic. An SMT-based approach to coverability analysis. In Armin Biere and Roderick Bloem, editors, *Computer Aided Verification - 26th International Conference, CAV 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 18-22, 2014. Proceedings*, volume 8559 of *Lecture Notes in Computer Science*, pages 603–619. Springer, 2014. `doi:10.1007/978-3-319-08867-9_40`.

13 Diego Figueira, Santiago Figueira, Sylvain Schmitz, and Philippe Schnoebelen. Ackermannian and primitive-recursive bounds with Dickson's lemma. In *Proceedings of the 26th Annual IEEE Symposium on Logic in Computer Science*, pages 269–278, 2011. `doi:10.1109/LICS.2011.39`.

14 Alain Finkel and Philippe Schnoebelen. Well-structured transition systems everywhere! *Theor. Comput. Sci.*, 256(1-2):63–92, 2001. `doi:10.1016/S0304-3975(00)00102-X`.

15 Estíbaliz Fraca and Serge Haddad. Complexity analysis of continuous Petri nets. *Fundam. Informaticae*, 137(1):1–28, 2015. `doi:10.3233/FI-2015-1168`.

16 Christoph Haase and Simon Halfon. Integer vector addition systems with states. In Joël Ouaknine, Igor Potapov, and James Worrell, editors, *Reachability Problems - 8th International Workshop, RP 2014, Oxford, UK, September 22-24, 2014. Proceedings*, volume 8762 of *Lecture Notes in Computer Science*, pages 112–124. Springer, 2014. doi:10.1007/978-3-319-11439-2_9.

17 Christoph Haase, Sylvain Schmitz, and Philippe Schnoebelen. The power of priority channel systems. *Log. Methods Comput. Sci.*, 10(4), 2014. doi:10.2168/LMCS-10(4:4)2014.

18 Slawomir Lasota. Improved Ackermannian lower bound for the petri nets reachability problem. In Petra Berenbrink and Benjamin Monmege, editors, *39th International Symposium on Theoretical Aspects of Computer Science, STACS 2022, March 15-18, 2022, Marseille, France (Virtual Conference)*, volume 219 of *LIPIcs*, pages 46:1–46:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.STACS.2022.46.

19 Jérôme Leroux. The reachability problem for petri nets is not primitive recursive. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 1241–1252. IEEE, 2021. doi:10.1109/FOCS52979.2021.00121.

20 Roland Meyer. On boundedness in depth in the pi-calculus. In Giorgio Ausiello, Juhani Karhumäki, Giancarlo Mauri, and C.-H. Luke Ong, editors, *Fifth IFIP International Conference On Theoretical Computer Science – TCS 2008, IFIP 20th World Computer Congress, TC 1, Foundations of Computer Science, September 7-10, 2008, Milano, Italy*, volume 273 of *IFIP*, pages 477–489. Springer, 2008. doi:10.1007/978-0-387-09680-3_32.

21 Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes, I. *Inf. Comput.*, 100(1):1–40, 1992. doi:10.1016/0890-5401(92)90008-4.

22 Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes, II. *Inf. Comput.*, 100(1):41–77, 1992. doi:10.1016/0890-5401(92)90009-5.

23 Sylvain Schmitz. Complexity bounds for ordinal-based termination - (invited talk). In *Reachability Problems – 8th International Workshop, RP 2014*, pages 1–19, 2014. doi:10.1007/978-3-319-11439-2_1.

24 Sylvain Schmitz. Complexity hierarchies beyond elementary. *ACM Trans. Comput. Theory*, 8(1):3:1–3:36, 2016. doi:10.1145/2858784.

25 Sylvain Schmitz and Philippe Schnoebelen. Multiply-recursive upper bounds with Higman's lemma. In *Automata, Languages and Programming – 38th International Colloquium, ICALP 2011*, pages 441–452, 2011. doi:10.1007/978-3-642-22012-8_35.

26 Sylvain Schmitz and Philippe Schnoebelen. The power of well-structured systems. In Pedro R. D'Argenio and Hernán C. Melgratti, editors, *CONCUR 2013 – Concurrency Theory – 24th International Conference, CONCUR 2013, Buenos Aires, Argentina, August 27-30, 2013. Proceedings*, volume 8052 of *Lecture Notes in Computer Science*, pages 5–24. Springer, 2013. doi:10.1007/978-3-642-40184-8_2.

27 Thomas Wies, Damien Zufferey, and Thomas A. Henzinger. Forward analysis of depth-bounded processes. In C.-H. Luke Ong, editor, *Foundations of Software Science and Computational Structures, 13th International Conference, FOSSACS 2010, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2010, Paphos, Cyprus, March 20-28, 2010. Proceedings*, volume 6014 of *Lecture Notes in Computer Science*, pages 94–108. Springer, 2010. doi:10.1007/978-3-642-12032-9_8.

A Appendix

A.1 Proofs for subsection 4.2

▶ **Lemma 14** ($\mathcal{N}$ simulates $\mathcal{N}'$). *Suppose $C \xrightarrow{*} C'$ is a path of non-zero length in $\mathcal{N}'$ such that 1) C is a good configuration and 2) in all the configurations between C and C', the base of the root is not in Q. Then, C' is a good configuration and there is a rule r such that $\mathbb{M}^{-1}(C) \xrightarrow{r} \mathbb{M}^{-1}(C')$.*

Proof. Let $P := C \rightarrow \gamma_0 \rightarrow \gamma_1 \ldots \rightarrow \gamma_m \rightarrow C'$ be a path in $\mathcal{N}'$. We split the proof into various steps.

Step 1. Since C is a good configuration, the only node which is a leader is the root, whose base must belong to Q. By construction of the rules of $\mathcal{N}'$, this implies that the step $C \rightarrow \gamma_0$ must be of the form $C \xrightarrow{start_0^r} \gamma_0$ for some rule r of $\mathcal{N}$. Let $r = ((q_0, \ldots, q_i), (q_0', \ldots, q_j'))$. This implies that the label of the root in C is $q_0[\top]$ and its label in γ_0 is $\texttt{rec}^r[\top]$.

Step 2. Now, for each $0 \leq l \leq i$, we state two claims:

- Claim A_l: There is a path $P_l := v_l^0, \ldots, v_l^l$ starting at the root in C with labels $q_0[\top], q_1[\bot], \ldots, q_l[\bot]$ such that γ_l is the same as C, except now the labels along P_l are $\underbrace{\texttt{fwd}^r[\bot], \ldots, \texttt{fwd}^r[\bot]}_{l \text{ times}}, \texttt{rec}^r[\top]$.
- Claim B_l: If $l \neq 0$, then $\gamma_{l-1} \xrightarrow{begin_{l-1}^r} \gamma_l$.

We have already shown that claim A_0 is true in step 1. Now, for each $0 \leq l \leq i-1$, assuming claim A_l is true, we shall prove that claims A_{l+1} and B_{l+1} are true.

Because of claim A_l and because C is a good configuration, it follows that the only node which is a leader in γ_l is v_l^l. Further, the base of v_l^l is $\texttt{rec}^r$. By construction of the rules in $\mathcal{N}'$, this implies that the only rule that can be fired from γ_l is $begin_l^r$. Hence, it must be the case that $\gamma_l \xrightarrow{begin_l^r} \gamma_{l+1}$, proving claim B_{l+1}. Further, since v_l^l is the only node which is a leader, firing this rule transforms the state of v_l^l to $\texttt{fwd}^r[\bot]$ and transforms the state of a child of v_l^l (say v') from $q_{l+1}[\bot]$ to $\texttt{rec}^r[\top]$. Taking P_{l+1} to be $v_l^0, \ldots, v_l^l, v'$ proves claim A_{l+1}.

In particular claim A_i implies that there is a path $\texttt{path} := v^0, \ldots, v^i$ starting at the root such that γ_i is the same as C, except that the labels of $\texttt{path}$ in C and γ_i are $q_0[\top], \ldots, q_i[\bot]$ and $\underbrace{\texttt{fwd}^r[\bot], \ldots, \texttt{fwd}^r[\bot]}_{i \text{ times}}, \texttt{rec}^r[\top]$ respectively.

Step 3. For each $i \leq l \leq j$, we state two claims:

- Claim A_l: γ_l is the same as γ_i, except that $\texttt{path}$ is extended to include $l - i$ new nodes and the labels along this extended path in γ_l is $\underbrace{\texttt{fwd}^r[\bot], \ldots, \texttt{fwd}^r[\bot]}_{l \text{ times}}, \texttt{rec}^r[\top]$.
- Claim B_l: If $i \neq l$, then $\gamma_{l-1} \xrightarrow{begin_{l-1}^r} \gamma_l$.

We have already shown that claim A_i is true in step 2. Similar to the arguments given in step 2, we can prove that these new claims are also true.

Step 4. By claim A_j it follows that there is a path $\texttt{ext-path} := \mathsf{n}^0, \ldots, \mathsf{n}^j$ starting at the root in γ_j such that the labels along $\texttt{ext-path}$ is $\underbrace{\texttt{fwd}^r[\bot], \ldots, \texttt{fwd}^r[\bot]}_{j \text{ times}}, \texttt{rec}^r[\top]$. Further, n^j is the only node which is a leader in γ_j. Hence, the only rule which can be fired from γ_j is $middle_j^r$ and so we have $\gamma_j \xrightarrow{middle_j^r} \gamma_{j+1}$. Notice that the only change that has occurred because of this step is that the label of n^j has been changed to $\texttt{fwd}^r[\top]$.

Step 5. For each $1 \leq l \leq j$, we state two claims:

- Claim A'_l: γ_{j+l} is the same as γ_j, except that the labels along `ext-path` in γ_{j+l} is $\underbrace{\texttt{fwd}^r[\bot], \ldots, \texttt{fwd}^r[\bot]}_{j-l+1 \text{ times}}, \texttt{fwd}^r[\top], q'_{j-l+2}[\bot], \ldots, q'_j[\bot]$.
- Claim B'_l: $\gamma_{j+l} \xrightarrow{end^r_{j-l}} \gamma_{j+l+1}$.

The proof of this is accomplished by similar arguments as given in step 2.

Step 6. By claim A'_j, it follows that γ_{2j} is the same as γ_j, except that the labels along `ext-path` is now $\texttt{fwd}^r[\top], q'_1[\bot], \ldots, q'_j[\bot]$. It follows that the only rule which can be fired from γ_{2j} is $finish^r_0$, and so it follows that $\gamma_{2j} \xrightarrow{finish^r_0} \gamma_{2j+1}$, where the only difference between γ_{2j+1} and γ_{2j} is that the label of the root in γ_{2j+1} is $q'_0[\top]$. Hence, by assumption of the run P, it follows that $\gamma_{2j+1} = C'$.

By combining the arguments given above, it follows then that γ_{2j+1} is a good configuration and also that $\mathbb{M}^{-1}(C) \xrightarrow{r} \mathbb{M}^{-1}(C')$. ◀

▶ **Theorem 15.** *q_{in} can cover q_f in $\mathcal{N}$ iff $q_{in}[\top]$ can cover $q_f[\top]$ in $\mathcal{N}'$.*

Proof. Suppose q_{in} can cover q_f in $\mathcal{N}$. Let $C_0 \to C_1 \to \ldots \to C_m$ be a run in $\mathcal{N}$ where C_0 is the initial configuration and the root of C_m is q_f. By Lemma 13, it follows that $\mathbb{M}(C_0) \xrightarrow{*} \mathbb{M}(C_1) \xrightarrow{*} \ldots \xrightarrow{*} \mathbb{M}(C_m)$ and so $q_{in}[\top]$ can cover $q_f[\top]$ in $\mathcal{N}'$.

Suppose $C \xrightarrow{*} C'$ is a run in $\mathcal{N}'$ such that C is the (unique good) configuration consisting of the single root vertex labelled by $q_{in}[\top]$ and C' is some configuration where the root is labelled by $q_f[\top]$. We split the run into parts of the form $C = C_0 \xrightarrow{*} C_1 \xrightarrow{*} C_2 \ldots \xrightarrow{*} C_m = C'$ such that for each $1 \leq l \leq m$, C_l is the first configuration after C_{l-1} where the base of the root is in Q. By Lemma 14, it follows that each C_l is a good configuration and also that $\mathbb{M}^{-1}(C_0) \to \mathbb{M}^{-1}(C_1) \to \ldots \to \mathbb{M}^{-1}(C_m)$. Hence, it follows that q_{in} can cover q_f in $\mathcal{N}$. ◀

A.2 Proofs for Section 5

We now give a proof of Theorem 19. We recall the backward exploration algorithm for well-structured transition systems (WSTS) here, adapted to the coverability problem for depth-bounded processes. Let $\mathcal{P} = (I, \mathcal{E})$ be a k-depth-bounded process and let P be some k-depth-bounded configuration, which we want to check is coverable in $\mathcal{P}$. Without loss of generality, we can assume that at most one name or thread is created during a step between two configurations of $\mathcal{P}$. Let $\mathcal{C}_k$ be the set of all k-depth-bounded configurations.

Given a set S of $\mathcal{C}_k$ we let $\uparrow S := \{\gamma' : \exists \gamma \in S, \gamma \preceq \gamma'\}$. A set S is called *upward-closed* if $S = \uparrow S$. Because $\preceq$ is a wqo and because of the definition of the operational semantics of $\mathcal{P}$, we have that:

- If S is upward-closed, then there exists a finite set B such that $\uparrow B = S$. Such a B will be called the basis of S.
- If S is upward-closed and if $Pre(S)$ is the set of all configurations $\gamma' \in \mathcal{C}_k$ such that there is a configuration $\gamma \in S$ with $\gamma' \to \gamma$, then $S \cup Pre(S)$ is upward-closed. Moreover, given a basis B of S, we can compute a basis B' of $S \cup Pre(S)$ such that the size of each configuration in B' is at most one more than the maximum size of any configuration of B.

Hence, by the generic backward exploration algorithm for WSTS [14], we get that the following algorithm terminates and decides coverability: Construct a sequence of finite sets $B_0, B_1, \ldots$, such that each $B_i \subseteq \mathcal{C}_k$, B_0 is simply $\{P\}$ and B_{i+1} is a basis for $\uparrow B_i \cup Pre(\uparrow B_i)$.

Then find the first m such that $\uparrow B_m = \uparrow B_{m+1}$ and check if there is an initial configuration in $\uparrow B_m$. If it is true, then P is coverable; otherwise P is not coverable.

The running time complexity of the algorithm is mainly dominated by the length of the sequence $B_0, B_1, \ldots, B_m$. Since m is the first index such that $\uparrow B_m = \uparrow B_{m+1}$, we can find a minimal element $\gamma_i \in \uparrow B_{i+1} \setminus \uparrow B_i$ for each $i < m$.

Consider the sequence $\gamma_0, \ldots, \gamma_{m-1}$. Notice that $\gamma_i \not\preceq \gamma_j$ for any $j > i$ and further the size of each γ_i is at most $H^i(n)$, where H is the successor function and n is the size of P. It follows that $\gamma_0, \ldots, \gamma_{m-1}$ is a (H, n)-controlled bad sequence. By the arguments given in [20], it follows that the length of controlled bad sequences of $\mathcal{C}_k$ under the $\preceq$ order can be upper bounded by the length of controlled bad sequences of K-bounded-depth trees with labels from a set A, for some A and K whose sizes are primitive recursive in the size of $\mathcal{P}$. By the known bounds for controlled bad sequences for labelled bounded-depth trees [2, 17], it follows that

▶ **Theorem 18.** *The length of (H, n)-controlled bad sequences for k-depth-bounded configurations of $\mathcal{P}$ is upper bounded by the function $F_{\epsilon_0}(p(|\mathcal{P}|, k, n))$.*

Here F_{ϵ_0} is the *fast-growing function* at level ϵ_0 and p is some primitive recursive function. For our purposes, we do not need the actual definition of F_{ϵ_0}, but we only need to know that $\mathbf{F}_{\epsilon_0}$ consists of problems whose running time is upper bounded by the function F_{ϵ_0} composed with any primitive recursive function (See [24]). Theorem 19 then follows.

Appendix D

Finding Cut-Offs in Leaderless Rendez-Vous Protocols is Easy (FoSSaCS 2021)

This section contains a reprinting of the following paper, which has been published as a peer-reviewed conference paper.

> A. R. Balasubramanian and Javier Esparza and Mikhail A. Raskin. *Finding Cut-Offs in Leaderless Rendez-Vous Protocols is Easy.* In conference proceedings of FoSSaCS 2021. Springer, 2021. Vol. 12650 of Lecture Notes in Computer Science, Pages - 42-61. doi: `10.1007/978-3-030-71995-1_3`

According to the Open Access policy of Lecture Notes in Computer Science of Springer, the author of this thesis is permitted to include the above paper in this thesis. The relevant excerpt is the following:

Summary

In rendez-vous protocols, an arbitrarily large number of indistinguishable finite-state agents interact in pairs. The cut-off problem asks if a number B exists so that all initial configurations of the protocol with at least B agents in a given initial state can reach a final configuration with all agents in a given final state. Previous work has shown that in the presence of a leader, the cut-off problem is decidable and at least as hard as the Petri net reachability problem, which is non-primitive recursive. In the absence of a leader, it is known that the cut-off problem is in $\mathsf{EXPSPACE}$. Further, for the special class of symmetric rendez-vous protocols, the cut-off problem is in PSPACE when a leader is present and is in NP otherwise. In this work, we improve upon some of these upper bounds and show that the cut-off problem is P-complete for leaderless protocols, NP-complete for symmetric protocols with a leader, and in NC for leaderless symmetric protocols.

Contributions of the author of this thesis

Contribution of Balasubramanian Ayikudi Ramachandrakumar	
Scientific findings	30%
Development and conceptual design of the research project	80%
Discussion and development of ideas	70%
Drafting of the manuscript	60%

Finding Cut-Offs in Leaderless Rendez-Vous Protocols is Easy [⋆]

A. R. Balasubramanian(✉)[1], Javier Esparza[1], Mikhail Raskin[1]

Technische Universität München, Munich, Germany
bala.ayikudi@tum.de, esparza@in.tum.de, raskin@in.tum.de

Abstract. In rendez-vous protocols an arbitrarily large number of indistinguishable finite-state agents interact in pairs. The cut-off problem asks if there exists a number B such that all initial configurations of the protocol with at least B agents in a given initial state can reach a final configuration with all agents in a given final state. In a recent paper [17], Horn and Sangnier prove that the cut-off problem is equivalent to the Petri net reachability problem for protocols with a leader, and in EXPSPACE for leaderless protocols. Further, for the special class of symmetric protocols they reduce these bounds to PSPACE and NP, respectively. The problem of lowering these upper bounds or finding matching lower bounds is left open. We show that the cut-off problem is P-complete for leaderless protocols, NP-complete for symmetric protocols with a leader, and in NC for leaderless symmetric protocols, thereby solving all the problems left open in [17].

Keywords: rendez-vous protocols · cut-off problem · Petri nets

1 Introduction

Distributed systems are often designed for an unbounded number of participant agents. Therefore, they are not just one system, but an infinite family of systems, one for each number of agents. Parameterized verification addresses the problem of checking that all systems in the family satisfy a given specification.

In many application areas, agents are indistinguishable. This is the case in computational biology, where cells or molecules have no identities; in some security applications, where the agents' identities should stay private; or in applications where the identities can be abstracted away, like certain classes of multithreaded programs [15,2,31,3,18,25]. Following [3,18], we use the term *replicated systems* for distributed systems with indistinguishable agents. Replicated systems include population protocols, broadcast protocols, threshold automata, and many other models [15,2,11,7,16]. They also arise after applying a *counter abstraction* [28,3]. In finite-state replicated systems the global state of the system is determined by the function (usually called a *configuration*) that assigns

[⋆] This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme under grant agreement No 787367 (PaVeS).

S. Kiefer and C. Tasson (Eds.): FOSSACS 2021, LNCS 12650, pp. 42–61, 2021.
https://doi.org/10.1007/978-3-030-71995-1_3

to each state the number of agents that currently occupy it. This feature makes many verification problems decidable [4,10].

Surprisingly, there is no a priori relation between the complexity of a parameterized verification question (i.e., whether a given property holds for all initial configurations, or, equivalently, whether its negation holds for some configuration), and the complexity of its corresponding single-instance question (whether the property holds for a fixed initial configuration). Consider replicated systems where agents interact in pairs [15,17,2]. The complexity of single-instance questions is very robust. Indeed, checking most properties, including all properties expressible in LTL and CTL, is PSPACE-complete [9]. On the contrary, the complexity of parameterized questions is very fragile, as exemplified by the following example. While the existence of a reachable configuration that populates a given state with *at least* one agent is in P, and so well below PSPACE, the existence of a reachable configuration that populates a given state with *exactly* one agent is as hard as the reachability problem for Petri nets, and so non-elementary [6]. This fragility makes the analysis of parameterized questions very interesting, but also much harder.

Work on parameterized verification has concentrated on whether every initial configuration satisfies a given property (see e.g. [15,11,3,18,7]). However, applications often lead to questions of the form "do all initial configurations *in a given set* satisfy the property?", "do infinitely many initial configurations satisfy the property?", or "do all but finitely many initial configurations satisfy the property?". An example of the first kind is proving correctness of population protocols, where the specification requires that for a given partition $\mathcal{I}_0$, $\mathcal{I}_1$ of the set of initial configurations, and a partition Q_0, Q_1 of the set of states, runs starting from $\mathcal{I}_0$ eventually trap all agents within Q_0, and similarly for $\mathcal{I}_1$ and Q_1 [12]. An example of the third kind is the existence of *cut-offs*; cut-off properties state the existence of an initial configuration such that for all larger initial configurations some given property holds [8,4]. A systematic study of the complexity of these questions is still out of reach, but first results are appearing. In particular, Horn and Sangnier have recently studied the complexity of the *cut-off problem* for parameterized rendez-vous networks [17]. The problem takes as input a network with one single initial state *init* and one single final state *fin*, and asks whether there exists a cut-off B such that for every number of agents $n \geq B$, the final configuration in which all agents are in state *fin* is reachable from the initial configuration in which all agents are in state *init*.

Horn and Sangnier study two versions of the cut-off problem, for leaderless networks and networks with a leader. Intuitively, a leader is a distinguished agent with its own set of states. They show that in the presence of a leader the cut-off problem and the reachability problem for Petri nets problems are inter-reducible, which shows that the cut-off problem is in the Ackermannian complexity class $\mathcal{F}_\omega$ [22], and non-elementary [6]. For the leaderless case, they show that the problem is in EXPSPACE. Further, they also consider the special case of symmetric networks, for which they obtain better upper bounds: PSPACE for the case of a

Horn and Sangnier	Asymmetric rendez-vous	Symmetric rendez-vous
Presence of a leader	Decidable, non-elementary	PSPACE
Absence of a leader	EXPSPACE	NP

This paper	Asymmetric rendez-vous	Symmetric rendez-vous
Presence of a leader	Decidable, non-elementary	NP-complete
Absence of a leader	P-complete	NC

Table 1. Summary of the results by Horn and Sangnier and the results of this paper.

leader, and NP in the leaderless case. These results are summarized at the top of Table 1.

In [17] the question of improving the upper bounds or finding matching lower bounds is left open. In this paper we close it with a surprising answer: All elementary upper bounds of [17] can be dramatically improved. In particular, our main result shows that the EXPSPACE bound for the leaderless case can be brought down to P. Further, the PSPACE and NP bounds of the symmetric case can be lowered to NP and NC, respectively, as shown at the bottom of Table 1. We also obtain matching lower bounds. Finally, we provide almost tight upper bounds for the size of the cut-off B; more precisely, we show that if B exists, then $B \in 2^{n^{O(1)}}$ for a protocol of size n.

Our results follow from two lemmas, called the Scaling and Insertion Lemmas, that connect the *continuous semantics* for Petri nets to their standard semantics. In the continuous semantics of Petri nets transition firings can be scaled by a positive rational factor; for example, a transition can fire with factor 1/3, taking "1/3 of a token" from its input places. The continuous semantics is a relaxation of the standard one, and its associated reachability problem is much simpler (polynomial instead of non-elementary [14,6,5]). The Scaling Lemma[1] states that given two markings M, M' of a Petri net, if M' is reachable from M in the continuous semantics, then nM' is reachable from nM in the standard semantics for some $n \in 2^{m^{O(1)}}$, where m is the total size of the net and the markings. The Insertion Lemma states that, given four markings M, M', L, L', if M' is reachable from M in the continuous semantics and the *marking equation* $L' = L + \mathcal{A}\mathbf{x}$ has a solution $\mathbf{x} \in \mathbb{Z}^T$ (observe that $\mathbf{x}$ can have negative components), then $nM' + L'$ is reachable from $nM + L$ in the standard semantics for some $n \in 2^{m^{O(1)}}$. We think that these lemmas can be of independent interest.

The paper is organized as follows. Section 2 contains preliminaries; in particular, it defines the cut-off problem for rendez-vous networks and reduces it to the cut-off problem for Petri nets. Section 3 gives a polynomial time algorithm for the leaderless cut-off problem for acyclic Petri nets. Section 4 introduces the Scaling and Insertion Lemmas, and Section 5 presents the novel polynomial

[1] Heavily based on previous results by Fraca and Haddad [14].

time algorithm for the cut-off problem. Sections 6 and 7 present the results for symmetric networks, for the cases with and without leaders, respectively.

Due to lack of space, full proofs of some of the lemmas can be found in the appendix.

2 Preliminaries

Multisets Let E be a finite set. For a semi-ring S, a vector from E to S is a function $v : E \to S$. The set of all vectors from E to S will be denoted by S^E. In this paper, the semi-rings we will be concerned with are the natural numbers $\mathbb{N}$, the integers $\mathbb{Z}$ and the non-negative rationals $\mathbb{Q}_{\geq 0}$ (under the usual addition and multiplication operators). The *support* of a vector v is the set $[\![v]\!] := \{e : v(e) \neq 0\}$ and its *size* is the number $\|v\| = \sum_{e \in [\![v]\!]} abs(v(e))$ where $abs(x)$ denotes the absolute value of x. Vectors from E to $\mathbb{N}$ are also called discrete multisets (or just multisets) and vectors from E to $\mathbb{Q}_{\geq 0}$ are called continuous multisets.

Given a multiset M and a number α we let $\alpha \cdot M$ be the multiset given by $(\alpha \cdot M)(e) = M(e) \cdot \alpha$ for all $e \in E$. Given two multisets M and M' we say that $M \leq M'$ if $M(e) \leq M'(e)$ for all $e \in E$ and we let $M + M'$ be the multiset given by $(M + M')(e) = M(e) + M'(e)$ and if $M' \leq M$, we let $M - M'$ be the multiset given by $(M - M')(e) = M(e) - M'(e)$. The empty multiset is denoted by $\mathbf{0}$. We sometimes denote multisets using a set-like notation, e.g. $\wr a, 2 \cdot b, c \int$ denotes the multiset given by $M(a) = 1, M(b) = 2, M(c) = 1$ and $M(e) = 0$ for all $e \notin \{a, b, c\}$.

Given an $I \times J$ matrix A with I and J sets of indices, $I' \subseteq I$ and $J' \subseteq J$, we let $A_{I' \times J'}$ denote the restriction of M to rows indexed by I' and columns indexed by J'.

Rendez-vous protocols and the cut-off problem. Let Σ be a fixed finite set which we will call the communication alphabet and we let $RV(\Sigma) = \{!a, ?a : a \in \Sigma\}$. The symbol $!a$ denotes that the message a is sent and $?a$ denotes that the message a is received.

Definition 1. *A rendez-vous protocol $\mathcal{P}$ is a tuple $(Q, \Sigma, init, fin, R)$ where Q is a finite set of states, Σ is the communication alphabet, $init, fin \in Q$ are the initial and final states respectively and $R \subseteq Q \times RV(\Sigma) \times Q$ is the set of rules.*

The size $|\mathcal{P}|$ of a protocol is defined as the number of bits needed to encode $\mathcal{P}$ in $\{0,1\}^*$ using some standard encoding. A configuration C of $\mathcal{P}$ is a multiset of states, where $C(q)$ should be interpreted as the number of agents in state q. We use $\mathcal{C}(\mathcal{P})$ to denote the set of all configurations of $\mathcal{P}$. An initial (final) configuration C is a configuration such that $C(q) = 0$ if $q \neq init$ (resp. $C(q) = 0$ if $q \neq fin$). We use C^n_{init} (C^n_{fin}) to denote the initial (resp. final) configuration such that $C^n_{init}(init) = n$ (resp. $C^n_{fin}(fin) = n$).

The operational semantics of a rendez-vous protocol $\mathcal{P}$ is given by means of a transition system between the configurations of $\mathcal{P}$. We say that there is a transition between C and C', denoted by $C \Rightarrow C'$ iff there exists $a \in \Sigma$, $p, q, p', q' \in Q$ such that $(p, !a, p'), (q, ?a, q') \in R$, $C \geq \wr p, q \wr$ and $C' = C - \wr p, q \wr + \wr p', q' \wr$. As usual, $\overset{*}{\Rightarrow}$ denotes the reflexive and transitive closure of $\Rightarrow$.

The cut-off problem for rendez-vous protocols, as defined in [17], is:

Given: A rendez-vous protocol $\mathcal{P}$
Decide: Is there $B \in \mathbb{N}$ such that $C^n_{init} \overset{*}{\Rightarrow} C^n_{fin}$ for every $n \geq B$?

If such a B exists then we say that $\mathcal{P}$ admits a cut-off and that B is a cut-off for $\mathcal{P}$.

Petri nets. Rendez-vous protocols can be seen as a special class of Petri nets.

Definition 2. *A Petri net is a tuple $\mathcal{N} = (P, T, Pre, Post)$ where P is a finite set of places, T is a finite set of transitions, Pre and Post are matrices whose rows and columns are indexed by P and T respectively and whose entries belong to $\mathbb{N}$. The incidence matrix $\mathcal{A}$ of $\mathcal{N}$ is defined to be the $P \times T$ matrix given by $\mathcal{A} = Post - Pre$. Further by the* weight *of $\mathcal{N}$, we mean the largest absolute value appearing in the matrices Pre and Post.*

The size $|\mathcal{N}|$ of $\mathcal{N}$ is defined as the number of bits needed to encode $\mathcal{N}$ in $\{0,1\}^*$ using some suitable encoding. For a transition $t \in T$ we let ${}^\bullet t = \{p : Pre[p,t] > 0\}$ and $t^\bullet = \{p : Post[p,t] > 0\}$. We extend this notation to set of transitions in the obvious way. Given a Petri net $\mathcal{N}$, we can associate with it a graph where the vertices are $P \cup T$ and the edges are $\{(p,t) : p \in {}^\bullet t\} \cup \{(t,p) : p \in t^\bullet\}$. A Petri net $\mathcal{N}$ is called acyclic if its associated graph is acyclic.

A *marking* of a Petri net is a multiset $M \in \mathbb{N}^P$, which intuitively denotes the number of *tokens* that are present in every place of the net. For $t \in T$ and markings M and M', we say that M' is reached from M by firing t, denoted $M \xrightarrow{t} M'$, if for every place p, $M(p) \geq Pre[p,t]$ and $M'(p) = M(p) + \mathcal{A}[p,t]$.

A *firing sequence* is any sequence of transitions $\sigma = t_1, t_2, \ldots, t_k \in T^*$. The support of σ, denoted by $[\![\sigma]\!]$, is the set of all transitions which appear in σ. We let $\sigma\sigma'$ denote the concatenation of two sequences σ, σ'.

Given a firing sequence $\sigma = t_1, t_2, \ldots, t_k \in T^*$, we let $M \xrightarrow{\sigma} M'$ denote that there exist $M_1, \ldots, M_{k-1}$ such that $M \xrightarrow{t_1} M_1 \xrightarrow{t_2} M_2 \ldots M_{k-1} \xrightarrow{t_k} M'$. Further, $M \rightarrow M'$ denotes that there exists $t \in T$ such that $M \xrightarrow{t} M'$, and $M \xrightarrow{*} M'$ denotes that there exists $\sigma \in T^*$ such that $M \xrightarrow{\sigma} M'$.

Marking equation of a Petri net system. In the following, a *Petri net system* is a triple $(\mathcal{N}, M, M')$ where $\mathcal{N}$ is a Petri net and $M \neq M'$ are markings. The *marking equation* for $(\mathcal{N}, M, M')$ is the equation

$$M' = M + \mathcal{A}\mathbf{v}$$

over the variables $\mathbf{v}$. It is well known that $M \xrightarrow{\sigma} M'$ implies $M' = M + \mathcal{A}\overrightarrow{\sigma}$, where $\overrightarrow{\sigma} \in \mathbb{N}^T$ is the the *Parikh image* of σ, defined as the vector whose component $\overrightarrow{\sigma}[t]$ for transition t is equal to the number of times t appears in σ. Therefore, if $M \xrightarrow{\sigma} M'$ then $\overrightarrow{\sigma}$ is a nonnegative integer solution of the marking equation. The converse does not hold.

From rendez-vous protocols to Petri nets. Let $\mathcal{P} = (Q, \Sigma, \mathit{init}, \mathit{fin}, R)$ be a rendez-vous protocol. Create a Petri net $\mathcal{N}_\mathcal{P} = (P, T, \mathit{Pre}, \mathit{Post})$ as follows. The set of places is Q. For each letter $a \in \Sigma$ and for each pair of rules $r = (q, !a, s), r' = (q', ?a, s') \in R$, add a transition $t_{r,r'}$ to $\mathcal{N}_\mathcal{P}$ and set

- $\mathit{Pre}[p, t] = 0$ for every $p \notin \{q, q'\}$, $\mathit{Post}[p, t] = 0$ for every $p \notin \{s, s'\}$
- If $q = q'$ then $\mathit{Pre}[q, t] = -2$, otherwise $\mathit{Pre}[q, t] = \mathit{Pre}[q', t] = -1$
- If $s = s'$ then $\mathit{Post}[s, t] = 2$, otherwise $\mathit{Post}[s, t] = \mathit{Post}[s', t] = 1$.

It is clear that any configuration of a protocol $\mathcal{P}$ is also a marking of $\mathcal{N}_\mathcal{P}$, and vice versa. Further, the following proposition is obvious.

Proposition 1. *For any two configurations C and C' we have that $C \overset{*}{\Rightarrow} C'$ over the protocol $\mathcal{P}$ iff $C \xrightarrow{*} C'$ over the Petri net $\mathcal{N}_\mathcal{P}$.*

Consequently, the cut-off problem for Petri nets, defined by

Given : A Petri net system $(\mathcal{N}, M, M')$
Decide: Is there $B \in \mathbb{N}$ such that $n \cdot M \xrightarrow{*} n \cdot M'$ for every $n \geq B$?

generalizes the problem for rendez-vous protocols.

3 The cut-off problem for acyclic Petri nets

We show that the cut-off problem for acyclic Petri nets can be solved in polynomial time. The reason for considering this special case first is that it illustrates one of the main ideas of the general case in a very pure form.

Let us fix a Petri net system $(\mathcal{N}, M, M')$ for the rest of this section, where $\mathcal{N} = (P, T, \mathit{Pre}, \mathit{Post})$ is acyclic and $\mathcal{A}$ is its incidence matrix. It is well-known that in acyclic Petri nets the reachability relation is characterized by the marking equation (see e.g. [24]):

Proposition 2 ([24]). *Let $(\mathcal{N}, M, M')$ be an acyclic Petri net system. For every sequence $\sigma \in T^*$, we have $M \xrightarrow{\sigma} M'$ iff $\overrightarrow{\sigma}$ is a solution of the marking equation. Consequently, $M \xrightarrow{*} M'$ iff the marking equation has a nonnegative integer solution.*

This proposition shows that the reachability problem for acyclic Petri nets reduces to the feasibilty problem (i.e., existence of solutions) of systems of linear diophantine equations over the nonnegative integers. So the reachability problem for acyclic Petri nets is in NP, and in fact both the reachability and the feasibility problems are NP-complete [13].

There are two ways to relax the conditions on the solution so as to make the feasibility problem polynomial. Feasibility over the nonnegative *rationals* and feasibility over all integers are both in P. The first is due to the polynomiality of linear programming. For the second, feasibility can be decided in polynomial time after computing the Smith or Hermite normal forms (see e.g. [29]), which can themselves be computed in polynomial time [19]. We show that the cut-off problem can be reduced to these two relaxed problems.

3.1 Characterizing acyclic systems with cut-offs

Horn and Sangnier proved in [17] a very useful charaterization of the rendez-vous protocols with a cut-off: A rendez-vous protocol $\mathcal{P}$ admits a cut-off iff there exists $n \in \mathbb{N}$ such that $C^n_{init} \overset{*}{\Rightarrow} C^n_{fin}$ and $C^{n+1}_{init} \overset{*}{\Rightarrow} C^{n+1}_{fin}$. The proof immediately generalizes to the case of Petri nets:

Lemma 1 ([17]). *A Petri net system $(\mathcal{N}, M, M')$ (acyclic or not) admits a cut-off iff there exists $n \in \mathbb{N}$ such that $n \cdot M \xrightarrow{*} n \cdot M'$ and $(n+1) \cdot M \xrightarrow{*} (n+1) \cdot M'$. Moreover if $n \cdot M \xrightarrow{*} n \cdot M'$ and $(n+1) \cdot M \xrightarrow{*} (n+1) \cdot M'$, then n^2 is a cut-off for the system.*

Using this lemma, we characterize those acyclic Petri net systems which admit a cut-off.

Theorem 1. *An acyclic Petri net system $(\mathcal{N}, M, M')$ admits a cut-off iff the marking equation has solutions $\mathbf{x} \in \mathbb{Q}^T_{\geq 0}$ and $\mathbf{y} \in \mathbb{Z}^T$ such that $[\![\mathbf{y}]\!] \subseteq [\![\mathbf{x}]\!]$.*

Proof. ($\Rightarrow$): Suppose $(\mathcal{N}, M, M')$ admits a cut-off. Hence there exists $b \in \mathbb{N}$ such that for all $n \geq b$ we have $nM \xrightarrow{*} nM'$. Let $bM \xrightarrow{\sigma'} bM'$ and $(b+1)M \xrightarrow{\tau'} (b+1)M'$. Then, notice that $(2b+1)M \xrightarrow{\sigma'\tau'} (2b+1)M'$ and $(2b+2)M \xrightarrow{\tau'\tau'} (2b+2)M'$. Hence, if we let $n = 2b+1$, $\sigma = \sigma'\tau'$ and $\tau = \tau'\tau'$ we have, $nM \xrightarrow{\sigma} nM'$, $(n+1)M \xrightarrow{\tau} (n+1)M'$ and $[\![\tau]\!] \subseteq [\![\sigma]\!]$. By Proposition 2, there exist $\mathbf{x}', \mathbf{y}' \in \mathbb{N}^T$ such that $[\![\mathbf{y}']\!] \subseteq [\![\mathbf{x}']\!]$, $nM' = nM + \mathcal{A}\mathbf{x}'$ and $(n+1)M' = (n+1)M + \mathcal{A}\mathbf{y}'$. Letting $\mathbf{x} = \mathbf{x}'/n$ and $\mathbf{y} = \mathbf{y}' - \mathbf{x}'$, we get our required vectors.

($\Leftarrow$): Suppose $\mathbf{x} \in \mathbb{Q}^T_{\geq 0}$ and $\mathbf{y} \in \mathbb{Z}^T$ are solutions of the marking equation such that $[\![\mathbf{y}]\!] \subseteq [\![\mathbf{x}]\!]$. Let μ be the least common multiple of the denominators of the components of $\mathbf{x}$, and let α be the largest absolute value of the numbers in the vector $\mathbf{y}$. By definition of μ we have $\alpha(\mu\mathbf{x}) \in \mathbb{N}^T$. Also, since $[\![\mathbf{y}]\!] \subseteq [\![\mathbf{x}]\!]$ it follows by definition of α that $\alpha(\mu\mathbf{x}) + \mathbf{y} \geq \mathbf{0}$ and hence $\alpha(\mu\mathbf{x}) + \mathbf{y} \in \mathbb{N}^T$. Since $M' = M + \mathcal{A}\mathbf{x}$ and $M' = M + \mathcal{A}\mathbf{y}$ we get

$$\alpha\mu M' = \alpha\mu M + \mathcal{A}(\alpha\mu\mathbf{x}) \qquad \text{and} \qquad (\alpha\mu+1)M' = (\alpha\mu+1)M + \mathcal{A}(\alpha\mu\mathbf{x} + \mathbf{y})$$

Taking $\alpha\mu = n$, by Proposition 2 we get that $nM \xrightarrow{*} nM'$ and $(n+1)M \xrightarrow{*} (n+1)M'$. By Lemma 1, $(\mathcal{N}, M, M')$ admits a cut-off.

Intuitively, the existence of the rational solution $\mathbf{x} \in \mathbb{Q}^T_{\geq 0}$ guarantees $nM \xrightarrow{*} nM'$ for infinitely many n, and the existence of the integer solution $\mathbf{y} \in \mathbb{Z}^T$ guarantees that for one of those n we have $(n+1)M \xrightarrow{*} (n+1)M'$ as well.

Example 1. The net system given by the net on Figure 1 along with the markings $M = \wr i \int$ and $M' = \wr f \int$ admits a cut-off. The conditions of the theorem are satisfied by $\mathbf{x} = (\frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5})$ and $\mathbf{y} = (-1, 1, 1, 1)$.

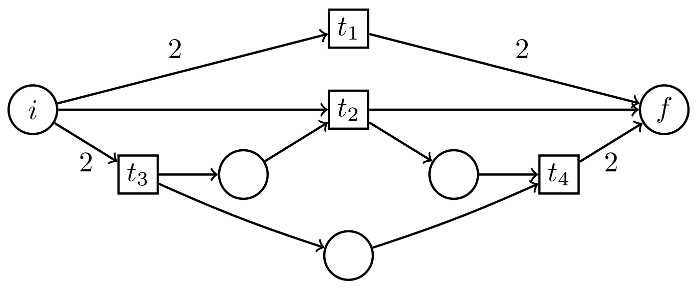

Fig. 1. A net with cut-off 2.

3.2 Polynomial time algorithm

We derive a polynomial time algorithm for the cut-off problem from the characterization of Theorem 1. The first step is the following lemma. A very similar lemma is proved in [14], but since the proof is short we give it for the sake of completeness:

Lemma 2. *If the marking equation is feasible over $\mathbb{Q}_{\geq 0}$, then it has a solution with maximum support. Moreover, such a solution can be found in polynomial time.*

Proof. If $\mathbf{y}, \mathbf{z} \in \mathbb{Q}^T_{\geq 0}$ are solutions of the marking equation, then we have $M' = M + \mathcal{A}((\mathbf{y} + \mathbf{z})/2)$ and $[\![\mathbf{y}]\!] \cup [\![\mathbf{z}]\!] \subseteq [\![(\mathbf{y} + \mathbf{z})/2]\!]$. Hence if the marking equation if feasible over $\mathbb{Q}_{\geq 0}$, then it has a solution with maximum support.

To find such a solution in polynomial time we proceed as follows. For every transition t we solve the linear program $M' = M + \mathcal{A}\mathbf{v}, \mathbf{v} \geq \mathbf{0}, \mathbf{v}(t) > 0$. (Recall that solving linear programs over the rationals can be done in polynomial time). Let $\{t_1, \ldots, t_n\}$ be the set of transitions whose associated linear programs are feasible over $\mathbb{Q}^T_{\geq 0}$, and let $\{\mathbf{u}_1, \ldots, \mathbf{u}_n\}$ be solutions to these programs. Then $1/n \cdot \sum_{i=1}^n \mathbf{u}_i$ is a solution of the marking equation with maximum support.

We now have all the ingredients to give a polynomial time algorithm.

Theorem 2. *The cut-off problem for acyclic net systems can be solved in polynomial time.*

Proof. First, we check that the marking equation has a solution over the non-negative rationals. If such a solution does not exist, by Theorem 1 the given net system does not admit a cut-off.

Suppose such a solution exists. By Lemma 2 we can find a non-negative rational solution $\mathbf{x}$ with maximum support in polynomial time. Let U contain all the transitions t such that $\mathbf{x}_t = 0$. We now check in polynomial time if the marking equation has a solution $\mathbf{y}$ over $\mathbb{Z}^T$ such that $\mathbf{y}_t = 0$ for every $t \in U$. By Theorem 1 such a solution exists iff the net system admits a cut-off.

The rendez-vous protocol given in Figure 2, which was stated in [17], is an example of a protocol where the smallest cut-off is exponential in the size of the protocol. In the next sections, we will actually prove that if a net system $\mathcal{N}$ (acyclic or not) admits a cut-off, then there is one with a polynomial number of bits in $|\mathcal{N}|$.

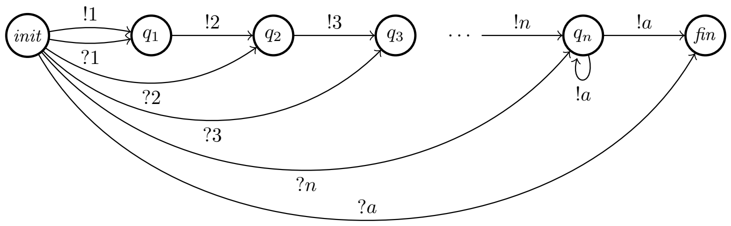

Fig. 2. Example of a protocol with an exponential cut-off

4 The Scaling and Insertion lemmas

Similar to the case of acyclic net systems, we would like to provide a characterization of net systems admitting a cut-off and then use this characterization to derive a polynomial time algorithm. Unfortunately, in general net systems there is no characterization of reachability akin to Proposition 2 for acyclic systems. To this end, we prove two intermediate lemmas to help us come up with a characterization for cut-off admissible net systems in the general case. We believe that these two lemmas could be of independent interest in their own right. Further, the proofs of both lemmas are provided so that it will enable us later on to derive a bound on the cut-off for net systems.

4.1 The Scaling Lemma

The Scaling Lemma shows that, given a Petri net system $(\mathcal{N}, M, M')$, whether $nM \xrightarrow{*} nM'$ holds for some $n \geq 1$ can be decided in polynomial time; moreover, if $nM \xrightarrow{*} nM'$ holds for some n, then it holds for some n with at most $(|\mathcal{N}|(\log \|M\| + \log \|M'\|))^{O(1)}$ bits. The name of the lemma is due to the fact that the firing sequence leading from nM to nM' is obtained by *scaling up* a *continuous firing sequence* from M to M'; the existence of such a continuous sequence can be decided in polynomial time [14].

In the rest of the section we first recall continuous Petri nets and the characterization of [14], and then present the Scaling Lemma[2].

[2] The lemma is implicitly proved in [14], but the bound on the size of n is hidden in the details of the proof, and we make it explicit.

Reachability in continuous Petri nets. Petri nets can be given a *continuous semantics* (see e.g. [1,30,14]), in which markings are continuous multisets; we call them *continuous markings*. A continuous marking M enables a transition t *with factor* $\lambda \in \mathbb{Q}_{\geq 0}$ if $M(p) \geq \lambda \cdot Pre[p,t]$ for every place p; we also say that M enables λt. If M enables λt, then λt can fire or occur, leading to a new marking M' given by $M'(p) = M(p) + \lambda \cdot \mathcal{A}[p,t]$ for every $p \in P$. We denote this by $M \xrightarrow[\mathbb{Q}]{\lambda t} M'$, and say that M' is reached from M by firing λt. A *continuous firing sequence* is any sequence of transitions $\sigma = \lambda_1 t_1, \lambda_2 t_2, \ldots, \lambda_k t_k \in (\mathbb{Q}_{\geq 0} \times T)^*$. We let $M \xrightarrow[\mathbb{Q}]{\sigma} M'$ denote that there exist continuous markings $M_1, \ldots, M_{k-1}$ such that $M \xrightarrow[\mathbb{Q}]{\lambda_1 t_1} M_1 \xrightarrow[\mathbb{Q}]{\lambda_2 t_2} M_2 \cdots M_{k-1} \xrightarrow[\mathbb{Q}]{\lambda_k t_k} M'$. Further, $M \xrightarrow[\mathbb{Q}]{*} M'$ denotes that $M \xrightarrow[\mathbb{Q}]{\sigma} M'$ holds for some continuous firing sequence σ.

The *Parikh image* of $\sigma = \lambda_1 t_1, \lambda_2 t_2, \ldots, \lambda_k t_k \in (\mathbb{Q}_{\geq 0} \times T)^*$ is the vector $\overrightarrow{\sigma} \in \mathbb{Q}_{\geq 0}^T$ where $\overrightarrow{\sigma}[t] = \sum_{i=1}^{k} \delta_{i,t} \lambda_i$, where $\delta_{i,t} = 1$ if $t_i = t$ and 0 otherwise. The support of σ is the support of its Parikh image $\overrightarrow{\sigma}$. If $M \xrightarrow[\mathbb{Q}]{\sigma} M'$ then $\overrightarrow{\sigma}$ is a solution of the marking equation over $\mathbb{Q}_{\geq 0}^T$, but the converse does not hold. In [14], Fraca and Haddad strengthen this necessary condition to make it also sufficient, and use the resulting characterization to derive a polynomial algorithm.

Theorem 3 ([14]). *Let $(\mathcal{N}, M, M')$ be a Petri net system.*

- *$M \xrightarrow[\mathbb{Q}]{\sigma} M'$ iff $\overrightarrow{\sigma}$ is a solution of the marking equation over $\mathbb{Q}_{\geq 0}^T$, and there exist continuous firing sequences τ, τ' and continuous markings L and L' such that $[\![\tau]\!] = [\![\sigma]\!] = [\![\tau']\!]$, $M \xrightarrow[\mathbb{Q}]{\tau} L$, and $L' \xrightarrow[\mathbb{Q}]{\tau'} M'$.*
- *It can be decided in polynomial time if $M \xrightarrow[\mathbb{Q}]{*} M'$ holds.*

Scaling. It follows easily from the definitions that $nM \xrightarrow{*} nM'$ holds for some $n \geq 1$ iff $M \xrightarrow[\mathbb{Q}]{*} M'$. Indeed, if $M \xrightarrow[\mathbb{Q}]{\sigma} M'$ for some $\sigma = \lambda_1 t_1, \lambda_2 t_2, \ldots, \lambda_k t_k \in (\mathbb{Q}_{\geq 0} \times T)^*$, then we can scale this continuous firing sequence to a discrete sequence $nM \xrightarrow[\mathbb{Q}]{n\sigma} nM'$ where n is the smallest number such that $n\lambda_1, \ldots, n\lambda_k \in \mathbb{N}$, and $n\sigma = t_1^{n\lambda_1} t_2^{n\lambda_2} \ldots t_k^{n\lambda_k}$. So Theorem 3 immediately implies that the existence of $n \geq 1$ such that $nM \xrightarrow{*} nM'$ can be decided in polynomial time. The following lemma also gives a bound on n.

Lemma 3. *Let $(\mathcal{N}, M, M')$ be a Petri net system with weight w such that $M \xrightarrow[\mathbb{Q}]{\sigma} M'$ for some continuous firing sequence $\sigma \in (\mathbb{Q}_{\geq 0} \times T)^*$. Let m be the number of transitions in $[\![\sigma]\!]$ and let ℓ be $\|\overrightarrow{\sigma}\|$. Let k be the smallest natural number such that $k\overrightarrow{\sigma} \in \mathbb{N}^T$. Then, there exists a firing sequence $\tau \in T^*$ such that $[\![\tau]\!] = [\![\sigma]\!]$ and*

$$\big(16w(w+1)^{2m} k\ell \cdot M\big) \xrightarrow{\tau} \big(16w(w+1)^{2m} k\ell \cdot M'\big)$$

Lemma 4. (Scaling Lemma). *Let $(\mathcal{N}, M, M')$ be a Petri net system such that $M \xrightarrow[\mathbb{Q}]{\sigma} M'$. There exists a number n with a polynomial number of bits in $|\mathcal{N}|(\log \|M\| + \log \|M'\|)$ such that $nM \xrightarrow{\tau} nM'$ for some τ with $[\![\tau]\!] = [\![\sigma]\!]$.*

4.2 The Insertion Lemma

In the acyclic case, the existence of a cut-off is characterized by the existence of solutions to the marking equation $\mathbb{Q}^T_{\geq 0}$ and $\mathbb{Z}^T$. Intuitively, in the general case we replace the existence of solutions over $\mathbb{Q}^T_{\geq 0}$ by the conditions of the Scaling Lemma, and the existence of solutions over $\mathbb{Z}^T$ by the Insertion Lemma:

Lemma 5 (Insertion Lemma). *Let M, M', L, L' be markings of $\mathcal{N}$ satisfying $M \xrightarrow{\sigma} M'$ for some $\sigma \in T^*$ and $L' = L + \mathcal{A}\mathbf{y}$ for some $\mathbf{y} \in \mathbb{Z}^T$ such that $[\![\mathbf{y}]\!] \subseteq [\![\sigma]\!]$. Then $\mu M + L \xrightarrow{*} \mu M' + L'$ for $\mu = \|\mathbf{y}\|(\|\overrightarrow{\sigma}\|nw + nw + 1)$, where w is the weight of $\mathcal{N}$, and n is the number of places in ${}^\bullet[\![\sigma]\!]$.*

The idea of the proof is a follows: In a first stage, we asynchronously execute multiple "copies" of the firing sequence σ from multiple "copies" of the marking M, until we reach a marking at which all places of ${}^\bullet[\![\sigma]\!]$ contain a sufficiently large number of tokens. At this point we temporarily interrupt the executions of the copies of σ to *insert* a firing sequence with Parikh mapping $\|\mathbf{y}\|\overrightarrow{\sigma} + \mathbf{y}$. The net effect of this sequence is to transfer some copies of M to M', leaving the other copies untouched, and exactly one copy of L to L'. In the third stage, we resume the interrupted executions of the copies of σ, which completes the transfer of the remaining copies of M to M' .

Proof. Let $\mathbf{x}$ be the Parikh image of σ, i.e., $\mathbf{x} = \overrightarrow{\sigma}$. Since $M \xrightarrow{\sigma} M'$, by the marking equation we have $M' = M + \mathcal{A}\mathbf{x}$

First stage: Let $\lambda_x = \|x\|$, $\lambda_y = \|y\|$ and $\mu = \lambda_y(\lambda_x nw + nw + 1)$. Let $\sigma := r_1, r_2, \ldots, r_k$ and let $M =: M_0 \xrightarrow{r_1} M_1 \xrightarrow{r_2} M_2 \ldots M_{k-1} \xrightarrow{r_k} M_k := M$. Notice that for each place $p \in {}^\bullet[\![\sigma]\!]$, there exists a marking $M_{i_p} \in \{M_0, \ldots, M_{k-1}\}$ such that $M_{i_p}(p) > 0$.

Since each of the markings in $\{M_{i_p}\}_{p \in {}^\bullet[\![\sigma]\!]}$ can be obtained from M by firing a (suitable) prefix of σ, it is easy to see that from the marking $\mu M + L = \lambda_y M + L + (\lambda_x\lambda_y nw + \lambda_y nw)M$ we can reach the marking $\texttt{First} := \lambda_y M + L + \sum_{p \in {}^\bullet[\![\sigma]\!]} (\lambda_x \lambda_y w + \lambda_y w) M_{i_p}$. This completes our first stage.

Second stage - Insert: Since $[\![\mathbf{y}]\!] \subseteq [\![\sigma]\!]$, if $\mathbf{y}(t) \neq 0$ then $\mathbf{x}(t) \neq 0$. Since $\mathbf{x}(t) \geq 0$ for every transition, it now follows that $(\lambda_y \mathbf{x} + \mathbf{y})(t) \geq 0$ for every transition t and $(\lambda_y \mathbf{x} + \mathbf{y})(t) > 0$ precisely for those transitions in $[\![\sigma]\!]$.

Let ξ be any firing sequence such that $\overrightarrow{\xi} = \lambda_y \mathbf{x} + \mathbf{y}$. Notice that for every place $p \in {}^\bullet[\![\sigma]\!]$, $\texttt{First}(p) \geq \lambda_x \lambda_y w + \lambda_y w \geq \|(\lambda_y \mathbf{x} + \mathbf{y})\| \cdot w$. By an easy induction on $\|\xi\|$, it follows that that $\texttt{First} \xrightarrow{\xi} \texttt{Second}$ for some marking $\texttt{Second}$. By the marking equation, it follows that $\texttt{Second} = \lambda_y M' + L' + \sum_{p \in {}^\bullet[\![\sigma]\!]} (\lambda_x \lambda_y w + \lambda_y w) M_{i_p}$. This completes our second stage.

Third stage: Notice that for each place $p \in {}^\bullet[\![\sigma]\!]$, by construction of M_{i_p}, there is a firing sequence which takes the marking M_{i_p} to the marking M'. It then follows that there is a firing sequence which takes the marking $\texttt{Second}$ to the marking $\lambda_y M' + L' + \sum_{p \in {}^\bullet[\![\sigma]\!]} (\lambda_x \lambda_y w + \lambda_y w) M' = \mu M' + L'$. This completes our third stage and also completes the desired firing sequence from $\mu M + L$ to $\mu M' + L'$.

5 Polynomial time algorithm for the general case

Let $(\mathcal{N}, M, M')$ be a net system with $\mathcal{N} = (P, T, Pre, Post)$, such that $\mathcal{A}$ is its incidence matrix. As in Section 3, we first characterize the Petri net systems that admit a cut-off, and then provide a polynomial time algorithm.

5.1 Characterizing systems with cut-offs

We generalize the characterization of Theorem 1 for acyclic Petri net systems to general systems.

Theorem 4. *A Petri net system $(\mathcal{N}, M, M')$ admits a cut-off iff there exists some rational firing sequence σ such that $M \xrightarrow[\mathbb{Q}]{\sigma} M'$ and the marking equation has a solution $\mathbf{y} \in \mathbb{Z}^T$ such that $[\![\mathbf{y}]\!] \subseteq [\![\sigma]\!]$.*

Proof. ($\Rightarrow$): Assume $(\mathcal{N}, M, M')$ admits a cut-off. Hence there exists $B \in \mathbb{N}$ such that for all $n \geq B$ we have $nM \xrightarrow{*} nM'$. Similar to the proof of theorem 1, we can show that there exist $n \in \mathbb{N}$ and firing sequences τ, τ' such that $nM \xrightarrow{\tau} nM'$, $(n+1)M \xrightarrow{\tau'} (n+1)M'$ and $[\![\tau']\!] \subseteq [\![\tau]\!]$.

Let $\tau = t_1 t_2 \cdots t_k$. Construct the rational firing sequence $\sigma := t_1/n\, t_2/n \cdots t_k/n$. From the fact that $nM \xrightarrow{\tau} nM'$, we can easily conclude by induction on k that $M \xrightarrow[\mathbb{Q}]{\sigma} M'$. Further, by the marking equation we have $nM' = nM + \mathcal{A}\overrightarrow{\tau}$ and $(n+1)M' = (n+1)M + \mathcal{A}\overrightarrow{\tau'}$. Let $\mathbf{y} = \overrightarrow{\tau'} - \overrightarrow{\tau}$. Then $\mathbf{y} \in \mathbb{Z}^T$ and $M' = M + \mathcal{A}\mathbf{y}$. Further, since $[\![\tau']\!] \subseteq [\![\tau]\!] = [\![\sigma]\!]$, we have $[\![\mathbf{y}]\!] \subseteq [\![\sigma]\!]$.

($\Leftarrow$): Assume there exists a rational firing sequence σ and a vector $\mathbf{y} \in \mathbb{Z}^T$ such that $[\![\mathbf{y}]\!] \subseteq [\![\sigma]\!]$, $M \xrightarrow[\mathbb{Q}]{\sigma} M'$ and $M' = M + \mathcal{A}\mathbf{y}$. Let $s = |\mathcal{N}|(\log \|M\| + \log \|M'\|)$. It is well known that if a system of linear equations over the integers is feasible, then there is a solution which can be described using a number of bits which is polynomial in the size of the input (see e.g. [20]). Hence, we can assume that $\|\mathbf{y}\|$ can be described using $s^{O(1)}$ bits.

By Lemma 4 there exists n (which can be described using $s^{O(1)}$ bits) and a firing sequence τ with $[\![\tau]\!] = [\![\sigma]\!]$ such that $nM \xrightarrow{\tau} nM'$. Hence $knM \xrightarrow{*} knM'$ is also possible for any $k \in \mathbb{N}$. By Lemma 5, there exists μ (which can once again be described using $s^{O(1)}$ bits) such that $\mu nM + M \xrightarrow{*} \mu nM' + M'$ is possible. By Lemma 1 the system $(\mathcal{N}, M, M')$ admits a cut-off with a polynomial number of bits in s.

Notice that we have actually proved that if a net system admits a cut-off then it admits a cut-off with a polynomial number of bits in its size. Since the cut-off problem for a rendez-vous protocol $\mathcal{P}$ can be reduced to a cut-off problem for the Petri net system $(\mathcal{N}_{\mathcal{P}}, \wr init \wr, \wr fin \wr)$, it follows that,

Corollary 1. *If the system $(\mathcal{N}, M, M')$ admits a cut-off then it admits a cut-off with a polynomial number of bits in $|\mathcal{N}|(\log \|M\| + \log \|M'\|)$. Hence, if a rendez-vous protocol $\mathcal{P}$ admits a cut-off then it admits a cut-off with a polynomial number of bits in $|\mathcal{P}|$.*

5.2 Polynomial time algorithm

We use the characterization given in the previous section to provide a polynomial time algorithm for the cut-off problem. The following lemma, which was proved in [14] and whose proof is given in the appendix, enables us to find a firing sequence between two markings with maximum support.

Lemma 6. *[14] Among all the rational firing sequences σ such that $M \xrightarrow[\mathbb{Q}]{\sigma} M'$, there is one with maximum support. Moreover, the support of such a firing sequence can be found in polynomial time.*

We now have all the ingredients to prove the existence of a polynomial time algorithm.

Theorem 5. *The cut-off problem for net systems can be solved in polynomial time.*

Proof. First, we check that there is a rational firing sequence σ with $M \xrightarrow[\mathbb{Q}]{\sigma} M'$, which can be done in polynomial time by ([14], Proposition 27). If such a sequence does not exist, by Theorem 4 the given net system does not admit a cut-off.

Suppose such a sequence exists. By Lemma 6 we can find in polynomial time, the maximum support S of all the firing sequences τ such that $M \xrightarrow[\mathbb{Q}]{\tau} M'$. We now check in polynomial time if the marking equation has a solution $\mathbf{y}$ over $\mathbb{Z}^T$ such that $\mathbf{y}(t) = 0$ for every $t \notin S$. By Theorem 4 such a solution exists iff the net system admits a cut-off.

This immediately proves that the cut-off problem for rendez-vous protocols is also in polynomial time. By an easy logspace reduction from the Circuit Value Problem [21], we prove that

Lemma 7. *The cut-off problem for rendez-vous protocols is P-hard.*

Clearly, this also proves that the cut-off problem for Petri nets is P-hard.

6 Symmetric rendez-vous protocols

In [17] Horn and Sangnier introduce symmetric rendez-vous protocols, where sending and receiving a message at each state has the same effect, and show that the cut-off problem is in NP. We improve on their result and shown that it is in NC.

Recall that NC is the set of problems in P that can be solved in polylogarithmic *parallel* time, i.e., problems which can be solved by a uniform family of circuits with polylogarithmic depth and polynomial number of gates. Two well-known problems which lie in NC are graph reachability and feasibility of linear equations over the finite field $\mathbb{F}_2$ of size 2 [27,23]. We proceed to formally define symmetric protocols and state our results.

Definition 3. *A rendez-vous protocol $\mathcal{P} = (Q, \Sigma, init, fin, R)$ is* symmetric*, iff its set of rules is symmetric under swapping $!a$ and $?a$ for each $a \in \Sigma$, i.e., for each $a \in \Sigma$, we have $(q, !a, q') \in R$ iff $(q, ?a, q') \in R$.*

Horn and Sangnier show that, because of their symmetric nature, there is a very easy characterization for cut-off admitting symmetric protocols.

Proposition 3. *([17], Lemma 18) A symmetric protocol $\mathcal{P}$ admits a cut-off iff there exists an even number e and an odd number o such that $C^e_{init} \xrightarrow{*} C^e_{fin}$ and $C^o_{init} \xrightarrow{*} C^o_{fin}$.*

From a symmetric protocol $\mathcal{P}$, we can derive a graph $G(\mathcal{P})$ where the vertices are the states and there is an edge between q and q' iff there exists $a \in \Sigma$ such that $(q, a, q') \in R$. The following proposition is immediate from the definition of symmetric protocols:

Proposition 4. *Let $\mathcal{P}$ be a symmetric protocol. There exists an even number e such that $C^e_{init} \xrightarrow{*} C^e_{fin}$ iff there is a path from $init$ to fin in the graph $G(\mathcal{P})$.*

Proof. The left to right implication is obvious. For the other side, suppose there is a path $init, q_1, q_2, \ldots, q_{m-1}, fin$ in the graph $G(\mathcal{P})$. Then notice that $\wr 2 \cdot init \int \rightarrow \wr 2 \cdot q_1 \int \rightarrow \wr 2 \cdot q_2 \int \cdots \rightarrow \wr 2 \cdot q_{m-1} \int \rightarrow \wr 2 \cdot q_f \int$ is a valid run of the protocol.

Since graph reachability is in NC , this takes care of the "even" case from Proposition 3. Hence, we only need to take care of the "odd" case from Proposition 3.

Fix a symmetric protocol $\mathcal{P}$ for the rest of the section. As a first step, for each state $q \in Q$, we compute if there is a path from $init$ to q and if there is a path from q to fin in the graph $\mathcal{G}(\mathcal{P})$. Since graph reachability is in NC this computation can be carried out in NC by parallely running graph reachability for each $q \in Q$. If such paths exist for a state q then we call q a good state, and otherwise a bad state. The following proposition easily follows from the symmetric nature of $\mathcal{P}$:

Proposition 5. *If $q \in Q$ is a good state, then $\wr 2 \cdot init \int \xrightarrow{*} \wr 2 \cdot q \int$ and $\wr 2 \cdot q \int \xrightarrow{*} \wr 2 \cdot fin \int$.*

Similar to the general case of rendez-vous protocols, given a symmetric protocol $\mathcal{P}$ we can construct a Petri net $\mathcal{N}_{\mathcal{P}}$ whose places are the states of $\mathcal{P}$ and which faithfully represents the reachability relation of configurations of $\mathcal{P}$. Observe that this construction can be carried out in parallel over all the states in Q and over all pairs of rules in R. Let $\mathcal{N} = (P, T, Pre, Post)$ be the Petri net that we construct out of the symmetric protocol $\mathcal{P}$ and let $\mathcal{A}$ be its incidence matrix. We now write the marking equation for $\mathcal{N}$ as follows: We introduce a variable $\mathbf{v}[t]$ for each transition $t \in T$ and we construct an equation system Eq enforcing the following three conditions:

- $\mathbf{v}[t] = 0$ for every $t \in T$ such that ${}^\bullet t \cup t^\bullet$ contains a bad state.
  By definition of a bad state, such transitions will never be fired on any run from an initial to a final configuration and so our requirement is safe.

– $\sum_{t \in T} \mathcal{A}[q,t] \cdot \mathbf{v}[t] = 0$ for each $q \notin \{\mathit{init}, \mathit{fin}\}$.
Notice that the net-effect of any run from an initial to a final configuration on any state not in $\{\mathit{init}, \mathit{fin}\}$ is 0 and hence this condition is valid as well.
– $\sum_{t \in T} \mathcal{A}[\mathit{init},t] \cdot \mathbf{v}[t] = -1$ and $\sum_{t \in T} \mathcal{A}[\mathit{fin},t] \cdot \mathbf{v}[t] = 1$.

It is clear that the construction of Eq can be carried out in parallel over each $q \in Q$ and each $t \in T$. Finally, we solve Eq over arithmetic modulo 2, i.e., we solve Eq over the field $\mathbb{F}_2$ which as mentioned before can be done in NC. We have:

Lemma 8. *There exists an odd number o such that $C^o_{\mathit{init}} \xrightarrow{*} C^o_{\mathit{fin}}$ iff the equation system Eq has a solution over $\mathbb{F}_2$.*

Proof. (Sketch.) The left to right implication is true because of taking modulo 2 on both sides of the marking equation. For the other side, we use an idea similar to Lemma 5. Let $\mathbf{x}$ be a solution to Eq over $\mathbb{F}_2$. Using Proposition 5 we first populate all the good states of Q with enough processes such that all the good states except *init* have an even number of processes. Then, we fire exactly once, all the transitions t such that $\mathbf{x}[t] = 1$. Since $\mathbf{x}$ satisfies Eq, we can now argue that in the resulting configuration, the number of processes at each bad state is 0 and the number of processes in each good state except *fin* is even. Hence, we can once again use Proposition 5 to conclude that we can move all the processes which are not at *fin* to the final state *fin*.

Theorem 6. *The problem of deciding whether a symmetric protocol admits a cut-off is in* NC.

Proof. By Proposition 3 it suffices to find an even number e and an odd number o such that $C^e_{\mathit{init}} \xrightarrow{*} C^e_{\mathit{fin}}$ and $C^o_{\mathit{init}} \xrightarrow{*} C^o_{\mathit{fin}}$. By Proposition 4 the former can be done in NC. By Lemma 8 and by the fact that the equation system Eq can be constructed and solved in NC, it follows that the latter can also be done in NC.

7 Symmetric protocols with leaders

In this section, we extend symmetric rendez-vous protocols by adding a special process called leader. We state the cut-off problem for such protocols and prove that it is NP-complete.

Definition 4. *A symmetric leader protocol is a pair of symmetric protocols $\mathcal{P} = (\mathcal{P}^L, \mathcal{P}^F)$ where $\mathcal{P}^L = (Q^L, \Sigma, \mathit{init}^L, \mathit{fin}^L, R^L)$ is the leader protocol and $\mathcal{P}^F = (Q^F, \Sigma, \mathit{init}^F, \mathit{fin}^F, R^F)$ is the follower protocol where $Q^L \cap Q^F = \emptyset$.*

A configuration of a symmetric leader protocol $\mathcal{P}$ is a multiset over $Q^L \cup Q^F$ such that $\sum_{q \in Q^L} C(q) = 1$. This corresponds to the intuition that exactly one process can execute the leader protocol. For each $n \in \mathbb{N}$, let C^n_{init} (resp. C^n_{fin}) denote the initial (resp. final) configuration of $\mathcal{P}$ given by $C^n_{\mathit{init}}(\mathit{init}^L) = 1$ (resp. $C^n_{\mathit{fin}}(\mathit{fin}^L) = 1$) and $C^n_{\mathit{init}}(\mathit{init}^F) = n$ (resp. $C^n_{\mathit{fin}}(\mathit{fin}^F) = n$). We say that $C \Rightarrow C'$

if there exists $(p, !a, p'), (q, ?a, q') \in R^L \cup R^F$, $C \geq \wr p, q \wr$ and $C' = C - \wr p, q \wr + \wr p', q' \wr$. Since we allow at most one process to execute the leader protocol, given a configuration C, we can let $lead(C)$ denote the unique state $q \in Q^L$ such that $C(q) > 0$.

Definition 5. *The cut-off problem for symmetric leader protocols is the following.*

> *Input: A symmetric leader protocol $\mathcal{P} = (\mathcal{P}^L, \mathcal{P}^F)$.*
> *Output: Is there $B \in \mathbb{N}$ such that for all $n \geq B$, $C^n_{init} \xRightarrow{*} C^n_{fin}$.*

We know the following fact regarding symmetric leader protocols.

Proposition 6. *([17], Lemma 18) A symmetric leader protocol admits a cut-off iff there exists an even number e and an odd number o such that $C^e_{init} \xRightarrow{*} C^e_{fin}$ and $C^o_{init} \xRightarrow{*} C^o_{fin}$.*

The main theorem of this section is

Theorem 7. *The cut-off problem for symmetric leader protocols is* NP*-complete*

7.1 A non-deterministic polynomial time algorithm

Let $\mathcal{P} = (\mathcal{P}^L, \mathcal{P}^F)$ be a symmetric leader protocol with $\mathcal{P}^L = (Q^L, \Sigma, init^L, fin^L, R^L)$ and $\mathcal{P}^F = (Q^F, \Sigma, init^F, fin^F, R^F)$. Similar to the previous section, from $\mathcal{P}^F$ we can construct a graph $\mathcal{G}(\mathcal{P}^F)$ where the vertices are given by the states Q^F and the edges are given by the rules in R^F. In $\mathcal{G}(\mathcal{P}^F)$, we can clearly remove all vertices which are not reachable from the state $init^F$ and which do not have a path to fin^F. In the sequel, we will assume that such vertices do not exist in $\mathcal{G}(\mathcal{P}^F)$.

Similar to the general case, we will construct a Petri net $\mathcal{N}_\mathcal{P}$ from the given symmetric leader protocol $\mathcal{P}$. However, the construction is made slightly complicated due to the presence of a leader.

From $\mathcal{P} = (\mathcal{P}^L, \mathcal{P}^F)$, we construct a Petri net $\mathcal{N} = (P, T, Pre, Post)$ as follows: Let P be $Q^L \cup Q^F$. For each $a \in \Sigma$ and $r = (q, !a, s), r' = (q', ?a, s') \in R^L \cup R^F$ such that *at most one of r and r' belongs to R^L*, we will have a transition $t_{r,r'} \in T$ in $\mathcal{N}$ such that

- $Pre[p, t] = 0$ for every $p \notin \{q, q'\}$, $Post[p, t] = 0$ for every $p \notin \{s, s'\}$
- If $q = q'$ then $Pre[q, t] = -2$, otherwise $Pre[q, t] = Pre[q', t] = -1$
- If $s = s'$ then $Post[s, t] = 2$, otherwise $Post[s, t] = Post[s', t] = 1$.

Transitions $t_{r,r'}$ in which exactly one of r, r' is in R^L will be called *leader transitions* and transitions in which both of r, r' are in R^F will be called *follower-only transitions*. Notice that if t is a leader transition, then there is a unique place $p \in {}^\bullet t \cap Q^L$ and a unique place $p \in t^\bullet \cap Q^L$. These places will be denoted by $t.from$ and $t.to$ respectively.

As usual, we let $\mathcal{A}$ denote the incidence matrix of the constructed net $\mathcal{N}$. The following proposition is obvious from the construction of the net $\mathcal{N}$

Proposition 7. *For two configurations C and C', we have that $C \overset{*}{\Rightarrow} C'$ in the protocol $\mathcal{P}$ iff $C \xrightarrow{*} C$ in the net $\mathcal{N}$.*

Because $\mathcal{P}$ is symmetric we have the following fact, which is easy to verify.

Proposition 8. *If $q \in Q^F$, then $\wr 2 \cdot init^F \wr \xrightarrow{*} \wr 2 \cdot q \wr \xrightarrow{*} \wr 2 \cdot fin^F \wr$*

For any vector $\mathbf{x} \in \mathbb{N}^T$, we define $lead(\mathbf{x})$ to be the set of all leader transitions such that $\mathbf{x}[t] > 0$. The graph of the vector $\mathbf{x}$, denoted by $\mathcal{G}(\mathbf{x})$ is defined as follows: The set of vertices is the set $\{t.from : t \in lead(\mathbf{x})\} \cup \{t.to : t \in lead(\mathbf{x})\}$. The set of edges is the set $\{(t.from, t.to) : t \in lead(\mathbf{x})\}$. Further, for any two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{N}^T$ and a transition $t \in T$, we say that $\mathbf{x} = \mathbf{y}[t\text{--}]$ iff $\mathbf{x}[t] = \mathbf{y}[t] - 1$ and $\mathbf{x}[t'] = \mathbf{y}[t']$ for all $t' \neq t$.

Definition 6. *Let C be a configuration and let $\mathbf{x} \in \mathbb{N}^T$. We say that the pair $(C, \mathbf{x})$ is compatible if $C + \mathcal{A}\mathbf{x} \geq \mathbf{0}$ and every vertex in $\mathcal{G}(\mathbf{x})$ is reachable from $lead(C)$.*

The following lemma states that *as long as there are enough followers in every state*, it is possible for the leader to come up with a firing sequence from a compatible pair.

Lemma 9. *Suppose $(C, \mathbf{x})$ is a compatible pair such that $C(q) \geq 2\|\mathbf{x}\|$ for every $q \in Q^F$. Then there is a configuration D and a firing sequence ξ such that $C \xrightarrow{\xi} D$ and $\overrightarrow{\xi} = \mathbf{x}$.*

Proof. (Sketch.) We prove by induction on $\|\mathbf{x}\|$. If $\mathbf{x}[t] > 0$ for some follower-only transition, then it is easy to verify that if we let C' be such that $C \xrightarrow{t} C'$ and $\mathbf{x}'$ be $\mathbf{x}[t\text{--}]$, then $(C', \mathbf{x}')$ is compatible and $C(q) \geq 2\|\mathbf{x}'\|$ for every $q \in Q^F$.

Suppose $\mathbf{x}[t] > 0$ for some leader transition. Let $p = lead(C)$. If p belongs to some cycle $S = p, r_1, p_1, r_2, p_2, \ldots, p_k, r_{k+1}, p$ in the graph $\mathcal{G}(\mathbf{x})$, then we let $C \xrightarrow{r_1} C'$ and $\mathbf{x}' = \mathbf{x}[t\text{--}]$. It is easy to verify that $C' + \mathcal{A}\mathbf{x}' \geq \mathbf{0}$, $C'(q) \geq 2\|\mathbf{x}'\|$ for every $q \in Q^F$ and $lead(C') = p_1$. Any path P in $\mathcal{G}(\mathbf{x})$ from p to some vertex s either goes through p_1 or we can use the cycle S to traverse from p_1 to p first and then use P to reach s. This gives a path from p_1 to every vertex s in $\mathcal{G}(\mathbf{x}')$.

If p does not belong to any cycle in $\mathcal{G}(\mathbf{x})$, then using the fact that $C + \mathcal{A}\mathbf{x} \geq 0$, we can show that there is exactly one out-going edge t from p in $\mathcal{G}(\mathbf{x})$. We then let $C \xrightarrow{t} C'$ and $\mathbf{x}' = \mathbf{x}[t\text{--}]$. Since any path in $\mathcal{G}(\mathbf{x})$ from p has to necessarily use this edge t, it follows that in $\mathcal{G}(\mathbf{x}')$ there is a path from $t.to = lead(C')$ to every vertex.

Lemma 10. *Let $par \in \{0, 1\}$. There exists $k \in \mathbb{N}$ such that $C^k_{init} \xrightarrow{*} C^k_{fin}$ and $k \equiv par \pmod 2$ iff there exists $n \in \mathbb{N}$, $\mathbf{x} \in \mathbb{N}^T$ such that $n \equiv par \pmod 2$, $(C^n_{init}, \mathbf{x})$ is compatible and $C^n_{fin} = C^n_{init} + \mathcal{A}\mathbf{x}$.*

Proof. (Sketch.) The left to right implication is easy and follows from the marking equation along with induction on the number of leader transitions in the run. For the other side, we use an idea similar to Lemma 5. Let $(C^n_{init}, \mathbf{x})$ be the given compatible pair. We first use Proposition 8 to populate all the states of Q^F with enough processes such that all the states of Q^F except $init^F$ have an even number of processes. Then we use Lemma 9 to construct a firing sequence ξ which can be fired from C^n_{init} and such that $\overrightarrow{\xi} = \mathbf{x}$. By means of the marking equation, we then argue that in the resulting configuration, the leader is in the final state, n followers are in the state fin^F and every other follower state has an even number of followers. Once again, using Proposition 8 we can now move all the processes which are not at fin^F to the final state fin^F.

Lemma 11. *Given a symmetric leader protocol, checking whether a cut-off exists can be done in NP.*

Proof. By Proposition 6 it suffices to find an even number e and an odd number o such that $C^e_{init} \xrightarrow{*} C^e_{fin}$ and $C^o_{init} \xrightarrow{*} C^o_{fin}$. Suppose we want to check that there exists $2k \in \mathbb{N}$ such that $C^{2k}_{init} \xrightarrow{*} C^{2k}_{fin}$. We first non-deterministically guess a set of leader transitions $S = \{t_1, \dots, t_k\}$ and check that for each $t \in S$, we can reach $t.from$ and $t.to$ from $init^L$ using only the transitions in S.

Once we have guessed all this, we write a polynomially sized integer linear program as follows: We let $\mathbf{v}$ denote $|T|$ variables, one for each transition in T and we let n be another variable, with all these variables ranging over $\mathbb{N}$. We then enforce the following conditions: $C^{2n}_{fin} = C^{2n}_{init} + \mathcal{A}\mathbf{v}$ and $\mathbf{v}[t] = 0 \iff t \notin S$ and solve the resulting linear program, which we can do in non-deterministic polynomial time [26]. If there exists a solution, then we accept. Otherwise, we reject.

By Lemma 10 and by the definition of compatibility, it follows that at least one of our guesses gets accepted iff there exists $2k \in \mathbb{N}$ such that $C^{2k}_{init} \xrightarrow{*} C^{2k}_{fin}$. Similarly we can check if exists $2l+1 \in \mathbb{N}$ such that $C^{2l+1}_{init} \xrightarrow{*} C^{2l+1}_{fin}$.

By a reduction from 3-SAT, we prove that

Lemma 12. *The cut-off problem for symmetric leader protocols is NP-hard.*

References

1. Alla, H., David, R.: Continuous and hybrid Petri nets. J. Circuits Syst. Comput. **8**(1), 159–188 (1998)
2. Angluin, D., Aspnes, J., Diamadi, Z., Fischer, M.J., Peralta, R.: Computation in networks of passively mobile finite-state sensors. Distributed Computing **18**(4), 235–253 (2006). https://doi.org/10.1007/s00446-005-0138-3
3. Basler, G., Mazzucchi, M., Wahl, T., Kroening, D.: Symbolic counter abstraction for concurrent software. In: Bouajjani, A., Maler, O. (eds.) 21st International Conference on Computer Aided Verification, CAV 2009, Grenoble, France, June 26 - July 2, 2009, Proceedings. Lecture Notes in Computer Science, vol. 5643, pp. 64–78. Springer (2009). https://doi.org/10.1007/978-3-642-02658-4_9

4. Bloem, R., Jacobs, S., Khalimov, A., Konnov, I., Rubin, S., Veith, H., Widder, J.: Decidability of Parameterized Verification. Synthesis Lectures on Distributed Computing Theory, Morgan & Claypool Publishers (2015). https://doi.org/10.2200/S00658ED1V01Y201508DCT013
5. Blondin, M.: The abc of Petri net reachability relaxations. ACM SIGLOG News **7**(3) (2020)
6. Czerwinski, W., Lasota, S., Lazic, R., Leroux, J., Mazowiecki, F.: The reachability problem for Petri nets is not elementary. In: Charikar, M., Cohen, E. (eds.) 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019, Proceedings. pp. 24–33. ACM (2019). https://doi.org/10.1145/3313276.3316369
7. Delzanno, G., Sangnier, A., Traverso, R., Zavattaro, G.: On the complexity of parameterized reachability in reconfigurable broadcast networks. In: FSTTCS. LIPIcs, vol. 18, pp. 289–300. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2012)
8. Emerson, E.A., Kahlon, V.: Model checking large-scale and parameterized resource allocation systems. In: TACAS. Lecture Notes in Computer Science, vol. 2280, pp. 251–265. Springer (2002)
9. Esparza, J.: Decidability and complexity of Petri net problems - an introduction. In: Petri Nets. Lecture Notes in Computer Science, vol. 1491, pp. 374–428. Springer (1996)
10. Esparza, J.: Parameterized verification of crowds of anonymous processes. In: Dependable Software Systems Engineering, NATO Science for Peace and Security Series - D: Information and Communication Security, vol. 45, pp. 59–71. IOS Press (2016)
11. Esparza, J., Finkel, A., Mayr, R.: On the verification of broadcast protocols. In: LICS. pp. 352–359. IEEE Computer Society (1999)
12. Esparza, J., Ganty, P., Leroux, J., Majumdar, R.: Verification of population protocols. Acta Informatica **54**(2), 191–215 (2017). https://doi.org/10.1007/s00236-016-0272-3
13. Esparza, J., Nielsen, M.: Decidability issues for Petri nets - a survey. J. Inf. Process. Cybern. **30**(3), 143–160 (1994)
14. Fraca, E., Haddad, S.: Complexity analysis of continuous Petri nets. Fundam. Informaticae **137**(1), 1–28 (2015)
15. German, S.M., Sistla, A.P.: Reasoning about systems with many processes. Journal of the ACM **39**(3), 675–735 (1992). https://doi.org/10.1145/146637.146681
16. Gmeiner, A., Konnov, I., Schmid, U., Veith, H., Widder, J.: Tutorial on parameterized model checking of fault-tolerant distributed algorithms. In: SFM. Lecture Notes in Computer Science, vol. 8483, pp. 122–171. Springer (2014)
17. Horn, F., Sangnier, A.: Deciding the existence of cut-off in parameterized rendezvous networks. In: CONCUR. LIPIcs, vol. 171, pp. 46:1–46:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2020)
18. Kaiser, A., Kroening, D., Wahl, T.: Dynamic cutoff detection in parameterized concurrent programs. In: Touili, T., Cook, B., Jackson, P.B. (eds.) 22nd International Conference on Computer Aided Verification, CAV 2010, Edinburgh, UK, July 15-19, 2010, Proceedings. Lecture Notes in Computer Science, vol. 6174, pp. 645–659. Springer (2010). https://doi.org/10.1007/978-3-642-14295-6_55
19. Kannan, R., Bachem, A.: Polynomial algorithms for computing the Smith and Hermite normal forms of an integer matrix. SIAM J. Comput. **8**(4), 499–507 (1979)

20. Kannan, R., Monma, C.L.: On the computational complexity of integer programming problems. In: Henn, R., Korte, B., Oettli, W. (eds.) Optimization and Operations Research. pp. 161–172. Springer Berlin Heidelberg, Berlin, Heidelberg (1978)
21. Ladner, R.E.: The circuit value problem is Log Space complete for P. SIGACT News **7**(1), 18–20 (1975)
22. Leroux, J., Schmitz, S.: Reachability in vector addition systems is primitive-recursive in fixed dimension. In: LICS. pp. 1–13. IEEE (2019)
23. Mulmuley, K.: A fast parallel algorithm to compute the rank of a matrix over an arbitrary field. Comb. **7**(1), 101–104 (1987). https://doi.org/10.1007/BF02579205
24. Murata, T.: Petri nets: Properties, analysis and applications. Proceedings of the IEEE **77**(4), 541–580 (1989)
25. Navlakha, S., Bar-Joseph, Z.: Distributed information processing in biological and computational systems. Communications of the ACM **58**(1), 94–102 (2015). https://doi.org/10.1145/2678280
26. Papadimitriou, C.H.: On the complexity of integer programming. J. ACM **28**(4), 765–768 (1981). https://doi.org/10.1145/322276.322287
27. Papadimitriou, C.H.: Computational complexity. Academic Internet Publ. (2007)
28. Pnueli, A., Xu, J., Zuck, L.D.: Liveness with (0, 1, infty)-counter abstraction. In: CAV. Lecture Notes in Computer Science, vol. 2404, pp. 107–122. Springer (2002)
29. Pohst, M.E., Zassenhaus, H.: Algorithmic algebraic number theory, Encyclopedia of mathematics and its applications, vol. 30. Cambridge University Press (1989)
30. Recalde, L., Haddad, S., Suárez, M.S.: Continuous Petri nets: Expressive power and decidability issues. Int. J. Found. Comput. Sci. **21**(2), 235–256 (2010)
31. Soloveichik, D., Cook, M., Winfree, E., Bruck, J.: Computation with finite stochastic chemical reaction networks. Nat. Comput. **7**(4), 615–633 (2008)

Appendix E

Complexity of Verification and Synthesis of Threshold Automata (ATVA 2020)

This section contains a reprinting of the following paper, which has been published as a peer-reviewed conference paper.

> A. R. Balasubramanian and Javier Esparza and Marijana Lazić. *Complexity of Verification and Synthesis of Threshold Automata.* In conference proceedings of ATVA 2020. Edited by Dang Van Hung and Oleg Sokolsky. Springer Nature Switzerland AG 2020, Springer Nature. Vol. 12302 of Lecture Notes in Computer Science, Pages - 144-160. doi: `10.1007/978-3-030-59152-6_8`

According to the Consent to Publish in Lecture Notes in Computer Science with Springer, the author of this thesis is allowed to include this paper in this thesis. The relevant excerpt from the Consent to Publish document for this paper is the following:

> Author retains the right to use his/her Contribution for his/her further scientific career by including the final published paper in his/her dissertation or doctoral thesis provided acknowledgment is given to the original source of publication.

Further, the following is found in the Permissions page of Springer.

> Authors have the right to reuse their article's Version of Record, in whole or in part, in their own thesis. Additionally, they may reproduce and make available their thesis, including Springer Nature content, as required by their awarding academic institution. Authors must properly cite the published article in their thesis according to

current citation standards.

For more information, please see `https://www.springer.com/gp/rights-permissions/obtaining-permissions/882`, in particular, the section on Author reuse.

Also, the following is present on the Rights, Permissions and Third Party Distribution page (`https://www.springernature.com/gp/partners/rights-permissions-third-party-distribution`) of Springer Nature.

> Springer Nature Book and Journal Authors have the right to reuse the Version of Record, in whole or in part, in their own thesis. Additionally, they may reproduce and make available their thesis, including Springer Nature content, as required by their awarding academic institution. Authors must properly cite the published work in their thesis according to current citation standards and include the following acknowledgement: 'Reproduced with permission from Springer Nature'.

Accordingly, we include the following acknowledgment here: The article in this section is reproduced with permission from Springer Nature.

Finally, a license has also been obtained from Springer Nature to include this article in the thesis. The license number is 5585250613477.

Summary

Threshold automata are a formalism for modeling and analyzing fault-tolerant distributed algorithms, describing protocols executed by a fixed but arbitrary number of processes. We conduct the first systematic study of the complexity of verification and synthesis problems for threshold automata. We prove that the coverability, reachability, safety, and liveness problems are NP-complete and that the bounded synthesis problem is Σ_p^2 complete. A key to our results is a novel characterization of the reachability relation of a threshold automaton as an existential Presburger formula. The characterization also leads to novel verification and synthesis algorithms. We report on an implementation and provide experimental results.

Contributions of the author of this thesis

Contribution of Balasubramanian Ayikudi Ramachandrakumar	
Scientific findings	60%
Development and conceptual design of the research project	50%
Discussion and development of ideas	50%
Experimental evaluation	100%
Drafting of the manuscript	50%

Complexity of Verification and Synthesis of Threshold Automata

A. R. Balasubramanian( ), Javier Esparza, and Marijana Lazić

Technische Universität München, Munich, Germany
bala.ayikudi@tum.de, {esparza,lazic}@in.tum.de

Abstract. Threshold automata are a formalism for modeling and analyzing fault-tolerant distributed algorithms, recently introduced by Konnov, Veith, and Widder, describing protocols executed by a fixed but arbitrary number of processes. We conduct the first systematic study of the complexity of verification and synthesis problems for threshold automata. We prove that the coverability, reachability, safety, and liveness problems are NP-complete, and that the bounded synthesis problem is Σ_p^2 complete. A key to our results is a novel characterization of the reachability relation of a threshold automaton as an existential Presburger formula. The characterization also leads to novel verification and synthesis algorithms. We report on an implementation, and provide experimental results.

Keywords: Threshold automata · Distributed algorithms · Parameterized verification

1 Introduction

Many concurrent and distributed systems consist of an arbitrary number of communicating processes. Parameterized verification investigates how to prove them correct for any number of processes [1].

Parameterized systems whose processes are indistinguishable and finite state are often called *replicated systems*. A global state of a replicated system is completely determined by the number of processes in each state. Models of replicated systems differ in the communication mechanism between processes. Vector Addition Systems (VAS) and their extensions [2,7,9,11] can model rendez-vous, multiway synchronization, global resets and broadcasts, and other mechanisms. The decidability and complexity of their verification problems is well understood [1,2,8,10,24].

Transition guards of VAS-based replicated systems are *local*: Whether a transition is enabled or not depends only on the current states of a *fixed* number of processes, independent of the total number of processes. Konnov *et al.*

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme under grant agreement No. 787367 (PaVeS).

observed in [15] that local guards cannot model fault-tolerant distributed algorithms. Indeed, in such algorithms often a process can only make a step if it has received a message from a *majority* or some fraction of the processes. To remedy this, they introduced *threshold automata*, a model of replicated systems with shared-variable communication and *threshold guards*, in which the value of a global variable is compared to an affine combination of the total numbers of processes of different types. In a number of papers, Konnov *et al.* have developed and implemented verification algorithms for safety and liveness of threshold automata [14–18]. Further, Kukovec *et al.* have obtained decidability and undecidability results [19] for different variants of the model. However, contrary to the VAS case, the computational complexity of the main verification problems has not yet been studied.

We conduct the first systematic complexity analysis of threshold automata.[1] In the first part of the paper we show that the parameterized coverability and reachability problems are NP-complete. Parameterized coverability asks if some configuration reachable from some initial configuration puts at least one process in a given state, and parameterized reachability asks if it puts processes in *exactly* a given set of states, leaving all other states unpopulated. The NP upper bound is a consequence of our main result, showing that the reachability relation of threshold automata is expressible in existential Presburger arithmetic. In the second part of the paper we apply this expressibility result to prove that the model checking problem of *Fault-Tolerant Temporal Logic* ($\mathsf{ELTL_{FT}}$) [18] is NP-complete, and that the problem of synthesizing the guards of a given automaton, studied in [21], is Σ_p^2 complete. The last part of the paper reports on an implementation of our novel approach to the parameterized (safety and liveness) verification problems. We show that it compares favorably to ByMC, the tool developed in [17].

2 Threshold Automata

We introduce threshold automata, illustrating the definitions on the example of Fig. 2, a model of the Byzantine agreement protocol of Fig. 1.

Environments. Threshold automata are defined relative to an *environment* $Env = (\Pi, RC, N)$, where Π is a set of *parameters* ranging over $\mathbb{N}_0$, $RC \subseteq \mathbb{N}_0^{\Pi}$ is a *resilience condition* expressible as an integer linear formula, and $N \colon RC \to \mathbb{N}_0$ is a linear function. Intuitively, a valuation of Π determines the number of processes of different kinds (e.g., faulty) executing the protocol, and RC describes the admissible combinations of parameter values. Finally, N associates to a each admissible combination, the number of copies of the automaton that are going to run in parallel, or, equivalently, the number of processes explicitly modeled. In a Byzantine setting, faulty processes behave arbitrarily, and so we do not model

[1] A full version of this paper containing additional details and proofs can be found at https://arxiv.org/abs/2007.06248.

```
 1 var $myval_i \in \{0,1\}$
 2 var $accept_i \in \{false, true\} \leftarrow false$
 3
 4 while true do (in one atomic step)
 5   if $myval_i = 1$
 6      and not sent ECHO before
 7   then send ECHO to all
 8
 9   if received ECHO from at least
10      $t + 1$ distinct processes
11      and not sent ECHO before
12   then send ECHO to all
13
14   if received ECHO from at least
15      $n - t$ distinct processes
16   then $accept_i \leftarrow true$
17 od
```

Fig. 1. Pseudocode of a reliable broadcast protocol from [26] for a correct process i, where n and t denote the number of processes, and an upper bound on the number of faulty processes. The protocol satisfies its specification (if $myval_i = 1$ for every correct process i, then eventually $accept_j = true$ for some correct process j) if $t < n/3$.

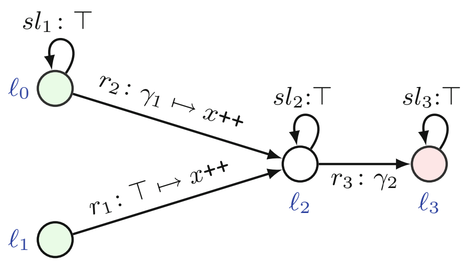

Fig. 2. Threshold automaton modeling the body of the loop in the protocol from Fig. 1. Symbols γ_1, γ_2 stand for the threshold guards $x \geq (t+1) - f$ and $x \geq (n-t) - f$, where n and t are as in Fig. 1, and f is the actual number of faulty processes. The shared variable x models the number of ECHO messages sent by correct processes. Processes with $myval_i = b$ (line 1) start in location ℓ_b (in green). Rules r_1 and r_2 model sending ECHO at lines 7 and 12. The self-loop rules $sl_1, \ldots, sl_3$ are stuttering steps. (Color figure online)

them explicitly; in this case, the system consists of one copy of the automaton for every correct process. In the crash fault model, processes behave correctly until they crash, they must be modeled explicitly, and the system has a copy of the automaton for each process, faulty or not.

Example 1. In the threshold automaton of Fig. 2, the parameters are n, f, and t, describing the number of processes, the number of faulty processes, and the maximum possible number of faulty processes, respectively. The resilience condition is the set of triples (i_n, i_f, i_t) such that $i_n/3 > i_t \geq i_f$; abusing language, we identify it with the constraint $n/3 > t \geq f$. The function N is given by $N(n,t,f) = n - f$, which is the number of correct processes.

Threshold Automata. A *threshold automaton* over an environment *Env* is a tuple $\mathsf{TA} = (\mathcal{L}, \mathcal{I}, \Gamma, \mathcal{R})$, where $\mathcal{L}$ is a nonempty, finite set of *local states* (or *locations*), $\mathcal{I} \subseteq \mathcal{L}$ is a nonempty subset of *initial locations*, Γ is a set of *global variables* ranging over $\mathbb{N}_0$, and $\mathcal{R}$ is a set of *transition rules* (or just *rules*), formally described below.

A *transition rule* (or just a *rule*) is a tuple $r = (\mathit{from}, \mathit{to}, \varphi, \boldsymbol{u})$, where *from* and *to* are the *source* and *target* locations, $\varphi \colon \Pi \cup \Gamma \to \{\mathit{true}, \mathit{false}\}$ is a conjunction of *threshold guards*, and $\boldsymbol{u} \colon \Gamma \to \{0, 1\}$ is an *update*. We often let $r.\mathit{from}, r.\mathit{to}, r.\varphi, r.\boldsymbol{u}$ denote the components of r. Intuitively, r states that a process can move from *from* to *to* if the current values of Π and Γ satisfy φ, and when it moves it updates the current valuation $\boldsymbol{g}$ of Γ by performing the update $\boldsymbol{g} := \boldsymbol{g} + \boldsymbol{u}$. Since all components of $\boldsymbol{u}$ are nonnegative, the values of global variables never decrease. A *threshold guard* φ has one of the following two forms:

- $x \geq a_0 + a_1 \cdot p_1 + \ldots + a_{|\Pi|} \cdot p_{|\Pi|}$, called a *rise guard*, or
- $x < a_0 + a_1 \cdot p_1 + \ldots + a_{|\Pi|} \cdot p_{|\Pi|}$, called a *fall guard*,

where $x \in \Gamma$ is a shared variable, $p_1, \ldots, p_{|\Pi|} \in \Pi$ are the parameters, and $a_0, a_1, \ldots, a_{|\Pi|} \in \mathbb{Q}$ are rational coefficients. Since global variables are initialized to 0, and they never decrease, once a rise (fall) guard becomes true (false) it stays true (false). We call this property *monotonicity of guards*. We let Φ^{rise}, Φ^{fall}, and Φ denote the sets of rise guards, fall guards, and all guards of TA.

Example 2. The rule r_2 of Fig. 2 has ℓ_0 and ℓ_2 as source and target locations, $x \geq (t+1) - f$ as guard, and the number 1 as update (there is only one shared variable, which is increased by one).

Configurations and Transition Relation. A *configuration* of TA is a triple $\sigma = (\boldsymbol{\kappa}, \boldsymbol{g}, \boldsymbol{p})$ where $\boldsymbol{\kappa} \colon \mathcal{L} \to \mathbb{N}_0$ describes the number of processes at each location, and $\boldsymbol{g} \in \mathbb{N}_0^{|\Gamma|}$ and $\boldsymbol{p} \in RC$ are valuations of the global variables and the parameters. In particular, $\sum_{\ell \in \mathcal{L}} \boldsymbol{\kappa}(\ell) = N(\boldsymbol{p})$ always holds. A configuration is *initial* if $\boldsymbol{\kappa}(\ell) = 0$ for every $\ell \notin \mathcal{I}$, and $\boldsymbol{g} = \boldsymbol{0}$. We often let $\sigma.\boldsymbol{\kappa}, \sigma.\boldsymbol{g}, \sigma.\boldsymbol{p}$ denote the components of σ.

A configuration $\sigma = (\boldsymbol{\kappa}, \boldsymbol{g}, \boldsymbol{p})$ *enables* a rule $r = (\mathit{from}, \mathit{to}, \varphi, \boldsymbol{u})$ if $\boldsymbol{\kappa}(\mathit{from}) > 0$, and $(\boldsymbol{g}, \boldsymbol{p})$ satisfies the guard φ, i.e., substituting $\boldsymbol{g}(x)$ for x and $\boldsymbol{p}(p_i)$ for p_i in φ yields a true expression, denoted by $\sigma \models \varphi$. If σ enables r, then TA can *move* from σ to the configuration $r(\sigma) = (\boldsymbol{\kappa}', \boldsymbol{g}', \boldsymbol{p}')$ defined as follows: (i) $\boldsymbol{p}' = \boldsymbol{p}$, (ii) $\boldsymbol{g}' = \boldsymbol{g} + \boldsymbol{u}$, and (iii) $\boldsymbol{\kappa}' = \boldsymbol{\kappa} + \boldsymbol{v}_r$, where $\boldsymbol{v}_r(\mathit{from}) = -1$, $\boldsymbol{v}_r(\mathit{to}) = +1$, and $\boldsymbol{v}_r = 0$ otherwise. We let $\sigma \to r(\sigma)$ denote that TA can move from σ to $r(\sigma)$.

Schedules and Paths. A *schedule* is a (finite or infinite) sequence of rules. A schedule $\tau = r_1, \ldots, r_m$ is *applicable* to configuration σ_0 if there is a sequence of configurations $\sigma_1, \ldots, \sigma_m$ such that $\sigma_i = r_i(\sigma_{i-1})$ for $1 \leq i \leq m$, and we define $\tau(\sigma_0) := \sigma_m$. We let $\sigma \xrightarrow{*} \sigma'$ denote that $\tau(\sigma) = \sigma'$ for some schedule τ, and say that σ' is *reachable* from σ. Further we let $\tau \cdot \tau'$ denote the concatenation of two schedules τ and τ', and, given $\mu \geq 0$, let $\mu \cdot \tau$ the concatenation of τ with itself μ times.

A *path* or *run* is a finite or infinite sequence $\sigma_0, r_1, \sigma_1, \ldots, \sigma_{k-1}, r_k, \sigma_k, \ldots$ of alternating configurations and rules such that $\sigma_i = r_i(\sigma_{i-1})$ for every r_i in the sequence. If $\tau = r_1, \ldots, r_{|\tau|}$ is applicable to σ_0, then we let $\mathsf{path}(\sigma_0, \tau)$ denote the path $\sigma_0, r_1, \sigma_1, \ldots, r_{|\tau|}, \sigma_{|\tau|}$ with $\sigma_i = r_i(\sigma_{i-1})$, for $1 \leq i \leq |\tau|$. Similarly, if τ is an infinite schedule. Given a path $\mathsf{path}(\sigma, \tau)$, the set of all configurations in the path is denoted by $\mathsf{Cfgs}(\sigma, \tau)$.

3 Coverability and Parameterized Coverability

We say that configuration σ *covers* location ℓ if $\sigma.\boldsymbol{\kappa}(\ell) > 0$. We consider the following two *coverability* questions in threshold automata:

Definition 1 ((Parameterized) coverability). *The* coverability problem *consists of deciding, given a threshold automaton TA, a location ℓ and an initial configuration σ_0, if some configuration reachable from σ_0 covers ℓ. The* parameterized coverability problem *consists of deciding, given TA and ℓ, if there is an initial configuration σ_0 and a configuration reachable from σ_0 that covers ℓ.*

Sometimes we also speak of the *non-parameterized coverability* problem, instead of the coverability problem, to avoid confusion. We show that both problems are NP-hard, even when the underlying threshold automaton is acyclic. In the next section, we show that the reachability and parameterized reachability problems (which subsume the corresponding coverability problems) are both in NP.

Theorem 1. *Parameterized coverability in threshold automata is NP-hard, even for acyclic threshold automata with only constant guards (i.e., guards of the form $x \geq a_0$ and $x < a_0$).*

Proof. (Sketch.) We prove NP-hardness of parameterized coverability by a reduction from 3-SAT. The reduction is as follows: (See Fig. 3 for an illustrative example). Let φ be a 3-CNF formula with variables $x_1, \ldots, x_n$. For every variable x_i we will have two shared variables y_i and $\bar{y_i}$. For every clause C_j, we will have a shared variable c_j. Intuitively, each process begins at some state ℓ_i and then moves to either $\top_i$ or $\bot_i$ by firing either $(\ell_i, \top_i, \bar{y_i} < 1, y_i\texttt{++})$ or $(\ell_i, \bot_i, y_i < 1, \bar{y_i}\texttt{++})$ respectively. Moving to $\top_i$ ($\bot_i$ resp.) means that the process has guessed the value of the variable x_i to be true (false resp). Once it has chosen a truth value, it then increments the variables corresponding to all the clauses which it satisfies and moves to a location ℓ_{mid}. If it happens that all the guesses were correct, a final rule gets unlocked and processes can move from ℓ_{mid} to ℓ_F. The key property we need to show is that if some process moves to $\top_i$ then no other process can move to $\bot_i$ (and vice versa). This is indeed the case because if a process moves to $\top_i$ from ℓ_i, it would have fired the rule $(\ell_i, \top_i, \bar{y_i} < 1, y_i\texttt{++})$ which increments the shared variable y_i, and so falsifies the guard of the corresponding rule $(\ell_i, \bot_i, y_i < 1, \bar{y_i}\texttt{++})$, and therefore no process can fire it. Similarly, if $(\ell_i, \bot_i, y_i < 1, \bar{y_i}\texttt{++})$ is fired, no process can fire $(\ell_i, \top_i, \bar{y_i} < 1, y_i\texttt{++})$.

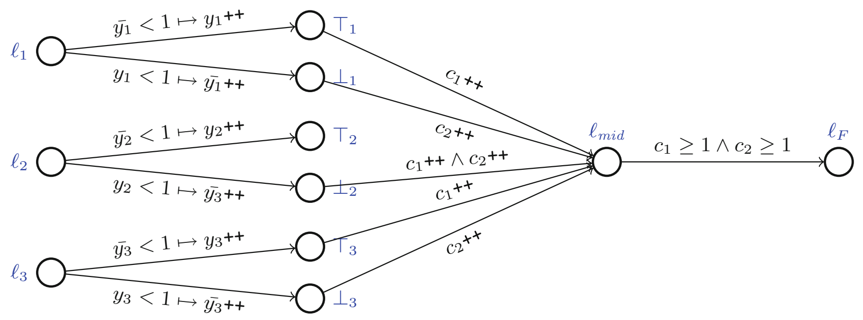
ℓ_1
$\bar{y_1} < 1 \mapsto y_1$++
$y_1 < 1 \mapsto \bar{y_1}$++
ℓ_2
$\bar{y_2} < 1 \mapsto y_2$++
$y_2 < 1 \mapsto \bar{y_3}$++
ℓ_3
$\bar{y_3} < 1 \mapsto y_3$++
$y_3 < 1 \mapsto \bar{y_3}$++
$\top_1$
$\bot_1$
$\top_2$
$\bot_2$
$\top_3$
$\bot_3$
c_1++
c_2++
c_1++ $\wedge$ c_2++
c_1++
c_2++
ℓ_{mid}
$c_1 \geq 1 \wedge c_2 \geq 1$
ℓ_F

Fig. 3. Threshold automaton TA_φ corresponding to the formula $\varphi = (x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3)$. Note that setting x_1 to true and x_2 and x_3 to false satisfies φ. Let σ_0 be the initial configuration obtained by having 1 process in each initial location ℓ_i, $1 \leq i \leq 3$, and 0 in every other location. From ℓ_1 we increment y_1 and from ℓ_2 and ℓ_3 we increment $\bar{y_2}$ and $\bar{y_3}$ respectively, thereby making the processes go to $\top_1, \bot_2, \bot_3$ respectively. From there we can move all the processes to ℓ_{mid}, at which point the last transition gets unlocked and we can cover ℓ_F.

A modification of the same construction proves

Theorem 2. *The coverability problem is NP-hard even for acyclic threshold automata with only constant rise guards (i.e., guards of the form $x \geq a_0$).*

Constant Rise Guards. Theorem 2 puts strong constraints to the class of TAs for which parameterized coverability can be polynomial, assuming P $\neq$ NP. We identify an interesting polynomial case.

Definition 2. *An environment $Env = (\Pi, RC, N)$ is* multiplicative *for a TA if for every $\mu \in \mathbb{N}_{>0}$ (i) for every valuation $\boldsymbol{p} \in RC$ we have $\mu \cdot \boldsymbol{p} \in RC$ and $N(\mu \cdot \boldsymbol{p}) = \mu \cdot N(\boldsymbol{p})$, and (ii) for every guard $\varphi := x \,\square\, a_0 + a_1 p_1 + a_2 p_2 + \ldots a_k p_k$ in TA (where $\square \in \{\geq, <\}$), if $(y, q_1, q_2, \ldots, q_k)$ is a (rational) solution to φ then $(\mu \cdot y, \mu \cdot q_1, \ldots, \mu \cdot q_k)$ is also a solution to φ.*

Multiplicativity is a very mild condition. To the best of our knowledge, all algorithms discussed in the literature, and all benchmarks of [18], have multiplicative environments. For example, in Fig. 2, if the resilience condition $t < n/3$ holds for a pair (n, t), then it also holds for $(\mu \cdot n, \mu \cdot t)$; similarly, the function $N(n, t, f) = n - f$ also satisfies $N(\mu \cdot n, \mu \cdot t, \mu \cdot f) = \mu \cdot n - \mu \cdot f = \mu \cdot N(n, t, f)$. Moreover, if $x \geq t + 1 - f$ holds in σ, then we also have $\mu \cdot x \geq \mu \cdot t + 1 - \mu \cdot f$ in $\mu \cdot \sigma$. Similarly for the other guard $x \geq n - t - f$.

This property allows us to reason about multiplied paths in large systems. Namely, condition (ii) from Definition 2 yields that if a rule is enabled in σ, it is also enabled in $\mu \cdot \sigma$. This plays a crucial role in Sect. 5 where we need the fact that a counterexample in a small system implies a counterexample in a large system.

Theorem 3. *Parameterized coverability of threshold automata over multiplicative environments with only constant rise guards is P-complete.*

Proof. (Sketch.) P-hardness is proved by giving a logspace-reduction from the Circuit Value problem ([20]) which is well known to be P-hard. In the following, we sketch the proof of inclusion in P.

Let $\mathsf{TA} = (\mathcal{L}, \mathcal{I}, \Gamma, \mathcal{R})$ be a threshold automaton over a multiplicative environment $Env = (\Pi, RC, N)$ such that the guard of each transition in $\mathcal{R}$ is a constant rise guard. We construct the set $\widehat{\mathcal{L}}$ of locations that can be reached by at least one process, and the set of transitions $\widehat{\mathcal{R}}$ that can occur, from at least one initial configuration. We initialize two variables $X_{\mathcal{L}}$ and $X_{\mathcal{R}}$ by $X_{\mathcal{L}} := \mathcal{I}$ and $X_{\mathcal{R}} := \emptyset$, and repeatedly update them until a fixed point is reached, as follows:

- If there exists a rule $r = (\ell, \ell', true, \boldsymbol{u}) \in \mathcal{R} \setminus X_{\mathcal{R}}$ such that $\ell \in X_{\mathcal{L}}$, then set $X_{\mathcal{L}} := X_{\mathcal{L}} \cup \{\ell'\}$ and $X_{\mathcal{R}} := X_{\mathcal{R}} \cup \{r\}$.
- If there exists a rule $r = (\ell, \ell', (\wedge_{1 \le i \le q} x_i \ge c_i), \boldsymbol{u}) \in \mathcal{R} \setminus X_{\mathcal{R}}$ such that $\ell \in X_{\mathcal{L}}$, and there exists rules $r_1, r_2, \ldots, r_q$ such that each $r_i = (\ell_i, \ell'_i, \varphi_i, \boldsymbol{u}_i) \in X_{\mathcal{R}}$ and $\boldsymbol{u}_i[x_i] > 0$, then set $X_{\mathcal{L}} := X_{\mathcal{L}} \cup \{\ell'\}$ and $X_{\mathcal{R}} := X_{\mathcal{R}} \cup \{r\}$.

In the full version of the paper, we prove that after termination $X_{\mathcal{L}} = \widehat{\mathcal{L}}$ holds. Intuitively, multiplicativity ensures that if a reachable configuration enables a rule, there are reachable configurations from which the rule can occur arbitrarily many times. This shows that any path of rules constructed by the algorithm is executable.

4 Reachability

We now consider reachability problems for threshold automata. Formally, we consider the following two versions of the reachability problem:

Definition 3 ((Parameterized) reachability). *The* reachability problem *consists of deciding, given a threshold automaton TA, two sets $\mathcal{L}_{=0}, \mathcal{L}_{>0}$ of locations, and an initial configuration σ_0, if some configuration σ reachable from σ_0 satisfies $\sigma.\boldsymbol{\kappa}(\ell) = 0$ for every $\ell \in \mathcal{L}_{=0}$ and $\sigma.\boldsymbol{\kappa}(\ell) > 0$ for every $\ell \in \mathcal{L}_{>0}$. The* parameterized reachability problem *consists of deciding, given TA and $\mathcal{L}_{=0}, \mathcal{L}_{>0}$, if there is an initial configuration σ_0 such that some σ reachable from σ_0 satisfies $\sigma.\boldsymbol{\kappa}(\ell) = 0$ for every $\ell \in \mathcal{L}_{=0}$ and $\sigma.\boldsymbol{\kappa}(\ell) > 0$ for every $\ell \in \mathcal{L}_{>0}$.*

Notice that the reachability problem clearly subsumes the coverability problem and hence, in the sequel, we will only be concerned with proving that both problems are in NP. This will be a consequence of our main result, showing that the reachability relation of threshold automata can be characterized as an existential formula of Presburger arithmetic. This result has several other consequences. In Sect. 5 we use it to give a new model checking algorithm for the fault-tolerant logic of [18]. In Sect. 7 we report on an implementation whose runtime compares favorably with previous tools.

Reachability Relation as an Existential Presburger Formula. Fix a threshold automaton $\mathsf{TA} = (\mathcal{L}, \mathcal{I}, \Gamma, \mathcal{R})$ over an environment *Env*. We construct an existential Presburger arithmetic formula ϕ_{reach} with $(2|\mathcal{L}|+2|\Gamma|+2|\Pi|)$ free variables such that $\phi_{reach}(\sigma, \sigma')$ is true iff σ' is reachable from σ.

Let the *context* of a configuration σ, denoted by $\omega(\sigma)$, be the set of all *rise* guards that evaluate to true and all *fall* guards that evaluate to false in σ. Given a schedule τ, we say that the path $\mathsf{path}(\sigma, \tau)$ is *steady* if all the configurations it visits have the same context. By the monotonicity of the guards of threshold automata, $\mathsf{path}(\sigma, \tau)$ is steady iff its endpoints have the same context, i.e., iff $\omega(\sigma) = \omega(\tau(\sigma))$. We have the following proposition:

Proposition 1. *Every path of a threshold automaton with k guards is the concatenation of at most $k+1$ steady paths.*

Using this proposition, we first construct a formula ϕ_{steady} such that $\phi_{steady}(\sigma, \sigma')$ holds iff there is a *steady* path $\mathsf{path}(\sigma, \tau)$ such that $\tau(\sigma) = \sigma'$.

The Formula $\boldsymbol{\phi_{steady}}$. For every rule $r \in \mathcal{R}$, let x_r be a variable ranging over non-negative integers. Intuitively, the value of x_r will represent the number of times r is fired during the (supposed) path from σ to σ'. Let $X = \{x_r\}_{r \in \mathcal{R}}$. We construct ϕ_{steady} step by step, specifying necessary conditions for σ, σ' and X to satisfy the existence of the steady path, which in particular implies that σ' is reachable from σ.

Step 1. σ and σ' must have the same values of the parameters, which must satisfy the resilience condition, the same number of processes, and the same context:

$$\phi_{base}(\sigma, \sigma') \equiv \ \sigma.\boldsymbol{p} = \sigma'.\boldsymbol{p} \ \wedge \ RC(\sigma.\boldsymbol{p}) \ \wedge \ N(\sigma.\boldsymbol{p}) = N(\sigma'.\boldsymbol{p}) \ \wedge \ \omega(\sigma) = \omega(\sigma').$$

Step 2. For a location $\ell \in \mathcal{L}$, let $out_1^\ell, \ldots, out_{a_\ell}^\ell$ be all outgoing rules from ℓ and let $in_1^\ell, \ldots, in_{b_\ell}^\ell$ be all incoming rules to ℓ. The number of processes in ℓ after the execution of the path is the initial number, plus the incoming processes, minus the outgoing processes. Since x_r models the number of times the rule r is fired, we have

$$\phi_{\mathcal{L}}(\sigma, \sigma', X) \equiv \quad \bigwedge_{\ell \in \mathcal{L}} \left(\sum_{i=1}^{a_\ell} x_{in_i^\ell} - \sum_{j=1}^{b_\ell} x_{out_j^\ell} = \sigma'.\boldsymbol{\kappa}(\ell) - \sigma.\boldsymbol{\kappa}(\ell) \right)$$

Step 3. Similarly, for the shared variables we must have:

$$\phi_{\Gamma}(\sigma, \sigma', X) \equiv \quad \bigwedge_{z \in \Gamma} \left(\sum_{r \in \mathcal{R}} (x_r \cdot r.\boldsymbol{u}[z]) = \sigma'.\boldsymbol{g}[z] - \sigma.\boldsymbol{g}[z] \right)$$

Step 4. Since $\mathsf{path}(\sigma, \tau)$ must be steady, if a rule is fired along $\mathsf{path}(\sigma, \tau)$ then its guard must be true in σ and so

$$\phi_{\mathcal{R}}(\sigma, X) \equiv \quad \bigwedge_{r \in \mathcal{R}} x_r > 0 \ \Rightarrow \ (\sigma \models r.\varphi)$$

Step 5. Finally, for every rule r that occurs in $\mathsf{path}(\sigma, \tau)$, the path must contain a "fireable" chain leading to r, i.e., a set of rules $S = \{r_1, \ldots, r_s\} \subseteq \mathcal{R}$ such that all rules of S are executed in $\mathsf{path}(\sigma, \tau)$, there is a process in σ at $r_1.\mathit{from}$, and the rules $r_1, \ldots, r_s$ form a chain leading from $r_1.\mathit{from}$ to $r.\mathit{from}$. We capture this by the constraint

$$\phi_{appl}(\sigma, X) \equiv \bigwedge_{r \in \mathcal{R}} \left(x_r > 0 \;\Rightarrow \bigvee_{S=\{r_1,r_2,\ldots,r_s\} \subseteq \mathcal{R}} \phi^r_{chain}(S, \sigma, X) \right)$$

where

$$\phi^r_{chain}(S, \sigma, X) \equiv \bigwedge_{r \in S} x_r > 0 \;\wedge\; \sigma.\boldsymbol{\kappa}(r_1.\mathit{from}) > 0 \;\wedge \bigwedge_{1 < i \leq s} r_{i-1}.\mathit{to} = r_i.\mathit{from} \;\wedge\; r_s = r$$

Combining the Steps. Define $\phi_{steady}(\sigma, \sigma')$ as follows:

$$\begin{aligned}\phi_{steady}(\sigma, \sigma') \equiv \;& \phi_{base}(\sigma, \sigma') \;\wedge \\ & \exists X \geq 0.\; \phi_{\mathcal{L}}(\sigma, \sigma', X) \wedge \phi_{\Gamma}(\sigma, \sigma', X) \wedge \phi_{\mathcal{R}}(\sigma, X) \wedge \phi_{appl}(\sigma, X) \;.\end{aligned}$$

where $\exists X \geq 0$ abbreviates $\exists x_{r_1} \geq 0, \ldots, \exists x_{r_{|\mathcal{R}|}} \geq 0$. By our discussion, it is clear that if there is a steady path leading from σ to σ', then $\phi_{steady}(\sigma, \sigma')$ is satisfiable. The following theorem proves the converse.

Theorem 4. *Let* TA *be a threshold automaton and let* $\sigma, \sigma' \in \Sigma$ *be two configurations. Formula* $\phi_{steady}(\sigma, \sigma')$ *is satisfiable if and only if there is a steady schedule* τ *with* $\tau(\sigma) = \sigma'$.

Observe that, while ϕ_{steady} has exponential length in TA when constructed naïvely (because of the exponentially many disjunctions in ϕ_{appl}), its satisfiability is in NP. Indeed, we first non-deterministically guess one of the disjunctions for each conjunction of ϕ_{appl} and then check in nondeterministic polynomial time that the (polynomial sized) formula with only these disjuncts is satisfiable. This is possible because existential Presburger arithmetic is known to be in NP [13].

The Formula $\boldsymbol{\phi_{reach}}$. By Proposition 1, every path from σ to σ' in a threshold automaton with a set Φ of guards can be written in the form

$$\sigma = \sigma_0 \xrightarrow{*} \sigma_0' \to \sigma_1 \xrightarrow{*} \sigma_1' \to \sigma_2 \ldots \sigma_K \xrightarrow{*} \sigma_K' = \sigma'$$

where $K = |\Phi| + 1$, and $\sigma_i \xrightarrow{*} \sigma_i'$ is a steady path for each $0 \leq i \leq K$. It is easy to see from the definition of the transition relation between configurations that we can construct a polynomial sized existential Presburger formula ϕ_{step} such that $\phi_{step}(\sigma, \sigma')$ is true iff σ' can be reached from σ by firing at most one rule. Thus, we define $\phi_{reach}(\sigma, \sigma')$ to be

$$\exists \sigma_0, \sigma_0', \ldots, \sigma_K, \sigma_K' \left(\sigma_0 = \sigma \wedge \sigma_K' = \sigma' \wedge \bigwedge_{0 \leq i \leq K} \phi_{steady}(\sigma_i, \sigma_i') \wedge \bigwedge_{0 \leq i \leq K-1} \phi_{step}(\sigma_i', \sigma_{i+1}) \right)$$

Theorem 5. *Given a threshold automaton TA, there is an existential Presburger formula ϕ_{reach} such that $\phi_{reach}(\sigma, \sigma')$ holds iff $\sigma \xrightarrow{*} \sigma'$.*

As deciding the truth of existential Presburger formulas is in NP, we obtain:

Corollary 1. *The reachability and parameterized reachability problems are in NP.*

Remark 1. In [14] an algorithm was given for parameterized reachability of threshold automata in which the updates of all rules contained in loops are equal to $\mathbf{0}$. Our algorithm does not need this restriction.

5 Safety and Liveness

We recall the definition of *Fault-Tolerant Temporal Logic* ($\mathsf{ELTL_{FT}}$), the fragment of LTL used in [18] to specify and verify properties of a large number of fault-tolerant algorithms. $\mathsf{ELTL_{FT}}$ has the following syntax, where $S \subseteq \mathcal{L}$ is a set of locations and $guard \in \Phi$ is a guard:

$$\psi ::= pf \mid \mathbf{G}\,\psi \mid \mathbf{F}\,\psi \mid \psi \wedge \psi \qquad cf ::= S = 0 \mid \neg(S = 0) \mid cf \wedge cf$$
$$pf ::= cf \mid gf \Rightarrow cf \qquad gf ::= guard \mid gf \wedge gf \mid gf \vee gf$$

An infinite path $\mathsf{path}(\sigma, \tau)$ starting at $\sigma = (\boldsymbol{\kappa}, \boldsymbol{g}, \boldsymbol{p})$, satisfies $S = 0$ if $\boldsymbol{\kappa}(\ell) = 0$ for every $\ell \in S$, and *guard* if $(\boldsymbol{g}, \boldsymbol{p})$ satisfies *guard*. The rest of the semantics is standard. The negations of specifications of the benchmarks [3–6, 12, 22, 23, 25, 26] can be expressed in $\mathsf{ELTL_{FT}}$, as we are interested in finding possible violations.

Example 3. One specification of the algorithm from Fig. 1 is that if $myval_i = 1$ for every correct process i, then eventually $accept_j = true$ for some correct process j. In the words of the automaton from Fig. 2, a violation of this property would mean that initially all correct processes are in location ℓ_1, but no correct process ever reaches location ℓ_3. In $\mathsf{ELTL_{FT}}$ we write this as

$$\{\ell_0, \ell_2, \ell_3\} = 0 \;\wedge\; \mathbf{G}\,(\{\ell_3\} = 0)\,.$$

This has to hold under the fairness constraint

$$\mathbf{G}\,\mathbf{F}\left(\,((x \geq t+1 \vee x \geq n-t) \Rightarrow \{\ell_0\}{=}0) \;\wedge\; \{\ell_1\}{=}0 \;\wedge\; (x \geq n-t \Rightarrow \{\ell_2\}{=}0)\right).$$

As we have self-loops at locations ℓ_0 and ℓ_2, a process could stay forever in one of these two states, even if it has collected enough messages, i.e., if $x \geq t+1$ or $x \geq n-t$. This is the behavior that we want to prevent with such a fairness constraint. Enough sent messages should force each process to progress, so the location eventually becomes empty. Similarly, as the rule leading from ℓ_1 has a trivial guard, we want to make sure that all processes starting in ℓ_1 eventually (send a message and) leave ℓ_1 empty, as required by the algorithm.

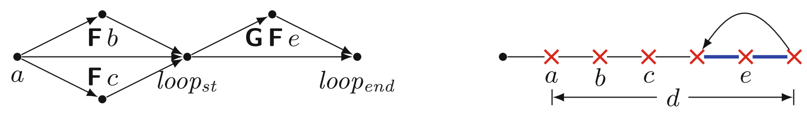

Fig. 4. The cut graph of a formula $\mathsf{F}\,(a \wedge \mathsf{F}\,b \wedge \mathsf{F}\,c \wedge \mathsf{G}\,d \wedge \mathsf{G}\,\mathsf{F}\,e)$ (left) and one lasso shape for a chosen topological ordering $a \leq \mathsf{F}\,b \leq \mathsf{F}\,c \leq loop_{st} \leq \mathsf{G}\,\mathsf{F}\,e \leq loop_{end}$ (right).

In this section we study the following problem:

Definition 4 (Parameterized safety and liveness). *Given a threshold automaton TA and a formula φ in $\mathit{ELTL}_{\mathit{FT}}$, check whether there is an initial configuration σ_0 and an infinite schedule τ applicable to σ_0 such that $\mathit{path}(\sigma_0, \tau) \models \varphi$.*

Since parameterized coverability is NP-hard, it follows that parameterized safety and liveness is also NP-hard. We prove that for automata with *multiplicative environments* (see Definition 2) parameterized safety and liveness is in NP.

Theorem 6. *Parameterized safety and liveness of threshold automata with multiplicative environments is in NP.*

The proof, which can be found in the full version, is very technical, and we only give a rough sketch here. The proof relies on two notions introduced in [18]. First, it is shown in [18] that every $\mathsf{ELTL}_{\mathsf{FT}}$ formula is equivalent to a formula in *normal form* of shape $\phi_0 \wedge \mathsf{F}\,\phi_1 \wedge \cdots \wedge \mathsf{F}\,\phi_k \wedge \mathsf{G}\,\phi_{k+1}$, where ϕ_0 is a propositional formula and $\phi_1, \ldots, \phi_{k+1}$ are themselves in normal form. Further, formulas can be put in normal form in polynomial time. The second notion introduced in [18] is the *cut graph* $Gr(\varphi)$ of a formula in normal form. For our sketch it suffices to know that $Gr(\varphi)$ is a directed acyclic graph with two special nodes $loop_{st}$ and $loop_{end}$, and every other node is a subformula of φ in normal form (see Fig. 4).

For a formula $\varphi \equiv \phi_0 \wedge \mathsf{F}\,\phi_1 \wedge \ldots \wedge \mathsf{F}\,\phi_k \wedge \mathsf{G}\,\phi_{k+1}$, we will say that its *local proposition* is ϕ_0 and its *global proposition* is the local proposition of ϕ_{k+1}. It is shown in [18] that, given $\varphi = \phi_0 \wedge \mathsf{F}\,\phi_1 \wedge \cdots \wedge \mathsf{F}\,\phi_k \wedge \mathsf{G}\,\phi_{k+1}$, some infinite path satisfies φ iff there exists a topological ordering $v_0, v_1, \ldots, v_c = loop_{st}, v_{c+1}, \ldots, v_l = loop_{end}$ of the cut graph and a path $\sigma_0, \tau_0, \sigma_1, \ldots, \sigma_c, \tau_c, \ldots, \sigma_{l-1}, \tau_{l-1}, \sigma_l$ such that, roughly speaking, (among other technical conditions) every configuration σ_i satisfies the local proposition of v_i and every configuration in $\mathsf{Cfgs}(\sigma_i, \tau_i)$ satisfies the global proposition of every v_j where $j \leq i$.

Using multiplicativity and our main result that reachability is definable in existential Presburger arithmetic, we show that for every proposition p, we can construct an existential formula $\phi_p(\sigma, \sigma')$ such that: If there is a path between σ and σ', *all* of whose configurations satisfy p, then $\phi_p(\sigma, \sigma')$ is satisfiable. Further, if $\phi_p(\sigma, \sigma')$ is satisfiable, then there is a path between $2 \cdot \sigma$ and $2 \cdot \sigma'$ *all* of whose configurations satisfy p. (Here $2 \cdot \sigma = ((2 \cdot \sigma.\boldsymbol{\kappa}), (2 \cdot \sigma.\boldsymbol{g}), (2 \cdot \sigma.\boldsymbol{p}))$). Then, once we have fixed a topological ordering $V = v_0, \ldots, v_l$, (among other conditions), we check if there are configurations $\sigma_0, \ldots, \sigma_l$ such that for every i, σ_i satisfies the local proposition of v_i and for every $j \leq i$, $\phi_{p_j}(\sigma_i, \sigma_{i+1})$ is satisfiable where p_j is the global proposition of v_j. Using multiplicativity, we then show that this procedure is sufficient to check if the given specification φ is satisfied.

Our algorithm consists therefore of the following steps: (1) bring φ in normal form; (2) construct the cut graph $Gr(\varphi)$; (3) guess a topological ordering of the nodes of $Gr(\varphi)$; (4) for the guessed ordering, check in nondeterministic polynomial time if the required sequence $\sigma_0, \ldots, \sigma_l$ exists.

Remark 2. From an algorithm given in [18] one can infer that parameterized safety and liveness is in NP for threshold automata with multiplicative environments, where all cycles are simple, and rules in cycles have update **0**. (The NP bound was not explicitly given in [18].) Our algorithm only requires multiplicativity.

6 Synthesis of Threshold Guards

We study the *bounded synthesis* problem for constructing parameterized threshold guards in threshold automata satisfying a given specification.

Sketch Threshold Automata. Let an *indeterminate* be a variable that can take values over rational numbers. We consider threshold automata whose guards can contain indeterminates. More precisely, a *sketch* threshold automaton is a tuple $\mathsf{TA} = (\mathcal{L}, \mathcal{I}, \Gamma, \mathcal{R})$, just as before, except for the following change. Recall that in a threshold automaton, a guard is an inequality of one of the following two forms:

$$x \geq a_0 + a_1 \cdot p_1 + \ldots + a_{|\Pi|} \cdot p_{|\Pi|} \quad \text{or} \quad x < a_0 + a_1 \cdot p_1 + \ldots + a_{|\Pi|} \cdot p_{|\Pi|}$$

where $a_0, a_1, \ldots, a_{|\Pi|}$ are rational numbers. In a sketch threshold automaton, some of the $a_0, a_1, \ldots, a_{|\Pi|}$ can be *indeterminates.* Moreover, indeterminates can be shared between two or more guards.

Given a sketch threshold automaton TA and an assignment μ to the indeterminates, we let $\mathsf{TA}[\mu]$ denote the threshold automaton obtained by substituting the indeterminates by their values in μ. We define the *bounded synthesis* problem:

> Given: An environment Env, a sketch threshold automaton TA with indeterminates $v_1, \ldots, v_m$, a formula φ of $\mathsf{ELTL}_{\mathsf{FT}}$, and a polynomial p.
> Decide: Is there an assignment μ to $v_1, \ldots, v_m$ of size $O(p(|\mathsf{TA}| + |\varphi|))$ (i.e., the vector $(\mu(v_1), \ldots, \mu(v_m))$ of rational numbers can be encoded in binary using $O(p(|\mathsf{TA}| + |\varphi|))$ bits) such that $\mathsf{TA}[\mu]$ satisfies $\neg\varphi$ (i.e., such that for every initial configuration σ_0 in $\mathsf{TA}[\mu]$, every infinite run starting from σ_0 satisfies $\neg\varphi$)?

We say that an assignment μ to the indeterminates makes the environment multiplicative if the conditions of Definition 2 are satisfied after plugging in the assignment μ in the automaton. In the following, we will only be concerned with assignments which make the environment multiplicative.

Since we can guess an assignment in polynomial time, by Theorem 6 it follows

Theorem 7. *Bounded synthesis is in Σ_2^p.*

By a reduction from the Σ_2-SAT problem, we also provide a matching lower bound.

Theorem 8. *Bounded synthesis is Σ_2^p-complete.*

The *synthesis* problem is defined as the bounded synthesis problem, but lifting the constraint on the size of μ. While we do not know the exact complexity of the synthesis problem, we can show that, for a large and practically motivated class of threshold automata introduced in [21], the synthesis problem reduces to the bounded synthesis problem. We proceed to describe and motivate the class.

The parameter variables of fault-tolerant distributed algorithms usually consist of a variable n denoting the number of processes running the algorithm and various "failure" variables for the number of processes exhibiting different kinds of failures (for example, a variable t_1 might be used to specify the number of Byzantine failures, a variable t_2 for crash failures, etc.). The following three observations are made in [21]:

(1) The resilience condition of these algorithms is usually of the form $n > \sum_{i=1}^{k} \delta_i t_i$ where t_i are parameter variables and δ_i are natural numbers.
(2) Threshold guards typically serve one of two purposes: to check if at least a certain fraction of the processes sends a message (for example, $x > n/2$ ensures that a strict majority of processes has sent a message), or to bound the number of processes that crash.
(3) The coefficients of the guards are rational numbers with small denominators (typically at most 3).

By (2), the structure of the algorithm guarantees that the value of a variable x never goes beyond n, the number of processes. Therefore, given a threshold guard template $x \bowtie \boldsymbol{u} \cdot \boldsymbol{\pi} + v$, where $\boldsymbol{u}$ is a vector of indeterminates, $\boldsymbol{\pi}$ is a vector of parameter variables, v is an indeterminate, and $\bowtie$ is either $\geq$ or $<$, we are only interested in assignments μ of $\boldsymbol{u}$ and v which satisfy $0 \leq \mu(\boldsymbol{u}) \cdot \nu(\boldsymbol{\pi}) + \mu(v) \leq n$ for every valuation $\nu(\boldsymbol{\pi})$ of $\boldsymbol{\pi}$ respecting the resilience condition. Guards obtained by instantiating guard templates with such a valuation μ are called *sane guards* [21].

The following result is proved in [21]: Given a resilience condition $n > \sum_{i=1}^{k} \delta_i t_i$, and an upper bound D on the denominator of the entries of μ (see (1) and (3) above), the numerators of the entries of μ are necessarily of polynomial size in $k, \delta_1, \ldots, \delta_k$. Therefore, the synthesis problem for sane guards and bounded denominator, as introduced in [21], reduces to the bounded synthesis problem, and so it can be solved in Σ_2^p time. Moreover, the reduction used in Theorem 8 to prove Σ_2^p-hardness yields sketch threshold automata with sane guards, and so the the synthesis problem for sane guards and bounded denominator is also Σ_2^p-complete.

7 Experimental Evaluation

Following the techniques presented in this paper, we have verified a number of threshold-based fault-tolerant distributed algorithms.

Table 1. The experiments were run on a machine with Intel® Core™ i5-7200U CPU with 7.7 GiB memory. The time limit was set to be 5 h and the memory limit was set to be 7 GiB. TLE (MLE) means that the time limit (memory limit) exceeded for the particular benchmark.

Input	Case (if more than one)	Threshold automaton		Time, seconds	
		$\lvert\mathcal{L}\rvert$	$\lvert\mathcal{R}\rvert$	Our tool	ByMC
nbacg		24	64	11.84	10.29
nbacr		77	1031	490.79	1081.07
aba	Case 1	37	202	251.71	751.89
aba	Case 2	61	425	2856.63	TLE
cbc	Case 1	164	2064	MLE	MLE
cbc	Case 2	73	470	2521.12	36.57
cbc	Case 3	304	6928	MLE	MLE
cbc	Case 4	161	2105	MLE	MLE
cf1s	Case 1	41	280	50.5	55.87
cf1s	Case 2	41	280	55.88	281.69
cf1s	Case 3	68	696	266.56	7939.07
cf1s	hand-coded TA	9	26	7.17	2737.53
c1cs	Case 1	101	1285	1428.51	TLE
c1cs	Case 2	70	650	1709.4	11169.24
c1cs	Case 3	101	1333	TLE	MLE
c1cs	hand-coded TA	9	30	37.72	TLE
bosco	Case 1	28	152	58.11	89.64
bosco	Case 2	40	242	157.61	942.87
bosco	Case 3	32	188	59	104.03
bosco	hand-coded TA	8	20	20.95	510.32

Benchmarks. Consistent broadcast (**strb**) [26] is given in Fig. 1 and its threshold automaton is depicted in Fig. 2. The algorithm is correct if in any execution either all correct processes or none set accept to true; moreover, if all correct processes start with value 0 then none of them accept, and if all correct processes start with value 1 then they all accept. The algorithm is designed to tolerate Byzantine failures of less than one third of processes, that is, if $n > 3t$. Folklore Reliable Broadcast (**frb**) [5] that tolerates crash faults and Asynchronous Byzantine agreement (**aba**) [3] satisfy the same specifications as consistent broadcast, under the same resilience condition.

Non-blocking atomic commit (**nbacr**) [23] and (**nbacg**) [12] deal with faults using failure detectors. We model this by introducing a special location such that a process is in it if and only if it suspects that there is a failure of the system.

Condition-based consensus (**cbc**) [22] reaches consensus under the condition that the difference between the numbers of processes initialized with 0 and 1 differ by at least t, an upper bound on the number of faults. We also check algorithms that allow consensus to be achieved in one communication step, such as **cfcs** [6], **c1cs** [4], as well as Byzantine One Step Consensus **bosco** [25].

Evaluation. Table 1 summarizes our results and compares them with the results obtained using the ByMC tool [17]. Due to lack of space, we have omitted those experiments for which both ByMC and our tool took less than 10 s.

We implemented our algorithms in Python and used Z3 as a back-end SMT solver for solving the constraints over existential Presburger arithmetic. Our implementation takes as input a threshold automaton and a specification in $\mathsf{ELTL}_{\mathsf{FT}}$ and checks if a counterexample exists. We apply to the latest version of the benchmarks of [17]. Each benchmark yields two threshold automata, a hand-coded one and one obtained by a data abstraction of the algorithm written in Parametric Promela. For automata of the latter kind, due to data abstraction, we have to consider different cases for the same algorithm. We test each automaton against all specifications for that automaton.

Our tool outperforms ByMC in all automata with more than 30 states, with the exception of the second case of cbc. It performs worse in most small cases, however in these cases, both ByMC and our tool take less than 10 s. ByMC works by enumerating all so-called *schemas* of a threshold automaton, and solving a SMT problem for each of them; the number of schemas can grow exponentially in the number of guards. Our tool avoids the enumeration. Since the number of schemas for the second case of cbc is just 2, while the second case of aba and third case of cf1s have more than 3000, avoiding the enumeration seems to be key to our better performance.

References

1. Bloem, R., et al.: Decidability of Parameterized Verification. Synthesis Lectures on Distributed Computing Theory. Morgan & Claypool Publishers, San Rafael (2015)
2. Blondin, M., Haase, C., Mazowiecki, F.: Affine extensions of integer vector addition systems with states. In: CONCUR. LIPIcs, vol. 118, pp. 14:1–14:17. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2018)
3. Bracha, G., Toueg, S.: Asynchronous consensus and broadcast protocols. J. ACM **32**(4), 824–840 (1985)
4. Brasileiro, F., Greve, F., Mostefaoui, A., Raynal, M.: Consensus in one communication step. In: Malyshkin, V. (ed.) PaCT 2001. LNCS, vol. 2127, pp. 42–50. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44743-1_4
5. Chandra, T.D., Toueg, S.: Unreliable failure detectors for reliable distributed systems. J. ACM **43**(2), 225–267 (1996)
6. Dobre, D., Suri, N.: One-step consensus with zero-degradation. In: DSN, pp. 137–146 (2006)
7. Dufourd, C., Finkel, A., Schnoebelen, P.: Reset nets between decidability and undecidability. In: Larsen, K.G., Skyum, S., Winskel, G. (eds.) ICALP 1998. LNCS, vol. 1443, pp. 103–115. Springer, Heidelberg (1998). https://doi.org/10.1007/BFb0055044
8. Esparza, J.: Decidability and complexity of Petri net problems—An introduction. In: Reisig, W., Rozenberg, G. (eds.) ACPN 1996. LNCS, vol. 1491, pp. 374–428. Springer, Heidelberg (1998). https://doi.org/10.1007/3-540-65306-6_20
9. Esparza, J., Finkel, A., Mayr, R.: On the verification of broadcast protocols. In: LICS, pp. 352–359. IEEE Computer Society (1999)
10. Esparza, J., Nielsen, M.: Decidability issues for petri nets - a survey. Bull. EATCS **52**, 244–262 (1994)
11. German, S.M., Sistla, A.P.: Reasoning about systems with many processes. J. ACM **39**(3), 675–735 (1992)
12. Guerraoui, R.: Non-blocking atomic commit in asynchronous distributed systems with failure detectors. Distrib. Comput. **15**(1), 17–25 (2002). https://doi.org/10.1007/s446-002-8027-4
13. Haase, C.: A survival guide to Presburger arithmetic. ACM SIGLOG News **5**(3), 67–82 (2018)
14. Konnov, I., Lazic, M., Veith, H., Widder, J.: Para^2: parameterized path reduction, acceleration, and SMT for reachability in threshold-guarded distributed algorithms. Formal Methods Syst. Des. **51**(2), 270–307 (2017)
15. Konnov, I., Veith, H., Widder, J.: On the completeness of bounded model checking for threshold-based distributed algorithms: reachability. In: Baldan, P., Gorla, D. (eds.) CONCUR 2014. LNCS, vol. 8704, pp. 125–140. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44584-6_10
16. Konnov, I., Veith, H., Widder, J.: On the completeness of bounded model checking for threshold-based distributed algorithms: reachability. Inf. Comput. **252**, 95–109 (2017)
17. Konnov, I., Widder, J.: ByMC: Byzantine model checker. In: Margaria, T., Steffen, B. (eds.) ISoLA 2018. LNCS, vol. 11246, pp. 327–342. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03424-5_22
18. Konnov, I.V., Lazic, M., Veith, H., Widder, J.: A short counterexample property for safety and liveness verification of fault-tolerant distributed algorithms. In: POPL 2017, pp. 719–734 (2017)

19. Kukovec, J., Konnov, I., Widder, J.: Reachability in parameterized systems: all flavors of threshold automata. In: CONCUR, pp. 19:1–19:17 (2018)
20. Ladner, R.E.: The circuit value problem is log space complete for p. SIGACT News **7**(1), 18–20 (1975)
21. Lazić, M., Konnov, I., Widder, J., Bloem, R.: Synthesis of distributed algorithms with parameterized threshold guards. In: OPODIS. LIPIcs, vol. 95, pp. 32:1–32:20 (2017)
22. Mostéfaoui, A., Mourgaya, E., Parvédy, P.R., Raynal, M.: Evaluating the condition-based approach to solve consensus. In: DSN, pp. 541–550 (2003)
23. Raynal, M.: A case study of agreement problems in distributed systems: non-blocking atomic commitment. In: HASE, pp. 209–214 (1997)
24. Schmitz, S., Schnoebelen, P.: The power of well-structured systems. CoRR abs/1402.2908 (2014)
25. Song, Y.J., van Renesse, R.: Bosco: one-step Byzantine asynchronous consensus. In: Taubenfeld, G. (ed.) DISC 2008. LNCS, vol. 5218, pp. 438–450. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-87779-0_30
26. Srikanth, T., Toueg, S.: Simulating authenticated broadcasts to derive simple fault-tolerant algorithms. Distrib. Comput. **2**, 80–94 (1987). https://doi.org/10.1007/BF01667080

Appendix F

Parameterized Complexity of Safety of Threshold Automata (FSTTCS 2020)

This section contains a reprinting of the following paper, which has been published as a peer-reviewed conference paper.

> A. R. Balasubramanian. *Parameterized Complexity of Safety of Threshold Automata.* In conference proceedings of FSTTCS 2020. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. Vol. 182 of LIPIcs, Pages - 37:1-37:15. doi: `10.4230/LIPIcs.FSTTCS.2020.37`

Summary

Threshold automata are a formalism for modeling fault-tolerant distributed algorithms. We study the parameterized complexity of reachability of threshold automata. As a first result, we show that the problem becomes W[1]-hard even when parameterized by parameters that are quite small in practice. We then consider two restricted cases which arise in practice and provide fixed-parameter

tractable algorithms for both these cases. Finally, we report on experimental results conducted on some protocols taken from the literature.

Contributions of the author of this thesis

I am the sole author of this paper.

Parameterized Complexity of Safety of Threshold Automata

A. R. Balasubramanian
Technische Universität München, Germany
bala.ayikudi@tum.de

Abstract

Threshold automata are a formalism for modeling fault-tolerant distributed algorithms. In this paper, we study the parameterized complexity of reachability of threshold automata. As a first result, we show that the problem becomes W[1]-hard even when parameterized by parameters which are quite small in practice. We then consider two restricted cases which arise in practice and provide fixed-parameter tractable algorithms for both these cases. Finally, we report on experimental results conducted on some protocols taken from the literature.

2012 ACM Subject Classification Theory of computation → Distributed computing models; Theory of computation → Logic and verification

Keywords and phrases Threshold automata, distributed algorithms, parameterized complexity

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2020.37

Funding This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme under grant agreement No 787367 (PaVeS).

Acknowledgements I would like to thank Prof. Javier Esparza for useful discussions regarding the paper and Christoph Welzel and Margarete Richter for their encouragement and support. I would also like to thank the anonymous reviewers whose comments greatly improved the presentation of the paper.

1 Introduction

Threshold automata [21] are a formalism for modeling and analyzing fault-tolerant distributed algorithms. In this setup, an arbitrary but fixed number of processes execute a given protocol as specified by a threshold automaton. Verification of these systems aims to prove these protocols correct for any number of processes [4].

One of the main differences between threshold automata and other formalisms for modeling distributed protocols (like replicated systems and population protocols [1, 16, 18]) is the notion of a *threshold guard*. Roughly speaking, a threshold guard specifies certain constraints that should hold between the number of messages received and the number of participating processes, in order for a transition to be enabled. For example, a guard of the form $x \geq n/2 - t$, (where x counts the number of messages of some specified type, n is the number of processes and t is the maximum number of faulty processes), specifies that the number of messages received should be bigger than $n/2 - t$, in order for the process to proceed. This feature is important in modeling fault-tolerant distributed algorithms where, often a process can make a move only if it has received a message from a majority or two-thirds of the number of processes. In a collection of papers [26, 3, 25, 24, 23], many algorithms have been developed for verifying various properties for threshold automata. These algorithms have then been used to verify a number of protocols from the distributed computing literature [24]. It is known that the reachability problem for threshold automata is NP-complete [2].

Parameterized complexity [13] attempts to study decision problems that come along with a *parameter*. In parameterized complexity, apart from the size of the input n, one considers further parameters k that capture the structure of the input and one looks for algorithms that run in time $f(k) \cdot n^{O(1)}$, where f is some function dependent on k alone. The hope is to find parameters which are quite small in practice and to base the dominant running time of the algorithm on this parameter alone. Problems solvable in such a manner are called fixed-parameter tractable (FPT).

In recent years, increasing effort has been devoted to studying the parameterized complexity of problems in verification for different models [10, 11, 15, 17, 9]. Motivated by this and by the hard theoretical complexity (NP-completeness) of reachability of threshold automata, we study the parameterized complexity of the same problem, parameterized by (among other parameters) the number of threshold guards and the given safety specification. Our first result is a parameterized equivalent of NP-hardness, which shows that reachability of threshold automata is W[1]-hard. However, motivated by practical concerns, we then study two restricted cases for which we provide fixed-parameter tractable algorithms. In the first case, we restrict ourselves to threshold automata whose underlying control structure is acyclic and provide a simple algorithm which reduces the size of the automaton to a function of the considered parameters. In the second case we consider multiplicative threshold automata where the number of *fall* guards is a constant. (For a definition of fall guards, see Section 2.1.) Roughly speaking, multiplicativity means that any run over a smaller population of processes can be "lifted" to a run over a bigger population as well. For this case, we use results from Petri net theory to provide an FPT algorithm. Finally, the usefulness of these cases is shown by a preliminary implementation of our algorithms on various protocols from the benchmark in [24]. Our implementation compares favorably with ByMC, the tool developed in [24] and also with the algorithm given in [2].

2 Preliminaries

We denote the set of non-negative integers by $\mathbb{N}_0$, the set of positive integers by $\mathbb{N}_{>0}$ and the set of all non-negative rational numbers by $\mathbb{Q}_{\geq 0}$.

2.1 Threshold Automata

We introduce threshold automata, mostly following the definition and notations used in [2]. Along the way, we also illustrate the definitions on the example of Figure 2 from [25], which is a model of the Byzantine agreement protocol of Figure 1.

Environments

Threshold automata are defined relative to an *environment* $Env = (\Pi, RC, Num)$, where Π is a set of *environment variables* ranging over $\mathbb{N}_0$, $RC \subseteq \mathbb{N}_0^{\Pi}$ is a *resilience condition* over the environment variables, expressible as a linear formula, and $Num\colon RC \to \mathbb{N}_0$ is a linear function called the *number function*. Intuitively, a valuation of Π determines the number of processes of different kinds (e.g. faulty) executing the protocol, and RC describes the admissible combinations of values of environment variables. Finally, Num associates to a each admissible combination, the number of copies of the automaton that are going to run in parallel, or, equivalently, the number of processes explicitly modeled. In a Byzantine setting, faulty processes behave arbitrarily, and so we do not model them explicitly; in this case, the system consists of one copy of the automaton for every correct process. In the crash fault model, processes behave correctly until they crash and they must be modeled explicitly.

```
1  var $myval_i \in \{0,1\}$
2  var $accept_i \in \{false, true\} \leftarrow false$
3
4  while true do (in one atomic step)
5   if $myval_i = 1$
6      and not sent ECHO before
7   then send ECHO to all
8
9   if received ECHO from at least
10       $t + 1$ distinct processes
11       and not sent ECHO before
12  then send ECHO to all
13
14  if received ECHO from at least
15       $n - t$ distinct processes
16  then $accept_i \leftarrow true$
17 od
```

Figure 1 Pseudocode of a reliable broadcast protocol from [28] for a correct process i, where n and t denote the number of processes, and an upper bound on the number of faulty processes. The protocol satisfies its specification (if $myval_i = 0$ for every correct process i, then no correct process sets its *accept* variable to *true*) if $t < n/3$.

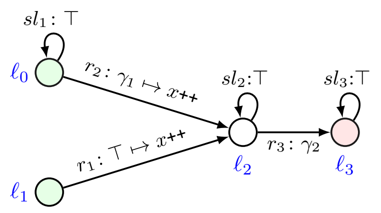

Figure 2 Threshold automaton from [25] modeling the body of the loop in the protocol from Fig. 1. Symbols γ_1, γ_2 stand for the threshold guards $x \geq (t+1) - f$ and $x \geq (n-t)-f$, where n and t are as in Fig. 1, and f is the actual number of faulty processes. The shared variable x models the number of ECHO messages sent by correct processes. Processes with $myval_i = b$ (line 1) start in location ℓ_b (in green). Rules r_1 and r_2 model sending ECHO at lines 7 and 12. The self-loop rules $sl_1, \ldots, sl_3$ are stuttering steps.

▶ **Example 1.** In the threshold automaton of Figure 2, the environment variables are n, f, and t, describing the number of processes, the number of (Byzantine) faulty processes, and the maximum possible number of faulty processes, respectively. The resilience condition is the constraint $n/3 > t \geq f$. The function *Num* is given by $Num(n,t,f) = n - f$, which is the number of correct processes.

Threshold automata

A *threshold automaton* over an environment *Env* is a tuple $\mathsf{TA} = (\mathcal{L}, \mathcal{I}, \Gamma, \mathcal{R})$, where $\mathcal{L}$ is a finite set of *local states* (or *locations*), $\mathcal{I} \subseteq \mathcal{L}$ is a nonempty subset of *initial locations*, Γ is a set of *shared variables* ranging over $\mathbb{N}_0$, and $\mathcal{R}$ is a set of *transition rules* (or just *rules*), formally described below.

A *transition rule* (or just a *rule*) is a tuple $r = (from, to, \varphi, \vec{u})$, where *from* and *to* are the *source* and *target* locations, $\varphi \subseteq \mathbb{N}_0^{\Pi \cup \Gamma}$ is a conjunction of *threshold guards* (described below), and $\vec{u}\colon \Gamma \to \{0,1\}$ is an *update*. We often let $r.from, r.to, r.\varphi, r.\vec{u}$ denote the components of r. Intuitively, r states that a process can move from *from* to *to* if the current values of Π and Γ satisfy φ, and when it moves, it updates the current valuation $\vec{\boldsymbol{g}}$ of Γ by performing the update $\vec{\boldsymbol{g}} := \vec{\boldsymbol{g}} + \vec{u}$. Since all components of $\vec{u}$ are nonnegative, the values of shared variables never decrease. A *threshold guard* φ has one of the following forms: $b \cdot x \bowtie a_0 + a_1 \cdot p_1 + \ldots + a_{|\Pi|} \cdot p_{|\Pi|}$ where $\bowtie \in \{\geq, <\}$, $x \in \Gamma$ is a shared variable, $p_1, \ldots, p_{|\Pi|} \in \Pi$ are the environment variables, $b \in \mathbb{N}_{>0}$ and $a_0, a_1, \ldots, a_{|\Pi|} \in \mathbb{Z}$ are integer coefficients. If $b = 1$, then the guard is called a simple guard. Additionally, if $\bowtie = \geq$, then the guard is called a *rise guard* and otherwise the guard is called a *fall guard*. We sometimes use $b \cdot x = a_0 + a_1 \cdot p_1 + \ldots + a_{|\Pi|} \cdot p_{|\Pi|}$ as a shorthand for $b \cdot x \geq a_0 + a_1 \cdot p_1 + \ldots + a_{|\Pi|} \cdot p_{|\Pi|} \wedge b \cdot x < (a_0 + 1) + a_1 \cdot p_1 + \ldots + a_{|\Pi|} \cdot p_{|\Pi|}$. Since shared variables are initialized to 0 and they never decrease, once a rise (resp. fall) guard becomes true (resp. false) it stays true (resp. false). We call this property *monotonicity of guards*. We let Φ^{rise}, Φ^{fall}, and Φ denote the sets of rise guards, fall guards, and all guards of TA. Finally, the *graph* of TA is the graph where the vertices are the locations and there is an edge between ℓ and ℓ' if there is a rule r in TA with $r.\mathit{from} = \ell$ and $r.\mathit{to} = \ell'$. We say that TA is acyclic if its graph is acyclic.

▶ **Example 2.** The rule r_2 of Figure 2 has ℓ_0 and ℓ_2 as source and target locations, $x \geq (t+1) - f$ as guard, and increments the value of the shared variable x by 1.

Configurations and transition relation

A *configuration* of TA is a triple $\sigma = (\vec{\boldsymbol{\kappa}}, \vec{\boldsymbol{g}}, \vec{\boldsymbol{p}})$ where $\vec{\boldsymbol{\kappa}} \colon \mathcal{L} \to \mathbb{N}_0$ describes the number of processes at each location, and $\vec{\boldsymbol{g}} \in \mathbb{N}_0^{|\Gamma|}$ and $\vec{\boldsymbol{p}} \in RC$ are valuations of the shared variables and the environment variables. In particular, $\sum_{\ell \in \mathcal{L}} \vec{\boldsymbol{\kappa}}(\ell) = \mathit{Num}(\vec{\boldsymbol{p}})$ always holds. A configuration is *initial* if $\vec{\boldsymbol{\kappa}}(\ell) = 0$ for every $\ell \notin \mathcal{I}$, and $\vec{\boldsymbol{g}} = \vec{0}$. We often let $\sigma.\vec{\boldsymbol{\kappa}}, \sigma.\vec{\boldsymbol{g}}, \sigma.\vec{\boldsymbol{p}}$ denote the components of σ.

A configuration $\sigma = (\vec{\boldsymbol{\kappa}}, \vec{\boldsymbol{g}}, \vec{\boldsymbol{p}})$ *enables* a rule $r = (\mathit{from}, \mathit{to}, \varphi, \vec{u})$ if $\vec{\boldsymbol{\kappa}}(\mathit{from}) > 0$, and $(\vec{\boldsymbol{g}}, \vec{\boldsymbol{p}})$ satisfies the guard φ, i.e., substituting $\vec{\boldsymbol{g}}(x)$ for x and $\vec{\boldsymbol{p}}(p_i)$ for p_i in φ yields a true expression, denoted by $\sigma \models \varphi$. If σ enables r, then TA can *move* from σ to the configuration $r(\sigma) = (\vec{\boldsymbol{\kappa}}', \vec{\boldsymbol{g}}', \vec{\boldsymbol{p}}')$ defined as follows: (i) $\vec{\boldsymbol{p}}' = \vec{\boldsymbol{p}}$, (ii) $\vec{\boldsymbol{g}}' = \vec{\boldsymbol{g}} + \vec{u}$, and (iii) $\vec{\boldsymbol{\kappa}}' = \vec{\boldsymbol{\kappa}} + \vec{v}_r$, where $\vec{v}_r = \vec{0}$ if $\mathit{from} = \mathit{to}$ and otherwise, $\vec{v}_r(\mathit{from}) = -1$, $\vec{v}_r(\mathit{to}) = +1$, and $\vec{v}_r(\ell) = 0$ for all other locations ℓ. We let $\sigma \to r(\sigma)$ denote that TA can move from σ to $r(\sigma)$.

Schedules and paths

A *schedule* is a (finite or infinite) sequence of rules. A schedule $\tau = r_1, \ldots, r_m$ is *applicable* to configuration σ_0 if there is a sequence of configurations $\sigma_1, \ldots, \sigma_m$ such that $\sigma_i = r_i(\sigma_{i-1})$ for $1 \leq i \leq m$, and we define $\tau(\sigma_0) := \sigma_m$. We let $\sigma \xrightarrow{*} \sigma'$ denote that $\tau(\sigma) = \sigma'$ for some schedule τ, and say that σ' is *reachable* from σ. Further we let $\tau \cdot \tau'$ denote the concatenation of two schedules τ and τ', and, given $\mu \geq 0$, let $\mu \cdot \tau$ the concatenation of τ with itself μ times.

A *path* or *run* is a finite or infinite sequence $\sigma_0, r_1, \sigma_1, \ldots, \sigma_{k-1}, r_k, \sigma_k, \ldots$ of alternating configurations and rules such that $\sigma_i = r_i(\sigma_{i-1})$ for every r_i in the sequence. If $\tau = r_1, \ldots, r_{|\tau|}$ is applicable to σ_0, then we let $\mathsf{path}(\sigma_0, \tau)$ denote the path $\sigma_0, r_1, \sigma_1, \ldots, r_{|\tau|}, \sigma_{|\tau|}$ with $\sigma_i = r_i(\sigma_{i-1})$, for $1 \leq i \leq |\tau|$. Similarly, if τ is an infinite schedule. Given a path $\mathsf{path}(\sigma, \tau)$, the set of all configurations in the path is denoted by $\mathsf{Cfgs}(\sigma, \tau)$.

The main focus of this paper will be the *reachability problem* and is defined as: Given TA and a set of locations $\mathcal{L}_{\mathsf{spec}} = \mathcal{L}_{=0} \cup \mathcal{L}_{>0}$ (called the specification), decide if there is a run of TA satisfying $\mathcal{L}_{\mathsf{spec}}$, i.e., decide if there is an initial configuration σ_0 such that some σ reachable from σ_0 satisfies $\sigma.\vec{\boldsymbol{\kappa}}(\ell) = 0$ for every $\ell \in \mathcal{L}_{=0}$ and $\sigma.\vec{\boldsymbol{\kappa}}(\ell) > 0$ for every $\ell \in \mathcal{L}_{>0}$. The *coverability problem* is the special case of the reachability problem where $\mathcal{L}_{=0} = \emptyset$.

2.2 Fixed-parameter tractability

We refer the reader to [13] for more information on parameterized complexity and only give the necessary definitions here. A *parameterized problem* L is a subset of $\Sigma^* \times \mathbb{N}_0$ for some alphabet Σ. A parameterized problem L is said to be *fixed-parameter tractable* (FPT) if there exists an algorithm A such that $(x, k) \in L$ iff $A(x, k)$ is true and A runs in time $f(k) \cdot |x|^{O(1)}$ for some computable function f, depending only on the *parameter* k. Given parameterized problems $L, L' \subseteq \Sigma^* \times \mathbb{N}_0$ we say that L is reducible to L' if there is an algorithm that, given an input (x, k), produces another input (x', k') in time $f(k) \cdot |x|^{O(1)}$ such that $(x, k) \in L \iff (x', k') \in L'$ and $k' \leq g(k)$ for some functions f and g depending only on k.

The parameterized clique problem is the set of all pairs (G, k) such that the graph G has a clique of size k. A parameterized problem L is said to be W[1]-hard if there is a parameterized reduction from L to the parameterized clique problem. If L is W[1]-hard and there is a parameterized reduction from L to L' then L' is W[1]-hard as well. W[1]-hardness is usually taken to be evidence that the problem does not have an FPT algorithm.

3 W[1]-hardness

We consider the reachability problem parameterized by the following parameters: $|\Phi|$ (the number of distinct guards), $|\mathcal{L}_{\mathsf{spec}}|$ (the size of the specification), $|RC|$ (the number of constraints in the resilience condition) and C (the maximum constant appearing in any of the guards of TA). (We note that if $x \in \Gamma \cup \Pi$ such that x does not appear in any of the guards in Φ or in any of the constraints in RC, then x can be removed from the input. Hence, we will always assume that $|\Gamma| + |\Pi| \leq |\Phi| + |RC|$ and for this reason, we do not consider $|\Gamma|$ and $|\Pi|$ explicitly as parameters.) In practice, all these values are quite small, roughly in the range of 10 to 25. Unfortunately, we prove the following negative result:

▶ **Theorem 3.** *Coverability (and hence reachability) for threshold automata parameterized by $|\Phi| + |\mathcal{L}_{spec}| + |RC| + C$ is $W[1]$-hard, even for acyclic automata where $|\Phi^{\text{fall}}|$ is a constant.*

Proof. We give a parameterized reduction from the *Unary Bin Packing* problem which is known to be $W[1]$-hard (See Theorem 2 of [20]) and is defined as follows:

Given: A finite set of items $I = \{0, 1, 2, \dots, w\}$, a size $size(i) \in \mathbb{N}_0$ for each $i \in I$, two positive integers B and k. (The integers $size(i)$ and B are encoded in unary)
Parameter: k
Decide: If there exists a partition of I into *bins* $I_1, \dots, I_k$ such that the sum of the sizes of the items in each bin I_j is less than or equal to B

Let $size = \sum_{i \in I} size(i)$. Our parameterized reduction works as follows: We will have $k+1$ *environment variables* $c_1, c_2, \dots, c_k, n$. Intuitively c_i will denote the sum of the sizes of the items in the i^{th} bin. The environment variable n will denote the number of processes modeled.

Further, we will have $k + 5$ *shared variables* $x_1, \dots, x_k, access_1, access_2, access_3$ and $count_1, count_2$. The variable x_i will denote the sum of the sizes of items which **do not belong** to the i^{th} bin. The role of $count_1$ and $count_2$ will be to set up two counters whose value will be exactly $size$ and B respectively. Our construction will have three gadgets and the role of $access_1, access_2$ and $access_3$ is to ensure that exactly one process can enter the first, second and third gadgets respectively.

We will have exactly one initial location $start$ and three rules of the form $r_1 : (start, access_1 < 1, access_1{+}{+}, p_0)$, $r_2 : (start, access_2 < 1, access_2{+}{+}, q_0)$ and $r_3 : (start, access_3 < 1, access_3{+}{+}, \ell_0)$. This means that once a process fires r_1, it increments $access_1$ and hence no other process can fire r_1 in the future. Similarly for the rules r_2 and r_3. Hence these three rules ensure that at most one process can enter p_0, q_0 and ℓ_0 respectively. For the specification, we set $\mathcal{L}_{=0} = \emptyset$ and $\mathcal{L}_{>0} = \{p_{corr}, q_{corr}, \ell_{w+1}\}$, whose locations we will now explain.

$$\begin{tikzcd}
p_0 \arrow[r, "count_1\text{++}"] & p_1 \arrow[r, "count_1\text{++}"] & p_2 & \cdots\cdots\cdots & p_{size-1} \arrow[r, "count_1\text{++}"] & p_{size} \arrow[r, "\sum_{1\leq i\leq k} c_i = count_1"] & p_{corr}
\end{tikzcd}$$

Figure 3 First gadget, which sets up the value of $count_1$ to be exactly $size$.

$$\begin{tikzcd}
q_0 \arrow[r, "count_2\text{++}"] & q_1 \arrow[r, "count_2\text{++}"] & q_2 & \cdots\cdots\cdots & q_{B-1} \arrow[r, "count_2\text{++}"] & q_B \arrow[r, "\bigwedge_{1\leq i\leq k}\ count_2 \geq c_i"] & q_{corr}
\end{tikzcd}$$

Figure 4 Second gadget, which sets up the value of $count_2$ to be exactly B.

$$\begin{tikzcd}
\ell_i \arrow[r, "cond_j"] & \ell^j_{i,0} \arrow[r, "upd_j"] & \ell^j_{i,1} \arrow[r, "upd_j"] & \ell^j_{i,2} & \cdots\cdots\cdots & \ell^j_{i,size(i)-1} \arrow[r, "upd_j"] & \ell^j_{i,size(i)} \arrow[r, "cond_j"] & \ell_{i+1}
\end{tikzcd}$$

Figure 5 Third gadget, which guesses the partition. Here $cond_j$ is the condition $x_j \geq \sum_{l\neq j} c_l$ and upd_j is the update $\wedge_{l\neq j}\ x_l\text{++}$.

The first gadget is given by Figure 3 and starts from the location p_0. It increments the shared variable $count_1$ to the value $size$. This gadget then ensures that we can reach p_{corr} only if the sum of the values of the environment variables $c_1, c_2, \ldots, c_k$ is exactly $size$. Notice that this gadget can be constructed in polynomial time, since each $size(i)$ is given in unary.

The second gadget is given by Figure 4 and starts from the location q_0. It increments the shared variable $count_2$ to the value B. This gadget then ensures that we can reach q_{corr} only if the values of the environment variables $c_1, c_2, \ldots, c_k$ are all at most B. Notice that this gadget can be constructed in polynomial time, since B is given in unary.

The third gadget is comprised of locations $\{\ell_i\}_{0\leq i\leq w+1}$ and $\{\ell^j_{i,q}\}^{1\leq j\leq k}_{0\leq i\leq w, 0\leq q\leq size(i)}$ and is comprised of various mini-gadgets. For every $0 \leq i \leq w$ and $1 \leq j \leq k$, the third gadget has a mini-gadget as given by Figure 5.

Recall that the shared variable x_j denotes the the sum of the sizes of items which **do not** belong to the j^{th} bin. Intuitively, if a process moves from ℓ_i to ℓ_{i+1} by going through $\ell^j_{i,0}, \ldots, \ell^j_{i,size(i)}$, this corresponds to putting the i^{th} item in the j^{th} bin and hence the mini-gadget increments the variables $\{x_l\}_{l\neq j}$ by the value $size(i)$. To ensure that we do not overshoot the bin size of the j^{th} bin, we have the guards $x_j \geq \sum_{l\neq j} c_l$ at the beginning and the end of the mini-gadget. Recall that the first gadget ensures that $\sum_{1\leq l\leq k} c_l = size$ and since x_j denotes the sum of sizes of items **not in** the j^{th} bin, the condition $x_j \geq \sum_{l\neq j} c_l$ ensures that the sum of the sizes of the items in the j^{th} bin is at most c_j. Since the second gadget forces $c_j \leq B$, it follows that the test $x_j \geq \sum_{l\neq j} c_l$ ensures that the sum of the sizes **in** the j^{th} bin is at most B. Notice that, once again this gadget can be constructed in polynomial time, since each $size(i)$ is given in unary.

Let RC be $n > 1$ and let $Num(c_1, \ldots, c_k, n) = n$. From the given construction it is clear that a configuration satisfying $\mathcal{L}_{\mathsf{spec}}$ is reachable iff we can partition I into k bins such that the sum of sizes of items in each bin does not exceed B.

It is clear that the reduction can be accomplished in polynomial time. Notice that the automaton is acyclic, $\mathcal{L}_{=0} = \emptyset$, $|\Phi| = O(k)$, $|\Phi^{\mathrm{fall}}| = 4$, $|RC| = 1$, $C = 1$ and $|\mathcal{L}_{\mathsf{spec}}| = 3$. Hence it is clear that $|\Phi| + |RC| + C + |\mathcal{L}_{\mathsf{spec}}| = O(k)$ and so the above reduction is indeed a parameterized reduction from the unary bin packing problem to the coverability problem. ◀

We now identify two special cases for which we give an FPT algorithm and discuss how these special cases arise in practice for a variety of distributed algorithms.

4 Acyclic threshold automata

The first case we consider is that of acyclic threshold automata, i.e., threshold automata whose underlying graph is acyclic. Except for one protocol, all the others in the benchmark of [24] are acyclic.[1] As the reduction of Theorem 3 produces acyclic threshold automata, we cannot hope for an FPT algorithm parameterized by $\{|\Phi|, |\mathcal{L}_{\mathsf{spec}}|, |RC|, C\}$. However, we show that

▶ **Theorem 4.** *Reachability of acyclic threshold automata parameterized by* $|\Phi| + |\mathcal{L}_{spec}| + |RC| + C + D$ *is in FPT, where* D *is the length of the longest path in the graph of the threshold automaton.*

Proof. Let TA be the given acyclic threshold automaton. First, we show that it is possible to incrementally "contract" the locations of TA in a bottom-up manner, while preserving the reachability property, such that, in the resulting automaton after contraction, the number of locations and rules is a function of $|\Phi| + |\mathcal{L}_{\mathsf{spec}}| + |RC| + C + D$. This then immediately implies our theorem, since the size of the whole automaton is now just a function of $|\Phi| + |\mathcal{L}_{\mathsf{spec}}| + |RC| + C + D$.

More formally, let the contraction of a subset $S = \{\ell_1, \dots, \ell_q\}$ of locations of TA be the following operation: We remove the locations $\ell_1, \dots, \ell_q$ from TA, introduce a new location ℓ_S and we replace all occurrences of $\ell_1, \dots, \ell_q$ in every rule of TA with ℓ_S. We say that a set S in TA is *good* if for every two locations $\ell, \ell' \in S$, if $(\ell, \ell'', \phi, \vec{u})$ is a rule in TA then $(\ell', \ell'', \phi, \vec{u})$ is also a rule in TA. Intuitively, this means that, for every rule that we can fire from ℓ, there is another rule we can fire from ℓ' which will have the exact same effect. Since TA is assumed to be acyclic, contracting a good set cannot introduce cycles. Let $\mathtt{Tar} = \{\ell : \ell \in \mathcal{L}_{\mathsf{spec}}\}$. The following is a very simple fact to verify:

Claim: Suppose S is a good set such that $S \cap \mathtt{Tar} = \emptyset$ and let TA' be the threshold automaton obtained by contracting S in TA. Then TA satisfies $\mathcal{L}_{\mathsf{spec}}$ iff TA' does.

Given a threshold automaton TA such that D is the length of the longest path in its graph, the "layers" of TA is a partition of the locations into subsets $L_0^{\mathsf{TA}}, L_1^{\mathsf{TA}}, \dots, L_D^{\mathsf{TA}}$ such that $\ell \in L_i^{\mathsf{TA}}$ iff the longest path ending at ℓ in the graph of TA is of length i. The subset L_i^{TA} will be called the i^{th} layer of TA. We will now construct a sequence of threshold automata $\mathsf{TA}_D, \mathsf{TA}_{D-1}, \dots, \mathsf{TA}_0$ such that for each i, $|L_i^{\mathsf{TA}_i}| + |L_{i+1}^{\mathsf{TA}_i}| + \dots + |L_D^{\mathsf{TA}_i}| \le g_i(|\Phi|, |RC|, |\mathcal{L}_{\mathsf{spec}}|, D)$ for some function g_i and such that TA_i satisfies $\mathcal{L}_{\mathsf{spec}}$ iff TA_{i+1} does.

For the base case of TA_D, we take the threshold automaton TA and consider the set $S_D := L_D^{\mathsf{TA}} \setminus \mathtt{Tar}$. We now contract S_D in TA to get a threshold automaton TA_D. Notice that S_D is a good set and by the above claim, TA_D satisfies $\mathcal{L}_{\mathsf{spec}}$ iff TA does.

For the induction step, suppose we have already constructed TA_{i+1}. For a location $\ell \in L_i^{\mathsf{TA}_{i+1}}$, define its *color* to be the set $\{(\ell', \phi, \vec{u}) : (\ell, \ell', \phi, \vec{u})$ is a rule in $\mathsf{TA}_{i+1}\}$. Observe that if $\ell \in L_i^{\mathsf{TA}_{i+1}}$ and $(\ell, \ell', \phi, \vec{u})$ is a rule in TA_{i+1} then $\ell' \in L_{i+1}^{\mathsf{TA}_{i+1}} \cup L_{i+2}^{\mathsf{TA}_{i+1}} \cup \dots \cup L_D^{\mathsf{TA}_{i+1}}$. By induction hypothesis, $|L_{i+1}^{\mathsf{TA}_{i+1}} \cup L_{i+2}^{\mathsf{TA}_{i+1}} \cup \dots \cup L_D^{\mathsf{TA}_{i+1}}| \le g_{i+1}(|\Phi|, |RC|, |\mathcal{L}_{\mathsf{spec}}|, D)$ for

[1] Some of the examples have self-loops on some locations, but since these self-loops do not update any of the shared variables, we can remove them without affecting the reachability relation.

some function g_{i+1}. It then follows that the number of possible colors is at most $2^{|\Phi|} \cdot 2^{|\Gamma|} \cdot g_{i+1}(|\Phi|, |\mathcal{L}_{\mathsf{spec}}|, D)$. Hence as long as the number of locations in $L_i^{\mathsf{TA}_{i+1}}$ is bigger than $2^{|\Phi|} \cdot 2^{|\Gamma|} \cdot g_{i+1}(|\Phi|, |\mathcal{L}_{\mathsf{spec}}|, D) + |\mathtt{Tar}|$ there will be two locations in $L_i^{\mathsf{TA}_{i+1}} \setminus \mathtt{Tar}$ which have the same color and can hence be contracted while maintaining the answer for $\mathcal{L}_{\mathsf{spec}}$. It then follows that by repeated contraction, we can finally end up at a threshold automaton TA_i such that $|L_i^{\mathsf{TA}_i}| + \cdots + |L_D^{\mathsf{TA}_i}| \leq O(2^{|\Phi|} \cdot 2^{|\Phi|+|RC|} \cdot g_{i+1}(|\Phi|, |RC|, |\mathcal{L}_{\mathsf{spec}}|, D) + |\mathtt{Tar}|)$. Taking this bound to be the function g_i, we get our required TA_i.[2]

Notice that the number of locations (and also rules) in TA_0 is only dependent on $|\Phi|, |RC|, |\mathcal{L}_{\mathsf{spec}}|$ and D. Since the reachability problem is decidable, it immediately follows that we have a parameterized algorithm for acyclic threshold automata running in time $f(|\Phi| + |RC| + |\mathcal{L}_{\mathsf{spec}}| + C + D) \cdot n^{O(1)}$ ◀

5 Threshold automata with constantly many fall guards

As a second case, we consider threshold automata in which the number of fall guards is a constant. In almost all of the benchmarks of [24], the number of fall guards is at most one. We provide some intuitive reason behind this phenomenon. In threshold automata, shared variables are usually used for two things: To record that some process has sent a message or to keep track of the number of processes which have crashed so far. If a shared variable v is used for the first purpose, then all guards containing v are typically rise guards, since we only want to check that enough messages have been received to proceed. On the other hand, if v is used to keep track of the number of crashed processes, then we will have a fall guard which allows a process to crash only if the value of v is less than the maximum number of processes allowed to crash. However, since we will only need one fall guard for this purpose, it follows that in practice we can hope to have very few fall guards in a threshold automaton.

Since the reduction of Theorem 3 produces threshold automata with constantly many fall guards, we need another restriction on this class as well, which we now describe.

▶ **Definition 5.** *A threshold automaton TA over an environment Env* $= (\Pi, RC, Num)$ *is called multiplicative if every fall guard is simple and for every* $\mu \in \mathbb{N}_{>0}$, *(i) for every rational vector* $\mathbf{p} \in \mathbb{Q}_{\geq 0}^{|\Pi|}$, *if* $RC(\mathbf{p})$ *is true then* $RC(\mu \cdot \mathbf{p})$ *is true and* $Num(\mu \cdot \mathbf{p}) = \mu \cdot Num(\mathbf{p})$ *and (ii) for every guard* $g := b \cdot x \bowtie a_0 + a_1 p_1 + \cdots + a_l p_l$ *in TA where* $\bowtie \in \{\geq, <\}$, *if* $(y, q_1, \ldots, q_l)$ *is a rational solution to g then* $(\mu \cdot y, \mu \cdot q_1, \ldots, \mu \cdot q_l)$ *is also a solution to g.*

To the best of our knowledge, many algorithms discussed in the literature (For example, see [8, 28, 6, 27, 19, 14, 7]), and more than two-thirds of all of the benchmarks of [24] satisfy multiplicativity. The main result of this section is

▶ **Theorem 6.** *Given a multiplicative threshold automaton TA with a constant number of fall guards and a specification* $\mathcal{L}_{spec}$, *it can be decided in time* $f(|\Phi|) \cdot n^{O(1)}$ *whether there is a run of TA satisfying* $\mathcal{L}_{spec}$.

The rest of this section is devoted to proving this result, which we do so in four parts. Let us fix a threshold automaton $\mathsf{TA} = (\mathcal{L}, \mathcal{I}, \Gamma, \mathcal{R})$, an environment $Env = (\Pi, RC, Num)$ and a specification $\mathcal{L}_{\mathsf{spec}}$ for the rest of this section. Let Φ denote the set of all guards which appear in TA.

[2] Though the function g_i as given here gives very huge bounds, we show in the experimental section that repeated contractions can sometimes reduce the number of locations by 50%. Intuitively, this is because the number of colors of a location in the benchmarks is much smaller than the worst-case analysis performed here.

First part: Decomposing paths into steady paths

First, similar to the paper [22], we show that the job of finding a path satisfying $\mathcal{L}_{\mathsf{spec}}$ can be reduced to that of finding a bounded number of concatenated "steady" paths. However, the result needs to be stated in a different manner than [22], so that later on, we could leverage the fact that the threshold automaton TA contains only constantly many fall guards.

A *context* ω is any subset of the guards of TA, i.e., $\omega \subseteq \Phi$. A rule r is said to be *activated* by a context ω if all the rise guards of r are present in ω and all the fall guards of r are **not** present in ω. The set of all rules activated by a context ω is denoted by $\mathcal{R}_\omega$.

The *context* of a configuration σ, denoted by $\omega(\sigma)$, is the set of all *rise* guards that evaluate to true and the set of all *fall* guards that evaluate to false in σ. Since the values of the shared variables can only increase along a path, it easily follows that for any configuration σ and any schedule τ applicable to σ, $\omega(\sigma) \subseteq \omega(\tau(\sigma))$.

We say that $\mathsf{path}(\sigma, \tau)$ is ω-*steady* if all the rules in the schedule τ are from $\mathcal{R}_\omega$ and for every configuration $\sigma' \in \mathsf{Cfgs}(\sigma, \tau)$, we have $\mathcal{R}_\omega \subseteq \mathcal{R}_{\omega(\sigma')}$. Intuitively, if $\mathsf{path}(\sigma, \tau)$ is ω-steady then the path only uses rules from $\mathcal{R}_\omega$. We have the following lemma.

▶ **Lemma 7.** *The specification $\mathcal{L}_{\mathsf{spec}}$ can be satisfied by a path of TA iff there exists $K \leq |\Phi|$, configurations $\sigma_0, \sigma'_0, \dots, \sigma_K, \sigma'_K$ and contexts $\omega_0 \subsetneq \omega_1 \subsetneq \dots \subsetneq \omega_K$ such that*

- *σ_0 is an initial configuration and σ'_K satisfies $\mathcal{L}_{\mathsf{spec}}$*
- *For every $i \leq K$, there is a ω_i-steady path $\sigma_i \xrightarrow{*} \sigma'_i$*
- *For every $i < K$, if $\mathcal{R}_{\omega_i} \subseteq \mathcal{R}_{\omega_{i+1}}$ then there is a ω_i-steady path $\sigma'_i \xrightarrow{*} \sigma_{i+1}$, otherwise $\sigma'_i \to \sigma_{i+1}$*

Proof. (Sketch.) Clearly if there exists such configurations and contexts then then there exists a path of TA which satisfies $\mathcal{L}_{\mathsf{spec}}$. To prove the other direction, suppose $\mathsf{path}(\sigma_0, \tau)$ is a path of TA which satisfies $\mathcal{L}_{\mathsf{spec}}$. Using the fact that $\omega(\sigma') \subseteq \omega(\tau'(\sigma'))$ for any configuration σ' and any schedule τ', we can decompose $\mathsf{path}(\sigma_0, \tau)$ into $\sigma_0, \tau_0, \sigma'_0, t_0, \sigma_1, \tau_1, \sigma'_1, t_1, \dots, \sigma_K, \tau_K, \sigma'_K$ such that for every i, $\omega(\sigma_i) = \omega(\sigma'_i)$, $\omega(\sigma'_i) \subsetneq \omega(\sigma_{i+1})$ and t_i is a rule of TA. We can then prove that the configurations $\sigma_0, \sigma'_0, \dots, \sigma_K, \sigma'_K$ and the contexts $\omega(\sigma_0), \dots, \omega(\sigma_K)$ satisfy the required conditions. ◀

Second part: Establishing a connection between continuous Petri nets and steady paths

Let us fix a context ω of the threshold automaton TA for the rest of this subsection. We say that a configuration σ is $\mathcal{R}_\omega$-applicable if $(\sigma.\vec{\boldsymbol{g}}, \sigma.\vec{\boldsymbol{p}})$ satisfies every guard of every rule in $\mathcal{R}_\omega$.

Continuous Petri nets

To define continuous Petri nets, we will mostly reuse the same notations from [5]. A continuous Petri net N is a tuple (P, T, F) where P is a finite set of places, T is a finite set of transitions and $F \subseteq P \times T \cup T \times P$ is the flow relation. For a transition t, let ${}^\bullet t = \{p : (p, t) \in F\}$ and $t^\bullet = \{p : (t, p) \in F\}$. A marking M of N is a function $M : P \to \mathbb{Q}_{\geq 0}$. Intuitively a marking M assigns $M(p)$ many *tokens* to each place $p \in P$. A marking is called integral if $M(p) \in \mathbb{N}_0$ for every place p. Given a marking M and a $k \in \mathbb{N}_{>0}$ let kM denote the marking $kM(p) = k \cdot M(p)$. The transition relation between two markings M and M' is defined as follows: For $\alpha \in (0, 1]$ and $t \in T$, we say that $M \xrightarrow{\alpha t} M'$ if for every $p \in {}^\bullet t$, $M(p) \geq \alpha$ and $M'(p) = M(p) - \alpha$ if $p \in {}^\bullet t \setminus t^\bullet$, $M'(p) = M(p) + \alpha$ if $p \in t^\bullet \setminus {}^\bullet t$ and $M'(p) = M(p)$ otherwise. We say that $M \to M'$ if $M \xrightarrow{\alpha t} M'$ for some α and t. Finally we say that $M \xrightarrow{*} M'$ if there exists $M_1, \dots, M_{k-1}$ such that $M \to M_1 \to \dots M_{k-1} \to M'$.

Constructing continuous Petri nets from contexts

We now construct a *continuous Petri net* N_ω for the context ω as follows: For every location ℓ of TA, we will have a place p_ℓ. Similarly for every variable $x \in \Gamma \cup \Pi$, we will have a place p_x. If $r = (\ell, \ell', \phi, \vec{u})$ is a rule in $\mathcal{R}_\omega$, we will have a transition t_r where ${}^\bullet t_r = \{p_\ell\}$ and $t_r^\bullet = \{p_{\ell'}\} \cup \{p_x : \vec{u}[x] = 1\}$.

We note that N_ω tries to simulate exactly the rules of $\mathcal{R}_\omega$, but it does not check whether the corresponding guard of a rule is true before firing it. To ensure that a proper simulation is carried out by N_ω, we will restrict ourselves to only runs of N_ω over *compatible markings* which are defined as follows.

A marking M of N_ω is called a *compatible marking* if $\sum_{\ell \in \mathcal{L}} M(p_\ell) = Num(\{M(p_x) : x \in \Pi\})$ and if for every $x \in \Gamma \cup \Pi$, the assignment $x \mapsto M(p_x)$ satisfies the resilience condition RC and all the guards of all the rules in $\mathcal{R}_\omega$. Notice that to every $\mathcal{R}_\omega$-applicable configuration σ of TA we can bijectively assign a canonical *compatible integral* marking $\mathbb{B}(\sigma)$ of N_ω where $(\mathbb{B}(\sigma))(p_x) = \sigma[x]$.

▶ **Proposition 8.** *The following are true:*

- *Suppose $\sigma \xrightarrow{*} \sigma'$ is an ω-steady run of TA. Then $\mathbb{B}(\sigma) \xrightarrow{*} \mathbb{B}(\sigma')$ in N_ω.*
- *Suppose M and M' are compatible markings of N_ω such that $M \xrightarrow{*} M'$. Then there exists $\mu \in \mathbb{N}_{>0}$ such that for all $k \in \mathbb{N}_{>0}$, $\mu k M$ and $\mu k M'$ are compatible integral markings and $\mathbb{B}^{-1}(\mu k M) \xrightarrow{*} \mathbb{B}^{-1}(\mu k M')$ is an ω-steady run of TA.*

Proof. (Sketch.) The first point is obvious from the definition. For the second point, if $M := M_0 \xrightarrow{\alpha_1 t_{r_1}} M_1 \xrightarrow{\alpha_2 t_{r_2}} M_2 \dots M_{l-1} \xrightarrow{\alpha_l t_{r_l}} M_l := M'$ is a run, then by multiplying the markings by the least common multiple of the denominators of $\{\alpha_i\}_{i \le l} \cup \{M_i(p_x) : i \le l,\ x \in \mathcal{L} \cup \Gamma \cup \Pi\}$ (which we take to be μ), we can get an integral run between $\mu k M$ and $\mu k M'$. Using multiplicativity of TA, we can translate this back to a run of TA. ◀

Third part: Characterizing steady paths

It was shown in ([5], Theorems 3.6 and 3.3) that there is a logic (which the authors of [5] call *convex semi-linear Horn formulas*) characterizing reachability in continuous Petri nets, whose satisfiability can be tested **in polynomial time**. Using this result, proposition 8 and multiplicativity, we show that

▶ **Lemma 9.** *Given a context ω, in polynomial time we can construct a convex semi-linear Horn formula $\phi_\omega(\mathbf{x}, \mathbf{y})$ with $2(|\mathcal{L}| + |\Gamma| + |\Pi|)$ free variables such that*

- *If $\sigma \xrightarrow{*} \sigma'$ is an ω-steady path of TA then $\phi_\omega(\sigma, \sigma')$ is true*
- *Suppose $\phi_\omega(M, M')$ is true. Then there exists $\mu \in \mathbb{N}$ such that for all $k \in \mathbb{N}$, $\mu k M, \mu k M'$ are configurations of TA such that $\mu k M \xrightarrow{*} \mu k M'$ is an ω-steady path in TA.*

▶ **Lemma 10.** *Given a rule r of TA, in polynomial time we can construct a convex semi-linear Horn formula $\phi_r(\mathbf{x}, \mathbf{y})$ with $2(|\mathcal{L}| + |\Gamma| + |\Pi|)$ free variables such that*

- *If σ and σ' are configurations of TA such that $\sigma' = r(\sigma)$, then $\phi_r(\sigma, \sigma')$ is true.*
- *Suppose $\phi_r(M, M')$ is true. Then there exists $\mu \in \mathbb{N}$ such that for all $k \in \mathbb{N}$, $\mu k M, \mu k M'$ are configurations of TA such that $\mu k M' = (\mu k \cdot r)(\mu k M)$, i.e., $\mu k M'$ can be obtained by applying the rule r to $\mu k M$, repeatedly for μk many steps.*

Fourth part: Bringing it all together

▶ **Theorem 11.** *Given a multiplicative threshold automaton* TA *with constant number of fall guards and a specification $\mathcal{L}_{spec}$, it can be decided in time $f(|\Phi|) \cdot n^{O(1)}$ whether there is a run of* TA *satisfying $\mathcal{L}_{spec}$.*

Proof. (Sketch.) One can easily show that if we have a monotonically increasing context sequence $\omega_0 \subsetneq \omega_1 \subsetneq \cdots \subsetneq \omega_K$, the size of the set $\{j : \mathcal{R}_{\omega_j} \not\subseteq \mathcal{R}_{\omega_{j+1}}\}$ is at most $|\Phi^{\text{fall}}|$. Using this observation, we proceed as follows. We iterate over all $K \leq |\Phi|$ and over all possible monotonically increasing context sequences $\omega_0 \subsetneq \omega_1 \subsetneq \cdots \subsetneq \omega_K$ of length $K+1$ and all possible rule sequences $r_1, \ldots, r_c$ of length $c = \#\{j : \mathcal{R}_{\omega_j} \not\subseteq \mathcal{R}_{\omega_{j+1}}\}$. Note that the number of such iterations is at most $O(|\Phi| \cdot |\Phi^{\text{fall}}| \cdot |\Phi|! \cdot 2^{|\Phi|} \cdot |\mathcal{R}|^{|\Phi^{\text{fall}}|})$. Since $|\Phi^{\text{fall}}|$ is assumed to be a constant, the exponential dependence only lies upon $|\Phi|$.

A position $0 \leq l \leq K$ is called *bad* if $\mathcal{R}_{\omega_l} \not\subseteq \mathcal{R}_{\omega_{l+1}}$. Let $j_1, \ldots, j_c$ be the set of all bad positions. Using lemmas 9 and 10 we can write down the following convex semi-linear Horn formula in polynomial time:

$$\xi_0(\mathbf{x}_0, \mathbf{y}_0, \mathbf{x}_1) \wedge \xi_1(\mathbf{x}_1, \mathbf{y}_1, \mathbf{x}_2) \wedge \cdots \wedge \xi_{K-1}(\mathbf{x}_{K-1}, \mathbf{y}_{K-1}, \mathbf{x}_K) \wedge \xi_K(\mathbf{x}_K, \mathbf{y}_K) \tag{1}$$

where $\xi_K(\mathbf{x}_K, \mathbf{y}_K) = \phi_{\omega_K}(\mathbf{x}_K, \mathbf{y}_K)$ and ξ_i for $i < K$ is defined as follows: If i is a bad position, i.e., if $i = j_l$ for some $1 \leq l \leq c$, then $\xi_i(\mathbf{x}_i, \mathbf{y}_i, \mathbf{x}_{i+1}) = \phi_{\omega_i}(\mathbf{x}_i, \mathbf{y}_i) \wedge \phi_{r_l}(\mathbf{y}_i, \mathbf{x}_{i+1})$. It i not a bad position, then $\xi_i(\mathbf{x}_i, \mathbf{y}_i, \mathbf{x}_{i+1}) = \phi_{\omega_i}(\mathbf{x}_i, \mathbf{y}_i) \wedge \phi_{\omega_i}(\mathbf{y}_i, \mathbf{x}_{i+1})$

To equation (1), we also add a constraint stating that $\mathbf{x}_0$ is an initial configuration and $\mathbf{y}_K$ satisfies $\mathcal{L}_{\text{spec}}$. By proposition 8 we can then easily show that, there is a run of TA satisfying $\mathcal{L}_{\text{spec}}$ iff in at least one iteration, the constructed formula (1) is satisfiable. ◀

6 NP-hardness of multiplicative threshold automata

A natural question arises from the results of the previous section. Can we do better than fixed-parameter tractability and instead solve the reachability problem for multiplicative threshold automata in polynomial time? We remark that the proof of NP-hardness of reachability for threshold automata given in [2] does not produce multiplicative threshold automata and hence does not answer this question. Nevertheless, we show that it is unlikely for reachability of multiplicative threshold automata to be in polynomial time.

▶ **Theorem 12.** *Coverability (and hence reachability) for multiplicative threshold automata is NP-hard even when there are no fall guards.*

Proof. We give an easy reduction from 3-SAT. Let φ be a propositional formula with variables $x_1, \ldots, x_k$ and clauses $C_1, \ldots, C_m$. We will have $2k$ shared variables $y_1, \ldots, y_k, \bar{y_1}, \ldots, \bar{y_k}$ and one environment variable n, denoting the number of processes. Incrementing y_i ($\bar{y_i}$ resp.) corresponds to setting x_i to true (false resp). We will have $2k+1$ locations $\ell_0, \ell'_0, \ell_1, \ell'_1, \ldots, \ell'_{k-1}, \ell_k$. Between ℓ_i and ℓ'_i we will have two rules which increment y_i and $\bar{y_i}$ respectively. To ensure that all the processes increment the same variable, we have two rules from ℓ'_i to ℓ_{i+1} which test that $y_i \geq n$ and $\bar{y_i} \geq n$ respectively. Hence if one process increments y_i and another increments $\bar{y_i}$, then all the processes get stuck at ℓ'_i.

Let $var(x_i) = y_i$ and let $var(\bar{x_i}) = \bar{y_i}$. We will then have m locations $\ell_{k+1}, \ell_{k+2}, \ldots, \ell_{k+m}$ and the following rules between ℓ_{k+i-1} and ℓ_{k+i} for every $1 \leq i \leq m$: If the clause C_i is of the form $a \vee b \vee c$ then there are three rules between ℓ_{k+i-1} and ℓ_{k+i}, each checking if $var(a) \geq 1$, $var(b) \geq 1$ and $var(c) \geq 1$ respectively. Hence if either one of $var(a)$ or $var(b)$ or $var(c)$ was incremented, the processes could move from ℓ_{k+i-1} to ℓ_{k+i}, otherwise all the processes get stuck at ℓ_{k+i-1}. Finally we set the initial location to be ℓ_0 and the specification to be $\mathcal{L}_{=0} = \emptyset$ and $\mathcal{L}_{>0} = \{\ell_{k+m}\}$. It is then easy to see that φ is satisfied iff there is a run which satisfies $\mathcal{L}_{\texttt{spec}}$. ◀

7 Experiments

We implemented the contraction procedure for the acyclic threshold automata as presented in section 4 and then used the algorithm for multiplicative threshold automata presented in section 5. To leverage the solid engineering work that has been put into modern SMT solvers, we used the Z3 solver to solve the convex semi-linear Horn formulas as well as to choose a context (and rule) sequence. We applied our implementations to all the multiplicative protocols in the latest version of the benchmark of [24], which contains various algorithms taken from the distributed computing literature. For more information on the protocols, we refer the reader to the benchmark of [24].

Table 1 The experiments were run on a machine with Intel® Core™ i5-7200U CPU with 7.7 GiB memory. The time limit was set to be 2 hours and the memory limit was set to be 7 GiB. TLE (MLE) means that the time limit (memory limit) exceeded for the particular benchmark.

Input	**Case**	**Time, seconds**		
	(if more than one)	**This paper**	**Algo from [2]**	**ByMC**
frb		0.38	0.32	0.07
frb	hand-coded TA	0.29	0.31	0.16
strb		0.44	0.43	0.14
strb	hand-coded TA	0.32	0.30	0.10
nbacg		2.92	8.43	9.71
aba	Case 1	4.49	10.26	25.6
aba	Case 2	18.29	41.92	704.9
cbc	Case 1	3579.24	MLE	MLE
cbc	Case 2	183.61	2035.5	26.37
cbc	Case 3	MLE	MLE	MLE
cbc	Case 4	MLE	MLE	MLE
cbc	hand-coded TA	3.27	0.91	0.26
cf1s	Case 1	13.81	13.53	37.09
cf1s	Case 2	12.47	16.14	186.5
cf1s	Case 3	84.95	86.98	7875
cf1s	hand-coded TA	1.75	1.31	2737.53
c1cs	Case 1	179.39	598.2	TLE
c1cs	Case 2	70.77	747.86	7119.71
c1cs	Case 3	604.91	1575.21	MLE
c1cs	hand-coded TA	4.87	6.63	TLE

Evaluation: Table 1 summarizes our results and compares them with the results obtained using ByMC, the tool presented in [24] and the algorithm from [2].

For some safety specifications, our contraction procedure was able to reduce the number of locations by more than 50% for the cbc protocol(s). This helped us save some memory, as we also noticed that running just the algorithm for multiplicative threshold automata took much more memory and the algorithm was not able to complete its execution. Our implementation compares favorably with both ByMC and the algorithm from [2] in some cases, but also performs worse in some of the hand-coded examples, the second case of cbc and the frb and strb protocols.

8 Conclusion

In this paper, we have investigated the parameterized complexity of safety in threshold automata. Though we have proved hardness results even in very restricted settings, we have also identified tractable special cases which arise in practice. A preliminary implementation of our algorithms suggest that these methods might be useful in practice as well.

For the sake of simplicity, we have only restricted to verifying safety properties in this paper. A special type of logic called $\mathsf{ELTL}_{\mathsf{FT}}$ [12] has been proposed for threshold automata which can express various safety and liveness properties. Since model checking this logic decomposes to a finite number of safety specifications (modulo some technical constraints), we believe that our algorithm for multiplicative threshold automata can be adapted to give an algorithm for model checking this logic as well.

References

1 Dana Angluin, James Aspnes, Zoë Diamadi, Michael J. Fischer, and René Peralta. Computation in networks of passively mobile finite-state sensors. *Distributed Comput.*, 18(4):235–253, 2006. doi:10.1007/s00446-005-0138-3.

2 A. R. Balasubramanian, Javier Esparza, and Marijana Lazić. Complexity of verification and synthesis of threshold automata. In *Accepted at ATVA 2020*, 2020. URL: https://arxiv.org/abs/2007.06248.

3 Nathalie Bertrand, Igor Konnov, Marijana Lazić, and Josef Widder. Verification of randomized consensus algorithms under round-rigid adversaries. In *CONCUR*, volume 140 of *LIPIcs*, pages 33:1–33:15, 2019.

4 Roderick Bloem, Swen Jacobs, Ayrat Khalimov, Igor Konnov, Sasha Rubin, Helmut Veith, and Josef Widder. *Decidability of Parameterized Verification.* Synthesis Lectures on Distributed Computing Theory. Morgan & Claypool Publishers, 2015.

5 Michael Blondin and Christoph Haase. Logics for continuous reachability in petri nets and vector addition systems with states. In *32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, June 20-23, 2017*, pages 1–12. IEEE Computer Society, 2017. doi:10.1109/LICS.2017.8005068.

6 Gabriel Bracha and Sam Toueg. Asynchronous consensus and broadcast protocols. *J. ACM*, 32(4):824–840, 1985.

7 Francisco Vilar Brasileiro, Fabíola Greve, Achour Mostéfaoui, and Michel Raynal. Consensus in one communication step. In *PaCT*, volume 2127 of *LNCS*, pages 42–50, 2001.

8 Tushar Deepak Chandra and Sam Toueg. Unreliable failure detectors for reliable distributed systems. *J. ACM*, 43(2):225–267, 1996.

9 Peter Chini, Jonathan Kolberg, Andreas Krebs, Roland Meyer, and Prakash Saivasan. On the complexity of bounded context switching. In Kirk Pruhs and Christian Sohler, editors, *25th Annual European Symposium on Algorithms, ESA 2017, September 4-6, 2017, Vienna, Austria*, volume 87 of *LIPIcs*, pages 27:1–27:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.ESA.2017.27.

10 Peter Chini, Roland Meyer, and Prakash Saivasan. Fine-grained complexity of safety verification. In Dirk Beyer and Marieke Huisman, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 24th International Conference, TACAS 2018, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2018, Thessaloniki,*

Greece, April 14-20, 2018, Proceedings, Part II, volume 10806 of *Lecture Notes in Computer Science*, pages 20–37. Springer, 2018. doi:10.1007/978-3-319-89963-3_2.

11 Peter Chini, Roland Meyer, and Prakash Saivasan. Complexity of liveness in parameterized systems. In Arkadev Chattopadhyay and Paul Gastin, editors, *39th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2019, December 11-13, 2019, Bombay, India*, volume 150 of *LIPIcs*, pages 37:1–37:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.FSTTCS.2019.37.

12 Peter Chini, Roland Meyer, and Prakash Saivasan. Liveness in broadcast networks. In *NETYS 2019, Revised Selected Papers*, pages 52–66, 2019.

13 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms.* Springer, 2015. doi:10.1007/978-3-319-21275-3.

14 Dan Dobre and Neeraj Suri. One-step consensus with zero-degradation. In *DSN*, pages 137–146, 2006.

15 Constantin Enea and Azadeh Farzan. On atomicity in presence of non-atomic writes. In Marsha Chechik and Jean-François Raskin, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 22nd International Conference, TACAS 2016, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2016, Eindhoven, The Netherlands, April 2-8, 2016, Proceedings*, volume 9636 of *Lecture Notes in Computer Science*, pages 497–514. Springer, 2016. doi:10.1007/978-3-662-49674-9_29.

16 Javier Esparza, Alain Finkel, and Richard Mayr. On the verification of broadcast protocols. In *LICS*, pages 352–359. IEEE Computer Society, 1999.

17 Azadeh Farzan and P. Madhusudan. The complexity of predicting atomicity violations. In Stefan Kowalewski and Anna Philippou, editors, *Tools and Algorithms for the Construction and Analysis of Systems, 15th International Conference, TACAS 2009, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2009, York, UK, March 22-29, 2009. Proceedings*, volume 5505 of *Lecture Notes in Computer Science*, pages 155–169. Springer, 2009. doi:10.1007/978-3-642-00768-2_14.

18 Steven M. German and A. Prasad Sistla. Reasoning about systems with many processes. *J. ACM*, 39(3):675–735, 1992.

19 Rachid Guerraoui. Non-blocking atomic commit in asynchronous distributed systems with failure detectors. *Distributed Computing*, 15(1):17–25, 2002.

20 Klaus Jansen, Stefan Kratsch, Dániel Marx, and Ildikó Schlotter. Bin packing with fixed number of bins revisited. *Journal of Computer and System Sciences*, 79(1):39–49, 2013. doi:10.1016/j.jcss.2012.04.004.

21 Igor Konnov, Helmut Veith, and Josef Widder. On the completeness of bounded model checking for threshold-based distributed algorithms: Reachability. In *CONCUR*, volume 8704 of *LNCS*, pages 125–140, 2014.

22 Igor Konnov, Helmut Veith, and Josef Widder. SMT and POR beat counter abstraction: Parameterized model checking of threshold-based distributed algorithms. In Daniel Kroening and Corina S. Pasareanu, editors, *Computer Aided Verification - 27th International Conference, CAV 2015, San Francisco, CA, USA, July 18-24, 2015, Proceedings, Part I*, volume 9206 of *Lecture Notes in Computer Science*, pages 85–102. Springer, 2015. doi:10.1007/978-3-319-21690-4_6.

23 Igor Konnov, Helmut Veith, and Josef Widder. On the completeness of bounded model checking for threshold-based distributed algorithms: Reachability. *Information and Computation*, 252:95–109, 2017.

24 Igor Konnov and Josef Widder. Bymc: Byzantine model checker. In *ISoLA (3)*, volume 11246 of *LNCS*, pages 327–342. Springer, 2018.

25 Igor V. Konnov, Marijana Lazic, Helmut Veith, and Josef Widder. A short counterexample property for safety and liveness verification of fault-tolerant distributed algorithms. In *POPL 2017*, pages 719–734, 2017.

26 Jure Kukovec, Igor Konnov, and Josef Widder. Reachability in parameterized systems: All flavors of threshold automata. In *CONCUR*, pages 19:1–19:17, 2018.

27 Achour Mostéfaoui, Eric Mourgaya, Philippe Raipin Parvédy, and Michel Raynal. Evaluating the condition-based approach to solve consensus. In *DSN*, pages 541–550, 2003.

28 T.K. Srikanth and Sam Toueg. Simulating authenticated broadcasts to derive simple fault-tolerant algorithms. *Dist. Comp.*, 2:80–94, 1987.