université
PARIS-SACLAY

Verification and Synthesis of Parameterized Concurrent Systems

Thèse de doctorat de l'Université Paris-Saclay

École doctorale n° 580, Sciences et Technologies de l'Information et de la Communication (STIC)
Spécialité de doctorat: Informatique
Unité de recherche: LMF, Université Paris-Saclay, CNRS, ENS Paris-Saclay
Référent: ENS Paris-Saclay

Thèse présentée et soutenue à Gif-sur-Yvette, le 30 Septembre 2021, par

Anirban MAJUMDAR

Composition du jury:

Prénom Nom Titre, Affiliation	Président/e
Giorgio DELZANNO Professor, Università di Genova, Italy	Rapporteur
Emmanuel FILIOT Maître de recherche, FNRS, Belgium	Rapporteur
Martin ZIMMERMANN Associate professor, Aalborg Universitet, Denmark	Examinateur
Benedikt BOLLIG Directeur de recherche, CNRS, France	Examinateur
Patricia BOUYER-DECITRE Directrice de recherche, CNRS, France	Directrice
Nathalie BERTRAND Directrice de recherche, Inria, France	Coencadrante

Thèse de doctorat

NNT: 2021UPASG059

Abstract

This thesis is at the crossroad of verification and synthesis of parameterized concurrent systems. The parameterized model checking problem asks whether a system satisfies a given specification independently of the number of its components, whereas synthesis requires an algorithmic design of protocols for its components so that the specification is satisfied.

The contribution of this thesis is two-fold. In the first part, we study a parameterized model of networks where processes are distributed over an undirected graph, running the same *broadcast protocol*. Processes communicate via broadcasts of messages, each process can broadcast messages to its neighbours. Then the coverability problem asks, given a state of the protocol, whether there exists an execution that reaches a configuration such that at least one process is in that state. We show that for positive instances of the coverability problem in reconfigurable semantics, the size (resp., the length) of a minimal covering execution, called the *cutoff* (resp., the *covering length*), is linearly (resp., quadratically) bounded. We introduce loss-on-broadcast semantics, and show similar bounds for the cutoff and the covering length.

The interactions between agents can be modelled using games. Games with a fixed number of players have been extensively studied in the literature. In the second part of the thesis, we introduce *parameterized concurrent games*, a model of concurrent games with arbitrarily many agents, where the edges are labelled by languages of finite yet *a priori* unbounded words, and study two different settings on this model. First, we consider a scenario where a distinguished player Eve is trying to achieve a given objective against the coalition of her opponents, not knowing *a priori* how many they are; and second, a coalition setting where all players collectively try to achieve a common goal.

In the first setting, we prove the existence of a winning strategy for Eve is decidable, and show tight complexity bounds for *reachability* objectives. More precisely, when the edges are labelled with regular languages, we show that the game problem is PSPACE-complete. In the coalition setting, we consider *safety* objectives and show that the existence of a winning coalition strategy is decidable, and prove complexity bounds for the same. We show that the safe coalition problem is in EXPSPACE and PSPACE-hard. Further, one can synthesize a coalition strategy, if it exists, that uses exponential memory, which can be computed in exponential space.

Contents

Chapter 1

Introduction

The generalization and everyday usage of modern distributed systems call both for the verification and synthesis of algorithms running on distributed systems. Concrete examples are cloud computing [AFG+09], blockchain technologies [NK18], servers with multiple clients [DN12], wireless sensor networks [YMG08], bio-chemical systems [FK13], or fleets of drones cooperating to achieve a common goal [CQSS19]. In their general form, these systems are not only distributed, but they may also involve an arbitrary large number of agents. This explains the interest of the model-checking community both for the verification of parameterized systems (see for instance the survey [Esp14], and the book [BJK+15]), and for the synthesis of distributed systems [PR90]. This thesis is at the crossroad of those topics.

1.1 Background

Model checking. Computerized systems are becoming an integral part of our daily life, be it a personal computer, smart-phone, or even parts of cars, planes, banking, medical equipments, etc. Needless to say, it is not only desired that these systems be flawless in their implementations, but in some scenarios, small hardware or software errors may lead to enormous damages. For example, when the brake of a car is applied, necessary hardwares need to react immediately. A software error in Ariane-5 missile led to a crash which cost a huge financial damage [Rep96]. This explains the need for developing techniques to automatically check correctness of the computerized systems.

Numerous verification techniques have been developed over the years, one of them being *formal verification*. The technique consists in checking correctness of a mathematical formalization of the system algorithmically. *Model checking* is a formal verification technique that performs an exhaustive search of the state space of the system model [CES09]. More precisely, the behaviours of a system are often represented by a transition system and its desired properties are described as a logical specification (for example, a formula in a temporal logic). Then the model checking techniques explore all possible behaviours of the transition system and check if all of them satisfy the specification. For negative

instances, most model checking algorithms output a counter-example. For further reading on model checking techniques, we refer to [BK08].

Parameterized verification. In *distributed systems*, many processes run concurrently, interacting with each other and also with the environment. Modern computerized systems are heavily dependent on the theory of concurrency. Typical examples are multicore processors, telephone networks, wireless networks, etc. One of the main challenges in verification of these systems is *state-space explosion*: the number of states of the system grows rapidly while keeping track of interactions between processes. Another issue is that some algorithms need to be proven correct independently of the system size. For instance, a mutual exclusion algorithm should ensure that two processes never enter the critical section simultaneously for any number of processes. Similarly, a cache-coherence protocol should be correct independently of the number of processes. This motivates the study of *parameterized model checking*, which is the central aspect of this thesis.

In general, *parameterized systems* are computerized systems that can be described using one or more *parameters*. Typically, the parameters are related to the size or the topology of a network, the initial values of variables, etc. In the first part of the thesis, we consider a particular class of parameterized systems: systems that consist of arbitrarily many identical processes, each of which follows a same finite-state protocol. Then the parameterized model checking problem (PMCP) asks whether each member of the infinite family satisfies some given specification. Notice that the number of processes here is a parameter; indeed, if that number is fixed *a priori* then, assuming each component of the system has a finite number of states, the above problem would fall under finite-state model checking. The verification of parameterized systems started in the late '80s [AK86, GS92, EN95, EFM99], and recently regained attention from the model-checking community [Esp14, BJK+15].

Unfortunately, the PMCP is in general an undecidable problem [AK86]. However, in the proof of the above undecidability result, the description of a component of the system depends on the parameter, the number of components. Therefore, their result does not directly apply to systems that are composed of arbitrarily many copies of a finite program whose description is independent of the size of the system. For instance, in [GS92], the authors study a model of systems which communicate via a rendez-vous mechanism (at a time, two entities synchronize via message passing) and show that the PMCP is decidable.

The decidability of the PMCP depends on various factors: the shape of the communication architecture (cliques, rings, stars, etc.), the means of communication (message-passing, token-passing, shared memory, etc.), the possibility of non-deterministic reconfiguration of the network topology, the presence of a leader, etc. Broadcast communication (a node broadcasts a message to all its neighbours simultaneously) was originally studied in [EFM99], where the entities synchronize via message passing. Later, broadcast communication was also considered, for instance, in [DSZ10, DSTZ12, DSZ12]. In [Suz88, EN95, CTTV04, AJKR14], the authors consider systems where processes communicate via token-passing. Asynchronous shared-memory systems were studied in [EGM13, BMR+16], where processes communicate through a shared, bounded-value

register. Angluin *et al.* [AAD+04] defined a model of computation based on pairwise interaction between agents, called *population protocol*, which was further studied, for instance, in [AAE06, EGLM17, LFI+17, BEG+20]. Other contributions in this line of research also include [BDG+19, CFO20], where an environment tries to control a population of arbitrary size uniformly, motivated by biological systems, for example, population of yeasts.

Cutoff detection technique. While the undecidability results for the PMCP are mostly proved by reducing the non-halting problem of Turing machines, or the reachability problem of two-counter machines, various techniques and abstractions have been applied to show decidability for specific subclasses of parameterized systems (see, for instance, the survey [BJK+15]), of which, the existence of a *cutoff* is of special interest. Such an approach was considered, for example, in [EN95, BHV03, CTTV04].

The intuitive idea behind this approach is that if a distributed system goes wrong for some large number of processes, then the bug already appears in a smaller instance. In other words, if everything goes fine with a specific number, say m, of processes, then the system will also behave correctly for any larger number of processes. For a system which satisfies the above criterion for a given specification, the number m is called the *cutoff* of that system for that particular property. The existence of a cutoff reduces the PMCP to a finite-state model checking problem. Indeed, if a cutoff exists, it is then sufficient to check if the property is satisfied for all instances of the system with at most m processes. The first part of this work fits in this scenario: we study the existence and bounds of a cutoff in the model of *broadcast protocols*. However, unfortunately, not every parameterized system enjoys a cutoff property (see, for example, [AKR+14]).

Ad hoc networks. We study in the first part of the thesis the parameterized verification of an automata-based model of *ad hoc networks*, called broadcast protocols, introduced in [DSZ10]. An ad hoc network is composed of several processes that execute the same broadcast protocol. The latter is a finite automaton, where transitions are labelled with (1) broadcasts of messages or (2) receptions of messages. A configuration in an ad hoc network is then comprised of a set of processes and their current local states, together with a communication topology that represents which processes are within transmission range. A transition represents the effect of one process sending a message to its neighbours. Parameterized verification of ad hoc networks amounts to checking whether a given property is satisfied independently of the initial configuration, and in particular, independently of the number of processes and communication topology. Among various properties of broadcast protocols, the *coverability problem* is of special interest and, in this part, we study the notion of cutoff for the same.

Program synthesis. Verification techniques algorithmically check correctness of a mathematical model of a system with respect to a particular property, and for negative instances, a counter-example is provided. However, after significant amount of resources

have been invested in the system design, verifying the system and fixing bugs several times before a correct program is developed is not the most desired solution. Given a specification, building a model of a system automatically such that all behaviours of the system satisfy the specification would be more ideal. This is known as the *synthesis problem*.

The synthesis problem was originally stated by Church [Chu62]. The author considered the problem of synthesizing a circuit from a logical specification expressed in Monadic Second Order logic (MSO) over the structure $(\mathbb{N}, +1)$. More precisely, given a specification $\varphi(X, Y)$ in MSO between inputs X and outputs Y defining the relation $R_\varphi \subseteq \{0,1\}^\omega \times \{0,1\}^\omega$, construct a circuit (if it exists) that, for any received input α, outputs β such that $R_\varphi(\alpha, \beta)$ holds. A decade later, Büchi and Landweber [BL69] solved the problem using a game-theoretic framework (an alternative proof was given by Rabin [Rab72] using tree automata techniques). The high level idea of Büchi and Landweber's proof is to first convert the MSO formula into an equivalent Muller automaton, which can further be reduced to a two-player turn-based game with a parity objective such that a winning strategy for the system player in that game corresponds to a correct program for the given specification. A turn-based game perfectly captures the interaction between the system and the environment in a *reactive system*. We refer to the tutorial [Tho09] for a simplified explanation of the proof.

Distributed synthesis. The synthesis problem was extended to distributed systems by Pnueli and Rosner [PR90]. The *distributed synthesis* problem asks, given a specification and a model of a system, to find a program for each of the processes such that the overall behaviour of the system satisfies the specification. There are several possible formalizations for distributed synthesis. For instance, an architecture of processes with communication links between agents was considered in [PR90] to represent a distributed system, whereas coordination games were used in [PR79, MW03, BKP11]. The two settings are related, and many (un-)decidability results have been proven, depending on various parameters. To name some of them, it was shown in [PR90] that the distributed synthesis problem is undecidable for a system with two parallel processes and an environment, even with a *safety specification*, and hence undecidable in general. In the same work, the authors study some subclasses where the problem is decidable, for example, *pipeline* architectures. In [MW03], the authors take a game-theoretic approach to the problem, where processes only have a partial view of the environment's states.

Games on graphs. Game theory provides an approach towards solving the synthesis problem. Interactions between agents in a system are modelled using games on graphs. In particular, in an open system with a single process, the interactions between the system and the environment are modelled as two-player turn-based games [Tho95, ALW89]: one player models the system and other the environment, and the players take moves in turn. The game is typically represented as a graph with vertices distributed between the players and the transition relation defines the moves of the players. On the other hand, the interactions between individual processes and the environment in a distributed system

can be modelled using a *concurrent game structure* [AHK98, AHK02]. In such structures, a vertex represents a global configuration of the system, and the transitions between vertices depend on the action of each component, including environment. A concurrent game arena for n agents is a directed graph, where the transitions are labelled by n-tuples of actions (or simply words of length n). At each vertex of the graph, all n agents select simultaneously and independently an action, and the next vertex is determined by the combined move consisting of all the actions (or word formed of all the actions). In both of the game structures, most often, one considers infinite duration plays, *i.e.*, plays generated by iterating this process forever. The notion of a concurrent game structure is illustrated below on a toy example.

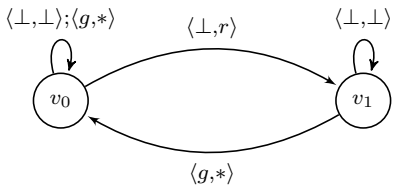


Figure 1.1: The concurrent game structure for a simple server-client model.

Example 1.1. *A simple example of a server-client model is depicted in Figure 1.1. Here, the first component belongs to the server and the other to the client. The client sends requests to the server by playing action r (and $\perp$ for no request), and the server grants (resp., does not grant) the request by playing action g (resp., $\perp$). The label '$*$' represents any action of the player. Intuitively, the vertex v_0 represents the state of the system where there is no pending requests, and v_1 denotes the situation where a request is pending; the edge labellings define the transition relation.*

One of the central problems on such game structures is to decide the existence of winning strategies for a set of agents (assuming the other components are uncontrollable) for a given objective. This is related to the (distributed) synthesis problem. For a single process system, the synthesis problem reduces to finding a finite-state winning strategy for the process player in a turn-based game. Unfortunately, this approach does not directly extend to the distributed case. However, game-theoretic approaches for the distributed synthesis problem have been considered in the literature [MW03]. Coming back to concurrent games, de Alfaro *et al.*, in [AHK98], study the problem of existence of a winning strategy for reachability objectives when there are two players in the game. In [AHK02], the authors propose a new logic, called *alternating-time temporal logic*, to specify winning behaviours of multiple agents.

Games with a parameterized number of players. Traditional games played on graphs are defined for a fixed number of players. Indeed, the transitions of a concurrent game are labelled with tuples of fixed size depending on the number of processes in the distributed system. The purpose of the second part of the thesis is to settle the foundations of concurrent games involving a parameterized number of players, paving the way to the modelling and verification of interactions involving an arbitrary number of agents. Such

games can model interactions of agents in a parameterized system, and, as mentioned earlier, these systems need to be proven correct independently of the system size. The transitions of a *parameterized concurrent game* are labelled with sets of words, each word representing an interaction between some agents. Such a parameterized arena represents infinitely many interaction situations, one for each possible number of agents. In the following, we illustrate the notion of a parameterized concurrent game on a server-client model, an extension of Example 1.1, with a parameterized number of clients.

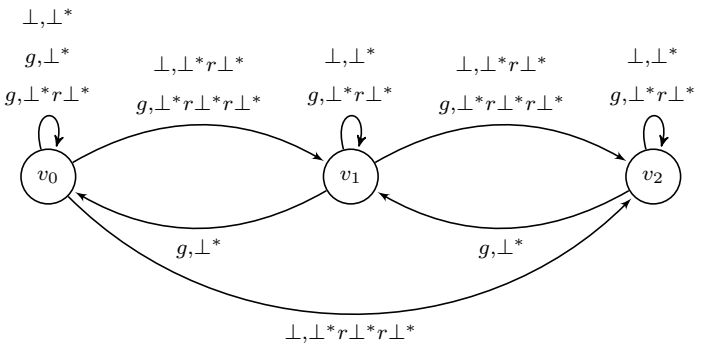


Figure 1.2: A simple server-client model with a parameterized number of clients.

Example 1.2. *An example of a server-client model with a parameterized number of clients is depicted in Figure 1.2. We assume, the server can have at most* 2 *pending requests at a certain time (otherwise the game simply moves to a sink losing vertex, not depicted here), and it can grant at most one request at a time. The actions of each client are either* r *(to send a request to the server), or* $\bot$ *(otherwise). Similarly, the server grants (resp., does not grant) the request by playing action* g *(resp.,* $\bot$*). In this example, for the sake of readability, the labels on the transitions are divided into two components: the first belongs to the server, the other to an arbitrary number of clients which is not fixed* a priori*. A word in the language on an edge represents actions of the agents, one for each of them. For instance, a word in* $\bot^* r \bot^*$ *denotes that exactly one client is sending a request. Therefore, the edge from* v_0 *to* v_1 *represents the situations when either a client is sending a request and the server does not grant the request, or two clients are sending requests and the server grants one of them. The vertex* v_0*, intuitively, indicates the state of the system where there is no request pending,* v_1 *denotes the fact that one request is pending, and at* v_2*, two requests are pending.*

Given a parameterized game and a winning condition, it is natural to ask whether a set of players – assuming there are at least that many players in the game – have a winning strategy in a parameterized game for a given specification. The set of agents for which we want to synthesize a strategy plays as a coalition against the environment (the other components of the game), and their strategies should be uniform in the sense that it should be winning irrespective of the number of players participating in the game and the choices of their opponents. In the second part of this thesis, we approach these kinds of problems on parameterized concurrent games. We summarize in the following the main contributions of the thesis.

1.2 Contributions of the thesis

The contribution of this thesis is two-fold. In the first part, we study a model of ad hoc networks, called *broadcast protocols*, of [DSZ10], whereas the second part is dedicated to the study of games on graphs extended to a parameterized setting, where the parameter is the number of players. We summarize here the main contributions of this thesis.

1.2.1 Verification of ad hoc networks

In ad hoc networks, several mobile devices are connected wirelessly without the presence of a pre-existing infrastructure. They communicate via message broadcasts: any node can transmit data to other nodes within its transmission range. Due to frequent movement of the nodes, topology can change rapidly. Delzanno *et al.* [DSZ10] proposed an automata-based model, called *broadcast protocols*, to represent the main characteristics of ad hoc networks. A broadcast protocol is a finite state machine equipped with the following actions: broadcast of a message or reception of a message. A network is represented by an undirected graph, nodes are processes, and the edges define the communication topology. A process in the network can broadcast a message and simultaneously, the adjacent processes (those who are connected to the broadcasting node) may receive the message. We consider various types of communication topology: static (the topology is fixed throughout an execution), reconfigurable (the topology can evolve non-deterministically), loss-on-reception (a message can be lost during reception), and loss-on-broadcast (a message can be lost during broadcast), the last one being a contribution of this part. We are interested in parameterized verification of ad hoc networks, where the parameter is the number of processes. In other words, in the problems we consider here, the number of nodes in the initial configuration is not fixed *a priori*, and given a property, we ask whether it is satisfied for any initial configuration, and in particular independently of the number of nodes and communication topology.

Among various decision problems considered in the literature, we are mostly interested in the *coverability problem*. Given a broadcast protocol $\mathcal{P}$, and a set of (unsafe) states F, the problem asks whether there exists an initial configuration, and an execution reaching a final configuration such that at least one process in the final configuration is in a state of F. When considering some kind of error states, a positive instance of the above corresponds to a network that exhibits a bad behaviour. Although the problem in general is undecidable for networks with static topology, decidability can be shown for reconfigurable [DSZ10, DSTZ12] and loss-on-reception networks [DSZ12]. Other decidability results were also shown, for instance, for node failures [DSZ12], for clique topologies in static networks [EFM99], for networks of k-bounded path topologies, where any simple path in the network has length at most k [DSZ10], etc.

As mentioned earlier, a tempting approach towards the decidability of the PMCP is to show that the system has a cutoff. This approach is relevant for the reconfigurable

and lossy semantics of broadcast protocols, since they enjoy a monotonicity property: if a state can be covered from a configuration, it can also be covered from any configuration with more nodes. We call this property the *copycat property*. The main contribution in this part of the thesis is the study of two quantities, *cutoff*, and *covering length*, of broadcast protocols, that determines how large and how fast some states of the protocol can be covered (for positive instances of coverability). In other words, the cutoff is the minimal number of processes in the network for which a covering execution exists; and the covering length is the minimal number of steps in a covering execution. In parameterized verification, the study of the notion cutoff comes quite naturally. One can expect that, for some parameterized systems, if a bug appears with a large number of processes, then it already appears in another instantiation of the system with small number of processes. Unfortunately, this is not the case in general, examples were shown in [AKR+14]. However, using the copycat property of the reconfigurable and lossy ad hoc networks, we will show that in these semantics, there is indeed a cutoff for positive instances of coverability.

We will show a linear upper bound on the cutoff and a quadratic upper bound on the covering length for positive instances of coverability in the reconfigurable and lossy ad hoc networks. In other words, we show that if there is a covering execution in those semantics, then there is one that has at most linear number of nodes and the length of the execution is at most quadratic in the size of the protocol. The proof goes by a fine analysis of a modified version of the saturation algorithm for coverability in reconfigurable semantics [DSTZ12]. We further show that the upper bounds for both cutoff and covering length are tight: we exhibit a family of protocols that achieves these bounds. While the loss-on-reception semantics is inter-reducible to the reconfigurable semantics [DSZ12], we show that for loss-on-broadcast semantics, the set of coverable states coincides with that of reconfigurable semantics. However, the executions in loss-on-broadcast semantics can in general be more succinct: there exists a family of protocols such that in reconfigurable semantics, the cutoff is 3, yet in loss-on-broadcast semantics, the cutoff is linear in the size of the protocol. Finally, we consider the MinNodes problem of deciding the minimal size of a covering execution and show that it is an NP-complete problem: the NP upper bound follows from a guess-and-check algorithm that non-deterministically guesses an execution of at most linear size and quadratic length efficiently; and the lower bound is achieved by a reduction from the SetCover problem which is known to be an NP-complete problem [Kar72].

1.2.2 Parameterized concurrent games

In the second part, we study *parameterized games*, a model of games on graphs, where the parameter is the number of players. These games extend classical two-player concurrent games of [AHK98] to the setting where the number of players are not fixed *a priori*. While in classical concurrent games, edges of the graph are labelled with a tuple of actions, one for each player, in parameterized games, we replace it by (regular) languages of finite but possibly unbounded words. For instance, the label a^+ represents that all players choose action a, while ab^+ is the situation where the first player chooses a, while all other players

play b. Such a parameterized arena can therefore represent infinitely many interaction situations, one for each possible number of agents. Given a winning condition described by a set of vertices of the graph, we are interested in typical decision problems, such as the existence of winning strategies for a (non-empty) collection of players. The set of agents for which we want to synthesize a strategy plays as a coalition, and their strategies should be uniform: it should be winning for any number of players and all possible choices of their opponents. In this work, we consider two particular cases: (1) when the set is singleton, *i.e.*, a distinguished player is trying to achieve a given objective against the coalition of her opponents, not knowing *a priori* how many they are; and (2) when the set consists of all players, *i.e.*, the players as a coalition are trying to achieve a common objective, not knowing *a priori* how many they are. Another subtlety here is that the arena need not necessarily be deterministic. Moreover, the number of players and the non-determinism of the arena are resolved by an antagonistic environment.

In the first setting, where a distinguished player, called Eve, wants to achieve an objective against her opponents, we show it reduces to a simpler model where the edges are labelled with languages that only constrain the number of her opponents, called *semi-parameterized games*. Intuitively, since Eve has to play uniformly that should be winning against any number of her opponents and their strategies, the knowledge of the number of them is necessary information to her, in fact we show that it is also sufficient. In semi-parameterized arenas, the number of Eve's opponents are described as constraints, and in this work, we consider the constraints as (finite unions of disjoint) intervals or semilinear predicates. Our main contribution in this setting is to show complexity bounds for the decision problem of the existence of a winning strategy for Eve for a reachability objective. As parameterized games reduce in polynomial time to semi-parameterized ones with semilinear predicates, we prove the complexity results for the latter. We show that the problem is PSPACE-complete for semilinear predicates or finite unions of intervals. When constraints are finite unions of intervals, but the arena is restricted to a deterministic one, the problem is NP-complete. Finally, for only interval constraints, the decision problem is PTIME-complete. The upper bound results are achieved in two steps: first, from a semi-parameterized arena, we construct the *knowledge game*, a two-player turn-based game, in which every vertex of Eve is equipped with a subset of integers that describes her knowledge of the number of opponents at the corresponding vertex in the original game. The decidability of semi-parameterized game immediately follows from the decidability of the knowledge game. Second, the PSPACE upper bound derives from a depth-first search algorithm on an exponential size tree, non-trivially extracted from the knowledge game. The NP upper bound for deterministic arenas with constraints as finite unions of intervals follows from the fact that if there is a winning strategy for Eve, then there is one which is polynomially bounded in the size of the arena. The lower bound results are shown by reductions from QBF-SAT and 3SAT, that are known to be PSPACE-complete [SM73] and NP-complete [Coo71], respectively.

The second contribution in this part of the thesis is to study a coalition setting for parameterized games. In this setting, players choose their actions such that collectively they achieve a common objective, irrespective of their number. There is no explicit communication between the players, and they do not have any prior information about

their number, however, they can infer knowledge about the number of them as the game evolves. While the problem seems quite non-trivial even for a reachability objective, we consider here a simpler objective - a safety objective, and show decidability and complexity results of the decision problem of the existence of a coalition winning strategy. To show decidability, we first consider an unfolding of the arena with the condition that we stop exploring a branch if either some vertex repeats in that branch, or an unsafe vertex is reached. Intuitively, in the former case, one can consider the same coalition strategy as in the first occurrence of the vertex, and in the latter, that play is immediately losing, so we do not need to explore that branch. The tree can be exponential in size, and the bound is tight. After showing the correctness of the construction, we consider the coalition game on this finite tree and give an algorithm which outputs yes if and only if there is a winning coalition strategy that runs in exponential space in the size of the game. The algorithm constructs a safety automaton $\mathcal{B}$ that runs in parallel all input automata labelling the edges in the parameterized game. Then we show that an accepting run in $\mathcal{B}$ corresponds to a winning coalition strategy in the parameterized game. As a bonus, we show that one can synthesize a coalition winning strategy also in exponential space. Furthermore, the size of the memory of a winning strategy is at most exponential. We finally show that the coalition game problem is PSPACE-hard, by a reduction from QBF-SAT, which is known to be a PSPACE-complete problem [SM73].

1.3 Organization of the thesis

The thesis is divided in two parts: Part I studies broadcast protocols, a model of ad hoc networks, whereas Part II is on parameterized games on graphs. Each part contains a self-contained Preliminaries chapter that recalls related notions and results from literature, and also defines the new models and problems we are interested in (Chapters 2 and 4). We show the results on cutoff and covering length for coverability in a broadcast protocol in Chapter 3. Among two game settings targetted in Part I, Chapter 5 studies the first one where Eve wants to achieve an objective against any number of opponents and their strategies, and Chapter 6 discusses the other where the players as a coalition want to achieve a common objective. Each contribution chapter (Chapters 3, 5 and 6) contains a dedicated conclusion that discusses possible future works in that particular topic. In addition, we discuss more general perspectives in Chapter 7.

Part I

Verification of Ad Hoc Networks

Chapter 2

Preliminaries

Ad Hoc Networks. An *ad hoc network* is a *local area network* (LAN) of autonomous mobile nodes connected by wireless links. They are decentralized type of networks, *i.e.*, they do not depend on any pre-existing infrastructure such as routers, they allow several portable computing devices to establish networks on-the-fly. Each *node* can transmit data to other nodes within its transmission range - such nodes are called *(single-hop) neighbours* of the transmitting node. Data transmissions to other nodes therefore have to be routed through some intermediate nodes. Further, nodes can move freely, leading to changes in the network topology - this special property of the network makes them difficult to model. There have been extensive studies to the development of protocols for ad hoc networks including [JM96, PB99, NH06, SRS08, SRS09]. The study of verification problems for networks of arbitrary size and unknown topology is an interesting and challenging problem for this class of systems.

Broadcast protocols. In [DSZ10], the authors propose an automata-based model, called *broadcast protocols*, to represent the main characteristics of ad hoc networks. A network is represented by an undirected graph, nodes are processes, and the edges define the communication topology. If a node n is connected to n' in the graph, it denotes that n' is a (single-hop) neighbour of n, and vice-versa. Finally, the nodes communicate by message broadcasts: all neighbours of the broadcasting node may receive the message (in some semantics, messages may be lost in the networks). We note that the model was previously studied by Esparza *et al.* [EFM99], where nodes communicate via rendez-vous communications or message broadcasts, but for message broadcasts, it reaches all other nodes in the network.

We are interested in parameterized verification of various properties of broadcast protocols. In particular, the size of the network is not fixed *a priori*, and can possibly be unbounded, *i.e.*, the number of nodes is the parameter. We consider, in this part of the thesis, various semantics of broadcast protocols, depending on (1) how the communication topology evolves, and (2) whether a message is non-deterministically lost in the network. Other semantics were also considered in the literature depending on, for example, whether a node crashes during an execution [DSZ12], etc.

Coverability problem. Parameterized verification of ad hoc networks amounts to checking if a given property holds independently of the initial configuration, and in particular, independently of the number of nodes and the communication topology. Several verification problems on this model have been studied in the literature, in which *coverability* is of special interest. Intuitively, the problem asks whether there exists an initial configuration and an execution leading to a configuration in which some node is in a given local state. When considering error states, a positive instance for the coverability problem thus corresponds to a network that can exhibit a bad behaviour.

The coverability problem, along with other verification problems, has been extensively studied in the literature for various semantics of broadcast protocols [DSZ10, DSZ11, DSTZ12] For instance, one can consider *static* topology where the connection between any two nodes remains unchanged throughout an execution, whereas in *reconfigurable* semantics, the connection relation may evolve arbitrarily. One can also consider lossy semantics, where a message can be lost non-deterministically in the network, either on reception or on broadcast. The problem has also been studied for different connectivity graphs, for instance, *cliques* [EFM99, DSZ11], *bounded path graphs* [DSZ10], *bounded diameter graphs* [DSZ11] etc.

Cutoff and covering length. A well-known approach towards solving a parameterized model checking problem is to check if a cutoff exists: if the property holds with a specific number of processes, then it also holds for any larger number of processes; and further finding this specific number where it exists. If it exists, then the parameterized model checking problem reduces to the model checking of finitely many systems; indeed, it is then enough to verify the property for the system with at most m number of processes (where m is the cutoff of the system for that property). This approach is particularly relevant for the coverability problem in the case of reconfigurable and lossy ad hoc networks since, as we will formally show later in this part of the thesis, they enjoy a monotonicity property: if a state can be covered from a configuration, it can also be covered from any configuration with more nodes. We are therefore interested in finding upper and lower bounds on the *cutoff* for positive instances of the coverability problem in the reconfigurable and lossy semantics. Furthermore, we introduce here a similar notion, called *covering length*, that measures the number of transitions needed for a state of the protocol to be covered. While considering error states, the covering length thus measures the number of steps for which a network will for sure be correct; in other words, how fast a network execution can go wrong. More precisely, for positive instances of the coverability problem, the *cutoff* is the minimal number of processes in the network for which a covering execution exists; and the *covering length* is the minimal number of steps for a covering execution. Thus, if one can show upper bounds m and l for cutoff and covering length, respectively, then given a set of states F, one checks if a state in F is coverable by an execution with at most m nodes, moreover, one only considers executions of length at most l. This may lead to efficient algorithms for parameterized model checking of coverability for ad hoc networks.

This part of the thesis is based on the publications [BBM19b] and [BBM21] co-authored with Nathalie Bertrand and Patricia Bouyer appeared in the proceedings of CONCUR

2019 and the journal LMCS 2021, respectively.

Related work

In [DSZ10], the authors study parameterized verification problems for ad hoc networks under different semantics. The authors show that the coverability problem is undecidable for general topology in static ad hoc networks, *i.e.*, when the communication topology is arbitrary but does not evolve over time. However, the problem becomes decidable for more restricted topologies, for instance, for *cliques* [EFM99], for *k-bounded path topologies* (any simple path in the network has length at most k) [DSZ10]. It is, however, still undecidable for graphs with *k-bounded diameter* (the shortest path between any two nodes has length at most k) for $k > 1$ [DSZ11].

The decidability of coverability can also be recovered by allowing non-deterministic reconfigurations of the communication topology. Moreover, under this semantics, the coverability problem is in PTIME [DSTZ12]. In [DSZ12], the authors study communication failures in the network, assuming non-deterministic message losses could happen upon reception and node failures, where a node can join or crash during any step along an execution. As observed by the authors, the coverability problem for such networks reduces to the coverability problem in reconfigurable networks.

Extensions of ad hoc networks with time constraints, registers, probabilistic transitions, etc. have been considered in literature. Verification of timed ad hoc networks was studied in [ADR+11]. In such networks, each node is equipped with a set of clocks, and a set of rules is defined in the protocol; a broadcast is performed if the clock values of the broadcasting node as well as the receiving nodes (if any) satisfy the guards of that transition; additionally, time can elapse arbitrarily for every node in the network. In [DST13], the underlying protocol is a register automaton, equipped with a set of registers. Each node in the network is labelled with a state of the protocol and register values; messages are allowed to carry data, that can be assigned to or tested against the local registers of receivers. Asynchronous broadcasts were considered in [DT13], where the broadcast and reception actions may not be synchronous: a local buffer associated to each node stores messages it receives from its neighbours during a transition, which may be processed eventually. A probabilistic variant, where a broadcast protocol is extended with probabilistic states (resp., transitions), was considered, for instance, in [BFS14].

The technique of finding a cutoff has been studied for various classes of systems with specific connection topologies among processes. Let us mention here some of them. Token-passing systems with uni-directional ring topologies was studied in [EN95] for various specifications described in *indexed* $\mathsf{CTL}^* \setminus \mathsf{X}$. Their work has later been extended in [CTTV04] for arbitrary network topologies, however, the properties of a system is specified in indexed $\mathsf{LTL} \setminus \mathsf{X}$ in the latter. Moreover, a new viewpoint of the notion of cutoff was introduced: in order to verify the parametrized system, it is enough to verify a constant number of small ones, each of which has a bounded number of processes. In [EK00], the authors study guarded protocols with disjunctive guards, or conjunctive

guards with initial state for specifications in indexed $\mathsf{LTL} \setminus \mathsf{X}$ whose atoms are indexed by the processes. In [KKW10] and [AHH13], the authors study the problem of finding cutoffs *dynamically* for a given parameterized system and a specification. However, in [AKR+14], the authors provides examples of systems for which no cutoff exists. Finally, in [HS20] and also in [BER21], the authors study the decidability of existence of a cutoff in rendez-vous networks.

Organization of the chapter

We begin with a formal introduction to broadcast protocols in Section 2.1 and elaborate its various semantics with illustrative examples in Section 2.2. This is followed by a formal description of the coverability problem for ad hoc networks with the complexity results for different semantics in Section 2.3. Further, in this section a polynomial time algorithm for coverability under reconfigurable semantics is recalled [DSTZ12]. In Section 2.4, the notion of cutoff and covering length are formally introduced. We close the chapter in Section 2.5 that announces our contributions that will be presented in the following chapter.

Notations

We denote by $\mathbb{N}$ the set of all natural numbers greater than or equal to 0 and by $\mathbb{N}_{>0}$ the set of all natural numbers strictly greater than 0. For a finite set S, S^k denotes the set of all finite sequences (or *words*) of length k; S^* denotes the set of all finite words over S; S^+ denotes the set of all non-empty finite words over S; and finally, S^ω denotes the set of all infinite words over S. For two words $u \in S^*$ and $w \in S^+ \cup S^\omega$, we write $u \sqsubseteq w$ to denote u is a prefix of w, and for any $k \in \mathbb{N}_{>0}$, $[w]_{\leq k}$ denotes the prefix of length k of w (belongs to S^k).

2.1 Broadcast protocols

In this section, we formally define broadcast protocols and its various semantics.

Definition 2.1. *A* broadcast protocol *is a tuple $\mathcal{P} = (Q, I, \Sigma, \Delta)$ where Q is a finite set of control states; $I \subseteq Q$ is the set of initial control states; Σ is a finite message alphabet; and $\Delta \subseteq (Q \times \{!\mathsf{a}, ?\mathsf{a} \mid \mathsf{a} \in \Sigma\} \times Q)$ is the transition relation.*

The label $!\mathsf{a}$ (resp., $?\mathsf{a}$) represents the broadcast (resp., reception) of the message $\mathsf{a} \in \Sigma$. For ease of readability, we often write $q \xrightarrow{!\mathsf{a}} q'$ (resp., $q \xrightarrow{?\mathsf{a}} q'$) for $(q, !\mathsf{a}, q') \in \Delta$ (resp., $(q, ?\mathsf{a}, q') \in \Delta$). We will assume the protocol to be complete for receptions: for every $q \in Q$ and $\mathsf{a} \in \Sigma$, there exists q' such that $q \xrightarrow{?\mathsf{a}} q'$.

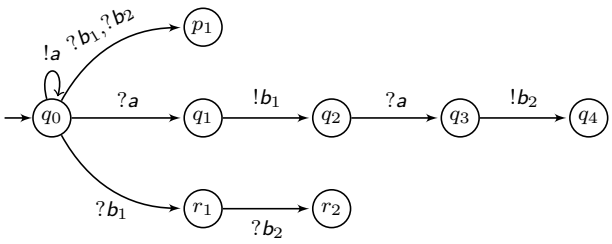


Figure 2.1: Example of a broadcast protocol.

Example 2.2. *A broadcast protocol is represented in Figure 2.1. In this and further examples, for concision purposes, we assume that if the reception of a message is unspecified from some state, it implicitly represents a self-loop. This will be our running example to explain various semantics of broadcast protocols; in the sequel, we will generalize this example to prove some of our results on the bounds for the cutoff and covering length of broadcast protocols.*

Here, the set of states $Q = \{p_1\} \cup \{q_0, q_1, q_2, q_3, q_4\} \cup \{r_1, r_2\}$*, and the set of actions* $\Sigma = \{a, b_1, b_2\}$*. The transition relation is defined as follows:* $q_0 \xrightarrow{!\mathsf{a}} q_0$*,* $q_0 \xrightarrow{?\mathsf{b}_1} r_1$*,* $r_1 \xrightarrow{?\mathsf{b}_i} r_2$*, and for every* $1 \leq i \leq 2$*,* $q_0 \xrightarrow{?\mathsf{b}_i} p_1$*,* $q_{2i-2} \xrightarrow{?\mathsf{a}} q_{2i-1}$*,* $q_{2i-1} \xrightarrow{!\mathsf{b}_i} q_{2i}$*. For example, here from* q_0*, broadcasting message* a *leads to* q_0 *again, and receiving the message* a *leads to* q_1*. Broadcast protocols can in general be non-deterministic, for instance, in this example, at* q_0*, there is a non-deterministic choice on reception of message* b_1*. Also note that the self-loops upon receiving any message, when not already indicated, are implicit; for example here,* $q_1 \xrightarrow{?\mathsf{a}} q_1$ *is such a transition.*

Remark 2.3. *Note that our definition of broadcast protocols slightly differs from the one considered in [DSZ10], where* Δ *also allows* internal actions *of the form* $q \xrightarrow{\tau} q'$*, where* τ *is a special symbol in* Σ *representing an internal action. However, these two definitions are equivalent (under all semantics we will consider in this thesis), since one can simulate internal actions by message broadcasts and receptions: for* $q \xrightarrow{\tau} q'$ *an internal action, add transitions* $q \xrightarrow{!\tau} q'$*, and for every* $q \in Q$*,* $q \xrightarrow{?\tau} q$ *in* Δ*.*

2.2 Semantics

Ad hoc networks comprises several *nodes* that execute the same broadcast protocol. A configuration is represented by an undirected graph whose nodes are labelled with states in Q. Transitions between configurations happen by broadcasts from a node to its neighbours.

Formally, given a broadcast protocol $\mathcal{P} = (Q, I, \Sigma, \Delta)$, a *configuration* is an undirected 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 symmetric and irreflexive relation describing the set of edges; finally, $\mathsf{L} \colon \mathsf{N} \to Q$ is the labelling function. We let $\Gamma(\mathcal{P})$ denote the (infinite) set of configurations of the broadcast protocol $\mathcal{P}$. Given a configuration $\gamma = (\mathsf{N}, \mathsf{E}, \mathsf{L}) \in \Gamma(\mathcal{P})$, we write $\mathsf{n} \sim \mathsf{n}'$ whenever $(\mathsf{n}, \mathsf{n}') \in \mathsf{E}$, and we let

$\mathsf{Neigh}_\gamma(\mathsf{n}) = \{\mathsf{n}' \in \mathsf{N} \mid \mathsf{n} \sim \mathsf{n}'\}$ be the neighbourhood of n, *i.e.* the set of nodes adjacent to n. Finally, $\mathsf{L}(\gamma) = \cup_{\mathsf{n}\in\mathsf{N}}\mathsf{L}(\mathsf{n})$ denotes the set of labels appearing in nodes of γ. A configuration $\gamma = (\mathsf{N}, \mathsf{E}, \mathsf{L})$ is called *initial* if $\mathsf{L}(\mathsf{N}) \subseteq I$, *i.e.* every node is in an initial state.

In the following, we define various semantics of broadcast protocols.

2.2.1 Static ad hoc networks

The operational semantics of a static ad hoc network for a given broadcast protocol $\mathcal{P}$ is an infinite-state transition system $\mathcal{T}(\mathcal{P})$. Intuitively, each node of a configuration runs protocol $\mathcal{P}$, and may send/receive messages to/from its neighbours. From a configuration $\gamma = (\mathsf{N}, \mathsf{E}, \mathsf{L})$, there is a *step* to $\gamma' = (\mathsf{N}, \mathsf{E}, \mathsf{L}')$ if there exists $\mathsf{n} \in \mathsf{N}$ and $\mathsf{a} \in \Sigma$ such that $(\mathsf{L}(\mathsf{n}), !\mathsf{a}, \mathsf{L}'(\mathsf{n})) \in \Delta$, and for every $\mathsf{n}' \in \mathsf{N}$, if $\mathsf{n}' \in \mathsf{Neigh}_\gamma(\mathsf{n})$, then $(\mathsf{L}(\mathsf{n}'), ?\mathsf{a}, \mathsf{L}'(\mathsf{n}')) \in \Delta$, otherwise, $\mathsf{L}'(\mathsf{n}') = \mathsf{L}(\mathsf{n}')$: a step thus reflects how nodes evolve when one of them broadcasts a message to its neighbours. We then write $\gamma \xrightarrow{\mathsf{n},!\mathsf{a}}_{\mathtt{s}} \gamma'$, or simply $\gamma \to_{\mathtt{s}} \gamma'$ (the $\mathtt{s}$ subscript emphasizes that the communication topology is *static*). We denote by $\xrightarrow{*}_{\mathtt{s}}$ the reflexive and transitive closure of $\to_{\mathtt{s}}$. Note here that the number of nodes is the same in a step and the connection topology is fixed.

An *execution* of the static ad hoc network is a sequence $\rho = (\gamma_i)_{0\leq i\leq t}$ of configurations $\gamma_i = (\mathsf{N}, \mathsf{E}, \mathsf{L}_i)$ such that γ_0 is an initial configuration, and for every $0 \leq i < t$, $\gamma_i \to_{\mathtt{s}} \gamma_{i+1}$. We write $\#\mathsf{nodes}(\rho)$ for the number of nodes $|\mathsf{N}|$ in γ_0, $\#\mathsf{steps}(\rho)$ for the number t of steps along ρ, and for any node $\mathsf{n} \in \mathsf{N}, \#\mathsf{steps}(\rho, \mathsf{n})$ for the number of broadcasts, called the *active length*, that node n performs along ρ. The set of all static executions is denoted $\mathtt{Exec}_{\mathtt{s}}(\mathcal{P})$. Here again, the subscript $\mathtt{s}$ indicates static semantics.

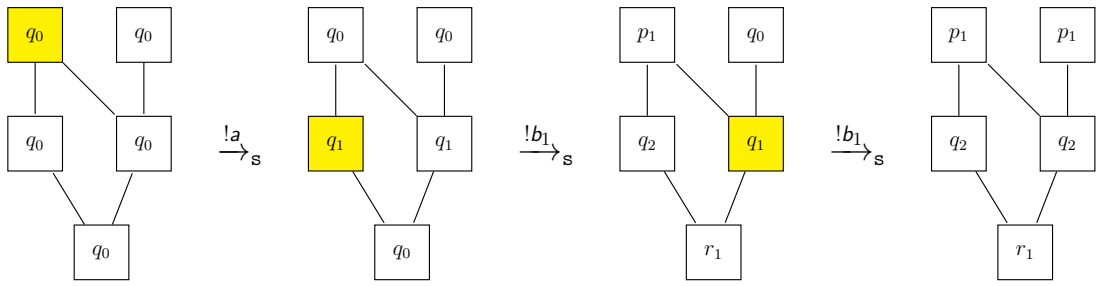


Figure 2.2: A sample static execution for the protocol from Figure 2.1.

Example 2.4. *We provide an example of a static execution in Figure 2.2 for the broadcast protocol of Figure 2.1. Nodes are represented by squares, and the labels are written within the nodes. For simplicity, the node names are not given. Note that the communication topology is the same throughout the execution. In the example, the yellow nodes broadcast a message to its neighbours in the step leading to the next configuration.*

2.2.2 Reconfigurable ad hoc networks

In a *reconfigurable* ad hoc network, the communication topology of a configuration may change non-deterministically along an execution.

Transitions between configurations are consistent with the static semantics, except now the communication topology can evolve non-deterministically after performing a broadcast. Formally, from a configuration $\gamma = (\mathsf{N}, \mathsf{E}, \mathsf{L})$, there is a step to $\gamma' = (\mathsf{N}, \mathsf{E}', \mathsf{L}')$ if there exists $\mathsf{n} \in \mathsf{N}$ and $\mathsf{a} \in \Sigma$ such that $(\mathsf{L}(\mathsf{n}), !\mathsf{a}, \mathsf{L}'(\mathsf{n})) \in \Delta$, and for every $\mathsf{n}' \in \mathsf{N}$, if $\mathsf{n}' \in \mathsf{Neigh}_\gamma(\mathsf{n})$, then $(\mathsf{L}(\mathsf{n}'), ?\mathsf{a}, \mathsf{L}'(\mathsf{n}')) \in \Delta$, otherwise $\mathsf{L}'(\mathsf{n}') = \mathsf{L}(\mathsf{n}')$: a step thus reflects that the communication topology may change unconditionally from E to E' after the broadcast of a message from a node to its neighbours in the old topology. For such a step, we write $\gamma \xrightarrow{\mathsf{n},!\mathsf{a}}_{\mathtt{r}} \gamma'$, or simply $\gamma \to_{\mathtt{r}} \gamma'$ (the $\mathtt{r}$ subscript emphasizes that the communication topology is *reconfigurable*). We denote by $\xrightarrow{*}_{\mathtt{r}}$ the reflexive and transitive closure of $\to_{\mathtt{r}}$. An execution is defined similarly as in the static semantics. We write $\mathtt{Exec}_{\mathtt{r}}(\mathcal{P})$ for the set of all reconfigurable executions in $\mathcal{P}$ (subscript $\mathtt{r}$ for reconfigurable semantics). For an execution ρ, similarly to the static semantics, $\#\mathsf{nodes}(\rho)$ is the number of nodes in ρ, $\#\mathsf{steps}(\rho)$ is the number of steps, and $\#\mathsf{steps}(\rho, \mathsf{n})$ is the number of broadcasts by node n along ρ.

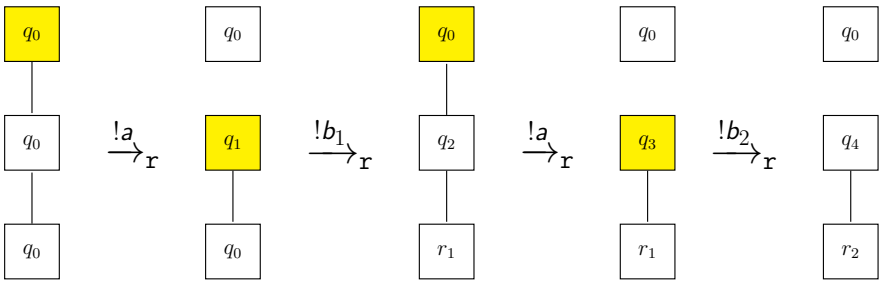


Figure 2.3: A sample reconfigurable execution for the broadcast protocol from Figure 2.1.

Example 2.5. *Figure 2.3 gives an example of a reconfigurable execution for the broadcast protocol of Figure 2.1. Note that the communication topology indeed evolves along the execution. As before, the yellow nodes broadcast a message in the step leading to the next configuration.*

Remark 2.6. *Note that our definition of a reconfigurable step slightly differs from an equivalent one which is often found in the literature, for instance in [DSTZ12], where the broadcast of a message and the non-deterministic reconfiguration of the communication topology are two separate steps. Clearly, these two notions are equivalent.*

2.2.3 Lossy ad hoc networks - loss-on-message reception

Under this semantics, a message can be lost non-deterministically during a step upon reception: after a message is broadcast, some neighbours of the broadcasting node may not receive it. Yet, the communication topology is fixed throughout an execution. However,

we will later show that one can simulate an execution in this semantics by one in the reconfigurable semantics such that the node labels in the final configuration are unchanged and vice-versa. This semantics was introduced and studied in [DSZ12]. Let us first define the semantics formally.

A configuration is defined the same way as under the static (or reconfigurable) semantics and let $\Gamma(\mathcal{P})$ denotes the set of all configurations. From a configuration $\gamma = (\mathsf{N}, \mathsf{E}, \mathsf{L})$, there is a step to $\gamma' = (\mathsf{N}, \mathsf{E}, \mathsf{L}')$ if there exists $\mathsf{n} \in \mathsf{N}$ and $\mathsf{a} \in \Sigma$ such that $(\mathsf{L}(\mathsf{n}), !\mathsf{a}, \mathsf{L}'(\mathsf{n})) \in \Delta$, and for every $\mathsf{n}' \in \mathsf{N}$, if $\mathsf{n}' \in \mathsf{Neigh}_\gamma(\mathsf{n})$, then either (a) $(\mathsf{L}(\mathsf{n}'), ?\mathsf{a}, \mathsf{L}'(\mathsf{n}')) \in \Delta$, or (b) $\mathsf{L}'(\mathsf{n}') = \mathsf{L}(\mathsf{n}')$ (this neighbour has not received the message, it has been lost); otherwise $\mathsf{L}'(\mathsf{n}') = \mathsf{L}(\mathsf{n}')$. A step thus reflects that a message may be lost at some nodes before receiving. Observe that the communication topology remains fixed during a transition. We write $\gamma \xrightarrow{\mathsf{n},!\mathsf{a}}_{\mathtt{lr}} \gamma'$ or simply $\gamma \to_{\mathtt{lr}} \gamma'$ (the subscript $\mathtt{lr}$ denotes the loss-on-reception semantics). We denote by $\xrightarrow{*}_{\mathtt{lr}}$ the reflexive and transitive closure of $\to_{\mathtt{lr}}$. An execution is defined similarly as in the static or reconfigurable semantics. We write $\mathtt{Exec}_{\mathtt{lr}}(\mathcal{P})$ for the set of all loss-on-reception executions in $\mathcal{P}$ (subscript $\mathtt{lr}$ for loss-on-reception semantics). For an execution ρ, similarly to the other semantics, $\#\mathsf{nodes}(\rho)$ is the number of nodes in ρ, $\#\mathsf{steps}(\rho)$ is the number of steps, and $\#\mathsf{steps}(\rho, \mathsf{n})$ is the number of broadcasts by node n along ρ.

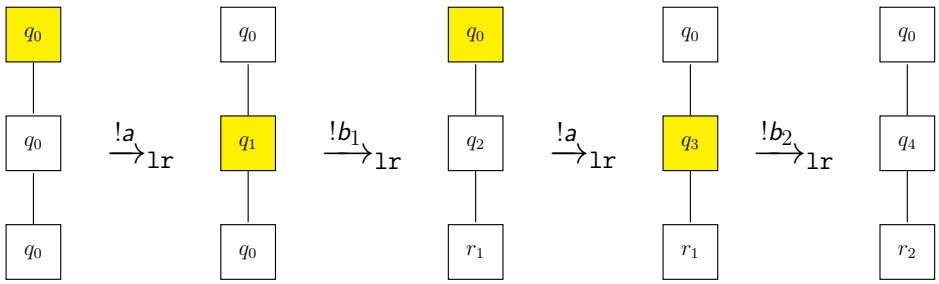


Figure 2.4: Example of a loss-on-reception execution of the protocol from Figure 2.1.

Example 2.7. *Figure 2.4 gives an example of a loss-on-reception execution for the broadcast protocol of Figure 2.1. As before, the yellow nodes broadcast a message in the step leading to the next configuration. Note that the communication topology indeed is fixed (*i.e.*, static) along the execution, however for instance, in the second and the last step, some neighbours of the broadcasting node do not receive the message and stay in the same states. Note here that this execution has the same number of nodes with same node labels as in Figure 2.3.*

However, a lossy execution can be simulated by a reconfigurable one and vice versa. Intuitively, a reconfigurable step can be considered as a lossy one with a complete graph topology where the non-neighbours of the broadcasting node lose the message, and conversely, given a lossy step, we can construct a reconfigurable one where only the nodes which have successfully received a message during the lossy step are connected to the broadcasting node in the corresponding reconfigurable step. The formal proof involves a bit more technicality, which we present in the following.

Lemma 2.8. *Let $\mathcal{P}$ be a broadcast protocol. Then $\exists \rho = \gamma_0 \to_{\mathtt{r}} \gamma_1 \cdots \to_{\mathtt{r}} \gamma_t = (\mathsf{N}, \mathsf{E}, \mathsf{L}) \in \mathtt{Exec}_{\mathtt{r}}(\mathcal{P})$ if and only if $\exists \rho' = \gamma'_0 \to_{\mathtt{lr}} \gamma'_1 \cdots \to_{\mathtt{lr}} \gamma'_t = (\mathsf{N}, \mathsf{E}', \mathsf{L}) \in \mathtt{Exec}_{\mathtt{lr}}(\mathcal{P})$.*

Proof. ($\Rightarrow$) First assume $\rho = \gamma_0 \to_{\mathtt{r}} \cdots \gamma_t$ is an execution in the reconfigurable semantics with $\gamma_i = (\mathsf{N}, \mathsf{E}_i, \mathsf{L}_i)$ for every $0 \leq i \leq t$. Consider the execution in the loss-on-reception semantics $\rho' = \gamma'_0 \to_{\mathtt{lr}} \cdots \gamma'_t$ such that for every $0 \leq i \leq t$, $\gamma'_i = (\mathsf{N}, \mathsf{E}', \mathsf{L}_i)$ with $\mathsf{E}' = \mathsf{N} \times \mathsf{N} \setminus \{(\mathsf{n}, \mathsf{n}) \mid \mathsf{n} \in \mathsf{N}\}$ (*i.e.*, a complete graph topology without self-loops) and the nodes in γ_i and γ'_i have the same labels. It only remains to prove that for any $0 \leq i \leq t$, $\gamma'_i \to_{\mathtt{lr}} \gamma'_{i+1}$ is indeed a valid step in the loss-on-reception semantics which we verify here.

Assume $\gamma_i \xrightarrow{\mathsf{n},!\mathsf{a}}_{\mathtt{r}} \gamma_{i+1}$. Note that for every $\mathsf{n}'' \notin \mathsf{Neigh}_{\gamma_i}(\mathsf{n})$, $\mathsf{L}_{i+1}(\mathsf{n}'') = \mathsf{L}_i(\mathsf{n}'')$. Then, indeed, in γ'_i, node n can perform the same broadcast of message a and move to $\mathsf{L}_{i+1}(\mathsf{n})$: $(\mathsf{L}_i(\mathsf{n}), !\mathsf{a}, \mathsf{L}_{i+1}(\mathsf{n})) \in \Delta$, and the neighbours of n in γ_i receive the message in the loss-on-reception semantics and all other nodes lose the message before receiving. Therefore, for every node $\mathsf{n}' \in \mathsf{N}$, if $\mathsf{n}' \in \mathsf{Neigh}_{\gamma_i}(\mathsf{n})$, $(\mathsf{L}_i(\mathsf{n}'), ?\mathsf{a}, \mathsf{L}_{i+1}(\mathsf{n}')) \in \Delta$, else, $\mathsf{L}_{i+1}(\mathsf{n}') = \mathsf{L}_i(\mathsf{n}')$. Thus, $\gamma'_i \xrightarrow{\mathsf{n},!\mathsf{a}}_{\mathtt{lr}} \gamma'_{i+1}$ is a valid step in the loss-on-reception semantics.

($\Leftarrow$) Let us now prove the other direction. Assume $\rho' = \gamma'_0 \to_{\mathtt{lr}} \cdots \gamma'_t$ is an execution in the loss-on-reception semantics with $\gamma'_i = (\mathsf{N}', \mathsf{E}', \mathsf{L}'_i)$ for each $0 \leq i \leq t$. Define $\mathsf{N} = \mathsf{N}'$. We will show by induction on i that for every $0 \leq i \leq t$, there is an execution $\rho_i = \gamma_0 \to_{\mathtt{r}} \cdots \gamma_i$ in the reconfigurable semantics with $\gamma_i = (\mathsf{N}, \mathsf{E}_i, \mathsf{L}_i)$ and for every $\mathsf{n} \in \mathsf{N}$, $\mathsf{L}_i(\mathsf{n}) = \mathsf{L}'_i(\mathsf{n})$. The base case is trivial: choose E_0 arbitrarily and set $\mathsf{L}_0(\mathsf{n}) = \mathsf{L}'_0(\mathsf{n})$ for each $\mathsf{n} \in \mathsf{N}$. To easily understand the inductive step, let us show the case for $i = 1$. Let $\gamma'_0 \xrightarrow{\mathsf{n},!\mathsf{a}}_{\mathtt{lr}} \gamma'_1$. Consider $\gamma''_0 = (\mathsf{N}, \mathsf{E}''_0, \mathsf{L}_0)$ such that the nodes, and their labels are the same as in γ_0, and, thanks to reconfiguration, $(\mathsf{n}, \mathsf{n}') \in \mathsf{E}''_0$ if and only if $\mathsf{n} \neq \mathsf{n}'$ and $\mathsf{L}'_0(\mathsf{n}') \neq \mathsf{L}'_1(\mathsf{n}')$ (*i.e.*, n' is a node different from the broadcasting node n that has received the message a from n in the corresponding lossy step without failure). Then in the reconfigurable semantics, each neighbour n' of n also receives the message a and moves to $\mathsf{L}'_1(\mathsf{n}')$. Hence, $\gamma''_0 \xrightarrow{\mathsf{n},!\mathsf{a}}_{\mathtt{r}} \gamma_1$ is a valid step in the reconfigurable semantics, additionally, for every node $\mathsf{n} \in \mathsf{N}$, $\mathsf{L}_1(\mathsf{n}) = \mathsf{L}'_1(\mathsf{n})$. With no loss of generality, set $\mathsf{E}_1 = \mathsf{E}''_0$. We obtain the execution ρ_1.

Let us now assume that we have constructed an execution $\rho_i = \gamma_0 \to_{\mathtt{r}} \cdots \gamma_i$ for some $1 \leq i < t$, and we will construct ρ_{i+1}. Let $\gamma'_i \xrightarrow{\mathsf{n},!\mathsf{a}}_{\mathtt{lr}} \gamma'_{i+1}$ be the $i+1$-th step in the loss-on-reception semantics. Consider the reconfigurable execution of length i: $\rho''_i = \gamma_0 \to_{\mathtt{r}} \cdots \gamma_{i-1} \to_{\mathtt{r}} \gamma''_i$ with $\gamma''_i = (\mathsf{N}, \mathsf{E}''_i, \mathsf{L}_i)$ such that the nodes and their labels in γ''_i are the same as in γ_i and, thanks to reconfiguration, the communication topology in γ''_i satisfies: $(\mathsf{n}, \mathsf{n}') \in \mathsf{E}''_i$ if and only if $\mathsf{n} \neq \mathsf{n}'$ and $\mathsf{L}'_i(\mathsf{n}') \neq \mathsf{L}'_{i+1}(\mathsf{n}')$ (*i.e.*, n' is a node different from the broadcasting node n that has received the message a from n in the $i+1$-th lossy step without failure). Extend ρ''_i with the braodcast $\gamma''_i \xrightarrow{\mathsf{n},!\mathsf{a}}_{\mathtt{r}} \gamma_{i+1}$ to obtain ρ_{i+1}. Then each neighbour n' of n in ρ''_i receives the message a and moves to $\mathsf{L}'_{i+1}(\mathsf{n}')$ in ρ_{i+1}, while the other node labels remain unchanged. Therefore, for every node $\mathsf{n} \in \mathsf{N}$, $\mathsf{L}_{i+1}(\mathsf{n}) = \mathsf{L}'_{i+1}(\mathsf{n})$, and set $\mathsf{E}_{i+1} = \mathsf{E}''_i$.

This concludes the proof of the lemma. □

Lemma 2.8 establishes the correspondence between reconfigurable and loss-on-reception semantics of a broadcast protocol. As we will see later, as a consequence of the above, these two semantics enjoy many similarities w.r.t. the problems considered, for instance, the *coverability problem*. In the following, we define an alternative semantics where, unlike loss-on-reception semantics, non-deterministic message losses may happen on broadcast.

2.2.4 Lossy ad hoc networks - loss-on-message broadcast

As we have witnessed in Section 2.2.3, an execution in loss-on-reception semantics can be simulated by one in reconfigurable semantics, and vice versa. In this part of the thesis, we introduce and study an alternative semantics of broadcast protocols, that is message loss upon broadcast: when a message is broadcast, it either reaches all neighbours of the sending node, or none of them; however the communication topology remains fixed. In contrast to message losses upon reception, it is not straight-forward to simulate arbitrary reconfigurations of the communication topology with such message losses. Below, the semantics is defined formally.

From a configuration $\gamma = (\mathsf{N}, \mathsf{E}, \mathsf{L})$, there is a step to $\gamma' = (\mathsf{N}, \mathsf{E}, \mathsf{L}')$ if there exists $\mathsf{n} \in \mathsf{N}$ and $\mathsf{a} \in \Sigma$ such that $(\mathsf{L}(\mathsf{n}), !\mathsf{a}, \mathsf{L}'(\mathsf{n})) \in \Delta$, and either (a) for every $\mathsf{n}' \neq \mathsf{n}$, $\mathsf{L}'(\mathsf{n}') = \mathsf{L}(\mathsf{n}')$ (no one has received the message, it has been lost), or (b) if $\mathsf{n}' \in \mathsf{Neigh}_\gamma(\mathsf{n})$, then $(\mathsf{L}(\mathsf{n}'), ?\mathsf{a}, \mathsf{L}'(\mathsf{n}')) \in \Delta$, otherwise $\mathsf{L}'(\mathsf{n}') = \mathsf{L}(\mathsf{n}')$ (every node in the neighbourhood of the broadcasting node has received the message): a step thus reflects that the broadcast message may be lost when it is sent. Observe that the communication topology remains fixed during a transition. We write $\gamma \xrightarrow{\mathsf{n},!\mathsf{a}}_{\mathtt{lb}} \gamma'$ or simply $\gamma \to_{\mathtt{lb}} \gamma'$ (the subscript $\mathtt{lb}$ denotes the loss-on-broadcast semantics). We denote by $\xrightarrow{*}_{\mathtt{lb}}$ the reflexive and transitive closure of $\to_{\mathtt{lb}}$. An execution is defined similarly as in the static or reconfigurable semantics. Similarly to the static and reconfigurable semantics, for ρ an execution, $\#\mathsf{nodes}(\rho)$ is the number of nodes in ρ, $\#\mathsf{steps}(\rho)$ is the number of steps, and $\#\mathsf{steps}(\rho, \mathsf{n})$ is the number of broadcasts (including lost ones) by node n along ρ; additionally, we write $\#\mathsf{nonlost\_steps}(\rho, \mathsf{n})$ for the number of successful broadcasts by node n along ρ. We use the notation $\mathtt{Exec}_{\mathtt{lb}}(\mathcal{P})$ for the set of all loss-on-broadcast executions in $\mathcal{P}$ (subscript $\mathtt{lb}$ stands for loss-on-broadcast semantics).

Example 2.9. *Figure 2.5 gives an example of a loss-on-broadcast execution for the broadcast protocol of Figure 2.1. Note that the topology is fixed throughout the execution. As before, the coloured nodes broadcast a message in the step leading to the next configuration. A lossy step is shown in the third transition: the yellow node performs a lossy broadcast, that is to say in this step none of the nodes in its neighbourhood receives the message and their states remain unchanged, emphasized in the figure by the subscript "lost".*

We now show that a loss-on-broadcast execution can be simulated by a reconfigurable one. Intuitively, if a node performs a real broadcast, it also does in reconfigurable semantics and if it performs a lossy broadcast, we disconnect every node and take the same broadcast so that no other node is affected in the reconfigurable semantics. This can be formalized as follows.

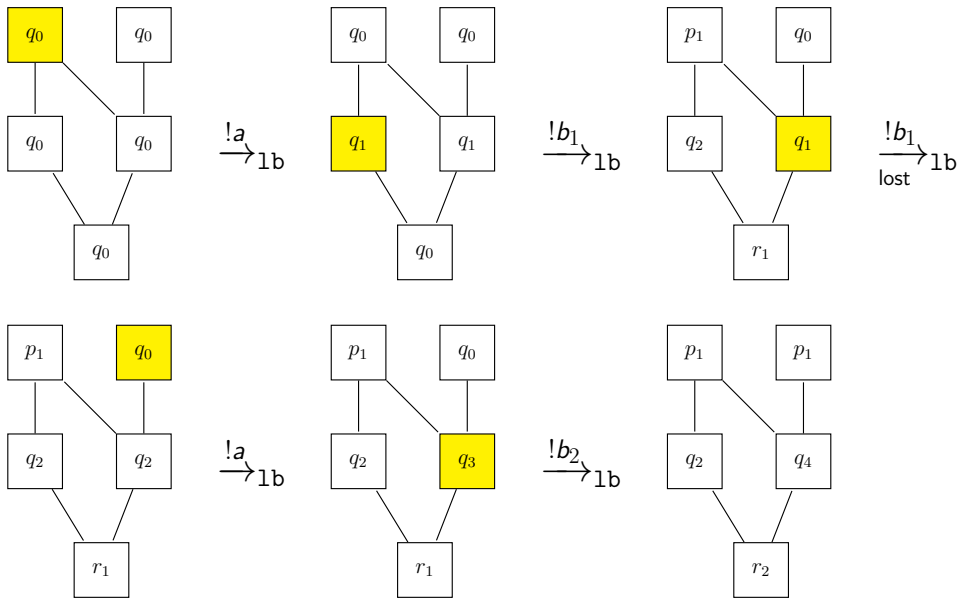


Figure 2.5: Example of a loss-on-broadcast execution of the protocol from Figure 2.1.

Lemma 2.10. *Let $\mathcal{P}$ be a broadcast protocol $\mathcal{P}$. For any $\rho = \gamma_0 \to_{\texttt{lb}} \gamma_1 \cdots \to_{\texttt{lb}} \gamma_t$ an execution in $\texttt{Exec}_{\texttt{lb}}(\mathcal{P})$, with $\gamma_t = (\mathsf{N}, \mathsf{E}, \mathsf{L})$, there is an execution $\rho' = \gamma'_0 \to_{\texttt{r}} \gamma'_1 \cdots \to_{\texttt{r}} \gamma'_t$ in $\texttt{Exec}_{\texttt{r}}(\mathcal{P})$ of same length t, with $\gamma'_t = (\mathsf{N}, \mathsf{E}', \mathsf{L})$.*

Proof. Assume $\rho = \gamma_0 \to_{\texttt{lb}} \gamma_1 \cdots \to_{\texttt{lb}} \gamma_t$ is an execution in the loss-on-broadcast semantics with $\gamma_i = (\mathsf{N}, \mathsf{E}, \mathsf{L}_i)$ for every $0 \leq i \leq t$. We will show by induction on i that for every $0 \leq i \leq t$, there is an execution $\rho'_i : \gamma'_0 \to_{\texttt{r}} \gamma'_1 \cdots \to_{\texttt{r}} \gamma'_i$ in the reconfigurable semantics with $\gamma'_i = (\mathsf{N}, \mathsf{E}'_i, \mathsf{L}'_i)$ such that for every $\mathsf{n} \in \mathsf{N}$, $\mathsf{L}'_i(\mathsf{n}) = \mathsf{L}_i(\mathsf{n})$. The base case is trivial: choose E'_0 arbitrarily and set $\mathsf{L}'_0(\mathsf{n}) = \mathsf{L}_0(\mathsf{n})$ for every $\mathsf{n} \in \mathsf{N}$.

Let us now assume that we have constructed an execution $\rho'_i = \gamma'_0 \to_{\texttt{r}} \cdots \gamma'_i$ for some $1 \leq i < t$, and we will construct ρ'_{i+1}. We use a similar idea as in the proof of Lemma 2.8. Let $\gamma_i \xrightarrow{\mathsf{n},!a}_{\texttt{lb}} \gamma_{i+1}$ be the $i+1$-th step in the loss-on-broadcast semantics. Consider the reconfigurable execution of length i: $\rho''_i = \gamma'_0 \to_{\texttt{r}} \cdots \gamma'_{i-1} \to_{\texttt{r}} \gamma''_i$, with $\gamma''_i = (\mathsf{N}, \mathsf{E}''_i, \mathsf{L}'_i)$, such that the nodes and their labels in γ''_i are same as in γ'_i and, thanks to reconfiguration, the communication topology in γ''_i satisfies the following:

- if n performs a real broadcast in the loss-on-broadcast semantics, then $(\mathsf{n}, \mathsf{n}') \in \mathsf{E}''_i$ if and only if $(\mathsf{n}, \mathsf{n}') \in \mathsf{E}_i$ (*i.e.*, n has the same set of neighbours in both semantics). Then consider the transition $\gamma''_i \xrightarrow{\mathsf{n},!a}_{\texttt{r}} \gamma'_{i+1}$ in reconfigurable semantics, each neighbour n' of n in γ''_i receives the message a and moves to $\mathsf{L}_{i+1}(\mathsf{n}')$ in γ'_{i+1}, while the other node labels remain unchanged. Therefore, for every node $\mathsf{n} \in \mathsf{N}$, $\mathsf{L}'_{i+1}(\mathsf{n}) = \mathsf{L}_{i+1}(\mathsf{n})$, and with no loss of generality, set $\mathsf{E}'_{i+1} = \mathsf{E}''_i$.
- if n performs a lossy broadcast, then $\mathsf{E}''_i = \emptyset$ (*i.e.*, nodes are disconnected from each other). Then consider the transition $\gamma''_i \xrightarrow{\mathsf{n},!a}_{\texttt{r}} \gamma'_{i+1}$ in reconfigurable semantics, n

performs the same broadcast and moves to $\mathsf{L}_{i+1}(\mathsf{n})$ but no other nodes are affected. Therefore, for every node $\mathsf{n} \in \mathsf{N}$, $\mathsf{L}'_{i+1}(\mathsf{n}) = \mathsf{L}_{i+1}(\mathsf{n})$ and set $\mathsf{E}'_{i+1} = \mathsf{E}''_i$.

We extend ρ''_i with the transition $\gamma''_i \xrightarrow{\mathsf{n},!a}_{\mathtt{r}} \gamma_{i+1}$ to obtain ρ'_{i+1}. This concludes the proof of the lemma. □

We have proved that every execution in loss-on-broadcast semantics has a corresponding reconfigurable execution. However, in contrast to Lemma 2.8, it is not immediate whether a reconfigurable execution can be simulated by one in the loss-on-broadcast semantics. In the next chapter, we will show a weaker result which states that if a configuration γ is reachable in reconfigurable semantics, then for any state q, label of a node in γ, there exists an execution reaching γ' in the loss-on-broadcast semantics such that at least one node in γ' is labelled with q.

In the next section, we introduce the coverability problem for broadcast protocols and discuss its decidability status for various semantics.

2.3 Coverability

One can be interested in various decision problems on the models for ad hoc networks, for instance *coverability* (whether there exists an execution in which a given state appears eventually), *repeated coverability* (whether there exists an execution in which a given state occurs infinitely often), *target reachability* (whether a configuration is reachable in which every node has a label from a given set of states). In this thesis, we are most interested in the coverability problem, which we formally define in the following.

Given a broadcast protocol $\mathcal{P}$ and a set of *target states* $F \subseteq Q$, the *coverability problem* (also called *control state reachability problem* in the literature) asks whether from an initial configuration a final configuration is reachable in which at least one node has label $q \in F$. Often in practice, the set F consists in some kind of error states and therefore such a *covering execution* is considered as an undesirable execution. In that situation, a positive instance of coverability problem thus corresponds to a network that can exhibit an undesirable behaviour. This emphasizes the importance and relevance of the coverability problem of broadcast protocols.

Before moving towards the formal definition of coverability, let us discuss a few examples.

Example 2.11. *Consider the protocol from Figure 2.1. Considering $F = \{r_2\}$, Figure 2.3 provides an example of an execution that covers F: the final configuration has a node labelled with r_2. The intuition is as follows: every configuration has 3 nodes. The top-most node always stays at q_0 and only broadcasts message a to the middle node when needed (for instance, in the first and third steps). The middle node follows the middle path in*

the protocol; it receives message a *from the node at the top and broadcasts messages* b_i *(*$i \in \{1, 2\}$*) to the bottom node. Finally, the node at the bottom reaches* r_2 *by receiving* b_i*'s from the middle node and following the bottom path in the protocol. Note here that, indeed, the communication topology has changed multiple times in that execution. Also notice that, under* reconfigurable *semantics, every state in this protocol is* coverable*: for every* $q \in Q$*, there exists an execution such that the final configuration contains a node labelled with state* q*.*

Example 2.12. *The execution presented in Figure 2.2 for the protocol of Figure 2.1. does not cover* r_2*. In fact, one can show that, under* static *semantics, the state* r_2 *cannot be covered by any execution,* i.e.*, for any configuration reachable from an initial configuration, none of the nodes has label* r_2*. Indeed, in order to cover* r_2*, one node –say* n*– must reach* q_4 *by performing the broadcast of* b_2*. Node* n *must be linked to at least one other node* n' *that performs the second broadcast of* a *and make* n *move from* q_2 *to* q_3*. To reach* q_2*, node* n *must have performed a broadcast of* b_1*. However, because of the static topology after* n *had broadcast* b_1*,* n' *could no longer be in* q_0 *(and moved to either* p_1 *or to* r_1*), and thus would not have been able to broadcast* a*. Thus, node* n *could not have reached* r_2*.*

Example 2.13. *Every state in the protocol from Figure 2.1 is coverable under both* loss-on-reception *and* loss-on-broadcast *semantics. For loss-on-reception semantics, the statement follows from Lemma 2.8 and the fact that it is already the case for reconfigurable semantics. As we will see later in this part of the thesis, the set of coverable states in loss-on-broadcast semantics also coincides with the ones in reconfigurable semantics. Examples of a loss-on-reception execution and a loss-on-broadcast execution are presented in Figure 2.4 and Figure 2.5, respectively, where the final configuration in each of them has a node labelled with state* r_2*.*

In the following, we formally define the coverability problem.

Given a broadcast protocol $\mathcal{P}$ and a subset of target states $F \subseteq Q$, $\mathtt{COVER}_{\mathtt{sem}}(\mathcal{P}, F)$ denotes the set of all *covering executions* under the semantics $\mathtt{sem} \in \{\mathtt{s}, \mathtt{r}, \mathtt{lr}, \mathtt{lb}\}$ (recall that $\mathtt{s}, \mathtt{r}, \mathtt{lr}, \mathtt{lb}$ represent static, reconfigurable, loss-on-reception and loss-on-broadcast semantics, respectively), that is, executions that reach a configuration with a node labelled by a state in F:

$$\mathtt{COVER}_{\mathtt{sem}}(\mathcal{P}, F) = \{(\gamma_i = (\mathsf{N}, \mathsf{E}_i, \mathsf{L}_i))_{0 \leq i \leq t} \in \mathtt{Exec}_{\mathtt{sem}}(\mathcal{P}) \mid \mathsf{L}_t(\gamma_t) \cap F \neq \emptyset\}.$$

Then the coverability problem is defined as follows:

Coverability problem
Input: A broadcast protocol $\mathcal{P}$, a set of states $F \subseteq Q$, $\mathtt{sem} \in \{\mathtt{s}, \mathtt{r}, \mathtt{lr}, \mathtt{lb}\}$.
Output: Yes if and only if $\mathtt{COVER}_{\mathtt{sem}}(\mathcal{P}, F) \neq \emptyset$.

In the following, we discuss the decidability status of the coverability problem for various semantics.

In [DSZ10], it was shown that the coverability problem is undecidable for broadcast protocols under static semantics. The proof goes by a reduction from the halting problem of a two-counter machine, which is known to be undecidable [Min67].

Theorem 2.14. *[DSZ10] The coverability problem is* undecidable *for static ad hoc networks.*

Decidability can be recovered in reconfigurable semantics. In fact, under reconfigurable semantics, the coverability problem can be solved in polynomial time. More precisely, coverability is PTIME-complete for reconfigurable networks [DSTZ12]. The polynomial time algorithm for coverability in reconfigurable semantics, called *saturation algorithm*, will be crucial to prove most of our results in the following chapter. We therefore describe the saturation algorithm from [DSTZ12] in the following.

2.3.1 Saturation algorithm

Fix a broadcast protocol $\mathcal{P} = (Q, I, \Sigma, \Delta)$. Delzanno *et al.* proposed a polynomial time saturation algorithm to compute the set of all states that can be covered under reconfigurable semantics for ad hoc networks [DSTZ12]. In the following chapter, we shall slightly modify this algorithm to achieve our results. For the sake of completeness, we present their algorithm here.

The algorithm maintains a set S of states that are known to be coverable. Initially, S is set to I. At each iteration, one adds to S all states that can be covered in one step from S. Formally, S is augmented with all $q' \in Q$ such that, either there exists $q \in S$ and $\mathsf{a} \in \Sigma$ with $(q, !\mathsf{a}, q') \in \Delta$, or there exist $p, q \in S$, $p' \in Q$ and $\mathsf{a} \in \Sigma$ such that $(p, !\mathsf{a}, p') \in \Delta$ and $(q, ?\mathsf{a}, q') \in \Delta$.

Algorithm 1 Saturation algorithm for coverability

```
1: S := I; S' := ∅
2: while S ≠ S' do
3:     S' := S
4:     for all (q1, !a, q2) ∈ Δ s.t. q1 ∈ S' do
5:         S := S ∪ {q2} ∪ {q2' ∈ Q | (q1', ?a, q2') ∈ Δ ∧ q1' ∈ S'}
6:     end for
7: end while
8: return S
```

We will write $\texttt{REACH}_{\texttt{sem}}(\mathcal{P})$ for the set of states that can be covered in semantics $\texttt{sem} \in \{\texttt{s}, \texttt{r}, \texttt{lr}, \texttt{lb}\}$:

$$\texttt{REACH}_{\texttt{sem}}(\mathcal{P}) = \{q \in Q \mid \exists \gamma_0 \xrightarrow{*}_{\texttt{sem}} \gamma \in \texttt{Exec}_{\texttt{sem}}(\mathcal{P}) \text{ with } \gamma = (\mathsf{N}, \mathsf{E}, \mathsf{L}) \text{ s.t. } q \in \mathsf{L}(\gamma)\}.$$

We then state the following result from [DSTZ12] that establishes the correspondence between the set of states returned by the saturation algorithm and the set of coverable states in reconfigurable semantics of broadcast protocols.

Lemma 2.15 ([DSTZ12]). *Algorithm 1 terminates and the set S the algorithm returns is exactly the set of coverable states in reconfigurable semantics. Formally, $S = \mathtt{REACH}_{\mathtt{r}}(\mathcal{P})$. Moreover, there exists an execution $\rho = \gamma_0 \to_{\mathtt{r}} \gamma_1 \cdots \to_{\mathtt{r}} \gamma$ with $\gamma = (\mathsf{N}, \mathsf{E}, \mathsf{L})$, such that $\mathsf{L}(\gamma) = S$.*

By Lemma 2.15, deciding the coverability problem for reconfigurable networks thus reduces to checking if $F \cap S \neq \emptyset$ (*i.e.*, $\mathtt{COVER}_{\mathtt{r}}(\mathcal{P}, F) \neq \emptyset$ iff $F \cap S \neq \emptyset$). Notice that Algorithm 1 performs at most $|Q|$ iterations and each iteration requires at most $|\Delta|^2$ checks (check for a message broadcast or reception), and each check can be done in constant time. Therefore, the set S can be computed in *polynomial time* ($O(|Q|.|\Delta|^2)$) in the size of the input protocol. Finally, checking non-emptiness of the set $F \cap S$ can also be done in polynomial time. We conclude that the coverability problem can be decided in polynomial time for reconfigurable ad hoc networks.

Lemma 2.16 ([DSTZ12]). *The coverability problem is in* PTIME *for reconfigurable ad hoc networks.*

Delzanno *et al.* have further shown that coverability in reconfigurable semantics is PTIME-hard by a LOGSPACE reduction from the Circuit Value Problem, which is known to be PTIME-complete [Lip76].

Theorem 2.17 ([DSTZ12]). *The coverability problem is* PTIME*-complete for reconfigurable ad hoc networks.*

As a corollary of Lemma 2.8, we conclude that Lemma 2.15 also holds for ad hoc networks in loss-on-reception semantics, that is the set Algorithm 1 returns is also the set of coverable states in loss-on-reception semantics (*i.e.*, $S = \mathtt{REACH}_{\mathtt{lr}}(\mathcal{P})$). Indeed, there is a covering execution in the reconfigurable semantics if and only if there is one in the loss-on-reception semantics. Hence the coverability problem for loss-on-reception networks can also be solved in polynomial time. Similarly, the PTIME-hardness result for coverability also applies to the loss-on-reception semantics.

Corollary 2.18. *The coverability problem is* PTIME*-complete for broadcast protocols in loss-on-reception semantics.*

In the next section, we define two relevant quantities of broadcast protocols, namely the cutoff and the covering length, and the problems we are most interested in, that is finding bounds for these quantities.

2.4 Cutoff and covering length

While considering a parameterized model checking problem for a distributed system, a typical approach is to check if a cutoff exists and then to find the cutoff where it exists.

Intuitively, the cutoff is a bound on the number of processes such that if a given property is satisfied for this specific number of processes then it also holds for any larger number of processes. If a cutoff exists, then a parameterized model checking problem boils down to a finite model checking problem, that is, checking if the property holds with finitely many processes, the number being less than or equal to the cutoff.

This approach is particularly relevant for the coverability problem in reconfigurable, loss-on-reception and loss-on-broadcast ad hoc networks, since they follow a monotonicity property that we will show in the following chapter: if a state can be covered from a configuration, it can also be covered from any configuration with more nodes. We are therefore interested in finding bounds on the cutoff for the positive instances of the coverability problem in those semantics. More precisely, the *cutoff* is the minimal number of processes in the network for which a covering execution exists. While considering error states, the cutoff therefore measures the maximum size of a network that can avoid those faulty states.

Definition 2.19. *Given a broadcast protocol $\mathcal{P}$, a target set F, and* $\mathtt{sem} \in \{\mathtt{s}, \mathtt{r}, \mathtt{lr}, \mathtt{lb}\}$*, the* cutoff *under the semantics* `sem` *is the minimal number of processes in the network for which a covering execution exists that reaches a state in F:*

$$\mathtt{CUTOFF}_{\mathtt{sem}}(\mathcal{P}, F) = \min_{\rho \in \mathtt{COVER}_{\mathtt{sem}}(\mathcal{P},F)} \#\mathit{nodes}(\rho).$$

Additionally, we introduce here a similar notion, called *covering length*, that is the minimal number of steps for a covering execution. When considering faulty states, the covering length weighs how fast a network execution can go wrong.

Definition 2.20. *Given a broadcast protocol $\mathcal{P}$, a target set F, and* $\mathtt{sem} \in \{\mathtt{s}, \mathtt{r}, \mathtt{lr}, \mathtt{lb}\}$*, the* covering length *under the semantics* `sem` *is the minimal number of steps in an execution that reaches a state in F:*

$$\mathtt{COVLEN}_{\mathtt{sem}}(\mathcal{P}, F) = \min_{\rho \in \mathtt{COVER}_{\mathtt{sem}}(\mathcal{P},F)} \#\mathit{steps}(\rho).$$

In both the cases, we will use the convention that if the set $\mathtt{COVER}_{\mathtt{sem}}(\mathcal{P}, F) = \emptyset$, then both the cutoff and the covering length are ∞.

Example 2.21. *Figure 2.3 provides an example execution of the protocol in Figure 2.1 under reconfigurable semantics with 3 nodes that covers the state r_2. One can in fact show that there is no reconfigurable execution that reaches r_2 with 2 nodes. Indeed, a node n_2 has to reach r_2 by receiving messages b_1 and b_2 sequentially from another node n_1. However, n_1 has to receive an a between those broadcasts, which is not possible by a broadcast from n_2 since n_2 would be in r_1 after receiving b_1. Therefore, we must need another node than n_1 and n_2. Thus, the cutoff for $F = \{r_2\}$ in the reconfigurable semantics is 3.*

Given a broadcast protocol $\mathcal{P}$ and a target set of states F, our aim is to find upper and lower bounds for $\texttt{CUTOFF}_{\texttt{sem}}(\mathcal{P}, F)$ and $\texttt{COVLEN}_{\texttt{sem}}(\mathcal{P}, F)$.

However, as a consequence of Lemma 2.8, the bounds on the cutoff and the covering length for coverability in the reconfigurable semantics will also apply to the loss-on-reception semantics. We will therefore mainly focus on the reconfigurable and loss-on-broadcast semantics.

We further investigate the complexity of the decision problem of determining the minimum size of covering executions. More precisely, given a broadcast protocol, a set of target states, a natural number k, and $\texttt{sem} \in \{\texttt{s}, \texttt{r}, \texttt{lr}, \texttt{lb}\}$, decide if the cutoff for the coverability problem under the semantics $\texttt{sem}$ is at most k. This can be formalized as follows:

Minimum number of nodes for coverability (MinNodes)
Input: A broadcast protocol $\mathcal{P}$, a set of target states $F \subseteq Q$, a natural number $k \in \mathbb{N}$, and $\texttt{sem} \in \{\texttt{s}, \texttt{r}, \texttt{lr}, \texttt{lb}\}$.
Output: Yes if and only if $\texttt{CUTOFF}_{\texttt{sem}}(\mathcal{P}, F) \leq k$.

We will close the chapter by announcing our contributions on the cutoff and the covering length for coverability that we will present in the following chapter.

2.5 Discussion

The saturation algorithm presented in Section 2.3.1 computes the set of coverable states in reconfigurable semantics in polynomial time. Let $S_0, S_1, \ldots, S_m$ be the sets after each iteration of the algorithm, with $S_0 = I$ and $S_m = S$. In the proof of Lemma 2.15, the authors inductively construct a witness covering execution for the set S. More precisely, for every $0 \leq j \leq m$, they construct an execution $\rho_j = \gamma_0 \to_{\texttt{r}} \gamma_1 \cdots \to_{\texttt{r}} \gamma_{f(j)}$ such that γ_0 is an initial configuration, and $\gamma_{f(j)} = (\mathsf{N}, \mathsf{E}, \mathsf{L})$ with $\mathsf{L}(\gamma_{f(j)}) = S_j$. However, in their proof, in the inductive step the number of nodes is at least doubled, *i.e.*, $\#\mathsf{nodes}(\rho_{j+1}) \geq 2\#\mathsf{nodes}(\rho_j)$. This produces a covering execution for S of size exponential in the size of the input protocol. Consequently, the length of the execution is also considerably large. As another consequence, the same bounds also apply to the loss-on-reception semantics, thanks to Lemma 2.8. To the best of our knowledge, any bounds on the minimum number of nodes (*i.e.*, cutoff) and the minimum length (*i.e.*, covering length) of an execution that is necessary to reach a state in S have not been given before.

In the following chapter, we investigate the problem of finding upper and lower bounds on cutoff and covering length, formally defined in Section 2.4. Among all the semantics discussed in Section 2.2, notice that coverability is undecidable for static semantics, and bounds for loss-on-reception semantics will coincide with the ones for reconfigurable

semantics, as a consequence of Lemma 2.8. We therefore focus on reconfigurable and loss-on-broadcast semantics.

In Chapter 3, we show a linear upper bound for cutoff and a quadratic upper bound for the covering length for these two semantics. We first observe that in both semantics, the networks satisfy a monotonicity property, called *copycat property*, which informally states that any node in a reachable configuration can be duplicated. This allows us to refine the saturation algorithm presented in Section 2.3 in a simple manner to achieve the desired upper bounds. We further show that the upper bounds on the cutoff and the covering length are in fact tight in both the semantics: we exhibit a family of protocols that meet these bounds, although, the reconfigurable executions can in general be more succinct than the loss-on-broadcast executions. We also show that the set Algorithm 1 returns coincide with the set of all coverable states in loss-on-broadcast semantics. In other words, a state is coverable in the reconfigurable semantics if and only if it is coverable in the loss-on-broadcast semantics. Finally, we show that MinNodes is an NP-complete problem.

Chapter 3

Tight Bounds on Cutoff and Covering Length

In the previous chapter, we have seen that the coverability problem is decidable in polynomial time for broadcast protocols under reconfigurable (hence, also under loss-on-reception) semantics. A saturation algorithm was presented in Section 2.3 which computes all states that can be covered in these semantics.

In this chapter, we will prove tight bounds on cutoff and covering length for various semantics of broadcast protocols. Notice that as a consequence of Lemma 2.8, the bounds for the reconfigurable semantics will also apply to the loss-on-reception semantics, hence, we will mainly focus on reconfigurable and loss-on-broadcast semantics. In view of proving the upper bounds, we first refine the saturation algorithm described in Section 2.3 in a simple manner, so as to keep track of the size of an execution that covers the set returned by the algorithm. We then show that, based on the underlying computation, one can construct small witness executions for the positive instances of the coverability problem in reconfigurable semantics. These small witnesses have a linear number of nodes and a quadratic number of steps. We further show that the same upper bounds also hold for loss-on-broadcast semantics. We then show that these bounds are tight in both semantics: we give a family of protocols for which any covering execution needs at least a linear number of nodes and a quadratic number of steps. Despite satisfying the same bounds, we show that reconfiguration can be in some cases more succinct by a linear factor than message losses on broadcast. Finally, we prove that the problem of deciding the size of a minimal witness for the coverability problem is NP-complete.

While considering message losses, since our main focus is on the loss-on-broadcast semantics, further in this chapter, by "lossy semantics", we will refer to the semantics of message losses on broadcast, unless otherwise specified.

Organization of the chapter

Section 3.1 is dedicated to the refined saturation algorithm. First, we observe an important property, called *copycat property*, for both reconfigurable and lossy networks, and then present the modified algorithm. Complexity upper bounds on cutoff and covering length together with matching lower bounds are presented in Section 3.2. We then show that reconfigurable executions can be in some cases more succinct than the lossy ones in Section 3.3. The MinNodes problem is studied in Section 3.4. We close the chapter with a remark on application of our results in other communication models and a discussion on the possible directions for future work in Section 3.5.

3.1 Refined saturation algorithm

In this section, we modify the saturation algorithm recalled in Chapter 2, Section 2.3 so that it also keeps track of the size of an execution that covers the set returned by the algorithm. This new measure will play a crucial role in proving the complexity bounds for cutoff and covering length. Our results on the upper bounds of cutoff and covering length, however, rely on a monotonicity property, called *copycat property*, for reconfigurable and lossy ad hoc networks. We therefore begin the section describing this property in more details.

Intuitively, the copycat property states that any node in a reachable configuration in these semantics can be duplicated. More precisely, for every reachable configuration γ, and any node n in γ, one can build another execution such that the final configuration γ' has an extra copy of node n with the same label, keeping the other nodes the same as in γ. This monotonicity property has already been observed in [DSTZ12], and it also appears in several other contexts, for instance for asynchronous shared-memory systems [EGM13]. We now formally state and prove this property, starting with the reconfigurable semantics.

3.1.1 Copycat property for reconfigurable semantics

The copycat property for reconfigurable ad hoc networks can be formalized as follows.

Proposition 3.1. *Let $\rho = \gamma_0 \to_{\mathtt{r}} \gamma_1 \cdots \to_{\mathtt{r}} \gamma_s$ be an execution in $\mathtt{Exec}_{\mathtt{r}}(\mathcal{P})$, with $\gamma_s = (\mathsf{N}, \mathsf{E}_s, \mathsf{L}_s)$. Then for any $q \in \mathsf{L}_s(\gamma_s)$, for any $\mathsf{n}^q \in \mathsf{N}$ such that $\mathsf{L}_s(\mathsf{n}^q) = q$, there exists $t \in \mathbb{N}$ and an execution $\rho' = \gamma'_0 \to_{\mathtt{r}} \gamma'_1 \cdots \to_{\mathtt{r}} \gamma'_t$ with $\gamma'_t = (\mathsf{N}', \mathsf{E}'_t, \mathsf{L}'_t)$ such that $|\mathsf{N}'| = |\mathsf{N}|+1$, there is an injection $\iota : \mathsf{N} \to \mathsf{N}'$ with for every $\mathsf{n} \in \mathsf{N}$, $\mathsf{L}'_t(\iota(\mathsf{n})) = \mathsf{L}_s(\mathsf{n})$, and for the extra node $\mathsf{n}_{\mathsf{fresh}} \in \mathsf{N}' \setminus \iota(\mathsf{N})$, $\mathsf{L}'_t(\mathsf{n}_{\mathsf{fresh}}) = q$; moreover, $\#\mathit{steps}(\rho', \mathsf{n}_{\mathsf{fresh}}) = \#\mathit{steps}(\rho, \mathsf{n}^q)$.*

Let us first give an intuitive idea of the proof. The new node $\mathsf{n}_{\mathsf{fresh}}$ will copy the moves of node n^q: it performs the same broadcasts (yet to an empty set of neighbours) and receives the same messages, thanks to reconfiguration of the topology. More precisely,

yet informally, when n^q broadcasts in ρ, it does so also in ρ'; in the following step we disconnect every node and $\mathsf{n}_{\mathsf{fresh}}$ repeats the same broadcast (no other node is affected because of the disconnection). On the other hand, when n^q receives a message from another node n in ρ, in the corresponding step in ρ', we connect $\mathsf{n}_{\mathsf{fresh}}$ to the node $\iota(\mathsf{n})$, *i.e.*, $(\iota(\mathsf{n}), \mathsf{n}_{\mathsf{fresh}}) \in \mathsf{E}'$ so that $\mathsf{n}_{\mathsf{fresh}}$, along with the others, receives the same message in ρ'. We now prove the proposition formally.

Proof. Fix an execution $\rho = \gamma_0 \to_{\mathtt{r}} \gamma_1 \cdots \to_{\mathtt{r}} \gamma_s$ with $\gamma_i = (\mathsf{N}, \mathsf{E}_i, \mathsf{L}_i)$ for every $0 \leq i \leq s$. We denote by ρ_i the execution $\gamma_0 \to_{\mathtt{r}} \cdots \gamma_i$ whenever $0 \leq i \leq s$. Let $\mathsf{n}^q \in \mathsf{N}$ be such that $\mathsf{L}_s(\mathsf{n}^q) = q$. Define N' a finite set such that $|\mathsf{N}'| = |\mathsf{N}| + 1$, and fix an injection $\iota : \mathsf{N} \to \mathsf{N}'$. Write $\mathsf{n}_{\mathsf{fresh}}$ for the unique element of $\mathsf{N}' \setminus \iota(\mathsf{N})$. We first define $\gamma'_0 = (\mathsf{N}', \mathsf{E}'_0, \mathsf{L}'_0)$. Set $\mathsf{L}'_0(\iota(\mathsf{n})) = \mathsf{L}_0(\mathsf{n})$ for every $\mathsf{n} \in \mathsf{N}$, and $\mathsf{L}'_0(\mathsf{n}_{\mathsf{fresh}}) = \mathsf{L}_0(\mathsf{n}^q)$. We will show by induction on i that for every $0 \leq i \leq s$, there is an execution $\rho'_i = \gamma'_0 \to_{\mathtt{r}} \gamma'_1 \cdots \to_{\mathtt{r}} \gamma'_{f(i)}$ for some $f(i)$, with $\gamma'_{f(i)} = (\mathsf{N}', \mathsf{E}'_{f(i)}, \mathsf{L}'_{f(i)})$, such that $\mathsf{L}'_{f(i)}(\iota(\mathsf{n})) = \mathsf{L}_i(\mathsf{n})$ for every $\mathsf{n} \in \mathsf{N}$ and $\mathsf{L}'_{f(i)}(\mathsf{n}_{\mathsf{fresh}}) = \mathsf{L}_i(\mathsf{n}^q)$; and finally $\#\mathsf{steps}(\rho'_i, \mathsf{n}_{\mathsf{fresh}}) = \#\mathsf{steps}(\rho_i, \mathsf{n}^q)$. The base case $i = 0$ is trivial. We assume that for some $i < s$, we have constructed a corresponding $\rho'_i = \gamma'_0 \to_{\mathtt{r}} \cdots \gamma'_{f(i)}$. Then we will extend it to ρ'_{i+1} as follows. We will apply a similar idea used in the second part of the proof of Lemma 2.8 for reconfigurations of the communication topology. We make a case distinction depending on the nature of the step $\gamma_i \to_{\mathtt{r}} \gamma_{i+1}$:

- Assume $\gamma_i \xrightarrow{\mathsf{n},!a}_{\mathtt{r}} \gamma_{i+1}$ is a broadcast message such that $\mathsf{n}^q \neq \mathsf{n}$. For checking correctness, we distinguish two cases:
  - n^q is not connected to n in γ_i. Then it is the case that the label of n^q remains unchanged during the step $\gamma_i \xrightarrow{\mathsf{n},!a}_{\mathtt{r}} \gamma_{i+1}$. Consider $\rho''_i = \gamma'_0 \to_{\mathtt{r}} \cdots \gamma'_{f(i)-1} \to_{\mathtt{r}} \gamma''_{f(i)}$ such that the nodes and their labels in $\gamma''_{f(i)}$ are same as in $\gamma'_{f(i)}$; furthermore, thanks to reconfiguration, the communication topology among the nodes in $\iota(\mathsf{N})$ of $\gamma''_{f(i)}$ is the same as in γ_i, additionally, $\mathsf{n}_{\mathsf{fresh}}$ is disconnected from every other node. We define ρ'_{i+1} the execution obtained by extending ρ''_i with the broadcast $\gamma''_{f(i)} \xrightarrow{\iota(\mathsf{n}),!a}_{\mathtt{r}} \gamma'_{f(i)+1}$. We write $f(i+1) = f(i)+1$. Notice that neither of $\iota(\mathsf{n}^q)$ and $\mathsf{n}_{\mathsf{fresh}}$ are affected in this step. Since they both had label $\mathsf{L}_i(\mathsf{n}^q)$ in $\gamma'_{f(i)}$ (by induction hypothesis), they will also have the same label in $\gamma''_{f(i)}$ and $\gamma'_{f(i+1)}$, respectively. Since n^q was not affected in the last step of ρ_{i+1}, $\mathsf{L}'_{f(i+1)}(\iota(\mathsf{n}^q)) = \mathsf{L}_{i+1}(\mathsf{n}^q)$. The condition on the number of broadcasts by $\mathsf{n}_{\mathsf{fresh}}$ is therefore also trivially satisfied. Note also that all other nodes in $\iota(\mathsf{N})$ can progress to the same states as those of N in γ_{i+1}.
  - n^q is connected to n in γ_i. Then in this step n^q receives message a from n. Consider again $\rho''_i = \gamma'_0 \to_{\mathtt{r}} \cdots \gamma'_{f(i)-1} \to_{\mathtt{r}} \gamma''_{f(i)}$ such that the nodes and their labels in $\gamma''_{f(i)}$ are same as in $\gamma'_{f(i)}$; furthermore, thanks to reconfiguration, the communication topology among the nodes in $\iota(\mathsf{N})$ of $\gamma''_{f(i)}$ is the same as in γ_i, additionally, $\mathsf{n}_{\mathsf{fresh}}$ is connected to $\iota(\mathsf{n})$. We define ρ'_{i+1} the execution obtained by extending ρ''_i with the broadcast $\gamma''_{f(i)} \xrightarrow{\iota(\mathsf{n}),!a}_{\mathtt{r}} \gamma'_{f(i)+1}$. We write $f(i+1) = f(i)+1$. In this case, both nodes $\iota(\mathsf{n}^q)$ and $\mathsf{n}_{\mathsf{fresh}}$ receive the message

a from $\iota(\mathsf{n})$. Since they both had label $\mathsf{L}_i(\mathsf{n}^q)$ in $\gamma'_{f(i)}$ (by induction hypothesis), they will also have the same label in $\gamma''_{f(i)}$, and hence, resolving properly the non-determinism of the protocol, both of them reach the label $\mathsf{L}_{i+1}(\mathsf{n}^q)$ in $\gamma'_{f(i+1)}$. The condition on the number of broadcasts by $\mathsf{n}_{\mathsf{fresh}}$ is also trivially satisfied. Note also that all other nodes in $\iota(\mathsf{N})$ can progress to the same states as those of N in γ_{i+1}.

- Assume $\gamma_i \xrightarrow{\mathsf{n}^q,!\mathsf{a}}_{\mathtt{r}} \gamma_{i+1}$ is a broadcast message by node n^q. Consider the reconfigurable execution of length $f(i)$: $\rho''_i = \gamma'_0 \to_{\mathtt{r}} \cdots \gamma'_{f(i)-1} \to_{\mathtt{r}} \gamma''_{f(i)}$ such that the nodes and their labels in $\gamma''_{f(i)}$ are same as in $\gamma'_{f(i)}$; furthermore, thanks to reconfiguration, the communication topology among the nodes in $\iota(\mathsf{N})$ of $\gamma''_{f(i)}$ is the same as in γ_i, additionally, $\mathsf{n}_{\mathsf{fresh}}$ is disconnected from every other node. We then extend ρ''_i in two steps. First, $\gamma''_{f(i)} \xrightarrow{\iota(\mathsf{n}^q),!\mathsf{a}}_{\mathtt{r}} \gamma'_{f(i)+1}$ such that nodes in $\gamma'_{f(i)+1}$ are disconnected from each other (*i.e.* $\mathsf{E}'_{f(i)+1} = \emptyset$); and second, $\gamma'_{f(i)+1} \xrightarrow{\mathsf{n}_{\mathsf{fresh}},!\mathsf{a}}_{\mathtt{r}} \gamma'_{f(i)+2}$. The resulting execution is denoted ρ'_{i+1}, and we write $f(i+1) = f(i) + 2$.

  Notice that in the first step, $\mathsf{n}_{\mathsf{fresh}}$ is not affected since it is not connected with any other node in $\gamma''_{f(i)}$, and all nodes in $\iota(\mathsf{N})$ progress to the same states as those of N in γ_{i+1}. In the second step, $\mathsf{n}_{\mathsf{fresh}}$ broadcasts message a and, resolving properly the non-determinism of the protocol, reaches the same label as of n^q in γ_{i+1}; and every other node is unaffected in the second step, therefore they remain in the same states as those of N in γ_{i+1}. Each of n^q and $\mathsf{n}_{\mathsf{fresh}}$ performs a broadcast and hence $\#\mathsf{steps}(\rho'_{i+1}, \mathsf{n}_{\mathsf{fresh}}) = \#\mathsf{steps}(\rho_{i+1}, \mathsf{n}^q)$.

This concludes the proof of Proposition 3.1. □

Let us illustrate the copycat property on an example of a reconfigurable network. Consider the reconfigurable execution from Figure 2.3. Below is an example of an execution where the middle node from that example is duplicated (and represented bottom-most).

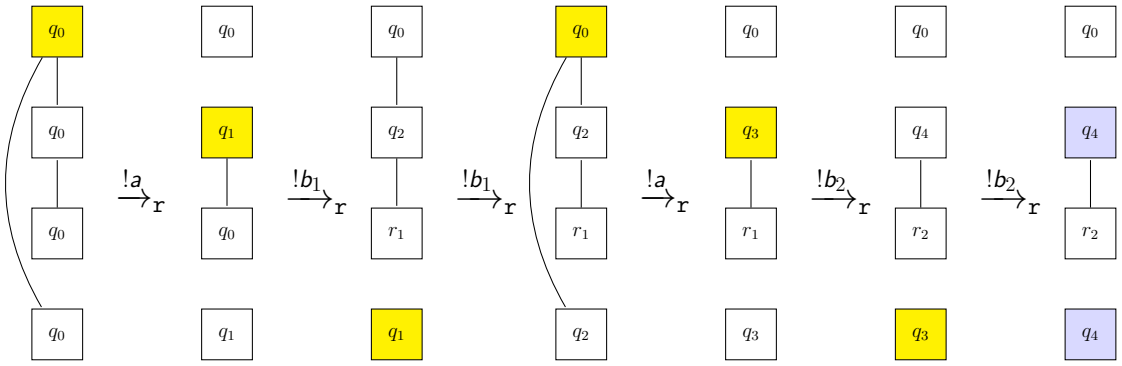


Figure 3.1: Illustration of the copycat property for reconfigurable semantics.

Example 3.2. *Figure 3.1 illustrates the copycat property for the reconfigurable semantics. Here, in this example, the nodes follow the protocol in Figure 2.1. In this example, the bottom-most node* ($\mathsf{n}_{\mathsf{fresh}}$) copies *the middle node from the execution in Figure 2.3* (n^q).

Note here that a broadcast by n^q *(for instance, broadcast of* b_1 *and* b_2*) is duplicated in two steps and a reception by* n^q *(for instance the reception of* a*) is duplicated by connecting* $\mathsf{n}_{\mathsf{fresh}}$ *with the broadcasting node. Finally, observe that in the final configuration, both nodes* n^q *and* $\mathsf{n}_{\mathsf{fresh}}$ *are at* q_4 *that are highlighted in blue.*

We now prove that loss-on-broadcast ad hoc networks also satisfy this property.

3.1.2 Copycat property for loss-on-broadcast semantics

The copycat property for loss-on-broadcast ad hoc networks can be formalized as follows.

Proposition 3.3. *Given* $\rho = \gamma_0 \to_{\mathtt{lb}} \gamma_1 \cdots \to_{\mathtt{lb}} \gamma_s$ *an execution, with* $\gamma_s = (\mathsf{N}, \mathsf{E}, \mathsf{L}_s)$*, for every* $q \in \mathsf{L}_s(\gamma_s)$*, for every* $\mathsf{n}^q \in \mathsf{N}$ *such that* $\mathsf{L}_s(\mathsf{n}^q) = q$*, there exists* $t \in \mathbb{N}$ *and an execution* $\rho' = \gamma'_0 \to_{\mathtt{lb}} \gamma'_1 \cdots \to_{\mathtt{lb}} \gamma'_t$ *with* $\gamma'_t = (\mathsf{N}', \mathsf{E}', \mathsf{L}'_t)$ *such that* $|\mathsf{N}'| = |\mathsf{N}|+1$*, there is an injection* $\iota : \mathsf{N} \to \mathsf{N}'$ *with for every* $\mathsf{n} \in \mathsf{N}$*,* $\mathsf{L}'_t(\iota(\mathsf{n})) = \mathsf{L}_s(\mathsf{n})$*, and for the extra node* $\mathsf{n}_{\mathsf{fresh}} \in \mathsf{N}' \setminus \iota(\mathsf{N})$*,* $\mathsf{L}'_t(\mathsf{n}_{\mathsf{fresh}}) = q$*, for every* $\mathsf{n} \in \mathsf{N}$*,* $\mathsf{n}_{\mathsf{fresh}} \sim' \iota(\mathsf{n})$ *iff* $\mathsf{n}^q \sim \mathsf{n}$*,* $\#\mathit{steps}(\rho', \mathsf{n}_{\mathsf{fresh}}) = \#\mathit{steps}(\rho, \mathsf{n}^q)$*, and* $\#\mathit{nonlost\_steps}(\rho', \mathsf{n}_{\mathsf{fresh}}) = 0$*.*

The idea behind the proof is to make $\mathsf{n}_{\mathsf{fresh}}$ mimic n^q. Notice that in the lossy semantics, the communication topology is fixed throughout an execution. We connect $\mathsf{n}_{\mathsf{fresh}}$ to nodes to which n^q is connected. Then if n^q is receiving a message to progress, $\mathsf{n}_{\mathsf{fresh}}$ also receives the message; if n^q is broadcasting a message, then we will make $\mathsf{n}_{\mathsf{fresh}}$ lossy broadcast the same message so that no other node is impacted.

Proof. Fix an execution $\rho = \gamma_0 \to_{\mathtt{lb}} \gamma_1 \cdots \to_{\mathtt{lb}} \gamma_s$. First notice that, from our definition of lossy semantics, the topology should be the same throughout the execution. Therefore, we write $\gamma_0 = (\mathsf{N}, \mathsf{E}, \mathsf{L}_0)$, and more generally, for every $0 \leq i \leq s$, $\gamma_i = (\mathsf{N}, \mathsf{E}, \mathsf{L}_i)$. Furthermore, we denote by ρ_i the execution $\gamma_0 \to_{\mathtt{lb}} \cdots \gamma_i$ whenever $0 \leq i \leq s$. Define N' as a finite set such that $|\mathsf{N}'| = |\mathsf{N}| + 1$, and fix an injection $\iota : \mathsf{N} \to \mathsf{N}'$. Write $\mathsf{n}_{\mathsf{fresh}}$ for the unique element of $\mathsf{N}' \setminus \iota(\mathsf{N})$. We first define $\gamma'_0 = (\mathsf{N}', \mathsf{E}', \mathsf{L}'_0)$. Set $\mathsf{L}'_0(\iota(\mathsf{n})) = \mathsf{L}_0(\mathsf{n})$ for every $\mathsf{n} \in \mathsf{N}$, and $\mathsf{L}'_0(\mathsf{n}_{\mathsf{fresh}}) = \mathsf{L}_0(\mathsf{n}^q)$. Define the edge relation E' by its induced edge relation $\sim'$ such that $\iota(\mathsf{n}) \sim' \iota(\mathsf{n}')$ iff $\mathsf{n} \sim \mathsf{n}'$, and $\mathsf{n}_{\mathsf{fresh}} \sim' \iota(\mathsf{n}')$ iff $\mathsf{n}^q \sim \mathsf{n}'$.

We will show by induction on i that for every $0 \leq i \leq s$, there is an execution $\rho'_i : \gamma'_0 \to_{\mathtt{lb}} \gamma'_1 \cdots \to_{\mathtt{lb}} \gamma'_{f(i)}$ for some $f(i)$, with $\gamma'_{f(i)} = (\mathsf{N}', \mathsf{E}', \mathsf{L}'_{f(i)})$, such that $\mathsf{L}'_{f(i)}(\iota(\mathsf{n})) = \mathsf{L}_i(\mathsf{n})$ for every $\mathsf{n} \in \mathsf{N}$ and $\mathsf{L}'_{f(i)}(\mathsf{n}_{\mathsf{fresh}}) = \mathsf{L}_i(\mathsf{n}^q)$. The base case $i = 0$ is obvious. We assume that for some $i < s$ we have constructed a corresponding ρ'_i , and we will extend it to ρ'_{i+1} as follows. We make a case distinction depending on the nature of the step $\gamma_i \to_{\mathtt{lb}} \gamma_{i+1}$:

- Assume $\gamma_i \xrightarrow{\mathsf{n},!\mathsf{a}}_{\mathtt{lb}} \gamma_{i+1}$ is a broadcast message with $\mathsf{n}^q \neq \mathsf{n}$, then ρ'_{i+1} is obtained by extending ρ'_i with the broadcast $\gamma'_{f(i)} \xrightarrow{\iota(\mathsf{n}),!\mathsf{a}}_{\mathtt{lb}} \gamma'_{f(i)+1}$, with the condition that it should be lost if and only if it was lost in the original execution. For checking correctness, we distinguish two cases:

– the broadcast message was not lost, and $\mathsf{n}^q \sim \mathsf{n}$. Then, it is the case that $\mathsf{n}_{\mathsf{fresh}} \sim' \iota(\mathsf{n})$, hence $\mathsf{n}_{\mathsf{fresh}}$ also receives the message. By resolving properly the nondeterminism, we can make the label of $\mathsf{n}_{\mathsf{fresh}}$ become the same as the label of n^q in γ_{i+1}. Note also that all nodes in $\iota(\mathsf{N})$ can progress to the same states as those of N in γ_{i+1};

– the broadcast message was lost, or $\mathsf{n}^q \not\sim \mathsf{n}$, then it is the case that the label of n^q has not been changed in $\gamma_i \xrightarrow{\mathsf{n},!a}_{\mathtt{lb}} \gamma_{i+1}$, and so will the label of the fresh node in $\gamma'_{f(i)}$.

- Assume $\gamma_i \xrightarrow{\mathsf{n}^q,!a}_{\mathtt{lb}} \gamma_{i+1}$ is a broadcast message, then we extend ρ'_i with the two steps $\gamma'_{f(i)} \xrightarrow{\iota(\mathsf{n}^q),!a}_{\mathtt{lb}} \gamma'_{f(i)+1} \xrightarrow{\mathsf{n}_{\mathsf{fresh}},!a}_{\mathtt{lb}} \gamma'_{f(i)+2}$ (resolving nondeterminism in a similar way as in $\gamma_i \xrightarrow{\mathsf{n}^q,!a}_{\mathtt{lb}} \gamma_{i+1}$), and we make the last broadcast lossy whereas the broadcast from $\iota(\mathsf{n}^q)$ is lossy if and only if it was lossy in $\gamma_i \rightarrow_{\mathtt{lb}} \gamma_{i+1}$.

This concludes the induction. Notice that in the constructed execution, node $\mathsf{n}_{\mathsf{fresh}}$ does not make any real sending, so that $\#\mathsf{nonlost\_steps}(\rho'_{i+1}, \mathsf{n}_{\mathsf{fresh}}) = 0$. □

Let us illustrate the copycat property in a lossy network. Consider the lossy execution from Figure 2.5. Below is an example of an execution where the node in middle-right (that reaches the state q_4) from that example is duplicated.

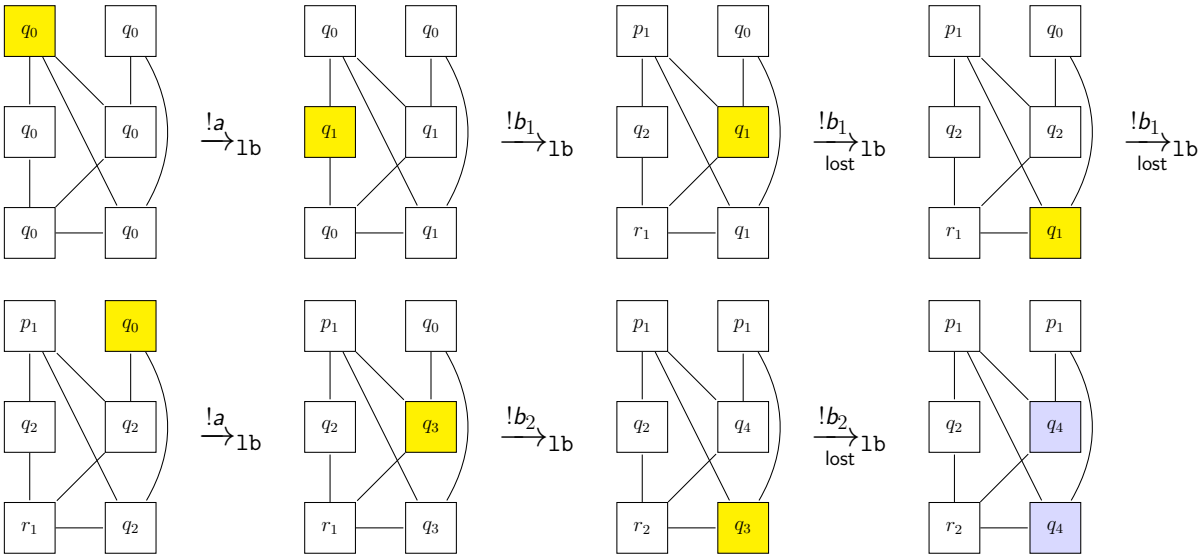


Figure 3.2: Illustration of the copycat property for loss-on-broadcast semantics.

Example 3.4. *Figure 3.2 illustrates the copycat property for the loss-on-broadcast semantics. Here, in this example, the nodes follow the protocol in Figure 2.1. In this example, the node at bottom-right, call* $\mathsf{n}_{\mathsf{fresh}}$, copies *the node at middle-right (that reaches the state* q_4*), call* n^q*, from the execution in Figure 2.3. Note here that* $\mathsf{n}_{\mathsf{fresh}}$ *is connected to only those who were connected to* n^q *in the original execution. A real broadcast (for instance, broadcast of* b_2 *in the last step) by* n^q *is duplicated in two steps: first the original node*

performs the (real) broadcast followed by a lossy broadcast of the same message by $\mathsf{n}_{\mathsf{fresh}}$*; a lossy broadcast by* n^q *is also duplicated in two steps: first the original node performs the (lossy) broadcast followed by another lossy broadcast of the same message by* $\mathsf{n}_{\mathsf{fresh}}$*; and a message that is received by* n^q *is also received by* $\mathsf{n}_{\mathsf{fresh}}$ *because of the communication topology (for instance the reception of* a*). Finally, observe that in the final configuration, both nodes* n^q *and* $\mathsf{n}_{\mathsf{fresh}}$ *are at* q_4*, that are highlighted in blue.*

We have shown that both reconfigurable and lossy networks satisfy a strong copycat property. We now describe the modified saturation algorithm.

3.1.3 Refined Saturation Algorithm

Fix a broadcast protocol $\mathcal{P} = (Q, I, \Sigma, \Delta)$. Once again, we maintain a set, say S, of states that are known to be coverable. Initially, S is set to I. However, in contrast to Algorithm 1, now we augment the saturation set S by at most one element in each iteration. Formally, S is augmented with at most one $q' \in Q$ such that, either there exists $q \in S$ and $\mathsf{a} \in \Sigma$ with $(q, !\mathsf{a}, q') \in \Delta$, or there exist $p, q \in S$, $p' \in Q$ and $\mathsf{a} \in \Sigma$ such that $(p, !\mathsf{a}, p') \in \Delta$ and $(q, ?\mathsf{a}, q') \in \Delta$. Additionally, we associate an integer-valued variable c that counts the number of nodes that are sufficient to cover the set S at the current iteration. Intuitively, when a state is added as the target of a broadcast transition, we copy the corresponding node responsible for the broadcast, whereas in case of a reception transition we need to copy two nodes involved in the action (one for broadcasting, one for receiving).

The reason for this modification is that the value of the new variable c at the end of the algorithm, as we will show in the following section, gives a bound on the number of nodes to cover all states in S that the algorithm returns, which is at most linear in the number of states in Q. We will also prove this linear bound on cutoff is tight, *i.e.*, we show an example of a family of protocols for which any covering execution needs at least linearly many nodes.

Algorithm 2 Refined saturation algorithm for coverability

$S := I$; $c := |I|$; $S' := \emptyset$
while $S \neq S'$ **do**
  $S' := S$
  if $\exists (q_1, !\mathsf{a}, q_2) \in \Delta$ s.t. $q_1 \in S'$ and $q_2 \notin S'$ **then**
    $S := S \cup \{q_2\}$; $c := c + 1$
  else if $\exists (q_1, !\mathsf{a}, q_2) \in \Delta$ and $(q_1', ?\mathsf{a}, q_2') \in \Delta$ s.t. $q_1, q_2, q_1' \in S'$ and $q_2' \notin S'$ **then**
    $S := S \cup \{q_2'\}$; $c := c + 2$
  end if
end while
return S

The refined saturation algorithm is presented in Algorithm 2.

Recall that $\texttt{REACH}_{\texttt{r}}(\mathcal{P})$ denotes the set of coverable states in the reconfigurable semantics of the broadcast protocol $\mathcal{P}$:

$$\texttt{REACH}_{\texttt{r}}(\mathcal{P}) = \{q \in Q \mid \exists \gamma_0 \xrightarrow{*}_{\texttt{r}} \gamma \in \texttt{Exec}_{\texttt{r}}(\mathcal{P}) \text{ with } \gamma = (\mathsf{N}, \mathsf{E}, \mathsf{L}) \text{ s.t. } q \in \mathsf{L}(\gamma)\}.$$

We adapt Lemma 2.15 for this modified version of the algorithm:

Lemma 3.5. *Algorithm 2 terminates and the set S the algorithm returns is exactly the set of coverable states in reconfigurable semantics. Formally, $S = \texttt{REACH}_{\texttt{r}}(\mathcal{P})$. Moreover, there exists an execution $\rho = \gamma_0 \to_{\texttt{r}} \gamma_1 \cdots \to_{\texttt{r}} \gamma$ with $\gamma = (\mathsf{N}, \mathsf{E}, \mathsf{L})$, such that $\mathsf{L}(\gamma) = S$.*

The correctness of Lemma 3.5 follows from that of Lemma 2.15. Indeed, both algorithms perform the same actions on the set S, except in the earlier, in each iteration, we augment the set with every possible states reachable in a single step, whereas in the latter we add them one at a time.

3.2 Tight bounds on cutoff and covering length

In this section, we show matching upper and lower bounds for cutoff and covering length. We achieve a linear bound on cutoff and a quadratic bound on covering length. Concerning the upper bounds, notice that it would be enough to show the result only for the lossy semantics and the result for reconfigurable semantics will automatically follow. Indeed, given a broadcast protocol $\mathcal{P}$, by Lemma 2.10, a lossy execution ρ reaching a configuration γ can be simulated by a reconfigurable one, say ρ', of same length that reaches γ' with same set of nodes as γ with same labels. Therefore, upper bounds for cutoff and covering length in the lossy semantics also apply to reconfigurable semantics. While both proofs use a similar idea, it is more technical in case of lossy semantics than the other. To ease the understanding, we begin with the reconfigurable semantics.

3.2.1 Upper bounds in reconfigurable semantics

Fix a broadcast protocol $\mathcal{P} = (Q, I, \Sigma, \Delta)$. Let S be the set of states computed by Algorithm 2 on protocol $\mathcal{P}$. Further assume $S_0, S_1, \ldots, S_m$ are the sets after each iteration of the algorithm, with $S_0 = I$ and $S_m = S$; and $c_0, c_1 \ldots, c_m$ be the respective values of the variable c with $c_0 = |I|$. We fix an ordering on the states in S (except the states in I) on the basis of insertion in S: for all $1 \leq i \leq m$, q_i is such that $q_i \in S_i \setminus S_{i-1}$.

We show that there exists an execution of size $O(n)$ and length $O(n^2)$ covering at the same time all states of S for reconfigurable networks. This can be formalized as follows.

Lemma 3.6. *Let $\mathcal{P} = (Q, I, \Sigma, \Delta)$ be a broadcast protocol and S be the set of states returned by Algorithm 2 on input $\mathcal{P}$. Then there exists an execution $\rho = \gamma_0 \xrightarrow{*}_{\texttt{r}} \gamma$ with $\gamma = (\mathsf{N}, \mathsf{E}, \mathsf{L})$ such that $\mathsf{L}(\gamma) = S$, $\#\mathsf{nodes}(\rho) \leq 2|Q|$ and $\#\mathsf{steps}(\rho) \leq 2|Q|^2$.*

Proof. We will show by induction on i that for every step $0 \leq i \leq m$ of Algorithm 2, there exists an initial configuration γ_0 and a reconfigurable execution $\rho_i = \gamma_0 \to_{\mathtt{r}} \cdots \gamma_{f(i)}$ for some $\gamma_{f(i)}$, with $\gamma_{f(i)} = (\mathsf{N}_i, \mathsf{E}_{f(i)}, \mathsf{L}_{f(i)})$, such that $\mathsf{L}_{f(i)}(\gamma_{f(i)}) = S_i$, $\#\mathsf{nodes}(\rho_i) = c_i$, and $\max_{\mathsf{n}} \#\mathsf{steps}(\rho_i, \mathsf{n}) \leq i$.

The base case $i = 0$ is obvious: take the initial configuration γ_0 with $|I|$ nodes and arbitrary topology, furthermore, label each node with a different initial state; its size is $|I|$, and the length of the execution is 0, hence so is the maximum active length.

To prove the induction step, we distinguish two cases: depending on whether q_{i+1} was added as the target state of a broadcast transition from some $q \in S_i$; or whether q_{i+1} is the target state of a reception from some $q \in S_i$ with matching broadcast between two states already in S_i.

Case 1: There exists $q \in S_i$ with $q \xrightarrow{!a} q_{i+1}$. We apply the induction hypothesis to step i, and exhibit an execution $\rho_i = \gamma_0 \to_{\mathtt{r}} \cdots \gamma_{f(i)}$ for some $f(i)$, with $\gamma_{f(i)} = (\mathsf{N}_i, \mathsf{E}_{f(i)}, \mathsf{L}_{f(i)})$, such that $\mathsf{L}_{f(i)}(\gamma_{f(i)}) = S_i$, $\#\mathsf{nodes}(\rho_i) = c_i$ and $\max_{\mathsf{n}} \#\mathsf{steps}(\rho_i, \mathsf{n}) \leq i$. Applying the copycat property (see Proposition 3.1), we construct an execution $\rho' = \gamma'_0 \to_{\mathtt{r}} \ldots \gamma'$ such that γ'_0 has one node more than γ_0, and, focusing on the nodes, γ' coincides with $\gamma_{f(i)}$, with an extra node n labelled by q, and finally, all nodes in γ' are disconnected (reconfiguration can be done in the last transition in ρ'). We then extend ρ' with a transition $\gamma' \xrightarrow{\mathsf{n},!a}_{\mathtt{r}} \gamma''$, which makes only progress node n from q to q_{i+1}. We write $\gamma'' = (\mathsf{N}'', \mathsf{E}'', \mathsf{L}'')$, and the resulting execution is denoted ρ_{i+1}. Then:

1. $\mathsf{L}''(\gamma'') = S_i \cup \{q_{i+1}\} = S_{i+1}$,
2. $\#\mathsf{nodes}(\rho_{i+1}) = c_i + 1 = c_{i+1}$,
3. $\max_{\mathsf{n}} \#\mathsf{steps}(\rho_{i+1}, \mathsf{n}) \leq \max_{\mathsf{n}} \#\mathsf{steps}(\rho_i, \mathsf{n}) + 1 \leq i + 1$; indeed, the active length of the copycat node along ρ' coincides with the active length of some existing node along ρ_i (see Proposition 3.1), and it is increased only by 1 in ρ_{i+1}.

This proves the induction step in the first case.

Case 2: There exists $q, q', q'' \in S_i$ with $q \xrightarrow{?a} q_{i+1}$ and $q' \xrightarrow{!a} q''$. The idea is similar to the previous case, but one should apply the copycat property twice, to both q and q'. We formalize this.

We apply the induction hypothesis to step i, and exhibit an execution $\rho_i = \gamma_0 \to_{\mathtt{r}} \ldots \gamma_{f(i)}$, with $\gamma_{f(i)} = (\mathsf{N}_i, \mathsf{E}_{f(i)}, \mathsf{L}_{f(i)})$, such that $\mathsf{L}_{f(i)}(\gamma_{f(i)}) = S_i$, $\#\mathsf{nodes}(\rho_i) = c_i$ and $\max_{\mathsf{n}} \#\mathsf{steps}(\rho_i, \mathsf{n}) \leq i$. Applying the copycat property (see Proposition 3.1) twice, to both q and q', we construct an execution $\rho' = \gamma'_0 \to_{\mathtt{r}} \ldots \gamma'$ such that γ'_0 has two nodes more than γ_0, and, focusing on the nodes, γ' coincides with $\gamma_{f(i)}$, with two extra nodes n, n' labelled by q, q' respectively and finally in γ', n and n' are connected to each other whereas all other nodes are disconnected (reconfiguration can be done in the last transition

in ρ'). We then extend ρ' with a transition $\gamma' \xrightarrow{\mathsf{n}',!a}_{\mathtt{r}} \gamma''$; this makes node n progress from q to q_{i+1} and node n' progress from q' to q''; all other nodes are unchanged. We write $\gamma'' = (\mathsf{N}'', \mathsf{E}'', \mathsf{L}'')$, and the resulting execution is denoted ρ_{i+1}. Then:

1. $\mathsf{L}''(\gamma'') = S_i \cup \{q'', q_{i+1}\} = S_{i+1}$ since $q'' \in S_i$,
2. $\#\mathsf{nodes}(\rho_{i+1}) = c_i + 2 = c_{i+1}$,
3. $\max_{\mathsf{n}} \#\mathsf{steps}(\rho_{i+1}, \mathsf{n}) \leq \max_{\mathsf{n}} \#\mathsf{steps}(\rho_i, \mathsf{n}) + 1 \leq i + 1$; indeed, the active length of any of the copycat nodes along ρ' coincides with the active length of some existing node along ρ_i (see Proposition 3.1), and it is increased by at most 1 in ρ_{i+1}.

This proves the induction step in the second case.

We therefore deduce that there exists an execution $\rho = \gamma_0 \xrightarrow{*}_{\mathtt{r}} \gamma$ with $\gamma = (\mathsf{N}, \mathsf{E}, \mathsf{L})$ such that:

1. $\mathsf{L}(\gamma) = S_m$;
2. $\#\mathsf{nodes}(\rho) = c_m \leq |I| + 2m \leq |I| + 2(|Q| - |I|) = 2|Q| - |I|$;
3. $\max_{\mathsf{n}} \#\mathsf{steps}(\rho, \mathsf{n}) \leq m \leq |Q| - |I|$.

Therefore $\#\mathsf{steps}(\rho) \leq \big(\#\mathsf{nodes}(\rho)\big) \cdot \Big(\max_{\mathsf{n}} \#\mathsf{steps}(\rho, \mathsf{n})\Big) \leq 2|Q|^2$, so that we established the desired bounds for Lemma 3.6. □

Lemma 3.6 states that we can effectively construct a reconfigurable execution with linear number of nodes and quadratic length that reaches a configuration such that for every state $q \in S$ (S is the set returned by Algorithm 2), there is a node labelled with q. Lemma 3.5 establishes the correlation between the set of coverable states in reconfigurable semantics and the set S, that they coincide. Thus, we conclude that for positive instances of coverability, there exists a covering execution ρ with the same bounds on the size and length, *i.e.*, $\#\mathsf{nodes}(\rho) \leq 2|Q|$ and $\#\mathsf{steps}(\rho) \leq 2|Q|^2$, where Q is the set of states in the broadcast protocol. This yields a linear upper bound on the cutoff and a quadratic upper bound on the covering length in reconfigurable semantics. This can be formalized as follows.

Theorem 3.7. *Let $\mathcal{P} = (Q, I, \Sigma, \Delta)$ be a broadcast protocol, $F \subseteq Q$ a set of target states. If $\mathtt{COVER}_{\mathtt{r}}(\mathcal{P}, F) \neq \emptyset$, then $\mathtt{CUTOFF}_{\mathtt{r}}(\mathcal{P}, F) \leq 2|Q|$ and $\mathtt{COVLEN}_{\mathtt{r}}(\mathcal{P}, F) \leq 2|Q|^2$.*

By Lemma 2.8, there is a reconfigurable execution if and only if there is one in the loss-on-reception semantics of the same size and length, such that the final configurations have the same set of nodes with same labels. Therefore, we conclude that the same upper bounds for the minimal covering executions also apply to the loss-on-reception semantics. This can be formalized as follows:

Corollary 3.8. *Let $\mathcal{P} = (Q, I, \Sigma, \Delta)$ be a broadcast protocol, $F \subseteq Q$ a set of target states. If $\texttt{COVER}_{\texttt{lr}}(\mathcal{P}, F) \neq \emptyset$, then $\texttt{CUTOFF}_{\texttt{lr}}(\mathcal{P}, F) \leq 2|Q|$ and $\texttt{COVLEN}_{\texttt{lr}}(\mathcal{P}, F) \leq 2|Q|^2$.*

We now prove similar upper bounds on cutoff and covering length for loss-on-broadcast semantics.

3.2.2 Upper bounds in loss-on-broadcast semantics

Perhaps surprisingly, Algorithm 2 also computes the set of states that can be covered by lossy executions. Concerning coverable states, the reconfigurable and lossy semantics thus agree. Yet, in Section 3.3, we will show that reconfigurable covering executions can be more succinct than lossy covering executions.

Fix a broadcast protocol $\mathcal{P} = (Q, I, \Sigma, \Delta)$. Let S be the set of states computed by Algorithm 2 on protocol $\mathcal{P}$. Again assume $S_0, S_1, \ldots, S_m$ are the sets after each iteration of the algorithm, with $S_0 = I$ and $S_m = S$; and $c_0, c_1 \ldots, c_m$ be the respective values of the variable c with $c_0 = |I|$. We fix an ordering on the states in S (except the states in I) on the basis of insertion in S: for all $1 \leq i \leq m$, q_i is such that $q_i \in S_i \setminus S_{i-1}$.

First we show an analogous result to Lemma 3.6, but for lossy semantics in which we effectively construct a lossy execution with linear number of nodes and of quadratic length that reaches a configuration such that for every $q \in S$, there is a node labelled with q. Then in Lemma 3.11, we will prove that Algorithm 2 indeed computes the set of coverable states for lossy semantics as well.

We show that there exists an execution of size $O(n)$ and length $O(n^2)$ covering at the same time all states of S for lossy networks, where n is the number of states in the broadcast protocol. This can be formalized as follows.

Lemma 3.9. *Let $\mathcal{P} = (Q, I, \Sigma, \Delta)$ be a broadcast protocol and S be the set of states returned by Algorithm 2 on input $\mathcal{P}$. Then there exists an execution $\rho = \gamma_0 \xrightarrow{*}_{\texttt{lb}} \gamma$ with $\gamma = (\mathsf{N}, \mathsf{E}, \mathsf{L})$ such that $\mathsf{L}(\gamma) \supseteq S$, $\#\mathsf{nodes}(\rho) \leq 2|Q|$ and $\#\mathsf{steps}(\rho) \leq 2|Q|^2$.*

For any configuration $\gamma = (\mathsf{N}, \mathsf{E}, \mathsf{L})$ and every node n, we write $\mathsf{L}(\mathsf{n}) = \times$ if n is not important any more in the execution, in other words all the required conditions in γ' such that $\gamma \xrightarrow{*}_{\texttt{lb}} \gamma'$ are still satisfied whatever $\mathsf{L}(\mathsf{n})$ is.

We will refine the construction from the proof of Lemma 3.6 (in the context of reconfigurable broadcast networks), and build inductively a lossy execution covering all states in S_i. Since the topology is static, some nodes which have "finished their jobs" will remain connected to other nodes, and may therefore continue to change states (contrary to Lemma 3.6 where they could be fully disconnected). Hence, in every such execution, every state $q \in S_i$ (which is then covered by the execution) will have a "main" corresponding node, whose label will remain q. All nodes which are not the main node of a state will

eventually be assigned $\times$ (when they would have finished their jobs), since their labels will become meaningless. We prove Lemma 3.9 below.

Proof. We will show by induction on i that for every step $0 \leq i \leq m$ of Algorithm 2, there exists an execution $\rho_i = \gamma_0 \to_{\mathtt{lb}} \dots \gamma_{f(i)}$ for some $f(i)$, with $\gamma_{f(i)} = (\mathsf{N}_i, \mathsf{E}_i, \mathsf{L}_{f(i)})$ (notice that the communication topology is fixed throughout ρ_i), and:

1. $\mathsf{L}_{f(i)}(\gamma_{f(i)}) \setminus \{\times\} = S_i$ and $\#\mathsf{nodes}(\rho) = c_i$,
2. $\max_{\mathsf{n}} \#\mathsf{steps}(\rho_i, \mathsf{n}) \leq i$ and $\max_{\mathsf{n}} \#\mathsf{nonlost\_steps}(\rho_i, \mathsf{n}) \leq 1$,
3. for every $q \in S_i$, there exists $\mathsf{n}_q^{\mathsf{main}} \in \mathsf{N}$ such that
   - $\mathsf{L}_{f(i)}(\mathsf{n}_q^{\mathsf{main}}) = q$ and $\#\mathsf{nonlost\_steps}(\rho_i, \mathsf{n}_q^{\mathsf{main}}) = 0$,
   - $\mathsf{n}_q^{\mathsf{main}} \sim \mathsf{n}$ implies $\mathsf{L}_{f(i)}(\mathsf{n}) = \times$, and if $\mathsf{n} \notin \{\mathsf{n}_q^{\mathsf{main}} \mid q \in S_i\}$, then $\mathsf{L}_{f(i)}(\mathsf{n}) = \times$.

The case $i = 0$ is obvious, by picking one main node per initial state in I, and by disconnecting all nodes; hence forming an initial configuration satisfying all the requirements.

To prove the induction step, we distinguish two cases: depending on whether q_{i+1} was added as the target state of a broadcast action $!a$ from some $q \in S_i$; or whether q_{i+1} is the target state of a reception from some $q \in S_i$ with matching broadcast between two states already in S_i.

Case 1: There exists $q \in S_i$ with $q \xrightarrow{!a} q_{i+1}$. We apply the induction hypothesis to step i, and exhibit the various elements of the statement. Applying the copycat property for lossy broadcast systems (that is, Proposition 3.3) with node $\mathsf{n}_q^{\mathsf{main}}$ on ρ_i, we build an execution $\rho' = \gamma'_0 \to_{\mathtt{lb}} \dots \gamma'$ such that $\gamma' = (\mathsf{N}', \mathsf{E}', \mathsf{L}')$ with $|\mathsf{N}'| = |\mathsf{N}| + 1$, and an appropriate injection ι. The fresh node $\mathsf{n}_{\mathsf{fresh}}$ is connected to nodes to which $\mathsf{n}_q^{\mathsf{main}}$ was connected before. Then we extend ρ' with $\gamma' \xrightarrow{\mathsf{n}_{\mathsf{fresh}}, !a}_{\mathtt{lb}} \gamma''$ and lose the message (this is for condition $\#\mathsf{nonlost\_steps}(\rho, \mathsf{n}_q^{\mathsf{main}}) = 0$ to be satisfied). We write $\gamma'' = (\mathsf{N}'', \mathsf{E}'', \mathsf{L}'')$, and the resulting execution is denoted ρ_{i+1}. We declare $\mathsf{n}_{q_{i+1}}^{\mathsf{main}} = \mathsf{n}_{\mathsf{fresh}}$. Then:

1. $\mathsf{L}''(\gamma'') \setminus \{\times\} = S_i \cup \{q_{i+1}\} = S_{i+1}$, and $\#\mathsf{nodes}(\rho_{i+1}) = c_i + 1 = c_{i+1}$,
2. $\max_{\mathsf{n}} \#\mathsf{steps}(\rho_{i+1}, \mathsf{n}) \leq \max_{\mathsf{n}} \#\mathsf{steps}(\rho_i, \mathsf{n}) + 1 \leq i + 1$; indeed, the active length of the copycat node along ρ' coincides with the active length of some existing node along ρ_i (see Proposition 3.3), and it is increased only by 1 in ρ_{i+1}, and $\max_{\mathsf{n}} \#\mathsf{nonlost\_steps}(\rho_{i+1}, \mathsf{n}) \leq 1$,
3. further,
   - $\mathsf{L}''(\mathsf{n}_{q_{i+1}}^{\mathsf{main}}) = q_{i+1}$ and $\#\mathsf{nonlost\_steps}(\rho_{i+1}, \mathsf{n}_{q_{i+1}}^{\mathsf{main}}) = 0$,

- $\mathsf{n}^{\mathsf{main}}_{q_{i+1}} \sim \mathsf{n}$ implies $\mathsf{n}^{\mathsf{main}}_{q} \sim \mathsf{n}$, and by induction hypothesis, $\mathsf{L}_{f(i)}(\mathsf{n}) = \times$, hence $\mathsf{L}''(\mathsf{n}) = \times$. Furthermore, if $\mathsf{n} \notin \{\mathsf{n}^{\mathsf{main}}_{q} \mid q \in S_{i+1}\}$, then $\mathsf{L}''(\mathsf{n}) = \times$. Indeed, all such nodes n are present in $\gamma_{f(i)}$ with label $\times$ (by induction hypothesis), and hence it remains labelled by $\times$ in γ' and γ'', respectively.

This proves the induction step in the first case.

Case 2: There exist $q, q', q'' \in S_i$ such that $q \xrightarrow{?a} q_{i+1}$ and $q' \xrightarrow{!a} q''$. We apply the induction hypothesis to step i, and exhibit the various elements of the statement. Applying twice the copycat property (that is, Proposition 3.3), once with node $\mathsf{n}^{\mathsf{main}}_{q}$ and once with node $\mathsf{n}^{\mathsf{main}}_{q'}$, we build an execution $\rho' = \gamma'_0 \to_{\mathtt{1b}} \dots \gamma'$ such that $\gamma' = (\mathsf{N}', \mathsf{E}', \mathsf{L}')$ with $|\mathsf{N}'| = |\mathsf{N}| + 2$, and an appropriate injection ι. The two fresh nodes $\mathsf{n}_{\mathsf{fresh}}$ and $\mathsf{n}'_{\mathsf{fresh}}$ are only connected to $\times$-nodes in γ' (by induction hypothesis on $\mathsf{n}^{\mathsf{main}}_{q}$ and $\mathsf{n}^{\mathsf{main}}_{q'}$ respectively). We transform γ'_0 into γ''_0 by connecting the two nodes $\mathsf{n}_{\mathsf{fresh}}$ and $\mathsf{n}'_{\mathsf{fresh}}$. By Proposition 3.3, we know that those two nodes do not perform any real sending (*i.e.*, $\#\mathsf{nonlost\_steps}(\rho', \mathsf{n}_{\mathsf{fresh}}) = 0$ and $\#\mathsf{nonlost\_steps}(\rho', \mathsf{n}'_{\mathsf{fresh}}) = 0$), hence this new connection will not affect the labels of the nodes, and we can safely apply the same transitions as in ρ' from γ''_0 to get an execution $\rho'' = \gamma''_0 \to_{\mathtt{1b}} \dots \gamma''$, where γ'' coincides with γ', with an extra connection between nodes $\mathsf{n}_{\mathsf{fresh}}$ and $\mathsf{n}'_{\mathsf{fresh}}$. Then, we extend ρ'' with $\gamma'' \xrightarrow{\mathsf{n}'_{\mathsf{fresh}}, !a}_{\mathtt{1b}} \gamma'''$. We assume it is a real sending, hence: node $\mathsf{n}_{\mathsf{fresh}}$ can progress from state q to q_{i+1}, and node $\mathsf{n}'_{\mathsf{fresh}}$ can progress from q' to q''. All other nodes which are connected to $\mathsf{n}'_{\mathsf{fresh}}$ are labelled by $\times$ in γ'', hence their labels will remain $\times$ in γ'''. We write $\gamma''' = (\mathsf{N}''', \mathsf{E}''', \mathsf{L}''')$, and the resulting execution is denoted ρ_{i+1}. We relabel $\mathsf{n}'_{\mathsf{fresh}}$ to $\times$, and declare $\mathsf{n}^{\mathsf{main}}_{q_{i+1}} = \mathsf{n}_{\mathsf{fresh}}$. Then:

1. $\mathsf{L}'''(\gamma''') \setminus \{\times\} = S_i \cup \{q'', q_{i+1}\} = S_{i+1}$, and $\#\mathsf{nodes}(\rho_{i+1}) = c_i + 2 = c_{i+1}$,
2. $\max_{\mathsf{n}} \#\mathsf{steps}(\rho_{i+1}, \mathsf{n}) \leq \max_{\mathsf{n}} \#\mathsf{steps}(\rho_i, \mathsf{n}) + 1 \leq i + 1$; indeed, the active length of the copycat node along ρ' coincides with the active length of some existing node along ρ_i (see Proposition 3.3), and it is increased only by 1 in ρ_{i+1}, and $\max_{\mathsf{n}} \#\mathsf{nonlost\_steps}(\rho_{i+1}, \mathsf{n}) \leq 1$,
3. further,
   - $\mathsf{L}'''(\mathsf{n}^{\mathsf{main}}_{q_{i+1}}) = q_{i+1}$ and $\#\mathsf{nonlost\_steps}(\rho_{i+1}, \mathsf{n}^{\mathsf{main}}_{q_{i+1}}) = 0$,
   - $\mathsf{n}^{\mathsf{main}}_{q_{i+1}} \sim \mathsf{n}$ implies $\mathsf{n}^{\mathsf{main}}_{q} \sim \mathsf{n}$, and by induction hypothesis, $\mathsf{L}_{f(i)}(\mathsf{n}) = \times$, hence $\mathsf{L}'''(\mathsf{n}) = \times$. Furthermore, if $\mathsf{n} \notin \{\mathsf{n}^{\mathsf{main}}_{q} \mid q \in S_{i+1}\}$, then $\mathsf{L}'''(\mathsf{n}) = \times$. Indeed, nodes of this kind that were already present in $\gamma_{f(i)}$ has label $\times$ in $\gamma_{f(i)}$ (by induction hypothesis), thus remains labelled by $\times$ in γ', γ'' and γ''', respectively, moreover, the new node $\mathsf{n}'_{\mathsf{fresh}}$ that is not a "main" node is labelled by $\times$ in γ'''.

This proves the induction step in the second case.

We therefore deduce that there exists an execution $\rho = \gamma_0 \xrightarrow{*}_{\mathtt{1b}} \gamma$ with $\gamma = (\mathsf{N}, \mathsf{E}, \mathsf{L})$ such that:

1. $\mathsf{L}(\gamma) \setminus \{\times\} = S_m$, therefore, $\mathsf{L}(\gamma) \supseteq S$;
2. $\#\mathsf{nodes}(\rho) = c_m \leq |I| + 2m \leq |I| + 2(|Q| - |I|) = 2|Q| - |I|$;
3. $\max_{\mathsf{n}} \#\mathsf{steps}(\rho, \mathsf{n}) \leq m \leq |Q| - |I|$.

Therefore $\#\mathsf{steps}(\rho) \leq \big(\#\mathsf{nodes}(\rho)\big) \cdot \Big(\max_{\mathsf{n}} \#\mathsf{steps}(\rho, \mathsf{n})\Big) \leq 2|Q|^2$, so that we established the desired bounds for Lemma 3.9. □

Let us illustrate the inductive construction of a witness execution in the proof of Lemma 3.9 for lossy semantics on an example.

Example 3.10. *Consider the broadcast protocol from Figure 3.3. The construction of a lossy covering execution for this protocol is presented in Figure 3.4. Configurations here are represented vertically: they involve 10 nodes, and the static communication topology is given for the first configuration only, for the sake of readability. For a compact representation of the figure, several broadcasts (of the same message type, from different nodes) may happen in a* macro step *that merges several consecutive steps. This is for instance the case in the first macro step, where* a *is being broadcast from the node in set* S_1*, as well as from the first node in set* S_2*. Dashed arrows are used to represent that a node is not involved in some macro step and thus stays in the same state. In the execution, the nodes that are performing a real broadcast are coloured yellow, the ones which change their state upon reception of a message are coloured grey, and blue nodes indicate the main nodes for the coverable states. Note also that the main nodes never perform a real broadcast, and they are indeed only connected to '*$\times$*' nodes.*

We now show that the set S is indeed the set of all coverable states in lossy semantics. Recall that $\mathtt{REACH}_{\mathtt{lb}}(\mathcal{P})$ denotes the set of coverable states in the loss-on-broadcast semantics of the broadcast protocol $\mathcal{P}$:

$$\mathtt{REACH}_{\mathtt{lb}}(\mathcal{P}) = \{q \in Q \mid \exists \gamma_0 \xrightarrow{*}_{\mathtt{lb}} \gamma \in \mathtt{Exec}_{\mathtt{lb}}(\mathcal{P}) \text{ with } \gamma = (\mathsf{N}, \mathsf{E}, \mathsf{L}) \text{ s.t. } q \in \mathsf{L}(\gamma)\}.$$

We present the above statement in the style of Lemma 3.5 for reconfigurable networks.

Lemma 3.11. *The set* S *returned by Algorithm 2 is the set of all coverable states of* $\mathcal{P}$ *in loss-on-broadcast semantics. Formally,* $S = \mathtt{REACH}_{\mathtt{lb}}(\mathcal{P})$*. Moreover, there exists an execution* $\rho = \gamma_0 \to_{\mathtt{lb}} \gamma_1 \cdots \to_{\mathtt{lb}} \gamma$ *with* $\gamma = (\mathsf{N}, \mathsf{E}, \mathsf{L})$*, such that* $\mathsf{L}(\gamma) = S$*.*

Proof. First notice that from Lemma 3.9, there is an execution $\rho = \gamma_0 \xrightarrow{*}_{\mathtt{lb}} \gamma$ with $\gamma = (\mathsf{N}, \mathsf{E}, \mathsf{L})$, such that $\mathsf{L}(\gamma) \supseteq S$. In other words, all states in S are coverable: $S \subseteq \mathtt{REACH}_{\mathtt{lb}}(\mathcal{P})$. To prove the other direction, assume $q \in \mathtt{REACH}_{\mathtt{lb}}(\mathcal{P})$. Then there exists a loss-on-broadcast execution $\rho = \gamma_0 \to_{\mathtt{lb}} \gamma_1 \cdots \to_{\mathtt{lb}} \gamma$ with $\gamma = (\mathsf{N}, \mathsf{E}, \mathsf{L})$, such that $q \in \mathsf{L}(\gamma)$. By Lemma 2.10, we can construct a reconfigurable execution $\rho' = \gamma'_0 \to_{\mathtt{r}} \gamma'_1 \cdots \to_{\mathtt{r}} \gamma'$ with $\gamma' = (\mathsf{N}, \mathsf{E}', \mathsf{L})$ such that γ and γ' have the same set of nodes with the same labels, therefore, $q \in \mathsf{L}(\gamma')$. Hence, $q \in \mathtt{REACH}_{\mathtt{r}}(\mathcal{P})$. By Lemma 3.5, we conclude $q \in S$.

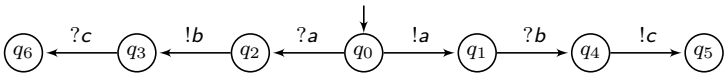


Figure 3.3: Illustrating example for the saturation algorithm.

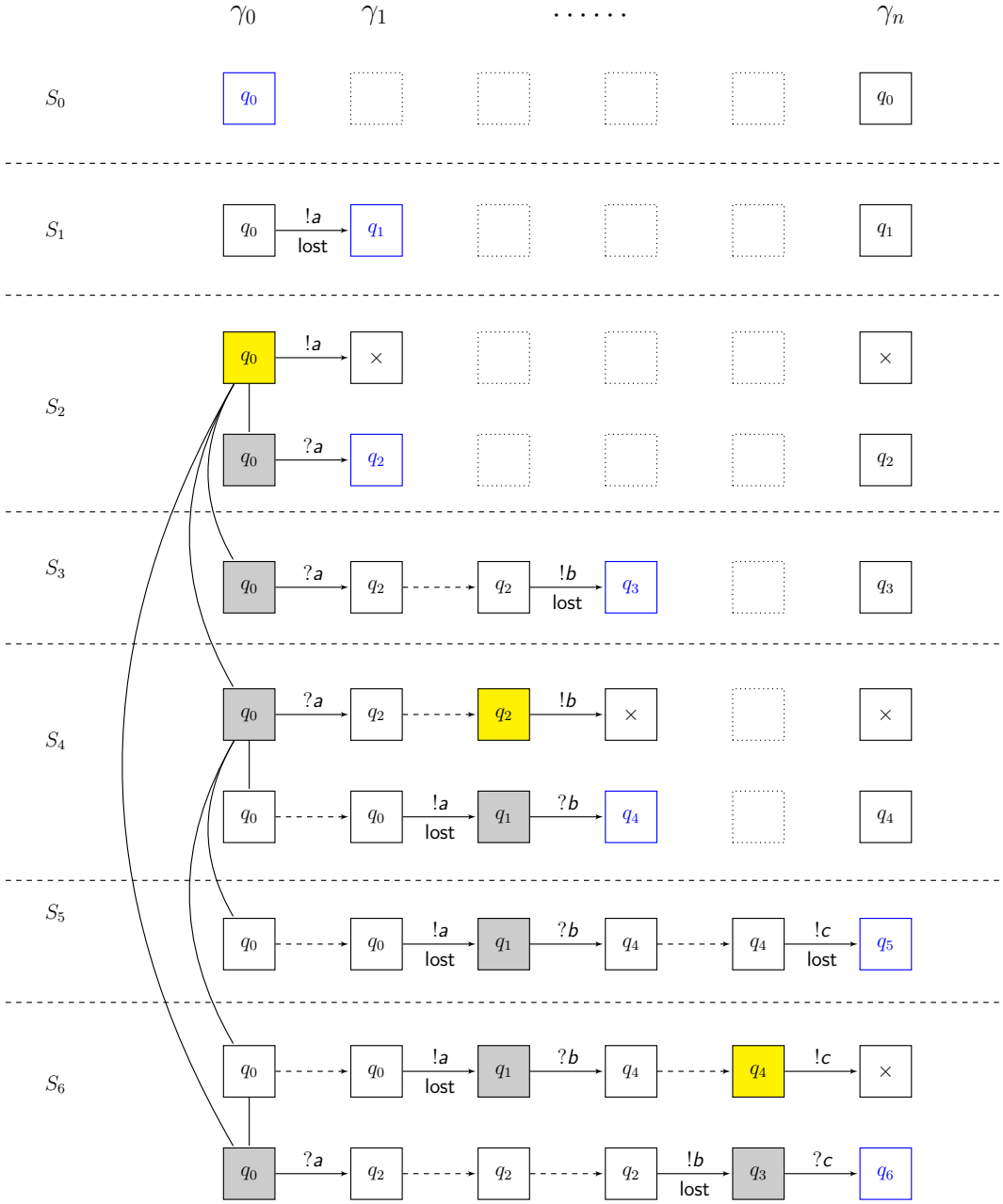


Figure 3.4: Lossy covering execution from the saturation algorithm on the protocol in Figure 3.3. Configurations are represented vertically; for readability, macro steps merge several broadcasts.

Let us prove the last statement. Again from Lemma 3.9, there is an execution $\rho = \gamma_0 \xrightarrow{*}_{\mathtt{lb}} \gamma$ with $\gamma = (\mathsf{N}, \mathsf{E}, \mathsf{L})$, such that $\mathsf{L}(\gamma) \supseteq S$. Furthermore, since $S = \mathtt{REACH}_{\mathtt{lb}}(\mathcal{P})$, all states in $\mathsf{L}(\gamma)$ are in S. Hence, $\mathsf{L}(\gamma) = S$. □

By Lemma 3.9, one can effectively construct a lossy execution with linear number of nodes and quadratic length that reaches a configuration such that for every state $q \in S$ (S is the set returned by Algorithm 2), there is a node labelled with q. Furthermore, from Lemma 3.11, the set S is exactly the set of coverable states of the broadcast protocol $\mathcal{P}$ in lossy semantics. Thus, we conclude that for positive instances of coverability, there exists a covering execution ρ with the same bounds on the size and length, *i.e.*, $\#\mathsf{nodes}(\rho) \leq 2|Q|$ and $\#\mathsf{steps}(\rho) \leq 2|Q|^2$, where Q is the set of states in the broadcast protocol. This yields a linear upper bound on the cutoff and a quadratic upper bound on the covering length in loss-on-broadcast semantics. This can be formalized as follows.

Theorem 3.12. *Let $\mathcal{P} = (Q, I, \Sigma, \Delta)$ be a broadcast protocol, $F \subseteq Q$ a set of target states. If $\mathtt{COVER}_{\mathtt{lb}}(\mathcal{P}, F) \neq \emptyset$, then $\mathtt{CUTOFF}_{\mathtt{lb}}(\mathcal{P}, F) \leq 2|Q|$ and $\mathtt{COVLEN}_{\mathtt{lb}}(\mathcal{P}, F) \leq 2|Q|^2$.*

Having shown the upper bounds for cutoff and covering length in both semantics, we now show lower bounds for them. We show that the linear bound on cutoff and quadratic bound on covering length are tight in both semantics.

3.2.3 Matching lower bounds for cutoff and covering length

In this section, we show that the linear bound on the cutoff and the quadratic bound on the length of witness executions are tight, both for the reconfigurable and the lossy broadcast networks. We give an example of a family of protocols for which any covering execution needs at least linearly many nodes and a quadratic number of steps (in both semantics).

Theorem 3.13. *There exists a family of broadcast protocols $(\mathcal{P}_n)_n$ with $\mathcal{P}_n = (Q_n, I_n, \Sigma_n, \Delta_n)$, and target states $F_n \subseteq Q_n$ with $|Q_n| \in O(n)$, such that for every n, $\mathtt{COVER}_{\mathtt{r}}(\mathcal{P}_n, F_n) \neq \emptyset$, $\mathtt{COVER}_{\mathtt{lb}}(\mathcal{P}_n, F_n) \neq \emptyset$, and any witness reconfigurable (resp., lossy) execution has size $O(n)$ and length $O(n^2)$.*

Proof. Consider $\mathcal{P}_n$, as depicted in Figure 3.5 with $2n{+}1$ states: $Q_n = \{q_0, q_1, \dots q_{2n}\}$ where we identify q_{2n} with ☺, $\Sigma_n = \{\mathsf{a}_i, \mathsf{b}_i \mid 1 \leq i \leq n\}$, and let $F_n = \{☺\}$. The transition relation is defined as follows: for every $1 \leq i \leq n$, $q_{2i-2} \xrightarrow{!a_i} q_{2i-1}$, $q_{2i-1} \xrightarrow{!b_i} q_{2i-1}$ and $q_{2i-1} \xrightarrow{?b_i} q_{2i}$. Recall that the self-loops upon receiving any message are implicit.

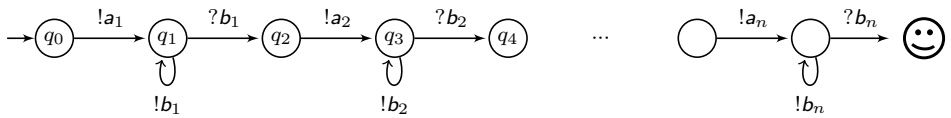


Figure 3.5: Broadcast protocol with linear cutoff and quadratic covering length.

Any covering reconfigurable execution involves at least $n{+}1$ nodes. Indeed, to reach ☺, all the b_i's have to be broadcast at least once and the node responsible for the last

broadcast of b_i stays forever in state q_{2i-1}. Therefore, a separate node is required for each $!\mathsf{b}_i$. An additional node must reach ☺, so that, at least $n+1$ nodes are required.

Any covering reconfigurable execution involves at least $\frac{n^2+5n}{2}$ steps. Indeed, in a minimal covering execution, there will be exactly one node in each q_i. Moreover, a broadcast of message b_i to happen, at least $n+2-i$ broadcasts of a_i also need to happen by that node. As a consequence, the covering length in reconfigurable semantics is at least $n + \sum_{i=1}^{n}(n+2-i) = n + n^2 + 2n - \frac{n(n+1)}{2} = \frac{n^2+5n}{2}$.

By Lemma 2.10, from an execution in the lossy semantics, one can construct one in the reconfigurable semantics with same number of nodes and same execution length. Therefore, the above bounds also provide lower bounds on the cutoff and covering length for lossy networks. □

The linear bound for cutoff and quadratic bound for covering length are indeed tight in both semantics. However, we now show that reconfigurable executions can be more succinct than the lossy ones, in terms of number of nodes.

3.3 Succinctness of reconfigurations compared to message losses

In the previous section, we have seen the reconfigurable and lossy networks mostly enjoy the same properties: the coverability problem is decidable in both semantics, and for positive instances of the coverability problem, the same tight bounds on cutoff and covering length hold. We, however, now show that the reconfigurable executions can be more succinct by a linear factor than the lossy ones, in terms of number of nodes. This can be formalized as follows.

Theorem 3.14. *There exists a family of broadcast protocols $(\mathcal{P}_n)_n$ with $\mathcal{P}_n = (Q_n, I_n, \Sigma_n, \Delta_n)$ and target states $F_n \subseteq Q_n$ such that for every n:*

- *there exists a reconfigurable covering execution in $\mathcal{P}_n$ with 3 nodes; and*
- *any lossy covering execution in $\mathcal{P}_n$ requires $O(n)$ nodes.*

Proof. Consider $\mathcal{P}_n$, as depicted in Figure 3.6 with $3n+2$ states: $Q_n = \{q_0\} \cup \{q_{2i-1}, q_{2i}, r_i \mid 1 \leq i \leq n\} \cup \{☺, \bot\}$, where we identify r_n with ☺, $\Sigma_n = \{\mathsf{a}\} \cup \{\mathsf{b}_i \mid 1 \leq i \leq n\}$, and let $F_n = \{☺\}$. The transition relation is defined as follows: $q_0 \xrightarrow{!\mathsf{a}} q_0$; $q_0 \xrightarrow{?\mathsf{b}_1} r_1$; for every $1 \leq i \leq n$, $q_0 \xrightarrow{?\mathsf{b}_i} \bot$, $q_{2i-2} \xrightarrow{?\mathsf{a}} q_{2i-1}$, $q_{2i-1} \xrightarrow{!\mathsf{b}_i} q_{2i}$; and for every $2 \leq i \leq n$, $r_{i-1} \xrightarrow{?\mathsf{b}_i} r_i$. Recall that the self-loops upon receiving any message are implicit.

In reconfigurable semantics, there exists a covering execution with 3 nodes for the protocol in Figure 3.6, which is depicted in Figure 3.7. Coloured nodes here broadcast a

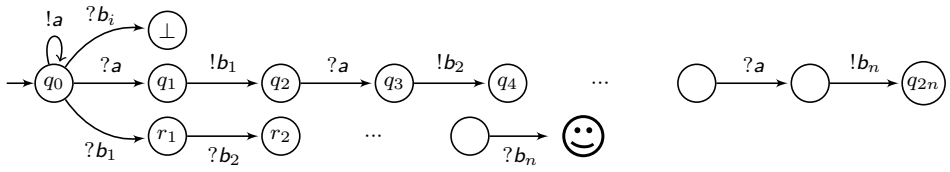


Figure 3.6: Example where reconfigurable semantics needs less nodes than lossy semantics.

message in the step leading to the next configuration. Along that execution, the top node always remains at q_0 and alternatively broadcasts a to the middle node and disconnects; the middle node follows the chain of q_i's and alternatively broadcasts b_i's to the bottom node, which gradually progresses along the chain of states r_i and reaches ☺.

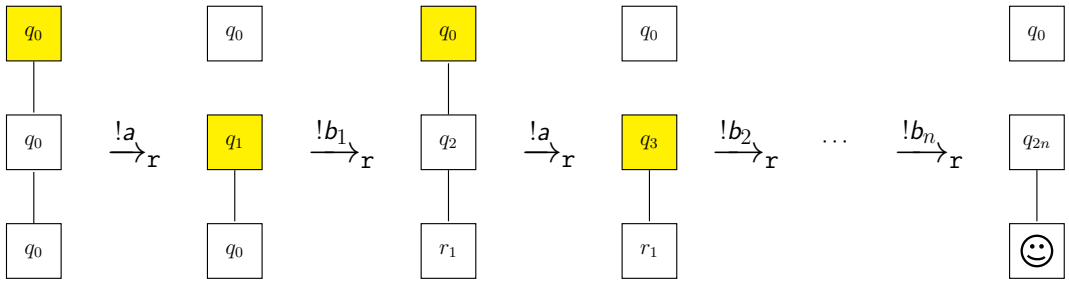


Figure 3.7: A covering reconfigurable execution with 3 nodes on the protocol from Figure 3.6.

Let us now argue that in the lossy semantics, $O(n)$ nodes are needed to cover ☺. Firstly, one node, say $\mathsf{n}_{☺}$, is needed to reach the target state, after having received sequentially all the b_i's (which should then correspond to real broadcasts). Towards a contradiction, assume there is a node n which makes $\mathsf{n}_{☺}$ progress twice, that is, n is connected to $\mathsf{n}_{☺}$ and performs at least two real broadcasts, say $!b_i$ and $!b_j$ with $i < j$. Node n needs to receive $j - i > 0$ times the message a after the real $!b_i$ has occurred, hence there must be at least one node in state q_0 connected to n after the real $!b_i$ by n. This is not possible, since this node has received the real $!b_i$ while being in q_0, leading to $\bot$ if $i > 1$, otherwise $\bot$ or r_1. Hence, each broadcast $!b_i$ needs to be sent by a different node. This requires at least $n{+}1$ nodes, say $\{\mathsf{n}_i \mid 1 \leq i \leq n\} \cup \{\mathsf{n}_{☺}\}$: node n_i is responsible for broadcasting (with no loss) b_i and $\mathsf{n}_{☺}$ progresses towards ☺. Notice that $\mathsf{n}_{☺}$ might be the node responsible for broadcasting all the a's. We conclude that $n{+}1$ is a lower bound on the number of nodes needed to cover ☺ in the lossy semantics.

In fact, we can prove that $n{+}1$ nodes do actually suffice in lossy semantics to cover ☺. Let $\mathsf{N} = \{\mathsf{n}_i \mid 1 \leq i \leq n\} \cup \{\mathsf{n}_{☺}\}$ and consider the static communication topology defined by $\mathsf{n}_i \sim \mathsf{n}_{☺}$ for every i. In the covering lossy execution, node $\mathsf{n}_{☺}$ initially broadcasts a's, so that all its neighbours, the n_i's can move to q_{2i-1}, using lossy broadcasts. Then each node n_i broadcasts its message b_i to $\mathsf{n}_{☺}$, starting with n_1 until n_n, so that $\mathsf{n}_{☺}$ reaches ☺. Figure 3.8 depicts a lossy execution with $n{+}1$ nodes. In this picture, yellow nodes perform real broadcasts, light blue nodes perform lossy broadcasts and grey nodes receive messages corresponding to the real broadcasts. □

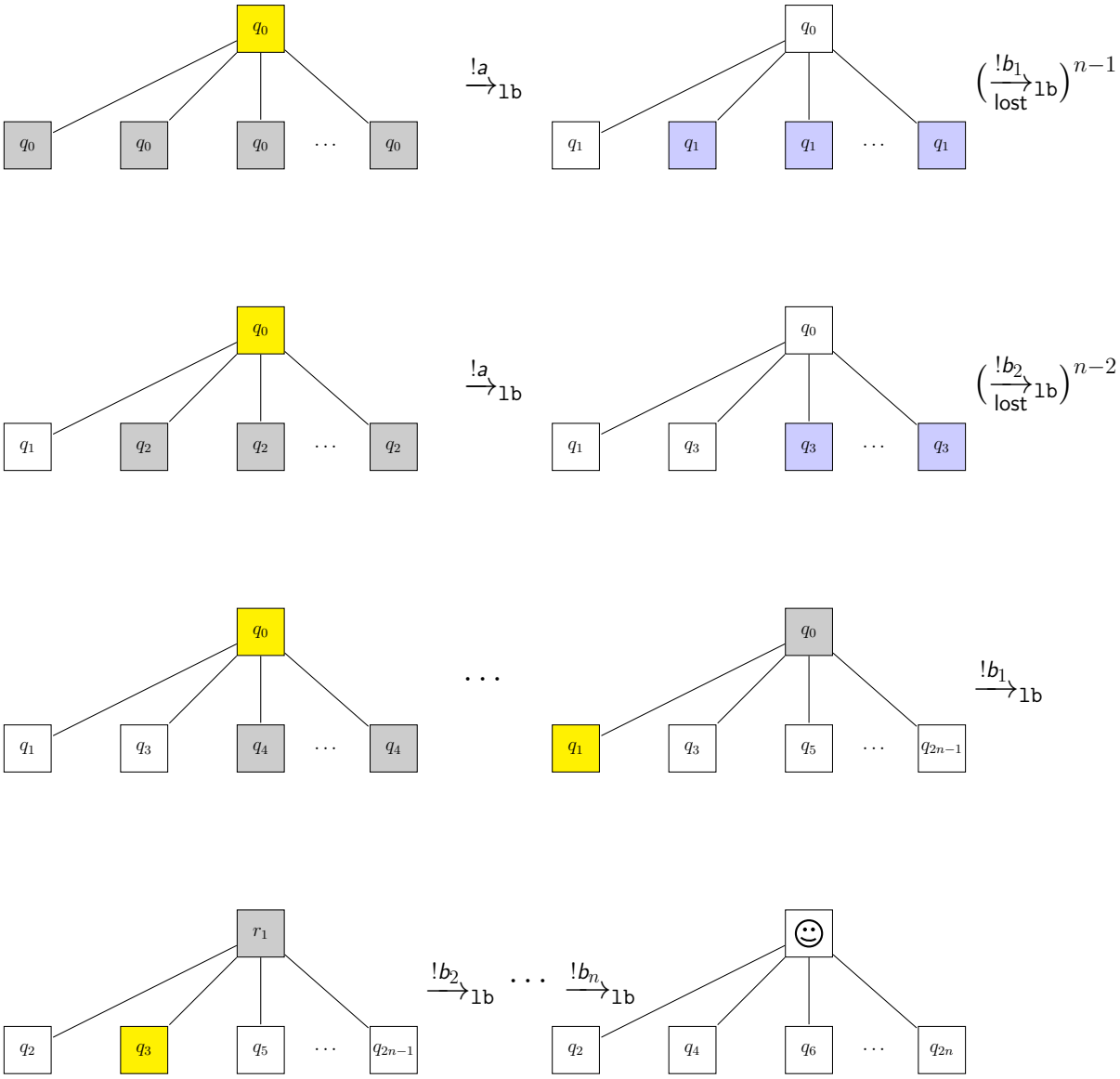


Figure 3.8: A covering loss-on-broadcast execution on the protocol from Figure 3.6.

Despite satisfying the same tight bounds, we have shown that reconfiguration can be more succinct than message losses on broadcast. In the following section we study the complexity of the MinNodes problem.

3.4 Complexity of deciding the size of minimal witnesses

We now consider the decision problem of determining the minimal size of coverability witnesses for the reconfigurable, as well as the lossy semantics.

Theorem 3.15. MinNodes *is* NP*-complete for both reconfigurable and lossy broadcast*

networks.

Proof. The NP-hardness of MinNodes is proved by reduction from SetCover, which is known to be NP-complete [Kar72]. Recall that SetCover takes as input a finite set of elements $\mathcal{U}$, a collection $\mathcal{S}$ of subsets of $\mathcal{U}$ and an integer k, and returns yes iff there exists a sub-collection of $\mathcal{S}$ of size at most k that covers $\mathcal{U}$.

SetCover
Input: A finite set of elements $\mathcal{U}$, a family $\mathcal{S}$ of subsets of $\mathcal{U}$ and $k \in \mathbb{N}$.
Output: Yes if and only if there exists a subfamily $\mathcal{C}$ of $\mathcal{S}$ whose union is $\mathcal{U}$ and $|\mathcal{C}| \leq k$.

Lemma 3.16. SetCover *reduces to* MinNodes *in logarithmic space.*

Proof. Given an instance of the SetCover problem $(\mathcal{U}, \mathcal{S}, k)$, let us explain how to construct in logarithmic space, a protocol $\mathcal{P}$ with a set of target states F and some integer k' such that $(\mathcal{U}, \mathcal{S}, k)$ is a positive instance of SetCover if and only if $(\mathcal{P}, F, k')$ is a positive instance of MinNodes.

For $\mathcal{U} = \{a_1, a_2, \ldots, a_n\}$ and $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$, we define the protocol $\mathcal{P} = (Q, I, \Sigma, \Delta)$ (depicted in Figure 3.9) as follows:

- $Q = \{s_1, s_2, \ldots, s_m\} \uplus \{q_1, q_2, \ldots, q_n\} \uplus \{\text{☺}\}$;
- $I = \{s_1, s_2, \ldots, s_m\} \cup \{q_1\}$;
- $\Sigma = \mathcal{U}$;
- $\Delta = \{(s_j, !a, s_j) \mid a \in S_j, 1 \leq j \leq m\} \cup \{(q_i, ?a_i, q_{i+1}) \mid 1 \leq i < n\} \cup \{(q_n, ?a_n, \text{☺})\}$.

We further let $F = \{\text{☺}\}$, and $k' = k + 1$. Clearly this reduction can be done in logarithmic space. It remains to show that $\mathcal{U}, \mathcal{S}$ has a cover of size k if and only if there exists a reconfigurable/lossy execution for $\mathcal{P}$ covering F and with k' nodes.

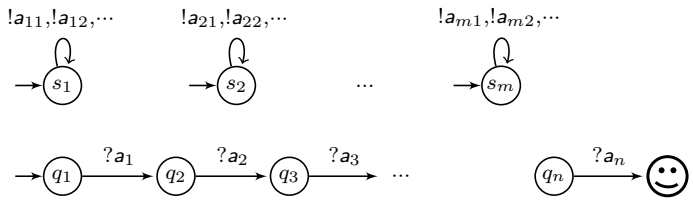


Figure 3.9: Reduction from SetCover to MinNodes.

Suppose the SetCover instance is positive, and $\mathcal{C} = \{S'_1, \ldots, S'_k\}$ is a cover of size k (notice that if there exists a cover of size less than k, there is one of size exactly k). Let us build a *static* execution ρ (with no message losses) that covers F in $\mathcal{P}$. Of course, then ρ is

also a reconfigurable/lossy execution. Choose $\gamma_0 = (\mathsf{N}, \mathsf{E}, \mathsf{L}_0)$ where $\mathsf{N} = \{\mathsf{n}_1, \ldots, \mathsf{n}_k\} \cup \{\mathsf{n}\}$, $\mathsf{L}_0(\mathsf{n}) = q_1$, $\mathsf{L}_0(\mathsf{n}_i) = s'_i$ for all $1 \leq i \leq k$ (assuming $s'_i = s_j$ whenever $S'_i = S_j$). Since $\mathcal{C}$ covers each $\mathsf{a}_i \in \mathcal{U}$, there exists j such that $\mathsf{a}_i \in S'_j$. Thus, the broadcast transition $(s'_j, !\mathsf{a}_i, s'_j)$ will be enabled, and some node may take the corresponding reception transition $(q_i, ?\mathsf{a}_i, q_{i+1})$. As for the static communication topology, it is sufficient to assume that $\mathsf{n}_j \sim \mathsf{n}$ for all $1 \leq j \leq k$. Messages a_i's are then being broadcast one after the other, and node n sequentially receives them, so that we reach a configuration $\gamma = (\mathsf{N}, \mathsf{E}, \mathsf{L})$ such that $\mathsf{L}(\mathsf{n}) = ☺$. The execution has $k{+}1$ nodes.

Assume now, that the SetCover instance is negative, thus there is no cover of size k or less. For a contradiction, assume there exists an initial configuration $\gamma_0 = (\mathsf{N}, \mathsf{E}_0, \mathsf{L}_0)$ of size $k{+}1$ and a reconfigurable (resp., lossy) execution ρ from γ_0 that covers ☺. A necessary condition is that $q_1 \in \mathsf{L}_0(\gamma_0)$ and a single such node is sufficient, so we let $\mathsf{N} = \{\mathsf{n}_1, \ldots, \mathsf{n}_k, \mathsf{n}\}$ with $\mathsf{L}_0(\mathsf{n}) = q_1$, $\mathsf{L}_0(\mathsf{n}_j) = s'_j$ for some s'_j where $1 \leq j \leq k$. Since the SetCover instance is negative, there exists $\mathsf{a}_i \in \mathcal{U}$ such that $\mathsf{a}_i \notin \bigcup_{1 \leq j \leq k} S'_j$. Therefore, none of the nodes will be able to broadcast the message a_i (assuming $s'_i = s_j$ whenever $S'_i = S_j$) and the corresponding reception $(q_i, ?\mathsf{a}_i, q_{i+1})$ will never be performed. This contradicts the fact that ρ covers ☺. □

To prove the NP-membership of the MinNodes problem, it suffices to observe that both the size and length of a minimal covering execution ρ are polynomially bounded, thanks to the Theorems 3.7 and 3.12, respectively. The size of any configuration $\gamma = (\mathsf{N}, \mathsf{E}, \mathsf{L})$ in ρ is thus polynomially bounded: N is polynomially bounded, hence so is E, and finally, the labelling function L can be represented as pairs (n, q) such that $\mathsf{L}(\mathsf{n}) = q$, which is again of size polynomial in the size of $\mathcal{P}$. Furthermore, a step $\gamma \to \gamma'$, with $\gamma' = (\mathsf{N}', \mathsf{E}', \mathsf{L}')$, can be verified efficiently: in reconfigurable semantics, we have to check whether there exists $\mathsf{n} \in \mathsf{N}$, $\mathsf{n}' \in \mathsf{N}'$ such that $(\mathsf{L}(\mathsf{n}), !a, \mathsf{L}'(\mathsf{n})) \in \Delta$, for every $\mathsf{n}' \in \mathsf{Neigh}_\gamma(\mathsf{n})$, $(\mathsf{L}(\mathsf{n}), ?a, \mathsf{L}'(\mathsf{n}')) \in \Delta$, and for every $\mathsf{n}'' \notin \mathsf{Neigh}_\gamma(\mathsf{n})$, $\mathsf{L}'(\mathsf{n}'') = \mathsf{L}(\mathsf{n}'')$; similarly for the lossy semantics (in that case, we need to first guess if the transition is a lossy one). It is thus possible to implement a guess-and-check non-deterministic polynomial time algorithm for the MinNodes problem, that non-deterministically guesses an execution with k nodes, and of length polynomial in the size of the broadcast protocol. Hence, the MinNodes problem is in NP. This concludes the proof of Theorem 3.15. □

The problem of determining the size of a minimal witness is therefore NP-complete in both reconfigurable and loss-on-broadcast semantics.

3.5 Concluding remarks

We have defined the notions of cutoff and covering length with respect to coverability for broadcast protocols. In this chapter, we have shown tight bounds on these measures for reconfigurable and lossy semantics. To be precise, we have shown a linear upper bound on the cutoff and a quadratic upper bound on the covering length for the reconfigurable and

lossy networks. The proof is mainly based on a monotonicity property of the networks: if a state can be covered from a configuration, it can also be covered from any configuration with more nodes, called *copycat property*. We show in Section 3.1 that both reconfigurable and loss-on-broadcast semantics satisfy this property. We then exploit this property to do a fine analysis on the number of nodes required to cover a particular state of the broadcast protocol $\mathcal{P}$. We modify the saturation algorithm of [DSTZ12] in a way that additionally keeps track of the size of an execution sufficient to cover any state in the set returned by the algorithm. The modified saturation algorithm is presented in Algorithm 2.

We present the main results of this chapter in Section 3.2. The upper bound results are constructive. We show by induction that for any state q in the set returned by Algorithm 2, one can construct a covering execution with linear number of nodes and of length quadratic in size of the protocol such that the final configuration has a node in q. Conversely, we show that the set returned by Algorithm 2 is exactly the set of coverable states: for reconfigurable semantics, the result follows from the correctness of saturation algorithm of [DSTZ12], and for loss-on-broadcast semantics, we show that in Lemma 3.11. We then prove matching lower bounds for the cutoff and covering length: we give a family of protocols that achieves those bounds. Although as a bonus of the upper bound results, we conclude that the set of states which can be covered in reconfigurable and loss-on-broadcast semantics is actually the same, yet we show in Section 3.3 that the reconfigurable semantics can be in some cases more succinct by a linear factor, in terms of number of nodes (see Theorem 3.14). Finally, we show that deciding whether, given some k, the cutoff is at most k is a NP-complete problem.

Application to other models of communication

We remark here that some of the results presented in this chapter apply to various other communication models. Let us mention some of them here. In [DT13], the authors consider asynchronous rendez-vous protocols, where at a certain instance, at most two entities communicate via message passing, and additionally, each process contains a *mailbox* where it stores the actions that are yet to be read. So, intuitively, the broadcasts of messages are non-blocking and when a broadcast happens, the message is stored in the mailboxes of the neighbours of the broadcasting node, which will eventually be read by the receiving nodes. The authors show that when the mailbox is described as a bag of actions, or a lossy FIFO channel, the coverability problem reduces to the same in reconfigurable semantics of the same protocol seen as a broadcast protocol. Moreover, for a protocol $\mathcal{P}$, from a covering execution in reconfigurable semantics when $\mathcal{P}$ is seen as a broadcast protocol, one can construct another execution of $\mathcal{P}$ in asynchronous rendez-vous semantics, such that the latter has the same number of nodes as the former. Therefore, in those cases, the upper bounds for cutoff for the coverability problem apply. More precisely, for positive instances of coverability in an asynchronous rendez-vous network with the mailbox described as a bag or a lossy FIFO channel, there is a covering execution with linear number of nodes.

Another example is broadcast protocols with intermittent nodes considered in [DSTZ12]. The authors show that the coverability problem in this semantics reduces to the same in reconfigurable semantics. That is, there exists a covering execution in one semantics if and only if there is one in the other with same number of nodes. Therefore, the upper bounds for cutoff for coverability problem applies. More precisely, for positive instances of coverability in broadcast networks with intermittent nodes, there is a covering execution with linear number of nodes.

This work opens roads towards various research directions. Let us discuss some of them here.

Tradeoff between cutoff and covering length

Consider the family of broadcast protocols $\mathcal{P}_n$ represented in Figure 3.10 with $2n{+}1$ states: $Q_n = \{q_0\} \cup \{q_i, r_i \mid 1 \leq i \leq n\}$ where we identify r_n with ☺, $\Sigma_n = \{\mathsf{a}\} \cup \{\mathsf{b}_i \mid 1 \leq i \leq n\}$, and let $F_n = \{☺\}$. The transition relation is defined by: $q_0 \xrightarrow{!\mathsf{a}} q_0$; for every $1 \leq i \leq n$, $q_{i-1} \xrightarrow{?\mathsf{a}} q_i$; $q_n \xrightarrow{!\mathsf{b}_i} q_0$; $q_n \xrightarrow{?\mathsf{b}_1} r_1$; and for every $2 \leq i \leq n$, $r_{i-1} \xrightarrow{?\mathsf{b}_i} r_i$.

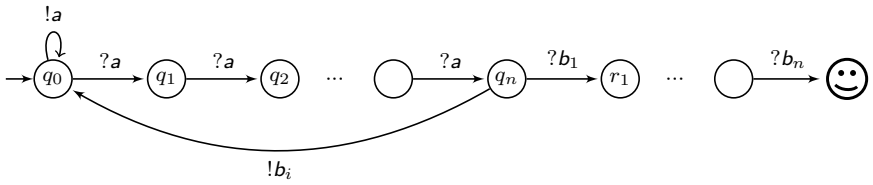


Figure 3.10: Tradeoff between cutoff and covering length: constant number of nodes need quadratic length, whereas linear number of nodes only require linear length.

Under static semantics (hence also in reconfigurable/lossy semantics), 3 nodes are sufficient to cover ☺, independently of the value of n. One node always performs the broadcasts of a, a second node performs all the broadcasts of b_i's and the third receives the b_i's to reach ☺. One can show 3 nodes are also necessary in any semantics to reach ☺. Indeed, a node, call it $\mathsf{n}_{☺}$, has to reach ☺ by receiving all b_i's, the node that broadcasts message a remains at q_0 and a final node has to traverse through the cycle of q_i's to perform $!\mathsf{b}_i$ for $\mathsf{n}_{☺}$ to successfully reach ☺. An example static execution with 3 nodes is given in Figure 3.11. However, with 3 nodes, the length of any covering execution is quadratic in n: the node that performs the broadcasts of b_i's needs to go n times through the cycle of q_i's of length n.

In contrast, with a linear number of nodes (precisely $n{+}2$), there exists a static (hence also in reconfigurable/lossy semantics) covering execution of length also linear in n (precisely $2n$). One node initially sends all others to q_n broadcasting a for n times, and then n successive broadcasts of b_1 to b_n are sufficient to cover ☺, see Figure 3.12. The precise interplay between number of nodes and length of covering execution is thus an interesting direction for future work.

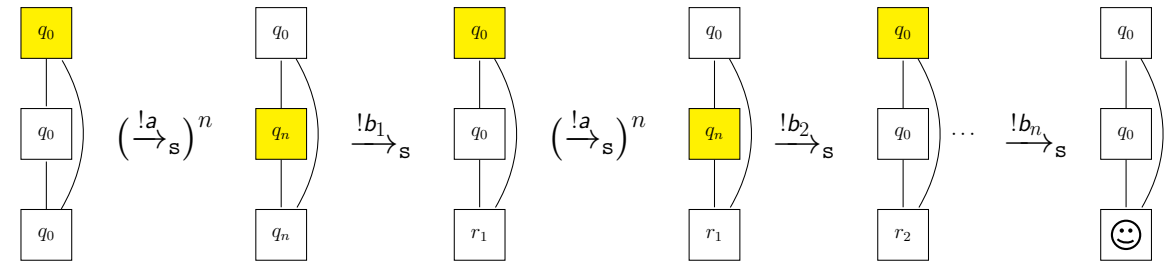


Figure 3.11: A static execution of size 3 and length $O(n^2)$ covering ☺ for the protocol of Figure 3.10.

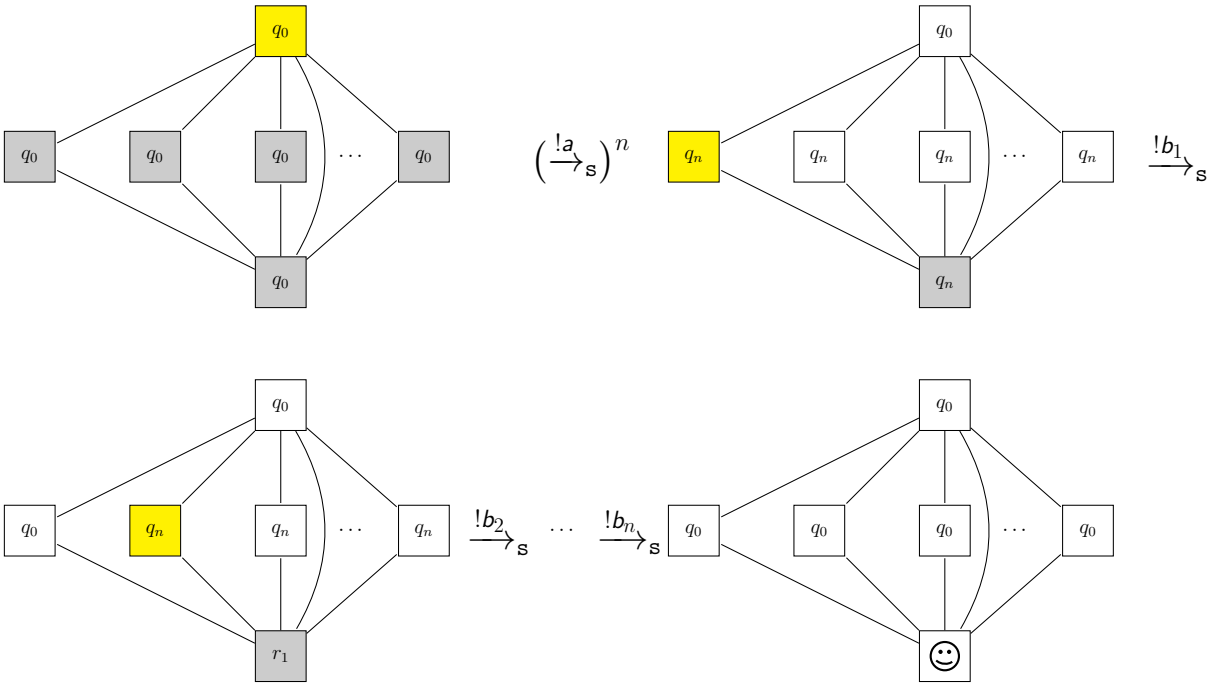


Figure 3.12: A static execution of size n+2 and length $2n$ covering ☺ for the protocol of Figure 3.10.

(In-)Approximability

In Section 3.4, the MinNodes problem was shown to be NP-complete. That is, given a broadcast protocol with a set of target states, we cannot design any algorithm which runs in polynomial time and returns a covering execution with minimal number of nodes. A natural question is thus can we design a polynomial time approximation algorithm for the problem? Recall that the hardness result was shown by a reduction from the SetCover problem, for which $O(\log n)$- factor approximation algorithms are known (see for instance the book [Vaz01]). This raises hope for existence of a similar factor approximation scheme for MinNodes problem as well. We leave the problem of finding an approximation scheme, if it exists, for MinNodes as future work.

Part II

Parameterized Concurrent Games

Chapter 4

Preliminaries

Games on Graphs. In computer science, *games on graphs* are a powerful but elegant framework to model interactions between entities (also called *players* or *agents* in game terminology). One of the motivations to study this model is to tackle the *reactive synthesis* problem. In a *reactive system*, a *controller* communicates continuously with an *environment*. For instance, one can think of an elevator (system) taking *inputs* from a person (environment), or a computer receiving inputs from a user. In such scenarios, the system has to react (and produce an *output* for an input given by the environment) in such a way that the input-output sequence satisfies some desired properties (or *specifications*). For instance, in the example of an elevator, one can think of a property that says "if it is requested to stop at a certain floor, it must eventually stop at that floor". These properties are usually specified in some logic. Note here that the actions of the environment are uncontrollable whereas the actions of the system are controllable, meaning we can program them. One can then be interested in various problems, for instance, given a specification, one can ask whether it is possible to design a system that could produce an output for every possible input from the environment such that the input-output sequence satisfies the specification; furthermore, for positive instances, automatically construct one. This is known as the *synthesis problem*. This was first formalized by Church in [Chu62] for a simple system with only one process. It turns out that one can reduce Church's synthesis problem into a game between *system* and *environment* on a finite graph such that a positive answer to the problem corresponds to the existence of a *winning strategy* for the system, see for instance the tutorial [Tho09].

In general, we consider a finite graph with a token moving along the edges. The token is controlled by the *players* (or *agents*). We often consider infinite games, *i.e.*, there are infinitely many rounds. The game therefore results in an infinite sequence of vertices called a *play*, and the way the players choose their moves are called *strategies*. One can consider different models of games, for instance, one where the players take moves in turn (*turn-based games*); or where they could play simultaneously (*concurrent games*). In this chapter, we recall the basic notions and some classical results on those models.

Two-player turn-based games. The model of two-player turn-based games on graphs is very much appreciated and well-studied in the literature. A turn-based game is played

between two players - Eve and Adam- on a graph. In most of the scenarios and, more specifically, in this thesis, we only consider games on finite graphs. A graph is composed of a (finite) set of *vertices* and an *edge* relation between them. Some vertices belong to Eve and the rest to Adam. The game starts at a designated initial vertex. At any step, the player who owns the vertex chooses an outgoing edge, the game proceeds to the successor vertex and the game continues from there. A *history* (resp., *play*) is any finite (resp., infinite) sequence of consecutive vertices.

A *strategy* for a player is a partial function that dictates the player's next move: it associates to every history ending at the player's vertex a unique successor vertex. A pair of strategies, one for each player, induces a play: the sequence of vertices selected by the strategies. We often consider the game to be of infinite duration, *i.e.*, the plays have infinite length. This is a natural assumption: for instance in the synthesis problem, the environment provides inputs forever and the system has to respond correspondingly, producing an infinite sequence of inputs and outputs.

We associate to the game a winning condition Win for Eve which is a (possibly infinite) set of plays. In *zero-sum games*, the winning condition for Adam is defined dually: a play is winning for Adam if it is not in Win. We say Eve wins the game if she has a strategy such that for any counter-strategy of Adam, the induced play belongs to Win. One of the central problems is then, given a game graph and a winning set of plays Win, to decide if Eve has a winning strategy.

Concurrent games. The model of turn-based games falls short in some scenarios, for instance in situations where the system and the environment choose their actions independently of each other and their combined actions determine the next state. In distributed systems, processes run concurrently, interact with each other and also with the environment. In such systems, it is typical that each component has private variables that are not visible to the other components. This concurrent behaviour of processes can be captured in the model of concurrent games.

In a two-player concurrent game, we are given a game graph with finitely many vertices and the edges are labelled with pairs of *actions*, one for each player. At a certain vertex, the players simultaneously choose their actions that are *enabled* at that vertex, and the pair of chosen actions then determines the next vertex. The model can be non-deterministic: from a vertex, a certain pair can lead to more than one successor in the game graph, then the game moves to one of the successor vertices non-deterministically. Note that turn-based games are particular types of concurrent games, where at each vertex, at most one player has enabled actions. A history (resp., play) is a finite (resp., infinite) sequence of vertices. A strategy for a player is a mapping that associates with a history an action for that player. Similarly to the turn-based games, we define winning conditions for the players as a (possibly infinite) set of plays, and we adapt the definition of winning strategies. Then in a zero-sum setting, the problem one can be interested in is, given a game graph and a winning condition Win, whether Eve has a winning strategy in the game.

Games with arbitrarily many players. In the above models, the number of players is fixed *a priori*. In fact, in these cases, there are exactly two-players. Although the concurrent game model can easily be extended to a multi-player setting where, instead of pairs of actions, the edges are labelled with tuples of actions, this model clearly cannot capture a scenario where the number of participants is not fixed *a priori*. For instance, a server may receive requests from an arbitrary number of clients or any number of devices may connect to a wireless ad hoc network.

In this part of the thesis, we extend the model of concurrent two-player games to a setting with an arbitrary number of players (or *agents*). The number of players is not fixed *a priori*, however, we do not allow agents to join or leave in the middle of the game, that is to say, when the game has started, the number of agents remains fixed throughout the play. More specifically, this number k, which is fixed at the beginning of the game, is initially unknown to the players, but they can gain some knowledge about k as the game proceeds. We call them *parameterized concurrent games*, or simply *parameterized games.*

In this setting, once again, we are given a game graph with finitely many vertices, but now the edges are labelled with languages of finite, yet *a priori* unbounded words to represent the choices of agents. Here we implicitly assume that the agents have identifiers which are known to them: the i-th letter of a word represents the action of the i-th player. For instance, the label a^+ represents that all players choose action a, similarly, $(ab)^+$ is the situation where each player with an odd identifier chooses a, and all other players (with an even identifier) play b. If from a vertex v, $(aa)^+$ leads to the unique successor v' in the arena, it represents that if there are an even number of players and each of them plays a at vertex v, then the game moves to v'. More precisely, in this part of the thesis, we consider the edge labellings are *regular languages* over a finite set of actions. The model we consider here is in general non-deterministic: from a vertex, a certain word can lead to more than one successor in the game graph, the non-determinism is resolved by an environment. A history is then a finite sequence of vertices, similarly a play is defined as an infinite sequence of vertices. A strategy for a player is a mapping that associates with a history an action for that player. Similarly to the previous games, we define a winning condition as an infinite set of plays, and we adapt the definition of winning strategies of a player: a strategy of a player applies with no prior information on the number of players, therefore it should be winning independently of the number of players; we will formally define them later.

We then consider two different scenarios, first, where a distinguished player, Eve, wants to achieve a goal irrespective of the number of players and their choices, and second, a *coalition game*, where all players together form a coalition and want to come up with a strategy to fulfil a common goal. In both settings, the winning condition is given by a (possibly infinite) set of plays. We are then interested in finding if Eve in the first setting or the coalition in the second has a winning strategy in the game.

Related work

The objectives considered in this part are *qualitative*: one can think of a payoff function with values 1 (in case a play is winning for a player) or 0 (otherwise). More generally, a *quantitative* objective assigns payoffs to players corresponding to each play, which need not be Boolean in general. A model of quantitative games is where the edges of a graph are labelled with rational payoffs for each player, and then the goal of a player is to maximize her reward, examples are *mean-payoff games* where Eve needs to maximize the limit average weight [EM79, ZP96, CDHR10], or *energy games* where Eve needs to keep the sum of weights positive [BJK10, CDHR10]. One can also combine qualitative and quantitative objectives: for example, Chatterjee *et al.* considers *energy parity games* in [CD10] while in [CRR12], the authors study the strategy synthesis problem for games with a multidimensional mean-payoff or energy objective along with a parity condition.

The turn-based games we consider in this part are *perfect information*, where each player has all information about the other player's actions and also about the current state of the game. In some scenarios, it is reasonable to relax this assumption: in an *imperfect information* model, some vertices may be indistinguishable to the system. While the concurrent setting introduces some level of imperfect information between the players, where they do not see each other's actions, one can also consider such restrictions in the turn-based setting, and ask if the system has a winning strategy in the game. For instance, [Rei84] studies the complexity of such games; a fixed-point algorithm for computing winning regions in such games with ω-regular objectives is given in [CDHR06]; in [BD08], the authors prove tight bounds for games with safety condition. One can also consider quantitative objectives, for example, energy and mean-payoff objectives in imperfect information games has been studied in [DDG$^+$10]. Note that the number of players in parameterized concurrent games is imperfect information to the players.

Non-zero-sum games with a fixed number of players, where each player has a different goal, has been studied in the literature. In such setting, a payoff function is defined that associates with any play a payoff to each player, and one asks whether there exists an *equilibrium*. An example is a *Nash equilibrium*, which is a set of strategies of the players that are rational to them in the sense that no one has an incentive to deviate from it [Nas50]. Chatterjee *et al.* in [CMJ04] show that for a turn-based two-player game with *ω-regular* objectives, a Nash equilibrium in *pure* strategies always exists. Further, the complexity of Nash equilibria in turn-based games were studied in [Umm08] where an equilibrium may need to additionally satisfy some qualitative constraints. Bouyer *et al.* in [BBMU11, BBMU15] extend the study to the concurrent game setting.

The models of games on graphs have also been extended to numerous other settings. To mention a few among them, Condon in [Con92] considers a *stochastic* model of graph games with three types of vertices, the vertices of the third type is a probabilistic choice vertex: the environment picks an outgoing edge from such vertex uniformly at random. That paper studies the model for a reachability condition; later in [CJH03], the authors consider stochastic games with a parity condition. In [MPS95], the authors consider *timed* games where the outcome of a game not only depends on the players' actions but also on

their timing; de Alfaro *et al.* in [dAFH+03] extend the model with an element of surprise where a player cannot anticipate when the opponent's action will occur at a certain round. Gutierrez *et al.* in [GKW19] consider a cooperative setting with a fixed number of agents in a concurrent game.

Our model of parameterized concurrent games mixes interactions and an arbitrary number of agents. As far as we are aware, only a couple of other works in parameterized verification have defined a game semantics, and they all largely differ from the current setting. First, to study broadcast networks of many identical Markov decision processes, broadcast networks of two-player games were introduced [BFS14]. There, the behaviour of each agent is the same and is described by a two-player turn-based game. Second, a control problem for an arbitrary size population of identical agents was studied in [BDG+19]. In that work, a controller plays against a parameterized number of agents, similarly to Eve playing against an unknown number of opponents. However, in contrast to our parameterized games, in the population control problem, the semantics is a turn-based game, and, most importantly, the arena is not centralized.

Organization of the chapter

We begin with a two-player turn-based setting of games on finite graphs in Section 4.1. We recall useful definitions and results from the literature. This is followed by a two-player concurrent games setting played on finite graphs in Section 4.2. We then extend this to a parameterized setting in Section 4.3 where the parameter is the number of players participating in the game. We define the decision problems on this model that we are interested in this part of the thesis. Finally, we close the chapter with a discussion on our contributions to this model in Section 4.4 that will be presented in the following chapters.

4.1 Two-player turn-based games

In this section, we recall the basic notions of a two-player turn-based game. The reader can refer to [GThW02, Chap. 2] for further reading. We, among many terminologies used in literature, will call the players Eve and Adam.

Definition 4.1. *A* game arena *is a tuple* $\mathcal{A} = \langle V, E \rangle$ *where*

- V *is a finite sets of vertices, which is a disjoint union of two sets:* $V = V_{\mathsf{E}} \uplus V_{\mathsf{A}}$*; and*
- $E \subseteq V \times V$ *is an* edge relation *on the set of vertices.*

The vertices in V_{E} are controlled by Eve and vertices in V_{A} are by Adam. In the graphical representation of an arena, vertices in V_{E} (resp., V_{A}) are represented by circles (resp., squares). We assume that the arena has no *dead-end*: for every $v \in V$, there exists

v' such that $(v, v') \in E$. This is a natural assumption since in practice, for instance, in reactive synthesis, one is particularly interested in infinite behaviours of the players. Figure 4.1 represents a game arena with $V_{\mathsf{E}} = \{v_0, v_3, v_4\}$ and $V_{\mathsf{A}} = \{v_1, v_2\}$.

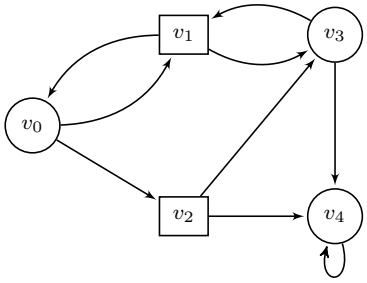


Figure 4.1: Example of a two-player turn-based game arena.

Intuitively, the game starting from a designated *initial vertex* v_0 proceeds as follows. One can think of a token moving along the arena. At the beginning, the token is at v_0. At a certain step, when the token is at v, the player who controls this vertex chooses a successor vertex v' and the token moves to v'. The game then proceeds similarly from v'. To formally capture this, one needs to define *histories*, *plays*, and *strategies* for the players.

Fix a game arena $\mathcal{A} = \langle V, E \rangle$. Then we define the following.

Definition 4.2. *A* history *is a finite sequence of vertices $h = v_0 v_1 \dots v_t$ such that for every $0 \leq i < t$, $(v_i, v_{i+1}) \in E$. A* play *is an infinite sequence of vertices $\rho = v_0 v_1 \dots$ such that for every $i \geq 0$, $(v_i, v_{i+1}) \in E$.*

We now define a *strategy* for a player in a game arena. Intuitively, a strategy dictates a player how to play the game. More precisely, yet informally, a strategy for a player is a mapping that assigns with any history ending at the player's vertex a successor vertex. It can be formally defined as follows.

Definition 4.3. *A* strategy *for* Eve *from a vertex v_0 in $\mathcal{A}$ is a partial function $\sigma : V^+ \to V$ that associates to every history $v_0 \dots v$, with $v \in V_{\mathsf{E}}$, a vertex $v' \in V$ such that $(v, v') \in E$. A play $\rho = v_0 v_1 \dots$ from v_0 is* induced by *(or,* compatible with *or,* follows*) strategy σ if for every $i \geq 0$, whenever $v_i \in V_{\mathsf{E}}$, we have $v_{i+1} = \sigma(v_0 \dots v_i)$.*

A strategy τ for Adam is defined analogously, for any history ending in a vertex of Adam. One can also similarly define an induced play by τ.

Given an arena $\mathcal{A}$ and a vertex v_0, a pair of strategies (σ, τ) from v_0 for Eve and Adam, respectively, defines a unique play $\rho = v_0 v_1 \dots$ such that for every $i \geq 0$, if $v_i \in V_{\mathsf{E}}$, then $v_{i+1} = \sigma(v_0 \dots v_i)$ and if $v_i \in V_{\mathsf{A}}$, then $v_{i+1} = \tau(v_0 \dots v_i)$. We call ρ the *outcome* of σ and τ from v_0 and denote it by $\mathsf{Out}_{\mathcal{A}}(v_0, \sigma, \tau)$.

We distinguish a special form of strategies, called *memoryless strategies*, which depend only on the current position of the token on the arena. This type of strategies does not need any memory of the history. Formally, it is defined as follows.

Definition 4.4. *A strategy σ for* Eve *(resp.,* Adam*) from v_0 is* memoryless *or* positional *if there exists a function $f : V_{\mathsf{E}} \to V$ (resp., $f : V_{\mathsf{A}} \to V$) such that for every history $v_0 \dots v$ with $v \in V_{\mathsf{E}}$ (resp., $v \in V_{\mathsf{A}}$), $\sigma(v_0 \dots v) = f(v)$.*

For memoryless strategies, we will not distinguish between σ and f.

Sometimes a strategy might need finite memory. We can represent a finite-memory strategy as an automaton with output where each state acts as a memory state, additionally equipped with an act function that defines the strategy corresponding to the current memory state and the current vertex and an upd function which updates the memory state when necessary.

Definition 4.5. *Let $\mathcal{A} = \langle V, E \rangle$ be a two-player turn-based arena and v_0 an initial vertex. A* Mealy automaton *(also called a* strategy automaton*) for* Eve *in $\mathcal{A}$ is a tuple $\mathcal{M} = (\mathtt{M}, V, m_0, \mathsf{act}, \mathsf{upd})$ where*

- *$\mathtt{M}$ is a finite set of* memory states*;*
- *$m_0 \in \mathtt{M}$ is the initial state;*
- *$\mathsf{upd} : \mathtt{M} \times V \to \mathtt{M}$ is the memory update function; and*
- *$\mathsf{act} : \mathtt{M} \times V_{\mathsf{E}} \to V$ is the transition choice function.*

A Mealy automaton $\mathcal{M}$ describes a strategy $\sigma_{\mathcal{M}}$ from v_0 as follows: first, for any history $h \in V^+$, inductively define $\mathtt{m}[h] \in \mathtt{M}$ by $\mathtt{m}[v_0] = m_0$, and $\mathtt{m}[h \cdot v] = \mathsf{upd}(\mathtt{m}[h], v)$; and then $\sigma_{\mathcal{M}}$ is defined as $\sigma_{\mathcal{M}}(h) = \mathsf{act}(\mathtt{m}[h], \mathsf{last}(h))$ whenever $\mathsf{last}(h) \in V_{\mathsf{E}}$, where $\mathsf{last}(h)$ is the last vertex of history h.

We say a strategy σ from v_0 for Eve *is* finite memory *if there exists a strategy automaton $\mathcal{M}$ such that $\sigma = \sigma_{\mathcal{M}}$.*

A strategy automaton for Adam can be defined analogously.

One can show that a strategy is memoryless if and only if it can be represented by a strategy automaton with a single memory state. Intuitively, since a memoryless strategy only depends on the current position of the game, the memory state is irrelevant, hence one can forever loop in a single memory state. For the other direction, if a strategy automaton consists of only one state, then the corresponding strategy cannot distinguish among the prefixes of a history inducing a memoryless one.

Lemma 4.6. *Given a turn-based arena $\mathcal{A}$ and an initial vertex v_0, a strategy σ for* Eve *in $\mathcal{A}$ is memoryless if and only if there exists a strategy automaton $\mathcal{M} = (\mathtt{M}, V, m_0, \mathsf{act}, \mathsf{upd})$ with a single memory state: $\mathtt{M} = \{m_0\}$ such that $\sigma = \sigma_{\mathcal{M}}$.*

We will now define when a play, more generally, a game is winning for a player. For that, we associate a winning criterion with a game arena that defines winning plays for the players. Later, we define some natural winning conditions that we will use in this part of the thesis. Here we only consider *zero-sum* games, where a play is winning for Eve if and only if it is not winning for Adam.

Definition 4.7. *A* winning condition *(or a* winning objective*) on an arena $\mathcal{A}$ is a subset* $\mathsf{Win} \subseteq V^\omega$ *of plays. A play ρ is winning for* Eve *if and only if* $\rho \in \mathsf{Win}$*, otherwise* Adam *wins the play. A strategy σ from v is* winning *for* Eve *if every play starting at v and induced by σ belongs to* Win*. In other words, σ is winning for* Eve *from v if for every counter-strategy τ of* Adam *from v, the outcome ρ of (σ, τ) belongs to* Win*. Finally,* Eve wins *from a vertex v if she has a winning strategy from v.*

An arena $\mathcal{A}$, together with a winning condition Win defines a *game*: $\mathcal{G} = (\mathcal{A}, \mathsf{Win})$. We denote the set of vertices that are winning for Eve by $\mathcal{W}_\mathsf{E}$ and the set of vertices which are winning for Adam by $\mathcal{W}_\mathsf{A}$. They are called the *winning region* for Eve and Adam, respectively.

Definition 4.8. *A game $\mathcal{G} = (\mathcal{A}, \mathsf{Win})$ with $\mathcal{A} = \langle V, E \rangle$ is* determined *if and only if from every vertex either* Eve *or* Adam *has a winning strategy. In other words, $\mathcal{G}$ is determined if and only if $V = \mathcal{W}_\mathsf{E} \uplus \mathcal{W}_\mathsf{A}$.*

Given a game $\mathcal{G} = (\mathcal{A}, \mathsf{Win})$, and an intial vertex v_0, one can ask whether Eve has a winning strategy from v_0. We can formalize this as follows.

Two-player turn-based game problem
Input: A game $\mathcal{G} = (\mathcal{A}, \mathsf{Win})$ and an initial vertex v_0.
Question: Does Eve have a winning strategy from v_0 in $\mathcal{G}$?

We recall here some of the winning conditions, which we will use in this part of this thesis. One can consider the objectives that a certain vertex is surely visited (*reachability*), or a certain vertex is always avoided (*safety*), or a vertex is visited infinitely often (*Büchi*). Notice that while it can be decided from a finite prefix of a play whether it satisfies a reachability condition, or violates a safety condition, for Büchi condition, that is not the case. Let us formally define these winning conditions.

Fix a two-player turn-based arena $\mathcal{A} = \langle V, E \rangle$. Then:

- Reachability: A *reachability condition* is specified by a subset F of V that describes the winning condition $\mathsf{Reach}(F) = \{v_0 v_1 \dots \mid \exists i \geq 0 : v_i \in F\}$. Intuitively, a play is winning if and only if it visits a vertex from F.
- Safety: A *safety condition* is specified by a subset F of V that describes the winning condition $\mathsf{Safe}(F) = \{v_0 v_1 \dots \mid \forall i \geq 0 : v_i \in F\}$. Intuitively, a play is winning if and only if it only visits vertices from F.

Notice that safety condition is the dual of reachability in the sense that given any finite subset F of V, $\mathsf{Safe}(F) = V^\omega \setminus \mathsf{Reach}(V \setminus F)$. Indeed, a play only visits vertices from the set F if and only if it never visits a vertex from $V \setminus F$.

- Büchi: A *Büchi condition* is specified by a subset F of V that describes the winning condition $\mathsf{Büchi}(F) = \{v_0 v_1 \dots \mid \exists^\omega i \geq 0 : v_i \in F\}$, where the notation $\exists^\omega i$ denotes 'there exists infinitely many i'. Intuitively, a play is winning if and only if it visits vertices from F infinitely often.

Let us illustrate the winning conditions on an example. Consider again the example in Figure 4.1. Assume the vertex v_0 is the initial vertex of the arena depicted with an incoming arrow to this vertex.

Example 4.9 (Reachability)**.** *Consider first a reachability winning condition for* Eve *with the set of target vertices* $F = \{v_4\}$*.* Eve *has a winning strategy in this case. A winning strategy* σ *is as follows:* $\sigma(v_0) = v_2; \sigma(v_3) = v_4$*;* $\sigma(v_4) = v_4$*. Indeed, no plays induced by* σ *ever visit vertex* v_1*, and* Adam *has two choices from vertex* v_2*, in both cases the game eventually reaches* v_4*. Notice that this is a memoryless strategy. In fact one can show that for reachability conditions, memoryless strategies are enough, in the sense that if a player has a strategy to win a reachability game then he or she has a memoryless one. One can further show that this is the unique memoryless winning strategy for* Eve *in this example. Indeed, if* Eve *chooses to always move to* v_1 *from* v_0*,* Adam *can force the game to stay within the loop between* v_0 *and* v_1*, that never visits vertex* v_4*.*

Example 4.10 (Safety)**.** *Consider the safety winning condition for* Eve *described by the set of 'safe' vertices* $F = V \setminus \{v_4\}$*.* Eve *has a winning strategy in this case. A winning strategy* σ *is as follows:* $\sigma(v_0) = v_1; \sigma(v_3) = v_1$*;* $\sigma(v_4) = v_4$*. Indeed, the game induced by* σ *never visits vertex* v_2*, and from* v_3*,* Eve *avoids vertex* v_4 *by moving to* v_1*; therefore, no induced play ever visits* v_4*. Notice that this is a memoryless strategy. In fact, one can show that for safety conditions as well, memoryless strategies are enough. One can also show this is the unique memoryless winning strategy for* Eve *in this example. Indeed, if* Eve *chooses to always move to* v_2 *from* v_0*,* Adam *can force the game to vertex* v_4*.*

Example 4.11 (Büchi)**.** *Let us now consider a Büchi winning condition for* Eve *on this arena described by the set* $F = \{v_1\}$*. Once again,* Eve *has a winning strategy to visit vertex* v_1 *infinitely often. A winning strategy* σ *is as follows:* $\sigma(v_0) = v_1; \sigma(v_3) = v_1$*;* $\sigma(v_4) = v_4$*. One can easily verify that all plays induced by* σ *visits* v_1 *infinitely often. Notice that this is again a memoryless strategy. In fact, one can show that also for Büchi conditions, memoryless strategies are enough. One can further show this is the unique memoryless winning strategy for* Eve *in this example. Indeed, if* Eve *chooses to always move to* v_2 *from* v_0*,* Adam *can simply move to* v_4 *and stay there forever, which is a losing play for* Eve*.*

Let us now recall some complexity results on the two-player turn-based game problem defined earlier for the aforementioned winning conditions, which we will use in the following chapters of the thesis.

Theorem 4.12. *For reachability (resp., safety) winning conditions, the two-player turn-based game problem can be solved in linear time in the size of the arena. More precisely, one can compute* memoryless *winning strategies for both players. Furthermore, these games are determined.*

Let us briefly sketch the idea for the proof of the result for reachability objective, which we will later use in proving our results. Given a set of target vertices F for Eve, we inductively construct the winning region $\mathcal{W}_{\mathsf{E}}$ as follows. We start with F and gradually append vertices to it. At i-th iteration, for a vertex $v \in V$, if $v \in V_{\mathsf{E}}$ and there is an outgoing edge to a vertex in $\mathcal{W}^i_{\mathsf{E}}$; or if $v \in V_{\mathsf{A}}$ and all the outgoing edges reach vertices in $\mathcal{W}^i_{\mathsf{E}}$, we add v to this set. Finally, we stop at saturation (since the arena is finite, it eventually saturates) and the set it returns is the winning region for Eve. One can then show that Eve indeed has a winning strategy from every vertex in $\mathcal{W}_{\mathsf{E}}$ and, further, Adam has a winning strategy from any vertex outside $\mathcal{W}_{\mathsf{E}}$. This construction is called *attractor computation*, and the sets calculated in this procedure are called *attractors*.

A similar result as Theorem 4.12 also holds for Büchi winning condition:

Theorem 4.13. *Büchi games are determined in polynomial time in the size of the arena, and one can effectively compute* memoryless *winning strategies for both players.*

In the next section, we recall basic definitions and important results for another model of games, called *concurrent games*, that differs from the model of turn-based games in a way that captures the notion of parallelism among the players.

4.2 Two-player concurrent games

In the previous section, we have seen a model of games on graphs where the players play in turn. That is, at a certain instant, when the token is at a certain vertex of the arena, exactly one player who owns that vertex chooses the next transition. Often this model of games falls short in describing parallelism between the players or concurrency in distributed systems. Consider a toy example. Suppose Eve and Adam decide to play the following game: each of them chooses a natural number between 1 and 10, and whoever chooses the larger number wins the game. Notice that in a turn-based setting, whoever first chooses the number has an unfair advantage in this game. Here concurrent game model comes to the rescue; it models the concurrency between the players. To model the above game, one can consider an arena where the edges are labelled with pairs of naturals between 1 and 10, and from the initial vertex, edges with labels (n_1, n_2) with $n_1 > n_2$ lead to a vertex that represents the winning vertex of player 1, for pairs (n_1, n_2) with $n_1 < n_2$ lead to a vertex that represents the winning vertex of player 2, and the rest of them lead to a neutral vertex.

Another example is the game of rock-paper-scissors, where two (or more) players simultaneously choose their moves. A game of rock-paper-scissors between two players is

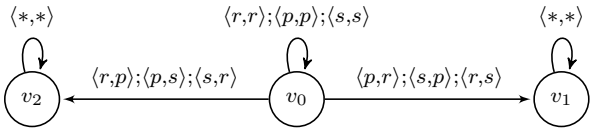


Figure 4.2: A game of rock-paper-scissors.

shown in Figure 4.2. Here r, p, s are the actions of the players representing rock, paper and scissor, respectively. Different components of an arena will be formally defined in the sequel. Intuitively, from v_0, if both players choose the same action, the game loops at v_0; for the choices of actions that are winning for player 1 (paper wins against rock, rock wins against scissor, and scissor wins against paper) lead the game to v_1, and the rest to v_2. In computer science, often in some concurrency models, the system and the environment choose their actions simultaneously and their combined actions determine the next transition. Other examples are distributed systems where multiple processes of a system cannot see each other's actions and therefore at a certain instance they must play actions simultaneously and independently of each other.

A concurrent game arena is again described by a finite graph, however, in contrast to the turn-based model, every vertex is of the same type, that is to say we do not distinguish vertices corresponding to the players, Eve and Adam. Additionally, we are given a finite set of actions Σ for each of the players. In the case of a two-player scenario, the edges of the graph are labelled with pairs of actions from Σ, for Eve and Adam, in that order. Notice that this formalism can easily be extended to a setting where there are more than two-players, in which case the edges would be labelled with a tuple of actions, one for each player. Although a deterministic model of concurrent games was initially considered for instance in [AHK98, AHK02], one can easily extend this to a non-deterministic framework, where from a vertex, a pair of actions may lead to different vertices in the graph; such a model was considered, for example, in [BBMU15]. One can also consider a probabilistic transition relation in this model that associates with a vertex and a pair of actions a probability distribution over vertices [dAHK07]. However, in this thesis, since we are interested in non-probabilistic models, in the rest of this section, we will consider the model of concurrent games equipped with non-deterministic transitions without probabilities.

Given an arena $\mathcal{A}$ and an initial vertex v_0, a concurrent game with two players proceeds as follows. Consider a token moving along the edges of the graph. Initially, the token is at v_0. At any instance when the token is at some v, each player independently chooses an action enabled for her/him at that vertex and the token non-deterministically moves to one of the successor vertices according to the transition relation. We can analogously define a history (resp., play) as a finite (resp., infinite) sequence of vertices and a (*pure* or *non-randomized*) strategy for each player as a mapping that associates with every finite history an action that is enabled at the last vertex of the history. Similarly to the turn-based model, we define winning objectives Win as the set of winning plays for Eve. While we mainly consider reachability and safety conditions for the problems we are interested in, some of our results will also apply for various other objectives. We say a

strategy of Eve is winning for her w.r.t. Win if for any counter-strategies of Adam, all induced plays are in Win. Finally, given an initial vertex and a winning objective, we say Eve wins a two-player concurrent game if she has a winning strategy for that objective.

We now formally define the model.

Definition 4.14. *A (two-player) concurrent game arena is a tuple* $\mathcal{A} = \langle V, \Sigma, \Gamma_{\mathsf{E}}, \Gamma_{\mathsf{A}}, \Delta \rangle$

- V *is a finite set of vertices;*
- Σ *is a finite set of actions for the players;*
- $\Gamma_{\mathsf{E}} : V \to 2^{\Sigma}$ *(resp.,* $\Gamma_{\mathsf{A}} : V \to 2^{\Sigma}$*) assigns with each vertex a set of actions that are* enabled *for* Eve *(resp.,* Adam*);*
- $\Delta : V \times \Sigma \times \Sigma \to 2^{V}$ *is the* transition function*, a mapping that associates with every vertex* $v \in V$ *and any two actions* $a \in \Gamma_{\mathsf{E}}$ *and* $b \in \Gamma_{\mathsf{A}}$*, a set of successor vertices.*

Let us give another example of a concurrent game arena.

Example 4.15. *Figure 4.3 represents a two-player concurrent game arena with the set of vertices* $V = \{v_0, v_1, v_2\}$*. The set of actions for the players is given by* $\Sigma = \{p_1, p_2, q_1, q_2\}$*. Without loss of generality, we assume that the first component of each pair represents the actions of* Eve *and the second component of* Adam*. Unlike Figure 4.2, in this example, we assume that the set of actions are disjoint for the players, more precisely, we assume that* p_i *(*$i \in \{1, 2\}$*) are never enabled for* Adam *and similarly* q_i *for* Eve*. The edge labels described in the picture define the transition relation. For instance, the tuple* (p_2, q_2) *from* v_0 *to* v_2 *represents that at vertex* v_0*, if* Eve *chooses* p_2 *and* Adam *chooses* q_2*, then the game moves to* v_2*. In this picture and throughout this part of the thesis, the label '*$*$*' will stand for any enabled action for the respective player. Note that there is a non-deterministic choice from* v_0 *when* Eve *chooses the action* p_1 *and* Adam *plays* q_1*. Notice also that action* q_1 *from* v_1 *is not enabled for* Adam*.*

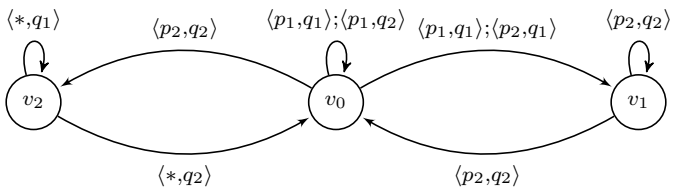


Figure 4.3: Example of a two-player concurrent game arena.

Remark that the definition of a concurrent arena for two players can naturally be extended for more than two players. More precisely, one can consider a set Agt of players and modify the transition function as follows: $\Delta : V \times \Sigma^{Agt} \to 2^{V}$ that maps a vertex and an action for each player to a set of vertices. However, in the rest of the section, we will consider the two-player setting and recall results for that setting.

Since we are interested in infinite behaviours of the players, like in the turn-based setting, we assume that a concurrent arena is complete for enabled actions. Formally, for every $v \in V$, and any two actions $a \in \Gamma_{\mathsf{E}}(v)$ and $b \in \Gamma_{\mathsf{A}}(v)$, we assume $\Delta(v, a, b) \neq \emptyset$.

A special case of a concurrent arena is a deterministic one. In a deterministic arena, from every vertex, for each pair of actions of the players, there is at most one successor vertex. Formally, an arena $\mathcal{A} = \langle V, \Sigma, \Delta \rangle$ is *deterministic* if for every $v \in V$, and any two actions $a \in \Gamma_{\mathsf{E}}(v)$ and $b \in \Gamma_{\mathsf{A}}(v)$, there is at most one vertex $v' \in V$ such that $v' \in \Delta(v, a, b)$. For example, the game arena in Figure 4.2 is a deterministic one.

The game on a concurrent arena from a designated initial vertex v_0 proceeds as follows. Think of a token moving along the arena, initially it is at v_0. At any instance, when the token is at v, each player independently chooses an action, say a and b respectively, enabled at v. Then a vertex v' from $\Delta(v, a, b)$ is selected non-deterministically and the game proceeds to v'.

In the following, we recall the definitions of a history, play and a strategy for a player from a vertex in a given concurrent arena analogous to the turn-based setting. While in this part of this thesis, we will only focus on non-randomized strategies, we will note some remarks on randomized strategies in the sequel.

Definition 4.16. *A* history *is a finite sequence of vertices $h = v_0 v_1 \dots v_t$ such that for every $0 \leq i < t$, there exist $a \in \Gamma_{\mathsf{E}}(v_i)$ and $b \in \Gamma_{\mathsf{A}}(v_i)$ such that $v_{i+1} \in \Delta(v_i, a, b)$. A* play *is an infinite sequence of vertices $\rho = v_0 v_1 \dots$ such that for every $i \geq 0$, there exist $a \in \Gamma_{\mathsf{E}}(v_i)$ and $b \in \Gamma_{\mathsf{A}}(v_i)$ such that $v_{i+1} \in \Delta(v_i, a, b)$.*

A strategy dictates a player how to move the token along the game. More precisely, a (*pure* or *non-randomized*) strategy for a player is a mapping that assigns to any history an action enabled for her/him at that vertex. It can be defined formally as follows.

Definition 4.17. *A* strategy *for* Eve *from a vertex v_0 in $\mathcal{A}$ is a function $\sigma : V^+ \to \Sigma$ that associates to every history $h = v_0 \dots v$ an action $a \in \Gamma_{\mathsf{E}}(v)$ enabled for her at v. We say a play $\rho = v_0 v_1 \dots$ from v_0 is* compatible with *or* induced by *strategy σ of* Eve *if for every $i \geq 0$, there exists $b_i \in \Gamma_{\mathsf{A}}(v_i)$ such that $v_{i+1} \in \Delta(v_i, \sigma(v_0 \dots v_i), b_i)$.*

Notice that a strategy can only assign actions that are enabled for the player at a particular vertex. A strategy for Adam can be defined similarly.

Given an arena $\mathcal{A}$ and an initial vertex v_0, a pair of strategies (σ, τ) from v_0 for Eve and Adam, respectively, defines *outcome*, denoted by $\mathsf{Out}_{\mathcal{A}}(v_0, \sigma, \tau) \subseteq V^\omega$, the set of plays compatible with σ and τ. Formally, a play $\rho = v_0 v_1 \dots$ is in $\mathsf{Out}_{\mathcal{A}}(v_0, \sigma, \tau)$ if for every $i \geq 0$, there exists $a_i \in \Gamma_{\mathsf{E}}(v_i)$ and $b_i \in \Gamma_{\mathsf{A}}(v_i)$ such that $a_i = \sigma(v_0 \dots v_i)$, $b_i = \tau(v_0 \dots v_i)$ and $v_{i+1} \in \Delta(v_i, a_i, b_i)$. Notice that, unlike turn-based setting, the outcome of a pair of strategies in the concurrent setting may not be unique. This is because a concurrent arena can be non-deterministic. For deterministic arenas, however, the outcome of a pair of strategies is unique.

A *memoryless* strategy only depends on the last vertex of a history. Formally, a strategy σ for a player from v_0 is *memoryless* if there exists a function $f : V \to \Sigma$ such that for every history $v_0 \dots v$, $\sigma(v_0 \dots v) = f(v)$. A Mealy automaton for finite memory strategies can be defined similarly as in the turn-based setting. However, in concurrent game framework, one needs to modify the act function in the definition of a Mealy automaton from Page 69 with $\mathsf{upd} : \mathtt{M} \times V \to \Sigma$ which associates to a memory state and a vertex an action in Γ_{E} (in case of Eve's strategy) or in Γ_{A} (in case of Adam's strategy). Given a strategy automaton $\mathcal{M}$, we can define the corresponding strategy described by it, $\sigma_{\mathcal{M}}$, similarly to the turn-based setting. We then say a strategy σ from v_0 for a player is *finite-memory* if there exists a strategy automaton $\mathcal{M}$ with $\sigma = \sigma_{\mathcal{M}}$.

We can define a winning objective for Eve on a *zero-sum* concurrent arena, analogous to the turn-based setting. Recall that in a zero-sum game a play is winning for Eve if and only if it is not winning for Adam.

Definition 4.18. *A* winning condition *on an arena $\mathcal{A}$ is a subset $\mathsf{Win} \subseteq V^\omega$ of plays. A play ρ is winning for* Eve *if and only if $\rho \in \mathsf{Win}$, otherwise* Adam *wins the play. A strategy σ from v is* winning *for* Eve *if every play starting at v and induced by σ belongs to* Win*. In other words, σ is winning for* Eve *from v if for every counter-strategy τ of* Adam *from v, $\mathsf{Out}_{\mathcal{A}}(v, \sigma, \tau) \subseteq \mathsf{Win}$. Finally,* Eve wins *from a vertex v if she has a winning strategy from v.*

An arena $\mathcal{A}$, together with a winning condition Win, defines a *game*: $\mathcal{G} = (\mathcal{A}, \mathsf{Win})$. We denote the set of vertices that are winning for Eve by $\mathcal{W}_{\mathsf{E}}$ and the set of vertices which are winning for Adam by $\mathcal{W}_{\mathsf{A}}$. They are called *winning region* for Eve and Adam, respectively. While in this part of the thesis we mostly focus on reachability and safety winning conditions, one can also consider other ω-regular conditions, for example a Büchi condition. We give an example of a concurrent game with a Büchi objective below.

Example 4.19. *Consider again the game arena from Figure 4.3. Let* Eve*'s objective here is to satisfy a Büchi condition described by the set of vertices $F = \{v_0, v_1\}$. One can show that* Eve *has a winning strategy σ from v_0 to satisfy the condition. Define σ as follows: $\sigma(hv_0) = p_1, \sigma(hv_1) = p_2, \sigma(hv_2) = p_1$ for any history $h \in V^*$. At v_0, on* Eve*'s action, the game either stays at v_0 or it might move to v_1 (because of the non-determinism in the arena on* Adam*'s action q_1); at v_1, on* Eve*'s action p_2, the game can either stay at v_1 or may go back to v_0. Therefore, there are only three types of induced plays: first, after finitely many rounds, the game stays at v_0; or after finitely many rounds, it stays at v_1; or both of them are visited infinitely often. In all cases, the Büchi condition described by set F is satisfied, and σ is indeed a winning strategy for* Eve.

Given a concurrent game $\mathcal{G} = (\mathcal{A}, \mathsf{Win})$, and an initial vertex v_0, we are interested in the problem of deciding whether Eve has a winning strategy in $\mathcal{G}$ from v_0. We can formalize the problem as follows.

Two-player concurrent game problem
Input: A concurrent game $\mathcal{G} = (\mathcal{A}, \mathsf{Win})$ and an initial vertex v_0.
Question: Does Eve have a winning strategy from v_0 in $\mathcal{G}$?

We recall here a result from [AHK98] for the above problem with reachability objectives.

Theorem 4.20 ([AHK98])**.** *For* reachability *objectives, the two-player concurrent game problem can be solved in linear time in the size of the arena. More precisely, one can compute* memoryless *(pure) winning strategies for* Eve*.*

The winning set of Eve in a concurrent reachability game can be computed in a similar way as in the turn-based setting. Given a set of target vertices F for Eve, we inductively construct the winning region $\mathcal{W}_{\mathsf{E}}$ as follows. We start with F and gradually append vertices to it. At i-th iteration, let $\mathcal{W}_{\mathsf{E}}^i$ be the set of vertices computed so far. Then for a vertex $v \in V$, if there exists an action a_i in $\Gamma_{\mathsf{E}}(v)$ such that for any action b_i in $\Gamma_{\mathsf{A}}(v)$, the vertices in $\Delta(v, a_i, b_i)$ belong to $\mathcal{W}_{\mathsf{E}}^i$, we add v to the set and call it $\mathcal{W}_{\mathsf{E}}^{i+1}$. Finally, we stop at saturation (since the arena is finite, the set eventually saturates) and call it $\mathcal{W}_{\mathsf{E}}$. We claim that $\mathcal{W}_{\mathsf{E}}$ is the winning region for Eve. One can then show that Eve indeed has a memoryless pure winning strategy from every vertex in $\mathcal{W}_{\mathsf{E}}$ and, further, Adam has a memoryless randomized spoiling strategy from any vertex outside $\mathcal{W}_{\mathsf{E}}$. Furthermore, the set $\mathcal{W}_{\mathsf{E}}$ can be computed in linear time on the size of the arena.

We here note a couple of remarks on the model of concurrent games. First, a turn-based game is a particular case of the concurrent ones. Indeed, consider a deterministic concurrent arena in which at every vertex, at most one player has more than one action enabled, then the arena is equivalent to a turn-based one. For example, if only Eve has more than one action enabled from a vertex in the concurrent arena, it can be recognized as a vertex of Eve in the corresponding turn-based arena; and similarly for Adam.

In this thesis, we only consider pure strategies of the players. However, one can also consider randomized strategies on a concurrent game arena, which maps every history to a probability distribution over actions enabled to the players at the last vertex of the history. In [AHK98], the authors consider such a definition of strategies and study various decision problems. They give an example of a concurrent game with a reachability condition such that Eve has no pure strategy that is winning against all strategies of Adam, but she has a randomized strategy that reaches F with probability 1 regardless of Adam's strategies.

In the following section, we extend the model of a two-player concurrent game to a parameterized setting, where the parameter is precisely the number of players participating in the game. We can then naturally extend the definitions of histories and plays. Finally, we introduce the two kinds of problems we are interested in that we will discuss in the next chapters.

4.3 Parameterized concurrent games

In the previous section, we have recalled the basic concepts of a (fixed) two-player concurrent game played on a finite graph. In such setting, players are aware of the fact that there are exactly two players throughout the game. We here extend this model to the

setting where the number of players is not fixed *a priori*. Generalizing concurrent games to a parameterized number of players can be done by replacing, on edges of the arena, tuples representing the choice of each of the players by languages of finite yet *a priori* unbounded words. Indeed, pairs of actions in the two-player setting can equivalently be represented by words of length 2. In this part of the thesis, we consider regular languages, given by NFAs, and represented by regular expressions in the examples. For instance, the label a^+ represents that all players choose action a, while ab^+ is the situation where the first player chooses a and all other players play b. Such a parameterized arena can represent infinitely many interaction situations, one for each possible number of agents: length k-words are for interactions among k players. We will further assume that the parameterized arenas are complete w.r.t. the actions of the players, that is from every vertex v, for any $k \in \mathbb{N}_{>0}$, and for every word $w \in \Sigma^k$ of length k, there is a successor v' of v with edge label L such that $w \in L$.

In parameterized concurrent games, the agents do not know *a priori* the number of agents participating in the interaction. Furthermore, we will restrict the model such that once the game has started, the number of players is fixed throughout the game. Yet, the difficulty lies in the fact that this number is not known to the players and can be unbounded *a priori*. The actions of selecting the number of players and resolving non-determinism in the game graph are performed by an adversarial environment. Each player can only observe the sequence of vertices played so far and the actions it has played. These pieces of information may refine the knowledge each player has on the number of involved players. To be precise, they infer knowledge on the possible number of players from a history and play the next move accordingly. Before going into formal definitions, let us first illustrate the model through examples.

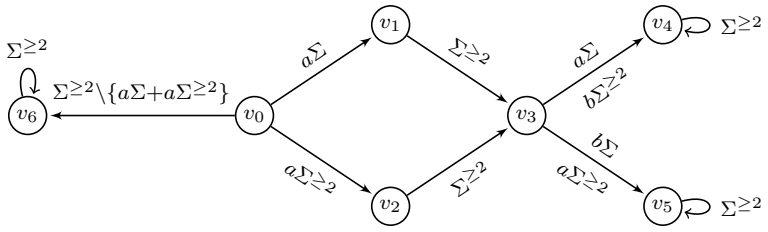


Figure 4.4: An example of a parameterized concurrent game.

Example 4.21. *Consider the parameterized arena depicted in Figure 4.4. Vertices are* $V = \{v_0, \ldots, v_6\}$*, and the set of actions is* $\Sigma = \{a, b\}$*. As such, the arena is not complete, we assume that all unspecified words from any vertex lead to vertex* v_6*. The arena is deterministic in this example, that is for every vertex* v *and every word* $w \in \Sigma^+$*, there is at most one outgoing edge from* v *with label* L *with* $w \in L$*. The regular language on the edge from* v_0 *to* v_1*, for example, represents that if there are exactly two-players and if* Eve *(not knowing* a priori *how many players are there, unless they inferred from the past history) plays action* a*, then for any action of the second player, the game moves from* v_0 *to* v_1*. At vertex* v_3*, if the first player* Eve *plays action* a*, then against one opponent it moves to* v_4*, and otherwise to* v_5*. Similarly, at* v_3*, if she plays* b*, then against one opponent the game goes to* v_5*, and otherwise to* v_4*.*

On the previous example, one can consider decision problems of existence of winning strategies of Eve against any number of opponents and their actions, for a given set of winning plays. Moreover, she does not know *a priori* how many opponents she has, therefore, she has to play uniformly, whatever the number of opponents she has. A strategy of a player is a function from histories to the set of actions. Given an objective for Eve, a strategy for her is winning w.r.t. objective Win, if for any number of opponents $k \in \mathbb{N}_{>0}$ and for every strategy of the opponents, all induced plays belong to Win. In this setting, Eve wants to achieve a particular objective, whereas her opponents want to prevent her from achieving her goal.

Let us illustrate how a game proceeds on a parameterized arena over an example. Consider the arena in Figure 4.4, and let v_0 be the initial vertex. The game from v_0 proceeds as follows: a positive integer k, the number of players, is first selected by an environment that is unknown to the players. At vertex v_0, all k players choose actions simultaneously, then this forms a k-length word: assuming Eve is the first player, if the i-th player plays action a_i, then the word formed is $w = a_1 \dots a_k$. Notice that we assume all players have identifiers, and they know their own identifiers; they are implicitly used to model the game. Then depending on whether the word w belongs to $a\Sigma$ (in case Eve plays action a and k is exactly 2) or $a\Sigma^{\geq 2}$ (in case Eve plays action a and k is more than 2), the game proceeds to vertex v_1 or v_2, respectively, and to v_6 otherwise. This process is repeated ad infinitum, generating an infinite play on the arena. In this example, and more generally, in the setting where Eve wants to achieve a goal against arbitrarily many opponents, we assume that there are at least one opponent of Eve in the game. Notice that in this example, the regular languages on the edges are particularly simple: they only constrain the number of opponents Eve has, that is, a move here only depends on Eve's action at a certain vertex and the number of her opponents.

Consider a reachability objective for Eve, on that example, described by the set of target vertices $F = \{v_4\}$. Let us informally argue here that Eve indeed has a winning strategy to reach v_4 in this example. For instance, at v_0, v_1 and v_2, she chooses action a, (notice that in this example her choice at v_1 and v_2 is irrelevant) and at v_3, depending on the history, she infers whether she has a single opponent or more than one opponent and plays accordingly. More precisely, if the history is $v_0v_1v_3$, she knows she has a single opponent in which case she plays action a at v_3, and if the history is $v_0v_2v_3$, she infers that the number of her opponents is more than one, and she plays b. In both cases, she ensures the game reaches to vertex v_4, irrespective of her number of opponents and their actions.

Let us give here another example of a game on a parameterized arena. This is an example of a coalition game where, given a game arena and an objective, every potential agent involved in the game applies a strategy such that collectively a global objective is satisfied. That is, the players have the same objective, and they must cooperate with each other and select respective strategies to win collectively against the environment. The collective strategies of the agents is called a *strategy profile*. Then a natural problem one can be interested in is to decide if there exists a strategy profile that is winning for the coalition, and if it exists, synthesize it.

In some situations, it can be more effective for the players to form a coalition. Let us first give an example of the coalition setting with a fixed number of players. Suppose 3 individuals are planning a tour to the same place. If they book their mode of transportation individually, it costs 80 euros each. However, if 2 of them (form a coalition and) share it, it costs 50 euros for each in the coalition, and for a coalition of 3, it costs 40 euros each. Clearly, it is the most effective for all of them to make a coalition in this situation. The notion of coalition games appears in literature, see for instance [OR94, Part IV]. One can also refer to [Pri14, Chap. 35] for more involved, but interesting examples of coalition games. Coalition games are also called *cooperative games* in the literature. In the following, we consider coalition games on parameterized arenas with a qualitative objective for the coalition.

We study a parameterized extension of coalition games where the parameter is again the number of players participating in the game. Given a parameterized arena informally defined in the beginning of the section and a global objective, the problem consists in synthesizing for every potential agent involved in the game, a strategy that she should apply, so that, collectively, a global objective is satisfied. Informally, a strategy of individual i assigns with every history an action a_i; then a coalition strategy is an infinite tuple of individual strategies of the players. We will later argue that this definition is equivalent to the one that maps each history to an ω-word in Σ^ω of which the i-th letter represents the action for agent with identifier i. Given a winning objective Win for the coalition, a coalition winning strategy must satisfy that for every $k \in \mathbb{N}_{>0}$, all induced plays with k players satisfy the objective. Let us illustrate the coalition setting on an example.

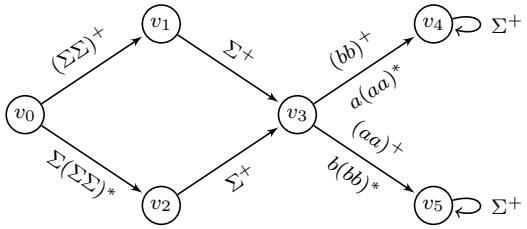


Figure 4.5: Example of a coalition game.

Example 4.22. *Consider the parameterized arena depicted in Figure 4.5. Vertices are $V = \{v_0, \ldots, v_5\}$, and the set of actions is $\Sigma = \{a, b\}$. As such the arena is not complete, we assume that all unspecified words from every vertex lead to the sink vertex v_5. The arena is deterministic in this example, that is for every vertex v and every word $w \in \Sigma^+$, there is at most one outgoing edge from v with label L with $w \in L$. The regular language on the edge from v_0 to v_1, for example, represents that if there are an even number of players, then for any action by each of them moves the game from v_0 to v_1. At vertex v_3, if everyone chooses a, then with even (resp., odd) number of players, the game moves to v_5 (resp., v_4); similarly if all of them choose b, then with even (resp., odd) number of players, the game moves to v_4 (resp., v_5).*

Consider a safety objective for the coalition, on that example, described by the set of target vertices $F = V \setminus \{v_5\}$. Let us informally argue here that the coalition indeed

has a winning strategy to avoid v_5 in this example. For instance, at v_0, v_1 and v_2, the coalition can play a^ω (they can in fact choose any word in Σ^ω), and at v_3, depending on the history, the coalition infers whether the number of players is even (resp., odd), in that case the coalition plays b^ω (resp., a^ω), and reaches v_4. More precisely, if the history is $v_0v_1v_3$, the coalition infers the number of players is even, in which case they play b^ω, and if the history is $v_0v_2v_3$, they infer that the number of players is odd, in which case they play a^ω. In both cases, it ensures the game avoids vertex v_5, irrespective of the number of players at the start of the game. Notice that the choice of the coalition at v_1 and v_2 is irrelevant.

We now formally define a parameterized arena and formalize the problems we will consider in this part of the thesis on such arenas.

Definition 4.23. *A* parameterized arena *is a tuple* $\mathcal{A} = \langle V, \Sigma, \Delta \rangle$ *where*

- V *is a finite set of vertices;*
- Σ *is a finite set of actions;*
- $\Delta : V \times V \to 2^{\Sigma^+}$ *is a transition function.*

In this work, we assume that for every $(v, v') \in V \times V$, $\Delta(v, v')$, *when non-empty, describes a regular language given by a non-deterministic finite automaton (NFA).*

Definition 4.24. *An* NFA *is a tuple* $\mathcal{N} = (\mathsf{Q}_\mathcal{N}, \Sigma_\mathcal{N}, \Delta_\mathcal{N}, \mathsf{q}_{\mathsf{init}}, \mathsf{F}_\mathcal{N})$ *where* $\mathsf{Q}_\mathcal{N}$ *is a finite set of states;* $\Sigma_\mathcal{N}$ *is a finite alphabet;* $\Delta_\mathcal{N} : \mathsf{Q}_\mathcal{N} \times \Sigma_\mathcal{N} \to 2^{\mathsf{Q}_\mathcal{N}}$ *is a transition function;* $\mathsf{q}_{\mathsf{init}}$ *is an initial state in* $\mathsf{Q}_\mathcal{N}$*; and finally,* $\mathsf{F}_\mathcal{N} \subseteq \mathsf{Q}_\mathcal{N}$ *is a set of accepting states.*

A word $\mathsf{w} = \mathsf{a}_1 \dots \mathsf{a}_\mathsf{n}$ over $\Sigma_\mathcal{N}$ is *accepted* by $\mathcal{N}$ if there exists a *run* $\mathsf{R} = \mathsf{q}_0\mathsf{q}_1 \dots \mathsf{q}_\mathsf{n}$ of $\mathcal{N}$ on w such that $\mathsf{q}_0 = \mathsf{q}_{\mathsf{init}}$ and $\mathsf{q}_\mathsf{n} \in \mathsf{F}_\mathcal{N}$.

Fix a parameterized arena $\mathcal{A} = \langle V, \Sigma, \Delta \rangle$. We assume that there is at least one player participating in the game, this is ensured in the definition of the transition function. The arena $\mathcal{A}$ is *deterministic* if for every $v \in V$, and every word $u \in \Sigma^+$, there is at most one vertex $v' \in V$ such that $u \in \Delta(v, v')$. The arena is assumed to be *complete*: for every $v \in V$ and $u \in \Sigma^+$, there exists $v' \in V$ such that $u \in \Delta(v, v')$. This assumption is natural: such an arena will be used to play games with an arbitrary number of agents, hence for the game to be non-blocking, successor vertices should exist whatever that number is and irrespective of the choices of actions. For conciseness, examples (for instance, the one in Figure 4.6) might depict incomplete arenas, however, a sink vertex can be added, and all unspecified moves can lead to that vertex to obtain a complete one.

Examples of deterministic arenas were presented in Figure 4.4 and 4.5. In the following, let us give an example of a non-deterministic one. We illustrate different components of the arena on this example.

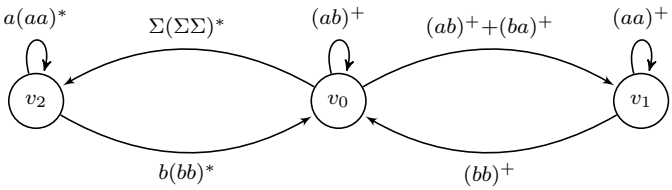


Figure 4.6: Example of a non-deterministic parameterized arena.

Example 4.25. *Figure 4.6 presents a non-deterministic parameterized arena. The vertices are* $V = \{v_0, v_1, v_2, \top\}$ *and the alphabet is* $\Sigma = \{a, b\}$*;* $\top$ *is a sink vertex which is not depicted here, all unspecified transitions from any vertex go to* $\top$ *and loop forever. The edge labels define the transition function, for instance, the language that labels the edge from* v_2 *to* v_0 *is* $b(bb)^*$*. That is to say, at vertex* v_2*, if there is an odd number of players and all of them play* b *then the game will move to* v_0*. Notice that at* v_0*, there is a non-deterministic transition on any word in the language* $(ab)^+$*, that is to say, at vertex* v_0*, if there is an even number of players and if they choose actions* a *and* b *alternatively, starting with an* a *(*i.e.*, the first player chooses action* a*), then the game can either stay at* v_0 *or move to* v_1*; the non-determinism is resolved by an adversarial environment.*

Notice that in parameterized concurrent games, we assume each player has a distinct identifier, yet they only know their own identifiers. However, players do not communicate; their identifiers are only used to select the vertices the game proceeds to. Moreover, the agents do not know *a priori* the number of agents participating in the interaction. Each agent observes the action it plays and the vertices the play goes through. These pieces of information may refine the knowledge each agent has on the number of involved agents.

Fix a parameterized arena $\mathcal{A} = \langle V, \Sigma, \Delta \rangle$ and an initial vertex v_0. A game on $\mathcal{A}$ proceeds as follows. At the starting of the game, environment chooses k the number of players that remains unknown to them. At v_0, player i chooses action a_i (all players choose their actions simultaneously); this forms a k-length word $w = a_1 \dots a_k$. Then the game moves to v_1 such that $w \in \Delta(v_0, v_1)$. If there exists more than one vertex satisfying the above, one of them is chosen non-deterministically, and the non-determinism is resolved by the environment. The game then proceeds from v_1, with the same k. This process is repeated ad infinitum, generating an infinite play in the arena.

We now define histories, plays and strategies of the players. Histories and plays are defined similarly to the two-player concurrent game setting. More precisely, a history is a finite sequence of vertices satisfying the transition function, and similarly a play is defined as an infinite sequence of vertices.

Fix a parameterized arena $\mathcal{A} = \langle V, \Sigma, \Delta \rangle$.

Definition 4.26. *A* history *in* $\mathcal{A}$ *is a finite sequence of vertices, that is compatible with the edges: formally,* $h = v_0 v_1 \dots v_p \in V^+$ *such that for every* $0 \leq j < p$*,* $\Delta(v_j, v_{j+1}) \neq \emptyset$*. We write* $\mathsf{Hist}_{\mathcal{A}}$ *for the set of all histories. A* play *is an infinite sequence of vertices compatible with the edges:* $\rho = v_0 v_1 \dots \in V^\omega$ *such that for every* $j \geq 0$*,* $\Delta(v_j, v_{j+1}) \neq \emptyset$*.*

The set of plays is denoted $\mathsf{Play}_{\mathcal{A}}$.

The above definition does not say anything about the number of agents forming a history, or a play. It is possible that there is no such k corresponding to a history (or, a play). This motivates us to define the following notion of *realizability*. Intuitively, a history (resp., a play) is realizable if there is some number of agents k that is compatible with the history (resp., play).

Definition 4.27. *For $k \in \mathbb{N}_{>0}$, a history $h = v_0 \cdots v_p$ is* k-realizable *if it corresponds to a history for k agents,* i.e., *if for all $0 \leq j < p$, there exists $u \in \Sigma^k$ with $u \in \Delta(v_j, v_{j+1})$. A history is* realizable *if it is k-realizable for some $k \in \mathbb{N}_{>0}$. Similarly to histories for finite sequences of consecutive vertices, one can define the notions of* (k-)realizable plays *for infinite sequences.*

Let us now define strategies for the players. A strategy dictates a player potentially involved in the game which action to choose from a vertex depending on the past history. A *strategy profile* is a tuple of strategies, one for each agent. Since the number of agents is not fixed *a priori*, a strategy profile is an infinite tuple of strategies. We will implicitly assume that each agent potentially involved in the game has a unique identifier that will be used to select the vertices the game proceeds to. A strategy for an agent and a strategy profile is formally defined below.

Definition 4.28. *A* strategy *for agent i from vertex v_0 is a mapping $\sigma_i : V^+ \to \Sigma$ that associates to every history $h = v_0 v_1 \dots v_p$ in $\mathsf{Hist}_{\mathcal{A}}$ an action a in Σ. A* strategy profile *is an infinite tuple of strategies: $\widetilde{\sigma} = \langle \sigma_1, \sigma_2, \dots \rangle \in (V^+ \to \Sigma)^\omega$.*

A strategy profile $\widetilde{\sigma} = \langle \sigma_1, \sigma_2, \dots \rangle$ from vertex v_0 induces a set of plays, called *outcome*. First, for any natural number k, we define the *k-outcome* $\mathsf{Out}^k_{\mathcal{A}}(v_0, \sigma_1, \dots, \sigma_k)$ as the set of plays induced by $\sigma_1, \dots, \sigma_k$ that are k-realizable; and then, the *outcome* $\mathsf{Out}_{\mathcal{A}}(v_0, \widetilde{\sigma})$ of the strategy profile $\widetilde{\sigma}$ is simply the union of all k-outcomes. Note that the completeness assumption in parameterized arenas ensures that the set $\mathsf{Out}^k_{\mathcal{A}}(v, \sigma_1, \dots, \sigma_k)$ is non-empty, and hence $\mathsf{Out}_{\mathcal{A}}(v_0, \widetilde{\sigma})$ is also not empty. They are formally defined below.

Definition 4.29. *Given a strategy profile $\widetilde{\sigma} = \langle \sigma_1, \sigma_2, \dots \rangle$, an initial vertex v_0 and a number of agents $k \in \mathbb{N}_{>0}$, we define the* k-outcome $\mathsf{Out}^k_{\mathcal{A}}(v_0, \sigma_1, \dots, \sigma_k) = \{v_0 v_1 \cdots \mid \forall j \geq 0,\ \sigma_1(v_0 \cdots v_j)\sigma_2(v_0 \cdots v_j) \dots \sigma_k(v_0 \cdots v_j) \in \Delta(v_j, v_{j+1})\}$. *Then the* outcome *of strategy profile $\widetilde{\sigma}$ is* $\mathsf{Out}_{\mathcal{A}}(v_0, \widetilde{\sigma}) = \bigcup_{k \in \mathbb{N}_{>0}} \mathsf{Out}^k_{\mathcal{A}}(v_0, \sigma_1, \dots, \sigma_k)$.

We have informally described the notion of strategy profiles on the examples of Figure 4.4 and 4.5. We now illustrate these notions on the example of Figure 4.6.

Example 4.30. *Consider again the example in Figure 4.6. Let the initial vertex be v_0. Consider the strategy σ_i for agent i defined as follows: for every $h \in V^*$, $\sigma_i(hv_0) = a$, if i is odd, $\sigma_i(hv_0) = b$, if i is even; and $\sigma_i(hv_1) = \sigma_i(hv_2) = a$ for every $i \geq 1$. This defines a strategy profile $\widetilde{\sigma} = \langle \sigma_1, \sigma_2, \dots \rangle$. Recall that the initial choice of the number*

of players k and the resolution of non-determinism during the game is performed by an adversarial environment. Examples of plays induced by $\widetilde{\sigma}$ are v_0^ω if k is even and adversary chooses to stay at v_0 forever; $v_0v_0v_1^\omega$ if k is even and adversary chooses to loop once at v_0 and then move to v_1; $v_0v_2^\omega$ if k is odd, etc. When k is even, the k-outcome is the set $\{v_0^\omega\} \cup \{v_0^j v_1^\omega \mid j \geq 1\}$, whereas for any odd k, the k-outcome is the singleton $\{v_0v_2^\omega\}$. Therefore, the outcome of $\widetilde{\sigma}$ is the set $\{v_0^\omega, v_0v_2^\omega\} \cup \{v_0^j v_1^\omega \mid j \geq 1\}$.

We can define a winning objective as a set of plays and a winning condition then naturally defines a game.

Definition 4.31. *A winning objective is a set of plays $\mathsf{Win} \subseteq V^\omega$. Given a parameterized arena $\mathcal{A}$, a winning objective* Win *defines a* parameterized game $\mathcal{G} = (\mathcal{A}, \mathsf{Win})$.

As for traditional concurrent games, one can consider natural questions such as, for instance, the distributed synthesis problem informally explained on the example of Figure 4.4, or the existence and computation of Nash equilibria etc. In this part of the thesis, we consider two of them. First, Player 1 (called Eve in the sequel) is distinguished, and she aims at achieving an objective independently of the number of opponents she has and of their strategies. Second, the players collectively form a coalition and try to come up with a strategy profile to achieve a common goal, not knowing *a priori* how many they are. Let us now describe the problems in more details.

Eve against unknown number of opponents

We first consider a setting where the first player, called Eve, is distinguished, and she wants to achieve a given objective against the coalition of the other players, not knowing *a priori* the number of her opponents. She therefore must play uniformly, whatever the number of opponents she has. Given a winning objective Win, it defines a parameterized game $\mathcal{G} = (\mathcal{A}, \mathsf{Win})$ for Eve. We then study the problem of existence of a winning strategy for Eve. Chapter 5 is dedicated to the resolution of this problem and to the study of complexity bounds for the same for the reachability objectives. We give here a brief description of the setting along with an example.

We assume in this setting, there are at least two players in the game, *i.e.*, there is at least one opponent of Eve. Given a parameterized arena $\mathcal{A}$ and an initial vertex v_0, we define the outcome $\mathsf{Out}_{\mathcal{A}}(v_0, \sigma_1)$ of a strategy σ_1 of Eve as the set of plays $\bigcup_{k \geq 2} \bigcup_{\sigma_2, \ldots, \sigma_k} \mathsf{Out}^k_{\mathcal{A}}(v_0, \sigma_1, \ldots, \sigma_k)$, where $\mathsf{Out}^k_{\mathcal{A}}(v_0, \sigma_1, \ldots, \sigma_k)$ is the k-outcome induced by the strategies of k players. A strategy σ_1 of Eve from v_0 is winning w.r.t. Win for the coalition, if all plays in the outcome of σ_1 belongs to Win. We then define the following decision problem:

Eve AGAINST UNKNOWN NUMBER OF OPPONENTS
Input: A parameterized game $\mathcal{G} = (\mathcal{A}, \mathsf{Win})$ and an initial vertex v_0.
Question: Yes if and only if $\exists \sigma_1$ for Eve such that $\mathsf{Out}_{\mathcal{A}}(v_0, \sigma_1) \subseteq \mathsf{Win}$.

We have earlier illustrated the coalition game on a parameterized arena for a reachability objective on the example of Figure 4.4. We have informally argued that in that example, Eve indeed has a winning strategy from v_0 for the reachability objective described by the set $F = \{v_4\}$. Let us now illustrate this setting on another example with a Büchi objective.

Example 4.32. *Consider a Büchi objective for* Eve *on the example of Figure 4.6 described by the set* $F = \{v_0, \top\}$. *We can show that* Eve *has a winning strategy from* v_0 *for the above objective. Consider the strategy* σ_1 *of* Eve *defined as follows:* $\sigma_1(v_0) = \sigma_1(hv_0) = a; \sigma_1(hv_1) = \sigma_1(hv_2) = b$, *for any (non-empty) history* h *in* V^+. *We show that* σ_1 *is indeed winning for* Eve. *At* v_0, Eve *plays an* a. *Then*

- *either it goes to* $\top$, *which is immediately winning for* Eve;
- *or, it loops at* v_0;
- *or, for an odd number of opponents, it moves to* v_2. *At* v_2, *for any prefix,* Eve *chooses* b *(since* Eve *infers that the number of players* k *is odd). Then the game either goes back to* v_0 *(if all her opponents play* b*), or to* $\top$, *which is winning for her;*
- *or, it goes to* v_1, *where again* Eve *infers that the number of players* k *is even, and hence she plays* b. *Similarly to the previous case, if all her opponents also play* b, *the game goes back to* v_0, *otherwise it moves to* $\top$, *which is winning for her;*

At every visit of v_0, Eve *can play the same action, and one of the above holds. Thus, the game either eventually moves to* $\top$ *and stays there forever, or it visits* v_0 *infinitely often. In both cases,* Eve *wins the Büchi game. Notice that since* $\top$ *is winning for* Eve, *in this example the choices of actions for her opponents are quite restricted (in the sense that they would try to avoid* $\top$ *as long as possible). Also remark that, at* v_0, Eve *can also play action* b, *yet the game would have the same effect. Finally, the set of realizable plays is restricted: after* v_0 *repeats for the first time, since* k *is fixed throughout the game, either* v_1 *or* v_2 *is visited infinitely often, but not both (or, neither of them, in case the game eventually moves to* $\top$*).*

Second, we introduce another game setting on a parameterized arena, namely *coalition games*. In this setting, the agents play actions so that they collectively achieve a common goal, irrespective of the number of them participating in the game. We describe the problem in more details.

Coalition games

We consider a *coalition game* setting where the agents play as a coalition to achieve a common goal. At each round, depending on the history and their knowledge on the

number of agents, which is unknown to them at the beginning of the game, each agent chooses an action. Given a set of winning plays Win, it defines a parameterized game $\mathcal{G} = (\mathcal{A}, \mathsf{Win})$ for the coalition. The goal for the coalition is to ensure Win for any number of agents involved in the game. We then study the decision problem of whether there exists a strategy profile that is winning for the coalition. Chapter 6 is dedicated to the resolution of this problem and studies the complexity when Win is a safety objective. We give here a brief description of the setting along with an example.

Given a parameterized arena, one can observe that in the coalition setting, the existence of a strategy profile of agents is equivalent to the existence of a *coalition strategy*, described by a function $\boldsymbol{\sigma} : V^+ \to \Sigma^\omega$, that maps each history to an ω-word over Σ of which the k-th letter represents the action for agent k. Given an initial vertex, we similarly define the (k-)outcome of a coalition strategy $\boldsymbol{\sigma}$ in this setting. A coalition strategy $\boldsymbol{\sigma}$ from v_0 is winning w.r.t. Win for the coalition, if all plays in the outcome of $\boldsymbol{\sigma}$ belongs to Win. We then define the following decision problem:

Coalition problem
Input: A parameterized game $\mathcal{G} = (\mathcal{A}, \mathsf{Win})$ and an initial vertex v_0.
Question: Yes if and only if $\exists \boldsymbol{\sigma}$ for the coalition such that $\mathsf{Out}_{\mathcal{A}}(v_0, \boldsymbol{\sigma}) \subseteq \mathsf{Win}$.

We have earlier illustrated the coalition game on a parameterized arena for a safety objective on the example of Figure 4.5. We have informally argued that in that example, the coalition indeed has a winning strategy from v_0 for the safety objective described by the set $F = V \setminus \{v_5\}$. Let us now illustrate the setting on an example with a reachability objective.

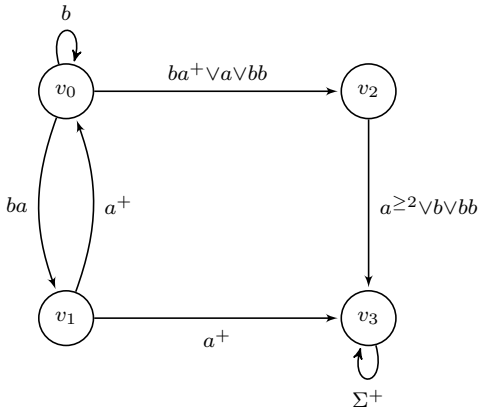


Figure 4.7: Example of a reachability coalition game.

Example 4.33. *Consider the parameterized arena depicted in Figure 4.7. Vertices are* $V = \{v_0, v_1, v_2, v_3, \bot\}$*, actions of the players are* $\Sigma = \{a, b\}$*. The arena is as such not complete, to obtain a complete arena, all unspecified words from each vertex go to a sink vertex* $\bot$*, which is not depicted here. The labels on the edges in the figure define the transition function. Consider a* reachability *objective described by* $F = \{v_3\}$ *for the coalition. We can show that the coalition has a winning strategy from* v_0 *for the above objective.*

We will describe the coalition strategies as functions $V^+ \to \Sigma^\omega$. Consider the coalition strategy $\boldsymbol{\sigma}$ defined as follows: $\boldsymbol{\sigma}(v_0) = ba^\omega$; for any (non-empty) history $h \in V^+$: $\boldsymbol{\sigma}(hv_1) = \boldsymbol{\sigma}(v_0v_0) = a^\omega$; $\boldsymbol{\sigma}(v_0v_1v_0) = b^\omega$; $\boldsymbol{\sigma}(v_0v_2) = a^\omega$; $\boldsymbol{\sigma}(v_0v_0v_2) = \boldsymbol{\sigma}(v_0v_1v_0v_2) = b^\omega$; and arbitrary otherwise. We show that $\boldsymbol{\sigma}$ is indeed winning for the coalition. At v_0, the coalition first plays ba^ω. Then:

- *either it loops at v_0. In that case the coalition infers $k = 1$ and plays a^ω, hence the game reaches vertex v_2;*
- *or, the game goes to v_1 and the coalition infers $k = 2$, and plays a^ω from v_1. Either it reaches v_3, in which case the coalition wins, or it goes back to v_0. At v_0, the coalition plays b^ω (since according to the history, they infer $k = 2$) and reaches v_2;*
- *or, the game directly goes to v_2 after the first choice of actions of the players. Then the coalition infers $k \geq 2$.*

At v_2, depending on history, they either play a^ω (when the common knowledge is $k \geq 2$), or b^ω (when the common knowledge is $k \in \{1, 2\}$). In both cases, the game reaches v_3 and the coalition wins. We conclude $\boldsymbol{\sigma}$ is a winning strategy for the coalition.

Notice that $\boldsymbol{\sigma}$ uses memory. Indeed, at v_0, if the coalition always plays ba^ω, it is losing against $k = 1$, similarly for other words. Even at v_2, if the coalition always plays a^ω (resp., b^ω), then it is losing for $k = 1$ (resp., $k \geq 3$).

In the next section, we briefly describe the results which we present in the following chapters.

4.4 Discussion

In this chapter, we described games played on a finite graph. In Section 4.1, we revisited the basic definitions and results from literature on a two-player turn-based setting where two-players choose their moves in turn and then Section 4.2 was dedicated to a concurrent game setting where at each round, the players play their moves simultaneously and independently of each other. We then extend the latter in Section 4.3 in a way suitable for a parameterized setting, where the parameter is the number of players participating in the game. In a parameterized arena, edges are labelled with regular languages given by NFA's, and represented by regular expressions in the examples. An arena can be non-deterministic, and the non-determinism is resolved by an adversarial environment. Such a parameterized arena can represent infinitely many interaction situations, one for each possible number of agents. We are then interested in two different settings for the parameterized games, first where a distinguished player (Eve) plays uniformly against all other players, not knowing how many are there; and second, we consider a coalition game

where the coalition collectively wants to satisfy a winning condition for any number of agents.

In the first setting, we shall show that the decision problem of finding whether there exists a winning strategy for Eve with a reachability objective against any number of opponents is PSPACE-complete, when the languages that label the edges are regular. For simpler languages, we show that the problem is even easier. More precisely, if the languages are such that their projections to the lengths of the words are only intervals, then the problem can be solved in *polynomial time* and if they are finite union of intervals, then it is NP-complete, if the arena is deterministic and PSPACE-complete, otherwise. The proof technique involves a reduction of such a game to a two-player turn-based game. Intuitively, Eve must win against her opponents playing as a coalition and also against an adversarial environment who chooses the number of players and resolves the non-determinism. Therefore, the game naturally seems close to a two-player setting between Eve and the environment, where the environment also chooses her opponents' actions and the environment's objective is to prevent her from winning. The two-player game constructed from a parameterized game is called a *knowledge game* (Eve's vertices in that game correspond to her knowledge in the original game) and then the complexity results follow from an analysis of the knowledge game.

In the coalition game setting, we consider the winning condition for the coalition is a safety condition. We then prove that the *safe coalition problem* can be solved in space exponential in the size of the input, and it is hard for the complexity class PSPACE. We also show that for positive instances, one can synthesize a coalition strategy profile using exponential space and the strategy uses an exponential memory, which is tight in the sense that there is a family of games $(\mathcal{G}_n)_n$ of size polynomial in n for which any coalition winning strategy needs exponential size memory. The upper bound result for the safe coalition problem is shown by first unfolding the game graph to a finite tree and then analysing the complexity of finding a coalition strategy for a game suitably defined on that finite tree structure. A key observation here is that, since the winning condition for the coalition is safety, the coalition can play the same strategy every time the game comes back to a certain vertex, and if it was winning in the first round, then it is also in the subsequent rounds - this is the main idea behind the finite tree unfolding. However, this argument does not hold in the case of, for instance, a reachability condition which makes the coalition problem harder for a reachability condition.

Chapter 5

Playing against Arbitrarily Many Opponents

This chapter studies parameterized games for a setting where a distinguished player (we call that player Eve throughout this chapter) wants to achieve a goal irrespective of her number of opponents and their choices of actions. The opponents play as a coalition against Eve, and she must play uniformly, whatever the number of opponents she has. Recall that, we assume that the languages on the transitions of a parameterized arena are regular languages given by an NFA and represented by regular expressions in the examples.

We first argue that such games reduce to a simpler setting, called *semi-parameterized arenas*, where the languages on the edges of the arena are particularly simple: they only constrain the number of opponents Eve has. The existence of a uniform winning strategy for Eve in the latter then reduces to the resolution of the *knowledge game*, a two-player turn-based game. We show that Eve has a winning strategy in the semi-parameterized game if and only if she also has one in the corresponding knowledge game for the same winning objective. Intuitively, a vertex of Eve in the knowledge game corresponds to her knowledge about the number of opponents in the parameterized game.

We investigate the existence of a winning strategy for Eve in such games. We will show tight bounds for the decision problem on semi-parameterized arenas, hence on parameterized arenas, with reachability objective for Eve. Furthermore, we distinguish several cases, depending on whether the arena is deterministic or not, and on whether constraints on the number of opponents are intervals, unions of intervals, or semilinear sets. We show that the knowledge game is *a priori* exponential in the size of the original arena. Intuitively, the vertices correspond to the knowledge Eve has on the possible number of her opponents, and there are in general exponentially many such knowledge sets. Furthermore, we show that this exponential blow-up in the size of knowledge game is unavoidable. However, when constraints are only intervals, the size of the knowledge game is polynomial, in that case, we prove the semi-parameterized reachability game problem to be PTIME-complete. For finite unions of intervals, and when the parameterized arena is deterministic, we show that if Eve has a winning strategy, she has one that can be represented by a polynomial size strategy tree. This small model property, together with

the encoding of 3SAT allows us to prove the problem to be NP-complete. Finally, for finite unions of intervals and non-deterministic arenas, or for semilinear sets (with no assumption of non-determinism) the semi-parameterized reachability problem is PSPACE-complete. The lower bound is obtained by a reduction from QBF-SAT, while the upper bound derives from a depth-first search algorithm on an exponential size tree, non-trivially extracted from the knowledge game.

This chapter is based on the publication [BBM19a] co-authored with Nathalie Bertrand and Patricia Bouyer appeared in FSTTCS 2019.

Organization of the chapter

In Section 5.1, we first recall briefly the setting *Eve against unknown number of opponents* on parameterized arenas and define a simpler setting, called *semi-parameterized games*. We then show a reduction from the former to the latter. Section 5.2 describes a decision procedure for the semi-parameterized game problem by construction of a two-player turn-based game, called *knowledge game*. Then we restrict the winning objective to reachability and show tight bounds for the game problem on a semi-parameterized arena in Section 5.3. This bounds entail bounds on the original problem on parameterized arenas. Finally, we close the chapter with a discussion in Section 5.4.

5.1 Game setting

Let us first briefly recall the game setting *Eve against unknown number of opponents* on parameterized arenas from Section 4.3 that we aim to study in this chapter. Note that in this setting, it is reasonable to assume Eve is not alone in the game. Therefore, we will assume there is at least one opponent of Eve in the parameterized game. That is, the languages labelling the edges of the arena consist of words of length greater than or equal to 2.

Fix a parameterized arena $\mathcal{A} = \langle V, \Sigma, \Delta \rangle$. A strategy for agent i is a mapping $\sigma_i : V^+ \to \Sigma$ from histories to the set of actions. We distinguish the actions of player 1 (called Eve) and consider a winning objective Win for her. This defines a parameterized game $\mathcal{G} = (\mathcal{A}, \mathsf{Win})$. We define the *outcome* of a strategy σ_1 from v_0 for Eve as the union of *k-outcomes* over all $k \geq 2$ and all possible strategies σ_i for player i with $2 \leq i \leq k$; formally: $\mathsf{Out}_{\mathcal{A}}(v_0, \sigma_1) = \bigcup_{k \geq 2} \bigcup_{\sigma_2, \ldots, \sigma_k} \mathsf{Out}^k_{\mathcal{A}}(v_0, \sigma_1, \ldots, \sigma_k)$. A strategy σ_1 from v_0 for Eve is *winning* for her if all plays in the outcome $\mathsf{Out}_{\mathcal{A}}(v_0, \sigma_1)$ belong to Win. More precisely, σ_1 is winning for Eve if for any $k-1 \in \mathbb{N}_{>0}$ number of opponents, and their strategies $\sigma_2, \ldots, \sigma_k$, the following is satisfied: $\mathsf{Out}^k_{\mathcal{A}}(v_0, \sigma_1, \sigma_2, \ldots, \sigma_k) \subseteq \mathsf{Win}$. We define the following decision problem:

Eve AGAINST UNKNOWN NUMBER OF OPPONENTS
Input: A parameterized game $\mathcal{G} = (\mathcal{A}, \mathsf{Win})$ and an initial vertex v_0.
Question: Yes if and only if $\exists \sigma_1$ for Eve such that $\mathsf{Out}_{\mathcal{A}}(v_0, \sigma_1) \subseteq \mathsf{Win}$.

Notice that Eve does not know the number of opponents *a priori* and their strategies, so the opponents play as a coalition against Eve. Furthermore, the environment chooses the number of her opponents before the game starts, and also resolves the non-determinism of the transitions. Eve must win whatever the adversarial environment. One can therefore think of environment also choosing strategies of the coalition of her opponents and, this reduces to the problem to a simpler setting where Eve's strategy only takes into account the number of her opponents, not their strategies. This motivates to define simpler game arenas where the edges are labelled with pairs of Eve's actions and sets of number of opponents.

In the following, we introduce *semi-parameterized arenas* and define a game for Eve on such arenas. Edges are labelled with pairs (a, S) for a an action of Eve, and S a set of number of opponents. Notice that unlike the game on a parameterized arena, here k denotes the number of opponents (and, $k+1$ is the total number of agents including Eve participating in the game). Then we later show a reduction from the first setting to the latter such that the winning region for Eve is preserved, that is Eve has a winning strategy in the former setting if and only if she has one in the reduced game.

5.1.1 Semi-parameterized arenas

We introduce *semi-parameterized arenas*, a simpler and restricted version of parameterized arenas, where the languages on the edges of the arena only constrain the number of opponents Eve has. Furthermore, for simplicity, we distinguish the actions of Eve. More precisely, the edges here are labelled with pairs (a, S) for a an action of Eve, and S a set of number of opponents. In other words, if at vertex v, Eve chooses action a, and if there is a set of naturals $S \subseteq \mathbb{N}_{>0}$ such that $\Delta(v, a, v') = S$, then for any $k \in S$ number of opponents, the game can move to v'. We will later show a correspondence between winning strategies of Eve in parameterized arenas and semi-parameterized arenas.

Definition 5.1. *A* semi-parameterized arena *is a tuple* $\mathcal{A} = \langle V, \Sigma, \Delta \rangle$ *where*

- V *is a finite set of vertices;*
- Σ *is a finite set of actions;*
- $\Delta : V \times \Sigma \times V \to 2^{\mathbb{N}_{>0}}$ *is a transition function.*

Fix a semi-parameterized arena $\mathcal{A} = \langle V, \Sigma, \Delta \rangle$. The assumption that there is at least one opponent of Eve participating in the game is ensured in the definition of the transition function. For some $v, v' \in V$ and $a \in \Sigma$, $\Delta(v, a, v')$ is the set of numbers of Eve's opponents

that may lead the game from v to v' on Eve's action a: if at vertex v, Eve chooses action a, and if the number of her opponents k belongs to the set $\Delta(v, a, v')$, then the game can move to v' (up to non-determinism, which is resolved by the environment).

The arena $\mathcal{A}$ is *deterministic* if for every $v \in V$, every action $a \in \Sigma$ of Eve and for any k number of her opponents, there is at most one vertex $v' \in V$ such that $k \in \Delta(v, a, v')$. The *completeness* assumption can be described as follows: for every $v \in V$, action $a \in \Sigma$, and $k \in \mathbb{N}_{>0}$, there exists $v' \in V$ such that $k \in \Delta(v, a, v')$. The idea behind the assumption is that Eve does not know how many opponents she has, and so the successor vertex must exist whatever that number is, and for every possible action of her. Recall that for conciseness, examples (for instance, the one in Figure 4.6) might depict incomplete arenas, however, a sink vertex can be added and all unspecified transitions can lead to that vertex to obtain a complete one.

For algorithmic reasons, we assume that the transition function Δ of $\mathcal{A}$ can be described in a finite way. We consider different types of constraints to represent the transition function. We first consider constraints described by closed intervals (since we deal with sets of natural numbers, it is no restriction to assume intervals to be closed) or finite unions of closed intervals. If $[a, b]$ (resp., $[a, \infty)$) is an interval, then we say a is a left endpoint and b (resp., ∞) is a right endpoint. As a complexity parameter, we use $\#\mathsf{endpoints}_{\mathcal{A}}$, the number of endpoints used in constraints in $\mathcal{A}$. All the complexities will be functions of this parameter, independently of the precise values of the endpoints (or their encoding sizes). More generally, we also consider semilinear predicates over $\mathbb{N}$. We recall the definition of a semilinear set in the following. In that context, as a complexity parameter, we use $\#\mathsf{pred}_{\mathcal{A}}$, the number of predicates used on edges of $\mathcal{A}$. All numbers used in the predicates are assumed to be represented in binary.

Definition 5.2. *A set $L \subseteq \mathbb{N}$ is called* linear *if it is of the form*

$$L = L(b, P) = \{b + \sum_{i=1}^{m} \lambda_i p_i : \lambda_i \in \mathbb{N},\ p_i \in P\}$$

where $b \in \mathbb{N}$ and $P \subseteq \mathbb{N}$ is a finite set. We call b the basis *and $p \in P$ the* periods *of L. A set $S \subseteq \mathbb{N}$ is called* semilinear *if it is a finite union of linear sets.*

Example 5.3. *A simple example of a semilinear predicate is the predicate "divisible by p", where $p \in \mathbb{N}_{>0}$.*

Note here that the above is defined for subsets of natural numbers, and one can naturally extend it for any fixed dimension $d \in \mathbb{N}_{>0}$, however, in this chapter, we will only use semilinear sets in dimension 1. Remark that a finite union of arithmetic progressions of the form $\{c + d\mathbb{N} : c, d \in \mathbb{N}\}$ is semilinear. Moreover, the converse is also true: any semilinear set over $\mathbb{N}$ can be equivalently represented as a finite union of arithmetic progressions, for example see [Mat94].

Let us now give an example of a semi-parameterized arena.

Example 5.4. *An example of a deterministic semi-parameterized arena is presented in Figure 5.1, with* $V = \{v_0, \dots, v_6\}$, $\Sigma = \{a, b\}$. *The edge labels represent the transition function: for instance, the label '*$a, = 1$*' on the transition from* v_0 *to* v_1 *represents* $\Delta(v_0, a, v_1) = \{1\}$, *and the label '*$a, \neq 1$*' on the transition from* v_0 *to* v_2 *means that* $\Delta(v_0, a, v_2) = \mathbb{N}_{>0} \setminus \{1\} = [2, \infty)$. *As such, the arena is not complete, we assume that all unspecified transitions from any vertex lead to* v_6. *Moreover, we omit the trivial components in a transition, for instance,* $\Delta(v_0, b, v_6) = \mathbb{N}_{>0}$; *further, we denote a transition by '*$*$*' if it is trivial, for example, for every action* $a \in \Sigma$ *of* Eve, $\Delta(v_1, a, v_3) = \mathbb{N}_{>0}$. *Notice that this example coincides with the one in Figure 4.4, except here we have replaced the languages on the edges by their projections to lengths of words.*

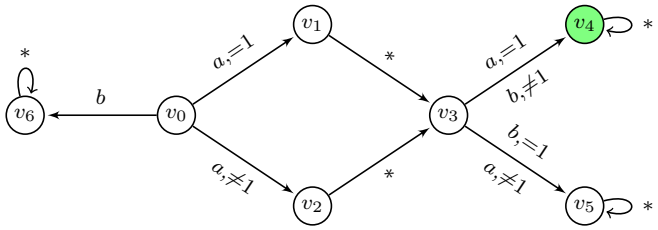


Figure 5.1: An example of a semi-parameterized arena.

We define the notions of *history* and *play* in a semi-parameterized arena similarly as in the parameterized arenas.

Definition 5.5. *Fix a semi-parameterized arena* $\mathcal{A} = \langle V, \Sigma, \Delta \rangle$. *A* history *in* $\mathcal{A}$ *is a finite sequence of vertices* $h = v_0 v_1 \dots v_p \in V^+$ *such that for every* $0 \leq j < p$, *there exists* $a_j \in \Sigma$ *such that* $\Delta(v_j, a_j, v_{j+1}) \neq \emptyset$. *We write* $\mathsf{Hist}_{\mathcal{A}}$ *for the set of all histories. A* play *is an infinite sequence of vertices compatible with the edges:* $\rho = v_0 v_1 \dots \in V^\omega$ *such that for every* $j \geq 0$, *there exists* $a_j \in \Sigma$ *such that* $\Delta(v_j, a_j, v_{j+1}) \neq \emptyset$. *The set of plays is denoted* $\mathsf{Play}_{\mathcal{A}}$.

Next, we adapt the definition of *realizability* of histories and plays from Chapter 4 to the context of semi-parameterized arenas. A history (resp., play) is *k-realizable* if it corresponds to a history (resp., play) compatible with some k number of Eve's opponents. Notice that for the game setting on a semi-parameterized arena, the parameter k denotes the number of opponents Eve has whereas in the definitions of Chapter 4, k was the total number of agents participating in the game including Eve.

Definition 5.6. *For* $k \in \mathbb{N}_{>0}$, *a history* $h = v_0 \cdots v_p$ *is* k-realizable *if there exists* $k \in \mathbb{N}_{>0}$ *such that for all* $0 \leq j < p$, *there exists* $a_j \in \Sigma$ *with* $k \in \Delta(v_j, a_j, v_{j+1})$ *(or equivalently,* $k \in \bigcap_{0 \leq j < p} \Delta(v_j, a_j, v_{j+1})$*). A history is* realizable *if it is* k*-realizable for some* $k \in \mathbb{N}_{>0}$. *Similarly to histories for finite sequences of consecutive vertices, one can define the notions of* (k-)realizable plays *for infinite sequences.*

Let us define the notion of strategy for Eve, but on a semi-parameterized arena. A strategy dictates her how to move from a vertex depending on the past history. A strategy for Eve must be defined with no prior information on the number of her opponents. It is formally defined below.

Definition 5.7. *A* strategy *for* Eve *from vertex* v_0 *is a mapping* $\sigma : V^+ \to \Sigma$ *that associates to every history* $h = v_0 v_1 \dots v_p$ *in* $\mathsf{Hist}_{\mathcal{A}}$ *an action* a *in* Σ.

Given a strategy σ for Eve, an initial vertex v and $k \in \mathbb{N}_{>0}$ a number of opponents of Eve, the notions of (k-)outcome are defined similarly: the *k-outcome* $\mathsf{Out}^k_{\mathcal{A}}(v_0, \sigma)$ is the set of plays that σ induces from v_0 that are k-realizable; and then the *outcome* is simply the union of k-outcomes for all $k \in \mathbb{N}_{>0}$. It is defined formally below.

Definition 5.8. *Given a strategy* σ *for* Eve*, an initial vertex* v_0 *and* $k \in \mathbb{N}_{>0}$ *a number of opponents of* Eve*, the* k-outcome *of strategy* σ *is the set* $\mathsf{Out}^k_{\mathcal{A}}(v_0, \sigma) = \{v_0 v_1 \cdots \mid \forall j \geq 0, k \in \Delta(v_j, \sigma(v_0 \dots v_j), v_{j+1})\}$. Outcome *of strategy* σ *is* $\mathsf{Out}_{\mathcal{A}}(v_0, \sigma) = \bigcup_{k \in \mathbb{N}_{>0}} \mathsf{Out}^k_{\mathcal{A}}(v_0, \sigma)$.

A game on a semi-parameterized arena proceeds as follows. Fix an arena $\mathcal{A} = \langle V, \Sigma, \Delta \rangle$ and an initial vertex v_0. Before the game starts, the environment chooses k the number of opponents of Eve that remains unknown to her. She chooses an action a_1; and then the game moves to v_1 such that $k \in \Delta(v_0, a_1, v_1)$; if there exists more than one vertex satisfying the above, the environment chooses one of them non-deterministically. The game then proceeds similarly from v_1.

Fix a semi-parameterized arena $\mathcal{A}$ and an initial vertex v_0. Then a set $\mathsf{Win} \subseteq V^\omega$ of plays defines a semi-parameterized game $\mathcal{G} = (\mathcal{A}, \mathsf{Win})$ for Eve. A strategy σ for Eve from v_0 is *winning* if all the plays in the outcome of σ is in Win, that is if for all $k \in \mathbb{N}_{>0}$ number of opponents, $\mathsf{Out}^k_{\mathcal{A}}(v_0, \sigma) \subseteq \mathsf{Win}$. In that case, we say that v_0 belongs to the *winning region* of Eve. We now define the following decision problem:

Semi-parameterized game problem
Input: A semi-parameterized game $\mathcal{G} = (\mathcal{A}, \mathsf{Win})$ and an initial vertex v_0.
Question: Yes if and only if $\exists \sigma$ for Eve such that $\mathsf{Out}_{\mathcal{A}}(v_0, \sigma) \subseteq \mathsf{Win}$.

Let us illustrate this setting on an example.

Example 5.9. *Resuming Example 5.4, one can show that* Eve *has a winning strategy from* v_0 *ensuring the* reachability *objective described by* $F = \{v_4\}$*, depicted in green in the picture. Her winning strategy* σ *is given by* $\sigma(v_0) = \sigma(v_0 v_1) = \sigma(v_0 v_2) = a$*,* $\sigma(v_0 v_1 v_3) = a$ *and* $\sigma(v_0 v_2 v_3) = b$*, and arbitrary otherwise. Intuitively, the decision at vertex* v_3 *depends on whether the play went through* v_1 *–in this case* Eve *deduces that she has a single opponent– or* v_2*; in the first case, she chooses action* a*, and* b *in the second. In both cases, the game moves to* v_4 *and loops there forever. Note that no memoryless strategy is winning for* Eve*: if she always chooses* a *at* v_3*, she is losing against more than 1 opponents; and similarly, if she always chooses* b *at* v_3*, she is losing when she has exactly 1 opponent. The winning region for the reachability objective described above for* Eve *is* $\{v_0, v_4\}$.

Among all ω-regular winning objectives, the reachability objective is of special interest to us. We show tight complexity bounds for the game problem with reachability objectives

on semi-parameterized arenas. A reachability game, with no loss of generality, can be described by a unique target vertex t. A strategy σ for Eve is winning if all plays in the outcome of σ eventually visits t. Then the reachability game problem asks whether there exists a winning strategy for Eve from an initial vertex. This can be defined as follows:

Semi-parameterized reachability game problem
Input: A semi-parameterized reachability game $\mathcal{G} = (\mathcal{A}, t)$ and an initial vertex v_0.
Question: Yes if and only if $\exists\sigma$ for Eve such that $\mathsf{Out}_{\mathcal{A}}(v_0, \sigma) \subseteq V^+ \cdot t \cdot V^\omega$.

Notice first that parameterized games generalize semi-parameterized ones. Indeed, It is well-known that from a semilinear set S in dimension 1, one can construct an automaton $\mathcal{B}$ over a unary alphabet such that the lengths of words in the language of $\mathcal{B}$ is S *i.e.*, $|\mathcal{L}(\mathcal{B})| = S$, for instance see [Mat94]. Then one can replace the transitions of the form $\Delta_{\mathcal{A}}(v, a, v')$ in a semi-parameterized arena $\mathcal{A}$ by *regular* languages $a\mathtt{L}$ (*i.e.*, set $\Delta_{\mathcal{A}'}(v, v') = a\mathtt{L}$) to construct a parameterized arena $\mathcal{A}'$, such that $\mathtt{L}$ is a regular language over a unary alphabet with $|\mathtt{L}| = \Delta_{\mathcal{A}}(v, a, v')$. Indeed, one can show that if $\mathtt{L}$ is regular, then for any $a \in \Sigma$, $a\mathtt{L}$ is also regular. Since the alphabet is unary, $\mathtt{L}$ essentially only constrains the number of opponents of Eve. The winning region for Eve is also preserved.

Lemma 5.10. *Semi-parameterized games reduce in polynomial time to the parameterized ones.*

In the following, we show that the converse also holds, *i.e.*, the existence of a winning strategy for Eve in a parameterized arena reduces to the one in a semi-parameterized arena, by simply taking the projection of the languages to lengths of the words.

5.1.2 Reduction to semi-parameterized arenas

Let $\mathcal{G} = \langle \mathcal{A}, \mathsf{Win} \rangle$ be a parameterized game. We shall construct an equivalent (*i.e.*, preserving the winning region for Eve) semi-parameterized game in polynomial time such that the edges are labelled with semilinear constraints. Intuitively, the constraints in the latter are the set of lengths of words in the corresponding regular languages in the original arena quotiented by an action of Eve.

For a language $L \subseteq \Sigma^*$, and $a \in \Sigma$, the *left quotient* of L by a is the set $a^{-1}L = \{w \mid aw \in L\}$. It is well-known that for a regular L, the quotient language $a^{-1}L$ is also regular.

We will use another result from literature, namely that the set of lengths of words of a regular language L, denoted $|L|$, is a semi-linear set in dimension 1 [Par66]. Moreover, given an automaton $\mathcal{A}$ for L with n states, one can compute in polynomial time the set $|L|$ represented as union of $O(n^2)$ arithmetic progressions of the form $\{c + d\mathbb{N} \mid c, d \in \mathbb{N}\}$ [Chr86, Mar02, Saw10].

Lemma 5.11. *The decision problem '*Eve against unknown number of opponents*' on a parameterized arena reduces in polynomial time to the semi-parameterized game problem*

(with semilinear predicates).

Proof. Let $\mathcal{G} = (\mathcal{A}, \mathsf{Win})$ be a parameterized game with $\mathcal{A} = \langle V, \Sigma, \Delta \rangle$, we construct $\mathcal{G}' = (\mathcal{A}', \mathsf{Win})$ a semi-parameterized game with $\mathcal{A}' = \langle V, \Sigma, \Delta' \rangle$ as follows. The edges in $\mathcal{A}'$ are labelled with pairs (a, S) for $a \in \Sigma$ an action for Eve and S is obtained by first taking a left quotient of language on that edge in $\mathcal{A}$ by action a, and then projecting the obtained language to lengths of words. Formally, for $v, v' \in V$ and $a \in \Sigma$, define $\Delta'(v, a, v') = |a^{-1}\Delta(v, v')|$. Note that given an automaton for $\Delta(v, v')$, one can compute in polynomial time a representation for $\Delta(v, a, v')$ as a union of polynomially many arithmetic progressions of the form $\{c + d\mathbb{N}\}$. Thus, a parameterized game $\mathcal{G}$ can be transformed into $\mathcal{G}'$ a semi-parameterized one in polynomial time in the size of $\mathcal{G}$. We fix the winning conditions to be the same in both games.

Assume Eve has a winning strategy σ_1 from an initial vertex v_0 in $\mathcal{G}$. The same strategy can also be applied in $\mathcal{G}'$. We show σ_1 is winning in $\mathcal{G}'$. Let $\rho = v_0 v_1 \ldots \in \mathsf{Out}^k_{\mathcal{A}'}(v_0, \sigma_1)$ be a k-realizable play in the outcome of σ_1 in $\mathcal{G}'$. Then for all $j \geq 0$, $\exists L$: $\Delta'(v_j, a_j, v_{j+1}) = |a_j^{-1}L|$, where $a_j = \sigma_1(v_0 \ldots v_{j-1})$ for which $k \in |a_j^{-1}L|$. Therefore, there exists $b_1, \ldots, b_k$ actions of opponents such that $a_j b_1 \ldots b_k \in L$. We conclude ρ is a $(k+1)$-realizable play induced by σ_1 in $\mathcal{G}$, and hence $\rho \in \mathsf{Win}$.

For the converse, assume σ_1 from v_0 is winning for Eve in $\mathcal{G}'$. Apply σ_1 in $\mathcal{G}$, we show it is winning for Eve in $\mathcal{G}$. Consider a play $\rho = v_0 v_1 \ldots \in \mathsf{Out}^k_{\mathcal{A}}(v_0, \sigma_1, \sigma_2, \ldots \sigma_k)$ for arbitrary strategies of her opponents. Then for all $j \geq 0$, $\exists L$: $\Delta(v_j, v_{j+1}) = L$ and the word $w = a_1 a_2 \ldots a_k \in L$ where $a_k = \sigma_i(v_0 \ldots v_{j-1})$ for each $1 \leq i \leq k$. Therefore, $k - 1 \in |a_1^{-1}L|$. We conclude ρ is a $(k-1)$-realizable play induced by σ_1 in $\mathcal{G}'$, and hence conclude $\rho \in \mathsf{Win}$. □

Lemma 5.11 establishes the correspondence between games on parameterized and semi-parameterized arenas. Further, we concentrate on the latter arenas and discuss a decision procedure of the game problem. In the next section, we begin with the description of the *knowledge game*, a two-player turn-based game associated with a semi-parameterized arena.

5.2 The knowledge game

Recall that to win a game on the semi-parameterized arena, Eve must win against any number of opponents as well as an adversarial environment who (selects that number at the beginning and also) resolves the non-determinism of transitions. It therefore seems quite natural to reduce this game to a two-player turn-based game between Eve and Adam, who corresponds to the environment in the original setting. In the so-called *knowledge game*, a vertex of Eve intuitively embeds her knowledge against the number of her opponents at the corresponding position in the semi-parameterized game. Later we will show that this reduction preserves the winning region for Eve.

Let us first define the *knowledge arena*, a two-player turn-based game arena constructed from a semi-parameterized arena.

Definition 5.12. *Let $\mathcal{A} = \langle V, \Sigma, \Delta \rangle$ be a semi-parameterized arena. Then the* knowledge arena *associated with $\mathcal{A}$ is the two-player turn-based arena $\mathcal{K}_\mathcal{A} = \langle V_\mathcal{K}, E_\mathcal{K} \rangle$, where $V_\mathcal{K} = V_\mathsf{E} \uplus V_\mathsf{A}$ such that $V_\mathsf{E} \subseteq V \times 2^{\mathbb{N}_{>0}}$ and $V_\mathsf{A} \subseteq V_\mathsf{E} \times \Sigma$ are vertices of* Eve *and* Adam*, respectively, and $E_\mathcal{K} \subseteq (V_\mathsf{E} \times V_\mathsf{A}) \cup (V_\mathsf{A} \times V_\mathsf{E})$ is the edge relation. The vertices and edge relation are defined inductively as follows.*

- $\{(v, \mathbb{N}_{>0}) \mid v \in V\} \subseteq V_\mathsf{E}$;
- $\forall (v, K) \in V_\mathsf{E}$, *and* $\forall a \in \Sigma$, $(v, K, a) \in V_\mathsf{A}$ *and* $\big((v, K), (v, K, a)\big) \in E_\mathcal{K}$;
- $\forall (v, K, a) \in V_\mathsf{A}$, $\forall v' \in V$ *such that* $K \cap \Delta(v, a, v') \neq \emptyset$, $(v', K \cap \Delta(v, a, v')) \in V_\mathsf{E}$ *and* $\big((v, K, a), (v', K \cap \Delta(v, a, v'))\big) \in E_\mathcal{K}$.

In words, the knowledge arena has a vertex $(v, \mathbb{N}_{>0})$ for every $v \in V$ which belongs to Eve. A vertex of Eve (v, K) intuitively represents the situation when the token in the semi-parameterized arena is currently on vertex v and the knowledge she has about her number of opponents is K. For instance, $(v, \mathbb{N}_{>0})$ in the knowledge arena represents that the corresponding position in the semi-parameterized arena is v and Eve does not have any particular knowledge about the number of her opponents. Then for any choice of action by Eve, the game moves to a vertex of Adam, which intuitively represents the position where the environment resolves the non-determinism in the semi-parameterized arena. That is, for any $(v, K) \in V_\mathsf{E}$ and any $a \in \Sigma$, there is a successor (v, K, a) which belong to Adam. Each transition from of a vertex of Adam corresponds to a similar transition in the original arena and the successor is again a vertex of Eve where she updates her knowledge accordingly. For instance, if there is a transition in $\mathcal{G}$ from v to v' on Eve's action a such that $K \cap \Delta(v, a, v') \neq \emptyset$, then $(v', K \cap \Delta(v, a, v'))$ is a successor of (v, K, a) in the knowledge arena. The game played on that arena is a two-player turn-based game discussed in Section 4.1. Notice further that we construct the knowledge arena such that any successor of a vertex of Eve must be a vertex of Adam and vice versa.

Let us illustrate the construction of knowledge arena on an example.

Example 5.13. *Figure 5.2 represents a part of the knowledge arena associated with the semi-parameterized arena of Figure 5.1. Circles represent* Eve*'s vertices, and rectangular vertices belong to* Adam*. Note that circles in Figure 5.2 should not be confused with the ones in the semi-parameterized arena. Also note that, for a concise representation, the knowledge arena depicted here is only a part of the full knowledge arena of the corresponding semi-parameterized arena; only the vertices which are constructed inductively from $(v_0, \mathbb{N}_{>0})$ are depicted, and furthermore, the outgoing transitions from $(v_4, K), (v_5, K')$ for all possible K, K' and transitions from $(v_6, \mathbb{N}_{>0})$ are also not depicted. They are the only reachable vertices of* Eve *in the knowledge arena when the game starts at vertex v_0 in the corresponding semi-parameterized arena.*

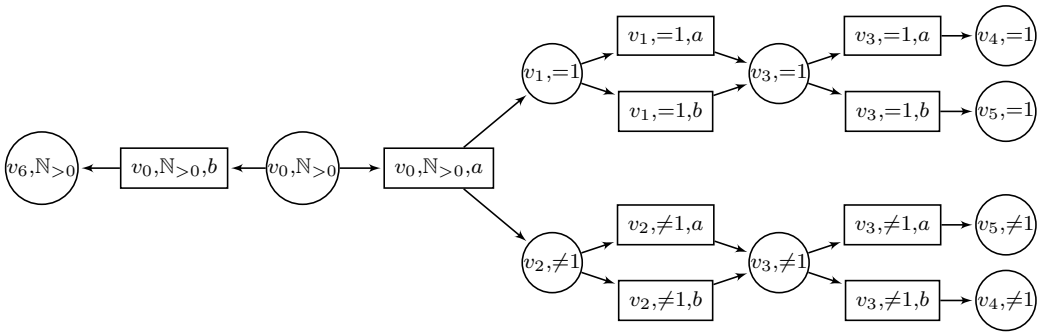


Figure 5.2: A part of the knowledge arena corresponding to the arena in Figure 5.1.

We recall the notions of histories and plays of a two-player turn-based game in this context. A *history* in the knowledge arena is a finite alternating sequence of vertices of Eve and Adam compatible with the edge relation. Formally, $\mathsf{h} = \mathsf{v}_0\mathsf{v}_1 \dots \mathsf{v}_t \in V_{\mathcal{K}}^+$ such that for every $0 \leq i < t$, $(\mathsf{v}_i, \mathsf{v}_{i+1}) \in E_{\mathcal{K}}$. The set of histories is denoted $\mathsf{Hist}_{\mathcal{K}}$. Similarly, a play is an infinite sequence of vertices that preserves the edge relation. The set of plays is denoted $\mathsf{Play}_{\mathcal{K}}$.

A *strategy* for Eve from v_0 in $\mathcal{K}_{\mathcal{G}}$ is a partial function $\lambda : V_{\mathcal{K}}^+ \to V_{\mathsf{A}}$ compatible with $E_{\mathcal{K}}$ that associates to every history $\mathsf{v}_0 \dots \mathsf{v}$, with $\mathsf{v} \in V_{\mathsf{E}}$, a vertex $\lambda(\mathsf{v}) \in V_{\mathsf{A}}$ such that $(\mathsf{v}, \lambda(\mathsf{v})) \in E_{\mathcal{K}}$. A play $\rho = \mathsf{v}_0\mathsf{v}_1 \dots$ from v_0 is *induced by* strategy λ if for every $i \geq 0$, whenever $\mathsf{v}_i \in V_{\mathsf{E}}$, we have $\mathsf{v}_{i+1} = \lambda(\mathsf{v}_0 \dots \mathsf{v}_i)$.

Size of the knowledge arena. Fix a semi-parameterized arena $\mathcal{A} = \langle V, \Sigma, \Delta \rangle$, and let $\mathcal{K}_{\mathcal{A}}$ be the corresponding knowledge arena. First, there is a vertex of Eve $(v, \mathbb{N}_{>0})$ for every $v \in V$. Notice that a vertex of Eve (v, K) is constructed such that the set K is formed by taking an intersection of some set K' present in its predecessor vertex with a $S \subseteq \mathbb{N}_{>0}$ from the description of input $\mathcal{A}$. Since the number of subsets $S \subseteq \mathbb{N}_{>0}$ given in the semi-parameterized arena $\mathcal{A}$ is finite, one can construct only finitely many sets by taking intersection of such sets S. Therefore, we conclude that the number of vertices of Eve is finite. A vertex (v, K, a) of Adam is constructed such that (v, K) is a vertex of Eve and $a \in \Sigma$ is an action; therefore, the number of vertices of Adam is also finite since Σ is finite. Hence, the knowledge arena is indeed finite.

Next we present a lemma investigating closely the size of $\mathcal{K}_{\mathcal{A}}$, that is the number of its vertices and edges, w.r.t. the complexity measures we introduced for the parameterized arena $\mathcal{A}$.

Lemma 5.14. *Let $\mathcal{A} = \langle V, \Sigma, \Delta \rangle$ be a semi-parameterized arena. Then the size of the associated knowledge arena $\mathcal{K}_{\mathcal{A}}$ is polynomial in both $|V|$ and $|\Sigma|$, and*

1. *exponential in $\#\mathsf{pred}_{\mathcal{A}}$, for sets of number of opponents defined by semilinear predicates;*
2. *exponential in $\#\mathsf{endpoints}_{\mathcal{A}}$, for sets of number of opponents defined by finite unions of disjoint intervals; and*

3. *polynomial in* $\#\mathsf{endpoints}_{\mathcal{A}}$*, for sets of number of opponents defined by intervals.*

Furthermore, the exponential blow-up is unavoidable in the two first cases.

Proof. Let $\mathcal{K}_{\mathcal{A}} = \langle V_{\mathcal{K}}, E_{\mathcal{K}} \rangle$ be the knowledge arena associated with $\mathcal{A}$ where $V_{\mathcal{K}} = V_{\mathsf{E}} \uplus V_{\mathsf{A}}$. By definition, all pairs $(v, \mathbb{N}_{>0})$ for $v \in V$ belong to V_{E} representing that Eve has no initial knowledge of the number of her opponents. Further knowledge sets for vertices in $\mathcal{K}_{\mathcal{A}}$ are obtained by taking the intersection of existing knowledge sets with subsets $S \subseteq \mathbb{N}_{>0}$ from the description of $\mathcal{A}$. Then we show the following.

1. First, when the sets of number of opponents in the arena are given by semilinear predicates, the number of knowledge sets is bounded by $2^{\#\mathsf{pred}_{\mathcal{A}}}$. Indeed, one can consider at most those many combinations of the predicates. A vertex of Eve is of the form (v, K) for some $v \in V$ and K a knowledge set. Hence, in this case, $|V_{\mathsf{E}}| \leq 2^{\#\mathsf{pred}_{\mathcal{A}}}|V|$; and a vertex of Adam is of the form (v, K, a) for some $(v, K) \in V_{\mathsf{E}}$ and $a \in \Sigma$ an action, therefore $|V_{\mathsf{A}}| \leq |V_{\mathsf{E}}||\Sigma| \leq 2^{\#\mathsf{pred}_{\mathcal{A}}}|V||\Sigma|$, yielding an overall exponential bound on $|\mathcal{K}_{\mathcal{A}}|$. Note that it is exponential in the number of predicates, but not in the size of their encodings.

2. Let us now show that when the sets of number of opponents are defined by finite unions of disjoint intervals, the number of knowledge sets is bounded by $3^{\#\mathsf{endpoints}_{\mathcal{A}}}$. Note that a finite union of intervals can be encoded by a word on the alphabet formed of the set of endpoints, with a repetition for singletons. Let us illustrate the statement by an example. Let $Y = \{2, 5, 8, 11, 17, 23, \infty\}$ be a set of endpoints appearing in an arena, then writing a_i for the i-th number in Y in that particular order, the set $S = [2, 8] \cup \{11\} \cup [17, \infty)$, for instance, can be represented by the string $a_1a_3a_4a_4a_5a_7$. Also recall that any intersection of unions of disjoint intervals with endpoints in Y is again a union of disjoint intervals with endpoints from the same set Y. Then indeed, given a finite set Y, one can construct at most $3^{|Y|}$ disjoint unions of intervals: in the word representation, each letter can occur either 0, 1 or 2 times in a word; and since the intervals must be disjoint, when valid, a word uniquely represents a union of intervals (some words might not be valid, for instance, the word $a_1a_3a_2a_4$ in the previous example does not correspond to any union of disjoint intervals). Hence, $|V_{\mathsf{E}}| \leq 3^{\#\mathsf{endpoints}_{\mathcal{A}}}|V|$ and $|V_{\mathsf{A}}| \leq 3^{\#\mathsf{endpoints}_{\mathcal{A}}}|V||\Sigma|$, yielding an overall exponential upper bound on $|\mathcal{K}_{\mathcal{A}}|$. Note that it is exponential in the number of endpoints, but not in the size of their encodings or their precise values.

3. Finally, when the sets of number of opponents are defined by intervals, a better upper bound can be obtained. Again observe that intersection of two intervals with endpoints in Y is again an interval with endpoints from the same set Y. All knowledge sets in $\mathcal{K}_{\mathcal{A}}$ are therefore intervals whose endpoints appear in the description of $\mathcal{A}$. There can be at most $\#\mathsf{endpoints}^2_{\mathcal{A}}$ such intervals, so that $|V_{\mathsf{E}}| \leq \#\mathsf{endpoints}^2_{\mathcal{A}}|V|$ and $|V_{\mathsf{A}}| \leq \#\mathsf{endpoints}^2_{\mathcal{A}}|V||\Sigma|$, yielding an overall polynomial upper bound on $|\mathcal{K}_{\mathcal{A}}|$.

The exponential upper bound is met by the family $(\mathcal{A}_n)_{n \in \mathbb{N}_{>0}}$ of *deterministic* semi-parameterized arenas depicted on Figure 5.3, and for which the constraints are unions

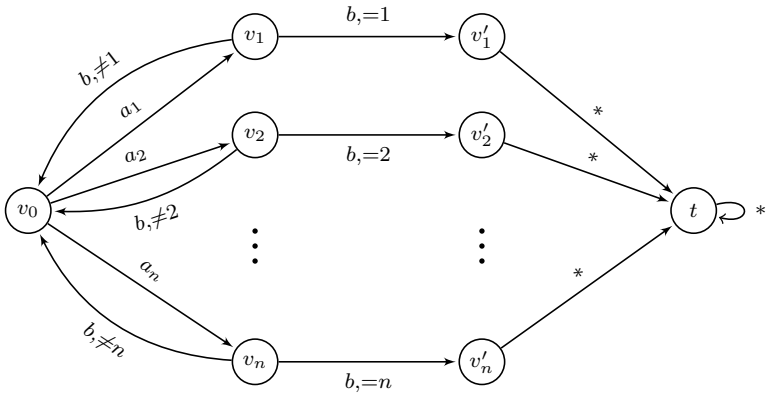


Figure 5.3: A deterministic semi-parameterized arena $\mathcal{A}_n$ ($n \in \mathbb{N}_{>0}$), whose size is polynomial in n and whose knowledge arena is exponential in n.

of disjoint intervals. Here a set $\{\neq i\}$, for instance, is a representation of the union of intervals $\{1\} \cup [3, \infty)$, and sets of the form $\{= i\}$ are self-explanatory. The sets here are a particular case of semilinear predicates. For example, a set $\{\neq i\}$ is ultimately periodic with threshold i and period 1; similarly for sets of the form $\{= i\}$.

As in earlier figures, all unspecified transitions go to a sink vertex $\perp$, which is not depicted here. In this example, from v_0 Eve can choose action a_k ($1 \leq k \leq n$) and go to v_k. From v_k, if she chooses action b, if there are exactly k opponents, the game moves to v'_k, otherwise comes back to v_0; from v'_k, for any choice of action of Eve, it moves to t. Here both the number of endpoints, and the number of predicates are linear in n. However, the associated knowledge arena $\mathcal{K}_{\mathcal{A}_n}$ contains vertices (v_0, K) for every non-empty subset K of $\{1, \ldots, n\}$. Indeed, from vertex (v_0, K) with $K \neq \emptyset$, for any $k \in K$, consider the successor vertex in two steps by a_k and b (to be precise, two steps in the semi-parameterized arena, but four steps in knowledge arena). Observe that in case the number of opponents is not k, that vertex is $(v_0, K \setminus \{k\})$. Such an example is depicted in Figure 5.4, a part of the knowledge game corresponding to the arena in Figure 5.3. In that figure, notice that from $(v_0, \mathbb{N}_{>0})$, after consecutive actions a_2 and b of Eve, it reaches vertex $(v_0, \neq 2)$; similarly for the vertex $(v_0, \neq 2)$ and for the actions a_5 and b. Finally, notice also that there is a vertex (v_k, K) for every $k \in K$ and every non-empty $K \subseteq \mathbb{N}_{>0}$. The number of possible subsets of $\{1, \ldots, n\}$ is exponential in n, and hence $|\mathcal{K}_{\mathcal{A}_n}| \in O(n2^n)$.

This concludes the proof of Lemma 5.14. □

We have constructed from a semi-parameterized arena the knowledge arena, a two-player turn-based game arena. The knowledge arena is finite, yet exponentially large in the size of input. In the following, we define a winning condition for Eve on the knowledge arena that reflects her winning objective in the semi-parameterized game in the sense that she has a winning strategy from v_0 in the original arena if and only if she has one from $(v_0, \mathbb{N}_{>0})$ in the knowledge arena. The knowledge set $\mathbb{N}_{>0}$ here represents that to win from a vertex in the semi-parameterized arena, she must have a strategy with no prior knowledge of her number of opponents. The *knowledge game* associated to a

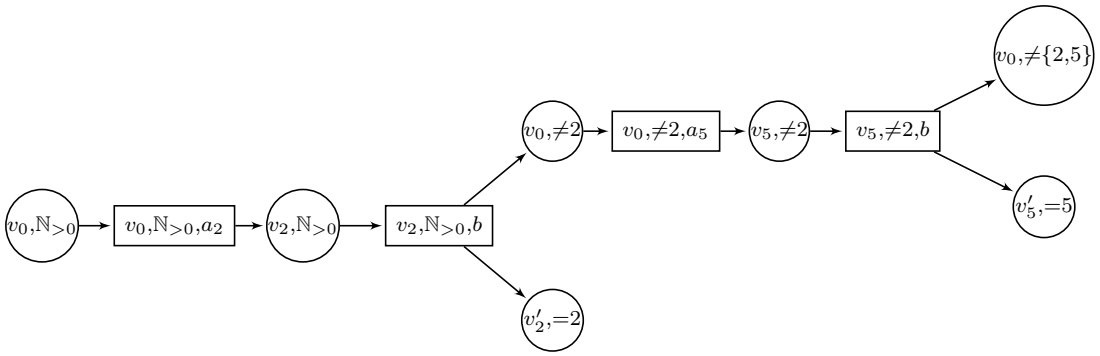


Figure 5.4: A part of the knowledge arena constructed from the arena $\mathcal{A}_n$ in Figure 5.3.

semi-parameterized game can be formally defined as follows.

Definition 5.15. *Let $\mathcal{G} = (\mathcal{A}, \mathsf{Win})$ be a semi-parameterized game with $\mathcal{A} = \langle V, \Sigma, \Delta \rangle$. Then the* knowledge game *associated with $\mathcal{G}$ is the two-player turn-based game $\mathcal{K}_\mathcal{G} = (\mathcal{K}_\mathcal{A}, \mathsf{Win}')$ such that $\mathcal{K}_\mathcal{A} = \langle V_\mathcal{K}, E_\mathcal{K} \rangle$ is the knowledge arena associated with $\mathcal{A}$, and the set Win' is defined as follows: $\mathsf{Win}' = \{\mathsf{v}_0^\mathsf{E}\mathsf{v}_0^\mathsf{A}\mathsf{v}_1^\mathsf{E}\mathsf{v}_1^\mathsf{A} \ldots \in \mathsf{Play}_\mathcal{K} \mid \exists v_0 v_1 \ldots \in \mathsf{Win}$ s.t. 1) $\mathsf{v}_0^\mathsf{E} = (v_0, \mathbb{N}_{>0})$, and 2) for each $i \geq 0$: $\mathsf{v}_i^\mathsf{E} = (v_i, K_i)$ for some $K_i \subseteq \mathbb{N}_{>0}\}$.*

A strategy λ for Eve in $\mathcal{K}_\mathcal{G}$ is winning from v_0 if all plays from v_0 induced by λ is in Win'. The winning region of Eve in $\mathcal{K}_\mathcal{G}$ is the set of vertices v_0 from which she has a winning strategy.

There is a one-to-one correspondence between the plays in $\mathcal{A}$ and $\mathcal{K}_\mathcal{A}$. Intuitively, a history hvv' in $\mathcal{A}$ has a corresponding history $\mathsf{h}\mathsf{v}^\mathsf{E}\mathsf{v}^\mathsf{A}\mathsf{v}'^\mathsf{E}$ in $\mathcal{K}_\mathcal{A}$ where v^E and v'^E are vertices of Eve with first component v and v', respectively, and v^A is a vertex of Adam, with first component v, that is a successor of v^E. Relying on this correspondence, one can translate winning strategies from a semi-parameterized game to its associated knowledge game. We formally show that below.

Lemma 5.16. Eve *has a winning strategy σ from v_0 in $\mathcal{G} = (\mathcal{A}, \mathsf{Win})$ if and only if she has a winning strategy λ from $(v_0, \mathbb{N}_{>0})$ in the associated knowledge game $\mathcal{K}_\mathcal{G} = (\mathcal{K}_\mathcal{A}, \mathsf{Win}')$.*

Proof. There is a correspondence between histories in $\mathcal{A}$ and $\mathcal{K}_\mathcal{A}$. Every history $h = v_0 v_1 \cdots v_i$ in $\mathcal{A}$ which is realizable can be lifted to a history in $\mathcal{K}_\mathcal{A}$ using the lifting function κ defined as $\kappa(h) = (v_0, K_0)(v_0, K_0, a_0)(v_1, K_1) \cdots (v_i, K_i)$ where: $K_0 = \mathbb{N}_{>0}$, and for every $1 \leq j \leq i$, $a_j \in \Sigma$ and $K_j = K_{j-1} \cap \Delta(v_{j-1}, a_{j-1}, v_j)$. Note that $\kappa(h)$ is well-defined because h is realizable and therefore, each K_j must be non-empty. Conversely, any history $\mathsf{h} = (v_0, K_0)(v_0, K_0, a_0)(v_1, K_1) \cdots (v_i, K_i)$ in $\mathcal{K}_\mathcal{A}$ projects to $\iota(\mathsf{h}) = v_0 v_1 \cdots v_i$ which is a history in $\mathcal{G}$. Moreover, for every $k \in K_i$, $\iota(\mathsf{h})$ is k-realizable in $\mathcal{A}$. Using κ and ι, one shows the correspondence between winning strategies in $\mathcal{G}$ and $\mathcal{K}_\mathcal{G}$ as follows.

Assume Eve has a winning strategy λ from $(v_0, \mathbb{N})$ in $\mathcal{K}_\mathcal{G}$. We define σ a strategy for Eve from v_0 in $\mathcal{G}$ by applying λ to the lifting of histories to $\mathcal{K}_\mathcal{G}$: for a history $h \in \mathsf{Hist}_\mathcal{A}$, we define $\sigma(h)$ as the third component of $\lambda(\kappa(h))$. To prove that σ is winning in $\mathcal{G}$ from

vertex v_0, let $k \in \mathbb{N}_{>0}$ be a number of opponents, and $\rho = v_0 v_1 \ldots \in \mathsf{Out}^k_{\mathcal{A}}(v_0, \sigma)$ a play in the k-outcome. Let $\kappa(\rho)$ be the lifting of ρ to $\mathcal{K}_{\mathcal{G}}$. By construction, $\kappa(\rho)$ is a play in $\mathcal{K}_{\mathcal{G}}$ induced by λ such that the second components of the vertices in $\kappa(\rho)$ contain k. Since λ is winning in $\mathcal{K}_{\mathcal{G}}$, $\kappa(\rho) \in \mathsf{Win}'$; and hence $\rho \in \mathsf{Win}$. Since this is true for all plays in $\mathsf{Out}^k_{\mathcal{A}}(v_0, \sigma)$ for every $k \in \mathbb{N}_{>0}$, σ is a winning strategy from v_0 for Eve in $\mathcal{G}$.

Assume now that Eve has a winning strategy σ from v_0 in $\mathcal{G}$. We define λ a strategy for her in $\mathcal{K}_{\mathcal{G}}$ by applying σ to the projection of histories to $\mathcal{G}$: for a history $\mathsf{h} \cdot (v, K) \in \mathsf{Hist}_{\mathcal{K}}$, define $\lambda(\mathsf{h} \cdot (v, K)) = (v, K, \sigma(\iota(H)))$. To prove that λ is winning in $\mathcal{K}_{\mathcal{G}}$, consider a play $R = (v_0, K_0)(v_0, K_0, a_0)(v_1, K_1) \cdots$ induced by λ in $\mathcal{K}_{\mathcal{G}}$. Let $\iota(R)$ be the projection of R to $\mathcal{G}$. By construction, for each $k \in \bigcap_{i \geq 0} K_i$, $\iota(R) \in \mathsf{Out}^k_{\mathcal{A}}(v_0, \sigma)$. Moreover, $\bigcap_i K_i \neq \emptyset$. Since σ is winning in $\mathcal{G}$, $\iota(R) \in \mathsf{Win}$; and hence $R \in \mathsf{Win}'$. Since this is true for all induced plays by λ, it is a winning strategy for Eve from $(v_0, \mathbb{N})$ in $\mathcal{K}_{\mathcal{G}}$. □

The above lemma establishes the correspondence between winning strategies in $\mathcal{G}$ and the associated knowledge game $\mathcal{K}_{\mathcal{G}}$. The definition of winning objective in the knowledge game (Definition 5.15) is such that Lemma 5.16 holds for any winning condition Win for Eve in $\mathcal{G}$. Moreover, for usual ω-regular objectives, Win' falls into the same class as Win. In particular,

- for Win a reachability objective described by the set of target vertices $F \subseteq V$, Win' is again a reachability condition described by the set $F' = V_{\mathsf{E}} \cap \{(v, K) \mid v \in F,\ K \subseteq \mathbb{N}_{>0}\}$. Indeed, it is easy to see that a play $R \in \mathsf{Win}'$ eventually reaches a vertex in F'. For the converse, consider a play R that eventually reaches F'. Then its projection $\iota(R)$ to $\mathcal{G}$ reaches a vertex in F, and hence belongs to Win, hence we conclude $R \in \mathsf{Win}'$.

- for Win a safety objective described by the set of safe vertices $F \subseteq V$, Win' is again a safety condition described by the set $F' = (V_{\mathsf{E}} \cap \{(v, K) \mid v \in F,\ K \subseteq \mathbb{N}_{>0}\}) \cup (V_{\mathsf{A}} \cap \{(v, K, a) \mid v \in F,\ K \subseteq \mathbb{N}_{>0},\ a \in \Sigma\})$. Indeed, it is easy to see that a play $R \in \mathsf{Win}'$ only visits vertices from the set F'. For the converse, consider a play R that only visits vertices in F'. Then its projection $\iota(R)$ to $\mathcal{G}$ only visits vertices from F, and hence belongs to Win, hence we conclude $R \in \mathsf{Win}'$.

- for Win a Büchi objective described by the set of vertices $F \subseteq V$, Win' is again a Büchi condition described by the set $F' = (V_{\mathsf{E}} \cap \{(v, K) \mid v \in F,\ K \subseteq \mathbb{N}_{>0}\})$. Indeed, it is easy to see that a play $R \in \mathsf{Win}'$ visits vertices in F' infinitely often. For the converse, consider a play R that visits vertices in F' infinitely often. Then its projection $\iota(R)$ to $\mathcal{G}$ visits vertices from F infinitely often, and hence belongs to Win, hence we conclude $R \in \mathsf{Win}'$.

Since the knowledge game is finite, and since one can effectively compute Eve's winning regions in a two-player turn-based game with the above winning conditions, we conclude that the semi-parameterized game problem is decidable for these classes of winning

objectives. Indeed, one can simply construct the associated knowledge game and decide if Eve has a winning strategy from $(v_0, \mathbb{N}_{>0})$.

Proposition 5.17. *The semi-parameterized game problem is decidable for reachability, safety and Büchi winning objectives.*

In the rest of the chapter, we consider Win a reachability objective for Eve and prove tight complexity bounds for the semi-parameterized game problem, unless otherwise specified. Recall that with no loss of generality, a reachability condition in the parameterized game $\mathcal{G}$ can be described by a single target vertex $t \in V$. In that case, Win' is described by the set of target vertices $F = V_{\mathsf{E}} \cap \{(t, K) \mid K \subseteq \mathbb{N}_{>0}\}$.

5.3 Tight bounds for reachability games

We now study the complexity of reachability objectives for semi-parameterized arenas. Let us first summarize the results we are going to prove in this section. The complexity of the semi-parameterized reachability game problem is stated in Table 5.1. Notice that the complexity may vary depending on the type of the constraints and also on whether the arena is deterministic or not.

		Deterministic arenas	Non-deterministic arenas
Constraints	Intervals	PTIME-complete	
	Finite unions of intervals	NP-complete	PSPACE-complete
	Semilinear sets	PSPACE-complete	

Table 5.1: Complexity of the semi-parameterized reachability game problem

Here note that the complexities for constraints given as (finite unions of) intervals are independent of values of endpoints used in the description of the input, but depend on their number. When constraints are given as semilinear sets, the complexity does depend on $\#\mathsf{pred}_{\mathcal{A}}$ as well as the size of the encodings of the semilinear sets.

Fix for the rest of the section a semi-parameterized reachability game $\mathcal{G} = (\mathcal{A}, t)$ with $\mathcal{A} = \langle V, \Sigma, \Delta \rangle$ and v_0 an initial vertex. Let $\mathcal{K}_{\mathcal{G}} = (\mathcal{K}_{\mathcal{A}}, F)$ be the knowledge game corresponding to $\mathcal{G}$ such that $\mathcal{K}_{\mathcal{A}} = \langle V_{\mathcal{K}}, E_{\mathcal{K}} \rangle$ with $V_{\mathcal{K}} = V_{\mathsf{E}} \uplus V_{\mathsf{A}}$. The rest of this section is devoted to proving these complexity results. We start with the simple case of intervals.

5.3.1 Constraints as intervals

When the constraints defining the sets of number of opponents are given as intervals, the semi-parameterized reachability problem is complete for the complexity class PTIME.

Proposition 5.18. *When constraints are intervals, the semi-parameterized reachability game problem is* PTIME*-complete.*

Proof. When constraints are intervals only, the size of the knowledge arena $\mathcal{K}_{\mathcal{A}}$ is polynomial in the size of the semi-parameterized arena $\mathcal{A}$ (see Lemma 5.14) and it can be computed in polynomial time. Further, the two-player reachability game on the knowledge arena can be solved in linear time in the size of $\mathcal{K}_{\mathcal{A}}$ (see Theorem 4.12). Hence, the semi-parameterized reachability game problem is in PTIME.

It is moreover complete for this class, since two-player reachability games are PTIME-hard (by a reduction from the CIRCUIT-SAT problem), and the knowledge games are generalizations of the same. We thus obtain the above complexity result, independently of whether the arena is deterministic or not. □

Next, we study the case of constraints given as finite unions of intervals or as semilinear sets.

5.3.2 General PSPACE upper bound

We show that when constraints are given as semilinear predicates, or in non-deterministic arenas with constraints given as finite unions of intervals, the reachability problem is in PSPACE. The result is not straightforward since from Lemma 5.14, the knowledge arena can be exponential. To show a PSPACE upper bound, we first break the knowledge game into smaller (polynomial size) subgames, solve those subgames in polynomial time, and come up with a method (based on depth-first search) to aggregate the results of those subgames to decide the reachability problem on the knowledge arena using only polynomial space in $|\mathcal{G}|$. We first describe the construction of the subgame arenas, followed by the method to combine the outputs of reachability game problems on those sub-arenas to decide the reachability problem on the knowledge game and then prove the correctness of this method. We finally show that the above procedure runs in polynomial space in $|\mathcal{G}|$.

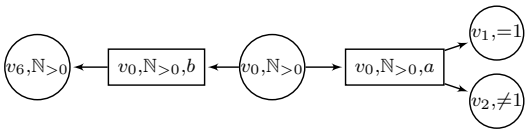
Subgame arenas $\mathcal{K}_{\mathcal{A}}[v, K]$

For each vertex $(v, K) \in V_{\mathsf{E}}$ of Eve in $\mathcal{K}_{\mathcal{G}}$, we define a game arena $\mathcal{K}_{\mathcal{A}}[v, K]$, which is the restriction of $\mathcal{K}_{\mathcal{A}}$ to vertices (v', K, a) and (v', K') that are reachable from (v, K) via vertices with same knowledge set K only. Formally, $\mathcal{K}_{\mathcal{A}}[v, K]$ is the restriction of $\mathcal{K}_{\mathcal{A}}$ to

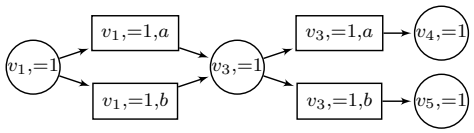
the following sets of vertices, defined inductively:

$$\begin{cases} V_{\mathsf{E}}^0 = \{(v, K)\} \\ V_{\mathsf{A}}^i = \{(v', K, a) \mid v' \neq t \text{ and } (v', K) \in V_{\mathsf{E}}^i \text{ and } \big((v', K), (v', K, a)\big) \in E_{\mathcal{K}}\} \\ V_{\mathsf{E}}^{i+1} = \{(v', K') \mid \exists (v'', K, a) \in V_{\mathsf{A}}^i \text{ s.t. } \big((v'', K, a), (v', K')\big) \in E_{\mathcal{K}}\} \end{cases}$$

In this definition, as usual, sets of vertices with a suffix E belong to Eve and others to Adam. Notice that in $\mathcal{K}_{\mathcal{A}}[v, K]$, every vertex of Adam has knowledge set K. Also, vertices (v', K') of Eve with knowledge $K' \subsetneq K$ or with $v' = t$ have no successors: we refer to them as the *output vertices of* $\mathcal{K}_{\mathcal{A}}[v, K]$, and we write $O^{[v,K]}$ for the set of such vertices. However, to be consistent with the notion of a two-player turn-based game – in which there are no dead-ends – with no loss of generality, we put self-loops on the vertices in $O^{[v,K]}$.



(a) A part of the subgame arena $\mathcal{K}_{\mathcal{A}}[v_0, \mathbb{N}_{>0}]$ corresponding to the knowledge arena in Figure 5.2.



(b) A part of the subgame arena $\mathcal{K}_{\mathcal{A}}[v_1, \{1\}]$ corresponding to the knowledge arena in Figure 5.2.

Figure 5.5: Examples of subgame arenas.

Examples of subgame arenas are given in Figure 5.5. On the left, a part of the subgame arena corresponding to the vertex $(v_0, \mathbb{N}_{>0})$ of the knowledge arena of Figure 5.2 is presented; and on the right, a part of the subgame arena $\mathcal{K}_{\mathcal{A}}[v_1, \{1\}]$ from the same knowledge arena is presented (outgoing vertices from $(v_6, \mathbb{N}_{>0})$ and $(v_5, \{1\})$ are not shown, respectively, in the pictures).

We prove the following lemma analysing the size of this sub-arenas.

Lemma 5.19. *Each subgame arena $\mathcal{K}_{\mathcal{A}}[v, K]$ is polynomial in the size of $\mathcal{K}_{\mathcal{A}}$. Furthermore, one can compute each $\mathcal{K}_{\mathcal{A}}[v, K]$*

1. *in polynomial time in $\#\mathsf{endpoints}_{\mathcal{A}}$ for constraints defined by finite unions of intervals; and*
2. *in polynomial space in $\#\mathsf{pred}_{\mathcal{A}}$ and size of their encodings for constraints defined by semilinear predicates.*

Proof. First notice that there are at most $|V|$ many vertices of Eve ((v, K) for every $v \in V$), and $|\Sigma||V|$ many vertices of Adam ((v, a, K) for every $v \in V$, $a \in \Sigma$) in $\mathcal{K}_{\mathcal{A}}[v, K]$ with second component exactly K. Since vertices of Eve with second component strictly smaller than K have no successors, only the immediate successors of v in $\mathcal{A}$ such that the knowledge of Eve refines are considered in $\mathcal{K}_{\mathcal{A}}[v, K]$, and there are at most $|E||V| \leq |V|^3$ ($|E|$ is the number of edges in $\mathcal{A}$) many such vertices of Eve. There is no vertex of

Adam with knowledge set different from K. Therefore, the size of the arena $\mathcal{K}_{\mathcal{A}}[v, K]$ is polynomial in the size of $\mathcal{K}_{\mathcal{A}}$.

1. The intersection of two unions of disjoint intervals with endpoints from a set S is again a finite union of disjoint intervals with endpoints from S. One can compute the intersection by scanning the endpoints of the input sets in an increasing order. Therefore, it can be computed in polynomial time in $\#\mathsf{endpoints}_{\mathcal{A}}$.

2. For semilinear sets, $\mathcal{K}_{\mathcal{A}}[v, K]$ can be computed in polynomial space in the size of the encodings of the predicates. In this case, for P a semilinear predicate from the arena and K a knowledge set one needs to check whether $(P \cap K) \neq \emptyset$ and also if $(P \cap K) \subsetneq K$ (to decide whether one obtains an output vertex of $\mathcal{K}_{\mathcal{A}}[v, K]$). First notice that semilinear sets are closed under intersection and moreover, one can effectively compute the intersection of two semilinear sets, see for instance [GS64, Theorem 6.1]. It is well-known from [Huy82] that the inequality problem for semilinear sets is complete for the complexity class Σ_2^P in the polynomial hierarchy. For the non-emptiness checking, one can reduce it to the inequality problem. Indeed, $P \cap K = \emptyset$ iff $P \cap \overline{K} = P$, where $\overline{K}$ denotes the complement of set K. Therefore, both of the above checks can be done in polynomial space in $\#\mathsf{pred}_{\mathcal{A}}$ and the size of their encodings.

This concludes the proof of Lemma 5.19. □

One can consider a winning objective for Eve on a sub-arena $\mathcal{K}_{\mathcal{A}}[v, K]$ to form a two-player turn-based game $\mathcal{K}_{\mathcal{G}}[v, K]$. We will consider reachability objectives for Eve in $\mathcal{K}_{\mathcal{G}}[v, K]$. Because of its polynomial size, once constructed, $\mathcal{K}_{\mathcal{G}}[v, K]$ with a reachability objective can be solved in polynomial time in $|\mathcal{G}|$, see Theorem 4.12. We use these games as sub-routines for solving the semi-parameterized reachability game problem. Let us now describe that in the following.

Tagged tree of subgames

Using the subgames $\mathcal{K}_{\mathcal{G}}[v, K]$, we consider the following exponential-size tree $\mathcal{T}$. The root of the tree is $(v_0, \mathbb{N}_{>0})$, the initial vertex of the knowledge game. A node (v, K) has children the vertices in $O^{[v,K]}$. Our aim is then to tag each node $\mathsf{n} = (v, K)$ of $\mathcal{T}$ with `Win` or `Lose`, to reflect whether Eve has a winning strategy from (v, K) in $\mathcal{K}_{\mathcal{G}}$. We first formally define the construction of $\mathcal{T}$ followed by a tag function.

$\mathcal{T}$ is defined inductively as follows: the root $\mathsf{n}_0 = (v_0, \mathbb{N}_{>0})$ is the initial vertex of $\mathcal{K}_{\mathcal{A}}$, and (v', K') is a child of (v, K) if $(v', K') \in O^{[v,K]}$ is an output vertex in $\mathcal{K}_{\mathcal{A}}[v, K]$. We then define the following tagging function on the vertices of $\mathcal{T}$. Intuitively, a node is tagged `Win` if its first component is t or if Eve has a strategy in the subgame $\mathcal{K}_{\mathcal{G}}[v, K]$ to reach an output vertex in $O^{[v,K]}$ that has been tagged `Win`.

$$\mathsf{tag}((v,K)) = \begin{cases} \texttt{Win} & \text{if } v = t \\ \texttt{Win} & \text{if } \mathsf{Eve} \text{ has a winning strategy in } \mathcal{K}_{\mathcal{G}}[v,K] \text{ from } (v,K) \text{ to reach} \\ & \text{the set } \{\alpha \in O^{[v,K]} \mid \mathsf{tag}(\alpha) = \texttt{Win}\}, \text{ if non-empty} \\ \texttt{Lose} & \text{otherwise.} \end{cases}$$

We now show the correctness of the tagging function in the sense that a node (v,K) in $\mathcal{T}$ is tagged `Win` if and only if Eve has a winning strategy in $\mathcal{K}_{\mathcal{G}}$ from (v,K).

Lemma 5.20. *For each node (v,K) in $\mathcal{T}$, $\mathsf{tag}((v,K)) = \texttt{Win}$ if and only if Eve has a winning strategy in $\mathcal{K}_{\mathcal{G}}$ from (v,K).*

Proof. We show by induction on the height of the tree that for every node (v,K) in $\mathcal{T}$, $\mathsf{tag}((v,K)) = \texttt{Win}$ if and only if the corresponding vertex (v,K) in $\mathcal{K}_{\mathcal{G}}$ is in Eve's winning region. First pick a leaf (v,K): if $v = t$, it is in winning region for Eve in $\mathcal{K}_{\mathcal{G}}$, and by definition, it is tagged `Win` in $\mathcal{T}$; otherwise, since $\mathcal{K}_{\mathcal{A}}[v,K]$ has no output vertex, the game never exits that subgame and hence is not winning for Eve, and by definition, it is tagged `Lose` in $\mathcal{T}$.

Now pick an intermediary node (v,K), and by induction hypothesis, assume all the children of that node are already tagged and satisfy the claim. We show it also holds for (v,K). Let $\mathsf{tag}((v,K)) = \texttt{Win}$. Then there is a winning strategy of Eve in $\mathcal{K}_{\mathcal{G}}[v,K]$ to reach the set S of output vertices which are tagged `Win`, and by induction hypothesis, she has a winning strategy from each of those vertices in $\mathcal{K}_{\mathcal{G}}$. Therefore, she also has a winning strategy from (v,K) in $\mathcal{K}_{\mathcal{G}}$: it first reaches a vertex in S and follows the winning strategy from that vertex.

Conversely, assume there is a winning strategy σ from (v,K) in $\mathcal{K}_{\mathcal{G}}$; one can play it in $\mathcal{K}_{\mathcal{G}}[v,K]$ as well. Since σ is winning in $\mathcal{K}_{\mathcal{G}}$, it eventually reaches (t,K'') for some K'', hence when we apply it in $\mathcal{K}_{\mathcal{G}}[v,K]$, output vertices in $O^{[v,K]}$ are reached eventually. Towards a contradiction, assume it reaches one such output vertex (v',K') such that $\mathsf{tag}((v',K')) = \texttt{Lose}$. Then by the induction hypothesis, there should not be any winning strategy from (v',K') in $\mathcal{K}_{\mathcal{G}}$; this is a contradiction since (v',K') is reached by applying σ which is a winning strategy from (v,K) in $\mathcal{K}_{\mathcal{G}}$. We conclude that $\mathsf{tag}((v,K)) = \texttt{Win}$ if and only if Eve has a winning strategy in $\mathcal{K}_{\mathcal{G}}$ from (v,K). □

As a direct corollary of the above lemma, we conclude that the root of $\mathcal{T}$ is tagged `Win` if and only if Eve has a winning strategy in $\mathcal{K}_{\mathcal{G}}$ from $(v_0,\mathbb{N}_{>0})$, which is equivalent to saying she has a winning strategy in $\mathcal{G}$ to reach the target vertex t, by Lemma 5.16.

We now show the root of the tree can be tagged in polynomial space, by a depth-first search algorithm on $\mathcal{T}$. An illustration of the algorithm is given in Figure 5.6. The intuitive idea is that once the tag of a node has been computed, its whole subtree can be forgotten and one can reuse the space to repeatedly solve the games $\mathcal{K}_{\mathcal{G}}[v,K]$ for different

v and K. We show that polynomial space is sufficient to compute the value of the tag function for the root node in $\mathcal{T}$.

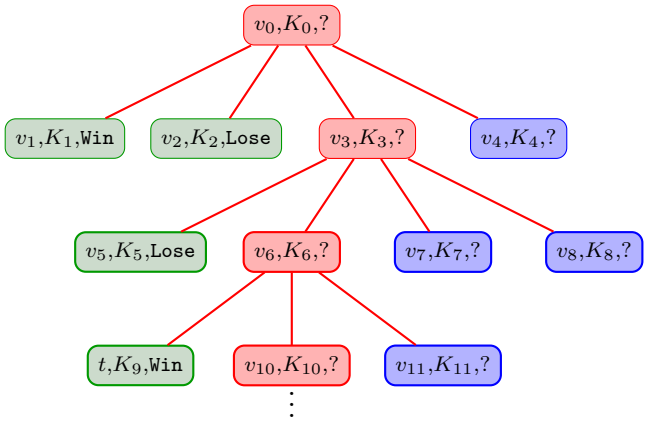


Figure 5.6: Illustration of the polynomial space DFS tagging algorithm: the `Win`/`Lose` tags of green nodes have already been computed (and their subtrees have been removed); the tags of red nodes are being computed (hence the label '?'); and the blue nodes are waiting to be processed (we also use label '?'). For instance, before tagging (v_6, K_6), one needs to first compute the tag of (v_{10}, K_{10}) (which is ongoing), then compute the tag of (v_{11}, K_{11}) (which is waiting).

Lemma 5.21. *One can determine* $\mathsf{tag}(\mathsf{n}_0) = \mathsf{tag}((v_0, \mathbb{N}_{>0}))$ *in polynomial space in* $|\mathcal{G}|$.

Proof. First recall that assuming all its children nodes are already tagged, the tag of a node $\mathsf{n} = (v, K)$ can be determined in polynomial time by solving the polynomial size game $\mathcal{K}_{\mathcal{G}}[v, K]$.

Storing the full tagged tree $\mathcal{T}$ may require exponential space, however the height of $\mathcal{T}$, denoted $h(\mathcal{T})$ is polynomially bounded, in $\#\mathsf{endpoints}_{\mathcal{A}}$ in the case of finite unions of intervals, and in $\#\mathsf{pred}_{\mathcal{A}}$ in the case of semilinear predicates. Indeed, along a branch the knowledge sets decrease, and we only take intersections of constraints given in the input. A polynomial space algorithm thus consists in a depth-first search tagging. Once the tag of a node has been computed, its whole subtree is forgotten. In the DFS tagging, the size of the stack is at most the height of the tree times the maximal number of successors of a vertex v in $\mathcal{G}$. That is, the size of the stack is at most $|V|h(\mathcal{T})$, which is polynomial in $|\mathcal{G}|$. Now, to store one node of the tree, polynomial space is sufficient, since the only critical part is the knowledge set. In both cases (unions of intervals and general predicates), a knowledge set can be stored in polynomial space (even linear for unions of intervals):

- for constraints given by finite unions of intervals, it is sufficient to store a list of left endpoints and right endpoints, ordered properly. Since each endpoint is used in $\mathcal{G}$, this list can be stored in space linearly bounded by $|\mathcal{G}|$.
- for constraints given by semilinear predicates, a knowledge set is characterized by the list of predicates which are satisfied, hence if $\mathcal{P}$ is the finite set of predicates

$\mathcal{P} = \{P_1, \ldots, P_l\}$ in $\mathcal{A}$, it is sufficient to store a subset of $\{1, \ldots, l\}$, which can be done in space $l \log(l)$, with $l \leq |\mathcal{G}|$.

Hence, $\mathsf{tag}(\mathsf{n}_0)$ can be determined in polynomial space in $|\mathcal{G}|$. □

Lemma 5.21 proves that one can compute $\mathsf{tag}(\mathsf{n}_0)$ in polynomial space in the size of the input. Then, by the correctness of the tagged tree (see Lemma 5.20), and the correctness of the knowledge game construction (see Lemma 5.16), we conclude that the semi-parameterized reachability game can be solved in polynomial space:

Proposition 5.22. *When constraints are given by finite unions of disjoint intervals or semilinear sets, the semi-parameterized reachability game problem is in* PSPACE.

Before proving the lower bound results, let us show in the following that the problem falls into a lower complexity class for deterministic arenas when constraints are given by finite union of intervals.

5.3.3 An NP upper bound for deterministic arenas for constraints given by finite unions of intervals

The previous PSPACE upper bound can be improved when the arena is deterministic and constraints are given by finite unions of intervals. We show that if there is a winning strategy for Eve in $\mathcal{G}$ then there is one which is polynomially bounded in $|\mathcal{G}|$. Therefore, one can guess a "small" strategy for Eve and check in polynomial time if it is winning for her. We detail this idea below.

We assume that Eve has a winning strategy in the semi-parameterized reachability game $\mathcal{G}$. We pick σ, an arbitrary one that is winning, and we consider the tree $\mathcal{T}_\sigma$ it induces: nodes are histories, and the children of a node are the possible next histories (depending on the number of opponents); we label each of the nodes with the knowledge of Eve w.r.t. the number of opponents.

Formally, the root of $\mathcal{T}_\sigma$ is the history v_0, and it is labelled with knowledge $\mathbb{N}_{>0}$. Consider a node $hv \in \mathsf{Hist}_\mathcal{A}$ in $\mathcal{T}_\sigma$ labelled with knowledge K; its children will be all hvv', where $v' \in V$ is such that there is K' with $K \cap K' \neq \emptyset$ and $(v, \sigma(hv), K', v') \in \Delta$; such a node hvv' will then be labelled with $K \cap K'$.

The tree $\mathcal{T}_\sigma$ for a winning strategy σ satisfies the following properties:

Lemma 5.23. *Let σ be a winning strategy for* Eve *from v_0 in $\mathcal{G}$ and $\mathcal{T}_\sigma$ be the corresponding strategy tree. Then:*

1. *along any path in $\mathcal{T}_\sigma$, the number of distinct knowledge sets is at most $\#\mathsf{endpoints}_\mathcal{A}$;*

2. the knowledge at sibling nodes form a partition of the knowledge at their parent node.

Proof. 1. The knowledge sets labelling the nodes of the tree are intersections of finite unions of intervals from $\mathcal{A}$. More precisely, a node $v_0 v_1 \dots v_t$ of $\mathcal{T}_\sigma$ has label the set $\bigcap_{i=0}^{t-1} \Delta(v_i, a_i, v_{i+1})$ where $a_i = \sigma(v_0 \dots v_i)$. In particular, all intervals are defined using endpoints of intervals appearing in a constraint of $\mathcal{A}$ (if $[i_1, i_2]$ is an interval of the above set, then i_1 is a left endpoint, and i_2 is a right endpoint of some interval in $\mathcal{A}$).

2. To prove the second property, it is sufficient to notice that the arena is deterministic and complete w.r.t. Eve's actions. More precisely, for any $k \in \mathbb{N}_{>0}$ number of opponents, from a history hv in $\mathcal{G}$, the action $\sigma(hv)$ of Eve leads to a *unique* v' such that $k \in \Delta(v, a, v')$. Hence, the knowledge sets attached to all successors hvv' form a partition of the knowledge attached to node hv.

This concludes the proof of Lemma 5.23. □

The second property has the following consequences, that we use in the sequel. First, at each level of the tree, the knowledge of all nodes form a partition of $\mathbb{N}_{>0}$ using endpoints from the arena description, so that the number of nodes at each level is bounded by $\#\mathsf{endpoints}_{\mathcal{A}}$. Second, if a node has the same knowledge as its parent, it cannot have siblings.

These properties allow us to transform an arbitrary winning strategy of Eve in $\mathcal{G}$ into one whose tree is "small", more precisely, polynomially bounded by the size of the arena:

Lemma 5.24. *Eve has a winning strategy in $\mathcal{G}$ if and only if she has a winning strategy σ such that $\mathcal{T}_\sigma$ is of height at most $\#\mathsf{endpoints}_{\mathcal{A}}|V|$, and of width at most $\#\mathsf{endpoints}_{\mathcal{A}}$.*

Proof. From Lemma 5.23, the tree induced by an arbitrary winning strategy σ' has width bounded by $\#\mathsf{endpoints}_{\mathcal{A}}$. We now explain how to transform the strategy such that the height is bounded by $\#\mathsf{endpoints}_{\mathcal{A}}|V|$. Assume there is a path in $\mathcal{T}_{\sigma'}$ of length more than $\#\mathsf{endpoints}_{\mathcal{A}}|V|$. By Lemma 5.23, there must be two nodes along that path whose histories end with the same vertex, and labelled with the same knowledge. Moreover, there is no branching on the path portion between these two nodes, because the knowledge is unchanged. It thus suffices to shorten the path by removing the corresponding path portion. Applying this reduction iteratively, one obtains a strategy tree of height bounded by $\#\mathsf{endpoints}_{\mathcal{A}}|V|$. This tree can be transformed into a strategy in a straightforward way. □

Using Lemma 5.24, we derive an algorithm in non-deterministic polynomial time to decide the semi-parameterized reachability game problem when arenas are deterministic and constraints are finite unions of intervals. It suffices to guess a small strategy tree (see Figure 5.7), of size polynomial in the size of $\mathcal{A}$, and check in polynomial time that it is consistent with the arena and satisfies the reachability criterion.

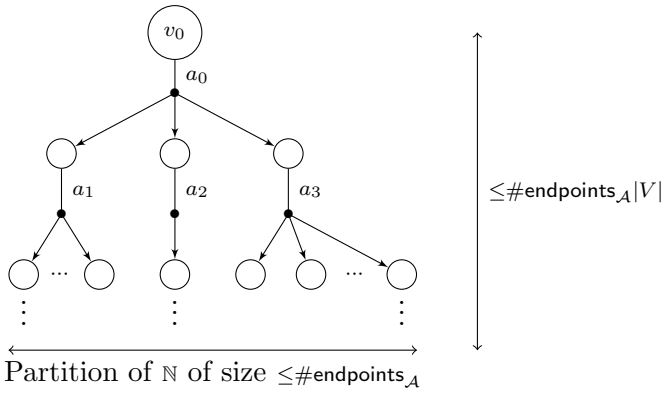


Figure 5.7: Strategy tree of a small winning strategy in deterministic arenas with finite unions of intervals.

We derive the following complexity result.

Theorem 5.25. *The parameterized reachability game problem is in* NP*, when constraints are finite unions of intervals and when restricting to deterministic arenas.*

The rest of this section is devoted to proving lower bounds for the reachability problem on semi-parameterized arenas. We shall show a PSPACE lower bound for the general constraints and an NP lower bound for deterministic arenas with constraints finite unions of intervals. This proves that the bounds proved so far are tight for the respective complexity classes.

5.3.4 Lower bounds

We prove all lower bounds mentioned in Table 5.1. We start with the PSPACE-hardness when constraints are finite unions of intervals and arenas are *a priori* non-deterministic.

Proposition 5.26. *When constraints are finite unions of intervals, the semi-parameterized reachability game problem is* PSPACE*-hard.*

Proof. The proof is by reduction from QBF-SAT, which is known to be PSPACE-complete from [SM73]. Let $\varphi = \exists x_1 \forall x_2 \exists x_3 \ldots \forall x_{2r} \cdot \big(C_1 \wedge C_2 \wedge \ldots \wedge C_m\big)$ be a quantified Boolean formula in prenex normal form, where for every $1 \leq h \leq m$, $C_h = \ell_{h,1} \vee \ell_{h,2} \vee \ell_{h,3}$ are the clauses, and for every $1 \leq j \leq 3$, $\ell_{h,j} \in \{x_i, \neg x_i \mid 1 \leq i \leq 2r\}$ are the literals. From φ, we construct a semi-parameterized arena $\mathcal{A}_\varphi = \langle V, \Sigma, \Delta \rangle$ as follows:

- $V = \{v_0, v_1, \ldots, v_{2r-1}, v_{2r}\} \cup \{v_{x_1}, v_{\bar{x}_1}, \ldots, v_{x_{2r}}, v_{\bar{x}_{2r}}\} \cup \{v_{C_1}, v_{C_2}, \ldots, v_{C_m}, v_{C_{m+1}}\} \cup \{\bot, \top\}$, where we identify v_{2r} with v_{C_1}, and $v_{C_{m+1}}$ with $\top$.
- $\Sigma = \{u, c\} \cup \bigcup_{1 \leq i \leq 2r} \{a_i, \bar{a}_i\}$.

- For every $0 \leq s \leq r-1$, $1 \leq i \leq 2r$, $1 \leq h \leq m$ and $1 \leq j \leq 3$:
  1. $\Delta(v_{2s}, a_{2s+1}, v_{x_{2s+1}}) = \Delta(v_{2s}, \bar{a}_{2s+1}, v_{\bar{x}_{2s+1}}) = \mathbb{N}_{>0}$;
  2. $\Delta(v_{2s+1}, u, v_{x_{2s+2}}) = \Delta(v_{2s+1}, u, v_{\bar{x}_{2s+2}}) = \mathbb{N}_{>0}$;
  3. $\Delta(v_{x_i}, c, v_i) = \mathbb{N}_{>0} \setminus \{2i\}$ and $\Delta(v_{x_i}, c, \top) = \{2i\}$;
  4. $\Delta(v_{\bar{x}_i}, c, v_i) = \mathbb{N}_{>0} \setminus \{2i-1\}$ and $\Delta(v_{\bar{x}_i}, c, \top) = \{2i-1\}$;
  5. $\Delta(v_{C_h}, a_i, v_{C_{h+1}}) = \mathbb{N}_{>0} \setminus \{2i\}$ if $\ell_{h,j} = x_i$; and
     $\Delta(v_{C_h}, \bar{a}_i, v_{C_{h+1}}) = \mathbb{N}_{>0} \setminus \{2i-1\}$ if $\ell_{h,j} = \neg x_i$.

  To obtain a complete arena, all unspecified transitions lead to a sink vertex $\bot$.

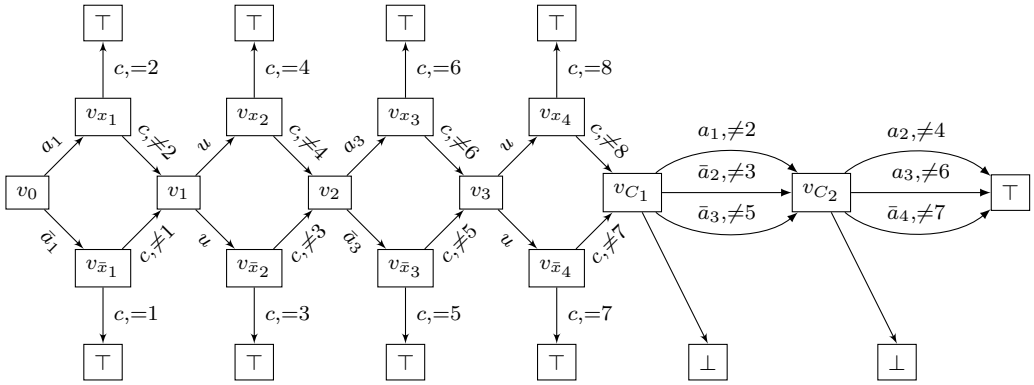


Figure 5.8: Reduction for formula $\varphi = \exists x_1 \forall x_2 \exists x_3 \forall x_4 \cdot (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_2 \vee x_3 \vee \neg x_4)$. Knowledge of Eve at v_{C_1} contains for every variable x_i, either $2i$ or $2i-1$ (and not both); containing $2i$ (resp., $2i-1$) encodes that x_i has been set to false (resp., true).

An example of the construction is given in Figure 5.8 for the formula $\varphi = \exists x_1 \forall x_2 \exists x_3 \forall x_4 \cdot (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_2 \vee x_3 \vee \neg x_4)$. The nodes are depicted in square for a better representation of the picture. Constraints of the form $\mathbb{N}_{>0} \setminus \{2i\}$ are presented as $\neq 2i$, for instance, $\Delta(v_{x_1}, c, v_1) = \mathbb{N}_{>0} \setminus \{2\}$.

Intuitively, from v_0, a first phase consists in choosing a valuation for the variables: Eve can choose the truth values of existentially quantified variables in vertices v_{2s} (with actions a_{2s+1} for true and $\bar{a}_{2s+1}$ for false), and the environment resolves the non-determinism of action u (u stands for universal) to choose the truth values of universally quantified variables in vertices v_{2s-1}. Due to the constraints on the edges, the knowledge of Eve at v_{C_1} contains for every variable x_i, either $2i$ or $2i-1$ (and not both); where containing $2i$ (resp., $2i-1$) encodes the fact that x_i has been set to false (resp., true) by Eve or the environment.

From v_{C_1}, a second phase starts, where one checks whether the generated valuation makes all clauses in φ true. Sequentially, Eve chooses for every clause a literal that makes the clause true and these choices must be consistent with the first phase. To enforce this, plays with $2i-1$ and $2i$ opponents check the consistency of the assignment for variable x_i. For instance, if action a_i (encoding x_i set to true) against $2i-1$ opponents leads from v_{C_h}

to $v_{C_{h+1}}$, this means that v_{x_i} was visited, hence that x_i was set to true. On the contrary, if x_i was set to false, hence v_{x_i} was not visited, then against $2i{-}1$ opponents, action a_i will lead to $\bot$. The role of $\bar{a}_i$ is dual; it encodes assigning false to x_i, and will be checked with plays against $2i$ opponents.

Let us now formally show that the above reduction ensures the following equivalence: Eve has a winning strategy from v_0 in the semi-parameterized reachability game $\mathcal{G}_\varphi = (\mathcal{A}_\varphi, \top)$ if and only if φ is true.

Here, for $\ell_{h,j}$ a literal of φ, we write $\gamma(\ell_{h,j})$ for the constraint associated with $\ell_{h,j}$ in the second phase of the reduction. More precisely,

$$\gamma(\ell_{h,j}) = \begin{cases} \neq 2i & \text{if } \ell_{h,j} = x_i \\ \neq 2i{-}1 & \text{if } \ell_{h,j} = \neg x_i \end{cases}$$

We rely on the following correspondence between plays from v_0 to v_{C_1} and valuations over $\{x_1, \cdots, x_{2r}\}$: the valuation ι_π associated with a history $\pi = v_0 v_1' v_1 v_2' v_2 \cdots v_{C_1}$ is such that $\iota_\pi(x_i) = 1$ if $v_i' = v_{x_i}$ and $\iota_\pi(x_i) = 0$ if $v_i' = v_{\bar{x}_i}$. This correspondence defines a bijection, so that, given a valuation ι over $\{x_1, \cdots, x_{2r}\}$, there is a unique history $\pi_\iota = v_0 v_1' v_1 v_2' v_2 \cdots v_{C_1}$ with $v_i' = v_{x_i}$ if $\iota(x_i) = 1$, and $v_i' = v_{\bar{x}_i}$ if $\iota(x_i) = 0$.

Moreover, after a finite history $\pi = v_0 v_1' v_1 v_2' v_2 \cdots v_{C_1}$, the information Eve has about the number of her opponents is $K_\pi = \{2i{-}\iota_\pi(x_i) \mid 1 \leq i \leq 2r\} \cup \{k \mid k \geq 4r{+}1\}$. Indeed:

$$\begin{array}{rcl} K_\pi & = & \mathbb{N} \setminus \big(\{2i{-}1 \mid v_i' = v_{\bar{x}_i}, 1 \leq i \leq 2r\} \cup \{2i \mid v_i' = v_{x_i}, 1 \leq i \leq 2r\}\big) \\ & = & \mathbb{N} \setminus \Big(\{2i{-}1 \mid \iota_\pi(x_i) = 0, 1 \leq i \leq 2r\} \cup \{2i \mid \iota_\pi(x_i) = 1, 1 \leq i \leq 2r\}\Big) \\ & = & \{2i \mid \iota_\pi(x_i) = 0, 1 \leq i \leq 2r\} \cup \{2i - 1 \mid \iota_\pi(x_i) = 1, 1 \leq i \leq 2r\} \cup \{k \mid k \geq 4r{+}1\} \\ & = & \{2i{-}\iota_\pi(x_i) \mid 1 \leq i \leq 2r\} \cup \{k \mid k \geq 4r{+}1\} \end{array}$$

There is thus also a bijection between valuations and possible knowledges at v_{C_1}, so that we abusively write K_ι when ι is a fixed valuation over $\{x_1, \cdots, x_{2r}\}$. Note that for every $1 \leq i \leq 2r$, $K_\iota \cap \{2i{-}1, 2i\}$ is a singleton: $\{2i{-}1\}$ if $\iota(x_i) = 1$ and $\{2i\}$ if $\iota(x_i) = 0$. Also, an extension of a history $\pi = v_0 v_1' v_1 v_2' v_2 \cdots v_{C_1}$ either leads to $\bot$, or continues in the main part of the game, in which case it does not refine the knowledge set K_π further. We then show the following equivalence for the second phase of the reduction:

Lemma 5.27. *Let ι be a valuation of $\{x_1, \cdots, x_{2r}\}$. Then $\iota \models C_1 \wedge \cdots \wedge C_m$ if and only if* Eve *has a strategy that ensures winning from v_{C_1} against k opponents, for all $k \in K_\iota$.*

Proof. Assume $\iota \models C_1 \wedge \cdots \wedge C_m$. For every $1 \leq h \leq m$, there is $\ell_{h,j}$ (literal of C_h) such that $\iota(\ell_{h,j}) = 1$. We define the following strategy for Eve from v_{C_1}: at vertex v_{C_h}, she plays action α_h defined by $\alpha_h = a_i$ if $\ell_{h,j} = x_i$, and $\alpha_h = \bar{a}_i$ if $\ell_{h,j} = \neg x_i$. By property on K_ι, $K_\iota \subseteq \gamma(\ell_{h,j})$. Thus, the strategy avoids $\bot$, and is therefore winning for every $k \in K_\iota$.

Conversely, pick a winning strategy for Eve from v_{C_1} against K_ι. This strategy plays uniformly for all $k \in K_\iota$, say action α_h from v_{C_h}. If $\alpha_h = a_i$, then x_i is a literal of C_h,

which is such that $K_\iota \subseteq \gamma(x_i)$; the construction guarantees that $\iota(x_i) = 1$. A similar reasoning applies to the case $\alpha_h = \bar{a}_i$. In all cases, C_h is satisfied by ι. □

Lemma 5.27 formalizes the link between winning strategies for Eve in the second phase (that is, from v_{C_1}) and satisfying valuations. It remains to relate strategies for Eve in the first phase and decision functions for the variable quantifications in φ.

The two-player game underlying the quantifications of φ coincides with the game in the first phase. In particular, strategies coincide in the two games. Fix a strategy σ in the quantification game or equivalently in the first phase. For every valuation ι which is generated by σ in the quantification game, there is an outcome π ending in v_{C_1} such that $\iota = \iota_\pi$. Conversely, for every outcome π ending in v_{C_1}, ι_π is generated by σ in the quantification game. □

Note that the reduction could also be done with three actions only, which is the maximal number of actions necessary from any vertex. Also the reduction uses unions of intervals (due to $\neq i$ constraints). Finally, the arena is non-deterministic at each vertex corresponding to universal quantifiers in φ.

We extend this reduction in two ways to get rid of nondeterminism. First, instead of QBF-SAT, one can encode 3SAT (which is known to be NP-complete [Coo71]) and obtain a deterministic parameterized game:

Proposition 5.28. *When constraints are finite unions of intervals, and arenas are deterministic, the semi-parameterized reachability game problem is* NP*-hard.*

Proof sketch. The proof of the proposition builds on the previous reduction with the simple modification that, since all Boolean variables are existentially quantified in a SAT formula, the non-determinism at vertices v_{2s+1} is resolved by actions of Eve. Formally, we modify the statement (2) in the definition of the transition function of $\mathcal{A}_\varphi$ from Page 111 with:

$$\Delta(v_{2s+1}, a_{2s+2}, v_{x_{2s+2}}) = \Delta(v_{2s+1}, \bar{a}_{2s+2}, v_{\bar{x}_{2s+2}}) = \mathbb{N}_{>0}.$$

Then one can mimic the rest of the proof of Proposition 5.26 to establish the NP-hardness for deterministic arenas when constraints are finite union of intervals. □

Second, we extend the reduction to show a PSPACE lower bound for the problem with semilinear predicates, even for deterministic arenas:

Proposition 5.29. *When constraints are semilinear sets and arenas are deterministic, the semi-parameterized reachability game problem is* PSPACE*-hard.*

Proof sketch. The proof again builds on the reduction used in the proof of Proposition 5.26 where we replace the unions of intervals with appropriate semilinear predicates. More precisely, for every $1 \leq i \leq 2r$, we use the semilinear set P_i consisting of the set of

multiples of p_i, where p_i is the i-th prime number. The reduction from the same QBF formula as in Figure 5.8 to an arena with constraints as semilinear predicates is illustrated in Figure 5.9. Formally, the definition of the transition relation of $\mathcal{A}_\varphi$ from Page 111 are modified as follows: for every $0 \leq s \leq r-1$, $1 \leq i \leq 2r$, $1 \leq h \leq m$ and $1 \leq j \leq 3$:

1. $\Delta(v_{2s}, a_{2s+1}, v_{x_{2s+1}}) = P_{2s+1}$, and $\Delta(v_{2s}, a_{2s+1}, \top) = \neg P_{2s+1}$;
   $\Delta(v_{2s}, \bar{a}_{2s+1}, v_{\bar{x}_{2s+1}}) = \neg P_{2s+1}$, and $\Delta(v_{2s}, \bar{a}_{2s+1}, \top) = P_{2s+1}$;
2. $\Delta(v_{2s+1}, u, v_{x_{2s+2}}) = P_{2s+2}$, and $\Delta(v_{2s+1}, u, v_{\bar{x}_{2s+2}}) = \neg P_{2s+2}$;
3. $\Delta(v_{x_i}, c, v_i) = \Delta(v_{\bar{x}_i}, c, v_i) = \mathbb{N}_{>0}$;
4. $\Delta(v_{C_h}, a_i, v_{C_{h+1}}) = P_i$ if $\ell_{h,j} = x_i$; and
   $\Delta(v_{C_h}, \bar{a}_i, v_{C_{h+1}}) = \neg P_i$ if $\ell_{h,j} = \neg x_i$.

To obtain a complete arena, all unspecified transitions lead to a sink vertex $\bot$.

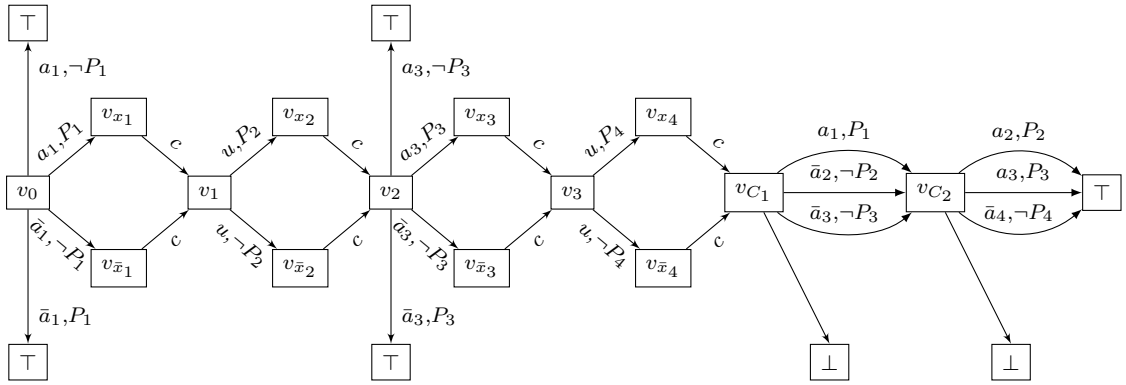


Figure 5.9: Reduction for formula $\varphi = \exists x_1 \forall x_2 \exists x_3 \forall x_4 \cdot (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_2 \vee x_3 \vee \neg x_4)$. Predicate P_i is "divisible by i-th prime number".

Intuitively, at the end of the first phase, the truth value of variable x_i is witnessed by the fact that the set of possible number of opponents is a multiple of p_i if x_i is set to true (that is P_i is satisfied), and it is not a multiple of p_i if x_i is set to false (that is, $\neg P_i$ is satisfied). The rest of the proof is identical to that of Proposition 5.26. □

We have now proved all the bounds mentioned in Table 5.1 for a semi-parameterized reachability game. In particular, in the most general case, when the constraints in the arena are semilinear predicates, the problem is PSPACE-complete. However, it falls into lower complexity classes for simpler constraints, for instance, (finite unions of) intervals.

5.4 Concluding remarks

In this chapter, we considered the decision problem of existence of a winning strategy for Eve on a parameterized arena against an unknown number of opponents. Her opponents play as a coalition and the task of an adversarial environment is two-fold: first, it chooses the number of agents participating in the game at the beginning and second, it resolves the non-determinisms in the arena. We show that such a game can be reduced to a two-player game between Eve and an adversary in which, we show that, Eve has a winning strategy only if she has one in the earlier. We call the latter the semi-parameterized game. A semi-parameterized game can then be reduced to a turn-based game with appropriate winning objective for Eve, and we show a correspondence between the winning strategies for Eve in both games. We have established tight complexity bounds for the semi-parameterized games when the winning objective is a reachability one. The complexity of the problem differs on the type of the constraints, whether they are (finite unions of) intervals or semilinear sets, and the type of the arena, whether it is deterministic or not. Table 5.1 presents the complexity results for semi-parameterized reachability problem in a nutshell.

Bounds on parameterized reachability problem

While we have only shown the bounds for semi-parameterized reachability problem, using the ingredients from this chapter, we can also show a similar tight complexity bound for the game problem *Eve against unknown number of opponents* on parameterized arenas for reachability objective. Indeed, recall from Lemmas 5.10 and 5.11 that parameterized games and semi-parameterized ones, with constraints as semilinear predicates, are inter-reducible in polynomial time, furthermore, the winning region of Eve is preserved. Therefore, the upper and lower bounds shown in Proposition 5.22 and Proposition 5.29, respectively, apply to the game problem on parameterized arenas. This can be formalized as follows.

Proposition 5.30. *For reachability objectives for* Eve*, the decision problem '*Eve *against unknown number of opponents' on a parameterized arena is* PSPACE*-complete.*

Beyond reachability

The upper and lower bounds were established for semi-parameterized game problem in Section 5.3 and Proposition 5.30 shows that the similar tight bounds hold for games on parameterized arenas where, in both cases, the winning objective for Eve is reachability. A natural question is then to ask whether the same bounds also apply for other winning objectives or Eve.

Notice that the knowledge game abstraction is correct for any winning objective, as witnessed in Theorem 5.16. As a consequence, we immediately get a corollary that semi-parameterized games are decidable for usual ω-regular objectives, for which a decision

procedure exists in a two-player turn-based setting. The decidability result is stated in Proposition 5.17. Again, due to Lemma 5.11, those games are also decidable on parameterized arenas.

Although the knowledge game abstraction leads us to the decidability of the decision problem for the aforementioned objectives, the DFS algorithm using a tagged tree would not directly apply for arbitrary winning objectives. Indeed, by construction, we stop exploring a node (v, k) in that tree whenever $v = t$ (t is the target vertex for reachability objective) is visited or if there is no output vertices in the subgame arena $\mathcal{K}_{\mathcal{G}}[v, K]$. Here, we implicitly use the intuition that in a reachability game, the winning status of a play is determined by only a finite prefix of the play (whether the target is reached, and for positive instances, it can be reached within a linear bound in the size of the arena). However, for other objectives, this might not be the case in general, and hence, it is not straight forward to conclude the same upper bounds for those objectives. However, for safety and Büchi objectives, the same PSPACE lower bound result apply. Indeed, first notice that the same reduction as in Proposition 5.29 works with a safety objective described by the set $F = V \setminus \{\bot\}$; and second, for Büchi objectives, it is enough to see that they subsume reachability objectives. Nonetheless, a naive exponential upper bound can be proved for (semi-)parameterized safety (resp., Büchi) games: one can reduce the semi-parameterized arena to the exponential size knowledge arena and apply standard algorithms on the latter that runs in linear (resp., polynomial) time in the size of the knowledge game.

Proposition 5.31. *For safety and Büchi objectives for* Eve*, the decision problem '*Eve *against unknown number of opponents' on a parameterized arena is in* EXPTIME *and* PSPACE*-hard.*

While exponential upper bounds for the safety and Büchi objectives are almost immediate consequences of the knowledge game abstraction, we believe one can achieve PSPACE upper bounds for these objectives by first modifying the winning conditions for Eve in the subgames $\mathcal{K}_{\mathcal{G}}[v, K]$ appropriately, then defining a tagged tree of polynomial height similarly to the reachability case, and applying a DFS tagging algorithm that runs in polynomial space in the size of the input. Formalizing this idea, and closing the complexity gap in Proposition 5.31 is therefore an immediate follow-up question we would like to investigate. Further, one can also consider other ω-regular objectives for the game problem.

Chapter 6

Synthesizing Safe Coalition Strategies

In Chapter 4, we introduced parameterized arenas, an extension of classical two-player concurrent game arenas, with an arbitrary number of agents. However, the agents are assigned a unique identifier that is implicitly used in describing the model. In contrast to two-player arenas, edges in parameterized arenas are labelled with (regular) languages of finite yet possibly unbounded words. In parameterized games, the agents do not know *a priori* the number of agents participating in the interaction.

Chapter 5 was devoted to solving the parameterized game problem on a setting where the first player, called Eve, is distinguished, and we ask if there exists a winning strategy for her against any number of opponents and any strategies of them. We have shown the game problem is PSPACE-complete when the winning objective for Eve is a reachability objective.

We are further interested in another setting, called coalition game, where agents play collectively and try to achieve a common goal. More precisely, at any vertex, depending on the past history, agents choose actions according to their strategies, and a strategy profile is winning if every play in its outcome is winning for the coalition. This chapter studies this setting, mainly for safety winning objectives for the coalition.

We first define *coalition strategies*. In contrast to the notion of a strategy profile, which is an infinite tuple of individual strategies of the agents, a coalition strategy is a function that assigns to every history an ω-word. We remark that, in the coalition setting, these two notions are equivalent: there exists a strategy profile for coalition if and only if there exists a *coalition strategy*. The main result of this chapter is that the *coalition problem* is decidable for safety objectives. Moreover, we provide an algorithm to decide the above that runs in exponential space in the size of input, and we also show a PSPACE lower bound.

The exponential space algorithm consists in two major steps. The first step is to unfold the game arena $\mathcal{A}$ into a finite tree $\mathcal{T}$ with the same winning objective for the coalition. The tree is finite because, since the winning objective is a safety one, if a vertex

repeats along a history, the coalition can play the same strategy as it played in the first occurrence of the vertex. This idea will be formalized later in this chapter. Based on the above idea, we show the correctness of the tree unfolding: there is a winning strategy for the coalition in the parameterized arena if and only if there is one in the tree.

In the second step, we construct a deterministic finite automaton $\mathcal{B}$ from the NFA's corresponding to the regular languages in the arena. The automaton $\mathcal{B}$ accepts ω-words over alphabet Σ^m where Σ is the set of actions and m is the number of internal nodes in the finite tree unfolding. We set the accepting condition to a safety condition, and show that an ω-word in the language of $\mathcal{B}$ corresponds to a winning coalition strategy in $\mathcal{T}$ and vice-versa.

The EXPSPACE upper bound for the safe coalition problem is then achieved by proving that an accepting run of $\mathcal{B}$ can be guessed using exponential space. A PSPACE lower bound for the safe coalition problem follows from the idea used in Chapter 5 to prove Propositions 5.26 and 5.29. Moreover, a winning strategy can be synthesized, when it exists, from an accepting run of $\mathcal{B}$ using exponential space. We prove an optimal bound on the (exponential) size of the memory required for the coalition to win a parameterized safety game: if the coalition has a winning strategy, it has one of size at most exponential; furthermore, there exists a family of games, whose size is linear in n, such that any winning coalition strategy requires a memory of size $O(2^n)$.

This chapter is based on the publication [BBM20] co-authored with Nathalie Bertrand and Patricia Bouyer appeared in FSTTCS 2020.

Organization of the chapter

This chapter is organized as follows. Section 6.1 first recalls briefly the coalition game setting on parameterized arenas and illustrative examples are discussed. We then proceed to prove the decidability of the safe coalition problem. We construct a finite tree unfolding of an arena and show its correctness in Section 6.2. Then in Section 6.3, we provide an algorithm to solve the coalition problem on the tree unfolding that runs in exponential space in the size of the arena, proving an EXPSPACE upper bound for the problem. A PSPACE lower bound result is shown in Section 6.4. Section 6.5 talks about synthesis of a winning strategy for coalition, if exists, in parameterized safety games. Finally, we close the chapter with a discussion on future research directions in Section 6.6.

6.1 Game setting

Let us first briefly recall the coalition setting on parameterized arenas from Section 4.3 that we aim to study in this chapter.

Fix a parameterized arena $\mathcal{A} = \langle V, \Sigma, \Delta \rangle$. A strategy for agent i is a mapping $\sigma_i : V^+ \to$

Σ from histories to the set of actions. Then a strategy profile is an infinite tuple of strategies: $\widetilde{\sigma} = \langle \sigma_1, \sigma_2, \ldots \rangle \in (V^+ \to \Sigma)^\omega$. For a tuple $\langle \sigma_1, \ldots, \sigma_k \rangle$ of strategies from v_0 of k agents, we define k-*outcome* the set of k-realizable plays compatible with the strategies; formally, $\mathsf{Out}^k_{\mathcal{A}}(v_0, \sigma_1, \ldots, \sigma_k) = \{v_0 v_1 \cdots \mid \forall j \geq 0,\ \sigma_1(v_0 \cdots v_j)\sigma_2(v_0 \cdots v_j) \ldots \sigma_k(v_0 \cdots v_j) \in \Delta(v_j, v_{j+1})\}$. Then a strategy profile $\widetilde{\sigma} = \langle \sigma_1, \sigma_2, \ldots \rangle$ from v_0 defines *outcome*, the set of induced plays: $\mathsf{Out}_{\mathcal{A}}(v_0, \widetilde{\sigma}) = \bigcup_{k \in \mathbb{N}_{>0}} \mathsf{Out}^k_{\mathcal{A}}(v_0, \sigma_1, \ldots, \sigma_k)$.

Observe that a strategy profile can equivalently be described as a *coalition strategy* $\boldsymbol{\sigma} : V^+ \to \Sigma^\omega$, a partial function that maps each history to an ω-word over Σ, as illustrated in Table 6.1. Indeed, if an enumeration of histories $(h_j)_{j \in \mathbb{N}}$ is fixed, a strategy profile can be seen as a table with infinitely many rows –one for each agent– and infinitely many columns indexed by histories. Reading the table vertically provides the coalition strategy view: each history is mapped to an ω-word, obtained by concatenating the actions chosen by each of the agents.

	h_0	h_1	h_2	h_3	$\ldots$
σ_1	a	b	b	b	$\ldots$
σ_2	b	b	b	b	$\ldots$
σ_3	b	a	a	a	$\ldots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	

Table 6.1: From strategy profile to coalition strategy.

While considering the existence problem of a winning strategy profile $\widetilde{\sigma}$ for the coalition, it is equivalent to ask whether such a table exists that corresponds to a winning strategy for them, or equivalently, whether there exists a coalition strategy $\boldsymbol{\sigma}$ that is winning. Since, in this chapter, we are interested in such problems, in the sequel, we mostly take the coalition strategy view, but may interchangeably also consider strategy profiles.

Remark that this might not be the case for some other decision problems. For instance, where actions of Eve are distinguished and we want to decide whether there exists a strategy σ_1 for her that is winning for any k and any $\sigma_2, \ldots, \sigma_k$, the coalition view of a strategy profile is of lesser use. Indeed, for a fixed σ_1, one needs to check if it is winning for all possible $\sigma_2, \sigma_3, \ldots$ strategies of opponents of Eve.

We define here the outcome of a coalition strategy $\boldsymbol{\sigma}$. Recall that for two words $u \in \Sigma^*$ and $w \in \Sigma^+ \cup \Sigma^\omega$, we write $u \sqsubseteq w$ to denote u is a prefix of w, and for any $k \in \mathbb{N}_{>0}$, $[w]_{\leq k}$ denotes the prefix of length k of w (belongs to Σ^k).

Given a coalition strategy $\boldsymbol{\sigma}$, an initial vertex v_0 and a number of agents $k \in \mathbb{N}_{>0}$, we define the k-*outcome* $\mathsf{Out}^k_{\mathcal{A}}(v_0, \boldsymbol{\sigma})$ as the set of all k-realizable plays induced by $\boldsymbol{\sigma}$ from v_0. Formally, $\mathsf{Out}^k_{\mathcal{A}}(v_0, \boldsymbol{\sigma}) = \{v_0 v_1 \cdots \mid \forall j \geq 0,\ [\boldsymbol{\sigma}(v_0 \cdots v_j)]_{\leq k} \in \Delta(v_j, v_{j+1})\}$. Note that the completeness assumption ensures that the set $\mathsf{Out}^k_{\mathcal{A}}(v, \boldsymbol{\sigma})$ is not empty. Then the *outcome* of coalition strategy $\boldsymbol{\sigma}$ is simply $\mathsf{Out}_{\mathcal{A}}(v_0, \boldsymbol{\sigma}) = \bigcup_{k \in \mathbb{N}_{>0}} \mathsf{Out}^k_{\mathcal{A}}(v_0, \boldsymbol{\sigma})$.

We recall the definition of a Mealy automaton in the setting of a coalition game, for a

finite memory coalition strategy $\boldsymbol{\sigma}$. A *Mealy automaton* is a tuple $\mathcal{M} = (\mathtt{M}, V, m_0, \mathsf{act}, \mathsf{upd})$ where $\mathtt{M}$ is a finite set of *memory states*; $m_0 \in \mathtt{M}$ is the initial state; $\mathsf{act} : \mathtt{M} \times V \to \Sigma^\omega$ is the transition choice function; and $\mathsf{upd} : \mathtt{M} \times V \to \mathtt{M}$ is the memory update function. A Mealy automaton $\mathcal{M}$ describes a strategy $\sigma_\mathcal{M}$ from v_0 as follows: first, for any $h \in \mathsf{Hist}_\mathcal{A}$, inductively define $\mathtt{m}[h] \in \mathtt{M}$ by $\mathtt{m}[v_0] = m_0$, and $\mathtt{m}[h \cdot v] = \mathsf{upd}(\mathtt{m}[h], v)$; and then $\sigma_\mathcal{M}$ is defined as $\sigma_\mathcal{M}(h) = \mathsf{act}(\mathtt{m}[h], \mathsf{last}(h))$, where $\mathsf{last}(h)$ is the last vertex of history h.

We say that a coalition strategy $\boldsymbol{\sigma}$ *uses memory* $\mathtt{M}$ if there exists a Mealy automaton $\mathcal{M}$ such that $\boldsymbol{\sigma} = \sigma_\mathcal{M}$, and $\boldsymbol{\sigma}$ is *memoryless* whenever $\mathtt{M}$ is a singleton.

Given a parameterized arena $\mathcal{A} = \langle V, \Sigma, \Delta \rangle$, and an initial vertex v_0, a set $\mathsf{Win} \subseteq V^\omega$ of plays defines a game $\mathcal{G} = (\mathcal{A}, \mathsf{Win})$ for the coalition. A coalition strategy $\boldsymbol{\sigma}$ from v_0 in $\mathcal{G}$ is said *winning* if $\mathsf{Out}_\mathcal{A}(v_0, \boldsymbol{\sigma}) \subseteq \mathsf{Win}$. Our goal is to study the decidability and complexity of the existence of winning coalition strategies, and to synthesize such winning coalition strategies when they exist. We therefore introduce the following decision problem:

COALITION PROBLEM
Input: A parameterized game $\mathcal{G} = (\mathcal{A}, \mathsf{Win})$ and an initial vertex v_0.
Question: Yes if and only if $\exists \boldsymbol{\sigma}$ such that $\mathsf{Out}_\mathcal{A}(v_0, \boldsymbol{\sigma}) \subseteq \mathsf{Win}$.

This is a coordination problem: agents should agree on a joint strategy which, when played in the arena and no matter how many agents are involved, the resulting play is winning. Note that, due to the correspondence between coalition strategies and infinite tuples of individual strategies mentioned on Page 121, the coalition strategies are distributed: the only information required for an agent to play her strategy is the history so far, not the number of agents selected by the environment; however she can infer some information about the number of agents from the history. Note that there is no direct communication between agents.

In this chapter, we consider safety objectives for the coalition, unless otherwise specified. A safe coalition game is described as $\mathcal{G} = (\mathcal{A}, S)$ where $\mathcal{A} = \langle V, \Sigma, \Delta \rangle$ is a parameterized arena and $S \subseteq V$ is a set of 'safe' vertices. Without loss of generality, we assume that the vertices in $V \setminus S$ are sinks. A coalition strategy $\boldsymbol{\sigma}$ from v_0 in the safety game $\mathcal{G}$ is *winning* if all induced plays only visit vertices from S: $\mathsf{Out}_\mathcal{A}(v, \boldsymbol{\sigma}) \subseteq S^\omega$. The *safe coalition problem* can be described as follows.

SAFE COALITION PROBLEM
Input: A parameterized safety game $\mathcal{G} = (\mathcal{A}, S)$ and an initial vertex v_0.
Question: Yes if and only if $\exists \boldsymbol{\sigma}$ such that $\mathsf{Out}_\mathcal{A}(v_0, \boldsymbol{\sigma}) \subseteq S^\omega$.

We have already seen examples of the coalition game in Examples 4.22 and 4.33 for a safety and a reachability objective, respectively. Here we present another example which will be the running example of this chapter.

Example 6.1. *Figure 6.1 presents a non-deterministic parameterized arena. Here* $V = \{v_0, v_1, v_2, \bot\}$ *with* $\bot$ *a sink vertex which is not depicted here, and* $\Sigma = \{a, b\}$. *The edge*

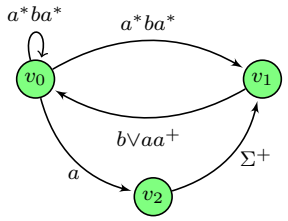


Figure 6.1: Example of a non-deterministic parameterized arena. Only safe vertices (coloured in green) have been depicted here. All unspecified transitions lead to an unsafe vertex $\perp$.

labels represent the transition function: for instance, if for some number of agents k (selected by the environment, initially not known to the agents), the k-length prefix of the word collectively chosen by the agents at v_0 belongs to a^*ba^*, then the play either stays at v_0 or moves to v_1 (non-determinism is resolved by the environment). The arena in the picture is not complete, we assume all unspecified transitions from any vertex go to $\perp$ to achieve a complete arena. We consider a safety objective for the coalition described by the set $S = V \setminus \{\perp\}$, i.e., *$\perp$ is the only unsafe vertex. The safe vertices are depicted in the picture and they have been coloured green.*

*In this example, one can show that the agents have a winning coalition strategy $\boldsymbol{\sigma}$ from v_0 to stay within green (*i.e., *safe) vertices. Consider the coalition strategy $\boldsymbol{\sigma}$ such that for any history $h \in V^*$, $\boldsymbol{\sigma}(hv_0) = aba^\omega$, $\boldsymbol{\sigma}(hv_0v_2) = a^\omega$, $\boldsymbol{\sigma}(hv_0v_1) = a^\omega$, and $\boldsymbol{\sigma}(hv_0v_2v_1) = b^\omega$. Intuitively, on playing aba^ω from v_0, in one step, the game either stays in v_0 (which is 'safe') or moves to v_2 (in case the number of agents $k = 1$) or to v_1 (in case $k \geq 2$); from v_1, depending on history, the coalition plays either b^ω (when the history is $v_0v_2v_1$ and hence they infer $k = 1$) or a^ω (otherwise) which leads the game back to v_0 (note that at vertex v_2, choice of actions of the agents is not important, they can collectively play any ω-word). Finally, the coalition plays the same each time the game goes back to v_0, and hence one of the above holds. In all cases, the game never reaches $\perp$. However, one can show that there is no memoryless coalition winning strategy. Indeed, the coalition strategy a^ω from v_1 is losing for $k = 1$, similarly b^ω from v_1 is losing for $k \geq 2$, and any other strategy is also losing. For instance, ba^ω from v_0 is losing because if the game moves to v_1, coalition has no information on the number of agents and hence any word from v_1 will be losing (a^ω is losing for $k = 1$, b^ω is losing for $k \geq 2$, and similarly for other words).*

The rest of the chapter is devoted to solve the safe coalition problem and proving complexity upper and lower bounds for the same. We show that the problem is decidable and solvable in exponential space, and prove a PSPACE lower bound. Moreover, for positive instances, we give a procedure to synthesize a winning coalition strategy in exponential space which uses exponential memory; furthermore, we show this exponential blow-up in the size of the memory is unavoidable in general.

To prove the decidability and establish the complexity upper bound, we first construct a finite tree unfolding of the arena, which is equivalent for deciding the existence of a winning coalition strategy for safety objectives. The unfolding is finite because, if a vertex

is repeated along a play, the coalition can play the same ω-word as in the first visit. We then show how to solve the safe coalition problem at the finite tree level.

6.2 The tree unfolding

From a safe coalition game $\mathcal{G} = (\mathcal{A}, S)$, we construct a finite tree as follows: we unfold the arena $\mathcal{A}$ until either some vertex is repeated along a branch or an unsafe vertex is reached. The nodes of the tree are labelled with the corresponding vertices and the edges are labelled with the same regular languages as in the arena $\mathcal{A}$. We then consider a safety objective for the coalition on the tree, described by the nodes that are labelled with vertices from S. The intuition behind this construction is that if a vertex is repeated in a winning play in $\mathcal{A}$, since the winning condition is a safety one, the coalition can play the same strategy as it played in the first occurrence of the vertex. Note however that multiple nodes in the tree may have the same label but different (winning) strategies depending on the history; for example, in Example 6.1, strategy at v_1 indeed depends on the history. This is why we need to consider a tree unfolding abstraction, and a DAG abstraction would not suffice.

Recall that, for any two nodes n, n' of a tree, we say n' is a *child* of n (and n the *parent* of n') if n' is an immediate successor of n according to the edge relation; and n an *ancestor* of n' if there exists a path from n to n' in the tree. The tree unfolding of a parameterized arena is formally defined as follows.

Definition 6.2. *Let $\mathcal{G} = (\mathcal{A}, S)$ be a parameterized safety game with $\mathcal{A} = \langle V, \Sigma, \Delta \rangle$ and $v_0 \in V$ an initial vertex. The* tree unfolding *of $\mathcal{G}$ w.r.t. v_0 is the tree $\mathcal{T} = \langle \mathsf{N}, \mathsf{E}, \ell_\mathsf{N}, \ell_\mathsf{E} \rangle$ rooted at $\mathsf{n}_0 \in \mathsf{N}$, where N is the finite set of nodes, $\mathsf{E} \subseteq \mathsf{N} \times \mathsf{N}$ is the set of edges, $\ell_\mathsf{N} : \mathsf{N} \to V$ is the node labelling function, $\ell_\mathsf{E} : \mathsf{N} \times \mathsf{N} \to 2^{\Sigma^+}$ is the edge labelling function, and:*

- *the root n_0 satisfies $\ell_\mathsf{N}(\mathsf{n}_0) = v_0$;*
- *$\forall \mathsf{n} \in \mathsf{N}$, if $\ell_\mathsf{N}(\mathsf{n}) \in S$ and for every ancestor n'' of n, $\ell_\mathsf{N}(\mathsf{n}'') \neq \ell_\mathsf{N}(\mathsf{n})$, then $\forall v' \in V$ such that $\Delta(v, v') \neq \emptyset$, there is n' a child of n with $\ell_\mathsf{N}(\mathsf{n}') = v'$ and $\ell_\mathsf{E}(\mathsf{n}, \mathsf{n}') = \Delta(v, v')$; otherwise, the node n has no successor.*

In words, the root of the unfolding tree is labelled with the initial vertex v_0 of $\mathcal{A}$. Then we construct the tree inductively as follows. For every $\mathsf{n} \in \mathsf{N}$ such that $\ell_\mathsf{N}(\mathsf{n})$ is in the set S and all ancestors of n has a different label (*i.e.* a label did not repeat), we extend the branch: for each transition from $\ell_\mathsf{N}(\mathsf{n})$ in $\mathcal{A}$, we add a child n' of n and set the labellings according to that transition - the node is labelled with the corresponding vertex and the edge with the corresponding language; for example, letting $\ell_\mathsf{N}(\mathsf{n}) = v$, if there is a transition from v to v' in $\mathcal{A}$, then we add a child n' of n with label v', and the corresponding edge is labelled $\Delta(v, v')$.

Note that each node in $\mathcal{T}$ corresponds to a unique history in $\mathcal{A}$, and the unfolding is stopped along a branch when a vertex repeats or an unsafe vertex is encountered. The set of nodes is further partitioned into $\mathsf{N} = \mathsf{N}_{\mathsf{int}} \uplus \mathsf{N}_{\mathsf{leaf}}$ where $\mathsf{N}_{\mathsf{int}}$ is the set of *internal nodes* and $\mathsf{N}_{\mathsf{leaf}}$ the set of *leaves* of $\mathcal{T}$; some leaves are unsafe, others have an ancestor with the same label.

Let us illustrate the construction on an example.

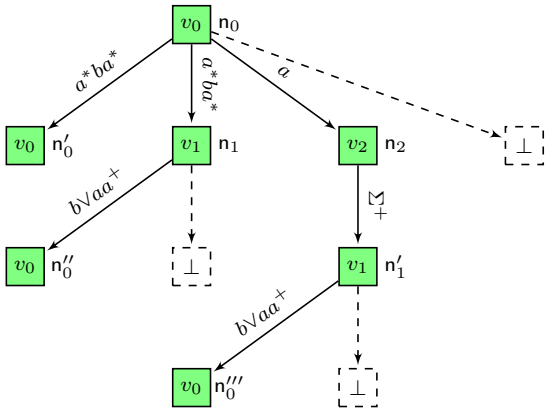


Figure 6.2: Tree unfolding of the arena in Figure 6.1. The nodes labelled with safe vertices are coloured green. Notice here that the unsafe leaves (and the edges leading to them) are presented with dashed squares (resp., arrows).

Example 6.3. *Figure 6.2 represents the tree unfolding of the parametrized arena depicted in Figure 6.1. On that example, the nodes are represented by squares, their names are written on the side of the nodes (other than nodes with label $\bot$) and labels within the nodes. The leaves that correspond to unsafe vertices (resp., the edges leading to them) are presented with dashed squares (resp., arrows). Notice that any leaf node is either labelled with an unsafe vertex (for instance, nodes with label $\bot$) or it has a unique ancestor with the same label (this is the case for nodes n'_0, n''_0 and n'''_0). However, multiple internal nodes in different branches can have the same label, for instance, both n_1 and n'_1 are labelled with v_1.*

In the following lemma, we show an upper bound on the size of the tree unfolding for a parameterized safety game w.r.t. an initial vertex.

Lemma 6.4. *Let $\mathcal{G} = (\mathcal{A}, S)$ be a parameterized safety game with $\mathcal{A} = \langle V, \Sigma, \Delta \rangle$ and $v_0 \in V$ an initial vertex, and $\mathcal{T}$ be the tree unfolding of $\mathcal{G}$ w.r.t. v_0. Then the size of $\mathcal{T}$ is at most $|V|^{|V|+1}$. Moreover, an exponential blow-up in size of $\mathcal{T}$ is unavoidable in general.*

Proof. Fix a parameterized safety game $\mathcal{G} = (\mathcal{A}, S)$ with $\mathcal{A} = \langle V, \Sigma, \Delta \rangle$ and $v_0 \in V$ an initial vertex. Let $\mathcal{T}$ be the corresponding tree unfolding w.r.t. v_0. Any branch in the tree has at most two nodes with the same label, since in that case we stop the unfolding. Therefore, the height of a branch of $\mathcal{T}$ is bounded by $|V| + 1$. Further, notice that any node can have at most one child for each outgoing transition. Hence, the branching degree of $\mathcal{T}$ is at most $|V|$. We conclude that the size of the tree is upper bounded by $|V|^{|V|+1}$.

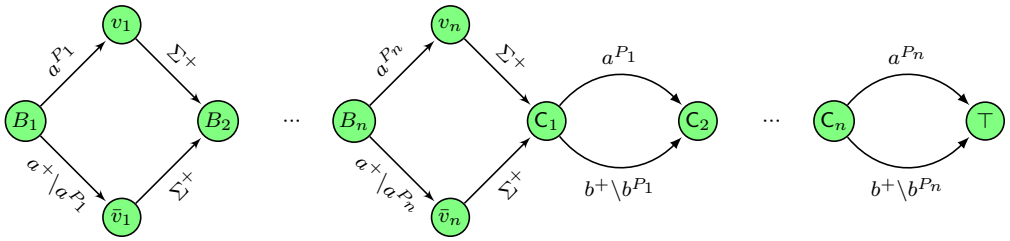


Figure 6.3: Example arena such that the tree unfolding is exponential. All unspecified transitions lead to a sink losing vertex $\bot$. Set P_i denotes multiples of the i-th prime number. For any play reaching C_1, for every i, the number of agents is in P_i iff the play went through v_i.

The exponential bound is reached by a family $(\mathcal{A}_n)_{n\in\mathbb{N}_{>0}}$ of deterministic arenas, shown in Figure 6.3, with $2n$ blocks and $O(n)$ vertices. Let p_i be the i-th prime number and P_i the set of multiples of p_i. For simplicity, we write a^{P_i} to denote the set of words in $(a^{p_i})^+$, that is words from a^+ whose length is a multiple of p_i. Intuitively, the transition at B_i determines whether the number of agents is a multiple of i-th prime: all agents must choose a at that vertex, and if the game moves to v_i (resp., $\bar{v}_i$), the number of agents is in set P_i. Depending on the history, at vertex C_i ($1 \le i \le n$), the coalition may choose action a (resp., b) in case the play visited v_i (resp., $\bar{v}_i$). Observe that, to win the game, coalition needs to keep track of the full histories in the first n blocks, and there are exponentially many such histories; moreover, each such history corresponds to a different node in its unfolding tree which causes the exponential blow-up in its tree unfolding.

Formally, Figure 6.3 represents a parameterized safety game $\mathcal{G}_n = (\mathcal{A}_n = \langle V_n, \Sigma, \Delta_n\rangle, S_n)$ where $V_n = \{B_i, v_i, \bar{v}_i, \mathsf{C}_i \mid 1 \le i \le n\} \cup \{\top, \bot\}$, and $\Sigma = \{a, b\}$. The edge labellings represent the transition function, for instance, for each $1 \le i \le n$, $\Delta_n(B_i, v_i) = a^{P_i}$ and $\Delta_n(B_i, \bar{v}_i) = a^+ \setminus a^{P_i}$; similarly, $\Delta_n(\mathsf{C}_i, \mathsf{C}_{i+1}) = a^{P_i} + (b^+ \setminus b^{P_i})$. The unspecified transitions move to the sink vertex $\bot$ which is not depicted here. We consider the initial vertex B_1; the safety condition is described by the set $S_n = V_n \setminus \{\bot\}$, presented in green in the picture.

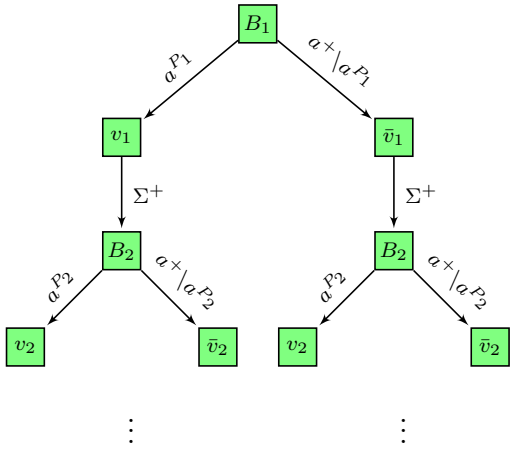


Figure 6.4: A part of the tree unfolding constructed from $\mathcal{G}_n$ in Figure 6.3.

In the tree unfolding of $\mathcal{G}_n$ w.r.t. B_1, there is a unique node for each different history in the first phase of the game (*i.e.*, until the visit of C_1) and there are exponentially many of them. Therefore, the tree is of size $O(2^n)$, exponential in the size of $\mathcal{G}_n$. A part of the tree is depicted in Figure 6.4. In that picture, node names are omitted, and node labels are written inside the nodes. □

We now define histories and plays in $\mathcal{T}$. Histories in $\mathcal{T}$ are defined similarly as in $\mathcal{G}$ (except vertices are replaced by nodes); the set of such histories is denoted $\mathsf{Hist}_{\mathcal{T}}$. Note that since the tree is finite, a play cannot be defined as in $\mathcal{G}$. In this context, we call a maximal history ending in a leaf a play in $\mathcal{T}$. Note that, contrary to the definition of a play in $\mathcal{A}$, a play in $\mathcal{T}$ is a *finite* sequence of nodes ending in a leaf. They are defined formally as follows.

Definition 6.5. *A* history *in $\mathcal{T}$ is a finite sequence of nodes $H = \mathsf{n}_0\mathsf{n}_1 \dots \mathsf{n}_p \in \mathsf{N}^+$ such that for every $0 \leq j < p$, $(\mathsf{n}_j, \mathsf{n}_{j+1}) \in \mathsf{E}$. We denote by $\mathsf{Hist}_{\mathcal{T}}$ the set of all histories in $\mathcal{T}$. A* play *in $\mathcal{T}$ is a maximal history,* i.e.*, a finite sequence of nodes ending with a leaf, thus in $\mathsf{N}^+_{\mathsf{int}} \cdot \mathsf{N}_{\mathsf{leaf}}$. We denote by $\mathsf{Play}_{\mathcal{T}}$ the set of all play in $\mathcal{T}$.*

First note that there is a trivial bijection ι from N to $\mathsf{Hist}_{\mathcal{T}}$, and we will swap between the two notions using ι when convenient. Indeed, a node n uniquely determines a history $\mathsf{n}_0 \dots \mathsf{n}$ in $\mathsf{Hist}_{\mathcal{T}}$ and vice-versa.

Similarly to the parameterized arena setting, we define naturally the notions of k-realizability and of realizability for histories and plays. Formally, a history (or, play) $H = \mathsf{n}_0\mathsf{n}_1 \dots \mathsf{n}_p \in \mathsf{N}^+$ is k-realizable if for all $0 \leq j < p$, there exists $u \in \Sigma^k$ with $u \in \ell_{\mathsf{E}}(\mathsf{n}_j, \mathsf{n}_{j+1})$; H is realizable if it is k-realizable for some $k \in \mathbb{N}_{>0}$.

For a tree unfolding $\mathcal{T}$ of a parameterized safety game $\mathcal{G}$, we extend the definition of the node labelling function ℓ_{N} to histories (resp., plays) naturally: for a history (or, play) $H = \mathsf{n}_0\mathsf{n}_1 \dots \mathsf{n}_p \in \mathsf{N}^+$, define $\ell_{\mathsf{N}}(H) = \ell_{\mathsf{N}}(\mathsf{n}_0)\ell_{\mathsf{N}}(\mathsf{n}_1) \dots \ell_{\mathsf{N}}(\mathsf{n}_p)$.

Below we define strategies for the coalition on the tree. It seems natural to define a coalition strategy in $\mathcal{T}$, similar to one in parameterized arenas, as a mapping from histories in $\mathcal{T}$ to the set of ω-words over Σ. However, recall that a node in the tree corresponds to a unique history in $\mathcal{A}$. Therefore, to reflect a strategy of the coalition at history h in $\mathcal{A}$, such that it has a corresponding node n in $\mathcal{T}$, it is enough to assign the ω-word the coalition plays at h to node n. More precisely, we define a coalition strategy in the tree unfolding as a mapping from its internal nodes to Σ^ω, which is by definition a memoryless one. Although it is not immediate that this definition is sufficient to capture all strategies in $\mathcal{G}$ (since in general not all histories in $\mathcal{G}$ correspond to a node in $\mathcal{T}$), we later show this is indeed the case w.r.t. the winning strategies: there is a coalition winning strategy in $\mathcal{G}$ from v_0 if and only if there is a (memoryless) winning strategy in $\mathcal{T}$ from n_0.

Definition 6.6. *A* coalition strategy *in the unfolding tree is a mapping $\boldsymbol{\lambda} : \mathsf{N}_{\mathsf{int}} \to \Sigma^\omega$ that assigns to every internal node $\mathsf{n} \in \mathsf{N}_{\mathsf{int}}$ an ω-word $\boldsymbol{\lambda}(\mathsf{n})$.*

Given a strategy $\boldsymbol{\lambda}$ for coalition from n_0 in $\mathcal{T}$, and $k \in \mathbb{N}_{>0}$ a number of agents, the notions of (k-)outcome are defined similarly: the *k-outcome* $\mathsf{Out}^k_{\mathcal{T}}(\mathsf{n}_0, \boldsymbol{\lambda})$ is the set of plays that $\boldsymbol{\lambda}$ induces from n_0 that are k-realizable; and then the *outcome* of $\boldsymbol{\lambda}$ is simply the union of k-outcomes for all $k \in \mathbb{N}_{>0}$. Formally, it is defined as follows.

Definition 6.7. *Given a strategy $\boldsymbol{\lambda}$ for the coalition in $\mathcal{T}$ from n_0, and $k \in \mathbb{N}_{>0}$ a number of agents, the* k-outcome *of strategy $\boldsymbol{\lambda}$ is the set* $\mathsf{Out}^k_{\mathcal{T}}(\mathsf{n}_0, \boldsymbol{\lambda}) = \{\mathsf{n}_0\mathsf{n}_1 \dots \mathsf{n}_p \mid \forall 0 \leq j < p, [\boldsymbol{\lambda}(\mathsf{n}_j)]_{\leq k} \in \ell_{\mathsf{E}}(\mathsf{n}_j, \mathsf{n}_{j+1})\}$. Outcome *of strategy $\boldsymbol{\lambda}$ is* $\mathsf{Out}_{\mathcal{T}}(\mathsf{n}_0, \boldsymbol{\lambda}) = \bigcup_{k \in \mathbb{N}_{>0}} \mathsf{Out}^k_{\mathcal{T}}(\mathsf{n}_0, \boldsymbol{\lambda})$.

Finally, we define a safe coalition game on the tree unfolding. Let $\mathcal{G} = (\mathcal{A}, S)$ be a parameterized safety game and v_0 an initial vertex. Let $\mathcal{T}$ be the tree unfolding of $\mathcal{G}$ w.r.t. v_0. The set S defines a safe coalition game on $\mathcal{T}$. We say the coalition wins the safety game on $\mathcal{T}$ if it has a strategy that ensures visits of nodes only from S. A winning strategy is defined formally as follows.

Definition 6.8. *A coalition strategy $\boldsymbol{\lambda}$ in $\mathcal{T}$ from n_0 is* winning *for the safety objective defined by the set S if every play in $\mathsf{Out}_{\mathcal{T}}(\mathsf{n}_0, \boldsymbol{\lambda})$ ends in a leaf with label in S,* i.e.*, if for every $R = \mathsf{n}_0 \dots \mathsf{n}_p \in \mathsf{Out}_{\mathcal{T}}(\mathsf{n}_0, \boldsymbol{\lambda})$, $\ell_{\mathsf{N}}(\mathsf{n}_p) \in S$, equivalently, $\ell_{\mathsf{N}}(\mathsf{Out}_{\mathcal{T}}(\mathsf{n}_0, \boldsymbol{\lambda})) \subseteq S^+$.*

Indeed, the two conditions in the above definition are equivalent: whenever the label of the leaf node of a play is in S, so are the labels of the intermediary nodes, since otherwise we would have stopped the unfolding at an ancestor node.

Below we show the correctness of the tree unfolding of a parameterized safety game in the sense that there is a winning strategy for coalition in the parameterized game if and only if there is one in its tree unfolding. One direction of the statement is immediate: since a node in the tree corresponds to a unique history, a winning strategy in the parameterized game can be seen as one in the tree. For the other direction, we use the idea that since the winning objective is a safety condition, if a vertex repeats in a play in the parameterized arena, the coalition can play the same strategy as in its first occurrence. Based on this idea, we define for every history in the parameterized arena a *virtual history* that maintains the property that each vertex appears at most once and hence corresponds to a node in the tree. We then project a strategy in the tree to the parameterized arena using virtual histories.

Lemma 6.9. *Let $\mathcal{G} = (\mathcal{A} = \langle V, \Sigma, \Delta\rangle, S)$ be a parameterized safety game and $v_0 \in V$ and $\mathcal{T} = \langle \mathsf{N}, \mathsf{E}, \ell_{\mathsf{N}}, \ell_{\mathsf{E}}\rangle$ be the associated tree unfolding with root n_0. There exists a winning coalition strategy from v_0 in $\mathcal{G}$ iff there exists a winning coalition strategy from n_0 in $\mathcal{T}$.*

Proof. Assume first that the coalition of agents has a winning strategy $\boldsymbol{\sigma}$ in $\mathcal{G}$. Any history $H \in \mathsf{Hist}_{\mathcal{T}}$ can be projected to the history $\ell_{\mathsf{N}}(H) \in \mathsf{Hist}_{\mathcal{A}}$. We can hence define for every $\mathsf{n} \in \mathsf{N}_{\mathsf{int}}$, $\boldsymbol{\lambda}(\mathsf{n}) = \boldsymbol{\sigma}(\ell_{\mathsf{N}}(\iota(\mathsf{n})))$, where recall that ι is the bijection mapping from nodes to histories in $\mathcal{T}$. To prove that $\boldsymbol{\lambda}$ is winning in $\mathcal{T}$, consider any play $R = \mathsf{n}_0 \cdots \mathsf{n}_p$ in $\mathsf{Out}_{\mathcal{T}}(\mathsf{n}_0, \boldsymbol{\lambda})$ and let $\rho = \ell_{\mathsf{N}}(R) = v_0 \cdots v_p$ be its projection in $\mathcal{G}$. By construction, $\ell_{\mathsf{E}}(\mathsf{n}_i, \mathsf{n}_{i+1}) = \Delta(v_i, v_{i+1})$ for every $i < p$, and hence from the definition of $\boldsymbol{\lambda}$, ρ is a history

in $\mathcal{G}$ induced by $\boldsymbol{\sigma}$. Since $\boldsymbol{\sigma}$ is winning, ρ only visits *safe* vertices. In particular, $\ell_{\mathsf{N}}(\mathsf{n}_p) \in S$. Since this is true for every play induced by $\boldsymbol{\lambda}$, strategy $\boldsymbol{\lambda}$ is winning from n_0 in $\mathcal{T}$.

For the other direction, assume that $\boldsymbol{\lambda}$ is a winning coalition strategy from n_0 in $\mathcal{T}$. The tree will be the basis of a memory structure sufficient to win the game; we thus explain how histories in $\mathcal{G}$ can be mapped to nodes of $\mathcal{T}$. We first define a mapping $\mathsf{zip} : \mathsf{Hist}_{\mathcal{A}} \to \mathsf{Hist}_{\mathcal{A}}$ that summarizes any history in $\mathcal{A}$ to its *virtual history*, where each vertex appears at most once. Intuitively, zip greedily shortens a history by appropriately removing the loops until an unsafe vertex is encountered (if any). The mapping zip is defined inductively, starting with $\mathsf{zip}(v_0) = v_0$, and letting for every $h \in \mathsf{Hist}_{\mathcal{A}}$ and every $v' \in V$ such that $h \cdot v' \in \mathsf{Hist}_{\mathcal{A}}$,

$$\mathsf{zip}(h \cdot v') = \begin{cases} \mathsf{zip}(h) \cdot v' & \text{if } v' \text{ does not appear in } \mathsf{zip}(h) \\ v_0 \dots v' \sqsubseteq \mathsf{zip}(h) & \text{otherwise} \end{cases}$$

The mapping zip is well-defined: by construction, for every history h, any vertex appears at most once in $\mathsf{zip}(h)$, so that when v' appears in $\mathsf{zip}(h)$, there is a unique prefix of $\mathsf{zip}(h)$ ending with v'. Note that, since unsafe vertices are sinks, as soon as h reaches an unsafe vertex, the value of $\mathsf{zip}(h)$ stays unchanged.

Let us explain the zip function on an example.

Example 6.10. *We illustrate the* zip *function on the arena in Figure 6.1. Take* $h = v_0v_1v_0v_1$. *First,* $\mathsf{zip}(v_0) = v_0$; *then* $\mathsf{zip}(v_0v_1) = \mathsf{zip}(v_0) \cdot v_1 = v_0v_1$; $\mathsf{zip}(v_0v_1v_0) = v_0$ *(which is the unique prefix of* $\mathsf{zip}(v_0v_1) = v_0v_1$, *ending at* v_0*); finally* $\mathsf{zip}(v_0v_1v_0v_1) = \mathsf{zip}(v_0v_1v_0) \cdot v_1 = v_0v_1$.

We can then define a bijection between the safe nodes of $\mathcal{T}$ and virtual histories of $\mathcal{A}$ as follows.

Lemma 6.11. *The application* $\beta \colon \mathsf{N} \mapsto \mathsf{Hist}_{\mathcal{A}}$ *defined by* $\beta(\mathsf{n}) = \ell_{\mathsf{N}}(\iota(\mathsf{n}))$ *is a bijection between* $Y = \mathsf{N}_{\mathsf{int}} \cup \{\mathsf{n} \in \mathsf{N}_{\mathsf{leaf}} \mid \ell_{\mathsf{N}}(\mathsf{n}) \notin S\}$ *and the set* $Z = \{\mathsf{zip}(h) \mid h \in \mathsf{Hist}_{\mathcal{A}}\}$.

Proof. First, it is immediate that this application is injective, since two nodes of the tree in Y correspond to different histories in $\mathcal{A}$ which, due to construction of $\mathcal{T}$, do not have any repetition of vertices, therefore, belong to Z.

Moreover, the application β is surjective: pick $h \in Z$; then, h has no repetition; furthermore, it forms a real history in $\mathsf{Hist}_{\mathcal{A}}$, which implies that it can be read as the label of some history in the tree unfolding, either leading to an internal node (if h is safe) or an unsafe leaf (if h is unsafe). □

We write $\alpha = \beta^{-1}$. Using the zip function and α, from a coalition strategy $\boldsymbol{\lambda}$ in $\mathcal{T}$, we define a coalition strategy $\boldsymbol{\sigma}$ in $\mathcal{G}$ by applying $\boldsymbol{\lambda}$ to the nodes corresponding to virtual histories: for every history $h = v_0 \dots v_p$ in $\mathcal{G}$ we let $\boldsymbol{\sigma}(h) = \boldsymbol{\lambda}(\alpha(\mathsf{zip}(h))$ whenever

$\alpha(\mathsf{zip}(h)) \in \mathsf{N}_{\mathsf{int}}$ and $\boldsymbol{\sigma}(h)$ is set arbitrarily otherwise (recall that if $\alpha(\mathsf{zip}(h))$ is a leaf node, then h is already a losing history).

Towards a contradiction, assume that $\boldsymbol{\sigma}$ is not winning in $\mathcal{G}$. Consider, some number of agents $k \in \mathbb{N}_{>0}$, and a losing play with k agents: $\rho = v_0 v_1 \ldots \in \mathsf{Out}^k_{\mathcal{A}}(v_0, \boldsymbol{\sigma})$. Let $h' = v_0 v_1 \ldots v_q \sqsubseteq \rho$ be the shortest prefix of ρ ending in an unsafe vertex $v_q \notin S$, and write $\mathsf{zip}(h') = v_0 v_{i_1} \ldots v_q$ for the corresponding virtual history. By definition of $\boldsymbol{\sigma}$, $\mathsf{zip}(h')$ is a k-outcome of $\boldsymbol{\sigma}$ from v_0. Moreover, the corresponding play $R = \iota(\alpha(\mathsf{zip}(h'))) = \mathsf{n}_0 \mathsf{n}_{i_1} \ldots \mathsf{n}_q$ in $\mathcal{T}$, belongs to the k-outcome of $\boldsymbol{\lambda}$ from n_0. Since $v_q \notin S$, $\boldsymbol{\lambda}$ is not winning in $\mathcal{T}$, we reach a contradiction. We conclude that $\boldsymbol{\sigma}$ is a winning coalition strategy in $\mathcal{G}$. □

The rest of the chapter is dedicated to proving decidability and complexity results for safe coalition problem. We have seen in Lemma 6.9 that there is an equivalence between strategies in a safety game $\mathcal{G}$ and its finite unfolding tree $\mathcal{T}$. In the following, we give an algorithm to decide the existence of a winning coalition strategy for a safety objective on $\mathcal{T}$ that runs in exponential space in the size of $\mathcal{G}$.

6.3 An EXPSPACE upper bound for the safe coalition problem

In the previous section, we showed that the safe coalition problem reduces to solving the existence of a winning coalition strategy in the associated finite tree unfolding. To solve the latter, from the tree unfolding $\mathcal{T}$, we construct a deterministic (safety) automaton $\mathcal{B}$ over the alphabet Σ^m, where $m = |\mathsf{N}_{\mathsf{int}}|$, which accepts the ω-words corresponding to winning coalition strategies in $\mathcal{T}$. We will write $(\Sigma^\omega)^m$ to denote the set of m-tuples of ω-words over Σ. Then observe that sets $(\Sigma^m)^\omega$ and $(\Sigma^\omega)^m$ are in one-to-one correspondence: indeed, an infinite word $\mathbf{w} \in (\Sigma^m)^\omega$ corresponds to m infinite words $\mathbf{w}_\mathsf{n}$, one for each component, thus representing a word in $(\Sigma^\omega)^m$, and on the other hand, a word in $(\Sigma^\omega)^m$ can also be read componentwise. Hence, an ω-word in the language accepted by $\mathcal{B}$ also represents a word in $(\Sigma^\omega)^m$ and, the ω-word in each component describes a coalition strategy at the corresponding internal node of $\mathcal{T}$.

6.3.1 Construction of a safety automaton

Fix $\mathcal{G} = (\mathcal{A}, S)$ a parameterized safety game with $\mathcal{A} = \langle V, \Sigma, \Delta \rangle$ and $v_0 \in V$ an initial vertex. Recall that for every $(v, v') \in V \times V$ such that $\Delta(v, v') \neq \emptyset$, $\Delta(v, v')$ is described by an NFA over Σ, we assume they are given as inputs to the algorithm. We then first construct the corresponding complete deterministic finite automata (DFA's) for the languages. A complete DFA is a special kind of NFA, in which to each state and letter, the transition function assigns exactly one state.

Let $\mathcal{T} = \langle \mathsf{N}, \mathsf{E}, \ell_{\mathsf{N}}, \ell_{\mathsf{E}} \rangle$ be the associated unfolding tree with root n_0. For the rest of this section, we fix an arbitrary ordering on the internal nodes of $\mathcal{T}$ and on the edges: $\mathsf{N}_{\mathsf{int}} = \{\mathsf{n}_1, \ldots, \mathsf{n}_m\}$ and $\mathsf{E} = \{e_1, \ldots, e_r\}$, with $|\mathsf{N}_{\mathsf{int}}| = m$ and $|\mathsf{E}| = r$.

Assuming there are t leaves –thus t plays– in $\mathcal{T}$, for every $1 \leq i \leq t$, the i-th play is denoted $\mathsf{n}^i_0 \ldots \mathsf{n}^i_{z_i}$ with $\mathsf{n}^i_0 = \mathsf{n}_0$, $\forall j < z_i$, $\mathsf{n}^i_j \in \mathsf{N}_{\mathsf{int}}$ and $\mathsf{n}^i_{z_i} \in \mathsf{N}_{\mathsf{leaf}}$. Also, for $0 \leq j < z_i$, we note $e^i_j = (\mathsf{n}^i_j, \mathsf{n}^i_{j+1})$, the j-th edge in i-th play.

The automaton for the winning coalition strategies in $\mathcal{T}$ builds on the finite automata that recognize the regular languages that label edges of $\mathcal{T}$. For each edge $e \in \mathsf{E}$, we write $\mathcal{B}_e = (Q_e, \Sigma, \delta_e, q^0_e, F_e)$ for the complete DFA over Σ such that $L(\mathcal{B}_e) = \ell_{\mathsf{E}}(e)$. Here Q_e is the set of states, δ_e the transition function, $q^0_e \in Q_e$ the initial state and F_e the set of accepting states of $\mathcal{B}_e$. Note that some $\mathcal{B}_e$'s may be identical when they correspond to the same original transition in $\mathcal{G}$.

We then define a deterministic safety automaton $\mathcal{B} = (Q, \Sigma^m, \delta, q^0, F)$ that simulates all $\mathcal{B}_e$'s in parallel and accepts ω-words over alphabet Σ^m if every prefix satisfies the following: on every branch of the tree, if all corresponding $\mathcal{B}_e$'s accept, then the leaf is labelled by a safe vertex. The intuitive idea is as follows. First recall that the coalition has to win for every k, the number of agents; in particular, a prefix of length k of an ω-word $\mathbf{w}$ over Σ^m checks for the condition with k agents. A state q in $\mathcal{B}$ has r components, one for each edge of $\mathcal{T}$, and it reads a letter $\mathbf{u} \in \Sigma^m$, one for each node; the e-th component of q reads the n-th component of $\mathbf{u}$ if e is an outgoing edge of n. After reading a k-length word $\mathbf{w}_k$, if for some $e \leq r$, $\mathcal{B}_e$ (with $e = (\mathsf{n}, \mathsf{n}')$) is in a final state, it represents that at n, if the coalition plays n-th component of $\mathbf{w}$, with $[\mathbf{w}]_{\leq k} = \mathbf{w}_k$, then with k agents the game may move to n'. We can then identify the edges in a play in $\mathcal{T}$ and check for a word of length k if the following is satisfied: if corresponding $\mathcal{B}_e$'s along a play are in final states, then the play ends in a safe leaf. We then ensure in the accepting condition of $\mathcal{B}$ the above is true for every play in $\mathcal{T}$ and for every $k \in \mathbb{N}_{>0}$. The latter is represented as a Boolean formula.

Formally, $Q \subseteq Q_1 \times \ldots \times Q_r$ is the set of states; $q^0 = (q^0_1, \ldots, q^0_r)$ is the initial state; the transition relation δ executes the r automata componentwise: if letter $\mathbf{u} \in \Sigma^m$ is read, then make the s-th component mimic $\mathcal{B}_{e_s}$ by reading the l-th letter of $\mathbf{u}$, where l is the index (in the enumeration fixed above) of the source node of e_s; and the accepting set F is composed of all states $q = (q_1, \ldots, q_r)$ that satisfy the following Boolean formula:

$$\varphi = \bigwedge_{1 \leq i \leq t} \varphi_i \qquad \text{where } \varphi_i = \left(\left[\bigwedge_{0 \leq j < z_i} q_{e^i_j} \in F_{e^i_j} \right] \Rightarrow \ell_{\mathsf{N}}(\mathsf{n}^i_{z_i}) \in S \right) \tag{6.1}$$

First notice that since all the $\mathcal{B}_e$'s are deterministic, so is $\mathcal{B}$. Furthermore, $\mathcal{B}$ is equipped with a safety acceptance condition: an infinite run $\zeta = q^0 q^1 q^2 \ldots$ of $\mathcal{B}$ is accepting if for every $k \geq 1$, $q^k \in F$, and $L(\mathcal{B})$ consists of all words $\mathbf{w}$ whose unique corresponding run is accepting (this is a slight abuse of language since q^0 need not be in the accepting set F).

In Equation 6.1, φ_i corresponds to the condition for the i-th play $\mathsf{n}^i_0 \ldots \mathsf{n}^i_{z_i}$ with $\mathsf{n}^i_0 = \mathsf{n}_0$

and $\mathsf{n}^i_{z_i} \in \mathsf{N}_{\mathsf{leaf}}$. Intuitively, φ_i expresses that if for some number of agents k, the languages along the i-th maximal path contain the k-length prefixes of the corresponding ω-words (which means the induced play is k-realizable), then it should lead to a safe leaf ; and then φ ensures that this should be true for all plays. This is formalized in the next lemma.

Lemma 6.12. *Let $\boldsymbol{\lambda} : \mathsf{N}_{\mathsf{int}} \to \Sigma^\omega$ be a coalition strategy in $\mathcal{T}$. Then, $\boldsymbol{\lambda}$ is winning if and only if $(\boldsymbol{\lambda}(\mathsf{n}_1), \boldsymbol{\lambda}(\mathsf{n}_2), \ldots, \boldsymbol{\lambda}(\mathsf{n}_m)) \in L(\mathcal{B})$.*

Notice that in the above statement, we slightly abuse notation: $(\boldsymbol{\lambda}(\mathsf{n}_1), \boldsymbol{\lambda}(\mathsf{n}_2), \ldots, \boldsymbol{\lambda}(\mathsf{n}_m))$ belongs to $(\Sigma^\omega)^m$, however it uniquely maps to a word in $(\Sigma^m)^\omega$, the alphabet of $\mathcal{B}$.

Proof. Assume first $\boldsymbol{\lambda} : \mathsf{N}_{\mathsf{int}} \to \Sigma^\omega$ is a winning coalition strategy in $\mathcal{T}$, and consider the word $\mathbf{w} = (\boldsymbol{\lambda}(\mathsf{n}_1), \boldsymbol{\lambda}(\mathsf{n}_2), \ldots, \boldsymbol{\lambda}(\mathsf{n}_m))$. We show $\mathbf{w} \in L(\mathcal{B})$. Consider the infinite run $\zeta = q^0 q^1 \ldots$ of $\mathcal{B}$ on $\mathbf{w}$. Fix a number of agents $k \in \mathbb{N}_{>0}$. Since $\boldsymbol{\lambda}$ is winning, all plays in $\mathsf{Out}^k_{\mathcal{T}}(\mathsf{n}_0, \boldsymbol{\lambda})$ is winning. Therefore, for any $1 \leq i \leq t$ such that $\mathsf{n}^i_0 \ldots \mathsf{n}^i_{z_i}$ is in $\mathsf{Out}^k_{\mathcal{T}}(\mathsf{n}_0, \boldsymbol{\lambda})$, the play satisfies for all $0 \leq j < z_i$, $[\boldsymbol{\lambda}(\mathsf{n}^i_j)]_{\leq k} \in \ell_{\mathsf{E}}(e^i_j)$ and furthermore, $\ell_{\mathsf{N}}(\mathsf{n}^i_{z_i}) \in S$; and hence $q^k \models \varphi_i$. Otherwise, for some $i \leq t$, if the i-th play is not in $\mathsf{Out}^k_{\mathcal{T}}(\mathsf{n}_0, \boldsymbol{\lambda})$, then φ_i is vacuously true. We conclude $q^k \models \varphi$. Since this is true for every $k \in \mathbb{N}_{>0}$, $\mathbf{w} \in L(\mathcal{B})$.

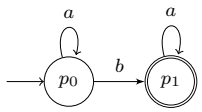
Now let $\boldsymbol{\lambda}$ be an arbitrary coalition strategy on $\mathcal{T}$, and $\mathbf{w} = (\boldsymbol{\lambda}(\mathsf{n}_1), \boldsymbol{\lambda}(\mathsf{n}_2), \ldots, \boldsymbol{\lambda}(\mathsf{n}_m)) \in L(\mathcal{B})$ with $\zeta = q^0 q^1 \ldots$ the accepting run on $\mathbf{w}$. Then for any number of agents $k \in \mathbb{N}_{>0}$, $q^k \in F$, and hence $q^k \models \varphi$. Therefore for all $1 \leq i \leq t$, $q^k \models \varphi_i$. Fix any such i; let $\mathsf{n}^i_0 \ldots \mathsf{n}^i_{z_i}$ be the i-th maximal path, and write $q^k = (q^k_1, \ldots, q^k_r)$. Then for $0 \leq j < z_i$, the condition $q^k_{e^j_i} \in F_{e^j_i}$ implies $[\boldsymbol{\lambda}(\mathsf{n}^i_j)]_{\leq k} \in \ell_{\mathsf{E}}(e^i_j)$. In case the above is true for all $0 \leq j < z_i$, we conclude $\mathsf{n}^i_0 \ldots \mathsf{n}^i_{z_i} \in \mathsf{Out}^k_{\mathcal{T}}(\mathsf{n}_0, \boldsymbol{\lambda})$ and φ_i ensures that $\ell_{\mathsf{N}}(\mathsf{n}^i_{z_i}) \in S$. Otherwise, $\mathsf{n}^i_0 \ldots \mathsf{n}^i_{z_i} \notin \mathsf{Out}^k_{\mathcal{T}}(\mathsf{n}_0, \boldsymbol{\lambda})$. Therefore, for $k \in \mathbb{N}_{>0}$, whenever a play is in $\mathsf{Out}^k_{\mathcal{T}}(\mathsf{n}_0, \boldsymbol{\lambda})$, it is winning. Since this is true for any number of agents k, $\boldsymbol{\lambda}$ is a winning coalition strategy in $\mathcal{T}$. □

Let us now illustrate the above construction on an example.

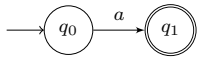
6.3.2 An example of the automaton construction

We illustrate here the construction of a safety automaton for the tree in Figure 6.2, the unfolding of the arena in Figure 6.1.

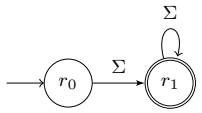
Example 6.13. *Figure 6.6 represents part of the automaton $\mathcal{B}$ corresponding to the tree $\mathcal{T}$ in Figure 6.2, the tree unfolding of the arena in Figure 6.1. The deterministic automata $\mathcal{B}_e$'s for the languages labelling the edges of $\mathcal{T}$ are depicted in Figure 6.5. Here notice that some $\mathcal{B}_e$'s in the picture may not be complete, however, we can naturally make them so by adding a sink state, here denoted '$\times$' for all of them (this is again an abuse of notation but is clear from context), and leading all unspecified transitions to that sink (as mentioned in Figure 6.5).*



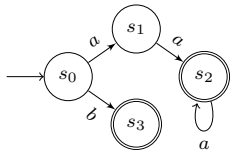
(a) Automaton for a^*ba^*.



(b) Automaton for a.



(c) Automaton for Σ^+.



(d) Automaton for $b \vee aa^+$.

Figure 6.5: Automata corresponding to the input languages of Figure 6.1. The automata are not complete for sake of readability; all unspecified letters lead to a (sink) non-accepting state '×'.

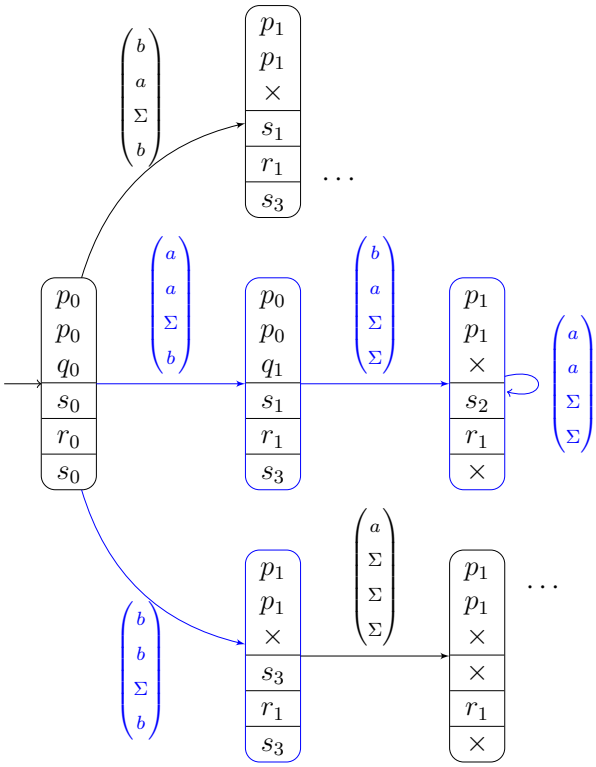


Figure 6.6: Automaton $\mathcal{B}$ corresponding to the tree given in Figure 6.2. Here we have only shown the accepting states (marked in blue) and some of the non-accepting states. Further explanations are given in Example 6.13.

Each state of $\mathcal{B}$ has as many components as the number of edges leading to a safe node in $\mathcal{T}$, we did not consider the edges leading to $\bot$. This is without loss of any generality: the language on any 'unsafe' edge leading to $\bot$, in this example, are disjoint from the languages on the edges leading to its siblings *(other children of its parent node). The first three positions in a state of $\mathcal{B}$, presented as a single cell in the picture, correspond to the outgoing edges of the* root n_0 *of $\mathcal{T}$ (hence they follow the same component – the first component – of Σ^m), and the other positions correspond to the other edges, in some chosen order - second cell corresponds to the edge $(\mathsf{n}_1, \mathsf{n}_0'')$, third cell corresponds to the outgoing edge from n_2, and the last cell corresponds to edge $(\mathsf{n}_1', \mathsf{n}_0''')$.*

We have only shown the accepting states (marked in blue) and some of the non-accepting states in the picture. Indeed, one can verify that the states which are coloured blue satisfy the formula φ; for instance, the state $(p_1, p_1, \times, s_2, r_1, \times)$ on the right corresponds to the two maximal paths in $\mathcal{T}$: $\mathsf{n}_0\mathsf{n}_0'$ (the edge described by the first position of that state) and $\mathsf{n}_0\mathsf{n}_1\mathsf{n}_0''$ (where $(\mathsf{n}_0, \mathsf{n}_1)$ corresponds to p_1 in the second position and $(\mathsf{n}_1, \mathsf{n}_0'')$ corresponds to s_2 in the second cell of that state). In both cases, the plays lead to safe leaves, hence that state satisfies φ.

Finally, the infinite run highlighted in blue is an accepting run of $\mathcal{B}$, and hence from

Lemma 6.12, it also corresponds to a winning coalition strategy in $\mathcal{T}$. In particular, the word read by the accepting run, i.e.*, in this example, any word in $(a,a,\Sigma,b)\cdot(b,a,\Sigma,\Sigma)\cdot(a,a,\Sigma,\Sigma)^\omega$ corresponds to a winning coalition strategy in the tree: for instance, $\boldsymbol{\lambda}(\mathsf{n}_0) = aba^\omega$; $\boldsymbol{\lambda}(\mathsf{n}_1) = a^\omega$; $\boldsymbol{\lambda}(\mathsf{n}_2) = a^\omega$; and $\boldsymbol{\lambda}(\mathsf{n}_1') = b^\omega$ is one of them (note here, for instance, that at node n_2, any word from Σ^ω could be played).*

We have constructed from a finite tree unfolding $\mathcal{T}$ a deterministic safety automaton $\mathcal{B}$ such that the accepting condition of $\mathcal{B}$ characterizes winning coalition strategies in $\mathcal{T}$. One then needs to check non-emptiness of $\mathcal{B}$ to decide if coalition has a winning strategy in $\mathcal{T}$. Next, we show that this algorithm runs in exponential space in size of $\mathcal{G}$.

6.3.3 An EXPSPACE upper bound

Fix a parameterized safety game $\mathcal{G} = (\mathcal{A} = \langle V, \Sigma, \Delta\rangle, S)$, $v_0 \in V$ an initial vertex and $\mathcal{T} = \langle \mathsf{N}, \mathsf{E}, \ell_{\mathsf{N}}, \ell_{\mathsf{E}}\rangle$ the tree unfolding of $\mathcal{G}$ w.r.t. v_0, rooted at n_0. As mentioned earlier, we assume that the arena $\mathcal{A}$ is initially given with all associated NFA's in the input. Then we prove the following theorem.

Theorem 6.14. *The safe coalition problem is in* EXPSPACE.

Proof. Solving the safe coalition problem reduces to checking non-emptiness of the language recognized by the deterministic safety automaton $\mathcal{B}$. We adapt to our setting the standard algorithm which runs in non-deterministic logarithmic space, when $\mathcal{B}$ is given as an input.

We write N for the number of states of $\mathcal{B}$ and show that N is doubly exponential in $|V|$, the number of vertices of the initial arena $\mathcal{A}$. Indeed, notice first that each state of $\mathcal{B}$ is of size exponential in $|V|$ since it has a component corresponding to each edge in $\mathcal{T}$. We know $|\mathsf{E}| = |\mathsf{N}| - 1$, therefore by Lemma 6.4, number of edges in $\mathcal{T}$ is at most $|V|^{|V|+1}$. Letting the size of each NFA representing the regular languages in $\mathcal{A}$ is bounded by $|Q_{\mathcal{A}}|$, the size of each $\mathcal{B}_e$, the corresponding DFA's, is then bounded by $2^{|Q_{\mathcal{A}}|}$. Then the number of possible states in $\mathcal{B}$ is bounded by $(2^{|Q_{\mathcal{A}}|})^{|V|^{|V|+1}}$ (*i.e.*, $2^{|Q_{\mathcal{A}}|}$ choices for each component), which is equal to $2^{|Q_{\mathcal{A}}|\cdot|V|^{|V|+1}}$. Although the size of $\mathcal{B}$ is doubly-exponential in $|V|$, we do not build $\mathcal{B}$ *a priori*. Instead, we non-deterministically guess a safe prefix of length at most N, and then a safe lasso on the last state of that prefix, again of length at most N. To guess the above, we only keep written two consecutive states of $\mathcal{B}$ and keep a counter to count up to N in each case. Then, provided one can check 'easily' whether a state of $\mathcal{B}$ is safe, the described procedure runs in non-deterministic exponential space, hence can be turned into a deterministic exponential space algorithm, by Savitch's theorem [Sav70].

It remains to explain how one checks that a given state in $\mathcal{B}$ is safe. Recall that the acceptance condition of $\mathcal{B}$ is described by a SAT formula φ. Notice that formula φ has size exponential in the size of $\mathcal{A}$. Indeed, there are exponentially many sub-formulas φ_i, one for each maximal path in $\mathcal{T}$. Since the hight of $\mathcal{T}$ is at most linear in size of $\mathcal{A}$, each φ_i performs linearly many checks (one for each play in $\mathcal{T}$); furthermore, each such check

can be done in constant time. Therefore, the satisfiability of φ can overall be checked in exponential time.

We therefore conclude that the safe coalition problem is in EXPSPACE. □

Remark here that in the above proof, we have assumed that the languages of $\mathcal{A}$ were described as NFA's. However, assuming their descriptions as DFA would not affect the exponential space upper bound for the safe coalition problem. Indeed, letting each $\mathcal{B}_e$ (DFA) has size at most $|q_{\mathcal{A}}|$, the bound on the possible number of states in $\mathcal{B}$ is $|q_{\mathcal{A}}|^{|V|^{|V|+1}}$, which is again exponential in $|V|$. The rest of the proof being the same, we conclude the safe coalition problem, when the input languages are described as DFA's, is in EXPSPACE.

We have presented an algorithm to solve the safe coalition game on the tree unfolding of a parameterized arena that runs in exponential space in the size of the arena. In the next section, we prove a lower bound result, namely that it is PSPACE-hard.

6.4 A PSPACE lower bound for the safe coalition problem

We show the safe coalition problem is PSPACE-hard by reduction from QBF-SAT, which is known to be PSPACE-complete [SM73]. The construction follows a similar pattern as in the proof of the lower bound results in Propositions 5.26 and 5.29. We formalize the result as follows.

Proposition 6.15. *The safe coalition problem is* PSPACE*-hard.*

Proof sketch. Let $\varphi = \exists x_1 \forall x_2 \exists x_3 \ldots \forall x_{2r} \cdot \big(C_1 \wedge C_2 \wedge \ldots \wedge C_m\big)$ be a quantified Boolean formula in prenex normal form, where for every $1 \leq h \leq m$, $C_h = \ell_{h,1} \vee \ell_{h,2} \vee \ell_{h,3}$ are the clauses, and for every $1 \leq j \leq 3$, $\ell_{h,j} \in \{x_i, \neg x_i \mid 1 \leq i \leq 2r\}$ are the literals.

In the reduction, we use sets of natural numbers (that represent the number of agents) corresponding to multiples of primes. Let thus p_i be the i-th prime number and P_i the set of all non-zero natural numbers that are multiples of p_i. For simplicity, we write a^{P_i} to denote the set of words in $(a^{p_i})^+$, that is words from a^+ whose length is a multiple of p_i. It is well-known that the i-th prime number requires $O(\log(i))$ bits in its binary representation, hence the description of each of the above languages is polynomial in the size of φ.

From φ, we construct a parameterized arena $\mathcal{A}_\varphi = \langle V, \Sigma, \Delta \rangle$ as follows:

- $V = \{v_0, v_1, \ldots, v_{2r-1}, v_{2r}\} \cup \{\mathsf{x}_1, \bar{\mathsf{x}}_1, \ldots, \mathsf{x}_{2r}, \bar{\mathsf{x}}_{2r}\} \cup \{\mathsf{C}_1, \mathsf{C}_2, \ldots, \mathsf{C}_m, \mathsf{C}_{m+1}\} \cup \{\bot, \top\}$, where we identify some vertices: $v_{2r} = \mathsf{C}_1$, and $\mathsf{C}_{m+1} = \top$.

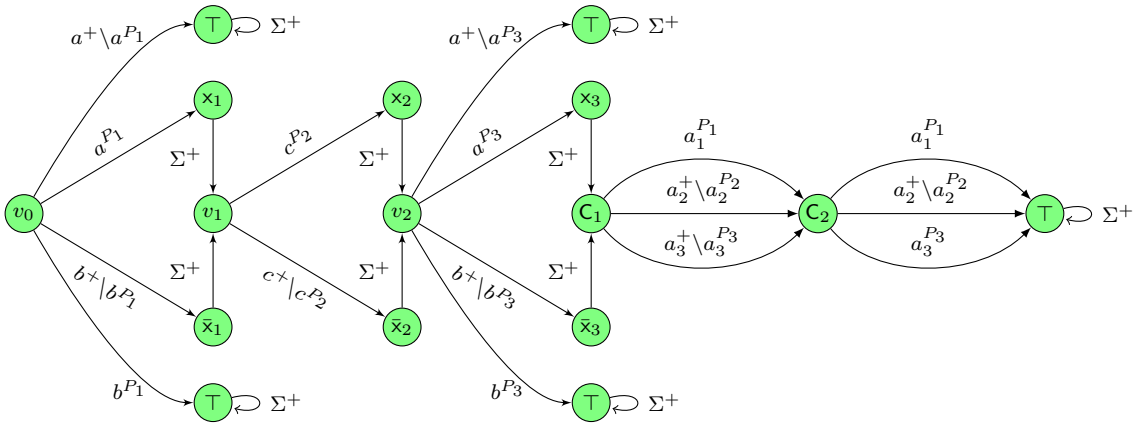


Figure 6.7: Parameterized arena for the formula $\varphi = \exists x_1 \forall x_2 \exists x_3 \cdot (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_1 \vee \neg x_2 \vee x_3)$. All unspecified transitions lead to the sink losing vertex $\bot$. Set P_i denotes multiples of the i-th prime number. Vertex x_i (resp., $\bar{\mathsf{x}}_i$) represents setting variable x_i to true (resp., false). For any play reaching C_1, for every i, the number of agents is in P_i iff the play went through x_i.

- $\Sigma = \{a, b, c\} \cup \bigcup_{1 \leq i \leq 2r} \{a_i\}$.
- For every $0 \leq s \leq r-1$, every $1 \leq i \leq 2r$ and every $1 \leq h \leq m$:
  1. $\Delta(v_{2s}, \mathsf{x}_{2s+1}) = a^{P_{2s+1}}$ and $\Delta(v_{2s}, \bar{\mathsf{x}}_{2s+1}) = b^+ \setminus b^{P_{2s+1}}$;
  2. $\Delta(v_{2s}, \top) = (a^+ \setminus a^{P_{2s+1}}) \cup b^{P_{2s+1}}$;
  3. $\Delta(v_{2s+1}, \mathsf{x}_{2s+2}) = c^{P_{2s+2}}$; and $\Delta(v_{2s+1}, \bar{\mathsf{x}}_{2s+2}) = c^+ \setminus c^{P_{2s+2}}$;
  4. $\Delta(\mathsf{x}_i, v_i) = \Sigma^+$ and $\Delta(\bar{\mathsf{x}}_i, v_i) = \Sigma^+$;
  5. $\Delta(\mathsf{C}_h, \mathsf{C}_{h+1}) = \bigcup_{1 \leq j \leq 3} L_{h,j}$ where $L_{h,j} = a_i^{P_i}$ if $\ell_{h,j} = x_i$; $L_{h,j} = a_i^+ \setminus a_i^{P_i}$ if $\ell_{h,j} = \neg x_i$.

To obtain a complete arena, all unspecified transitions lead to a sink vertex $\bot$.

On the arena $\mathcal{A}_\varphi$, we consider the safe coalition game $\mathcal{G}_\varphi = (\mathcal{A}_\varphi, S)$ with $S = V \setminus \{\bot\}$. The construction is illustrated on the simple formula $\varphi = \exists x_1 \forall x_2 \exists x_3 \cdot (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_1 \vee \neg x_2 \vee x_3)$ with 3 variables and 2 clauses in Figure 6.7. Only the safe vertices are depicted here.

From v_0, a first phase up to $v_{2r} = \mathsf{C}_1$ consists in choosing a valuation for the variables. The coalition chooses the truth values of existentially quantified variables x_{2s+1} in vertices v_{2s}: it plays a^ω for true, and b^ω for false. In the first (resp., second) case, if the number of agents involved in the coalition is (resp., is not) a multiple of p_{2s+1}, then the game proceeds to state x_{2s+1} (resp., $\bar{\mathsf{x}}_{2s+1}$) and from there on any choice of an ω-word, to v_{2s+1} for the next variable choice; otherwise the safe $\top$ state is reached (and stays forever).

For universally quantified variables the coalition must play c^ω in vertices v_{2s+1}, as any other choice would immediately lead to the sink losing vertex $\bot$; the choice of the

assignment then only depends on whether the number of agents involved in the coalition is a multiple of p_{2s+2} (in which case variable x_{2s+2} is assigned true) or not (in which case variable x_{2s+2} is assigned false).

Hence, depending on the number of agents involved in the coalition, either the play will proceed to state $v_{2r} = \mathsf{C}_1$, in which case the number of agents characterizes the valuation of the variables (it is a multiple of p_i if and only if variable x_i is set to true); or it will have escaped to the safe state $\top$.

Note that in terms of information, the coalition learns progressively assignments of variables, thanks to the visit to either vertex x_i or vertex $\bar{\mathsf{x}}_i$. Note also that the coalition can never learn assignments of next variables in advance - it can only know whether the number of players is a multiple of previously seen prime numbers, hence of previously quantified variables, not of variables quantified afterwards.

From C_1, a second phase starts where one checks whether the generated valuation makes all clauses in φ true. If it is the case, sequentially, the coalition chooses for every clause a literal that makes the clause true. The arena forces these choices to be consistent with the valuation generated in the first phase. For instance, on the example of Fig. 6.7, to set x_1 to true in the first phase, the coalition must play a^ω, and only plays with a number of agents in P_1 do not move to $\top$ and continue the first phase from x_1. Then, in the second phase, for instance for the first clause, one can choose literal $\ell_{1,1} = x_1$ by playing a_1^ω. The same language $-{a_1}^{P_1}-$ labels the edge from C_1 to C_2, so that the play proceeds to C_2. More generally, if a_i^ω leads from C_h to C_{h+1} with number of agents in P_i, this means that x_i was visited, hence indeed x_i was set to true. On the contrary, if a_i^ω leads from C_h to C_{h+1} with number of agents not in P_i, this means that $\bar{\mathsf{x}}_i$ was visited, hence indeed x_i was set to false.

The above idea yields the equivalence: there is a winning coalition strategy in the game $\mathcal{G}_\varphi = (\mathcal{A}_\varphi, S)$ if and only if φ is true, formalized in the following.

For $\ell_{h,j}$ a literal of φ, we write $\gamma(\ell_{h,j})$ for the constraint associated with $\ell_{h,j}$ in the second phase of the reduction. More precisely,

$$\gamma(\ell_{h,j}) = \begin{cases} P_i & \text{if } \ell_{h,j} = x_i \\ \neg P_i & \text{if } \ell_{h,j} = \neg x_i \end{cases}$$

We rely on the following correspondence between histories from v_0 to C_1 and valuations over $\{x_1, \cdots, x_{2r}\}$: the valuation ι_π associated with a history $\pi = v_0 v_1' v_1 v_2' v_2 \cdots \mathsf{C}_1$ is such that $\iota_\pi(x_i) = 1$ if $v_i' = \mathsf{x}_i$ and $\iota_\pi(x_i) = 0$ if $v_i' = \bar{\mathsf{x}}_i$. This correspondence defines a bijection, so that, given a valuation ι over $\{x_1, \cdots, x_{2r}\}$, there is a unique history $\pi_\iota = v_0 v_1' v_1 v_2' v_2 \cdots \mathsf{C}_1$ with $v_i' = \mathsf{x}_i$ if $\iota(x_i) = 1$, and $v_i' = \bar{\mathsf{x}}_i$ otherwise.

Moreover, after a finite history $\pi = v_0 v_1' v_1 v_2' v_2 \cdots \mathsf{C}_1$, the coalition can deduce the number of agents is in the set $K_\pi = \bigcap_{i \leq 2r} M_i$ where for each i, $M_i = P_i$ if $v_i' = \mathsf{x}_i$ and $M_i = \neg P_i$ if $v_i' = \bar{\mathsf{x}}_i$.

There is thus also a bijection between valuations and possible knowledges about the number of agents at C_1, so that we abusively write K_ι when ι is a fixed valuation over $\{x_1, \cdots, x_{2r}\}$. Note that for every $1 \leq i \leq 2r$, $K_\iota \cap \{P_i, \neg P_i\}$ is either a subset of P_i (when $\iota(x_i) = 1$) or a subset of $\neg P_i$ (when $\iota(x_i) = 0$). Also, an extension of a history $\pi = v_0 v_1' v_1 v_2' v_2 \cdots \mathsf{C}_1$ either leads to $\bot$, or continues in the main part of the game, in which case it does not refine the knowledge set K_ι further. We then show the following equivalence for the second phase of the reduction:

Lemma 6.16. *Let ι be a valuation of $\{x_1, \cdots, x_{2r}\}$. Then $\iota \models C_1 \wedge \cdots \wedge C_m$ if and only if coalition has a winning strategy in $\mathcal{G}_\varphi$ starting from C_1, if the number of agents k is in K_ι .*

Proof. Assume $\iota \models C_1 \wedge \cdots \wedge C_m$. For every $1 \leq h \leq m$, there is $\ell_{h,j}$ (literal of C_h) such that $\iota(\ell_{h,j}) = 1$. We define the following strategy $\boldsymbol{\sigma}$ for coalition: from vertex C_h, the strategy plays the word α_h^ω defined by $\alpha_h = a_i$ if $\ell_{h,j} = x_i$ or $\neg x_i$. By property on K_ι, $K_\iota \subseteq \gamma(\ell_{h,j})$. Thus, $\boldsymbol{\sigma}$ avoids $\bot$, and is therefore winning for every $k \in K_\iota$.

Conversely, pick a winning strategy $\boldsymbol{\sigma}$ for coalition from C_1 against K_ι. This strategy plays uniformly for all $k \in K_\iota$, say coalition chooses ${\alpha_h}^\omega$ from C_h. If for some $1 \leq i \leq 2r$, $\alpha_h = a_i$, then either x_i or $\neg x_i$ is a literal of C_h. If $\Delta(\mathsf{C}_h, \mathsf{C}_{h+1})$ is described by ${a_i}^{P_i}$, then x_i is the literal in C_h and hence, $K_\iota \subseteq \gamma(x_i)$; the construction guarantees that $\iota(x_i) = 1$. A similar reasoning applies to the case where $\Delta(\mathsf{C}_h, \mathsf{C}_{h+1})$ is described by $a_i^+ \setminus {a_i}^{P_i}$ to show $\iota(x_i) = 0$. In all cases, C_h is satisfied by ι. □

Lemma 6.16 formalizes the link between winning strategies for coalition in the second phase (that is, from C_1) and satisfying valuations. It remains to relate strategies for coalition in the first phase and decision functions for the variable quantifications in φ.

The two-player game underlying the quantifications of φ coincides with the coalition game played in the first phase. In particular, strategies coincide in the two games. Fix a coalition strategy $\boldsymbol{\sigma}$ in the quantification game or equivalently in the first phase. For every valuation ι which is generated by $\boldsymbol{\sigma}$ in the quantification game, there is an outcome π ending in C_1 such that $\iota = \iota_\pi$. Conversely, for every outcome π ending in C_1, ι_π is generated by $\boldsymbol{\sigma}$ in the quantification game. □

We have proved an EXPSPACE upper bound and a PSPACE lower bound for the safe coalition game. In the next section, we give an algorithm to synthesize a winning coalition strategy for a safe coalition game, if exists. We will use the tree unfolding construction and, in particular, we show that the tree can be seen as a memory structure. We show such a strategy can be computed in exponential space. Furthermore, we show there exists games where exponential memory is necessary to win.

6.5 Synthesizing a winning coalition strategy

Fix a parameterized safety game $\mathcal{G} = (\mathcal{A} = \langle V, \Sigma, \Delta\rangle, S)$, $v_0 \in V$ an initial vertex and $\mathcal{T} = \langle \mathsf{N}, \mathsf{E}, \ell_{\mathsf{N}}, \ell_{\mathsf{E}}\rangle$ the tree unfolding of $\mathcal{G}$ w.r.t. v_0, rooted at n_0. Further, N is divided into disjoint union of $\mathsf{N}_{\mathsf{int}}$ and $\mathsf{N}_{\mathsf{leaf}}$. We denote by m the number of nodes and by r the number of edges in $\mathcal{T}$. Recall that an ordering of the internal nodes and edges of $\mathcal{T}$ are fixed. We further recall the construction of the automaton $\mathcal{B}$ from the tree. First notice that from an accepting word of the form $\mathbf{u}\cdot\mathbf{v}^{\omega}$ in $L(\mathcal{B})$, where $\mathbf{u} \in \left(\Sigma^m\right)^*$ and $\mathbf{v} \in \left(\Sigma^m\right)^+$, one can synthesize a winning strategy $\boldsymbol{\lambda}$ in $\mathcal{T}$ by:

$$\boldsymbol{\lambda}(\mathsf{n}_i) = \mathbf{u}_i \cdot \mathbf{v}_i^{\omega} \quad \text{for every } 1 \leq i \leq m,$$

where $\mathsf{n}_i \in \mathsf{N}_{\mathsf{int}}$ is the i-th internal node of $\mathcal{T}$ in the fixed ordering. Then it is easy to transfer the coalition strategy $\boldsymbol{\lambda}$ in $\mathcal{T}$ to $\boldsymbol{\sigma}$ a coalition strategy in $\mathcal{G}$ by defining

$$\boldsymbol{\sigma}(h) = \boldsymbol{\lambda}(\alpha(\mathsf{zip}(h))) \quad \text{for every history } h \in \mathsf{Hist}_{\mathcal{A}},$$

that is, the ω-word corresponding to the internal node representing its virtual history. Using the result of Lemma 6.9, $\boldsymbol{\sigma}$ is a winning coalition strategy in $\mathcal{G}$. Recall here that, zip assigns to every history its virtual history (by greedily removing all the loops) and α associates to a virtual history its corresponding node in the tree $\mathcal{T}$.

We present here the main result of this section.

Proposition 6.17. *Let $\mathcal{G} = (\mathcal{A}, S)$ be a parameterized safety game with $\mathcal{A} = \langle V, \Sigma, \Delta\rangle$ and $v_0 \in V$ an initial vertex. Then, if there is a winning coalition strategy in $\mathcal{G}$, then there is one which uses exponential memory, and can be computed in exponential space. Furthermore, winning might require exponential memory.*

Proof. The tree unfolding can be seen as a memory structure for a winning strategy. Indeed, consider the Mealy automaton $\mathcal{M}$ with set of states $\mathtt{M} = \mathsf{N}_{\mathsf{int}}$, and initial memory state $m_0 = \mathsf{n}_0$. Define the application $\mathsf{upd} : \mathsf{N}_{\mathsf{int}} \times V \to \mathsf{N}_{\mathsf{int}}$ by $\mathsf{upd}(\mathsf{n}, v') = \mathsf{n}'$ such that for $v' \in S$,

- either $\mathsf{n}' \in \mathsf{N}_{\mathsf{int}}$ is a child of n with $\ell_{\mathsf{N}}(\mathsf{n}') = v'$,
- or $\mathsf{n}' \in \mathsf{N}_{\mathsf{int}}$ is an ancestor of $\mathsf{n}'' \in \mathsf{N}_{\mathsf{leaf}}$ such that $\ell_{\mathsf{N}}(\mathsf{n}'') = \ell_{\mathsf{N}}(\mathsf{n}') = v'$, and n'' is a child of n,

and for $v' \notin S$,

- either $\mathsf{n}' \in \mathsf{N}_{\mathsf{leaf}}$ is a child of $\mathsf{n} \in \mathsf{N}_{\mathsf{int}}$ with $\ell_{\mathsf{N}}(\mathsf{n}') = v'$,
- or $\mathsf{n}, \mathsf{n}' \in \mathsf{N}_{\mathsf{leaf}}$, and $\mathsf{n}' = \mathsf{n}$.

We also define the application $\mathsf{act} : \mathsf{N}_{\mathsf{int}} \times V \to \Sigma^\omega$ by $\mathsf{act}(\mathsf{n}, v) = \boldsymbol{\lambda}(\mathsf{n})$, where $\boldsymbol{\lambda}$ is a winning coalition strategy in $\mathcal{T}$ extracted from $L(\mathcal{B})$.

We recall that a Mealy automaton $\mathcal{M}$ describes a strategy $\sigma_{\mathcal{M}}$ from v_0 as follows: first, for any $h \in \mathsf{Hist}_{\mathcal{A}}$, inductively define $\mathtt{m}[h] \in \mathtt{M}$ by $\mathtt{m}[v_0] = m_0$, and $\mathtt{m}[h \cdot v] = \mathsf{upd}(\mathtt{m}[h], v)$; and then $\sigma_{\mathcal{M}}$ is defined as $\sigma_{\mathcal{M}}(h) = \mathsf{act}(\mathtt{m}[h], \mathsf{last}(h))$, where $\mathsf{last}(h)$ is the last vertex of history h.

Intuitively, the upd function updates the memory according to the zip function that it computes on-the-fly: whenever a safe vertex is visited, it computes the virtual history and finds the node in $\mathcal{T}$ corresponding to that virtual history; the tag function then assigns the ω-word according to strategy $\boldsymbol{\lambda}$ on that particular node of $\mathcal{T}$. More precisely, the second condition in the definition of upd, when $v' \in S$, makes use of the idea that if a vertex is repeated in a history of $\mathcal{G}$, the strategy plays as in the first occurrence of that vertex. We now formally show that $\mathcal{M}$ indeed induces a winning strategy, in particular, we show $\sigma_{\mathcal{M}} = \boldsymbol{\sigma}$ where $\boldsymbol{\sigma}$ is defined from $\boldsymbol{\lambda}$ by $\boldsymbol{\sigma}(h) = \boldsymbol{\lambda}(\alpha(\mathsf{zip}(h)))$, for every history $h \in \mathsf{Hist}_{\mathcal{A}}$.

Lemma 6.18. *$\sigma_{\mathcal{M}} = \boldsymbol{\sigma}$, where $\boldsymbol{\sigma}$ is defined from $\boldsymbol{\lambda}$ by $\boldsymbol{\sigma}(h) = \boldsymbol{\lambda}(\alpha(\mathsf{zip}(h)))$, for every history $h \in \mathsf{Hist}_{\mathcal{A}}$, and $\sigma_{\mathcal{M}}$ the strategy induced by the Mealy automaton $\mathcal{M}$ defined earlier.*

Proof. We shall show for every history $h \in \mathsf{Hist}_{\mathcal{A}}$, $\sigma_{\mathcal{M}}(h) = \boldsymbol{\sigma}(h)$. By definition, $\sigma_{\mathcal{M}}(h) = \mathsf{act}(\mathtt{m}[h], \mathsf{last}(h)) = \boldsymbol{\lambda}(\mathtt{m}[h])$; on the other side, $\boldsymbol{\sigma}(h) = \boldsymbol{\lambda}(\alpha(\mathsf{zip}(h)))$. Therefore, we need to show the following equivalence: for any $h \in \mathsf{Hist}_{\mathcal{A}}$, $\mathtt{m}[h] = \alpha(\mathsf{zip}(h))$.

We show the equivalence by induction on the structure of h. The base case is trivial: $h = v_0$, then $\mathtt{m}[h] = m_0 = \alpha(v_0)$. Now suppose the statement is true for every prefix of $h \in \mathsf{Hist}_{\mathcal{A}}$ and shall show for $hv' \in \mathsf{Hist}_{\mathcal{A}}$ such that $v' \in V$. We need to show, $\mathtt{m}[hv'] = \alpha(\mathsf{zip}(hv'))$.

We let $\alpha(\mathsf{zip}(h)) = \mathsf{n}$; then $\beta(\mathsf{n}) = \mathsf{zip}(h)$ (β is the inverse mapping of α) which, by definition of β, implies $\ell_{\mathsf{N}}(\iota(\mathsf{n})) = \mathsf{zip}(h)$ (recall that ι is the trivial bijection that maps a node in $\mathcal{T}$ to the unique history $\mathsf{n}_0 \dots \mathsf{n}$). We shall use this relation in the proof. We first deduce the following:

$$\mathtt{m}[hv'] = \mathsf{upd}(\mathtt{m}[h], v') = \mathsf{upd}(\alpha(\mathsf{zip}(h)), v') = \mathsf{upd}(\mathsf{n}, v'). \tag{6.2}$$

The first equality is by definition of upd, and the second equality is due to the induction hypothesis. Denote $\mathsf{upd}(\mathsf{n}, v')$ by n'. Therefore we need to show: $\alpha(\mathsf{zip}(hv')) = \mathsf{n}'$.

1. First let $v' \in S$. We consider two following cases:

   - assume $\mathsf{n}' \in \mathsf{N}_{\mathsf{int}}$ is a child of n with $\ell_{\mathsf{N}}(\mathsf{n}') = v'$. Then

   $$\beta(\mathsf{n}') = \ell_{\mathsf{N}}(\iota(\mathsf{n}')) = \ell_{\mathsf{N}}(\iota(\mathsf{n})) \cdot \ell_{\mathsf{N}}(\mathsf{n}') = \mathsf{zip}(h) \cdot v' \tag{6.3}$$

The second equality holds because $\iota(\mathsf{n}') = \mathsf{n}_0 \dots \mathsf{n}.\mathsf{n}'$. Since β is a bijection from set Y to Z (recall, $Y = \mathsf{N}_{\mathsf{int}} \cup \{\mathsf{n} \in \mathsf{N}_{\mathsf{leaf}} \mid \ell_{\mathsf{N}}(\mathsf{n}) \notin S\}$ and $Z = \{\mathsf{zip}(h) \mid h \in \mathsf{Hist}_{\mathcal{A}}\}$), $\mathsf{zip}(h) \cdot v'$ is a virtual history in Z and hence, v' does not appear in $\mathsf{zip}(h)$. Therefore, we can write $\mathsf{zip}(h) \cdot v' = \mathsf{zip}(hv')$. We can then rewrite Equation 6.3 as $\alpha(\mathsf{zip}(hv')) = \mathsf{n}'$.

- assume n has a child $\mathsf{n}'' \in \mathsf{N}_{\mathsf{leaf}}$ and $\mathsf{n}' \in \mathsf{N}_{\mathsf{int}}$ is an ancestor of n'' with $\ell_{\mathsf{N}}(\mathsf{n}'') = \ell_{\mathsf{N}}(\mathsf{n}') = v'$. Then,

$$\beta(\mathsf{n}') = \ell_{\mathsf{N}}(\iota(\mathsf{n}')) = \ell_{\mathsf{N}}(\mathsf{n}_0 \dots \mathsf{n}') = \ell_{\mathsf{N}}(\mathsf{n}_0) \dots \ell_{\mathsf{N}}(\mathsf{n}'). \tag{6.4}$$

Also,

$$\beta(\mathsf{n}) = \ell_{\mathsf{N}}(\mathsf{n}_0 \dots \mathsf{n}) = \ell_{\mathsf{N}}(\mathsf{n}_0) \dots \ell_{\mathsf{N}}(\mathsf{n}). \tag{6.5}$$

Since n' is an ancestor of n, $\ell_{\mathsf{N}}(\mathsf{n}')$ appears in $\beta(\mathsf{n})$. By induction hypothesis, the latter is equal to $\mathsf{zip}(h)$, and hence, $\ell_{\mathsf{N}}(\mathsf{n}')$ appears in $\mathsf{zip}(h)$. Moreover, since $\ell_{\mathsf{N}}(\mathsf{n}_0) \dots \ell_{\mathsf{N}}(\mathsf{n}')$ is an element in Z, no vertices repeat, therefore, $\mathsf{zip}(hv')$ is equal to $\ell_{\mathsf{N}}(\mathsf{n}_0) \dots \ell_{\mathsf{N}}(\mathsf{n}')$ that is same as $\beta(\mathsf{n}')$, by Equation 6.4.

2. If $v' \notin S$, we again consider two cases.

   - If $\mathsf{n}' \in \mathsf{N}_{\mathsf{int}}$ is a child of $\mathsf{n} \in \mathsf{N}_{\mathsf{int}}$ with $\ell_{\mathsf{N}}(\mathsf{n}') = v'$, the proof is similar to the first subcase of when $v \in S$.
   - If $\mathsf{n} = \mathsf{n}'$ and both are leaf nodes, n must be such that $\ell_{\mathsf{N}}(\mathsf{n}) \notin S$. Since unsafe vertices in $\mathcal{G}$ are sinks, $\mathsf{zip}(hv') = \mathsf{zip}(h) = \beta(\mathsf{n}) = \beta(\mathsf{n}')$.

Therefore, we conclude in all cases, $\mathtt{m}[hv'] = \alpha(\mathsf{zip}(hv'))$. By induction hypothesis, for any $h \in \mathsf{Hist}_{\mathcal{A}}$, $\sigma_{\mathcal{M}}(h) = \boldsymbol{\sigma}(h)$, and hence, $\sigma_{\mathcal{M}} = \boldsymbol{\sigma}$. □

We now show that a coalition winning strategy in $\mathcal{G}$ can be synthesized using exponential space. Recall from the proof of Theorem 6.14 that finding an accepting run of $\mathcal{B}$, if it exists, requires exponential space in the size of $\mathcal{G}$: one only needs to guess such a run on-the-fly storing only two consecutive states of $\mathcal{B}$. One can then extract a winning strategy for the coalition in $\mathcal{T}$ from the accepting run as described above; the words $\mathbf{u}$ and $\mathbf{v}$ may have size doubly-exponential in the size of $\mathcal{G}$ (because of the size of $\mathcal{B}$), however, again, one only needs to keep written two consecutive letters at a time. Therefore, the computation of $\boldsymbol{\lambda}$ needs exponential space and since the Mealy automaton $\mathcal{M}$ builds on $\boldsymbol{\lambda}$, it also needs exponential space. In particular, the definition of the act function uses $\boldsymbol{\lambda}$ and takes exponential space, also the set of memory states $\mathtt{M}$ and the upd function only use the tree structure, which is of size exponential in $|\mathcal{G}|$. We conclude, overall, the Mealy automaton can be computed in exponential space.

For the lower bound on the size of memory of a winning coalition strategy, we show the following lemma.

Lemma 6.19. *There is a family of games $(\mathcal{G}_n)_n$ such that the size of $\mathcal{G}_n$ is polynomial in n but winning coalition strategies require exponential memory.*

Proof. We again consider the game of Figure 6.3, described formally in the proof of Lemma 6.4. whose description can be made in polynomial time (since the i-th prime number uses only $\log(i)$ bits in its binary representation). We have already seen that its tree unfolding has exponential size. We will argue why exponential memory is required, that is, any winning strategy for coalition would require an exponential size memory.

First notice that there is a winning coalition strategy: play a^ω at every vertex B_i, and a^ω (resp., b^ω) at vertex C_i if the history went through v_i (resp., $\bar{v}_i$). This strategy can be implemented using the memory given by its tree unfolding, part of which was depicted in Figure 6.4.

Assume now one can do better and have a memory structure of size strictly smaller than 2^n. Since there are 2^n many histories from v_0 leading to vertex C_1, there exist two different histories h and h', arriving in C_1, that lead to the same memory state, and hence all the suffixes of h, h' also lead to same memory states. In particular, the memory states reached by extensions of h and h' to vertices C_1, C_2, ..., C_n are the same, and hence the coalition strategy will select exactly the same ω-words in all vertices C_1, C_2, ..., C_n. Consider ρ, ρ' two plays in $\mathcal{A}_n$ that are extensions of h, h' respectively. Since the two histories disagree at least on a predicate "be a multiple of the i-th prime number", the same ω-words from all vertices C_1, C_2, ..., C_n cannot be winning for both of the plays. This is a contradiction to the assumption that the size of the memory is smaller than 2^n. □

This concludes the proof of Proposition 6.17. □

We have shown that the safe coalition problem can be solved in exponential space in the size of input. We have further shown that the problem is PSPACE-hard. Moreover, if there is a coalition winning strategy, then there is one that uses exponential size memory, the exponential size in memory is somehow optimal: there exists a family of games, whose size is linear in n, such that any winning strategy requires a memory of size $O(2^n)$. Furthermore, we have seen a procedure to synthesize a winning strategy, if it exists, that uses exponential space in the size of input.

6.6 Concluding remarks

In this chapter, we considered the decision problem of the existence of a winning strategy for coalition on a parameterized arena such that the players collectively achieve a safety objective. We first show that such a game can be reduced to a coalition game on a finite tree, which is an unfolding of the arena, such that the coalition has a winning strategy in the parameterized arena if and only if they have one in the latter. We then show that the safe coalition problem on the tree is decidable. We construct a deterministic finite automaton $\mathcal{B}$ equipped with a safety acceptance condition that keeps track of the languages on the edges of the arena in parallel, and show that an accepting run of $\mathcal{B}$

corresponds to a winning coalition strategy on the tree and vice-versa. A strategy for coalition on the tree can then easily be transferred to the parameterized arena. The automaton constructed above can be doubly-exponential in size of the input, however, we show that the algorithm that non-deterministically guesses an accepting run of $\mathcal{B}$ is sufficient to solve the safe coalition problem which runs in exponential space. A PSPACE lower bound result is then shown by a reduction from QBF-SAT, which is known to be complete for that complexity class. Finally, we have shown that one can synthesize a winning strategy, when it exists, from an accepting run of $\mathcal{B}$, and compute it using exponential space. In particular, the tree unfolding of an arena can be seen as a memory structure associated to a strategy for coalition on a parameterized arena, which has size at most exponential in the size of the arena, moreover, the exponential bound on the memory size is optimal, as witnessed in Lemma 6.19. This work opens roads towards various research directions. Let us discuss some of them here.

Tight complexity bounds for safe coalition problem

The coalition problem on parameterized arenas with safety objective can be solved in exponential space in the size of the arena and is PSPACE-hard. However, no tight complexity bound is shown in this chapter. The EXPSPACE upper bound is naive, the automaton $\mathcal{B}$ described in Section 6.3 keeps track of all maximal paths in the exponential size tree unfolding of the arena. It is not clear whether every state in automaton $\mathcal{B}$ is indeed visited, which is not the case, for instance, in Example 6.13 (also see Figure 6.6). Closing the gap in complexities for the safe coalition problem and proving tight bounds is left open in this thesis, which we would like to pursue as a possible future research direction.

Beyond safety

We have considered safety objectives for coalition in a parameterized game and shown an EXPSPACE upper bound and a PSPACE lower bound for the coalition game problem. Further, one can synthesize a winning strategy, if it exists, which is of size at most exponential in the size of the input. A natural question is then to ask whether the decidability result still holds when the winning condition for the coalition is described by, for instance, a reachability objective.

A natural approach for solving the coalition game with different objectives would be to construct a tree unfolding of the arena, similar to the one used for safety objective. However, for other objectives, the tree needs not necessarily be finite. The finitary condition was ad hoc to the safety objectives, where the coalition can play the same ω-word every time a certain vertex is visited with no loss of generality. This might not be the case, for instance, for a reachability objective. Let us illustrate the scenario over an example.

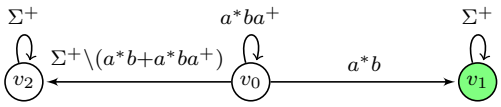


Figure 6.8: Example of a reachability coalition game: a winning coalition strategy is that agent i plays a for the first $i-1$ rounds, then b for one round, and finally a forever.

Consider the coalition game on the arena depicted in Figure 6.8. In this example, $\mathcal{A} = \langle V, \Sigma, \Delta \rangle$ where $V = \{v_0, v_1, v_2\}$, $\Sigma = \{a, b\}$, and regular languages labelling the edges are depicted in the picture. Consider a coalition game with a reachability objective described by $F = \{v_1\}$, depicted in green in the picture, and v_0 the initial vertex. Notice that coalition can win the game for any number of agents only if, at v_0, *the last* agent involved plays a b whereas all the others play a. On the other hand, at v_0, it is safe if exactly one agent plays a b and others play a - in that case, the game stays at v_0.

One can verify that the coalition has a winning strategy in this reachability game: agent i plays action a for the first $i-1$ rounds, then plays b, and finally plays a for the remaining steps. Doing so, each agent will in turn play action b, and when the last agent does so, the play will reach v_1. Notice that the order of identifiers of the agents is crucial here and is used in defining a coalition strategy.

Notice that, for each agent, the strategy informally described above when going through v_0 differs at some step (i-th step for agent i), and hence, same for the coalition - it chooses a different ω-word at each visit of v_0. In fact, every winning strategy must have this property in the sense that if a strategy always plays the same ω-word at each visit of v_0, in this example, the strategy is never winning. Indeed, to reach v_1, each agent has to play b at some point; if $a^k ba^\omega$ is chosen every time it visits v_0 (for a fixed k), the game is not winning when there are more (or, less) than k agents. Therefore, a tree unfolding of the arena must take into account this fact and hence, the stopping criterion used for a safety objective will no longer hold, inducing an infinite size unfolding tree. It is therefore not immediate whether the reachability coalition problem is decidable. Exploring decidability and, further, synthesizing automatically winning coalition strategies, when they exist, in such games is a challenging direction of research to pursue.

Chapter 7

Conclusion

This chapter summarizes the contributions of this thesis and discusses possible directions for future work. We considered two models in the context of parameterized verification. In Part I, we focused on *broadcast protocols*, an automata-based model representing the main characteristics of ad hoc networks, originally proposed in [DSZ10], whereas Part II introduces *parameterized games*, an extension of concurrent games, with parameterized number of players. Each chapter contains a separate conclusion, and in particular, Chapters 3, 5 and 6 mention some future research directions, whereas here we discuss more general perspectives for future work.

7.1 Summary of contributions and immediate follow-ups

7.1.1 Broadcast protocols

In Chapter 2, we first described the syntax and various semantics of broadcast protocols. We follow, among many formalisms, the model described in [DSZ10]. We recalled the static, reconfigurable and loss-on-reception semantics from the literature, and additionally, we introduced *loss-on-broadcast* semantics to model arbitrary message losses during broadcast. We then recalled the *coverability* problem for broadcast protocols, that, given a state, asks if there exists an initial configuration and an execution such that at least one process in the final configuration is in that given state. In reconfigurable semantics, the above problem is decidable, and we recalled a saturation algorithm, that runs in polynomial time [DSTZ12]. We then introduced two notions: *cutoff* and *covering length*, the former being the minimal number of nodes and the latter the minimal length of a covering execution for positive instances of the coverability problem.

The main contribution of this part of the thesis was to prove tight bounds on the cutoff and the covering length for positive instances of coverability in reconfigurable and lossy semantics. We detailed the results in Chapter 3. We have shown a linear tight bound

for cutoff and a quadratic tight bound for covering length in both semantics. The upper bounds are achieved by exploiting a monotonicity property of the networks, called *copycat property*, which states that any node in a reachable configuration can be duplicated, *i.e.*, for any reachable configuration and for any state present in that configuration, one can construct another execution with one more process in that state, while preserving the behaviours of the other nodes. Based on the copycat property, we first refined the saturation algorithm of [DSTZ12] in a simple manner that also keeps track of the size of a minimal execution that covers any state in the set returned by the algorithm, and then by a fine analysis of the algorithm, we proved the upper bounds. We further showed that these bounds are tight: we exhibited a family of protocols that achieves those bounds. However, we showed that the executions in the loss-on-broadcast semantics can in general be more succinct: we exhibited a family of protocols for which, in reconfigurable semantics, the cutoff is 3, yet in loss-on-broadcast semantics, the cutoff is linear in the size of the protocol. Finally, we showed NP-completeness of the problem of determining the exact cutoff of a protocol in both semantics.

As mentioned in the conclusion of Chapter 3, investigating the interplay between these two notions is one of our main interests. Coming up with approximation schemes that run in polynomial time for determining cutoff (or, proving an inapproximability result) is of independent interest that has potential to produce efficient algorithms for parameterized model checking of broadcast protocols, up to some non-optimality. We mention more general perspectives for future works in Section 7.2.

7.1.2 Parameterized games

In the second part of the thesis, we studied a model of games on finite graphs where the number of players is a parameter. It is an extension of two-player concurrent games to the parameterized setting. In contrast to the model considered in the first part, here the players do not directly communicate with each other but have identifiers that we implicitly use to define the model. Let us summarize the results we have achieved in this part.

In Chapter 4, we defined *parameterized games*. In such games, we label the edges of a graph with languages (regular languages in this work) of words of finite but unbounded length. A word of length k represents the interaction between k players. The players, however, do not know *a priori* the number of players participating in the game, but they can infer some knowledge on that as the game proceeds. The arena can be non-deterministic, that is, a word can lead from a vertex to more than one vertices. The non-determinism and the number of players are selected by an adversarial environment. We then considered two independent settings on this model of games, in the first setting, a distinguished player wants to achieve a goal against any strategies of her opponents and of the environment, and the second one is a coalition setting, where the players play as a coalition and try to achieve a common goal against the environment.

We presented our results on the decision problem of existence of a winning strategy for the distinguished player, called Eve, against any strategies of her opponents in the first

game setting in Chapter 5. Eve, in this case, has to play uniformly, against any number of opponents, which is *a priori* unknown to her. We showed that such games reduce to a simpler setting where the languages on the transitions only constrain the number of opponents of Eve. We then solved the decision problem in the latter. We called such a game a *semi-parameterized game*. The decidability of the semi-parameterized game problem is shown by a reduction of the game to a two-player turn-based game, called *knowledge game*, that keeps track of Eve's knowledge of her number of opponents in the original game. The knowledge game abstraction is correct for any winning objective, which proves that the semi-parameterized game problem is decidable. However, we are interested in proving complexity results for the above, and we restricted the winning condition to reachability objectives. We distinguished various cases depending on the type of constraints in the arena, namely whether they are (finite unions of disjoint) intervals or semilinear sets. We showed that the problem is PSPACE-complete when constraints are semilinear sets or finite unions of intervals, NP-complete for deterministic arenas with constraints finite unions of intervals, and PTIME-complete with interval constraints. Since the projection of a regular language to the lengths of words is a semilinear set, this shows PSPACE-completeness for the parameterized game problem in this setting. As mentioned in the conclusion of that chapter, investigating the game problem for other winning objectives and proving complexity bounds would be an immediate follow-up.

Chapter 6 presents our results on the coalition setting. We focused on safety objectives, and showed that the safety coalition problem is decidable. We give an algorithm that runs in exponential space in the size of the input. The algorithm consists of two major steps. First, we constructed a finite tree unfolding of the game arena and showed the correctness of this unfolding; and second, we constructed an automaton that runs the automata for the languages on the edges in parallel to capture a winning strategy for coalition on the unfolding tree. We set the acceptance condition of the automaton as a safety condition described by a Boolean formula and showed correctness of the construction. Then existence of a winning coalition strategy amounts to checking if there is an accepting run, which is a composition of a finite path and a safe lasso, in that automaton. The latter can be decided in exponential space in the size of the input arena. A PSPACE lower bound was also shown. Finally, we described a procedure to synthesize a winning strategy for the safe coalition game that can be computed again in exponential space. We show that a strategy might need exponential memory, and this bound on the memory size is optimal. As mentioned in the conclusion of that chapter, we are interested in closing the complexity gap for the safety coalition problem that was left open in this work. Another interesting direction of study is to solve the reachability coalition problem, where the winning condition for the coalition is given by a reachability objective. The same approach will not work, since the unfolding tree may be infinite. This is because, unlike safety, a coalition strategy may need to play differently when a vertex repeats. This is indeed the case in the example given in Figure 6.8, as explained in Page 144.

7.2 Perspectives

The work accomplished in this thesis opens many possibilities for future works. While some of them were already mentioned in the respective chapters, here we discuss more general research directions.

7.2.1 Broadcast protocols

Various properties. The notion of cutoff can be extended for various other properties of broadcast protocols, for instance, the *target reachability* problem of [DSZ10], that asks, given a control state q, if there exists an initial configuration and an execution of the protocol such that every process in the final configuration of that execution is in q. Then a cutoff for the (positive instances of) target reachability problem can naturally be defined as the minimal number of nodes for which the above is true. In reconfigurable semantics, the monotonicity property ensures that the above is well-defined since one can copycat a process in an execution to construct another execution with more nodes, and ensure if the former is a positive instance, so is the latter. More general properties of that kind can be described as Boolean combinations of formulas, each of them constraining the number of processes labelled with a particular state, called *cardinality constraints* [DSTZ12]. The cardinality reachability problem then asks, given a formula φ, if there exists an initial configuration and an execution of the protocol such that the final configuration of that execution satisfies φ. The above formalization subsumes the coverability problem as well as target reachability problem. The authors show that this problem is decidable for reconfigurable semantics of broadcast protocols. Then a natural idea would be to study the notion of cutoff for positive instances of that problem. However, one needs to be careful since the monotonicity property of a protocol may imply, for some formulas, that it is satisfied with some number of nodes but cannot be satisfied with more number of nodes. This is the case, for instance, when there is an upper bound on the number of processes in some states. Yet, one can define a cutoff for suitable formulas (for instance, formulas that correspond to coverability, or target reachability) and ask whether similar bounds on the cutoff hold.

Getting rid of lossy broadcasts. A key feature of the reconfigurable (as well as the lossy) semantics of broadcast protocols that led us to show the existence and find the bounds on the cutoff is the monotonicity property, described as the *copycat property*. We have argued that the copycat node performs some 'lossy' broadcasts (no process receives the message), and to perform such broadcasts in the reconfigurable semantics, the node is disconnected from its neighbours. However, it may be more realistic to assume, in contrast to our definitions, that each broadcast must be paired with at least one reception of the message. Such an assumption is made, for instance, in [HS20], for rendez-vous networks, where exactly two entities communicate at a certain instance via message passing. In that case, a cutoff may not always exist for some protocol. Then it is more

meaningful to consider the problem of deciding whether, given a protocol and a property, there exists a cutoff for that particular pair. The problem has been proved decidable for target reachability on rendez-vous protocols [HS20]. The complexity results were further improved in [BER21]. However, bounds on the cutoff have not been given for the positive instances. It would therefore be interesting to investigate whether our technique can be adapted to prove upper bounds on cutoff for positive instances of the problem.

Communication topology and beyond. The copycat property does not hold for static semantics (recall that in static semantics, the communication topology is fixed). Indeed, the key argument that the copycat node performs some 'lossy' broadcasts does not apply here because that node cannot be disconnected from other nodes. However, the coverability problem is decidable for static broadcast networks if we restrict the communication topology to, for instance, cliques [EFM99] or k-bounded path topologies (for a fixed k) [DSZ10]. Therefore, one can naturally consider the decision problem of the existence of a cutoff in these decidable fragments and look for (tight) bounds on the cutoff. Finally, it would be interesting to also look at the existence and bounds on cutoff for decidable properties on other parameterized models, such as probabilistic broadcast protocols [BFS14], rendez-vous networks [GS92], or asynchronous shared-memory systems [EGM13].

7.2.2 Parameterized games

The work accomplished in the second part of the thesis, to the best of our knowledge, is a first step towards the study of concurrent games with a parameterized number of players. It naturally opens a lot of directions for future research. While Chapters 5 and 6 discuss some of them, here we discuss more general perspectives for future works.

Language description. We have considered regular languages to describe interactions of the players. A natural extension would be to consider more general languages, for example, *context-free languages* (CFL) described by *context-free grammars* (CFG). Notice that, the setting where a distinguished player is trying to win against her opponents playing as a coalition is still decidable by construction of knowledge game. Indeed, by Parikh's theorem, the projection of a CFL to the length of the words is a semilinear set [Par66]. However, the size of the semilinear set needs not be polynomial in the size of the CFG. For instance, one can construct a CFG on one variable and linear number of terminals such that the corresponding language is a union of exponentially many linear sets [KT10]. Therefore, the algorithm described in Chapter 5 gives a naive exponential space upper bound for reachability condition. The PSPACE lower bound will still apply. It would, therefore, be interesting to investigate more closely the complexity of the problem. One can also consider other representations of languages such as logical formulas (for instance, MSO on words) as edge labellings.

Randomized strategies. So far, we have considered only pure (non-randomized) strategies of the agents. In classical two-player concurrent games, randomized strategies might be more powerful than deterministic ones in the sense that one can construct an example of a game with reachability objective such that Eve has no pure winning strategy, but she has a randomized strategy that is winning with probability 1 [AHK98]. In that context, one can consider decision problems of checking whether a player has a strategy that is winning with probability 1 (*almost-sure winning*); or for any threshold $\delta > 0$, it is winning with probability greater than $1 - \delta$ (*limit-sure winning*), see [AHK98]. One can similarly consider randomized strategies of agents in parameterized games and adapt almost-sure and limit-sure winning conditions for an agent, or even for the coalition.

Quantitative objectives. Quantitative extensions can also be considered. For instance, transitions can be assigned weights (rewards) over real numbers for a player (for the one player vs others setting) or even for the coalition (for the coalition setting). The reward after a play is then simply the sum of the weights on the transitions of the play. One can then consider, for instance, *mean-payoff* objectives (limit average of the weights) or *energy objectives* (all prefixes of the play must have non-negative weights).

Non-zero sum games. Other game theoretic concepts can also be considered. For instance, one can consider non-zero sum games, and the concept of (Nash) equilibria can be studied. Given a winning condition for each player, a Nash equilibrium is a strategy profile such that no player has incentive to deviate. Then one can ask questions like existence, or complexity of computing a Nash equilibrium.

Getting rid of identifiers. In our model of parameterized games, the agents have identifiers. However, in some practical examples of distributed systems, agents (processes) have no identifiers, for instance, in cache-coherence protocols, and ad hoc networks (this was the case in the first part of the thesis). Another example is a network of automated vehicles, where identifiers may affect the privacy of the individuals. Therefore, one can expect a game model with a parameterized number of players without identifiers of the agents. Towards that direction, one can consider shuffle-closed (regular) languages so that the ordering of the players do not impact on the transitions. For instance, the word $w = a^n b^n$ is in a language L if and only if for any w with $\#_a(w) = \#_b(w) = n$, $w \in L$. More generally, one can also consider a dependence relation on the set of actions and consider (regular) trace languages as edge labellings described by asynchronous automata.

Practical applications. Finally, we look forward to finding practical applications of the model of parameterized games. In the introductory chapter, we designed certain characteristics of a typical server-client model using parameterized games, yet, the model was quite restrictive in that example. We however envision that such games may be applied to a variety of contexts, such as telecommunications, networking protocols, distributed algorithms, etc. For instance, in *O-persistent* CSMA protocols, each node is assigned a

transmission order and they transmit data in their assigned order. Note that the number of nodes is unbounded *a priori*. Here, the assigned order can be seen as identifiers of the nodes, and one can hope to design a parameterized game for the model. Another possibility is ad hoc networks, where a transition between two configurations can be labelled with a language of the form $\bot^*(!\mathsf{a})\bot^*(?\mathsf{a})^*\bot^*$, where $\bot$ represents the behaviours of the nodes which are not involved in that transition. Then one can hope to achieve some correlations between the satisfiability of various properties of the network and coalition winning strategies in the corresponding coalition game. Here again, the number of nodes in the network can be unbounded in general. However, a difficulty here is that the nodes in an ad hoc network do not have identifiers, and one also has to be careful about the evolution of the communication topology, and define a set of rules on the languages that will correspond to a transition in the network.

Bibliography

[AAD+04] Dana Angluin, James Aspnes, Zoë Diamadi, Michael J. Fischer, and René Peralta. Computation in networks of passively mobile finite-state sensors. In *Proceedings of the 23rd Annual Symposium on Principles of Distributed Computing (PODC'04)*, pages 290–299. ACM, 2004.

[AAE06] Dana Angluin, James Aspnes, and David Eisenstat. Stably computable predicates are semilinear. In *Proceedings of the 25th Annual Symposium on Principles of Distributed Computing (PODC'06)*, pages 292–299. ACM, 2006.

[ADR+11] Parosh Aziz Abdulla, Giorgio Delzanno, Othmane Rezine, Arnaud Sangnier, and Riccardo Traverso. On the verification of timed ad hoc networks. In *Proceedings of the 9th International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS'11)*, volume 6919 of *Lecture Notes in Computer Science*, pages 256–270. Springer, September 2011.

[AFG+09] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy H. Katz, Andrew Konwinski, Gunho Lee, David A. Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. Above the clouds: A berkeley view of cloud computing. Technical report, University of California, Berkeley, February 2009.

[AHH13] Parosh Aziz Abdulla, Frédéric Haziza, and Lukás Holík. All for the price of few. In *Proceedings of the 14th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI'13)*, volume 7737 of *Lecture Notes in Computer Science*, pages 476–495. Springer, 2013.

[AHK98] Luca de Alfaro, Thomas A. Henzinger, and Orna Kupferman. Concurrent reachability games. In *Proceedings of the 39th Annual Symposium on Foundations of Computer Science (FOCS'98)*, pages 564–575. IEEE Computer Society, 1998.

[AHK02] Rajeev Alur, Thomas A. Henzinger, and Orna Kupferman. Alternating-time temporal logic. *Journal of the ACM*, 49(5):672–713, 2002.

[AJKR14] Benjamin Aminof, Swen Jacobs, Ayrat Khalimov, and Sasha Rubin. Parameterized model checking of token-passing systems. In *Proceedings of the 15th*

International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI'14), volume 8318 of *Lecture Notes in Computer Science*, pages 262–281. Springer, 2014.

[AK86] Krzysztof R. Apt and Dexter Kozen. Limits for automatic verification of finite-state concurrent systems. *Information Processing Letters*, 22(6):307–309, 1986.

[AKR+14] Benjamin Aminof, Tomer Kotek, Sasha Rubin, Francesco Spegni, and Helmut Veith. Parameterized model checking of rendezvous systems. In *Proceedings of the 21st International Conference on Concurrency Theory (CONCUR'14)*, volume 8704 of *Lecture Notes in Computer Science*, pages 109–124. Springer, September 2014.

[ALW89] Martín Abadi, Leslie Lamport, and Pierre Wolper. Realizable and unrealizable specifications of reactive systems. In *Proceedings of the 16th International Colloquium on Automata, Languages and Programming (ICALP'89)*, volume 372 of *Lecture Notes in Computer Science*, pages 1–17. Springer, 1989.

[BBM19a] Nathalie Bertrand, Patricia Bouyer, and Anirban Majumdar. Concurrent parameterized games. In *Proceedings of the 39th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'19)*, volume 150 of *LIPIcs*, pages 31:1–31:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[BBM19b] Nathalie Bertrand, Patricia Bouyer, and Anirban Majumdar. Reconfiguration and message losses in parameterized broadcast networks. In *Proceedings of the 30th International Conference on Concurrency Theory (CONCUR'19)*, volume 140 of *LIPIcs*, pages 32:1–32:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, August 2019.

[BBM20] Nathalie Bertrand, Patricia Bouyer, and Anirban Majumdar. Synthesizing safe coalition strategies. In *Proceedings of the 39th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'20)*, volume 182 of *LIPIcs*, pages 39:1–39:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[BBM21] Nathalie Bertrand, Patricia Bouyer, and Anirban Majumdar. Reconfiguration and message losses in parameterized broadcast networks. *Logical Methods in Computer Science*, 17(1), 2021.

[BBMU11] Patricia Bouyer, Romain Brenguier, Nicolas Markey, and Michael Ummels. Nash equilibria in concurrent games with büchi objectives. In *Proceedings of the 31st Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'12)*, volume 13 of *LIPIcs*, pages 375–386. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2011.

[BBMU15] Patricia Bouyer, Romain Brenguier, Nicolas Markey, and Michael Ummels. Pure Nash equilibria in concurrent games. *Logical Methods in Computer Science*, 11(2), 2015.

[BD08] Dietmar Berwanger and Laurent Doyen. On the power of imperfect information. In *Proceedings of the 27th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'08*, volume 2 of *LIPIcs*, pages 73–82. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2008.

[BDG+19] Nathalie Bertrand, Miheer Dewaskar, Blaise Genest, Hugo Gimbert, and Adwait Amit Godbole. Controlling a population. *Logical Methods in Computer Science*, 15(3), 2019.

[BEG+20] Michael Blondin, Javier Esparza, Blaise Genest, Martin Helfrich, and Stefan Jaax. Succinct population protocols for presburger arithmetic. In *Proceedings of the 37th International Symposium on Theoretical Aspects of Computer Science(STACS'20)*, volume 154 of *LIPIcs*, pages 40:1–40:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[BER21] A. R. Balasubramanian, Javier Esparza, and Mikhail A. Raskin. Finding cut-offs in leaderless rendez-vous protocols is easy. In *Proceedings of the 24th International Conference on Foundations of Software Science and Computation Structure (FOSSACS'21)*, volume 12650 of *Lecture Notes in Computer Science*, pages 42–61. Springer, March 2021.

[BFS14] Nathalie Bertrand, Paulin Fournier, and Arnaud Sangnier. Playing with probabilities in reconfigurable broadcast networks. In *Proceedings of the 17th International Conference on Foundations of Software Science and Computation Structure (FOSSACS'14)*, volume 8412 of *Lecture Notes in Computer Science*, pages 134–148. Springer, April 2014.

[BHV03] Ahmed Bouajjani, Peter Habermehl, and Tomás Vojnar. Verification of parametric concurrent systems with prioritized FIFO resource management. In *Proceedings of the 14th International Conference on Concurrency Theory (CONCUR'03)*, volume 2761 of *Lecture Notes in Computer Science*, pages 172–187. Springer, 2003.

[BJK10] Tomás Brázdil, Petr Jancar, and Antonín Kucera. Reachability games on extended vector addition systems with states. In *Proceedings of the 37th International Colloquium on Automata, Languages, and Programming (ICALP'10)*, volume 6199 of *Lecture Notes in Computer Science*, pages 478–489. Springer, 2010.

[BJK+15] 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.

[BK08] Christel Baier and Joost-Pieter Katoen. *Principles of model checking.* MIT Press, 2008.

[BKP11] Dietmar Berwanger, Lukasz Kaiser, and Bernd Puchala. A perfect-information construction for coordination in games. In *Proceedings of the 31th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'11)*, volume 13 of *LIPIcs*, pages 387–398. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2011.

[BL69] J. Richard Buchi and Lawrence H. Landweber. Solving sequential conditions by finite-state strategies. *Transactions of the American Mathematical Society*, 138:295–311, 1969.

[BMR+16] Patricia Bouyer, Nicolas Markey, Mickael Randour, Arnaud Sangnier, and Daniel Stan. Reachability in networks of register protocols under stochastic schedulers. In *Proceedings of the 43rd International Colloquium on Automata, Languages, and Programming, (ICALP 2016)*, volume 55 of *LIPIcs*, pages 106:1–106:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016.

[CD10] Krishnendu Chatterjee and Laurent Doyen. Energy parity games. In *Proceedings of the 37th International Colloquium on Automata, Languages, and Programming (ICALP'10)*, volume 6199 of *Lecture Notes in Computer Science*, pages 599–610. Springer, 2010.

[CDHR06] Krishnendu Chatterjee, Laurent Doyen, Thomas A. Henzinger, and Jean-François Raskin. Algorithms for omega-regular games with imperfect information› . In *Proceedings of the 20th International Workshop on Computer Science Logic (CSL'06)*, volume 4207 of *Lecture Notes in Computer Science*, pages 287–302. Springer, 2006.

[CDHR10] Krishnendu Chatterjee, Laurent Doyen, Thomas A. Henzinger, and Jean-François Raskin. Generalized mean-payoff and energy games. In *Proceedings of the 30thnd Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'10)*, volume 8 of *LIPIcs*, pages 505–516. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2010.

[CES09] Edmund M. Clarke, E. Allen Emerson, and Joseph Sifakis. Model checking: algorithmic verification and debugging. *Communications of the ACM*, 52(11):74–84, 2009.

[CFO20] Thomas Colcombet, Nathanaël Fijalkow, and Pierre Ohlmann. Controlling a random population. In *Proceedings of the 23rd International Conference on Foundations of Software Science and Computation Structures (FOSSACS'20)*, volume 12077 of *Lecture Notes in Computer Science*, pages 119–135. Springer, 2020.

[Chr86] Marek Chrobak. Finite automata and unary languages. *Theoretical Computer Science*, 47(3):149–158, 1986.

[Chu62] Alonzo Church. Logic, arithmetic, and automata. In *Proceedings of the International Congress of Mathematicians*, volume 29, page 23–35, 1962.

[CJH03] Krishnendu Chatterjee, Marcin Jurdzinski, and Thomas A. Henzinger. Simple stochastic parity games. In *Proceedings of the 17th International Workshop on Computer Science Logic (CSL'03)*, volume 2803 of *Lecture Notes in Computer Science*, pages 100–113. Springer, 2003.

[CMJ04] Krishnendu Chatterjee, Rupak Majumdar, and Marcin Jurdzinski. On nash equilibria in stochastic games. In *Proceedings of the 18th International Workshop on Computer Science Logic (CSL'04)*, volume 3210 of *Lecture Notes in Computer Science*, pages 26–40. Springer, 2004.

[Con92] Anne Condon. The complexity of stochastic games. *Information and Computation*, 96(2):203–224, 1992.

[Coo71] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing (STOC'71)*, pages 151–158. ACM, 1971.

[CQSS19] Tristan Charrier, Arthur Queffelec, Ocan Sankur, and François Schwarzentruber. Reachability and coverage planning for connected agents. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI'19)*, pages 144–150. ijcai.org, 2019.

[CRR12] Krishnendu Chatterjee, Mickael Randour, and Jean-François Raskin. Strategy synthesis for multi-dimensional quantitative objectives. In *Proceedings of the 23rd International Conference on Concurrency Theory (CONCUR'12)*, volume 7454 of *Lecture Notes in Computer Science*, pages 115–131. Springer, 2012.

[CTTV04] Edmund M. Clarke, Muralidhar Talupur, Tayssir Touili, and Helmut Veith. Verification by network decomposition. In *Proceedings of the 21st International Conference on Concurrency Theory (CONCUR'04)*, volume 3170 of *Lecture Notes in Computer Science*, pages 276–291. Springer, September 2004.

[dAFH+03] Luca de Alfaro, Marco Faella, Thomas A. Henzinger, Rupak Majumdar, and Mariëlle Stoelinga. The element of surprise in timed games. In *Proceedings of the 14th International Conference on Concurrency Theory (CONCUR'03)*, volume 2761 of *Lecture Notes in Computer Science*, pages 142–156. Springer, 2003.

[dAHK07] Luca de Alfaro, Thomas A. Henzinger, and Orna Kupferman. Concurrent reachability games. *Theoretical Computer Science*, 386(3):188–217, 2007.

[DDG+10] Aldric Degorre, Laurent Doyen, Raffaella Gentilini, Jean-François Raskin, and Szymon Torunczyk. Energy and mean-payoff games with imperfect information. In *Proceedings of the 24th International Workshop on Computer Science Logic (CSL'10)*, volume 6247 of *Lecture Notes in Computer Science*, pages 260–274. Springer, 2010.

[DN12] Thuan Duong-Ba and Thinh P. Nguyen. Distributed client-server assignment. In *Proceedings of the 37th Annual IEEE Conference on Local Computer Networks (LCN'12)*, pages 296–299. IEEE Computer Society, 2012.

[DST13] Giorgio Delzanno, Arnaud Sangnier, and Riccardo Traverso. Parameterized verification of broadcast networks of register automata. In *Proceedings of the 7th International Workshop on Reachability Problems (RP'13)*, volume 8169 of *Lecture Notes in Computer Science*, pages 109–121. Springer, September 2013.

[DSTZ12] Giorgio Delzanno, Arnaud Sangnier, Riccardo Traverso, and Gianluigi Zavattaro. On the complexity of parameterized reachability in reconfigurable broadcast networks. In *Proceedings of the 32nd Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'12)*, volume 18 of *LIPIcs*, pages 289–300. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, December 2012.

[DSZ10] Giorgio Delzanno, Arnaud Sangnier, and Gianluigi Zavattaro. Parameterized verification of ad hoc networks. In *Proceedings of the 21st International Conference on Concurrency Theory (CONCUR'10)*, volume 6269 of *Lecture Notes in Computer Science*, pages 313–327. Springer, September 2010.

[DSZ11] Giorgio Delzanno, Arnaud Sangnier, and Gianluigi Zavattaro. On the power of cliques in the parameterized verification of ad hoc networks. In *Proceedings of the 14th International Conference on Foundations of Software Science and Computational Structures (FOSSACS'11)*, volume 6604 of *Lecture Notes in Computer Science*, pages 441–455. Springer, March 2011.

[DSZ12] Giorgio Delzanno, Arnaud Sangnier, and Gianluigi Zavattaro. Verification of ad hoc networks with node and communication failures. In *Proceedings of the 32nd International Conference on Formal Techniques for Distributed Systems (FMOODS/FORTE'12)*, volume 7273 of *Lecture Notes in Computer Science*, pages 235–250. Springer, June 2012.

[DT13] Giorgio Delzanno and Riccardo Traverso. Decidability and complexity results for verification of asynchronous broadcast networks. In *Proceedings of the 7th International Conference on Language and Automata Theory and Applications (LATA'13)*, volume 7810 of *Lecture Notes in Computer Science*, pages 238–249. Springer, April 2013.

[EFM99] Javier Esparza, Alain Finkel, and Richard Mayr. On the verification of broadcast protocols. In *Proceedings of the 14th Annual IEEE Symposium on Logic in Computer Science (LICS'99)*, pages 352–359. IEEE Computer Society, July 1999.

[EGLM17] Javier Esparza, Pierre Ganty, Jérôme Leroux, and Rupak Majumdar. Verification of population protocols. *Acta Informatica*, 54(2):191–215, 2017.

[EGM13] Javier Esparza, Pierre Ganty, and Rupak Majumdar. Parameterized verification of asynchronous shared-memory systems. In *Proceedings of the 25th International Conference on Computer Aided Verification (CAV'13)*, volume 8044 of *Lecture Notes in Computer Science*, pages 124–140. Springer, July 2013.

[EK00] E. Allen Emerson and Vineet Kahlon. Reducing model checking of the many to the few. In *Proceedings of the 17th International Conference on Automated Deduction (CADE'00)*, volume 1831 of *Lecture Notes in Computer Science*, pages 236–254. Springer, June 2000.

[EM79] Andrzej Ehrenfeucht and Jan Mycielski. Positional strategies for mean payoff games. *International Journal of Game Theory*, 8:109–113, 1979.

[EN95] E. Allen Emerson and Kedar S. Namjoshi. Reasoning about rings. In *Proceedings of the 22nd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'95)*, pages 85–94. ACM Press, January 1995.

[Esp14] Javier Esparza. Keeping a crowd safe: On the complexity of parameterized verification (invited talk). In *Proceedings of the 31st International Symposium on Theoretical Aspects of Computer Science (STACS'14)*, volume 25 of *LIPIcs*, pages 1–10. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2014.

[FK13] Ofer Feinerman and Amos Korman. Theoretical distributed computing meets biology: A review. In *Proceedings of the 9th International Conference on Distributed Computing and Internet Technology (ICDCIT'13)*, volume 7753 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2013.

[GKW19] Julian Gutierrez, Sarit Kraus, and Michael J. Wooldridge. Cooperative concurrent games. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS'19)*, pages 1198–1206. International Foundation for Autonomous Agents and Multiagent Systems, 2019.

[GS64] Seymour Ginsburg and Edwin H. Spanier. Bounded algol-like languages. *Transactions of the American Mathematical Society*, 113(2):333–368, 1964.

[GS92] Steven M. German and A. Prasad Sistla. Reasoning about systems with many processes. *Journal of the ACM*, 39(3):675–735, 1992.

[GThW02] Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors. *Automata, Logics, and Infinite Games: A Guide to Current Research*, volume 2500 of *Lecture Notes in Computer Science*. Springer, 2002.

[HS20] Florian Horn and Arnaud Sangnier. Deciding the existence of cut-off in parameterized rendez-vous networks. In *Proceedings of the 21st International Conference on Concurrency Theory (CONCUR'20)*, volume 171 of *LIPIcs*, pages 46:1–46:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, September 2020.

[Huy82] Thiet-Dung Huynh. The complexity of semilinear sets. *Journal of Information Processing and Cybernetics*, 18(6):291–338, 1982.

[JM96] David B Johnson and David A Maltz. Dynamic source routing in ad hoc wireless networks. In *Mobile computing*, pages 153–181. Springer, 1996.

[Kar72] Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, The IBM Research Symposia Series, pages 85–103, 1972.

[KKW10] Alexander Kaiser, Daniel Kroening, and Thomas Wahl. Dynamic cutoff detection in parameterized concurrent programs. In *Proceedings of the 25th International Conference on Computer Aided Verification (CAV'10)*, volume 6174 of *Lecture Notes in Computer Science*, pages 645–659. Springer, 2010.

[KT10] Eryk Kopczynski and Anthony Widjaja To. Parikh images of grammars: Complexity and applications. In *Proceedings of the 25th Annual IEEE Symposium on Logic in Computer Science (LICS'10)*, pages 80–89. IEEE Computer Society, 2010.

[LFI+17] Giuseppe Antonio Di Luna, Paola Flocchini, Taisuke Izumi, Tomoko Izumi, Nicola Santoro, and Giovanni Viglietta. Population protocols with faulty interactions: The impact of a leader. In *Proceedings of the 10th Internation Conference on Algorithms and Complexity (CIAC'17)*, volume 10236 of *Lecture Notes in Computer Science*, pages 454–466, 2017.

[Lip76] Richard J. Lipton. *The reachability problem requires exponential space.* Research report. Department of Computer Science, Yale University, 1976.

[Mar02] Andrew Martinez. Efficient computation of regular expressions from unary NFAs. In *Proceedings of the 5th International Workshop on Descriptional Complexity of Formal Systems (DCFS'02)*, pages 174–187. Department of Computer Science, The University of Western Ontario, Canada, 2002.

[Mat94] Armando B. Matos. Periodic sets of integers. *Theoretical Computer Science*, 127(2):287–312, 1994.

[Min67] Marvin L. Minsky. *Computation: Finite and Infinite Machines.* Prentice-Hall, Inc., 1967.

[MPS95] Oded Maler, Amir Pnueli, and Joseph Sifakis. On the synthesis of discrete controllers for timed systems (an extended abstract). In *Proceedings of the 12th International Symposium on Theoretical Aspects of Computer Science (STACS'95)*, volume 900 of *Lecture Notes in Computer Science*, pages 229–242. Springer, 1995.

[MW03] Swarup Mohalik and Igor Walukiewicz. Distributed games. In *Proceedings of the 23rd Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'03)*, volume 2914 of *Lecture Notes in Computer Science*, pages 338–351. Springer, 2003.

[Nas50] John F. Nash. Equilibrium points in n-person games. *Proc. of the National Academy of Sciences*, 36:48–49, 1950.

[NH06] Sebastian Nanz and Chris Hankin. A framework for security analysis of mobile wireless networks. *Theoretical Computer Science*, 367(1-2):203–227, 2006.

[NK18] Giang-Truong Nguyen and Kyungbaek Kim. A survey about consensus algorithms used in blockchain. *Journal of Information Processing Systems*, 14(1):101–128, 2018.

[OR94] Martin J. Osborne and Ariel Rubinstein. *A Course in Game Theory*. MIT Press, 1994.

[Par66] Rohit Parikh. On context-free languages. *Journal of the ACM*, 13(4):570–581, 1966.

[PB99] Charles E. Perkins and Elizabeth M. Belding-Royer. Ad-hoc on-demand distance vector routing. In *Proceedings of the 2nd Workshop on Mobile Computing Systems and Applications (WMCSA'99)*, pages 90–100. IEEE Computer Society, 1999.

[PR79] Gary L. Peterson and John H. Reif. Multiple-person alternation. In *Proceedings of the 20th Annual Symposium on Foundations of Computer Science (FOCS'79)*, pages 348–363. IEEE Computer Society Press, 1979.

[PR90] Amir Pnueli and Roni Rosner. Distributed reactive systems are hard to synthesize. In *Proceedings of the 31st Annual Symposium on Foundations of Computer Science (FOCS'90)*, pages 746–757. IEEE Computer Society, 1990.

[Pri14] Erich Prisner. *Game Theory through Examples*. Classroom Resource Materials. Mathematical Association of America, 2014.

[Rab72] Michael Oser Rabin. *Automata on Infinite Objects and Church's Problem*. American Mathematical Society, USA, 1972.

[Rei84] John H. Reif. The complexity of two-player games of incomplete information. *Journal of Computer and System Sciences*, 29(2):274–301, 1984.

[Rep96] Inquiry Board Report. Ariane 5 - flight 501 failure. `https://esamultimedia.esa.int/docs/esa-x-1819eng.pdf`, Paris, July 19, 1996.

[Sav70] Walter J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences*, 4(2):177–192, 1970.

[Saw10] Zdenek Sawa. Efficient construction of semilinear representations of languages accepted by unary NFA. In *Proceedings of the 4th International Workshop on Reachability Problems (RP'10)*, volume 6227 of *Lecture Notes in Computer Science*, pages 176–182. Springer, 2010.

[SM73] Larry J. Stockmeyer and Albert R. Meyer. Word problems requiring exponential time (preliminary report). In *Proceedings of the 5th Annual ACM Symposium on Theory of Computing (STOC'73)*, pages 1–9. ACM, 1973.

[SRS08] Anu Singh, C. R. Ramakrishnan, and Scott A. Smolka. A process calculus for mobile ad hoc networks. In *Proceedings of the 10th International Conference on Coordination Models and Languages (COORDINATION'08)*, volume 5052 of *Lecture Notes in Computer Science*, pages 296–314. Springer, 2008.

[SRS09] Anu Singh, C. R. Ramakrishnan, and Scott A. Smolka. Query-based model checking of ad hoc network protocols. In *Proceedings of the 20th International Conference on Concurrency Theory (CONCUR'09)*, volume 5710 of *Lecture Notes in Computer Science*, pages 603–619. Springer, 2009.

[Suz88] Ichiro Suzuki. Proving properties of a ring of finite-state machines. *Information Processing Letters*, 28(4):213–214, 1988.

[Tho95] Wolfgang Thomas. On the synthesis of strategies in infinite games. In *Proceedings of the 12th Annual Symposium on Theoretical Aspects of Computer Science (STACS'95)*, volume 900 of *Lecture Notes in Computer Science*, pages 1–13. Springer, 1995.

[Tho09] Wolfgang Thomas. Facets of synthesis: Revisiting church's problem. In *Proceedings of the 24th International Conference on Foundations of Software Science and Computation Structure (FOSSACS'09)*, volume 5504 of *Lecture Notes in Computer Science*, pages 1–14. Springer, 2009.

[Umm08] Michael Ummels. The complexity of nash equilibria in infinite multiplayer games. In *Proceedings of the 24th International Conference on Foundations of Software Science and Computation Structure (FOSSACS'08)*, volume 4962 of *Lecture Notes in Computer Science*, pages 20–34. Springer, 2008.

[Vaz01] Vijay V. Vazirani. *Approximation algorithms*. Springer, 2001.

[YMG08] Jennifer Yick, Biswanath Mukherjee, and Dipak Ghosal. Wireless sensor network survey. *Computer Networks*, 52(12):2292–2330, 2008.

[ZP96] Uri Zwick and Mike Paterson. The complexity of mean payoff games on graphs. *Theoretical Computer Science*, 158(1&2):343–359, 1996.

Index

université PARIS-SACLAY

ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre: Vérification et Synthèse de Systèmes Concurrents Paramétrés

Mots clés: model checking, systèmes distribués, vérification paramétrée, réseaux ad hoc, jeux sur graphes, synthèse de stratégies

Résumé: Cette thèse se situe au croisement de la vérification et de la synthèse des systèmes concurrents paramétrés. Le problème de la vérification de modèles paramétrés demande si un système satisfait une spécification donnée indépendamment du nombre de ses composants, alors que la synthèse vise la conception de protocoles pour ses composants afin que la spécification soit satisfaite.

Nous étudions un modèle paramétré de réseaux où les processus sont distribués sur un graphe non orienté; ils exécutent le même protocole et communiquent par des diffusions sélectives de messages. Le problème de couverture demande si un état donné du protocole est peut être atteint. Nous montrons que pour les instances positives du problème de couverture, en supposant que la topologie de communication est reconfigurable, la taille et la longueur d'une exécution couvrante minimale sont bornées linéairement et quadratiquement, respectivement. Nous introduisons une sémantique de perte à l'envoi et montrons des bornes similaires pour la taille et la longueur d'une exécution couvrante.

Les interactions entre différents agents peuvent être modélisées par des jeux. Nous introduisons et étudions deux cadres de ce que l'on appelle les jeux concurrents paramétrés, un modèle de jeux concurrents avec un nombre arbitraire d'agents. Tout d'abord, nous considérons le scénario où un joueur distingué Eve tente d'atteindre un objectif contre un nombre arbitraire d'adversaires, quelles que soient leurs stratégies. Nous prouvons que l'existence d'une stratégie gagnante pour Eve est décidable, et nous fournissons des bornes de complexité exactes pour ces jeux d'accessibilité. Deuxièmement, nous considérons un jeu de coalition où tous les joueurs essaient collectivement d'atteindre un objectif commun. Dans ce cadre, nous considérons des objectifs de sûreté et montrons que l'existence d'une stratégie de coalition gagnante est décidable, et nous établissons des bornes de complexité pour ce même problème.

Title: Verification and Synthesis of Parameterized Concurrent Systems

Keywords: model checking, distributed systems, parameterized verification, ad hoc networks, games on graphs, strategy synthesis

Abstract: This thesis is at the crossroad of verification and synthesis of parameterized concurrent systems. The parameterized model checking problem asks whether a system satisfies a given specification independently of the number of its components, whereas synthesis requires an algorithmic design of protocols for its components so that the specification is satisfied.

We study a parameterized model of networks where processes are distributed over an undirected graph, running the same broadcast protocol, and communicating via selective broadcasts of messages. The coverability problem asks whether a given state of the protocol is coverable. We show that for positive instances of the coverability problem in reconfigurable semantics, the size (cutoff) and the length (covering length) of a minimal covering execution is linearly and quadratically bounded, respectively. We introduce loss-on-broadcast semantics, and show similar bounds for the cutoff and the covering length.

The interactions between agents can be modelled using games. We introduce and study two different settings of the so-called parameterized concurrent games, a model of concurrent games with arbitrarily many agents. First, we consider a scenario of a distinguished player Eve trying to achieve a goal against arbitrarily many opponents, irrespective of their strategies. We prove the existence of a winning strategy for Eve is decidable, and show tight complexity bounds for reachability objectives. Second, we consider a coalition game where all players collectively try to achieve a common goal. We consider safety objectives and show the existence of a winning coalition strategy is decidable, and prove complexity bounds for the same.

Maison du doctorat de l'Université Paris-Saclay
2ème étage aile ouest, École normale supérieure Paris-Saclay
4 avenue des Sciences,
91190 Gif sur Yvette, France