Synthesizing Safe Coalition Strategies

Nathalie Bertrand

Université Rennes, Inria, CNRS, IRISA, Rennes, France

Patricia Bouyer

Université Paris-Saclay, ENS Paris-Saclay, CNRS, LSV, Gif-sur-Yvette, France

Anirban Majumdar

Université Rennes, Inria, CNRS, IRISA, Rennes, France
Université Paris-Saclay, ENS Paris-Saclay, CNRS, LSV, Gif-sur-Yvette, France

Abstract

Concurrent games with a fixed number of agents have been thoroughly studied, with various solution concepts and objectives for the agents. In this paper, we consider concurrent games with an arbitrary number of agents, and study the problem of synthesizing a coalition strategy to achieve a global safety objective. The problem is non-trivial since the agents do not know *a priori* how many they are when they start the game. We prove that the existence of a safe arbitrary-large coalition strategy for safety objectives is a PSPACE-hard problem that can be decided in exponential space.

2012 ACM Subject Classification Theory of computation → Verification by model checking

Keywords and phrases concurrent games, parameterized verification, strategy synthesis

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2020.39

Related Version `https://arxiv.org/abs/2008.03770`.

1 Introduction

Context. The generalisation and everyday usage of modern distributed systems call both for the verification and synthesis of algorithms or strategies running on distributed systems. Concrete examples are cloud computing, blockchain technologies, servers with multiple clients, wireless sensor networks, bio-chemical systems, or fleets of drones cooperating to achieve a common goal [11]. In their general form, these systems are not only distributed, but they may also involve an arbitrary number of agents. This explains the interest of the model-checking community both for the verification of parameterized systems [15, 9], and for the synthesis of distributed strategies [21]. Our contribution is at the crossroad of those topics.

Parameterized verification. Parameterized verification refers here to the verification of systems formed of an arbitrary number of agents. Often, the precise number of agents is unknown, yet, algorithms and protocols running on such distributed systems are designed to operate correctly independently of the number of agents. The automated verification and control of crowds, *i.e.*, in case the agents are anonymous, is challenging. Remarkably, subtle changes, such as the presence or absence of a controller in the system, can drastically alter the complexity of the verification problems [15]. In the decidable cases, the intuition that bugs appear for a small number of agents is sometimes confirmed theoretically by the existence of a cutoff property, which reduces the parameterized model checking to the verification of finitely many instances [14]. In the last 15 years, parameterised verification algorithms were successfully applied to *e.g.*, cache coherence protocols in uniform memory access multiprocessors [13], or the core of simple reliable broadcast protocols in asynchronous systems [17]. When agents have unique identifiers, most verification problems become undecidable, especially if one can use identifiers in the code agents execute [3].

40th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2020).
Editors: Nitin Saxena and Sunil Simon; Article No. 39; pp. 39:1–39:17
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

To our knowledge, there are few works on controlling parameterized systems. Exceptions are, control strategies for (probabilistic) broadcast networks [7] and for crowds of (probabilistic) automata [6, 18, 12].

Distributed synthesis. The problem of distributed synthesis asks whether strategies for individual agents can be designed to achieve a global objective, in a context where individuals have only a partial knowledge of the environment. There are several possible formalizations for distributed synthesis, for instance via an architecture of processes with communication links between agents [21], or using coordination games [20, 19, 8]. The two settings are linked, and many (un)decidability results have been proven, depending on various parameters.

Concurrent games on graphs. By allowing complex interactions between agents, concurrent games on graphs [1, 2] are a model of choice in several contexts, for instance for multi-agents systems, or for coordination or planning problems. An arena for n agents is a directed graph where the transitions are labeled 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). Most often, one considers infinite duration plays, *i.e.*, plays generated by iterating this process forever. Concepts studied on multiagent concurrent games include many borrowed from game theory, such as winning strategies (see *e.g.*, [1]), rationality of the agents (see *e.g.*, [16]), Nash equilibria (see *e.g.*, [23, 10]).

Parameterized concurrent games on graphs. In a previous work, we introduced concurrent games in which the number of agents is arbitrary [4]. These games generalize concurrent games with a fixed number of agents, and can be seen as a succinct representation of infinitely many games, one for each fixed number of agents. This is done by replacing, on edges of the arena, words representing the choice of each of the agents by languages of finite yet *a priori* unbounded words. Such a parameterized arena can represent infinitely many interaction situations, one for each possible number of agents. In parameterized concurrent games, the agents do not know *a priori* the number of agents participating to 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.

Such a game model raises new interesting questions, since the agents do not know beforehand how many they are. In [4], we first considered the question of whether Agent 1 can ensure a reachability objective independently of the number of her opponents, and no matter how they play. The problem is non trivial since Agent 1 must win with a *uniform* strategy. We proved that when edges are labeled with regular languages, the problem is PSPACE-complete; and for positive instances one can effectively compute a winning strategy in polynomial space.

Contribution. In this paper, we are interested in the coordination problem in concurrent parameterized games, with application to distributed synthesis. Given a game arena and an 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. In our setting, it is implicit that agents have identifiers. However agents do not communicate; their identifier will only be used to select the vertices the game proceeds to. Furthermore, agents do not know how many they are, they only see vertices which are visited, and can infer information about the number of agents involved in the game.

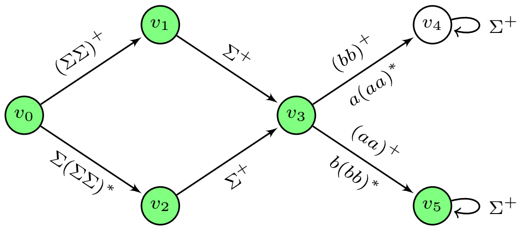


Figure 1 Example of a parameterized arena. All unspecified transitions lead to vertex v_4.

To better understand the model and the problem, consider the game arena depicted on Fig. 1. Edges are labeled by (regular) languages. Assuming the game starts at v_0, the game proceeds as follows: a positive integer k is selected by the environment, but is not revealed to the agents; then an infinite word $w \in \Sigma^\omega$ is selected collectively by the agents (this is the *coalition strategy*); the n-th letter of w represents the action played by Agent n;[1] depending on whether the prefix of length k of w belongs to $(\Sigma\Sigma)^+$ (in case k is even) or $\Sigma(\Sigma\Sigma)^*$ (in case k is odd), the game proceeds to vertex v_1 or v_2; the process is repeated ad infinitum, generating an infinite play in the graph. Depending on the winning condition, the play will be winning or losing for k. The coalition strategy will be said winning whenever the generated play is winning whatever the selected number k of agents is.

In this example, assuming the winning condition is to stay in the green vertices, there is a simple winning strategy: play a^ω in v_0, v_1 and v_2 (that is, all agents should play an a), and if the game has gone through v_1 (case of an even number of agents), then play a^ω in v_3 (all agents should play an a), otherwise play b^ω in v_3 (all agents should play a b). This ensures that the play never ends up in vertex v_4.

In this paper, we focus on safety winning conditions: the agents must collectively ensure that only safe vertices are visited along any play compatible with the coalition strategy in the game. We prove that the existence of a winning coalition strategy is decidable in exponential space, and that it is a PSPACE-hard problem. For positive instances, winning coalition strategies with an exponential-size memory structure can be synthesized in exponential space.

2 Game setting

We use $\mathbb{N}_{>0}$ for the set of positive natural numbers. For an alphabet Σ and $k \in \mathbb{N}_{>0}$, Σ^k denotes the set of all finite words of length k, Σ^+ denotes the set of all finite but non-empty words, and Σ^ω denotes the set of all infinite words. 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).

We introduced parameterized arenas in [4], a model of arenas with a parameterized number of agents. Parameterized arenas extend arenas for concurrent games with a fixed number of agents [1], by labeling the edges with languages over finite words, which may be of different lengths. Each word represents a joint move of the agents, for instance $u = a_1 \cdots a_k \in \Sigma^k$ assumes there are k agents, and for every $1 \leq n \leq k$, Agent n chooses action a_n.

▶ **Definition 1.** *A* parameterized arena *is a tuple* $\mathcal{A} = \langle V, \Sigma, \Delta \rangle$ *where*
- V *is a finite set of vertices;*

[1] This is where identifiers are implicitely used.

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

It is required that for every $(v, v') \in V \times V$*,* $\Delta(v, v')$ *describes a regular language.*

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

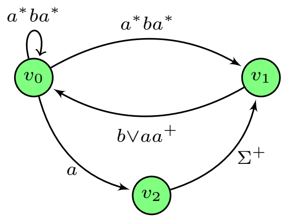


Figure 2 Example of a non-deterministic parameterized arena. Only safe vertices (colored in green) have been depicted here. All unspecified transitions lead to a non-safe vertex $\perp$.

▶ **Example 2.** We already gave an example in the introduction. Let us give another example, which will be useful for illustrating the constructions made in the paper. Fig. 2 presents a non-deterministic parameterized arena. As such the arena is not complete, we assume that all unspecified moves lead to an extra losing vertex $\perp$, not depicted here. If for some number of agents k (selected by environment and 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 (again selected by environment).

History, play and strategy. We fix a parameterized arena $\mathcal{A} = \langle V, \Sigma, \Delta \rangle$. 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 $1 \le j < p$, $\Delta(v_j, v_{j+1})$ is defined. We write $\mathsf{Hist}_{\mathcal{A}}$ for the set of all histories. An infinite sequence of vertices compatible with the edges is called a *play*.

A *strategy* for Agent n is a mapping $\sigma_n : \mathsf{Hist}_{\mathcal{A}} \to \Sigma$ that associates an action to every 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: $\widetilde{\sigma} = \langle \sigma_1, \sigma_2, \dots \rangle = (\mathsf{Hist}_{\mathcal{A}} \to \Sigma)^\omega$.

Table 1 From strategy profile to coalition strategy.

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

Observe that a strategy profile can equivalently be described as a *coalition strategy* $\boldsymbol{\sigma} : \mathsf{Hist}_{\mathcal{A}} \to \Sigma^\omega$, as illustrated in Table 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. Since, in this paper, we are interested in the existence of a winning strategy profile, it is equivalent to asking the existence of a winning coalition strategy (they may not be equivalent for some other decision problems). In the sequel, we mostly take the coalition strategy view, but may interchangeably also consider strategy profiles.

Finite memory coalition strategies. Let $\boldsymbol{\sigma} : \mathsf{Hist}_{\mathcal{A}} \to \Sigma^\omega$ be a coalition strategy and $\mathtt{M}$ be a set. We say that the strategy $\boldsymbol{\sigma}$ *uses memory* $\mathtt{M}$ whenever there exist $\mathtt{m}_{\mathtt{init}} \in \mathtt{M}$ and applications $\mathsf{upd} : \mathtt{M} \times V \to \mathtt{M}$ and $\mathsf{act} : \mathtt{M} \times V \to \Sigma^\omega$ such that by defining inductively $\mathtt{m}[h] \in \mathtt{M}$ by $\mathtt{m}[v_0] = \mathtt{m}_{\mathtt{init}}$ and $\mathtt{m}[h \cdot v] = \mathsf{upd}(\mathtt{m}[h], v)$, we have that for every $h \in \mathsf{Hist}_{\mathcal{A}}$, $\boldsymbol{\sigma}(h) = \mathsf{act}(\mathtt{m}[h], \mathit{last}(h))$, where $\mathit{last}(h)$ is the last vertex of history h. The structure $(\mathtt{M}, \mathsf{upd})$ records information on the history seen so far ($\mathtt{m}[h]$ is the memory state "reached" after history h), and act dictates how all the agents should play.

If $\mathtt{M}$ is finite, then $\boldsymbol{\sigma}$ is said *finite-memory*, and if $\mathtt{M}$ is a singleton, then $\boldsymbol{\sigma}$ is said *memoryless* (each choice only depends on the last vertex of the history).

Realizability and outcomes. For $k \in \mathbb{N}_{>0}$, we say 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 $j < p$, 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.

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, $\mathsf{Out}^k_{\mathcal{A}}(v_0, \boldsymbol{\sigma}) = \{v_0 v_1 \cdots \mid \ \forall j \in \mathbb{N}_{>0},\ [\boldsymbol{\sigma}(v_0 \cdots v_j)]_{\le k} \in \Delta(v_j, v_{j+1})\}$. 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 $\mathsf{Out}_{\mathcal{A}}(v_0, \boldsymbol{\sigma}) = \bigcup_{k \in \mathbb{N}_{>0}} \mathsf{Out}^k_{\mathcal{A}}(v_0, \boldsymbol{\sigma})$.

The safety coalition problem. We are now in a position to define our problem of interest. Given an arena $\mathcal{A} = \langle V, \Sigma, \Delta \rangle$, a set of *safe* vertices $S \subseteq V$ defines a *parameterized safety game* $\mathcal{G} = (\mathcal{A}, S)$. Without loss of generality we assume from now that $V \setminus S$ are sinks. A coalition strategy $\boldsymbol{\sigma}$ from v_0 in the safety game $\mathcal{G} = (\mathcal{A}, S)$ is said *winning* if all induced plays only visit vertices from S: $\mathsf{Out}_{\mathcal{A}}(v, \boldsymbol{\sigma}) \subseteq S^\omega$. 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:

SAFETY COALITION PROBLEM **Input**: A parameterized safety game $\mathcal{G} = (\mathcal{A}, S)$ and an initial vertex v_0. **Question**: Does there exist a coalition strategy $\boldsymbol{\sigma}$ such that $\mathsf{Out}_{\mathcal{A}}(v_0, \boldsymbol{\sigma}) \subseteq S^\omega$?

The safety coalition problem is a coordination problem: agents should agree on a joint strategy which, when played in the graph and no matter how many agents are involved, the resulting play is safe. Note that, due to the link between coalition strategies and tuples of individual strategies mentioned on Page 4, 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; this is for instance the case at vertex v_3 in the example of Fig. 2. Note that there is no direct communication between agents.

▶ **Example 3.** We have already given in the introduction a winning coalition strategy for the game in Fig. 1. On the arena in Fig. 2, assuming $\perp$ is the only unsafe vertex, one can also 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 $\boldsymbol{\sigma}(v_0) = aba^\omega$, $\boldsymbol{\sigma}(v_0v_2) = a^\omega$, $\boldsymbol{\sigma}(v_0v_1) = a^\omega$, and $\boldsymbol{\sigma}(v_0v_2v_1) = b^\omega$. 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, coalition plays either b^ω (when the history is $v_0v_2v_1$ and hence $k = 1$) 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). 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 paper is devoted to the proof of the following theorem:

▶ **Theorem 4.** *The safety coalition problem can be solved in exponential space, and is* PSPACE*-hard. For positive instances, one can synthesize a winning coalition strategy in exponential space which uses exponential memory; the exponential blowup in the size of the memory is tight.*

3 Solving the safety coalition problem

This section is devoted to the proof of Theorem 4. To prove the decidability and establish the complexity upper bound, we construct a tree unfolding of the arena, which is equivalent for deciding the existence of a winning coalition strategy. 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, which will be formalized in Section 3.1. We can then show how to solve the safety coalition problem at the tree level in Section 3.2. Synthesis and memory usage are analyzed in Section 3.3, and the running example game is discussed in Section 3.4

The hardness result is shown in Section 3.5 by a reduction from the QBF-SAT problem which is known to be PSPACE-complete [22].

3.1 Finite tree unfolding

From a parameterized safety 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 labeled with the corresponding vertices and the edges are labeled with the same regular languages as in the arena $\mathcal{A}$. 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 (recall Example 3). This is the reason why we need to consider a tree unfolding abstraction and not a DAG abstraction.

We assume the concept of tree is known. Traditionally, we call a node n' 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.

▶ **Definition 5.** *Let $\mathcal{G} = (\mathcal{A} = \langle V, \Sigma, \Delta\rangle, S)$ be a parameterized safety game and $v_0 \in V$ an initial vertex. The* tree unfolding *of $\mathcal{G}$ is the tree $\mathcal{T} = \langle N, E, \ell_N, \ell_E\rangle$ rooted at $\mathsf{n}_0 \in N$, where N is the finite set of nodes, $E \subseteq N \times N$ is the set of edges, $\ell_N : N \to V$ is the node labeling function, $\ell_E : N \times N \to 2^{\Sigma^+}$ is the edge labeling function, and:*

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

Each node in $\mathcal{T}$ corresponds to a unique history in $\mathcal{G}$, and the unfolding is stopped when a vertex repeats or an unsafe vertex is encountered. The set of nodes can be partitioned into $N = N_{\mathsf{int}} \sqcup N_{\mathsf{leaf}}$ where N_{int} is the set of internal nodes and N_{leaf} are the leaves of $\mathcal{T}$ (some leaves are unsafe, some leaves have an equilabeled ancestor). By construction, the height of $\mathcal{T}$ is bounded by $|V| + 1$ and its branching degree is at most $|V|$. The tree unfolding of $\mathcal{G}$ is hence at most in $O(|V|^{|V|})$ (and the exponential blowup is unavoidable in general).

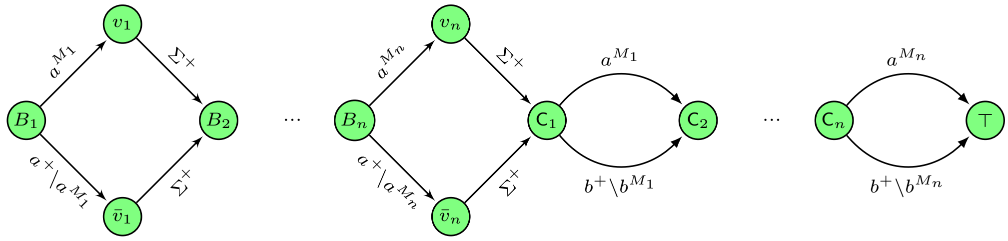
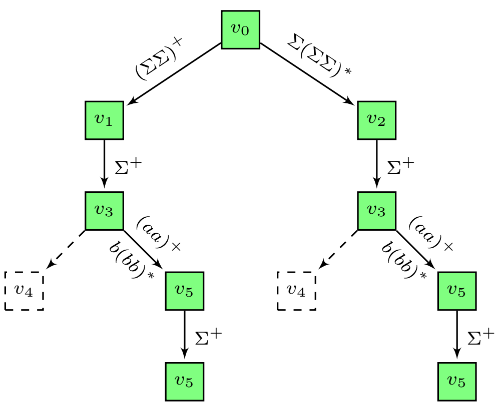
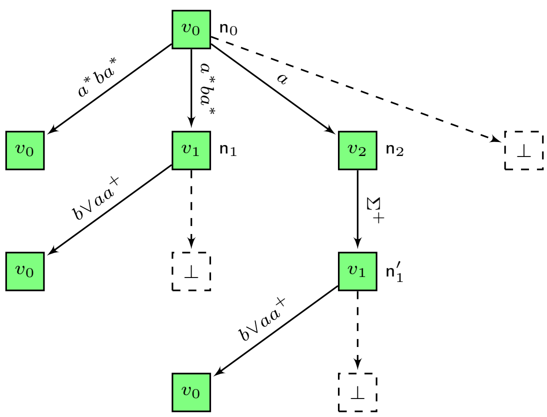


Figure 3 Example arena such that the tree unfolding is exponential. All unspecified transitions lead to the sink losing vertex $\perp$. Set M_i denotes multiples of the i-th prime number. For any play reaching C_1, for every i, the number of agents is in M_i iff the play went through v_i.

The exponential bound is reached by a a family $(\mathcal{A}_n)_{n\in\mathbb{N}_{>0}}$ of deterministic arenas, shown in Fig. 3, which is an extension of the example in Fig. 1, with $2n$ many blocks (and, $O(n)$ many vertices). 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.



(a) Tree unfolding of the arena in Fig. 1.



(b) Tree unfolding of the arena in Fig. 2.

Figure 4 Tree unfolding examples (green nodes correspond to safe vertices). Notice here that the unsafe leaves (and the edges leading to them) are presented with dashed rectangles (resp. arrows).

▶ **Example 6.** Fig. 4a and 4b represent the tree unfoldings of the parametrized arenas depicted in Fig. 1 and 2, respectively. On the left picture, the node names are avoided, and in all cases their labels are written within the nodes. The leaf nodes that correspond to unsafe vertices (and the edges leading to them) are presented with dashed rectangles (respectively, arrows). Notice that any leaf node is either labeled with an unsafe vertex (for instance, v_4 in Fig. 4a) or it has a unique ancestor with the same label. These two criteria ensure the tree is always finite (along all branches, some vertex has to repeat within $|V|$ many steps). However, multiple internal nodes in different branches can have same label but, coalition might have different (winning) strategies depending on their respective histories.

Let $\mathcal{G} = (\mathcal{A} = \langle V, \Sigma, \Delta\rangle, S)$ be a parameterized safety game with an initial vertex v_0 and $\mathcal{T} = \langle N, E, \ell_N, \ell_E\rangle$ be the tree unfolding corresponding to $\mathcal{A}$ with root n_0. We define the coalition game on $\mathcal{T}$ as follows.

History, play and strategy. Histories in $\mathcal{T}$ are defined similarly as in $\mathcal{G}$ (except vertices are replaced by nodes); the set of such histories is denoted $\mathsf{Hist}_\mathcal{T}$. A *history* in $\mathcal{T}$ is a finite sequence of nodes $H = \mathsf{n}_0\mathsf{n}_1 \ldots \mathsf{n}_p \in N^+$ such that for every $0 \leq j < p$, $(\mathsf{n}_j, \mathsf{n}_{j+1}) \in E$. 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 $N_{\mathsf{int}}^+ \cdot N_{\mathsf{leaf}}$. 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.

A *coalition strategy* in the unfolding tree is a mapping $\boldsymbol{\lambda} : N_{\mathsf{int}} \to \Sigma^\omega$ that assigns to every internal node $\mathsf{n} \in N_{\mathsf{int}}$ an ω-word $\boldsymbol{\lambda}(\mathsf{n})$. Notice that a coalition strategy in $\mathcal{T}$ is by definition memoryless (on N) which, as we will see later, is sufficient to capture winning strategies of the coalition in $\mathcal{G}$. We furthermore extend the definition of node labeling function ℓ_N to a history (resp. play) in the usual way.

Similarly to the parameterized arena setting, we define in a natural way the notions of k-realizability and of realizability for histories and plays. We also define for a coalition strategy $\boldsymbol{\lambda}$ in $\mathcal{T}$ (rooted at n_0), and $k \in \mathbb{N}_{>0}$ the sets $\mathsf{Out}^k_\mathcal{T}(\mathsf{n}_0, \boldsymbol{\lambda})$ and $\mathsf{Out}_\mathcal{T}(\mathsf{n}_0, \boldsymbol{\lambda})$.

A coalition strategy $\boldsymbol{\lambda}$ in $\mathcal{T}$ from n_0 is *winning* for the safety condition defined by the safe 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 $R = \mathsf{n}_0 \ldots \mathsf{n}_p \in \mathsf{Out}_\mathcal{T}(\mathsf{n}_0, \boldsymbol{\lambda})$, $\ell_N(\mathsf{n}_p) \in S$, written $\ell_N(\mathsf{Out}_\mathcal{A}(\mathsf{n}_0, \boldsymbol{\lambda})) \subseteq S^+$ for short.

Correctness of the tree unfolding. Next we show the equivalence of the winning strategies in the safety coalition game, and in the corresponding tree unfolding:

▶ **Lemma 7.** *Let $\mathcal{G} = (\mathcal{A} = \langle V, \Sigma, \Delta\rangle, S)$ be a parameterized safety game and $v_0 \in V$ and $\mathcal{T} = \langle N, E, \ell_N, \ell_E\rangle$ 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 $\ell_N(H) \in \mathsf{Hist}_\mathcal{A}$. We can hence define for every $\mathsf{n} \in N_{\mathsf{int}}$, $\boldsymbol{\lambda}(\mathsf{n}) = \boldsymbol{\sigma}(\ell_N(\iota(\mathsf{n})))$, where ι is the bijection mapping 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 $\rho = \ell_N(R) = v_0 \cdots v_p$ be its projection in $\mathcal{G}$. By construction $\ell_E(\mathsf{n}_i, \mathsf{n}_{i+1}) = \Delta(v_i, v_{i+1})$ for each $i < p$, 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, $\ell_N(\mathsf{n}_p) \in S$. 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.

▶ **Lemma 8.** *The application $\beta \colon \mathsf{n} \mapsto \ell_N(\iota(\mathsf{n}))$ defines a bijection between $N_{\mathsf{int}} \cup \{\mathsf{n} \in N_{\mathsf{leaf}} \mid \ell_N(\mathsf{n}) \notin S\}$ and the set $Z = \{\mathsf{zip}(h) \mid h \in \mathsf{Hist}_{\mathcal{A}}\}$.*

Proof. It is first obvious that this application is injective, since two nodes of the tree corresponds to different histories in $\mathcal{A}$ which all belong to Z.

This application is surjective: pick $h \in Z$; then, h has no repetition; furthermore it forms a real history in $\mathcal{G}$, which implies that it can be read as the label of some history in the tree unfolding. ◀

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 $\boldsymbol{\sigma}$ to the 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 $\alpha(\mathsf{zip}(h)) \in N_{\mathsf{int}}$ and $\boldsymbol{\sigma}(h)$ is set arbitrarily otherwise (recall that if $\alpha(\mathsf{zip}(h))$ is a leaf node, then h is actually 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: $\rho = v_0 v_1 \dots \in \mathsf{Out}^k_{\mathcal{A}}(v_0, \boldsymbol{\sigma})$. Let $h' = v_0 v_1 \dots v_q \sqsubseteq \rho$ be the shortest prefix of ρ ending in an unsafe vertex $v_q \notin S$, and write $\mathsf{zip}(h') = v_0 v_{i_1} \dots v_q$ 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 $R = \iota(\alpha(\mathsf{zip}(h'))) = \mathsf{n}_0 \mathsf{n}_{i_1} \dots \mathsf{n}_q$ 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 $\mathcal{T}$; which is a contradiction. We conclude that $\boldsymbol{\sigma}$ is a winning coalition strategy in $\mathcal{G}$. ◀

▶ **Example 9.** We illustrate the zip function on the arena in Fig. 2. Take $h = v_0 v_1 v_0 v_1$. First, $\mathsf{zip}(v_0) = v_0$; then $\mathsf{zip}(v_0 v_1) = \mathsf{zip}(v_0) \cdot v_1 = v_0 v_1$; $\mathsf{zip}(v_0 v_1 v_0) = v_0$ (which is the unique prefix of $\mathsf{zip}(v_0 v_1) = v_0 v_1$, ending at v_0); finally $\mathsf{zip}(v_0 v_1 v_0 v_1) = \mathsf{zip}(v_0 v_1 v_0) \cdot v_1 = v_0 v_1$. Then the function α uniquely maps each virtual history (*i.e.*, $\mathsf{zip}(h)$) ending at a safe vertex to an internal node in the tree, which is the heart of the proof of Lemma 7.

3.2 Existence of winning coalition strategy on the tree unfolding

In the previous subsection, we showed that the safety 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 over the alphabet Σ^m, where $m = |N_{\mathsf{int}}|$, which accepts the ω-words corresponding to winning coalition strategies in $\mathcal{T}$. More precisely, since $(\Sigma^m)^\omega$ and $(\Sigma^\omega)^m$, understood as the set of m-tuples of ω-words over Σ, are in one-to-one correspondence, an infinite word $\mathbf{w} \in (\Sigma^m)^\omega$ corresponds to m infinite words $\mathbf{w}_{\mathsf{n}}$, one for each internal node $\mathsf{n} \in N_{\mathsf{int}}$, thus representing a coalition strategy in $\mathcal{T}$.

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. We assume for every $(v, v') \in V \times V$ such that $\Delta(v, v') \neq \emptyset$, $\Delta(v, v')$ is given as a complete DFA over Σ. Those will be given as inputs to the algorithm.

Let $\mathcal{T} = \langle N, E, \ell_N, \ell_E\rangle$ 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: $N_{\mathsf{int}} = \{\mathsf{n}_1, \dots, \mathsf{n}_m\}$ and $E = \{e_1, \dots, e_r\}$, with $|N_{\mathsf{int}}| = m$ and $|E| = r$.

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

The 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 $e \in E$, let us write $\mathcal{B}_e = (Q_e, \Sigma, \delta_e, q^0_e, F_e)$ for the complete DFA over Σ such that $L(\mathcal{B}_e) = \ell_E(e)$. (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.) Note that some of the $\mathcal{B}_e$'s are identical since they correspond to the same original edge of $\mathcal{G}$.

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 labeled by a safe vertex. Formally, $Q \subseteq Q_1 \times \ldots \times Q_r$ is the set of states; $q^0 = (q^0_1, \dots, q^0_r)$ is the initial state; the transition relation δ executes the r automata $\mathcal{B}_e$'s componentwise: if letter $u \in \Sigma^m$ is read, then make the s-th component mimick $\mathcal{B}_{e_s}$ by reading the l-th letter of 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 $q = (q_1, \dots, q_r)$ that satisfy the following Boolean formula:

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

Note that $\mathcal{B}$ is equipped with a safety acceptance condition:[2] an infinite run $\zeta = q^0q^1q^2\dots$ 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.

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

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

Proof. Assume $\boldsymbol{\lambda} : N_{\mathsf{int}} \to \Sigma^\omega$ is a winning coalition strategy in $\mathcal{T}$, and consider the corresponding word $\mathbf{w} = (\boldsymbol{\lambda}(\mathsf{n}_1), \boldsymbol{\lambda}(\mathsf{n}_2), \dots, \boldsymbol{\lambda}(\mathsf{n}_m))$. Let us show that $\mathbf{w} \in L(\mathcal{B})$. Consider the infinite run $\zeta = q^0q^1\dots$ of $\mathcal{B}$ on $\mathbf{w}$. Fix a number of agents $k \in \mathbb{N}_{>0}$. Since $\boldsymbol{\lambda}$ is winning, any k-length prefix of $\boldsymbol{\lambda}$-induced play in $\mathsf{Out}^k_{\mathcal{T}}(\mathsf{n}_0, \boldsymbol{\lambda})$ is winning. Therefore for any $1 \leq i \leq t$

[2] This is a slight abuse of language since q^0 need not be in the set F.

such that $\mathsf{n}_0^i \dots \mathsf{n}_{z_i}^i$ is in $\mathsf{Out}_{\mathcal{T}}^k(\mathsf{n}_0, \boldsymbol{\lambda})$, the play satisfies for all $0 \le j < z_i$, $[\boldsymbol{\lambda}(\mathsf{n}_j^i)]_{\le k} \in \ell_E(e_j^i)$ and furthermore, $\ell_N(\mathsf{n}_{z_i}^i) \in S$; and hence $q^k \models \varphi_i$. Otherwise, if a length k-play is not induced by $\boldsymbol{\lambda}$, then φ_i is vacuously true for that $i \le t$. We conclude $q^k \models \varphi$. Since this is true for every $k \in \mathbb{N}_{>0}$, $\mathbf{w} \in L(\mathcal{B})$.

Let now $\boldsymbol{\lambda}$ be an arbitrary coalition strategy, and assume $\mathbf{w} = (\boldsymbol{\lambda}(\mathsf{n}_1), \boldsymbol{\lambda}(\mathsf{n}_2), \dots, \boldsymbol{\lambda}(\mathsf{n}_m)) \in L(\mathcal{B})$ with $\zeta = q^0 q^1 \dots$ 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 $1 \le i \le t$, $q^k \models \varphi_i$. Fix any such i; let $\mathsf{n}_0^i \dots \mathsf{n}_{z_i}^i$ be the i-th maximal path, and write $q^k = (q_1^k, \dots, q_r^k)$. Then for some $0 \le j < z_i$, the condition $q_{e_i^j}^k \in F_{e_i^j}$ implies $[\boldsymbol{\lambda}(\mathsf{n}_j^i)]_{\le k} \in \ell_E(e_j^i)$. In case the above is true for all $0 \le j < z_i$, we conclude $\mathsf{n}_0^i \dots \mathsf{n}_{z_i}^i \in \mathsf{Out}_{\mathcal{T}}^k(\mathsf{n}_0, \boldsymbol{\lambda})$ and φ_i ensures that $\ell_N(\mathsf{n}_{z_i}^i) \in S$. Otherwise, $\mathsf{n}_0^i \dots \mathsf{n}_{z_i}^i \notin \mathsf{Out}_{\mathcal{T}}^k(\mathsf{n}_0, \boldsymbol{\lambda})$. Finally φ ensures all k-length prefixes of $\boldsymbol{\lambda}$-induced plays in $\mathcal{T}$ are winning. Since this is true for any number of agents k, $\boldsymbol{\lambda}$ is a winning coalition strategy in $\mathcal{T}$. ◀

We now have all ingredients to solve the safety coalition problem, and to state a complexity upper-bound. As mentioned earlier, we assume that the arena is initially given with all associated complete DFAs (used by all $\mathcal{B}_e$) in the input.

▶ **Theorem 11.** *The safety coalition problem is in* EXPSPACE.

Proof. Solving the safety 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 notice that N is doubly exponential in $|V|$, the number of vertices of the initial arena $\mathcal{A}$ (each state of $\mathcal{B}$ is an exponential-size vector of states of automata given in the input). We do not build $\mathcal{B}$ a priori. Instead, we non-deterministically guess a safe prefix of length at most N (we only keep written two consecutive configurations and keep a counter to count up to N), and then a safe lasso on the last state of length at most N.

Provided one can check "easily" whether a state of $\mathcal{B}$ is safe, q the described procedure runs in non-deterministic exponential space, hence can be turned into a deterministic exponential space algorithm, by Savitch's theorem.

It remains to explain how one checks that a given state in $\mathcal{B}$ is safe. Formula φ is a SAT formula exponential in the size of $\mathcal{A}$, which can therefore be solved in exponential space as well.

Overall, we conclude that the safety coalition problem is in EXPSPACE. ◀

3.3 Synthesizing a winning coalition strategy

We assume all the notations of the two previous subsections, and we explain how we build a winning coalition strategy. From an accepting word of the form $\mathbf{u} \cdot \mathbf{v}^\omega$ in $\mathcal{B}$ (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:

$$\boldsymbol{\lambda}(\mathsf{n}_i) = \mathbf{u}_i \cdot \mathbf{v}_i^\omega \quad \text{for every } \mathsf{n}_i \in N_{\mathsf{int}}.$$

Then it is easy to transfer to a winning coalition strategy $\boldsymbol{\sigma}$ 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. Recall that, following the proof of Lemma 7, 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 $\mathcal{T}$.

▶ **Proposition 12.** *If there is a winning coalition strategy for a game $\mathcal{G} = (\mathcal{A}, S)$, then there is one which uses exponential memory, which can be computed in exponential space. Furthermore, winning might indeed require exponential memory.*

Proof. The tree unfolding can be seen as a memory structure for a winning strategy. Indeed, consider the memory set defined by N_{int}, starting from memory state n_0. Define the application $\mathsf{upd} : N_{\mathsf{int}} \times V \to N_{\mathsf{int}}$ by $\mathsf{upd}(\mathsf{n}, v') = \mathsf{n}'$ such that $v' \in S$ whenever

- either $\mathsf{n}' \in N_{\mathsf{int}}$ is a child of n such that $\ell_N(\mathsf{n}') = v'$
- or $\mathsf{n}' \in N_{\mathsf{int}}$ is an ancestor of $\mathsf{n}'' \in N_{\mathsf{leaf}}$ such that $\ell_N(\mathsf{n}'') = \ell_N(\mathsf{n}') = v'$, and n'' is a child of n.

We also define the application $\mathsf{act} : N_{\mathsf{int}} \times V \to \Sigma^\omega$ by $\mathsf{act}(\mathsf{n}, v) = \boldsymbol{\lambda}(\mathsf{n})$.

Then, it is easy to see that winning strategy $\boldsymbol{\sigma}$ can be defined using memory N_{int} and applications upd and act.

Furthermore, though the ω-words extracted fom $\mathcal{B}$ can be of doubly-exponential size, their computation and the overall procedure only requires exponential space.

For the lower bound, we show the following lemma.

▶ **Lemma 13.** *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 Fig. 3, 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, one cannot do better than the tree memory structure.

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 the tree unfolding.

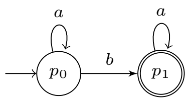
Assume one can do better and have a memory structure of size strictly smaller than 2^n. Then, arriving in vertex C_1, there are at least two different histories leading to the same memory state, hence the coalition strategy will select exactly the same ω-words in all vertices C_1, C_2, ..., C_n. We realize that it cannot be winning since the two histories disagree at least on a predicate "be a multiple of the i-th prime number". Contradiction. ◀

◀

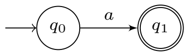
3.4 Illustration of the construction

We illustrate the construction on one example.

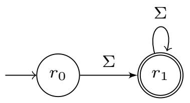
▶ **Example 14.** Fig. 6 represents part of the automaton $\mathcal{B}$ corresponding to the tree $\mathcal{T}$ in Fig. 4b (that is, the tree unfolding of the arena in Fig. 2). The automata $\mathcal{B}_e$ for the languages labeling the edges of $\mathcal{T}$ are depicted in Fig. 5. Here notice that 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 $\perp$. This is without loss of any generality: the language on any "unsafe" edge leading to $\perp$, 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 in Σ^m), and the other positions correspond to the other edges (in some chosen order). "$\times$" in a component of a state denotes the non-accepting sink state of the corresponding automaton (as mentioned in Fig. 5). Finally, here we have only shown the accepting states (marked in blue) and some of the non-accepting states. Indeed



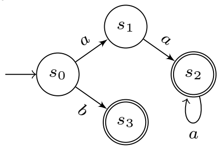
(a) Automaton for a^*ba^*.



(b) Automaton for a.



(c) Automaton for Σ^+.



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

Figure 5 Automata corresponding to the input languages of Fig. 2. The automata are not complete for sake of readability; all unspecified letters lead to a (sink) non-accepting state "×".

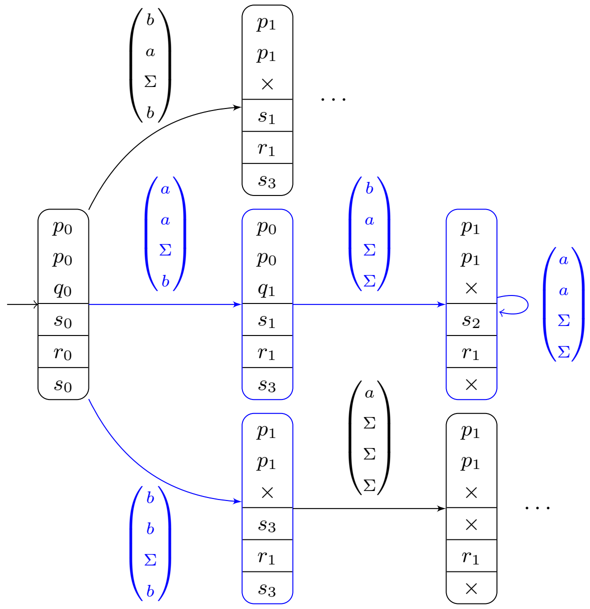


Figure 6 Automaton $\mathcal{B}$ corresponding to the tree given in Fig. 4b. Here we have only shown the accepting states (marked in blue) and some of the non-accepting states. Further explanations are given in Example 14.

one can verify that the states which are colored in 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 v_0v_0 and $v_0v_1v_0$ in $\mathcal{T}$ (notice we used the node labels here), and all of them lead to safe nodes. The infinite execution in blue (*i.e.*, all the words in $(a, a, \Sigma, b) \cdot (b, a, \Sigma, \Sigma) \cdot (a, a, \Sigma, \Sigma)^\omega$) corresponds to the winning coalition strategies in the tree: for instance, $\boldsymbol{\lambda}(\mathsf{n}_0) = aba^\omega$; $\boldsymbol{\lambda}(\mathsf{n}_1) = a^\omega$; for any $a \in \Sigma$, $\boldsymbol{\lambda}(\mathsf{n}_2) = a^\omega$; and $\boldsymbol{\lambda}(\mathsf{n}'_1) = b^\omega$ is a winning coalition strategy (note here, for instance, that at node n_2, any word from Σ^ω could be played).

3.5 PSPACE lower bound

We show the safety coalition problem is PSPACE-hard by reduction from QBF-SAT, which is known to be PSPACE-complete [22]. The construction is inspired by the one in [4], where the first agent was playing against the coalition of all other agents, with a reachability objective.

▶ **Proposition 15.** *The safety 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 $1 \leq h \leq m$, $C_h = \ell_{h,1} \vee \ell_{h,2} \vee \ell_{h,3}$, 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 M_i the set of all non-zero natural numbers that are multiples of p_i. For simplicity, we write a^{M_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 an 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 $\mathsf{C}_{m+1} = \top$.
- $\Sigma = \{a, b, c\} \cup \bigcup_{1 \le i \le 2r} \{a_i\}$.
- For every $0 \le s \le r-1$, every $1 \le i \le 2r$ and every $1 \le h \le m$:
  1. $\Delta(v_{2s}, \mathsf{x}_{2s+1}) = a^{M_{2s+1}}$ and $\Delta(v_{2s}, \bar{\mathsf{x}}_{2s+1}) = b^+ \setminus b^{M_{2s+1}}$
  2. $\Delta(v_{2s}, \top) = (a^+ \setminus a^{M_{2s+1}}) \cup b^{M_{2s+1}}$
  3. $\Delta(v_{2s+1}, \mathsf{x}_{2s+2}) = c^{M_{2s+2}}$ and $\Delta(v_{2s+1}, \bar{\mathsf{x}}_{2s+2}) = c^+ \setminus c^{M_{2s+2}}$
  4. $\Delta(\mathsf{x}_i, v_i) = \Sigma^+$ and $\Delta(\bar{\mathsf{x}}_i, v_i) = \Sigma^+$
  5. $\Delta(\mathsf{C}_h, \mathsf{C}_{h+1}) = \bigcup_{1 \le j \le 3} L_{h,j}$ where $L_{h,j} = a_i^{M_i}$ if $\ell_{h,j} = x_i$; $L_{h,j} = a_i^+ \setminus a_i^{M_i}$ if $\ell_{h,j} = \neg x_i$.

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

On the arena $\mathcal{A}_\varphi$, we consider the safety coalition game $\mathcal{G}_\varphi = (\mathcal{A}_\varphi, S)$ with $S = V \setminus \{\bot\}$. The construction is illustrated on a simple example with 3 variables and 2 clauses in Fig. 7.

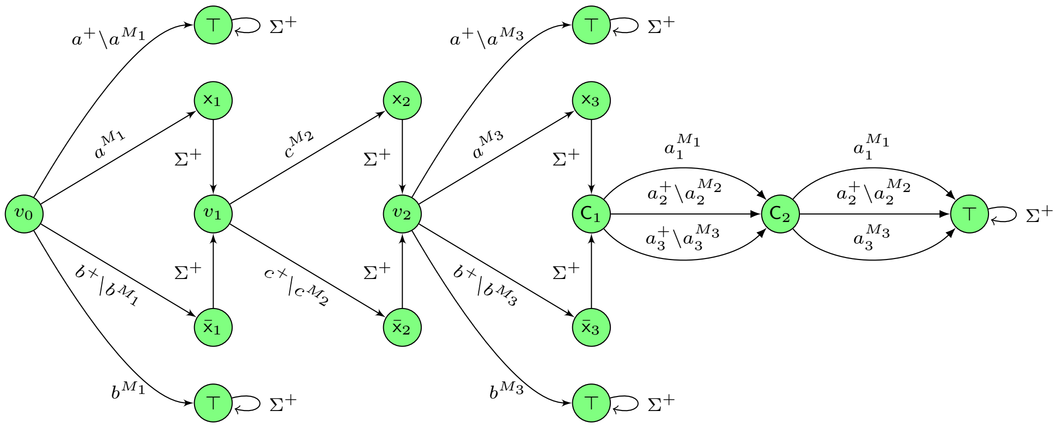


Figure 7 Parameterized arena for the formula $\varphi = \exists x_1 \forall x_2 \exists x_3 \cdot (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_1 \vee \neg x_2 \vee x_3)$. All unspecified transitions lead to the sink losing vertex $\bot$. Set M_i denotes multiples of the i-th prime number. Vertex x_i (resp. $\bar{\mathsf{x}}_i$) represents setting variable x_i to `true` (resp. `false`). For any play reaching C_1, for every i, the number of agents is in M_i iff the play went through x_i.

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 the next variable choice, otherwise the safe $\top$ state is reached (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 (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 it 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. 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 M_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 $-{a_1}^{M_1}-$ 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 M_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 M_i, this means that $\bar{\mathsf{x}}_i$ was visited, hence indeed x_i was set to `false`.

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

The proof in full details can be found in the arxiv version of this paper [5].

4 Future work

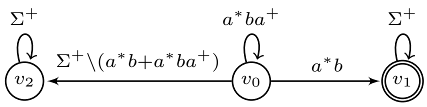


Figure 8 Example of a reachability coalition game: a winning coalition strategy is that Agent n plays a for the first $n-1$ rounds, then b for one round, and finally a forever.

In this paper, we focused on and obtained results for the coalition problem for safety objectives. The problem can obviously be defined for other objectives. The finite tree unfolding technique will not be correct in a general setting. We illustrate this on the game arena in Fig. 8. In this example, the goal is to collectively reach the target v_1. One can do so 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 the other plays an a. Coalition has a winning strategy: Agent n plays action a for the first $n-1$ 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, for each agent (and hence for the coalition), the strategy when going through v_0 differs at some step (at every step from the coalition point-of-view), so no finite tree unfolding will be correct.

As future work, we obviously would like to match lower and upper bounds for the safety coalition problem, but more importantly we would like to investigate more general objectives.

References

1 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 Press, 1998. doi:10.1109/SFCS.1998.743507.

2 Rajeev Alur, Thomas A. Henzinger, and Orna Kupferman. Alternating-time temporal logic. *Journal of the ACM*, 49:672–713, 2002. doi:10.1145/585265.585270.

3 Krzysztof Apt and Dexter C. Kozen. Limits for automatic verification of finite-state concurrent systems. *Information Processing Letters*, 22(6):307–309, May 1986. doi:10.1016/0020-0190(86)90071-2.

4 Nathalie Bertrand, Patricia Bouyer, and Anirban Majumdar. Concurrent parameterized games. In *Proceedings o 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. doi:10.4230/LIPIcs.FSTTCS.2019.31.

5 Nathalie Bertrand, Patricia Bouyer, and Anirban Majumdar. Synthesizing safe coalition strategies, 2020. URL: https://arxiv.org/abs/2008.03770.

6 Nathalie Bertrand, Miheer Dewaskar, Blaise Genest, Hugo Gimbert, and Adwait Amit Godbole. Controlling a population. *Logical Methods in Computer Science*, 15(3), 2019. doi:10.23638/LMCS-15(3:6)2019.

7 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, 2014. doi:10.1007/978-3-642-54830-7_9.

8 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. doi:10.4230/LIPIcs.FSTTCS.2011.387.

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

10 Patricia Bouyer, Romain Brenguier, Nicolas Markey, and Michael Ummels. Pure Nash equilibria in concurrent games. *Logical Methods in Computer Science*, 11(2:9), 2015. doi:10.2168/LMCS-11(2:9)2015.

11 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. doi:10.24963/ijcai.2019/21.

12 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. doi:10.1007/978-3-030-45231-5_7.

13 Giorgio Delzanno. Constraint-based verification of parameterized cache coherence protocols. *Formal Methods in System Design*, 23(3):257–301, 2003. doi:10.1023/A:1026276129010.

14 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, 2000. doi:10.1007/10721959_19.

15 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 fuer Informatik, 2014. doi:10.4230/LIPIcs.STACS.2014.1.

16 Dana Fisman, Orna Kupferman, and Yoad Lustig. Rational synthesis. In *Proceedings of the 16th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'10)*, volume 6015 of *Lecture Notes in Computer Science*, pages 190–201. Springer, 2010. doi:10.1007/978-3-642-12002-2_16.

17 Igor Konnov, Helmut Veith, and Josef Widder. What you always wanted to know about model checking of fault-tolerant distributed algorithms. In *Proceedings of the 10th International Andrei Ershov Informatics Conference (PSI'15)*, volume 9609 of *Lecture Notes in Computer Science*, pages 6–21. Springer, 2015. `doi:10.1007/978-3-319-41579-6_2`.

18 Corto Mascle, Mahsa Shirmohammadi, and Patrick Totzke. Controlling a random population is EXPTIME-hard, 2019. URL: `https://arxiv.org/abs/1909.06420`.

19 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. `doi:10.1007/978-3-540-24597-1_29`.

20 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. `doi:10.1109/SFCS.1979.25`.

21 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)*, volume 2, pages 746–757. IEEE Computer Society Press, 1990. `doi:10.1109/FSCS.1990.89597`.

22 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. `doi:10.1145/800125.804029`.

23 Michael Ummels and Dominik Wojtczak. The complexity of Nash equilibria in limit-average games. In *Proceedings of the 22nd International Conference on Concurrency Theory (CONCUR'11)*, volume 6901 of *Lecture Notes in Computer Science*, pages 482–496. Springer, 2011. `doi:10.1007/978-3-642-23217-6_32`.