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.

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].
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, \textit{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 \textit{e.g.}, [1]), rationality of the agents (see \textit{e.g.}, [16]), Nash equilibria (see \textit{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 \( a \)\textit{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 \( a \)\textit{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 \textit{uniform} strategy. We proved that when edges are labeled with regular languages, the problem is \textit{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.
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 $n$; 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 $a^\omega$ in $v_0$, $v_1$ and $v_2$ (that is, all agents should play an $a$), and if the game has gone through $v_3$ (case of an even number of agents), then play $a^\omega$ in $v_3$ (all agents should play an $a$), otherwise play $b^\omega$ 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 $\Sigma$ and $k \in \mathbb{N}_{>0}$, $\Sigma^k$ denotes the set of all finite words of length $k$, $\Sigma^+$ denotes the set of all finite but non-empty words, and $\Sigma^\omega$ 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 $\Sigma^k$).

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 $\mathcal{A} = (V, \Sigma, \Delta)$ where

\[ V \] is a finite set of vertices;

\[ \Delta \]
Σ is a finite set of actions;
Δ : V × V → 2Σ is a partial transition function.
It is required that for every (v, v′) ∈ V × V, ∆(v, v′) describes a regular language.

Fix a parameterized arena \( A = \langle V, \Sigma, \Delta \rangle \). The arena \( A \) 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 \( \bot \).

▶ **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 \( \bot \), 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 \( A = \langle V, \Sigma, \Delta \rangle \). A **history** in \( A \) is a finite sequence of vertices, that is compatible with the edges: formally, \( h = v_0v_1...v_p \in V^+ \) such that for every \( 1 \leq j < p \), \( \Delta(v_j, v_{j+1}) \) is defined. We write \( \text{Hist}_A \) for the set of all histories. An infinite sequence of vertices compatible with the edges is called a **play**.

A **strategy** for Agent 1 is a mapping \( \sigma_1 : \text{Hist}_A \to \Sigma \) 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: \( \tilde{\sigma} = (\sigma_1, \sigma_2, ...) = (\text{Hist}_A \to \Sigma)^\omega \).

**Table 1** From strategy profile to coalition strategy.

<table>
<thead>
<tr>
<th>( h )</th>
<th>( h_0 )</th>
<th>( h_1 )</th>
<th>( h_2 )</th>
<th>( h_3 )</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \sigma_1 )</td>
<td>( a )</td>
<td>( b )</td>
<td>( b )</td>
<td>( b )</td>
<td>...</td>
</tr>
<tr>
<td>( \sigma_2 )</td>
<td>( b )</td>
<td>( b )</td>
<td>( b )</td>
<td>( b )</td>
<td>...</td>
</tr>
<tr>
<td>( \sigma_3 )</td>
<td>( b )</td>
<td>( a )</td>
<td>( a )</td>
<td>( a )</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Observe that a strategy profile can equivalently be described as a **coalition strategy** \( \sigma : \text{Hist}_A \to \Sigma^\omega \), 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 \( \omega \)-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 \( \sigma : \text{Hist}_A \rightarrow \Sigma^\omega \) be a coalition strategy and \( M \) be a set. We say that the strategy \( \sigma \) uses memory \( M \) whenever there exist \( m_{\text{init}} \in M \) and applications \( \text{upd} : M \times V \rightarrow M \) and \( \text{act} : M \times V \rightarrow \Sigma^\omega \) such that by defining inductively \( m[h] \in M \) by \( m[v_0] = m_{\text{init}} \) and \( m[h \cdot v] = \text{upd}(m[h], v) \), we have that for every \( h \in \text{Hist}_A \), \( \sigma(h) = \text{act}(m[h], \text{last}(h)) \), where \( \text{last}(h) \) is the last vertex of history \( h \). The structure \((M, \text{upd})\) records information on the history seen so far (\( m[h] \) is the memory state “reached” after history \( h \)), and \( \text{act} \) dictates how all the agents should play.

If \( M \) is finite, then \( \sigma \) is said finite-memory, and if \( M \) is a singleton, then \( \sigma \) 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 \( \sigma \), an initial vertex \( v_0 \) and a number of agents \( k \in \mathbb{N}_{>0} \), we define the \( k \)-outcome \( \text{Out}^k_A(v_0, \sigma) \) as the set of all \( k \)-realizable plays induced by \( \sigma \) from \( v_0 \). Formally, \( \text{Out}^k_A(v_0, \sigma) = \{ v_0v_1 \cdots \mid \forall j \in \mathbb{N}_{>0}, [\sigma(v_0 \cdots v_j)]_{\leq k} \in \Delta(v_j, v_{j+1}) \} \). Note that the completeness assumption ensures that the set \( \text{Out}^k_A(v, \sigma) \) is not empty. Then the outcome of coalition strategy \( \sigma \) is simply \( \text{Out}_A(v_0, \sigma) = \bigcup_{k \in \mathbb{N}_{>0}} \text{Out}^k_A(v_0, \sigma) \).

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

<table>
<thead>
<tr>
<th><strong>Safety Coalition Problem</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> A parameterized safety game ( G = (A, S) ) and an initial vertex ( v_0 ).</td>
</tr>
<tr>
<td><strong>Question:</strong> Does there exist a coalition strategy ( \sigma ) such that ( \text{Out}_A(v_0, \sigma) \subseteq S^\omega ) ?</td>
</tr>
</tbody>
</table>

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 $\bot$ is the only unsafe vertex, one can also show that the agents have a winning coalition strategy $\sigma$ from $v_0$ to stay within green (i.e., safe) vertices. Consider the coalition strategy $\sigma$ such that $\sigma(v_0) = aba\omega$, $\sigma(v_0v_2) = a^\omega$, $\sigma(v_0v_1) = a^\omega$, and $\sigma(v_0v_2v_1) = b^\omega$. Intuitively, on playing $aba\omega$ 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 $b^\omega$ (when the history is $v_0v_2v_1$ and hence $k = 1$) or $a^\omega$ (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 $\omega$-word). However, one can show that there is no memoryless coalition winning strategy. Indeed, the coalition strategy $a^\omega$ from $v_1$ is losing for $k = 1$, similarly $b^\omega$ from $v_1$ is losing for $k \geq 2$, and any other strategy is also losing. For instance, $ba\omega$ 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 ($a^\omega$ is losing for $k = 1$, $b^\omega$ 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 $\omega$-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 $G = (A, S)$, we construct a finite tree as follows: we unfold the arena $A$ 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 $A$. The intuition behind this construction is that if a vertex is repeated in a winning play in $A$, 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 $G = (A = (V, \Sigma, \Delta), S)$ be a parameterized safety game and $v_0 \in V$ an initial vertex. The tree unfolding of $G$ is the tree $T = (N, E, \ell) \subseteq E \subset N \times N$ rooted at $n_0 \in N$, where $N$ is the finite set of nodes, $E \subseteq N \times N$ is the set of edges, $\ell : N \to V$ is the node labeling function, $\ell(E) : N \times N \to 2^{\Sigma^+}$ is the edge labeling function, and:

- the root $n_0$ satisfies $\ell(N_0) = v_0$;
- $\forall n \in N, \text{ if } \ell(n) \in S \text{ and for every ancestor } n'' \text{ of } n, \ell(n'') \not= \ell(n)$, then $\forall v' \in V$ such that $\Delta(v, v')$ is defined, there is a child of $n$ with $\ell(n') = v'$ and $\ell(E(n, n')) = \Delta(v, v')$; otherwise, the node $n$ has no successor.

Each node in $T$ corresponds to a unique history in $G$, and the unfolding is stopped when a vertex repeats or an unsafe vertex is encountered. The set of nodes can be partitioned into $N = N_{\text{int}} \cup N_{\text{leaf}}$ where $N_{\text{int}}$ is the set of internal nodes and $N_{\text{leaf}}$ are the leaves of $T$ (some leaves are unsafe, some leaves have an equilabeled ancestor). By construction, the height of $T$ is bounded by $|V| + 1$ and its branching degree is at most $|V|$. The tree unfolding of $G$ 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 $C_i$, for every $i$, the number of agents is in $M_i$ if the play went through $v_i$.

The exponential bound is reached by a family $(A_n)_{n \in N_{>0}}$ 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.

- 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).
Synthesizing Safe Coalition Strategies

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

**History, play and strategy.** Histories in \( T \) are defined similarly as in \( \mathcal{G} \) (except vertices are replaced by nodes); the set of such histories is denoted \( \text{Hist}_T \). A history in \( T \) is a finite sequence of nodes \( H = n_0n_1 \ldots n_p \in N^+ \) such that for every \( 0 \leq j < p, (n_j, n_{j+1}) \in E \). We denote by \( \text{Hist}_T \) the set of all histories in \( T \). A play in \( T \) is a maximal history, \( i.e. \), a finite sequence of nodes ending with a leave, thus in \( N^+_\text{Int} \cdot N_\text{leaf} \). Note that, contrary to the definition of a play in \( A \), a play in \( T \) is a finite sequence of nodes ending in a leaf.

A coalition strategy in the unfolding tree is a mapping \( \lambda : N_\text{Int} \to \Sigma^\omega \) that assigns to every internal node \( n \in N_\text{Int} \) an \( \omega \)-word \( \lambda(n) \). Notice that a coalition strategy in \( T \) 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 \( \lambda_N \) 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 \( \lambda \) in \( T \) (rooted at \( n_0 \)), and \( k \in \mathbb{N}_0 \) the sets \( \text{Out}_T^k(n_0, \lambda) \) and \( \text{Out}_T(n_0, \lambda) \).

A coalition strategy \( \lambda \) in \( T \) from \( n_0 \) is winning for the safety condition defined by the safe set \( S \) if every play in \( \text{Out}_T(n_0, \lambda) \) ends in a leaf with label in \( S \), \( i.e. \), if for every \( R = n_0 \ldots n_p \in \text{Out}_T(n_0, \lambda) \), \( \ell_N(n_p) \in S \), written \( \ell_N(\text{Out}_A(n_0, \lambda)) \subseteq S^+ \) 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 \( \mathcal{G} = (A = \langle V, \Sigma, \Delta \rangle, S) \) be a parameterized safety game and \( v_0 \in V \) and \( T = \langle N, E, \lambda_N, \ell_E \rangle \) 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 \( T \).

**Proof.** Assume first that the coalition of agents has a winning strategy \( \sigma \) in \( \mathcal{G} \). Any history \( H \in \text{Hist}_T \) can be projected to the history \( \ell_N(H) \in \text{Hist}_A \). We can hence define for every \( n \in N_\text{Int}, \lambda(n) = \sigma(\ell_N(\iota(n))) \), where \( \iota \) is the bijection mapping nodes to histories in \( T \). To prove that \( \lambda \) is winning in \( T \), consider any play \( R = n_0 \ldots n_p \in \text{Out}_T(n_0, \lambda) \) and let \( \rho = \ell_N(R) = v_0 \ldots v_p \) be its projection in \( \mathcal{G} \). By construction \( \ell_E(n_i, n_{i+1}) = \Delta(v_i, v_{i+1}) \) for each \( i < p \), and hence from the definition of \( \lambda, \rho \) is a history in \( \mathcal{G} \) induced by \( \sigma \). Since \( \sigma \) is winning, \( \rho \) only visits safe vertices. In particular, \( \ell_N(n_p) \in S \). Since this is true for every play induced by \( \lambda \), strategy \( \lambda \) is winning from \( n_0 \) in \( T \).

For the other direction, assume that \( \lambda \) is a winning coalition strategy from \( n_0 \) in \( T \). 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 \( T \). We first define a mapping \( \text{zip} : \text{Hist}_A \to \text{Hist}_A \)
that summarizes any history in $A$ to its \textit{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 $\text{zip}(v_0) = v_0$, and letting for every $h \in \text{Hist}_A$ and every $v' \in V$ such that $h \cdot v' \in \text{Hist}_A$,

$$\text{zip}(h \cdot v') = \begin{cases} 
\text{zip}(h) : v' & \text{if } v' \text{ does not appear in } \text{zip}(h) \\
0 \cdot v' \subseteq \text{zip}(h) & \text{otherwise}
\end{cases}$$

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

\textbf{Lemma 8.} The application $\beta : n \mapsto \ell_{N}(i(n))$ defines a bijection between $N_{\text{int}} \cup \{n \in N_{\text{leaf}} \mid \ell_{N}(n) \notin S\}$ and the set $Z = \{\text{zip}(h) \mid h \in \text{Hist}_A\}$.

\textbf{Proof.} It is first obvious that this application is injective, since two nodes of the tree corresponds to different histories in $A$ 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 $G$, which implies that it can be read as the label of some history in the tree unfolding.

We write $\alpha = \beta^{-1}$. Using the $\text{zip}$ function and $\alpha$, from a coalition strategy $\lambda$ in $T$, we define a coalition strategy $\sigma$ in $G$ by applying $\alpha$ to the virtual histories: for every history $h = v_0 \ldots v_p$ in $G$ we let $\sigma(h) = \lambda(\alpha(\text{zip}(h)))$ whenever $\alpha(\text{zip}(h)) \in N_{\text{int}}$ and $\sigma(h)$ is set arbitrarily otherwise (recall that if $\alpha(\text{zip}(h))$ is a leaf node, then $h$ is actually already a losing history).

Towards a contradiction, assume that $\sigma$ is not winning in $G$. Consider, some number of agents $k \in N_{>0}$, and a losing play with $k$ agents: $\rho = v_0v_1 \ldots v_q \in \text{Out}_A(v_0, \sigma)$. Let $h' = v_0v_1 \ldots v_q \subseteq \rho$ be the shortest prefix of $\rho$ ending in an unsafe vertex $v_q \notin S$, and write $\text{zip}(h') = v_0v_1 \ldots v_q$ for the corresponding virtual history. By definition of $\sigma$, $\text{zip}(h')$ is a $k$-outcome of $\sigma$ from $v_0$. Moreover, the corresponding play $R = \iota(\alpha(\text{zip}(h'))) = n_0n_1 \ldots n_q$ in $T$, belongs to the $k$-outcome of $\lambda$ from $v_0$. Since $v_q \notin S$, $\lambda$ is not winning in $T$; which is a contradiction. We conclude that $\sigma$ is a winning coalition strategy in $G$.

\textbf{Example 9.} We illustrate the $\text{zip}$ function on the arena in Fig. 2. Take $h = v_0v_1v_0v_1$. First, $\text{zip}(v_0) = v_0$; then $\text{zip}(v_0v_1) = \text{zip}(v_0) : v_1 = v_0v_1$; $\text{zip}(v_0v_1v_0) = v_0$ (which is the unique prefix of $\text{zip}(v_0v_1) = v_0v_1$, ending at $v_0$); finally $\text{zip}(v_0v_1v_0v_1) = \text{zip}(v_0v_1v_0) : v_1 = v_0v_1$. Then the function $\alpha$ uniquely maps each virtual history (i.e., $\text{zip}(h)$) 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 $T$, we construct a deterministic (safety) automaton over the alphabet $\Sigma^m$, where $m = |N_{\text{int}}|$, which accepts the $\omega$-words corresponding to winning coalition strategies in $T$. More precisely, since $(\Sigma^m)^\omega$ and $(\Sigma^\omega)^m$, understood as the set of $m$-tuples of $\omega$-words over $\Sigma$, are in one-to-one correspondence, an infinite word $w \in (\Sigma^m)^\omega$ corresponds to $m$ infinite words $w_n$, one for each internal node $n \in N_{\text{int}}$, thus representing a coalition strategy in $T$. 

\textbf{FSTTCS 2020}
Fix \( G = (\mathcal{A}, S) \) a parameterized safety game with \( \mathcal{A} = (V, \Sigma, \Delta) \) 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 \( \Sigma \). Those will be given as inputs to the algorithm.

Let \( \mathcal{T} = (N, E, \ell_N, \ell_E) \) 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: \( N_{\text{int}} = \{ n_1, \ldots, n_m \} \) and \( E = \{ e_1, \ldots, e_r \} \), with \( |N_{\text{int}}| = m \) 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 \( n_0^n_1 \ldots n_i^n_z \), with \( n_0^n_j = n_0 \), \( \forall j < z_i \), \( n_j^n_i \in N_{\text{int}} \) and \( n_i^n_z \in N_{\text{leaf}} \). Also, for \( 0 \leq j < z_i \), we note \( e_j^n_i = (n_j^n_i, n_{j+1}^i) \).

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 \( B_e = (Q_e, \Sigma, \delta_e, q_e^0, F_e) \) for the complete DFA over \( \Sigma \) such that \( L(B_e) = \ell_E(e) \). (Here \( Q_e \) is the set of states, \( \delta_e \) the transition function, \( q_e^0 \in Q_e \) the initial state and \( F_e \) the set of accepting states.) Note that some of the \( B_e \)'s are identical since they correspond to the same original edge of \( G \).

We then define a deterministic safety automaton \( B = (Q, \Sigma^m, \delta, q^0, F) \) that simulates all \( B_e \)'s in parallel and accepts \( \omega \)-words over alphabet \( \Sigma^m \) if every prefix satisfies the following: on every branch of the tree, if all corresponding \( B_e \)'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; \( q^0 = (q^0_1, \ldots, q^0_r) \) is the initial state; the transition relation \( \delta \) executes the \( r \) automata \( B_e \)'s componentwise: if letter \( u \in \Sigma^m \) is read, then make the \( s \)-th component mimick \( B_{e_s} \) by reading the \( t \)-th letter of \( u \), where \( t \) 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:

\[
\varphi = \bigwedge_{1 \leq i \leq t} \varphi_i \quad \text{where} \quad \varphi_i = \left( \bigwedge_{0 \leq j < z_i} q_{e_j}^i \in F_{e_j} \Rightarrow \ell_N(n_{z_i}^i) \in S \right).
\]

Note that \( B \) is equipped with a safety acceptance condition: an infinite run \( \zeta = q^0q^1q^2 \ldots \) of \( B \) is accepting if for every \( k \geq 1 \), \( q^k \in F \), and \( L(B) \) consists of all words \( w \) whose unique corresponding run is accepting.

Intuitively, \( \varphi_i \) expresses that if for some number of agents \( k \), the languages along the \( i \)-th maximal path contain the \( k \)-length prefixes of corresponding \( \omega \)-words (which means the induced play is \( k \)-realizable), then it should lead to a safe leaf; and then \( \varphi \) ensures that this should be true for all plays. This is formalized in the next lemma.

**Lemma 10.** Let \( \lambda : N_{\text{int}} \rightarrow \Sigma^\omega \) be a coalition strategy in \( \mathcal{T} \). Then, \( \lambda \) is winning if and only if \((\lambda(n_1), \lambda(n_2), \ldots, \lambda(n_m)) \in L(B)\).

Notice that in the above statement, we slightly abuse notation: \((\lambda(n_1), \lambda(n_2), \ldots, \lambda(n_m)) \) belongs to \((\Sigma^\omega)^m\), however it uniquely maps to a word in \((\Sigma^m)^\omega\), that can thus be read in \( B \).

**Proof.** Assume \( \lambda : N_{\text{int}} \rightarrow \Sigma^\omega \) is a winning coalition strategy in \( \mathcal{T} \), and consider the corresponding word \( w = (\lambda(n_1), \lambda(n_2), \ldots, \lambda(n_m)) \). Let us show that \( w \in L(B) \). Consider the infinite run \( \zeta = q^0q^1\ldots \) of \( B \) on \( w \). Fix a number of agents \( k \in \mathbb{N}_{\geq 0} \). Since \( \lambda \) is winning, any \( k \)-length prefix of \( \lambda \)-induced play in \( \text{Out}_k^\lambda(n_0, \lambda) \) 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 safe set \( F \).
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 \(\sigma\in \Sigma^*\) and
the associated complete DFAs (used by all \(B_j\)) in \(\mathcal{A}\), by Savitch’s theorem.

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 \(B_j\)) in the input.

\section*{Theorem 11.} The safety coalition problem is in \(\text{EXPSPACE}\).

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

We write \(N\) for the number of states of \(B\) and notice that \(N\) is doubly exponential in
\(|V|\), the number of vertices of the initial arena \(A\) (each state of \(B\) is an exponential-size
vector of states of automata given in the input). We do not build \(B\) a priori. Instead, we
don’t-terministically 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 \(B\) 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 \(B\) is safe. Formula \(\varphi\) is a SAT
formula exponential in the size of \(A\), which can therefore be solved in exponential space as
well.

Overall, we conclude that the safety coalition problem is in \(\text{EXPSPACE}\).

\section*{3.3 Synthesizing a winning coalition strategy}
We assume all the notations of the two previous subsections, and we explain how we build
the winning coalition strategy. From an accepting word of the form \(u \cdot v^\omega\) in \(B\) (where \(u \in \Sigma^*\)
and \(v \in \Sigma^+\)), one can synthesize a winning strategy \(\lambda\) in \(T\) by:

\[\lambda(n_i) = u_i \cdot v_i^\omega \quad \text{for every } n_i \in N_{\text{int}}.\]

Then it is easy to transfer to a winning coalition strategy \(\sigma\) in \(G\) by defining

\[\sigma(h) = \lambda(\alpha(\text{zip}(h))) \quad \text{for every history } h \in \text{Hist}_A,\]

that is, the \(\omega\)-word corresponding to the internal node representing its virtual history. Recall
that, following the proof of Lemma 7, \(\text{zip}\) assigns to every history its virtual history (by
greedily removing all the loops) and \(\alpha\) associates to a virtual history its corresponding node
in the tree \(T\).
Proposition 12. If there is a winning coalition strategy for a game $G = (A, S)$, 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 $N_{\text{int}}$, starting from memory state $n_0$. Define the application $\text{upd}: N_{\text{int}} \times V \rightarrow N_{\text{int}}$ by $\text{upd}(n, v') = n'$ such that $v' \in S$ whenever
- either $n' \in N_{\text{int}}$ is a child of $n$ such that $\ell_{N}(n') = v'$
- or $n' \in N_{\text{int}}$ is an ancestor of $n'' \in N_{\text{leaf}}$ such that $\ell_{N}(n'') = \ell_{N}(n') = v'$, and $n''$ is a child of $n$.

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

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

Furthermore, though the $\omega$-words extracted from $B$ 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 $(G_n)_n$ such that the size of $G_n$ 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 $a^\omega$ at every vertex $B_i$, and $a^\omega$ (resp. $b^\omega$) at vertex $C_i$ if the history went through $v_i$ (resp. $\neg v_i$). 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 $\omega$-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 $B$ corresponding to the tree $T$ in Fig. 4b (that is, the tree unfolding of the arena in Fig. 2). The automata $B_e$ for the languages labeling the edges of $T$ are depicted in Fig. 5. Here notice that each state of $B$ has as many components as the number of edges leading to a safe node in $T$, 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 $B$, presented as a single cell in the picture, correspond to the outgoing edges of the root $n_0$ of $T$ (hence they follow the same component in $\Sigma^m$), 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
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 $\varphi = \exists x_1 \forall x_2 \exists x_3 \ldots \forall x_{2r} \cdot (C_1 \land C_2 \land \ldots \land C_m)$ be a quantified Boolean formula in prenex normal form, where for every $1 \leq h \leq m$, $C_h = \ell_{h,1} \lor \ell_{h,2} \lor \ell_{h,3}$, and for every $1 \leq j \leq 3$, $\ell_{h,j} \in \{x_i, \neg x_i | 1 \leq i \leq 2r\}$ are the literals.

In the reduction, we use sets of natural numbers (that represent the number of agents) corresponding to multiplicities 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 $(a^+)^*$, 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 $\varphi$. 

---

**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 “$\times$”.

**Figure 6** Automaton $B$ 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.
From $\varphi$, we construct an arena $A_\varphi = (V, \Sigma, \Delta)$ as follows:

- $V = \{v_0, v_1, \ldots, v_{2r-1}, v_{2r}\} \cup \{x_1, \bar{x}_1, \ldots, x_{2r}, \bar{x}_{2r}\} \cup \{C_1, C_2, \ldots, C_m, C_{m+1}\} \cup \{\bot, \top\}$, where we identify some vertices: $v_{2r} = C_1$, and $C_{m+1} = \top$.
- $\Sigma = \{a, b, c\} \cup \bigcup_{1 \leq i \leq 2r} \{a_i\}$.
- For every $0 \leq s \leq r-1$, every $1 \leq i \leq 2r$ and every $1 \leq h \leq m$:
  1. $\Delta(v_{2s}, x_{2s+1}) = a^{M_{2s+1}}$ and $\Delta(v_{2s}, \bar{x}_{2s+1}) = b^+ \setminus b^{M_{2s+1}}$
  2. $\Delta(v_{2s}, \top) = (a^+ \setminus a^{M_{2s+1}}) \cup b^{M_{2s+1}}$
  3. $\Delta(v_{2s+1}, x_{2s+2}) = c^{M_{2s+2}}$ and $\Delta(v_{2s+1}, \bar{x}_{2s+2}) = c^+ \setminus c^{M_{2s+2}}$
  4. $\Delta(x_i, v_i) = \Sigma^+$ and $\Delta(\bar{x}_i, v_i) = \Sigma^+$
  5. $\Delta(C_h, C_{h+1}) = \bigcup_{1 \leq j \leq 3} L_{h,j}$ where $L_{h,j} = a_i^M$ if $\ell_{h,j} = x_i$; $L_{h,j} = a_i^+ \setminus a_i^M$ if $\ell_{h,j} = \neg x_i$.

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

On the arena $A_\varphi$, we consider the safety coalition game $G_\varphi = (A_\varphi, S)$ 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 $\varphi = \exists x_1 \forall x_2 \exists x_3 \cdot (x_1 \lor \neg x_2 \lor \neg x_3) \land (x_1 \lor \neg x_2 \lor x_3)$. 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. $\bar{x}_i$) represents setting variable $x_i$ to true (resp. false). For any play reaching $C_1$, for every $i$, the number of agents is in $M_i$, i.e. if the play went through $x_i$.

From $v_0$, a first phase up to $v_{2r} = C_1$ 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 $a^\omega$ for true, and $b^\omega$ 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 $c^\omega$ 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 $v_{2r} = C_1$, 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 $\bar{x}_i$). 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 $\varphi$ 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 $a^x_1$, 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 $a^x_1$. 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 $a^x_i$ 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 $a_i^{\bar{x}}$ leads from $C_h$ to $C_{h+1}$ with number of agents not in $M_i$, this means that $\bar{x}_i$ 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 $G_\varphi = (A_\varphi, S)$ if and only if $\varphi$ is true. ◻

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

4 Future work

![Figure 8](image)

**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 $a$ but 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**

Synthesizing Safe Coalition Strategies


