Controller Synthesis for Broadcast Networks with Data

Corto Mascle
LaBRI, Université de Bordeaux, France
MPI-SWS, Kaiserslautern, Germany

Abstract

We study the distributed controller synthesis problem in a parameterised setting. We search for strategies that guarantee that no error state is reached, no matter the number of processes involved. In the model at hand, processes communicate through unreliable broadcasts. Additionally, messages are signed with data from an infinite alphabet, representing identifiers. Processes have local registers, with which they can compare and store those data. Our main result is that controller synthesis is decidable for this model. We also characterise the complexity of the problem for each number of registers per process.

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

Keywords and phrases Parameterised verification, distributed synthesis, broadcast networks

Digital Object Identifier 10.4230/LIPIcs.CVIT.2016.23

1 Introduction

Distributed synthesis is a famously difficult problem, with many undecidability results on a variety of models [19, 21, 12, 6]. The difficulty can be attributed to the presence of multiple players with partial information. In order to find relevant models where the problem is decidable, we take inspiration from the parameterised approach to distributed verification: instead of verifying systems with large numbers of components, we abstract away that number, and ask whether a property holds for any number of components. This is also useful to verify protocols which are supposed to work on arbitrarily large networks.

In this paper we present a parameterised approach to distributed controller synthesis. We first study one of the most popular parameterised models, reconfigurable broadcast networks [9]. Then we go further and study the extension of those systems with data, as introduced in [8]. In this extension, agents have unique identifiers which they can use to sign messages and local registers which let them store and compare the signatures of received messages. This considerably extends the expressivity of the model. Each agent possesses two primary operations: broadcasting a letter from a finite alphabet along with the content of one of its registers or receiving a message, and comparing its datum with its registers and/or storing it in them. Broadcasts are lossy: when an agent sends a message, each other agent may or may not receive it; the set of agents receiving the broadcast is non-deterministic. A fundamental problem on such models is the coverability problem, which asks if a system has a run from an initial configuration to one with at least one agent in a given state.

We formalise the controller synthesis problem as follows: processes have controllable states, from which they can choose the next action, and uncontrollable ones, from which an adversary may decide the next step. The question is whether there exists a local strategy that chooses actions from controllable states so that for all N, a system made of N processes applying this strategy cannot reach an error state. We establish the decidability of this problem, and show tight complexity bounds on the problem, depending on the number of registers that each agent has access to.

42nd Conference on Very Important Topics (CVIT 2016).
Editors: John Q. Open and Joan R. Access; Article No. 23; pp. 23:1–23:45
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Related work This paper extends the work on verification of broadcast networks with data done in [8] and [13]. This model follows a recent effort to enrich parameterised models with data, usually to represent identifiers. For instance, the classic framework of population protocols has recently been extended with data [7], and some first decidability results have been established [26]. Let us also mention Petri nets with data [18]. The decidability of reachability in this model is an important open question, which saw some recent progress [16]. The relation between those models is still blurry, but we can draw hope from the known relations between these models without data: population protocols are tightly linked to Petri nets [10] and a common restriction of population protocols, called immediate observation, can be encoded in reconfigurable broadcast networks [2].

The problem we consider here fits in a family of what could be called parameterised games, which involve one player against an arbitrary number of opponents. Some other instances are concurrent parameterised games [3], where players choose their actions in parallel, and population control models, a full-information turn-based formalism [4]. In this work, we focus on distributed strategies: we want each process to act based only on its local history. Similar ideas have been explored in [5] and [24, Chapter 11].

Structure Section 2 describes the model and the main problem. The central result of this work is the decidability of controller synthesis for broadcast networks of register automata. We present it incrementally: In Section 3 we use the much simpler case of broadcast networks without data to illustrate our approach. In Section 4 we extend this proof to the subclass of systems called signature BGR where processes can only send messages with their initial identifier. Finally, we present the main decidability result in Section 5. We highlight the common structure between these sections by using similar sequences of definitions and lemmas. For instance, Definition 11, Lemma 12, and Theorem 13 have counterparts in Section 4: Definition 17, Lemma 18, and Theorem 24.

In Section 6 we discuss a different choice of definition of local strategies, and argue that the problem is unaffected by this choice. To complete the picture, in Section 7 we show tight complexity bounds for the SafeStrat problems when each process has a single register.

Problem \ Nb of registers	$r = 0$	$r = 1$	$r \geq 2$
Coverability	P [9]	NP [13]	$\mathbf{F}_{\omega^\omega}$ [13]
Safe strategy synthesis	NP (Thm 13)	NExpTime (Thm 33)	$\mathbf{F}_{\omega^\omega}$ (Thm 16)

Table 1 Complexity of Cover and SafeStrat, depending on the number of registers. All problems are complete for the indicated class. The results of the last column hold for any fixed $r \geq 2$ and when r is part of the input.

This paper uses hyperlinks. Occurrences of a term are linked to its definition. The reader can click on words and symbols (or just hover over them on some PDF readers) to see the definition.

2 Preliminaries

2.1 Register transducers

We start by describing the transition system of individual processes, which are register transducers. Then, in Section 2.2 we will define our broadcast network model as the composition of a finite but arbitrary number of those processes.

We fix an infinite set of *data* $\mathbb{D}$. We define a notion of register transducer that is well-suited for the definition of our distributed systems. They receive and send *messages*, which are pairs (m, d) made of a *letter* m from a finite alphabet and a *datum* d from $\mathbb{D}$.

▶ **Definition 1** (Register transducer). *A register transducer with r registers over domain $\mathbb{D}$ is given by a tuple $\mathcal{R} = (Q, \mathcal{M}, q_{init}, \Delta)$ with Q a finite set of states, with q_{init} the initial state, $\mathcal{M}$ a finite alphabet, and Δ a set of transitions which are of three kinds:*

- $q \xrightarrow{\mathbf{br}(m,i)} q'$ *broadcast transitions that broadcast a message (m, d) with d the content of register i,*
- $q \xrightarrow{\mathbf{rec}(m,=i)} q'$ *equality transitions that read a message (m, d) and check that d is in register i,*
- $q \xrightarrow{\mathbf{rec}(m,\downarrow i)} q'$ *record transitions that read a message (m, d) where d is not in any register and put d in register i.*

Transitions of the two last kinds are called reception transitions. The size of $\mathcal{R}$, written $|\mathcal{R}|$, is $|Q| + |\Delta| + r$.

A *local configuration* of $\mathcal{R}$ is an element of $Q \times \mathbb{D}^r$, describing the current state and the content of each register. A local configuration (q, c) is *initial* if $q = q_{init}$ and all registers have the same content, i.e., there exists $d \in \mathbb{D}$ such that $c(i) = d$ for all $i \in [1, r]$.

Given a record transition $q \xrightarrow{\mathbf{rec}(m,\downarrow i)} q'$ and a datum d , we can apply δ to go from (q, c) to (q', c') by reading (m, d) if for all $j \in [1, r]$, $c(j) \neq d$ and $c'(i) = d$, and for all $j \neq i$, $c'(j) = c(j)$.

Given an equality transition $q \xrightarrow{\mathbf{rec}(m,=i)} q'$ and a datum d , we can apply δ to go from (q, c) to (q', c') by reading (m, d) if $c(i) = d$ and $c' = c$.

If one of those cases applies, we write $(q, c) \xrightarrow{\mathbf{rec}(m,d)}_\delta (q', c')$ and call it a *reception step*.

Given a broadcast transition $\delta = q \xrightarrow{\mathbf{br}(m,i)} q'$ and a datum d, we can apply δ to go from (q, c) to (q', c') by broadcasting (m, d) if $c(i) = d$ and $c' = c$. If those conditions are met we write $(q, c) \xrightarrow{\mathbf{br}(m,d)}_\delta (q', c')$ and call it a *broadcast step*.

A *local step* $(q, c) \xrightarrow{\mathbf{op}(m,d)}_\delta (q', c')$ between two local configurations is either a reception step or a broadcast step. A *local run* u of $\mathcal{R}$ is a sequence of local steps $u = (q_0, c_0) \xrightarrow{\mathbf{op}_1(m_1,d_1)}_{\delta_1} (q_1, c_1) \xrightarrow{\mathbf{op}_2(m_2,d_2)}_{\delta_2} \cdots \xrightarrow{\mathbf{op}_n(m_n,d_n)}_{\delta_n} (q_n, c_n)$. It is *initial* if (q_0, c_0) is an initial configuration. In that case the common datum d of registers in c_0 is called is called the *initial datum* of u. Its *input* $\mathbf{In}(u) \in (\mathcal{M} \times \mathbb{D})^*$ is the sequence of messages received by input transitions $(q_{i-1}, c_{i-1}) \xrightarrow{\mathbf{rec}(m_i,d_i)}_{\delta_i} (q_i, c_i)$ in u. Similarly, its *output* $\mathbf{Out}(u) \in (\mathcal{M} \times \mathbb{D})^*$ is the sequence of messages sent by output transitions $(q_{i-1}, c_{i-1}) \xrightarrow{\mathbf{br}(m_i,d_i)}_{\delta_i} (q_i, c_i)$ in u.

Its *d-input* $\mathbf{In}_d(u) \in \mathcal{M}^*$ is the sequence of letters associated to datum d in $\mathbf{In}(u)$, and its *d-output* $\mathbf{Out}_d(u) \in \mathcal{M}^*$ is the sequence of letters associated to datum d in $\mathbf{Out}(u)$.

▶ Remark 2. Record transitions can only be taken if the received value is not already in the registers. This is not a restriction on the expressivity: instead of storing the same datum in several registers, the system use its registers to store each datum once, and use a function $[1, r] \to [1, r]$, stored in the states, to assign registers to their content.

2.2 Broadcast Networks and Games with Registers

Let $r \in \mathbb{N}$ and let $\mathcal{R} = (Q, \mathcal{M}, q_{init}, \Delta)$ be a register transducer with r registers. The *broadcast network of register automata* (BNRA for short) described by $\mathcal{R}$ is the infinite transition system described below. We call register transducer *protocols* when we use them to define BNRA.

A *configuration* is a function $\gamma : \mathbb{A} \to Q \times \mathbb{D}^{\mathbf{r}}$ with $\mathbb{A}$ a finite set of *agents*. It maps each agent to a local configuration.

We write $\mathsf{st}(\gamma)$ for the state component of γ and $\mathsf{data}(\gamma)$ for its register component. A configuration γ is *initial* if for all $a \in \mathbb{A}$, $\mathsf{st}(\gamma)(a) = q_{init}$, $\mathsf{data}(\gamma)(a, i) = \mathsf{data}(\gamma)(a, i')$ for all i, i' and $\mathsf{data}(\gamma)(a, i) \neq \mathsf{data}(\gamma)(a', i')$ for all $a \neq a'$ and i, i'. Intuitively, each agents starts with a unique identifier that is contained in all of its registers.

Given two configurations γ, γ' over $\mathbb{A}$, a *step* $\gamma \to \gamma'$ is defined when there exist $a_0 \in \mathbb{A}$, $m \in \mathcal{M}$, $d \in \mathbb{D}$ and a transition $\delta_0 \in \Delta_O$ such that $\gamma(a_0) \xrightarrow{\mathbf{br}(m,d)}_{\delta_0} \gamma'(a_0)$, and for all $a \neq a_0$, either $\gamma'(a) = \gamma(a)$, or there is a transition $\delta \in \Delta_I$ such that $\gamma(a) \xrightarrow{\mathbf{rec}(m,d)}_{\delta} \gamma'(a)$.

A (global) *run* ϱ is a sequence of steps $\gamma_0 \to \gamma_1 \to \gamma_2 \cdots \gamma_{n-1} \to \gamma_n$. It is *initial* if γ_0 is an initial configuration. The *projection* of ϱ on an agent a is the local run $\pi_a(\varrho)$ made of all transitions taken by a in ϱ. We write $\varrho : \gamma \xrightarrow{*} \gamma'$ when ϱ is a run from γ to γ'.

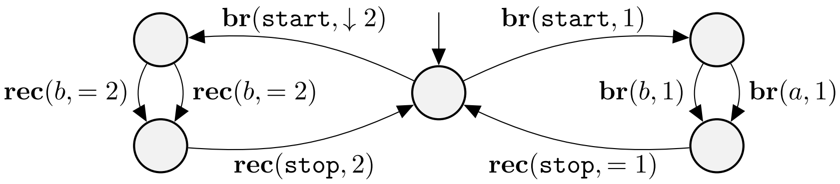

Figure 1 A protocol which can do two things: On the left, it receives `start` and a sequence of a and b while checking that they carry the same datum (i.e. come from the same sender). At any point it may stop by sending `stop` with the received datum, to confirm that the communication has been received. In the right part, it broadcasts a `start` message with its initial identifier, then a sequence of messages a and b with that same identifier. When it receives it back with the letter `stop`, it terminates.

▶ **Definition 3** (Coverability problem). *The coverability problem* Cover *asks, given a protocol $\mathcal{R}$ and a message m_{err}, whether there is an initial run in which m_{err} is broadcast.*

If there is an initial run ϱ in which an agent broadcasts m_{err}, then we say that ϱ *covers* m_{err}, and that m_{err} is *coverable*.

▶ Remark 4. The coverability problem is usually defined with an error state q_{err}: is there a an initial run where an agent reaches q_{err}? We define it with a message as it will be more convenient for some definitions, and the two versions are easily inter-reducible.

We will examine the controller synthesis problem on this model. The Cover problem defined earlier is the particular case in which no state is controllable.

▶ **Definition 5** (Broadcast Game with Registers). *A Broadcast Game with Registers with r registers $\mathcal{G} = (\mathcal{R}, Q_{ctrl}, Q_{env}, m_{err})$ is defined by a protocol with r registers $\mathcal{R} = (Q, \mathcal{M}, q_{init}, \Delta)$, a partition of its states $Q = Q_{\mathsf{ctrl}} \sqcup Q_{\mathsf{env}}$, and an error letter m_{err}.*

A *control strategy* for $\mathcal{G}$ is a function $\sigma : \Delta^* \to \Delta$ observing a sequence of transitions and choosing the next one [1].

[1] We choose to not give access to the data to Controller, as we want to be able to rename data at will. We will discuss the version of the game where Controller can see the data in Section 6.

A *σ-local run* is an initial local run $u = (q_0, c_0) \xrightarrow{\mathbf{op}_1(m_1,d_1)}_{\delta_1} \cdots \xrightarrow{\mathbf{op}_n(m_n,d_n)}_{\delta_n} (q_n, c_n)$ such that for all $i \in [1, n]$, if $q_{i-1} \in Q_{\mathsf{ctrl}}$ then $\sigma(\delta_1 \cdots \delta_{i-1}) = \delta_i$. A *$\sigma$-run* is an initial run whose projection on every agent a is a σ-local run.

A control strategy is *winning* if no σ-run covers m_{err}. In this game and all games we construct from it the player trying to construct a winning control strategy will be called Controller and her opponent Environment.

▶ **Definition 6** (Controller synthesis problem). *The safe strategy problem* SafeStrat *takes as input a BGR $\mathcal{G}$, and asks whether there is a winning control strategy for $\mathcal{G}$.*

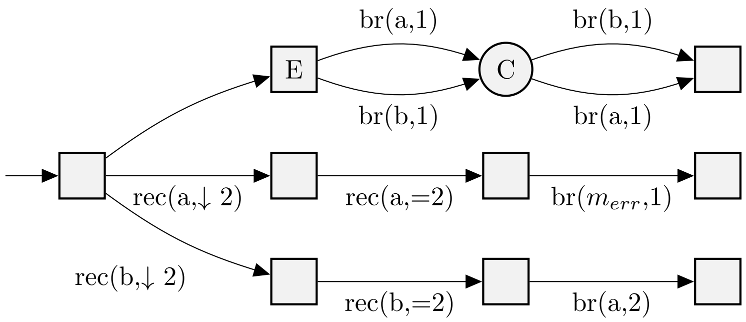

Figure 2 A BGR. The round state C belongs to Controller, square states to Environment.

▶ **Example 7.** In the BGR displayed in Figure 2, Controller has a single winning control strategy, which is to always choose a different letter from the one chosen by Environment from E. Indeed, doing otherwise would let an agent broadcast either aa or bb with its initial datum. In the first case, Environment could send an agent in the second row to receive aa and then broadcast m_{err}. In the second case, Environment could send two agents to the third row, who receive bb and broadcast a with the same datum. An agent in the second row could then receive both a broadcasts and then broadcast m_{err}. By contrast, it is easy to check that if Controller always picks a different letter from the one chosen by Environment in E, there cannot be two broadcasts of a or of b with the same datum. Hence agents sent to the second and third row will be unable to broadcast anything.

2.3 Subword order toolbox

A *well quasi-order* is a set equipped with a preorder relation $(S, \preceq)$ such that in every infinite sequence $s_0, s_1, \ldots$ there exist $i < j$ such that $s_i \preceq s_j$.

Given two words $v = a_1 \cdots a_m$ and $w = b_1 \cdots b_n$ in Σ^*, we say that v is a *subword* of w and write $v \sqsubseteq w$ if v can be obtained from w by removing letters, i.e., there are indices $i_1 < \cdots < i_m$ such that $v = b_{i_1} \cdots b_{i_m}$.

Given a set of words W, we define its *upward-closure* $W\!\uparrow = \{u \in \Sigma^* \mid \exists w \in W, w \sqsubseteq u\}$ and its *downward-closure* similarly $W\!\downarrow = \{u \in \Sigma^* \mid \exists w \in W, u \sqsubseteq w\}$. We say that W is *downward-closed* if $W = W\!\downarrow$, and *upward-closed* if $W = W\!\uparrow$. The set of minimal elements of an upward-closed set I is called its *basis*. Given a finite basis B, we define its *norm* as the maximum length of its words: $||B|| = \max\{|w| \mid w \in B\}$.

A seminal result in the study of well quasi-orders is *Higman's lemma* [15, 14], which states that $(\Sigma^*, \sqsubseteq)$ is a well quasi-order for all finite alphabet Σ. As a corollary, we obtain that every upward-closed set of words $I \subseteq \Sigma^*$ has a finite basis B such that $I = B\!\uparrow$. Another corollary is that the upward-closure of any language over a finite alphabet is regular. This is a consequence of the previous property, along with the following fact.

▶ **Lemma 8** (Folklore). *Given a finite set of words B over a finite alphabet Σ, one can construct a deterministic automaton $\mathcal{A}_{B\uparrow}$ recognising $B\uparrow$ with at most $(||B||+1)^{|B|}$ states.*

In this work we will show that our main problem is $\mathbf{F}_{\omega^\omega}$-complete, meaning that it is decidable but with a very high complexity, much higher than the Ackermann function for instance. For a formal definition of this complexity class $\mathbf{F}_{\omega^\omega}$ (and the related class of functions $\mathscr{F}_{\omega^\omega}$), see [22]. For our purpose, we will only use the Length function theorem, stated below.

A finite or infinite sequence of words $w_0, w_1, \ldots$ is *good* if there exist $i < j$ such that $w_i \sqsubseteq w_j$, and *bad* otherwise. Higman's lemma states that every bad sequence of words over a finite alphabet is finite, but we do not have a bound on their size. However, if we add a constraint so that each word can only have finitely many successors, then a uniform bound exists. Given a function $g : \mathbb{N} \to \mathbb{N}$ and an integer $n \in \mathbb{N}$, we say that a sequence of words $w_1, \ldots$ is *(g, n)-controlled* if $|w_i| \le g^{(i)}(n)$ for all $i \ge 1$ (where $g^{(i)}$ denotes g applied i times).

▶ **Theorem 9** (*Length Function Theorem* [23]). *Let Σ be a finite alphabet and $g : \mathbb{N} \to \mathbb{N}$ a primitive recursive function. There exists a function $f \in \mathscr{F}_{\omega^{|\Sigma|-1}}$ such that, for all $n \in \mathbb{N}$, every (g, n)-controlled bad sequence $w_1, w_2, \ldots$ has at most $f(n)$ terms.*

2.4 Games toolbox

We assume familiarity with automata and regular games, and simply fix some terms notations (see [11, Chapter 2] for an in-depth presentation). A *two-player game* $\mathcal{G}$ is given by a directed graph $G = (V, E)$ called the *arena*, along with a partition of V in two, $V = V_0 \sqcup V_1$, an initial vertex v_{init}, a colouring function $c : V \to C$ mapping vertices to a finite set of colours C, and a language $\mathcal{L} \subseteq C^\omega$ of infinite sequences of colours, called the *objective*. There are two players, called P_0 and P_1

A (finite or infinite path) in G starting in v_{init} is called a *play*. A play $v_0 \to v_1 \to \cdots$ is *winning* for P_0 if $c(v_0)c(v_1)\cdots \in \mathcal{L}$, and *losing* for P_0 otherwise.

A *strategy* for player P_i is a function $\sigma_\mathcal{G} : V^*V_i \to V$. A *$\sigma_\mathcal{G}$-play* is a path $v_0 \to v_1 \to \cdots$ in G such that for all $j \ge 1$, if $v_{j-1} \in V_i$ then $v_j = \sigma_\mathcal{G}(v_0 \cdots v_{j-1})$. A strategy for P_0 (resp. P_1) is *winning* if all infinite $\sigma_\mathcal{G}$-plays are winning (resp. losing) for P_0.

We call $\mathcal{G}$ a *reachability game* (resp. *safety game*), when the objective is of the form LV^ω (resp. $V^\omega \setminus LV^\omega$) with L a regular language of finite words (represented by a deterministic finite automaton). It is well-known that those games are *determined*, i.e., in every game one of the two players has a winning strategy.

▶ **Proposition 10** (Folklore). *One can compute the winner of a finite reachability game in polynomial time. Furthermore if P_0 has a winning strategy, then she has one that guarantees that she wins in at most $|V| \cdot |\mathcal{A}|$ steps.*

3 An introductory case: Broadcast networks without data

We start by showing the proof principles in an easy case, when processes do not have registers. In that case communication is made only through letters of $\mathcal{M}$. In this section we will forget the data in messages, and only consider letters. We simplify notations: we write $\mathbf{br}(m)$ for a broadcast of letter m and $\mathbf{rec}(m)$ for a reception of m. We obtain *Reconfigurable Broadcast Networks*, as introduced in [9]. From now on we will use the term RBN for this model.

COVER has been shown decidable and P-complete for those systems [9]. We could not find a result in the literature stating that SAFESTRAT is NP-complete, but closely-related results were proven in [5] and [24]. We prove it to illustrate our method.

To begin with, we show that we can characterise winning control strategies as the ones which force the set of letters sent to stay within some set $I \subseteq \mathcal{M} \setminus \{m_{err}\}$, called an *invariant*.

▶ **Definition 11** (Invariants for RBN). *An invariant for an RBN over alphabet $\mathcal{M}$ is a set of letters $I \subseteq \mathcal{M}$. We say that it is sufficient for a control strategy σ if:*

- *$m_{err} \notin I$, and*
- *If a σ-local run receives only messages of I then it broadcasts only messages of I.*

▶ **Lemma 12** (Invariants characterise winning control strategies). *A control strategy σ is winning if and only if there exists a sufficient invariant $I \subseteq \mathcal{M}$ for it.*

Proof sketch. Suppose σ is winning. Let I be the set of messages such that there exists a σ-run in which they are broadcast. As σ is winning, $m_{err} \notin I$. If a σ-local run u receives only messages of I, then we can build a σ-run where an agent follows u: for each letter m received in u, we make other agents execute a σ-run where m is broadcast: this is possible as $m \in I$. We match this broadcast with the reception in u. We obtain a σ-run where all letters broadcast in u are broadcast.

For the other direction, suppose by contradiction that σ has a sufficient invariant I and that there is a σ-run ϱ in which m_{err} is broadcast. Let m be the first message broadcast in ϱ that is not in I, and a the agent broadcasting it. Those are well-defined as $m_{err} \notin I$. Let ϱ' be the prefix of ϱ stopping right after that broadcast. The projection $\pi_a(\varrho')$ of ϱ' on agent a contains a broadcast of m but no reception of any $m' \notin I$, a contradiction. ◀

This lets us turn the distributed game into a sequential one: If we are given an invariant I, checking whether there is a strategy that maintains it comes down to a two-player safety game. We obtain an algorithm for strategy synthesis: guess an invariant, and then solve the resulting safety game, which can be done in polynomial time.

▶ **Theorem 13.** *Deciding the winner of a BGR without registers is* NP*-complete.*

Proof. For the upper bound, by Lemma 12, it suffices to guess a set $I \subseteq \Sigma$ such that $m_{err} \notin I$ and then check if there is a strategy that guarantees that we can only broadcast a message outside of I if we received one beforehand. This is easily encoded into a safety game: Take the states and transitions of the BGR, without the operations, add a sink state with no outgoing transitions, and redirect every reception of a message $m \notin I$ to it. The objective of the first player is to avoid transitions broadcasting letters of $\Sigma \setminus I$.

Clearly there is a winning control strategy for the BGR if and only if there is an invariant I and a strategy avoiding transitions broadcasting messages outside of I in this safety game. This can be checked in polynomial time, by Proposition 10.

The lower bound is shown in Appendix A ◀

To conclude this section, we present an argument in favour of *parameterized* distributed synthesis. The fact that we have an arbitrary amount of agents makes the existence of a winning control strategy less likely. One might wonder what happens if we simply want a strategy that works for a bounded, or even fixed amount of agents. We show that the problem becomes undecidable in this case, even for 3 agents. Hence considering arbitrary numbers of agents can be a good approximation as it spectacularly reduces the difficulty of the problem.

▶ **Theorem 14.** *Given a BGR $\mathcal{G} = (\mathcal{R}, Q_{\mathsf{ctrl}}, Q_{\mathsf{env}}, m_{err})$, it is undecidable whether there is a control strategy such that no σ-run* with 3 agents *covers m_{err}.*

4 Signature BGR

In this section we establish decidability of SafeStrat in a subcase of interest, that illustrates well the decidability proof for the general case, while requiring less technical complications.

▶ **Definition 15.** *A signature protocol is one where every broadcast is made with the value of register 1, and all receptions are made on other registers. The associated BGR are called signature BGR.*

In other words, there are no transitions of the form $\xrightarrow{\mathbf{br}(m,i)}$ with $i \geq 2$ or $\xrightarrow{\mathbf{rec}(m,\downarrow 1)}$ or $\xrightarrow{\mathbf{rec}(m,=1)}$. Such a protocol keeps its initial datum in register 1 and uses it for broadcasts, while the other registers are used to store and compare received values. An interesting property of those systems is that the datum of a message identifies its sender: Each agent only sends messages with its initial datum, and since those are unique, messages containing the same datum necessarily come from the same agent. In this section we will call *output* the d-output of a local run u with d its initial datum, and write it $\mathbf{Out}_{sign}(u)$.

▶ **Theorem 16.** *The SafeStrat problem is decidable $\mathbf{F}_{\omega^\omega}$-complete for signature BGR.*

The lower bound is provided by [13], as they show $\mathbf{F}_{\omega^\omega}$-hardness already for Cover with $r = 2$ registers. Fix $\mathcal{G} = (\mathcal{R}, Q_{\mathsf{ctrl}}, Q_{\mathsf{env}}, m_{err})$ a BGR with r registers. To prove the theorem, we once again use a characterisation of winning strategies in terms of invariants. Here an invariant is a downward-closed set of words of $\mathcal{M}^*$. A witness for non-coverability of a message m_{err} is a downward-closed set I of words that contains ε and not m_{err} and such that an agent whose d-inputs are all in I has an output in I[2]. Intuitively, if all agents respect that condition, then we can only obtain runs where all agents output words in the invariant, and thus no-one broadcasts m_{err}.

The downward-closed property comes from the fact that if an agent outputs a word w, then other agents can receive any subsequence of letters of that word, as broadcasts can be lost. It is crucial as it gives us a finite representation of invariants, their basis.

▶ **Definition 17** (Invariants for signature BGR)**.** *An invariant for a signature BGR over an alphabet $\mathcal{M}$ is a downward-closed set $I \subseteq \mathcal{M}^*$. We say that it is sufficient for a control strategy σ if it satisfies the following conditions:*

1. *$\varepsilon \in I$ and $m_{err} \notin I$*
2. *For all σ-local run u, if $\mathbf{In}_d(u) \in I$ for all $d \in \mathbb{D}$ then $\mathbf{Out}_{sign}(u) \in I$.*

The next step is to show that a winning strategy always comes with a sufficient invariant.

▶ **Lemma 18** (Invariants characterise winning strategies)**.** *A control strategy σ is winning if and only if there exists a sufficient invariant $I \subseteq \mathcal{M}^*$ for it.*

Recall that, as a corollary of Higman's lemma, upward-closed sets of words can be finitely described by their finite basis.

To solve SafeStrat, we cannot enumerate potential strategies as there are uncountably many. Instead, our algorithm enumerates invariants (represented by the basis of their

[2] As I is downward-closed, $\varepsilon \in I$ is synonymous with I being non-empty.

complement) and checks for each one whether there is a strategy such that the conditions listed in Lemma 18 are satisfied. While the first item is straightforward to check, the second is not. To verify it, we design a game in which the two players construct a local run, and the received data are chosen by Environment.

4.1 Invariant game for signature BGR

Intuitively, the two players construct a local run by picking the transitions from their respective states, and Environment picks the data received at each step, when they are not already determined by the chosen transition. If at some point the d-input gets out of I for some d then the game stops and Controller wins. If the output gets out of I then the game stops and Environment wins. If none of the two happen and the game goes on forever then Controller wins. This characterises the capacity of Controller to keep outputs within a given invariant, but if we made the choice of data explicit this game would be infinite.

We reduce it to a finite reachability game, called the invariant game which we can solve by a simple fix-point computation.

A first observation is that it is always in Environment's best interest to choose fresh data that were never seen before, as they come with the smallest d-input. Thus whenever we receive a datum that is not in the registers we can assume that the associated d-input is empty. This means that we do not need to remember the d-inputs associated to every datum of the local run, but only those that are currently in the registers.

To formalise this, we need to define the inputs and output of sequences of transitions. The idea is that we can assume that every datum that disappears from the registers will never appear again. In order to check whether some d-input gets out of I, we only need to keep track of the sequences of letters received with the data currently in the registers. We call those the recent inputs. Furthermore, in our model of register transducers, a received datum always appears in at most one register at a time, and while it is not forgotten, it stays in that one register. This will let us read the recent inputs directly from the sequence of transitions.

Given a sequence of transitions $\delta_1 \cdots \delta_k$ of $\mathcal{R}$, we define its *output* as the sequence of letters sent by broadcasts. For all registers $i \in [1, r]$, we also define its *recent input on i* as the sequence of letters received with an equality transition with register i since it was last updated. Formally, the output of $\delta_1 \cdots \delta_k$ is defined inductively as $\mathsf{Out}(\varepsilon) = \varepsilon$ and

$$\mathsf{Out}(\delta_1 \cdots \delta_{k+1}) = \begin{cases} \mathsf{Out}(\delta_1 \cdots \delta_k) & \text{if } \delta_{k+1} \text{ is a reception,} \\ \mathsf{Out}(\delta_1 \cdots \delta_k)m & \text{if } \delta_{k+1} = \xrightarrow{\mathbf{br}(m,1)} \text{for some } m. \end{cases}$$

The recent input on i is defined as $\mathsf{recentIn}_i(\varepsilon) = \varepsilon$ and:

$$\mathsf{recentIn}_i(\delta_1 \cdots \delta_{k+1}) = \begin{cases} m & \text{if } \delta_{k+1} = \xrightarrow{\mathbf{rec}(m,\downarrow i)} \text{ for some } m \text{ and } i, \\ \mathsf{recentIn}_i(\delta_1 \cdots \delta_k)m & \text{if } \delta_{k+1} = \xrightarrow{\mathbf{rec}(m,=i)} \text{ for some } m \text{ and } i, \\ \mathsf{recentIn}_i(\delta_1 \cdots \delta_k) & \text{otherwise.} \end{cases}$$

Note that we always have $\mathsf{recentIn}_1(\delta_1 \cdots \delta_k) = \varepsilon$, as we assumed that no reception is made using register 1.

The *invariant game* $\mathcal{IG}(\mathcal{G}, I)$ goes as follows. The set of vertices is simply $Q_{\mathcal{R}}$. From each vertex $q \in Q_{\mathcal{R}}$, players choose a transition from q in $\Delta_{\mathcal{R}}$.. Controller chooses the next transition when the current vertex is in Q_{ctrl}, Environment when it is in Q_{env}.

- If at some point the play $\pi = \delta_1 \cdots \delta_k$ is such that $\mathsf{recentIn}_i(\pi) \notin I$ for some $i \geq 2$ then Controller wins.
- If at some point the play $\pi = \delta_1 \cdots \delta_k$ is such that $\mathsf{Out}(\pi) \notin I$ then Environment wins.
- If the play goes on forever without any of those things happening then Controller wins.

We start by showing that we can solve this game by considering it as a regular safety game. We obtain as a corollary that if Environment wins then he can win in a bounded number of steps. Define $\varphi(\mathcal{R}, B) := |\mathcal{R}|(||B|| + 1)^{|\mathcal{R}|(|B|+1)}$

▶ **Lemma 19** (Decidability of the invariant game)**.** *Given a BGR over protocol $\mathcal{R}$ and a finite set of words B, we can decide in exponential time whether Controller has a winning strategy in $\mathcal{IG}(\mathcal{G}, (B\uparrow)^{\mathsf{c}})$. Furthermore, if Environment has a winning strategy then he has a strategy to win in at most $\varphi(\mathcal{R}, B)$ steps.*

Proof. By Lemma 8, $B\uparrow$ is a regular language, recognised by a deterministic finite automaton $\mathcal{A}_{B\uparrow} = (Q_B, \mathcal{M}, \Delta_B, q_0^B, F_B)$ with $(||B|| + 1)^{(|B|+1)}$ states.

We can construct a deterministic automaton $\mathcal{B}$ over the alphabet $\Delta_{\mathcal{R}}$ that reads plays $\delta_1 \cdots \delta_k$ of $\mathcal{IG}(\mathcal{G}, (B\uparrow)^{\mathsf{c}})$ and accepts exactly the winning plays for Environment. Its set of states is $(Q_B)^r$, plus a rejecting sink state $\bot$ and an accepting sink state $\top$, which is the only accepting state. The first component keeps track of the state reached in $\mathcal{A}_{B\uparrow}$ by the output of the sequence of transitions. The others keep track, for each register $i \geq 2$, of the state reached by the recent input on i in $\mathcal{A}_{B\uparrow}$. Transitions of that automaton are easy to infer from the definition of output and recent input on i. The automaton goes to $\bot$ if the recent input on i is in $B\uparrow$ for some i, or if it sees a reception transition of a message $m \in B\uparrow$. It goes to $\top$ if the output is in $B\uparrow$.

By Proposition 10, we can solve this game in polynomial time in the size of the automaton $\mathcal{B}$ and the size of the arena of $\mathcal{IG}(\mathcal{G}, (B\uparrow)^{\mathsf{c}})$ (i.e., $|\mathcal{R}|$), that is, in exponential time in $||B|| + |B| + |\mathcal{R}|$. Furthermore, if Environment has a winning strategy then he has one that guarantees that he wins in at most $\varphi(\mathcal{R}, B) = |\mathcal{A}_{B\uparrow}|^r |\mathcal{R}|$ steps. ◀

We have two things to prove: First that a winning strategy for Controller in $\mathcal{IG}(\mathcal{G}, I)$ yields a control strategy σ for which I is a sufficient invariant. Then, that a winning strategy for Environment in $\mathcal{IG}(\mathcal{G}, I)$ implies that there is no control strategy for which I is a sufficient invariant.

▶ **Lemma 20.** *Let $I \subseteq \mathcal{M}^*$ be a downward-closed set of words containing ε and not m_{err}. If Controller wins the invariant game $\mathcal{IG}(\mathcal{G}, I)$ then there is a control strategy σ such that I is a sufficient invariant for σ.*

▶ **Lemma 21.** *Let σ be a control strategy. Let $I \subseteq \mathcal{M}^*$ be a downward-closed set of words containing ε and not m_{err}, and let B be the basis of I^{c}.*
If Environment wins the invariant game $\mathcal{IG}(\mathcal{G}, I)$ then there is a σ-local run of length at most $\varphi(\mathcal{R}, B)$ with an output not in I and all d-inputs in I.

Proof. By Proposition 10 there exists $\tau_{\mathcal{IG}}$ a winning strategy $\tau_{\mathcal{IG}}$ for Environment in the invariant game $\mathcal{IG}(\mathcal{G}, I)$ such that Environment always wins in at most $\varphi(\mathcal{R}, B)$ steps.

We construct a σ-local run of length at most $\varphi(\mathcal{R}, B)$ with an output not in I and all d-inputs in I. To do so, we apply $\tau_{\mathcal{IG}}$ to choose transitions and we choose data by always picking a datum never seen before in the run, when the datum is not determined by the transition.

Let (s_0, c_0) be an initial configuration of $\mathcal{R}$. We define iteratively a sequence of steps $(s_{k-1}, c_{k-1}) \xrightarrow{\mathbf{op}_k(m_k, d_k)}_{\delta_k} (s_k, c_k)$ as follows. Suppose we defined them up to (s_{k-1}, c_{k-1}), and let u_{k-1} be the local run defined so far. We first choose δ_k: If $s_{k-1} \in Q_{\mathsf{ctrl}}$ then $\delta_k = \sigma(u_{k-1})$, otherwise $\delta_k = \tau_{\mathcal{IG}}(\delta_1 \cdots \delta_{k-1})$.

We then choose d_k:

- If δ_k is a broadcast transition of letter m, we set $d_k = c_k(1)$ (the initial datum of the local run).
- If δ_k is a record transition, we pick a datum d_k that does not appear in u_{k-1} before.
- If $\delta_k = s_{k-1} \xrightarrow{\mathbf{rec}(m,=i)} s_k$ is an equality transition of letter m, we set $d_k = c_{k-1}(i)$.

Clearly we maintain the fact that u_k is a σ-local run and $\delta_1 \cdots \delta_k$ is a $\tau_{\mathcal{IG}}$-play in $\mathcal{IG}(\mathcal{G}, I)$. We stop when $\delta_1 \cdots \delta_k$ is winning for Environment in $\mathcal{IG}(\mathcal{G}, I)$, which happens for some $K \leq \varphi(\mathcal{R}, B)$. Let $u = u_K$ be the local run obtained at the end.

It remains to show that the output of u is not in I while all its d-inputs are in I. To do so, we rely on the following claim:

▷ Claim 22. For all register i and index k, $\mathsf{recentIn}_i(\delta_1 \cdots \delta_k) = \mathbf{In}_{u_k}(c_k(i))$. Furthermore, $\mathsf{Out}(\delta_1 \cdots \delta_k) = \mathbf{Out}_{sign}(u_k)$

Proof. By a straightforward induction on k. ◀

By definition $\delta_1 \cdots \delta_K$ is a winning $\tau_{\mathcal{IG}}$-play for Environment, hence its output is not in I, thus $\mathbf{Out}_{sign}(u) = \mathbf{Out}_{sign}(u_K)$ is not in I either. Let $d \in \mathbb{D}$ a datum appearing in u, and let k be such that (s_k, c_k) is the last configuration in which d appears. Let i be the register such that $c_k(i) = d$. Then we have $\mathbf{In}_u(d) = \mathbf{In}_{u_k}(c_k(i)) = \mathsf{recentIn}_i(\delta_1 \cdots \delta_k)$. As $\tau_{\mathcal{IG}}$ is winning for Environment, $\mathsf{recentIn}_i(\delta_1 \cdots \delta_k) \in I$, and thus $\mathbf{In}_u(d) \in I$.

We have found a σ-local run of length at most $\varphi(\mathcal{R}, B)$ whose output is not in I while all its d-inputs are. ◀

Our next step is to bound the minimal size of a sufficient invariant for some winning control strategy σ when there is one. The idea is as follows: Take an invariant I such that the basis $\{w_1, \ldots, w_k\}$ of I^{c} has as few elements as possible. We can assume that $|w_1| \leq \cdots \leq |w_k|$. Then we know that, for all i, $\{w_1, \ldots, w_i\}$ is not a sufficient invariant for σ. Hence by Lemma 21 we get a σ-local run of bounded size breaking the invariant $\{w_1, \ldots, w_i\}$, which forces $\{w_{i+1}, \ldots, w_k\}$ to contain a word of bounded size. This bounds the size of w_{i+1} with respect to $w_1, \ldots, w_i$, as stated in the lemma below.

Define $\psi(n) = |\mathcal{R}|(n+1)^{|\mathcal{M}|^{n+1}+1}$

▶ **Lemma 23** (Bounding the size of the invariant)**.** *Let $\mathcal{G}$ a signature BGR. There is a winning control strategy for $\mathcal{G}$ if and only if there is a sequence of words $w_0, \ldots, w_k \in \mathcal{M}^*$ such that*

- *Controller wins $\mathcal{IG}(\mathcal{G}, \{w_1, \ldots, w_k\}\uparrow^{\mathsf{c}})$,*
- *and for all $i \in [1, k]$, $|w_i| \leq \psi(|w_{i-1}|)$.*

We will can now leverage the Length Function Theorem to bound the size of the basis of I^{c} in Lemma 23.

▶ **Theorem 24.** *SafeStrat is decidable and in $\mathbf{F}_{\omega^\omega}$ for signature BGR.*

Proof. Let $\mathcal{G}$ a BGR. We apply the Length Function Theorem with $\Sigma = \mathcal{M}$ and $g(n) = n(n+1)^{n^{n+1}+1}$. We obtain a function $f \in \mathscr{F}_{\omega^{|\mathcal{M}|-1}}$ such that every (g, n)-controlled bad sequence of words $w_0, w_1, ..., w_k$ has at most $f(n)$ terms.

We use a non-deterministic algorithm that guesses a sequence of words $w_1, ..., w_k$ such that $w_1 = m_{err}$ and $|w_i| \leq |w_{i+1}| \leq \psi(|w_i|)$ for all i. One can straightforwardly check that then we have $|w_i| \leq g^{(i)}(|\mathcal{R}| + |\mathcal{M}| + 1)$ for all i.

Let $B = \{w_0, w_1, ..., w_k\}$. The algorithm checks that there exists a strategy σ such that the complement of $\{w_0, w_1, ..., w_k\}\uparrow$ is a sufficient invariant for σ, by solving the invariant game $\mathcal{IG}(\mathcal{G}, (\{w_0, w_1, ..., w_k\}\uparrow)^c)$. This can be done in exponential time in $|\mathcal{R}| + k + |w_k|$, by Lemma 19. We accept if there is such a strategy and reject otherwise.

By Lemma 23, this algorithm is correct. We can make it deterministic with an exponential blow-up in the time complexity. The time required by this algorithm is therefore $h(f(|\mathcal{R}| + |\mathcal{M}|+1))$ with h a primitive recursive function. As $\mathscr{F}_{\omega^{|\mathcal{M}|-1}}$ is closed under composition with primitive recursive functions, the algorithm takes a time bounded by a function of $\mathscr{F}_{\omega^{|\mathcal{M}|-1}}$. As a consequence, the problem is in $\mathbf{F}_{\omega^\omega}$ (see [22] for details). ◀

5 General case

In this section we generalise the previous result to all BGR.

▶ **Theorem 25.** *The* SAFESTRAT *problem is decidable in $\mathbf{F}_{\omega^\omega}$ for general BGR.*

We fix a BGR $\mathcal{G} = (\mathcal{R}, Q_{ctrl}, Q_{env}, m_{err})$ for the rest of this section.

The general structure of the proof is the same as before, but the removal of the signature hypothesis makes it significantly more technical. The main difference between the signature and general models is that in the latter a process can send acknowledgements to a process it received messages from, as in the right protocol in Figure 1.

We make the following observation: Say an agent receives a message (m, d) with d its initial datum; this is possible in general BGR but not in signature ones. Then this means that other agents, which did not have this datum initially, received enough messages with datum d to be able to broadcast (m, d). Intuitively, we can copy these agents many times, which allows us to assume that we have an unlimited supply of messages (m, d). In sum, we will show that if an agent sends a message (m, d) with d that is not its initial datum, then from this point on we can assume that messages (m, d) are for free. This intuition justifies the definition of decomposition, which summarises the sequence of letters sent with a given datum during a run. It details the sequence of letters sent by the agent with that datum initially, and the points at which each letter is first broadcast with that datum by another agent. These decompositions were already used for the verification of those systems [13].

▶ **Definition 26.** *A decomposition is a tuple* $\mathsf{dec} = (v_0, m_1, \ldots, v_{k-1}, m_k, v_k)$ *with* $m_0, \ldots, m_k$ *distinct letters of* $\mathcal{M}$ *and* $v_i \in \mathcal{M}^*$ *for all* i.

A word $w \in \mathcal{M}^*$ *matches* dec *if* $w = w_0 \cdots w_k$ *where each* w_i *can be obtained by inserting letters from* $\{m_1, \ldots, m_i\}$ *in* v_i.

▶ **Example 27.** Let $\mathcal{M} = \{a, b, c\}$. Then $\mathsf{dec} = (abba, a, cbc, b, cc)$ is a decomposition. The word $abbacabaacbabcbca$ matches dec as we can cut in in three parts *abba**cabaac**babcbca*, and *cabaac* can be obtained by adding some a to cbc and *babcbca* can be obtained by adding some a and b to cc.

We write $\mathcal{L}_{\mathsf{dec}}$ for the language of words that match dec. Given a family of upward-closed sets of words $(J_m)_{m\in\mathcal{M}}$, we define $\mathcal{D}((J_m)_{m\in\mathcal{M}})$ as the set of decompositions

$$\mathcal{D}((J_m)_{m\in\mathcal{M}}) = \{(v_0, m_1, \ldots, v_{k-1}, m_k, v_k) \mid \forall i, \mathcal{L}_{(v_0,m_1,\ldots,v_{i-1})} \cap J_{m_i} \neq \emptyset\}.$$

With an additional downward-closed set I, we also define

$$\mathcal{D}(I, (J_m)_{m\in\mathcal{M}}) = \{(v_0, m_1, \ldots, v_{k-1}, m_k, v_k) \mid v_0 \cdots v_k \in I, \forall i, \mathcal{L}_{(v_0,m_1,\ldots,v_{i-1})} \cap J_{m_i} \neq \emptyset\}.$$

Finally, the set of words producible by $I, (J_m)_{m\in\mathcal{M}}$ is

$$\mathcal{L}(I, (J_m)_{m\in\mathcal{M}}) = \bigcup_{\mathsf{dec}\in\mathcal{D}(I,(J_m)_{m\in\mathcal{M}})} \mathcal{L}_{\mathsf{dec}}.$$

We say that a local run u with initial datum d is *compatible* with a decomposition $\mathsf{dec} = (v_0, m_1, \ldots, v_{k-1}, m_k, v_k)$ if $u = u_0 \cdots u_k$ where $v_i \sqsubseteq \mathbf{Out}_d(u_i)$ and $\mathbf{In}_d(u_i) \in \{m_1, \ldots, m_i\}^*$ for all i.

Here I should be thought of as the set of words over $\mathcal{M}$ that can be broadcast by an agent with its initial datum. Meanwhile, J_m represents the set of words w over $\mathcal{M}$ such that an agent can broadcast (m, d) with d not its initial datum while having received before only (a subword of) w with that datum. It can be read as the "cost" of a message m: in order to receive a message (m, d) you should first broadcast a sequence of letters of J_m with datum d.

A decomposition $(v_0, m_1, \ldots, v_k)$ is a scenario of the sequence of letters broadcast over a datum d during a run: The agent who has d as initial datum broadcasts $v_0 \cdots v_k$ with it, while $m_1, \ldots, m_k$ mark the points at which each of those letters is first broadcast with datum d by another agent.

Then, we can see $\mathcal{D}(I, (J_m)_{m\in\mathcal{M}})$ as the set of decompositions $(v_0, m_1, \ldots, v_k)$ that are compatible with the invariant $I, (J_m)_{m\in\mathcal{M}}$. The condition $v_0 \cdots v_k \in I$ means that an agent with d as initial datum should be able to broadcast $v_0 \cdots v_k$ with it. The other condition says that for all i there is a word $w \in \mathcal{L}_{(v_0,m_1,\ldots,v_{i-1})} \cap J_{m_i}$. This should be read as follows:

- $w \in J_{m_i}$ means that if we can broadcast the sequence w with datum d, we can make an agent broadcast (m_i, d)
- $w \in \mathcal{L}_{(v_0,m_1,\ldots,v_{i-1})}$ means that we can broadcast the sequence w with datum d, as we can obtain it from $v_0 \cdots v_{i-1}$ by adding enough $m_1, \ldots, m_{i-1}$.

5.1 Characterisation of winning strategies with invariants

An *invariant* for general BGR is made of a downward-closed set of words $I \subseteq \mathcal{M}^*$ (the sequences of letters that may be produced over some datum) and an upward-closed set of words $J_m \subseteq \mathcal{M}^*$ for each letter m (the sequences of letters that allow an agent to send a message (m, d) with d that is not its initial datum).

▶ **Definition 28** (Invariants for BGR). *An invariant for general BGR is a pair $(I, (J_m)_{m\in\mathcal{M}})$ with $I \subseteq \mathcal{M}^*$ a downward-closed set of words and, for all m, $J_m \subseteq \mathcal{M}^*$ an upward-closed set of words. We say that it is sufficient for a control strategy σ if the following conditions hold.*

1. *$\varepsilon \in I$, $m_{err} \notin I$ and $J_{m_{err}} \cap I = \emptyset$*
2. *$\mathcal{L}(I, (J_m)_{m\in\mathcal{M}}) \subseteq I$*
3. *For all initial σ-local run u with initial datum d, if:*
   - **(i)** *u is compatible with a decomposition $\mathsf{dec} \in \mathcal{D}((J_m)_{m\in\mathcal{M}})$, and*
   - **(ii)** *for all $d' \neq d$, $\mathbf{In}_{d'}(u) \in I$,*

   then we have that
   - **(a)** *$\mathbf{Out}_d(u) \in I$*
   - **(b)** *for all $m \in \mathcal{M}$ and $d' \neq d$, if u contains a broadcast of (m, d') then $\mathbf{In}_{d'}(u) \in J_m$.*

We once again prove that every winning control strategy has a sufficient invariant. The proof is presented in Appendix C

▶ **Lemma 29** (Invariants characterise winning strategies). *A control strategy σ is winning if and only if there exists a sufficient invariant $(I, (J_m)_{m\in\mathcal{M}})$ for it.*

5.2 The invariant game

We have characterised winning control strategies using invariants.The next step is to consider an invariant $(I, (J_m)_{m\in\mathcal{M}})$ and show that we can construct a game in which the two players determine whether there is a control strategy for which this invariant is sufficient.

We proceed as in Section 4. First we consider a game played on $\mathcal{R}$ where players pick a sequence of transitions (the next transition is chosen by the player owning the current state), and Environment picks the data when needed. The goal of Environment is to eventually obtain a local run u that satisfies either 3a or 3b but neither 3i nor 3ii, i.e., contradicting the invariant. The goal of Controller is to avoid this forever.

We define formally the invariant game $\mathcal{IG}(\mathcal{G}, I, (J_m)_{m\in\mathcal{M}})$ for general BGR in Appendix C. We show that Controller wins that game if and only if she has a control strategy in $\mathcal{G}$ for which $(I, (J_m)_{m\in\mathcal{M}})$ is sufficient. Furthermore, we show that when Environment wins we can obtain a local run contradicting the invariant of bounded size with respect to $\mathcal{G}$, I and $(J_m)_{m\in\mathcal{M}}$. This lets us bound the size of a sufficient invariant when it exists, using the Length Function Theorem.

▶ **Theorem 30** (Main theorem). *SafeStrat is decidable and $\mathbf{F}_{\omega^\omega}$-complete.*

6 Allowing agents to see data

So far we only considered control strategies that chose transitions based on the previous sequence of transitions, and not the sequence of data received. It is natural to wonder what happens if we use strategies of the form $\sigma : (\Delta\mathbb{D}^r)^* \to \Delta$. For this section we will only consider the signature case to make things easier. We conjecture that the following proof can be adapted to the general case.

A central ingredient in this proof is Ramsey's theorem on infinite hypergraphs, which extends naturally Ramsey's theorem on graphs [20]. It states that if we colour every subset of size k of an infinite set while using finitely many colours, then there is an infinite subset in which every k-subset has the same colour.

We now define *data-aware control strategies*. They are functions $\sigma : \mathbb{D}(\Delta \times \mathbb{D})^* \to \Delta$. The next transition is chosen based on the local run taken so far, including the initial datum and the data received. Notions of σ-local runs and σ-runs are extended naturally.

▶ **Theorem 31.** *There is a winning data-aware control strategy for $\mathcal{G}$ if and only if there is a winning control strategy for $\mathcal{G}$.*

Proof sketch. We show that there is a function h such that whenever a strategy is losing there is a losing σ-run of a certain shape where each agent has a local run of length at most $h(|\mathcal{G}|)$. Assume we have a winning data-aware control strategy. We then colour every set of $h(|\mathcal{G}|)$ data according to the behaviour of that strategy on local runs of length $\leq h(|\mathcal{G}|)$ where those data appear. We apply Ramsey's theorem to obtain an infinite set of data on which the strategy behaves the same on "short" local runs. This defines a control strategy which does not fail on runs where local runs are of length $\leq h(|\mathcal{G}|)$. By definition of h, the resulting strategy is winning. ◀

By combining this with Theorem 16, we obtain the following result.

▶ **Corollary 32.** *The existence of a winning data-aware control strategy for a BGR is decidable and $\mathbf{F}_{\omega^\omega}$-complete.*

7 The case of one register

The construction of [13] for the $\mathbf{F}_{\omega^\omega}$ lower bound only requires two registers. We studied in Section 3 the complexity of those problems when protocols do not have registers. The remaining gap is for protocols with one register. We call them *1BGR*. They offer an intermediate step in terms of tractability and expressivity between the protocols without registers and the general case. Those protocols can sign messages with their initial identifier, and check that several messages have the same datum, but not simultaneously. They relate to Petri nets and population protocols with data, as those only allow each process to store one datum. In particular, the subclass of *IO population protocols with data* can be seen as a particular case of BGR with one register.

We investigate the complexity of SafeStrat for 1BGR. In this case, a record transition essentially resets the memory of the process. This lets us split the invariant game used in the general case into simpler games: the output game and the echo games.

▶ **Theorem 33.** *SafeStrat is NExpTime-complete for 1BGR.*

A strategy $\sigma : V^* \to V$ for a two-player game is *positional* if its output only depends on the current state, that is, for all $w, w' \in V^*$ and $v \in V$ we have $\sigma(wv) = \sigma(w'v)$. We rely on the following criterion, which can be used to show that a player can win with a positional strategy. A language $\mathcal{L}$ is *submixing* (or *concave*) if whenever we have words $u_0, u_1, \dots$ and $v_0, v_1, \cdots$ such that $u_0 u_1 \cdots \notin \mathcal{L}$ and $v_0 v_1 \cdots \notin \mathcal{L}$ then $u_0 v_0 u_1 v_1 \cdots \notin \mathcal{L}$. It was shown in [17] that if an objective is submixing then player P_0 has a positional optimal strategy in all games with this objective.

We rely on the characterisation of winning control strategies by the invariant game, as stated in Lemma 45 and 48. It turns out that for 1BGR, the invariant game can be split into several simpler games. Essentially, we consider the recording of a new value in the register as a reset of the game. We define two different games: in the output game the players build the part of the local run before the first record transition. In the echo game the players build an interval of the local run between two record transitions.

We show that in the first game Controller can always use a positional strategy (Lemma 57) while in the second one it is Environment who can stick to positional strategies (Lemma 58). In both cases we use the submixing property of their objectives to prove it.

We also prove that the winners of those games determine the winner of the 1BGR (Lemma 59). The positionality of Environment's strategy in the echo game then lets us bound the size of the invariants necessary to witness the existence of a winning control strategy for Controller (Lemma 62). We exhibit an NExpTime algorithm, in which the non-deterministic guess is the invariant and a positional strategy for Controller in the output game. The lower bound follows from a reduction from the exponential grid tiling problem.

8 Conclusion

We showed decidability of SafeStrat for a powerful parameterised distributed model. We showcased a method for distributed controller synthesis through invariants by using it for increasingly complex versions of the model. We also match every resulting complexity class with a lower bound, which tends to show that this method makes sense for this model. The most promising future direction is to develop invariants for other models of distributed systems in order to obtain more decidability results. We can also investigate the relation between other distributed models with data and BGR, especially 1BGR.

References

1. Georg Bachmeier, Michael Luttenberger, and Maximilian Schlund. Finite automata for the sub- and superword closure of cfls: Descriptional and computational complexity. In Adrian-Horia Dediu, Enrico Formenti, Carlos Martín-Vide, and Bianca Truthe, editors, *Language and Automata Theory and Applications - 9th International Conference, LATA 2015, Nice, France, March 2-6, 2015, Proceedings*, volume 8977 of *Lecture Notes in Computer Science*, pages 473–485. Springer, 2015. doi:10.1007/978-3-319-15579-1_37.
2. A. R. Balasubramanian and Chana Weil-Kennedy. Reconfigurable broadcast networks and asynchronous shared-memory systems are equivalent. In Pierre Ganty and Davide Bresolin, editors, *Proceedings 12th International Symposium on Games, Automata, Logics, and Formal Verification, GandALF 2021, Padua, Italy, 20-22 September 2021*, volume 346 of *EPTCS*, pages 18–34, 2021. doi:10.4204/EPTCS.346.2.
3. Nathalie Bertrand, Patricia Bouyer, and Anirban Majumdar. Concurrent parameterized games. In Arkadev Chattopadhyay and Paul Gastin, editors, *39th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2019, December 11-13, 2019, Bombay, India*, volume 150 of *LIPIcs*, pages 31:1–31:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. URL: https://doi.org/10.4230/LIPIcs.FSTTCS.2019.31, doi:10.4230/LIPICS.FSTTCS.2019.31.
4. Nathalie Bertrand, Miheer Dewaskar, Blaise Genest, Hugo Gimbert, and Adwait Amit Godbole. Controlling a population. *Log. Methods Comput. Sci.*, 15(3), 2019. doi:10.23638/LMCS-15(3:6)2019.
5. Nathalie Bertrand, Paulin Fournier, and Arnaud Sangnier. Distributed local strategies in broadcast networks. In Luca Aceto and David de Frutos-Escrig, editors, *26th International Conference on Concurrency Theory, CONCUR 2015, Madrid, Spain, September 1.4, 2015*, volume 42 of *LIPIcs*, pages 44–57. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2015. URL: https://doi.org/10.4230/LIPIcs.CONCUR.2015.44, doi:10.4230/LIPICS.CONCUR.2015.44.
6. Raven Beutner, Bernd Finkbeiner, and Jesko Hecking-Harbusch. Translating asynchronous games for distributed synthesis. In Wan J. Fokkink and Rob van Glabbeek, editors, *30th International Conference on Concurrency Theory, CONCUR 2019, August 27-30, 2019, Amsterdam, the Netherlands*, volume 140 of *LIPIcs*, pages 26:1–26:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. URL: https://doi.org/10.4230/LIPIcs.CONCUR.2019.26, doi:10.4230/LIPICS.CONCUR.2019.26.
7. Michael Blondin and François Ladouceur. Population protocols with unordered data. In Kousha Etessami, Uriel Feige, and Gabriele Puppis, editors, *50th International Colloquium on Automata, Languages, and Programming, ICALP 2023, July 10-14, 2023, Paderborn, Germany*, volume 261 of *LIPIcs*, pages 115:1–115:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. URL: https://doi.org/10.4230/LIPIcs.ICALP.2023.115, doi:10.4230/LIPICS.ICALP.2023.115.
8. Giorgio Delzanno, Arnaud Sangnier, and Riccardo Traverso. Parameterized verification of broadcast networks of register automata. In *Reachability Problems , RP 2013*, volume 8169 of *Lecture Notes in Computer Science*, pages 109–121. Springer, 2013. doi:10.1007/978-3-642-41036-9_11.
9. Giorgio Delzanno, Arnaud Sangnier, and Gianluigi Zavattaro. Parameterized verification of ad hoc networks. In *CONCUR 2010*, volume 6269 of *Lecture Notes in Computer Science*, pages 313–327. Springer, 2010. doi:10.1007/978-3-642-15375-4_22.
10. Javier Esparza, Pierre Ganty, Jérôme Leroux, and Rupak Majumdar. Verification of population protocols. *Acta Informatica*, 54(2):191–215, 2017. URL: https://doi.org/10.1007/s00236-016-0272-3, doi:10.1007/S00236-016-0272-3.
11. Nathanaël Fijalkow, Nathalie Bertrand, Patricia Bouyer-Decitre, Romain Brenguier, Arnaud Carayol, John Fearnley, Hugo Gimbert, Florian Horn, Rasmus Ibsen-Jensen, Nicolas Markey, Benjamin Monmege, Petr Novotný, Mickael Randour, Ocan Sankur, Sylvain Schmitz, Olivier Serre, and Mateusz Skomra. Games on graphs. *CoRR*, abs/2305.10546, 2023.

URL: https://doi.org/10.48550/arXiv.2305.10546, arXiv:2305.10546, doi:10.48550/ARXIV.2305.10546.

12 Hugo Gimbert. Distributed asynchronous games with causal memory are undecidable. *Log. Methods Comput. Sci.*, 18(3), 2022. URL: https://doi.org/10.46298/lmcs-18(3:30)2022, doi:10.46298/LMCS-18(3:30)2022.

13 Lucie Guillou, Corto Mascle, and Nicolas Waldburger. Parameterized broadcast networks with registers: from NP to the frontiers of decidability. In Naoki Kobayashi and James Worrell, editors, *Foundations of Software Science and Computation Structures - 27th International Conference, FoSSaCS 2024, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2024, Luxembourg City, Luxembourg, April 6-11, 2024, Proceedings, Part II*, volume 14575 of *Lecture Notes in Computer Science*, pages 250–270. Springer, 2024. doi:10.1007/978-3-031-57231-9_12.

14 Leonard H Haines. On free monoids partially ordered by embedding. *Journal of Combinatorial Theory*, 6(1):94–98, 1969.

15 Graham Higman. Ordering by divisibility in abstract algebras. *Proceedings of the London Mathematical Society*, s3-2(1):326–336, 1952. doi:10.1112/plms/s3-2.1.326.

16 Lukasz Kaminski and Slawomir Lasota. Bi-reachability in petri nets with data. In Rupak Majumdar and Alexandra Silva, editors, *35th International Conference on Concurrency Theory, CONCUR 2024, September 9-13, 2024, Calgary, Canada*, volume 311 of *LIPIcs*, pages 31:1–31:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024. URL: https://doi.org/10.4230/LIPIcs.CONCUR.2024.31, doi:10.4230/LIPICS.CONCUR.2024.31.

17 Eryk Kopczynski. Half-positional determinacy of infinite games. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *Automata, Languages and Programming, 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part II*, volume 4052 of *Lecture Notes in Computer Science*, pages 336–347. Springer, 2006. doi:10.1007/11787006_29.

18 Ranko Lazic, Thomas Christopher Newcomb, Joël Ouaknine, A. W. Roscoe, and James Worrell. Nets with tokens which carry data. *Fundamenta Informaticae*, 88(3):251–274, 2008.

19 Amir Pnueli and Roni Rosner. On the synthesis of a reactive module. In *Proc. ACM POPL*, pages 179–190, 1989.

20 Frank P Ramsey. On a problem of formal logic. In *Classic Papers in Combinatorics*, pages 1–24. Springer, 1987.

21 Sven Schewe and Bernd Finkbeiner. Synthesis of asynchronous systems. In Germán Puebla, editor, *Logic-Based Program Synthesis and Transformation, 16th International Symposium, LOPSTR 2006, Venice, Italy, July 12-14, 2006, Revised Selected Papers*, volume 4407 of *Lecture Notes in Computer Science*, pages 127–142. Springer, 2006. doi:10.1007/978-3-540-71410-1_10.

22 Sylvain Schmitz. Complexity hierarchies beyond elementary. *ACM Transactions on Computation Theory*, 8(1):3:1–3:36, 2016. doi:10.1145/2858784.

23 Sylvain Schmitz and Philippe Schnoebelen. Multiply-recursive upper bounds with higman's lemma. In *International Colloquium on Automata, Languages and Programming, ICALP 2011*, volume 6756 of *Lecture Notes in Computer Science*, pages 441–452. Springer, 2011. doi:10.1007/978-3-642-22012-8_35.

24 Daniel Stan. *Randomized strategies in concurrent games. (Stratégies randomisées dans les jeux concurrents)*. PhD thesis, University of Paris-Saclay, France, 2017. URL: https://tel.archives-ouvertes.fr/tel-01519354.

25 Larry J. Stockmeyer. Planar 3-colorability is polynomial complete. *SIGACT News*, 5(3):19–25, 1973. doi:10.1145/1008293.1008294.

26 Steffen van Bergerem, Roland Guttenberg, Sandra Kiefer, Corto Mascle, Nicolas Waldburger, and Chana Weil-Kennedy. Verification of population protocols with unordered data. In Karl Bringmann, Martin Grohe, Gabriele Puppis, and Ola Svensson, editors, *51st International Colloquium on Automata, Languages, and Programming, ICALP 2024, July 8-12, 2024,*

Tallinn, Estonia, volume 297 of *LIPIcs*, pages 156:1–156:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024. URL: https://doi.org/10.4230/LIPIcs.ICALP.2024.156, doi:10.4230/LIPICS.ICALP.2024.156.

27 Peter van Emde Boas. The convenience of tilings. In *Complexity, Logic, and Recursion Theory*, pages 331–363. CRC Press, 2019.

A Missing proof from Section 3

▶ **Lemma 12** (Invariants characterise winning control strategies). *A control strategy σ is winning if and only if there exists a sufficient invariant $I \subseteq \mathcal{M}$ for it.*

Proof. $\Rightarrow$ Suppose σ is winning. Let I be the set of messages such that there exists a σ-run in which they are broadcast. We show that I is a sufficient invariant for σ.

As σ is winning, m_{err} can never be broadcast, thus $m_{err} \notin I$. Suppose by contradiction that we have a σ-local run $s_0 \xrightarrow{\mathbf{op}_1(m_1)}_{\delta_1} s_1 \xrightarrow{\mathbf{op}_2(m_2)}_{\delta_2} \cdots \xrightarrow{\mathbf{op}_k(m_k)}_{\delta_k} s_k$ with $s_0 = s_{init}$ broadcasting some $m_{out} \notin I$ and only receiving messages of I. Then we can construct a σ-run in which m_{out} is broadcast.

We proceed by induction: for all $i \in [0, k]$, we show that there is a run ϱ_i whose projection on some agent a is $s_0 \xrightarrow{\mathbf{op}_1(m_1)}_{\delta_1} \cdots \xrightarrow{\mathbf{op}_i(m_i)}_{\delta_i} s_i$. For $i = 0$ this is immediate. Let $i > 0$, suppose we constructed ϱ_{i-1}, and let us construct ϱ_i. Let $\mathbb{A}_{i-1}$ be the set of agents of ϱ_{i-1}.

- If $s_{i-1} \xrightarrow{\mathbf{op}_i(m_i)}_{\delta_i} s_i$ is a broadcast step, then we simply execute ϱ_{i-1} and then make a apply that broadcast, which no other agent receives.
- If $s_{i-1} \xrightarrow{\mathbf{op}_i(m_i)}_{\delta_i} s_i$ is a reception step, in which a message type m is received, then we have $m \in I$, by construction of the σ-local run. Hence there exists a σ-run ϱ_m over a set of agents $\mathbb{A}_m$ in which m is broadcast. Up to renaming agents, we can assume that $\mathbb{A}_{i-1}$ and $\mathbb{A}_m$ are disjoint. We then define ϱ_i over $\mathbb{A}_{i-1} \sqcup \mathbb{A}_m$ by executing ϱ_{i-1} over $\mathbb{A}_{i-1}$, then executing ϱ_m over $\mathbb{A}_m$ up to the point before an agent a_m broadcasts m. Finally, we make a_m broadcast m and a receive it.

In both cases we obtain a σ-run in which the local run of a is $s_0 \xrightarrow{\mathbf{op}_1(m_1)}_{\delta_1} \cdots \xrightarrow{\mathbf{op}_i(m_i)}_{\delta_i} s_i$. In particular, for $i = k$, we get a σ-run in which m_{out} is broadcast. As $m_{out} \notin I$, this contradicts the definition of I. Hence I satisfies both items of the lemma.

$\Leftarrow$ Suppose there exists $I \subseteq \mathcal{M}$ satisfying the conditions of the lemma. Suppose by contradiction that there is a σ-run ϱ in which m_{err} is broadcast. Let m be the first message broadcast in ϱ that is not in I, and a the agent broadcasting it. Those are well-defined as $m_{err} \notin I$. Let ϱ' be the prefix of ϱ stopping right after that broadcast. The projection $\pi_a(\varrho')$ of ϱ' on a contains a broadcast of m but no reception of any $m' \notin I$, a contradiction. ◀

▶ **Lemma 34.** *The* SafeStrat *problem is* NP*-hard for BGR without registers.*

We reduce from the graph 3-colouring problem [25].

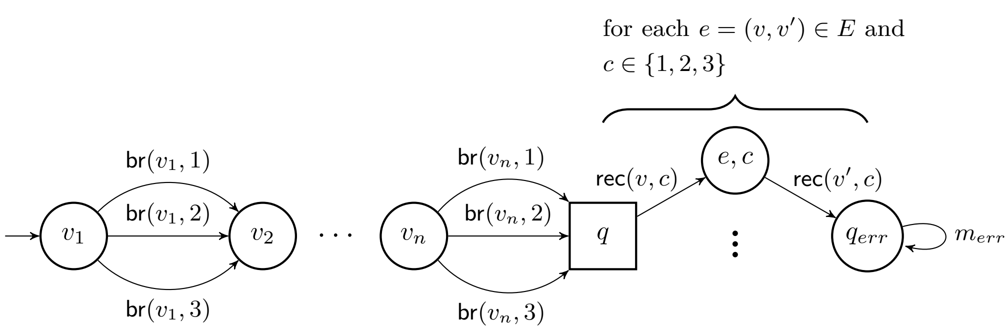

Figure 3 Illustration of the lower bound proof from Theorem 13.

Consider an undirected graph $G = (V, E)$ with $V = \{v_1, \dots, v_n\}$. We build a BGR with no registers as in Figure 3. From the initial state Controller chooses one of $(v_i, 1), (v_i, 2), (v_i, 3)$ for each $i \in [1, n]$ and broadcasts it. Then Environment picks an edge $e = (v, v') \in E$ and $c \in \{1, 2, 3\}$ and tries to reach q_{err} by receiving (v, c) and (v', c).

A strategy for Controller comes down to a colouring of V. It is winning if Environment can find an edge e and c such that both ends of e are coloured with c. In other words, Controller wins if and only if the selected colouring of V is a valid 3-colouring of G.

B Missing proofs from Section 4

We show that the invariants defined for signature BGR are accurate witnesses for winning control strategies. The idea behind this construction already existed in [13].

▶ **Lemma 18** (Invariants characterise winning strategies). *A control strategy σ is winning if and only if there exists a sufficient invariant $I \subseteq \mathcal{M}^*$ for it.*

Proof. $\Rightarrow$ Suppose σ is winning. Let I be the set of words such that there exists a σ-run, an agent a and a datum d such that w is a subword of the d-output of the projection of that run on a. The empty word is in I as it is the output of a local run of length 0, which is a σ-run. As σ is winning, m_{err} can never be broadcast, thus $m_{err} \notin I$. For the other condition, consider a σ-local run $u = (s_0, c_0) \xrightarrow{\mathbf{op}_1(m_1,d_1)}_{\delta_1} (s_1, c_1) \xrightarrow{\mathbf{op}_2(m_2,d_2)}_{\delta_2} \cdots \xrightarrow{\mathbf{op}_k(m_k,d_k)}_{\delta_k} (s_k, c_k)$ whose d-input is in I for every datum d.

Then we can construct a σ-run in which some agent has output $\mathbf{Out}_{sign}(u)$, thus proving that $\mathbf{Out}_{sign}(u) \in I$. This construction is illustrated in Figure 4.

Let D be the set of data appearing in u. For each datum $d \in D$, let w_d be the d-input of u. As $w_d \in I$, there exists a σ-run ϱ_d such that w_d is a subword of the output of an agent a_d. Let $\mathbb{A}_d$ be the set of agents of that σ-run.

Up to renaming data and agents, we can assume that the initial datum of a_d in ϱ_d is d, and that the σ-runs $(\varrho_d)_{d \in D}$ operate over disjoint sets of data and agents.

We take a fresh agent a. We construct a σ-run ϱ over $\{a\} \sqcup \bigsqcup_{d \in D} \mathbb{A}_d$ as follows. We make a follow the local run u. Whenever a needs to receive a message (m, d), we run ϱ_d over $\mathbb{A}_d$ until a message (m, d) is broadcast by a_d, and make a receive it. Then we continue running u. As $\mathbf{In}_d(u)$ is a subword of the output of $\pi_{a_d}(\varrho_d)$ for all $d \in D$, we eventually run u in full.

This yields a valid σ-run in which u is fully executed by a. Hence we have a σ-run in which agent a outputs $\mathbf{Out}_{sign}(u)$. By definition of I, we thus have $\mathbf{Out}_{sign}(u) \in I$.

$\Leftarrow$ Suppose there exists $I \subseteq \mathcal{M}^*$ satisfying the conditions of the lemma. Suppose by contradiction that there is a σ-run ϱ in which m_{err} is broadcast.

Let ϱ_- be the maximal prefix of ϱ such that the output of each agent is in I. It is well-defined as $\varepsilon \in I$, thus the prefix of ϱ with no step satisfies that condition. As $m_{err} \notin I$ and I is downward-closed, I does not contain any word containing m_{err}. Hence the output of ϱ is not in I, and thus ϱ_- is a strict prefix of ϱ.

Let a be the agent making the broadcast of the step right after ϱ_- in ϱ, and let m be the message it broadcasts. Let ϱ_+ be the prefix of ϱ made of ϱ_- and that extra step.

Let w_- be the output of a in ϱ_-. For all $d \in \mathbb{D}$, the d-input of a in ϱ_- must be a subword of the output of another agent. By definition of ϱ_-, the d-input of a in ϱ_- is thus in I for all d. As the d-input of a in ϱ_- and ϱ_+ is the same, the d-input of a in ϱ_+ is in I for all d. By maximality of ϱ_-, the output of a in ϱ_+ is not in I.

This contradicts the second condition on I given by the lemma. ◀

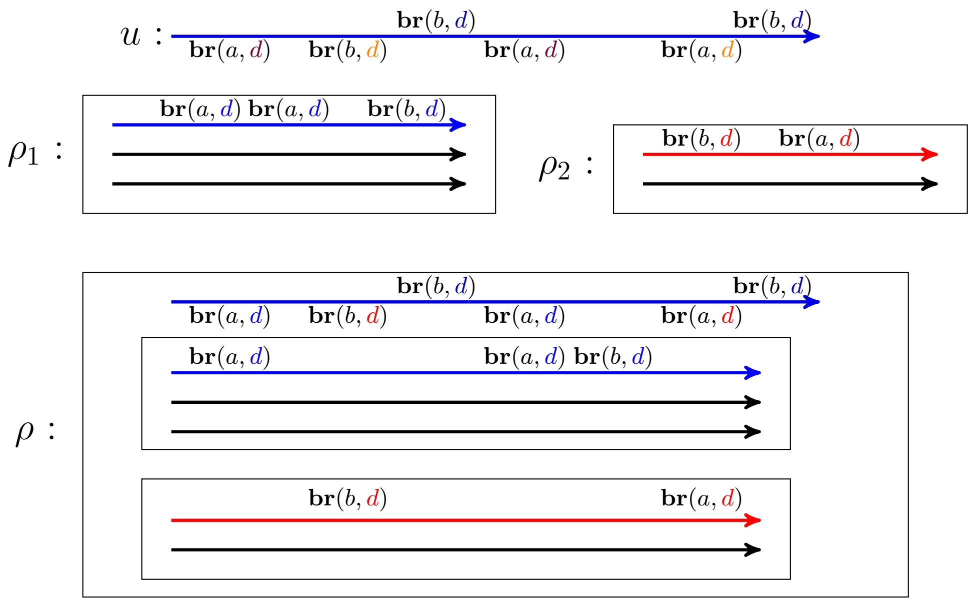

Figure 4 Illustration of the proof of Lemma 18. Most information is omitted, we only represent schematically the relevant broadcasts and receptions. Data are represented by colours. If we have a local run u outputting bb and for each d a run in which an agent outputs the d-input of u, then we can rename some data and compose those runs to form a run in which an agent outputs bb. Local runs of relevant agents are coloured with their initial datum.

▶ **Lemma 20.** *Let $I \subseteq \mathcal{M}^*$ be a downward-closed set of words containing ε and not m_{err}. If Controller wins the invariant game $\mathcal{IG}(\mathcal{G}, I)$ then there is a control strategy σ such that I is a sufficient invariant for σ.*

Proof. Let $\sigma_{\mathcal{IG}}$ be a winning strategy for Controller in $\mathcal{IG}(\mathcal{G}, I)$. We define σ as the control strategy in which Controller follows $\sigma_{\mathcal{IG}}$. That is, given a local run $u = (s_0, c_0) \xrightarrow{\mathbf{op}_1(m_1,d_1)}_{\delta_1} \cdots \xrightarrow{\mathbf{op}_k(m_k,d_k)}_{\delta_k} (s_k, c_k)$, we set $\sigma(u) = \sigma_{\mathcal{IG}}(\delta_1 \cdots \delta_k)$.

We show that I is a sufficient invariant for σ. Assume by contradiction that we have a σ-local run $u = (s_0, c_0) \xrightarrow{\mathbf{op}_1(m_1,d_1)}_{\delta_1} \cdots \xrightarrow{\mathbf{op}_k(m_k,d_k)}_{\delta_k} (s_k, c_k)$ such that u has an output outside of I, and its d-input is in I for all $d \in \mathbb{D}$. We then show that $\pi = \delta_1 \cdots \delta_k$ is a losing $\sigma_{\mathcal{IG}}$-play for Controller in $\mathcal{IG}(\mathcal{G}, I)$. As u is a σ-local run, by definition of σ, $\delta_1 \cdots \delta_k$ is a $\sigma_{\mathcal{IG}}$-play.

For all $j \in [0, k]$ let u_j be the prefix of u up to (s_j, c_j) and $\pi_j = \delta_1 \cdots \delta_j$.

▷ Claim 35. For all $j \in [0, k]$ and $i \in [2, r]$ we have $\mathsf{recentIn}_i(\pi_j) \sqsubseteq \mathbf{In}_{u_j}(c_j(i))$ and $\mathsf{Out}(\pi_j) = \mathbf{Out}_{sign}(u_j)$.

Proof. By a straightforward induction on j. ◁

We can instantiate the previous claim with $j = k$ to obtain $\mathsf{Out}(\pi) = \mathbf{Out}_{sign}(u)$. As we assumed that $\mathbf{Out}_{sign}(u) \notin I$, we have $\mathsf{Out}(\pi) \notin I$. As the d-input of u is in I for all $d \in \mathbb{D}$, and I is downward-closed, the letters of all messages received in u are in I. Moreover, by the previous claim, for all $j \in [0, k]$, we have $\mathsf{recentIn}_i(\pi_j) \sqsubseteq \mathbf{In}_{u_j}(c_j(i)) \sqsubseteq \mathbf{In}_u(c_j(i)) \in I$. As I is downward-closed, we have $\mathsf{recentIn}_i(\pi_j) \in I$ for all i, j.

As a result, π is a losing $\sigma_{\mathcal{IG}}$-play for Controller. This contradicts the assumption that $\sigma_{\mathcal{IG}}$ is a winning strategy for $\mathcal{IG}(\mathcal{G}, I)$. In consequence, there is no σ-local run u whose output is outside of I, and whose d-input is in I for all $d \in \mathbb{D}$.

This means that I is a sufficient invariant for σ. ◀

▶ **Lemma 23** (Bounding the size of the invariant). *Let $\mathcal{G}$ a signature BGR. There is a winning control strategy for $\mathcal{G}$ if and only if there is a sequence of words $w_0, \ldots, w_k \in \mathcal{M}^*$ such that*

- *Controller wins $\mathcal{IG}(\mathcal{G}, \{w_1, \ldots, w_k\}\!\uparrow^{\mathsf{c}})$,*
- *and for all $i \in [1, k]$, $|w_i| \leq \psi(|w_{i-1}|)$.*

Proof. Suppose there is a winning control strategy σ. By Lemma 18 there is a downward-closed sufficient invariant $I \subseteq \mathcal{M}^*$ for σ. By Lemma 21, Controller wins $\mathcal{IG}(\mathcal{G}, I)$, so the first condition is satisfied.

For the second condition, as I^{c} is upward-closed it has a finite basis B. Let $w_0, w_1, \ldots, w_k$ be the elements of B *sorted by length*. We can assume that we took I so that k is minimal. For all $j \in [1, k]$, we define $B_j = \{w_i \mid i < j\}$ and $I_j = B_j\!\uparrow^{\mathsf{c}}$. Note that we have $I \subseteq I_k \subseteq \ldots \subseteq I_0$. As I contains ε and not m_{err}, we can assume $w_0 = m_{err}$. By minimality of k, for all $j \in [1, k]$ the set I_j is not a sufficient invariant for σ.

By Lemma 21, there is a σ-local run of length at most $\varphi(\mathcal{R}, B_j)$ whose output is not in I_j and whose d-inputs are all in I_j. As I is a sufficient invariant for σ, one of those d-inputs must not be in I. We choose one of those and call it w. As a consequence, there exists w_ℓ with $\ell \geq j$ such that $w_\ell \sqsubseteq w$, and thus $|w_\ell| \leq |w| \leq \varphi(\mathcal{R}, B_j)$. As $|w_i| \leq |w_{i+1}|$ for all i, this implies $|w_j| \leq \varphi(\mathcal{R}, B_j) = |\mathcal{R}|(||B_j|| + 1)^{(|B_j|+1)}$. As w_{j-1} is of maximal length among words of B_j, we have $||B_j|| = |w_{j-1}|$ and $|B_j| \leq |\mathcal{M}|^{|w_{j-1}|+1}$.

As a result, $|w_j| \leq |\mathcal{R}|(|w_{j-1}| + 1)^{|\mathcal{M}|^{|w_{j-1}|+1}+1} = \psi(|w_{j-1}|)$. Thus the second condition of the lemma is also satisfied.

The other direction follows by Lemma 20 and Lemma 18. ◀

C Missing proofs from Section 5

C.1 Characterisation of winning strategies (Section 5)

▶ **Lemma 29** (Invariants characterise winning strategies). *A control strategy σ is winning if and only if there exists a sufficient invariant $(I, (J_m)_{m \in \mathcal{M}})$ for it.*

This section is dedicated to the proof of this lemma. To do so, we need an argument that resembles the construction illustrated in Figure 4. However, the construction gets more involved in this case.

We can start by proving the easier direction of the equivalence, given by the following lemma. The general structure of this construction already existed in [13]. However, the different nature of the objects used here and there make it difficult to use their proof as a black box. We have to go through all the steps here.

▶ **Lemma 36.** *If there exists a sufficient invariant $(I, (J_m)_{m \in \mathcal{M}})$ for a control strategy σ then σ is winning.*

Proof. Suppose σ has a sufficient invariant $(I, (J_m)_{m \in \mathcal{M}})$. Suppose by contradiction that there is a σ-run ϱ in which m_{err} is broadcast.

Let a be an agent broadcasting m_{err} in ϱ, let u be its local run.

We first show that the local run of a in ϱ does not satisfy 3a and 3b.

If m_{err} is broadcast in u with its initial datum then, as $m_{err} \notin I$ and I is downward-closed, we cannot have $\mathbf{Out}_d(u) \in I$. On the other hand, if m_{err} is broadcast in u with another datum d' then as $J_{m_{err}} = \emptyset$, $\mathbf{In}_{d'}(u) \notin J_{m_{err}}$. Hence u does not satisfy 3a and 3b.

Let ϱ_- be the maximal prefix of ϱ such that the local runs of all agents satisfy 3a and 3b. It is well-defined: we saw that the full run ϱ does not satisfy this requirement, and as $\varepsilon \in I$, the prefix of ϱ with no step satisfies it. Furthermore ϱ_- must be a strict prefix of ϱ.

Let a be the agent making the broadcast of the step right after ϱ_- in ϱ, and let m be the message it broadcasts. Let ϱ_+ be the prefix of ϱ made of ϱ_- and that extra step.

By maximality of ϱ_-, there must be an agent whose local run in ϱ_+ does not satisfy 3a and 3b. This agent can only be a: all agents satisfied both conditions in ϱ_-, an agent cannot switch from satisfying to not satisfying those conditions without making a broadcast (for 3b, this is due to the fact that all J_m are upward-closed), and a is the only one who made a broadcast in the last step.

As a consequence, the local run u_+ of a in ϱ_+ must dissatisfy either 3a or 3b. It remains to show that u_+ satisfies both 3i and 3ii to obtain a contradiction.

We start by showing that the local run u_- of a in ϱ_- satisfies 3i and 3ii.

- Let d be the initial datum of u_-. Let $m_1, \dots, m_k$ be the letters such that (m_i, d) is broadcast by an agent that is not a during ϱ_-. Let us cut ϱ_- into sections $\varrho_0 \cdots \varrho_k$ such that $\varrho_0 \cdots \varrho_i$ is the maximal prefix of ϱ_- in which (m_i, d) has not been broadcast by any agent apart from a. For each i let u_i be the projection of ϱ_i on a. We thus have $u_- = u_0 \cdots u_k$. Let a_{m_i} be the first agent different from a who broadcasts (m_i, d) and let u_{m_i} be the projection of ϱ_- on a_{m_i}. Let w_i be the sequence of letters broadcast in ϱ_i with datum d.

  Consider the decomposition $\mathsf{dec} = (v_0, m_1, \dots, v_{k-1}, m_k, v_k)$ where $v_i = \mathbf{Out}_d(u_i)$. By definition u_- must be compatible with it. Let $i \in [1, k]$. As u_{m_i} satisfies 3b, we have $\mathbf{In}_d(u_{m_i}) \in J_{m_i}$. By definition, we must have $\mathbf{In}_d(u_{m_i}) \sqsubseteq w_0 \cdots w_{i-1}$ and thus $w_0 \cdots w_{i-1} \in J_{m_i}$ as J_{m_i} is upward-closed. Furthermore, each w_j (the letters sent in ϱ_j with datum d) can be obtained from v_j (the ones sent by a) by adding letters of $\{m_1, \dots, m_j\}$ (the broadcasts of other agents). As a result, we have $w_0 \cdots w_{i-1} \in \mathcal{L}_{(v_0, m_1, \dots, v_{i-1})}$. We obtain that $w_0 \cdots w_{i-1} \in J_{m_i} \cap \mathcal{L}_{(v_0, m_1, \dots, v_{i-1})}$, thus $J_{m_i} \cap \mathcal{L}_{(v_0, m_1, \dots, v_{i-1})}$ is not empty. In conclusion, $\mathsf{dec} \in \mathcal{D}((J_m)_{m \in \mathcal{M}})$.
- We now show that u_- satisfies 3ii.

  Let $d' \neq d$. If d' does not appear in ϱ_- then $\mathbf{In}_{d'}(u_-) = \varepsilon \in \mathcal{L}(I, (J_m)_{m \in \mathcal{M}}))$. Otherwise, let a' be the agent whose initial datum in ϱ is d'. We set w' the sequence of letters broadcast with datum d' in ϱ_-. Clearly $\mathbf{In}_{d'}(u_-) \sqsubseteq w'$. In order to show that $\mathbf{In}_{d'}(u_-)$, it suffices to show that $w' \in \mathcal{L}(I, (J_m)_{m \in \mathcal{M}}))$.

  We use the same arguments as for the previous item: we cut ϱ_- into sections $\varrho'_0 \cdots \varrho'_k$ according to the times at which new letters are broadcast with d' by agents other than a. We then construct a decomposition $\mathsf{dec}' = (v'_0, m'_1, \dots, v'_{k-1}, m'_k, v'_k)$ where m'_i is the message broadcast with d' at the start of ϱ'_i and v'_i is the sequence of letters broadcast by a' in ϱ'_i.

  We argue as before that $w' \in \mathcal{L}_{\mathsf{dec}'}$ and $\mathsf{dec}' \in \mathcal{D}(I, (J_m)_{m \in \mathcal{M}})$.

We have shown that u_- satisfied 3i and 3ii. To obtain a contradiction, we must show that u_+ satisfies them as well. By definition, u_+ is u_- with an additional broadcast at the end.

- Let $\mathsf{dec} = (v_0, m_0, \dots, v_k) \in \mathcal{D}((J_m)_{m \in \mathcal{M}})$ be a decomposition such that u_- is compatible with dec. We have $u_- = u_0 \cdots u_k$ where $v_i \sqsubseteq \mathbf{Out}_d(u_i)$ and $\mathbf{In}_d(u_i) \in \{m_1, \dots, m_i\}^*$ for all i. Let u_k^+ be u_k to which we append the last broadcast in u_+. We obtain

$u_- = u_0 \cdots u_{k-1} u_k^+$. Since $v_k \sqsubseteq \mathbf{Out}_d(u_k) \sqsubseteq \mathbf{Out}_d(u_k^+)$ and $\mathbf{In}_d(u_k) = \mathbf{In}_d(u_k^+) \in \{m_1, \dots, m_k\}^*$, we conclude that u_+ is compatible with dec. Hence u_+ satisfies 3i.

- As $\mathbf{In}_{d'}(u_-) = \mathbf{In}_{d'}(u_+)$ for all $d' \in \mathbb{D}$, u_+ satisfies 3ii.

In conclusion, we have constructed a σ-local run u_+ such that u_+ satisfies 3i and 3ii but not 3a and 3b, yielding a contradiction.

◀

We must now prove the other implication of Lemma 29. Intuitively, the argument goes as follows.

We define a notion of partial run. This is a run of a set of agents, but some messages can be received without being broadcast. They are called unmatched receptions. A local run is a particular case of partial run, with a single agent.

We assume that σ is winning. We take I as the downward-closure of the set of words $w \in \mathcal{M}^*$ such that there is a σ-run ϱ in which the sequence of messages w is broadcast, all with the same datum. For each m, we set J_m to be the upward-closure of the set of words $w = m_1 \cdots m_n$ such that there is a σ-partial run in which the sequence of unmatched receptions is of the form $(m_1, d) \cdots (m_n, d)$ for some $d \in \mathbb{D}$, and (m, d) is broadcast at some point. This should be understood as follows: if we have a run in which $(m_1, d) \cdots (m_n, d)$ is broadcast, then we can compose it with the partial run above, match all the unmatched receptions and obtain an extra broadcast of m.

The difficulty is to show that those sets form a sufficient invariant for σ. In particular, we need to take a σ-local run u satisfying 3i and 3ii and show that it satisfies 3a and 3b. We do that by building σ-runs in which the local run of some agent is u.

We rely on several technical lemmas. Lemma 38, 39 and 40. Their statements are involved but they come with illustrations that should give helpful intuition. Before reading the details of those lemma we recommend that the reader reads the proof of Theorem 29 at the end of this section, to better understand how those lemmas are used.

C.1.1 Definitions for partial runs

For the following proof we need to introduce the notion of *partial run*, which describes the projection of a run on a subset of agents. We then show a key technical lemma that allows us to construct a run from a local run and a set of suitable partial runs.

We will use this lemma to prove a characterisation of winning control strategies using some invariants, like in the previous sections.

▶ **Definition 37.** *Let γ, γ' two configurations.*

A partial step $\gamma \to_p \gamma'$ is defined if either $\gamma \to \gamma'$ (normal step) or there exist $m \in \mathcal{M}$, $d \in \mathbb{D}$ such that for all agent a either $\gamma(a) = \gamma'(a)$ or $\gamma(a) \xrightarrow{rec(m,d)}_\delta \gamma'(a)$ for some reception transition δ (unmatched reception of (m, d)).

A partial run ϱ is a sequence of partial steps. It is initial if it starts in an initial configuration. Its d-input $\mathbf{In}_d(\varrho)$ is the sequence $m_0 \cdots m_k$ of letters corresponding to unmatched receptions with datum d in ϱ. Its d-output $\mathbf{Out}_d(\varrho)$ is the sequence of letters corresponding to broadcasts with datum d in ϱ.

Note that a local run can be seen as a partial run with a single agent. Given a control strategy σ, a *σ-partial run* is a partial run in which the local runs of all agents are σ-local runs.

A datum d is *initial* in ϱ if it appears in the first configuration. We extend the notion of compatible to partial runs: A partial run ϱ is *compatible* over d with a decomposition $\mathsf{dec} = (v_0, m_1, \ldots, v_k)$ if $\varrho = \varrho_0 \cdots \varrho_k$ and for all $i \in [0, k]$, $v_i \sqsubseteq \mathbf{Out}_d(\varrho_i)$ and $\mathbf{In}_d(\varrho_i) \in \{m_1, \ldots, m_i\}^*$, with d an initial datum of some agent in ϱ.

The following lemmas give us ways to compose partial runs to obtain complete runs.

Suppose we have a partial run ϱ compatible with a decomposition $\mathsf{dec} = (v_0, m_1, \ldots, v_k)$ over an initial datum d.

Suppose that we have, for each non-initial datum d', a run $\varrho_{d'}$ such that $\mathbf{In}_{d'}(\varrho) \sqsubseteq \mathbf{Out}_{d'}(\varrho_{d'})$.

Also suppose that for each $j \in [1, k]$ we have a partial run ϱ_j such that $\mathbf{In}_d(\varrho'_j) \in \mathcal{L}_{(v_0, m_1, \ldots, v_{i-1})}$ and which contains a broadcast of (m, d), and no unmatched receptions on data other than d.

- First, we show that given a word $w \in \mathcal{L}_{\mathsf{dec}}$ we can use the ϱ_i to extend ϱ and obtain a σ-partial run which is still compatible with dec and whose d-output contains w as a subword. This is done by composing ϱ with many copies of each ϱ_i to fill in the missing broadcasts.
- Then, we show that we can again use many copies of the ϱ_i to eliminate the unmatched receptions with datum d. We do this by carefully adding the necessary copies of ϱ_i, by decreasing i. Each time we fill in a missing broadcast of m_i while possibly adding new ones for some of the m_j with $j < i$. This terminates as the number of unmatched receptions of each letter m_i decreases with respect to the lexicographic ordering.
- We show that for each non-initial d' we can eliminate the unmatched receptions with datum d' by composing that partial run with the σ-runs $\varrho_{d'}$. We use the broadcasts in $\varrho_{d'}$ to match the unmatched receptions in ϱ over d'.
- Finally, we combine the two first steps to show that given a run compatible with a decomposition dec over some datum d and a word $w \in \mathcal{L}_{\mathsf{dec}}$, we can extend this run to obtain another run whose d-output contains w.

C.1.2 Extending the output

▶ **Lemma 38.** *Let* $\mathsf{dec} = (v_0, m_1, \ldots, v_k)$ *be a decomposition, let* $w \in \mathcal{L}_{\mathsf{dec}}$.

Let d a datum and ϱ an initial σ-partial run compatible with dec *over d.*

Suppose that for all $j \in [1, k]$ there exist an initial σ-partial run ϱ'_j such that $\mathbf{In}_d(\varrho'_j) \in \mathcal{L}_{\mathsf{dec}_j}$ *where* $\mathsf{dec}_j = (v_0, m_1, \ldots, v_{j-1})$, $\mathbf{In}_{d'}(\varrho'_j) = \varepsilon$ *for all* $d' \neq d$ *and* $m_j \sqsubseteq \mathbf{Out}_d(\varrho'_j)$.

Then, there is a partial run $\tilde{\varrho}$ such that

- $\tilde{\varrho}$ *is compatible with* dec *over d,*
- $w \sqsubseteq \mathbf{Out}_d(\tilde{\varrho})$
- *for all $d' \neq d$, either* $\mathbf{In}_{d'}(\tilde{\varrho}) = \varepsilon$ *or* $\mathbf{In}_{d'}(\tilde{\varrho}) = \mathbf{In}_{d'}(\varrho)$

Proof. As $w \in \mathcal{L}_{\mathsf{dec}}$, we have $w = w_0 \cdots w_k$, where each w_i can be obtained by adding some letters of $\{m_1, \ldots, m_i\}$ to v_i. As u is compatible with dec, $u = u_0 \cdots u_k$ with $v_i \sqsubseteq \mathbf{Out}_d(u_i)$ for all i and $\mathbf{In}_d(u_i) \in \{m_1, \ldots, m_i\}^*$. As a consequence, to obtain a d-output that contains w, it suffices to show that we can add a letter from $\{m_1, \ldots, m_i\}$ at any point of u_i. We do so using $\tilde{\varrho}_i$: Since $\mathbf{In}_d(\tilde{\varrho}_i) \in \mathcal{L}_{(v_0, m_1, \ldots, v_{i-1})}{\downarrow}$, we can split $\tilde{\varrho}_j$ into $\tilde{\varrho}_{j,0}, \ldots, \tilde{\varrho}_{j,j-1}$ so that $\mathbf{In}_d(\tilde{\varrho}_{j,i}) \sqsubseteq \tilde{w}_{j,i}$ where $\tilde{w}_{j,i}$ can be obtained by adding letters from $\{m_1, \ldots, m_j\}$ to v_i.

We use the following composition operation: consider ϱ and one of the ϱ'_j. We can build a new run in which we execute both runs in parallel over disjoint sets of agents. We match each $\tilde{\varrho}_{i,j}$ with ϱ_j so that the broadcasts of ϱ_j with d forming v_i are received in $\tilde{\varrho}_{i,j}$ and the only remaining missing broadcasts in that section of the run are with letters $m_1, \ldots, m_i$.

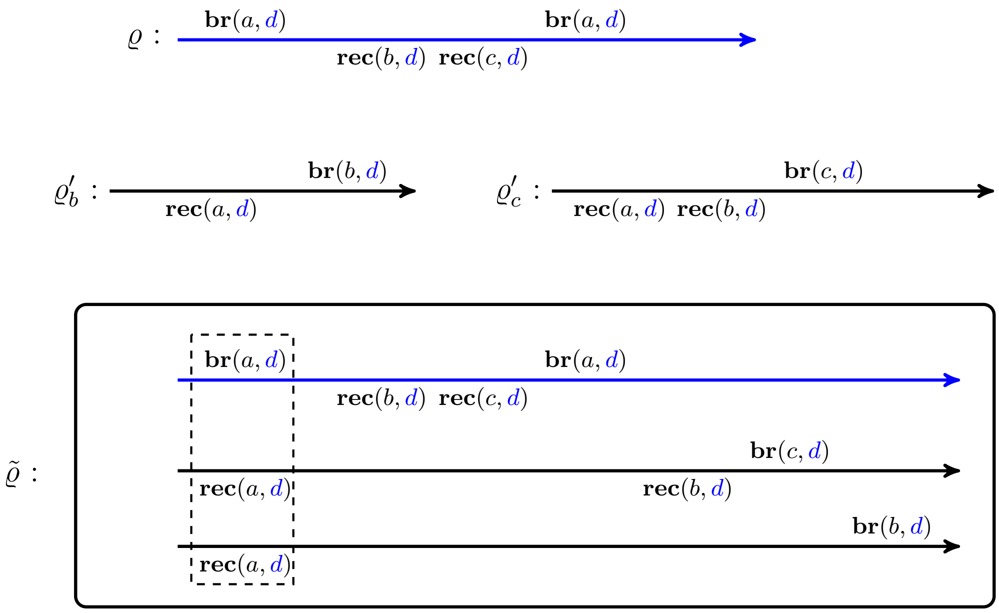

Figure 5 An illustration of the proof of Lemma 38. The partial run ϱ is compatible with decomposition $(a, b, a, c, \varepsilon)$. We have $\mathbf{In}_d(\varrho_b) = a \in \mathcal{L}_{(a)}$ and $\mathbf{In}_d(\varrho_b) = ab \in \mathcal{L}_{(a,b,a)}$. We build a partial run $\tilde{\varrho}$ such that $aacb \sqsubseteq \mathbf{Out}_d(\tilde{\varrho})$. Note that $\tilde{\varrho}$ is also compatible with decomposition $(a, b, a, c, \varepsilon)$. We ignore data other than d in this picture.

We obtain a run section whose d-output still contains v_i and whose d-input only contains $m_1, \dots, m_i$. This lets us get to a point where the next step in $\tilde{\varrho}_j$ is a broadcast of (m_j, d) and ϱ has been executed up to the beginning of ϱ_j. We may then use the (m_j, d) broadcast at any moment in the rest of ϱ to extend the d-output. As a consequence, we can compose ϱ with the ϱ'_i as many times as necessary to obtain a run $\tilde{\varrho}$ whose d-output contains w.

Each composition maintains the fact that the run is compatible with dec. Further, for all $d' \neq d$, either d' does not appear in ϱ and $\mathbf{In}_d(\tilde{\varrho}) = \varepsilon$ or d' appears in ϱ and then $\mathbf{In}_{d'}(\tilde{\varrho}) = \mathbf{In}_{d'}(\varrho)$. ◀

C.1.3 Unmatched receptions with initial data

▶ **Lemma 39.** *Let* $\mathsf{dec} = (v_0, m_1, \dots, v_k)$ *be a decomposition,* d *a datum,* ϱ *an initial* σ*-partial run compatible with* dec *over* d.

Suppose that for all $j \in [1, k]$ *there exist an initial partial run* ϱ'_j *in which* d *is not initial such that* $\mathbf{In}_{\varrho'_j}(d) \in \mathcal{L}_{\mathsf{dec}_j}$ *where* $\mathsf{dec}_j = (v_0, m_1, \dots, v_{j-1})$*,* $\mathbf{In}_{\varrho'_j}(d') = \varepsilon$ *for all* $d' \neq d$ *and* $m_j \sqsubseteq \mathbf{Out}_d(\varrho'_j)$.

Then, there exist a σ*-partial run* $\tilde{\varrho}$ *such that*

- $\mathbf{In}_d(\tilde{\varrho}) = \varepsilon$,
- $\mathbf{Out}_d(\varrho) \sqsubseteq \mathbf{Out}_d(\tilde{\varrho})$,
- *for all* $d' \neq d$, $\mathbf{In}_{d'}(\tilde{\varrho}) = \mathbf{In}_{d'}(\varrho)$

Proof. We proceed in the same way as in the previous part: the goal is now to use the partial runs ϱ'_j to eliminate the d-input of ϱ.

As ϱ is compatible with dec over d, we can split ϱ into $\varrho_0, \dots, \varrho_k$ with $w_i \sqsubseteq \mathbf{Out}_d(\varrho_i)$ and $\mathbf{In}_d(\varrho_i) \in \{m_1, \dots, m_i\}^*$ for all i. Again, we rename agents and data so that the sets of

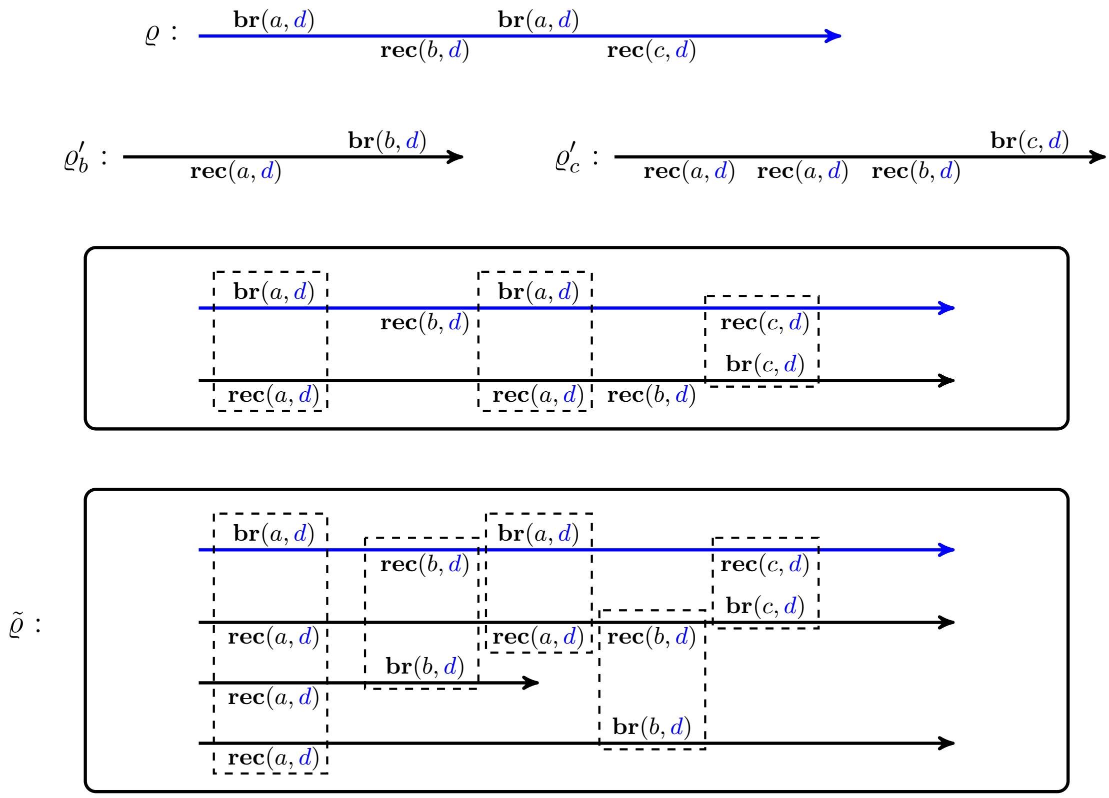

Figure 6 An illustration of the proof of Lemma 39. The partial run ϱ is compatible with decomposition $(a, b, a, c, \varepsilon)$. We have $\mathbf{In}_d(\varrho_b) = a \in \mathcal{L}_{(a)}$ and $\mathbf{In}_d(\varrho_b) = aab \in \mathcal{L}_{(a,b,a)}$. We build $\tilde{\varrho}$ such that $\mathbf{In}_d(\tilde{\varrho}) = \varepsilon$. We start by using ϱ'_c to eliminate the unmatched receptions of c (while adding some unmatched receptions of b), then we use ϱ'_b to eliminate the unmatched receptions of b. We ignore data other than d in this picture.

agents of ϱ and of every ϱ'_j are all disjoint and the only shared datum between any two of these runs is d.

We once again use the composition operation described in the proof of Lemma 38: consider ϱ and one of the ϱ'_j. We execute both runs in parallel and match each $\tilde{\varrho}_{i,j}$ with ϱ_j so that the broadcasts of ϱ_j with d forming v_i are received in $\tilde{\varrho}_{i,j}$, leaving only unmatched receptions with letters $m_1, \ldots, m_i$. We obtain a run section whose d-output still contains v_i and whose d-input only contains $m_1, \ldots, m_i$. We can do that until the next step in $\tilde{\varrho}_j$ is a broadcast of (m_j, d) and ϱ has been executed up to the beginning of ϱ_j. We may then use the (m_j, d) broadcast at any moment in the rest of ϱ to match an unmatched reception of ϱ. As a consequence, we can compose ϱ with the ϱ'_i as many times as necessary to obtain a run $\tilde{\varrho}$ with no unmatched receptions on d.

Each composition maintains the fact that the run is compatible with dec. When we do a composition with ϱ'_j to match a reception of (m_j, d), we may add some receptions of $m_1, \ldots, m_{j-1}$ to the run (the ones of ϱ'_j). However, every composition decreases the number of unmatched receptions of $m_k, \ldots, m_1$ for the lexicographic ordering.

As a result, in the end we obtain a run $\tilde{\varrho}$ without any unmatched reception on datum d. As ϱ is fully contained in $\tilde{\varrho}$, $\mathbf{Out}_d(\varrho) \sqsubseteq \mathbf{Out}_d(\tilde{\varrho})$. Moreover, for all $d' \neq d$, either d' does not appear in ϱ and then $\mathbf{In}_{d'}(\tilde{\varrho}) = \varepsilon$ or d' appears in ϱ and $\mathbf{In}_{d'}(\tilde{\varrho}) = \mathbf{In}_{d'}(\varrho)$ ◀

C.1.4 Unmatched receptions with non-initial data

▶ **Lemma 40.** *Let ϱ be an initial σ-partial run, d' a datum, ϱ' an initial σ-run. If $\mathbf{In}_d(\varrho) \sqsubseteq \mathbf{Out}_{d'}(\varrho')$ and d' an initial datum value in ϱ' but not in ϱ, then there exists an initial σ-partial run $\tilde{\varrho}$ such that*

- $\mathbf{In}_{d'}(\tilde{\varrho}) = \varepsilon$
- *for all $d'' \neq d'$, $\mathbf{In}_{d''}(\tilde{\varrho}) = \mathbf{In}_{d''}(\varrho)$*
- *for all $d'' \neq d'$, $\mathbf{Out}_{d''}(\varrho) \sqsubseteq \mathbf{Out}_{d''}(\tilde{\varrho})$*

Proof. Up to renaming agents, assume that ϱ and ϱ' have disjoint agents. We rename data in ϱ' so that ϱ' has no shared data with ϱ besides d'.

We build $\tilde{\varrho}$ by running ϱ and ϱ' over their respective agents separately. We use the broadcasts made by ϱ' with d' to match the unmatched receptions with datum d' in ϱ: this gives us a new partial run ϱ with no unmatched reception with datum d'. Furthermore, for every datum d'', either the sequence of broadcasts and unmatched receptions is the same as before, or $\mathbf{In}_d(\varrho) = \varepsilon$ (if d'' appears in $\varrho_{d'}$).

The d''-output can only increase as ϱ is fully executed within $\tilde{\varrho}$. ◀

C.1.5 How to obtain a word $w \in \mathcal{L}(I, (J_m)_{m\in\mathcal{M}})$

We now combine Lemmas 38 and 39 to obtain one last useful technical lemma for the proof of Lemma 29. It will be used to prove the second condition of Definition 28 when showing that an invariant $(I, (J_m)_{m\in\mathcal{M}})$ is sufficient for a strategy σ.

▶ **Lemma 41.** *Let σ be a control strategy, I a downward-closed set of words, and $(J_m)_{m\in\mathcal{M}}$ upward-closed ones.*

Suppose that for all $w \in I$ there is an initial σ-run and a datum d such that $w \sqsubseteq \mathbf{Out}_d(\varrho)$. Suppose also that for all $m \in \mathcal{M}$ and $w \in J_m$ there is a σ-partial run ϱ and a datum d that is not initial in ϱ such that $\mathbf{In}_d(\varrho) \sqsubseteq w$, $m \sqsubseteq \mathbf{Out}_d(\varrho)$ and $\mathbf{In}_{d'}(\varrho) = \varepsilon$ for all $d' \neq d$.

Then for all $w \in \mathcal{L}(I, (J_m)_{m\in\mathcal{M}})$, there is a σ-run ϱ and a datum d such that $w \sqsubseteq \mathbf{Out}_d(\varrho)$.

Proof. Let $w \in \mathcal{L}(I, (J_m)_{m\in\mathcal{M}})$, w matches a decomposition $\mathsf{dec} = (v_0, m_1, \dots, v_k)$ such that $v_0 \cdots v_k \in I$ and, for all j, $\mathcal{L}_{(v_0,m_1,\dots,v_{j-1})} \cap J_{m_j} \neq \emptyset$. Hence we have a σ-run ϱ and a datum d such that $v_0 \cdots v_k \sqsubseteq \mathbf{Out}_d(\varrho)$. Note that as ϱ has no unmatched reception, in particular, it is compatible with dec. Also, for all j we have a σ-partial run ϱ_j and a datum d_j not initial in ϱ_j such that $\mathbf{In}_{d_j}(\varrho_j) \in \mathcal{L}_{(v_0,m_1,\dots,v_{j-1})}\downarrow$, $m_j \sqsubseteq \mathbf{Out}_{d_j}(\varrho_j)$ and $\mathbf{In}_{d'}(\varrho_j) = \varepsilon$ for all $d' \neq d_j$.

By Lemma 38, this means that we can obtain a σ-partial run whose d-output contains w, with no unmatched receptions on data other than d, and compatible with dec.

We can then use Lemma 39 to eliminate all unmatched receptions and obtain a σ-run whose d-output contains w. ◀

C.1.6 Proof of the characterisation lemma

Proof of Lemma 29. $\Rightarrow$ Suppose σ is winning. Consider R the set of σ-runs.

Let $I = \{\mathbf{Out}_d(\varrho) \mid \varrho \in R, d \in \mathbb{D}\}\downarrow$ be the set of all outputs of all σ-runs.

For all $m \in \mathcal{M}$, we set J_m as the upward-closure of the set of $\mathbf{In}_d(\varrho)$ with ϱ a σ-partial run such that d is not an initial datum of ϱ, ϱ contains a broadcast of (m, d) and $\mathbf{In}_{d'}(\varrho) = \varepsilon$ for all $d' \neq d$.

Let us now prove that $(I, (J_m)_{m\in\mathcal{M}})$ is sufficient for σ. As σ is winning, m_{err} is never broadcast, and thus never received, in any σ-run. Hence $m_{err} \notin I$. Furthermore, if we had a word $w \in I \cap J_{m_{err}}$, then we would have a σ-run ϱ and a σ-partial run ϱ' such that m_{err} is broadcast in ϱ', $\mathbf{In}_{d'}(\varrho') \sqsubseteq w \sqsubseteq \mathbf{Out}_d(\varrho)$ and $\mathbf{In}_{d''}(\varrho') = \varepsilon$ for all $d'' \neq d'$. We can assume $d = d'$, as we can rename data.

As a result, we could form a σ-run by renaming data and agents such that their sets of data and agents are disjoint except for d. We then execute the two runs in parallel, and match the unmatched receptions of ϱ' with broadcasts in ϱ, to obtain a σ-run, with no unmatched receptions. This contradicts the fact that σ is winning. Hence $I \cap J_{m_{err}} = \emptyset$. Further, as an empty run is a σ-run, we have $\varepsilon \in I$.

For the second point, we can simply apply Lemma 41.

It remains to show that a σ-local run satisfying 3i and 3ii also satisfies 3a and 3b. Let u be a σ-local run satisfying 3i and 3ii.

- First, we construct a σ-run ϱ whose projection on some agent is u, which shows that u satisfies 3a. Let d be the initial datum of u. As u satisfies 3i, there is some $\mathsf{dec} = (v_0, m_1, \ldots, v_k) \in \mathcal{D}((J_m)_{m\in\mathcal{M}})$ such that u is compatible with dec.
  By definition of $(J_m)_{m\in\mathcal{M}}$, for each j we have a σ-partial run ϱ_j and a non-initial datum d_j such that $\mathbf{In}_{d_j}(\varrho_j) \in \mathcal{L}_{(v_0,m_1,\ldots,v_{j-1})}\!\downarrow$, there are no unmatched receptions with data other than d_j, and $m_j \sqsubseteq \mathbf{Out}_{d_j}(\varrho_j)$.
  We can thus apply Lemma 39, to obtain a σ-partial run with no unmatched reception over d such that $\mathbf{Out}_d(u) \sqsubseteq \mathbf{Out}_d(\varrho)$.
  Furthermore, as u satisfies 3ii, by definition of I, for all $d' \neq d$ there is a σ-run $\varrho_{d'}$ such that $\mathbf{In}_{d'}(u) \sqsubseteq \mathbf{Out}_{d'}(\varrho_{d'})$. We can then apply Lemma 40 on ϱ, with every $d' \neq d$ appearing in u to obtain a σ-run ϱ' such that $\mathbf{Out}_d(u) \sqsubseteq \mathbf{Out}_d(\varrho')$. This shows that $\mathbf{Out}_d(u) \in I$, by definition.
- Let $d' \neq d$ and $m \in \mathcal{M}$ be such that (m, d') is broadcast in u. We can apply Lemma 39 on u and then Lemma 40 on the resulting run, with every $d'' \notin \{d, d'\}$. We obtain a σ-partial run in which (m, d') is broadcast and whose d'-input is the same as u. As a consequence, u satisfies 3b by definition of $(J_m)_{m\in\mathcal{M}}$.

This concludes the proof of that direction.

$\Leftarrow$ By Lemma 36.

◀

C.2 The invariant game

The *invariant game* associated with BGR $\mathcal{G}$ and invariant $(I, (J_m)_{m\in\mathcal{M}})$, which we denote by $\mathcal{IG}(\mathcal{G}, I, (J_m)_{m\in\mathcal{M}})$ is defined as follows: The set of vertices is $Q_{\mathcal{R}}$: the current state in the protocol and a set of registers, which are the ones supposed to contain the initial datum. The initial vertex is q_{init}. From each vertex $q \in Q_{\mathcal{R}}$, players choose a transition from q in $\Delta_{\mathcal{R}}$. Controller chooses the next transition when q is in Q_{ctrl}, Environment when it is in Q_{env}. The state is updated to the target of the transition.

For all play π, we define $\mathtt{reg}(\pi)$ as the set of registers on which there were no record transition in π. Intuitively, $\mathtt{reg}(\pi)$ is the set of registers that contain the initial datum of the local run.

Given a play π, we define its *initial input* $\mathsf{initIn}(\pi)$ as the sequence of letters received with equality transitions with registers of $\mathtt{reg}$. This represents the sequence of letters received with the initial datum. Formally, $\mathsf{initIn}(\varepsilon) = \varepsilon$, and

$$\mathsf{initIn}(\delta_1 \cdots \delta_{k+1}) = \begin{cases} \mathsf{initIn}(\delta_1 \cdots \delta_k)m \text{ if } \delta_{k+1} \text{ is an equality transition } \xrightarrow{\mathbf{rec}(m,=i)} \\ \qquad\qquad\qquad \text{with } i \in \mathtt{reg}(\delta_1 \cdots \delta_k), \\ \mathsf{initIn}(\delta_1 \cdots \delta_k) \text{otherwise.} \end{cases}$$

For all registers i, we define its recent input on i, written $\mathsf{recentIn}_i(\pi)$ like in the previous section: it is the sequence of messages received with equality transitions over register i since its last reset.

We define the *output* $\mathsf{Out}(\pi)$ of π in a different way as in the signature case. It is the sequence of letters broadcast from registers that were in $\mathtt{reg}$ at the time of the broadcast. Intuitively, this is the sequence of letters that are broadcast with the initial datum in the local run. Formally,

$$\mathsf{Out}(\delta_1 \cdots \delta_{k+1}) = \begin{cases} \mathsf{Out}(\delta_1 \cdots \delta_k)m \text{ if } \delta_{k+1} \text{ is a broadcast transition } \xrightarrow{\mathbf{br}(m,i)} \\ \qquad\qquad\qquad \text{with } i \in \mathtt{reg}(\delta_1 \cdots \delta_k), \\ \mathsf{Out}(\delta_1 \cdots \delta_k) \text{ otherwise.} \end{cases}$$

Given a decomposition $\mathsf{dec} = (v_0, m_1, \ldots, v_k)$, we say that a play π is *compatible* with dec if $\pi = \pi_0 \cdots \pi_k$ and for all j we have $\mathsf{initIn}(\pi_j) \in \{m_1, \ldots, m_j\}^*$ and $v_j \sqsubseteq \mathsf{Out}(\pi_j)$.

The objective of the game is then described as follows.

(A) If at some point the play $\pi = \delta_1 \cdots \delta_k$ is not compatible with any decomposition of $\mathcal{D}((J_m)_{m\in\mathcal{M}})$ then Controller wins.

(B) If at some point the play $\pi = \delta_1 \cdots \delta_k$ is such that $\mathsf{recentIn}_i(\pi) \notin I$ for some $i \notin \mathtt{reg}$ then Controller wins.

(C) If at some point the play $\pi = \delta_1 \cdots \delta_k$ is such that $\mathsf{Out}(\pi) \notin I$ then Environment wins.

(D) If at some point of the play a broadcast transition with $i \notin \mathtt{reg}(\pi)$ and $\xrightarrow{\mathbf{br}(m,i)}$ is taken while $\mathsf{recentIn}_i(\pi) \notin J_m$ (with π the play formed so far) then Environment wins.

(E) If the play goes on forever without any of those things happening then Controller wins.

▶ **Lemma 42** (Deciding the invariant game)**.** *There is an elementary function $\varphi(N)$ such that:*

Given a BGR $\mathcal{G}$ over a protocol $\mathcal{R}$ and finite sets of words B and $(B_m)_{m\in\mathcal{M}}$, we can decide in time $\varphi(|\mathcal{R}| + ||B|| + |B| + \sum_{m\in\mathcal{M}} ||B_m|| + |B_m|)$ whether Controller has a winning strategy in $\mathcal{IG}(\mathcal{G}, (B\uparrow)^{\mathsf{c}}, (B_m\uparrow)_{m\in\mathcal{M}})$.

Furthermore, if Environment has a winning strategy then he has a strategy to win in at most $\varphi(|\mathcal{R}| + ||B|| + |B| + \sum_{m\in\mathcal{M}} ||B_m|| + |B_m|)$ steps.

Proof. By Lemma 8, $B\uparrow$ is a regular language, recognised by a deterministic finite automaton $\mathcal{A}_{B\uparrow} = (Q_B, \mathcal{M}, \Delta_B, q_0^B, F_B)$ with $(||B|| + 1)^{(|B|+1)}$ states. Similarly, for each m we can construct a deterministic automaton $\mathcal{A}_{B_m\uparrow} = (Q_{B_m}, \mathcal{M}, \Delta_{B_m}, q_0^{B_m}, F_{B_m})$

Let $I = B\uparrow^{\mathsf{c}}$ and for each $m \in \mathcal{M}$, $J_m = B_m\uparrow$.

We define a deterministic automaton over the alphabet $\Delta_{\mathcal{R}}$ that reads plays $\delta_1 \cdots \delta_k$ of $\mathcal{IG}(\mathcal{G}, I, (J_m)_{m\in\mathcal{M}})$ and accepts exactly the winning plays for Environment.

Consider the alphabet $\mathcal{M} \sqcup \bar{\mathcal{M}}$, where $\bar{\mathcal{M}} = \{\bar{m} \mid m \in \mathcal{M}\}$ is a copy of $\mathcal{M}$.

We define the useful automata in the following claims. Let us define $K = |\mathcal{R}| + |B| + ||B|| + \sum_{m\in\mathcal{M}} |B_m| + ||B_m||$

▷ Claim 43. We can construct an NFA of exponential size in K and recognising the language $\{v_0\bar{m}_1 \cdots v_k \mid (v_0, m_1 \cdots, v_k) \in \mathcal{D}((J_m)_{m\in\mathcal{M}})\}$.

Proof. Consider the language of decompositions defined as $\{v_0\bar{m}_1 \cdots v_{k-1}\bar{m}_k v_k \mid v_0, \ldots, v_k \in \mathcal{M}^*, \bar{m}_1, \ldots, \bar{m}_k \in \bar{\mathcal{M}}$ distinct.$\}$

This language is recognised by an automaton of exponential size which simply checks that each letter of $\bar{\mathcal{M}}$ appears at most once.

We can turn this automaton into a non-deterministic transducer $\mathcal{T}$ that reads a decomposition $v_0\bar{m}_1 \cdots v_{k-1}\bar{m}_k v_k$, outputs all the letters of $\mathcal{M}$ that it reads, and can output letters of $\bar{\mathcal{M}}$ arbitrarily as soon at it has read them before. If some letter of $\bar{\mathcal{M}}$ is repeated then the run is rejected. The set of images of $v_0\bar{m}_1 \cdots v_{k-1}\bar{m}_k v_k$ is exactly $\mathcal{L}_{(v_0,m_1,\ldots,v_k)}$.

By composing this transducer with an automaton recognising J_m, we obtain an automaton $\mathcal{A}_m$ recognising decompositions that have an image in J_m by the transducer, i.e., the language $\{v_0\bar{m}_1 \cdots v_{k-1}\bar{m}_k v_k \mid \mathcal{L}_{(v_0,m_1,\ldots,v_k)} \cap J_m \neq \emptyset\}$.

It is then easy to obtain an automaton recognising $\{v_0\bar{m}_1 \cdots v_k \mid (v_0, m_1 \cdots, v_k) \in \mathcal{D}((J_m)_{m\in\mathcal{M}})\}$ using a product of the automata $(\mathcal{A}_m)_{m\in\mathcal{M}}$.

The resulting automaton is of exponential size in K. ◁

▷ Claim 44. We can construct a deterministic automaton of double-exponential size in K recognising plays compatible with a decomposition of $\mathcal{D}((J_m)_{m\in\mathcal{M}})$.

Proof. We use the automaton recognising $\{v_0\bar{m}_1 \cdots v_k \mid (v_0, m_1 \cdots, v_k) \in \mathcal{D}((J_m)_{m\in\mathcal{M}})\}$ defined in the first claim.

We can define a non-deterministic transducer that takes as input a sequence of transitions $\pi = \delta_1 \cdots \delta_p$ and outputs some decomposition with which it is compatible. The transducer keeps track of $\texttt{reg}(\pi)$ while reading the play.

The transducer simply guesses a sequence $\bar{m}_1 \cdots \bar{m}_k$ of distinct letters of $\bar{\mathcal{M}}$. It outputs them in that order at arbitrary moments while reading π.

When it reads a broadcast transition $\xrightarrow{\mathbf{br}(m,i)}$ over a register currently in $\texttt{reg}(\pi)$, it non-deterministically outputs m or not.

When it reads an equality transition $\xrightarrow{\mathbf{rec}(m,=i)}$ over a register currently in $\texttt{reg}(\pi)$, if $\bar{m}$ has not been broadcast before it goes to a rejecting sink state.

The set of images of a play π are the decompositions it is compatible with. We compose this transducer with the automaton from the first claim to get an automaton recognising the set of plays compatible with some decomposition of $\mathcal{D}((J_m)_{m\in\mathcal{M}})$. ◁

We have automata for I and each J_m, as well as for plays compatible with a decomposition of $\mathcal{D}((J_m)_{m\in\mathcal{M}})$. From those it is straightforward to define an automaton $\mathcal{C}$ reading plays and accepting the ones winning for Environment. We can then determinise it at the cost of an exponential blow-up.

By Proposition 10, we can solve this game in polynomial time in the size of the resulting automaton $\mathcal{C}$ and the size of the arena of $\mathcal{IG}(\mathcal{G}, I, (J_m)_{m\in\mathcal{M}})$ (i.e., $|\mathcal{R}|$), that is, in double-exponential time in $||B|| + |B| + |\mathcal{R}|$.

Furthermore, if Environment has a winning strategy then he has one that guarantees that he wins in at most double-exponentially many steps in K. ◀

We have to show that Controller wins the invariant game if and only if she has a winning control strategy.

▶ **Lemma 45** (From the invariant game to control strategies). *Let $I \subseteq \Sigma^*$ be a downward-closed set and $(J_m)_{m\in\mathcal{M}}$ upward-closed sets such that I contains ε and not m_{err}, $J_{m_{err}} \cap I = \emptyset$, and $\mathcal{L}(I,(J_m)_{m\in\mathcal{M}}) \subseteq I$.*

If Controller wins the invariant game $\mathcal{IG}(\mathcal{G}, I, (J_m)_{m\in\mathcal{M}})$ then there is a control strategy σ such that $(I,(J_m)_{m\in\mathcal{M}})$ is a sufficient invariant for σ.

Proof. Let $\sigma_{\mathcal{IG}}$ be a winning strategy for Controller in $\mathcal{IG}(\mathcal{G}, I, (J_m)_{m\in\mathcal{M}})$.

We define σ as the control strategy in which Controller follows $\sigma_{\mathcal{IG}}$. That is, given a local run $u = (s_0, c_0) \xrightarrow{\mathbf{op}_1(m_1,d_1)}_{\delta_1} \cdots \xrightarrow{\mathbf{op}_k(m_\ell,d_\ell)}_{\delta_\ell} (s_\ell, c_\ell)$, we set $\sigma(u) = \sigma_{\mathcal{IG}}(\delta_1 \cdots \delta_\ell)$. Let d be the initial datum of u. We show that $(I,(J_m)_{m\in\mathcal{M}})$ is a sufficient invariant for σ. To do so, we assume by contradiction that we have a σ-local run $u = (s_0, c_0) \xrightarrow{\mathbf{op}_1(m_1,d_1)}_{\delta_1} \cdots \xrightarrow{\mathbf{op}_\ell(m_\ell,d_\ell)}_{\delta_\ell} (s_\ell, c_\ell)$ such that u satisfies 3i and 3ii but does not satisfy either 3a or 3b. Let d be its initial datum.

We then show that $\pi = \delta_1 \cdots \delta_\ell$ is a losing $\sigma_{\mathcal{IG}}$-play for Controller in $\mathcal{IG}(\mathcal{G}, I)$. As u is a σ-local run, by definition of σ, $\delta_1 \cdots \delta_\ell$ is a $\sigma_{\mathcal{IG}}$-play.

For all $j \in [0,\ell]$ let u_j be the prefix of u up to (s_j, c_j) and $\pi_j = \delta_1 \cdots \delta_j$.

▷ Claim 46. For all index j and $i \notin \mathtt{reg}(\pi_j)$ we have $\mathsf{recentIn}_i(\pi_j) \sqsubseteq \mathbf{In}_{u_j}(c_j(i))$ and $\mathsf{initIn}(\pi_j) \sqsubseteq \mathbf{In}_{u_j}(d)$. Furthermore, if $\mathtt{reg}(\pi_j) \neq \emptyset$ then $\mathsf{Out}(\pi_j) = \mathbf{Out}_d(u_j)$.

Proof. By a straightforward induction on j. ◁

As u satisfies 3i, it is compatible with a decomposition $\mathsf{dec} = (v_0, m_1, \ldots, v_k)$ in $\mathcal{D}((J_m)_{m\in\mathcal{M}})$. We thus have $u = u^0 \cdots u^k$ with $v_i \sqsubseteq \mathbf{Out}_d(u^i)$ and $\mathbf{In}_d(u^i) \in \{m_1, \ldots, m_i\}^*$ for all i.

Let j be the maximal index such that $\mathtt{reg}(\pi_j) \neq \emptyset$, and i_0 the maximal index such that $\pi^0 \cdots \pi^i$ is a prefix of π_j.

Hence we can cut π in the same way: $\pi = \pi^0 \cdots \pi^k$ where π^i is the sequence of transitions of u_i. We can infer using the previous claim that $v_j \sqsubseteq \mathbf{Out}_d(u^i) = \mathsf{Out}(\pi^i)$ for all $i \leq i_0$ and $\mathsf{initIn}(\pi^j) \sqsubseteq \mathbf{In}_d(u^i) \in \{m_1, \ldots, m_i\}^*$.

As a consequence, π_j is compatible with $\mathsf{dec}' = (v_0, m_1, \ldots, m_{i_0}, \varepsilon)$. Furthermore, we have $\mathsf{initIn}(\pi_j) = \mathsf{initIn}(\pi)$ and we can conclude that π is compatible with dec', which is in $\mathcal{D}((J_m)_{m\in\mathcal{M}})$.

We can also infer that all its prefixes π' are compatible with a decomposition of $\mathcal{D}((J_m)_{m\in\mathcal{M}})$: it suffices to consider the decomposition $(v_0, m_1, \ldots, m_i, \varepsilon)$, with i the maximal index such that m_i is appears in $\mathsf{initIn}(\pi')$.

Furthermore, as u satisfies 3ii, for all $d' \neq d$, we have $\mathbf{In}_d(u_\ell) \in I$. For all j,

$$\mathsf{recentIn}_{\pi_j}(i) \sqsubseteq \mathbf{In}_{c_j(i)}(u_j) \sqsubseteq \mathbf{In}_{c_\ell(i)}(u_\ell)$$

. As I is downward-closed, we have $\mathsf{recentIn}_{\pi_j}(i) \in I$ for all $j \in [0,\ell]$.

We know that either u does not satisfy 3a or does not satisfy 3b.

Let us first assume that u does not satisfy 3b. Let $d' \neq d$ and $m \in \mathcal{M}$ be such that u contains a broadcast of (m, d') while $\mathbf{In}_{d'}(u) \notin J_m$. Let j be the index of the first broadcast of (m, d') in u and i the register containing d' at that point. Then δ_j is a broadcast transition $\xrightarrow{\mathbf{br}(m,i)}$, while $\mathsf{recentIn}_{\pi_j}(i) \sqsubseteq \mathbf{In}_{d'}(u_j) \sqsubseteq \mathbf{In}_{d'}(u)$. As J_m is upward-closed, $\mathsf{recentIn}_{\pi_j}(i) \notin J_m$, which means that π is losing for Controller.

Now we assume that u does not satisfy 3a. Let $u = u_- u_+$ be such that u_- is the maximal prefix of u in which d appears at all times. We can cut $\pi = \pi_- \pi_+$ the same way: π_- is the sequence of transitions of u_-, and is also the maximal prefix of π such that $\mathtt{reg}(\pi_-) \neq \emptyset$.

▷ Claim 47. Suppose that π is a winning play for Controller. Then there is a decomposition $\mathsf{dec} = (v_0, m_1, \ldots, v_k) \in \mathcal{D}((J_m)_{m\in\mathcal{M}})$ such that $v_0 \cdots v_k \sqsubseteq \mathbf{Out}(\pi)$ and $\mathbf{Out}_d(u) \in \mathcal{L}_{\mathsf{dec}}$.

Proof. Let $\mathsf{dec} = (v_0, m_1, \ldots, v_k)$ be a decomposition of $\mathcal{D}((J_m)_{m\in\mathcal{M}})$ such that u is compatible with dec. It exists as u satisfies 3i.

Furthermore, we choose it so that $|v_0 \cdots v_k|$ is minimal. Among the ones with minimal $|v_0 \cdots v_k|$, we choose one with k maximal.

Suppose $v_0 \cdots v_k$ is not a subword of $\mathbf{Out}(\pi)$. Then, as $\mathbf{Out}(\pi) = \mathbf{Out}(\pi_-) = \mathbf{Out}_d(u_-)$, we get that $v_0 \cdots v_k \not\sqsubseteq \mathbf{Out}_d(u_-)$.

Let i be the minimal index such that $v_0 \cdots v_i \not\sqsubseteq \mathbf{Out}_d(u_-)$, and let v_{i-} be the maximal prefix of v_i such that $v_0 \cdots v_{i-1} v_{i-} \sqsubseteq \mathbf{Out}_d(u_-)$, and m the letter right after v_{i-} in v_i. Let v_{i+} such that $v_i = v_{i-} m v_{i+}$. The letter m must be broadcast with d in u_+. The same broadcast appears in π_+, say at step j on register i_0. As we assumed that π is winning, we have $\mathsf{recentIn}_{\pi_j}(i_0) \in J_m$. Hence $\mathbf{In}_d(u_j) \in J_m$, as J_m is upward-closed and $\mathsf{recentIn}_{\pi_j}(i_0) \sqsubseteq \mathbf{In}_d(u_j)$.

We have three cases:

- $m \in \{m_1, \ldots, m_{i-1}\}$: then it is easily checked that we can remove m from v_i without affecting the properties of dec, contradicting the minimality of $|v_0 \cdots v_k|$.
- $m = m_\ell$ for some $\ell \in [i, k]$: then we can use the following decomposition:

$$(v_0, m_1, \ldots, v_{i-1}, m_i, v_{i-}, m, v_{i+}, \ldots, m_{\ell-1}, v_{\ell-1} v_\ell, m_{\ell+1}, \ldots, v_k)$$

  instead of dec, again contradicting the minimality of $|v_0 \cdots v_k|$.
- $m \notin \{m_1, \ldots, m_k\}$. Then we use the following decomposition instead of dec:

$(v_0, m_1, \ldots, v_{i-1}, m_i, v_{i-}, m, v_{i+}, m_{i+1}, \ldots, v_k)$. This contradicts the minimality of $|v_0 \cdots v_k|$

.

As a consequence, we obtain that $v_0 \cdots v_k$ is a subword of $\mathbf{Out}(\pi)$. It remains to show that $\mathbf{Out}_d(u) \in \mathcal{L}_{\mathsf{dec}}$. To do that, let us assume that u_+ contains a broadcast with d of a letter that is not in $\{m_1, \ldots, m_k\}$. Let m be the letter in the first such broadcast of u_+, i the corresponding register, and j the index of the step. Since we assumed that π is winning, we have $\mathsf{recentIn}_{\pi_j}(i) \in J_m$. Hence $\mathbf{In}_d(u_j) \in J_m$, as J_m is upward-closed and $\mathsf{recentIn}_{\pi_j}(i) \sqsubseteq \mathbf{In}_d(u_j)$. Moreover, every letter in $\mathbf{In}_d(u_j)$ must be in $\{m_1, \ldots, m_k\}$, as u is compatible with dec.

As a result, $\mathbf{In}_d(u_j) \in J_m \cap \mathcal{L}_{\mathsf{dec}}$, hence $J_m \cap \mathcal{L}_{\mathsf{dec}} \neq$ and thus $(v_0, m_1, \ldots, v_k, m, \varepsilon) \in \mathcal{D}((J_m)_{m\in\mathcal{M}})$. Moreover, u is compatible with this decomposition. This contradicts the maximality of k.

In conclusion, we have shown that dec matches all the conditions of the claim. ◁

Suppose π is winning, then by this claim we have a decomposition $\mathsf{dec} = (v_0, m_1, \ldots, v_k) \in \mathcal{D}((J_m)_{m\in\mathcal{M}})$ such that $v_0 \cdots v_k \sqsubseteq \mathbf{Out}(\pi)$ and $\mathbf{Out}_d(u) \in \mathcal{L}_{\mathsf{dec}}$.

As π is winning, we have $\mathbf{Out}(\pi) \in I$, and thus $\mathbf{Out}_d(u) \in \mathcal{L}(I, (J_m)_{m\in\mathcal{M}})$. Since $\mathcal{L}(I, (J_m)_{m\in\mathcal{M}}) \subseteq I$, we get $\mathbf{Out}_d(u) \in I$, and thus u satisfies 3a, a contradiction.

In conclusion, we obtained that π is a losing $\sigma_{\mathcal{IG}}$-play, which contradicts the assumption that $\sigma_{\mathcal{IG}}$ is winning. As a consequence, $(I, (J_m)_{m\in\mathcal{M}})$ is a sufficient invariant for σ.¨ ◀

▶ **Lemma 48** (From control strategies to the invariant game). *Let σ be a control strategy. Let $I \subseteq \Sigma^*$ be a downward-closed set and $(J_m)_{m\in\mathcal{M}}$ upward-closed sets such that I contains ε and not m_{err}, $J_{m_{err}} \cap I = \emptyset$, and $\mathcal{L}(I, (J_m)_{m\in\mathcal{M}})) \subseteq I$.*

Let B be the basis of I^{c} and B_m the basis of J_m for all m.
If Environment wins the invariant game $\mathcal{IG}(\mathcal{G}, I, (J_m)_{m\in\mathcal{M}})$ then there is a σ-local run of length at most $\varphi(|\mathcal{R}|+|B|+||B||+\sum_{m\in\mathcal{M}}|B_m|+||B_m||)$ satisfying 3i and 3ii and dissatisfying either 3a or 3b.

Proof. Let $N = |\mathcal{R}| + |B| + ||B|| + \sum_{m\in\mathcal{M}}|B_m| + ||B_m||$. By Lemma 42 and Proposition 10 there exists $\tau_{\mathcal{IG}}$ a winning strategy $\tau_{\mathcal{IG}}$ for Environment in the invariant game $\mathcal{IG}(\mathcal{G}, I, (J_m)_{m\in\mathcal{M}})$ such that Environment always wins in at most $\varphi(N)$ steps. We construct a σ-local run of length at most $\varphi(N)$ satisfying 3i and 3ii and dissatisfying either 3a or 3b. To do so, we apply $\tau_{\mathcal{IG}}$ to choose transitions and we choose data by always picking a datum never seen before in the run, when the datum is not determined by the transition. Let (s_0, c_0) be an initial configuration of $\mathcal{R}$. We define iteratively a sequence of steps $(s_{\ell-1}, c_{\ell-1}) \xrightarrow{\mathbf{op}_\ell(m_\ell, d_\ell)}_{\delta_\ell} (s_\ell, c_\ell)$ as follows. Suppose we defined them up to $(s_{\ell-1}, c_{\ell-1})$, and let $u_{\ell-1}$ be the local run defined so far. We first choose δ_ℓ:

- If $s_{\ell-1} \in Q_{\mathsf{ctrl}}$ then $\delta_\ell = \sigma(\delta_1 \cdots \delta_{\ell-1})$,
- otherwise $\delta_\ell = \tau_{\mathcal{IG}}(\delta_1 \cdots \delta_{\ell-1})$.

We then choose d_ℓ:

- If δ_ℓ is a broadcast transition of letter m, we set d_ℓ as the initial datum of the local run.
- If δ_ℓ is a record transition, we pick a datum d_k that does not appear in $u_{\ell-1}$.
- If $\delta_\ell = s_{\ell-1} \xrightarrow{\mathbf{rec}(m,=i)} s_\ell$ is an equality transition of letter m, we set $d_\ell = c_{\ell-1}(i)$.

Clearly we maintain the fact that u_ℓ is a σ-local run and $\delta_1 \cdots \delta_\ell$ is a $\tau_{\mathcal{IG}}$-play in $\mathcal{IG}(\mathcal{G}, I, (J_m)_{m\in\mathcal{M}})$. We stop when $\delta_1 \cdots \delta_\ell$ is winning for Environment in $\mathcal{IG}(\mathcal{G}, I, (J_m)_{m\in\mathcal{M}})$, which happens for some $\ell \leq \varphi(N)$. Let M be the final value of ℓ and $u = u_M$ be the local run obtained at the end. Let d be its initial datum. It remains to show that u satisfies 3i and 3ii and dissatisfies either 3a or 3b. To do so, we rely on the following claim:

▷ Claim 49. For all register i and index ℓ such that $i \notin \mathtt{reg}(\delta_1 \cdots \delta_\ell)$, $\mathsf{recentIn}_i(\delta_1 \cdots \delta_\ell) = \mathbf{In}_{u_\ell}(c_\ell(i))$. Furthermore, $\mathsf{Out}(\delta_1 \cdots \delta_\ell) = \mathbf{Out}_d(u_\ell)$ and $\mathsf{initIn}(\delta_1 \cdots \delta_\ell) = \mathbf{In}_d(u_\ell)$.

Proof. By a straightforward induction on ℓ. ◁

Let $\pi_\ell = \delta_1 \cdots \delta_\ell$ for all ℓ, and let $\pi = \pi_M$.

First we show that u satisfies 3i: As π is winning for Environment, it is compatible with some decomposition $\mathsf{dec} = (v_0, m_1, \ldots, v_k) \in \mathcal{D}((J_m)_{m\in\mathcal{M}})$. Thus $\pi = \pi^0 \cdots \pi^k$ with $v_j \sqsubseteq \mathsf{Out}(\pi^j)$ and $\mathsf{initIn}(\pi^j) \in \{m_1, \ldots, m_j\}^*$, for all j.

We divide u like π, $u = u^0 \cdots u^k$. As a consequence of the claim, we obtain $v_j \sqsubseteq \mathbf{Out}_d(u^j)$ and $\mathbf{In}_d(u^j) \in \{m_1, \ldots, m_j\}^*$, for all j. Thus u is compatible with dec.

Now, we show that u satisfies 3ii. Let $d' \neq d$. If d' is stored in a register at some point in u, let ℓ be the maximal index such that $c_\ell(i) = d'$ for some i. There can be no step involving d' after ℓ, as d' would need to be stored in a register, contradicting the maximality of ℓ. As a consequence, $\mathbf{In}_{d'}(u) = \mathsf{recentIn}_{\pi_\ell}(i)$. As π is winning for Environment, we have $\mathsf{recentIn}_{\pi_\ell}(i) \in I$. If $\mathbf{In}_{d'}(u) = \varepsilon$ then clearly $\mathbf{In}_{d'}(u) \in I$ by assumption on I. If d' is never stored in a register then $\mathbf{In}_{d'}(u) = \varepsilon \in I$.

We have shown that u satisfies 3i and 3ii.

If $\mathsf{Out}(\pi) \notin I$ then $\mathbf{Out}_d(u) \notin I$, by the claim, hence u does not satisfy 3a.

If $\mathsf{Out}(\pi) \in I$, since π is winning for Environment, there must be an index ℓ such that the ℓth transition of π is a broadcast transition $\xrightarrow{\mathbf{br}(m,i)}$, but $\mathsf{recentIn}_\pi(i) \notin J_m$. In that case, we have $\mathbf{In}_{c_{\ell+1}(i)}(u_{\ell+1}) \notin J_m$ and $u_{\ell+1}$ contains a broadcast of $(m, c_{\ell+1}(i))$.

As a consequence, we have found a prefix $u_{\ell+1}$ of u which does not satisfy 3b. As 3i and 3ii hold for u, it is easy to see that they must also hold for all its prefixes.

In all cases we have found a σ-local run of length at most $\varphi(N)$ which satisfies 3i and 3ii but dissatisfies either 3a or 3b.

This concludes our proof. ◀

▶ **Lemma 50** (Bounding invariants). *There is an elementary function $\psi(N)$ such that the following statement holds.*

Let $\mathcal{G}$ a BGR. There is a winning control strategy for $\mathcal{G}$ if and only if there is a sequence of words $w_1, \dots, w_k \in \mathcal{M}^*$ and subsets $B, (B_m)_{m\in\mathcal{M}}$ of $\{w_1, \dots, w_k\}$ such that

- *B contains m_{err} and not ε and $B_{m_{err}}\uparrow \cap B\uparrow^{\mathsf{c}} = \emptyset$*
- *$\mathcal{L}(B\uparrow^{\mathsf{c}}, (B_m\uparrow)_{m\in\mathcal{M}}, \subseteq)B\uparrow^{\mathsf{c}}$*
- *Controller wins $\mathcal{IG}(\mathcal{G}, B\uparrow^{\mathsf{c}}, (B_m\uparrow)_{m\in\mathcal{M}})$,*
- *B and all B_m are antichains for the subword order $\sqsubseteq$,*
- *$B \cup \bigcup_{m\in\mathcal{M}} B_m = \{w_1, \dots, w_k\}$,*
- *for all $i \in [1, k]$, $|w_i| \leq \psi(|w_{i-1}|)$.*

Proof. By Lemma 45, if there are such sets of words B and $(B_m)_{m\in\mathcal{M}}$, then there is a control strategy such that $(B\uparrow^{\mathsf{c}}, (B_m\uparrow)_{m\in\mathcal{M}}$ is a sufficient invariant for σ. Hence, by Lemma 29, σ is a winning control strategy.

Conversely, suppose there is a winning control strategy σ. By Lemma 29 there is a sufficient invariant $(I, (J_m)_{m\in\mathcal{M}})$ for σ. As I^{c} is upward-closed it has a finite basis B. Similarly, each J_m has a finite basis B_m.

The first two conditions hold by definition, as $(I, (J_m))$ is a sufficient invariant.

By Lemma 48 Controller wins $\mathcal{IG}(\mathcal{G}, I, (J_m)_{m\in\mathcal{M}})$, so the third condition of the lemma is satisfied.

For the third condition, by definition, all basis are antichains.

Let $w_0, w_1, ..., w_k$ be the elements of $B \cup \bigcup_{m\in\mathcal{M}} B_m$ sorted by length, i.e., $|w_i| \leq |w_{i+1}|$ for all i. We can assume that we chose I and $(J_m)_{m\in\mathcal{M}}$ so that k would be minimal.

By minimality of k, for all $j \in [1, k]$, $(B', (B'_m)_{m\in\mathcal{M}})$ is not a sufficient invariant for σ, with $B' = B \cap \{w_i \mid i < j\}$ and for all m, $B'_m = B_m \cap \{w_i \mid i < j\}$. Let $I' = B'\uparrow^{\mathsf{c}}$ and $J'_m = B'_m\uparrow$ for all m. Note that $I \subseteq I'$ while $J'_m \subseteq J_m$ for all m.

A possibility is that $m_{err} \in I'$. As $m_{err} \in B$, we then have $|w_j| \leq 1$.

Another possibility is that $I' \cap J'_{m_{err}} \neq \emptyset$. As a consequence, there is a word $w \in B'_{m_{err}}$ with no subword in B'. As this word is of length at most $|w_{j-1}|$, we conclude that there is a word of length at most $|w_{j-1}|$ in $B \setminus B'$, hence $|w_j| \leq |w_{j-1}|$.

Thirdly, we may have $\mathcal{L}(I', (J'_m)_{m\in\mathcal{M}}) \not\subseteq I'$. Then there is a decomposition $\mathsf{dec} = (v_0, m_1, \dots, v_k) \in \mathcal{D}(I', (J'_m)_{m\in\mathcal{M}})$ and $w \in \mathcal{L}_{\mathsf{dec}}$ such that $w \notin I'$.

It is easy to construct deterministic automata recognising $\mathcal{L}(I', (J'_m)_{m\in\mathcal{M}})$ and I' of double-exponential size in $|\mathcal{R}|$, $|\mathcal{M}|$, B' and $(B'_m)_{m\in\mathcal{M}}$, by using Lemma 8 and Claim 43.

Hence we can find such a w of at most double-exponential size, and thus the decomposition $\mathsf{dec} = (v_0, m_1, \dots, v_k)$ also has at most double-exponential size. Now note that $v_0 \cdots v_k$ is in I', but cannot be in I: otherwise, we would have $w \in \mathcal{L}(I, (J'_m)_{m\in\mathcal{M}}) \subseteq \mathcal{L}(I, (J_m)_{m\in\mathcal{M}})$, while $w \notin I' \supseteq I$, a contradiction. Hence there is a word of at most double-exponential size in $|\mathcal{G}|$, B' and $(B'_m)_{m\in\mathcal{M}}$ (thus of at most triple-exponential size in $|w_{j-1}|$) that is in B' but not B. As a consequence, $|w_j|$ is at most triply-exponential in $|w_{j-1}| + |\mathcal{M}| + |\mathcal{R}|$.

The last case is when there is a run σ-local run which satisfies 3i and 3ii but dissatisfies either 3a or 3b, with respect to $(I', (J'_m)_{m\in\mathcal{M}})$. By Lemma 48, there is such a σ-local run u of length at most $K = \varphi(|\mathcal{R}| + ||B'|| + \sum_{m\in\mathcal{M}} ||B'_m||) \leq \varphi(|\mathcal{R}| + (|\mathcal{M}| + 1)|w_{j-1}|)$.

As $J'_m \subseteq J_m$ for all m, we have $\mathcal{D}((J'_m)_{m\in\mathcal{M}}) \subseteq \mathcal{D}((J_m)_{m\in\mathcal{M}})$. As a consequence, u satisfies 3i with respect to $(I, (J_m)_{m\in\mathcal{M}})$.

As $I \subseteq I'$, if u satisfies 3a with respect to $(I, (J_m)_{m\in\mathcal{M}})$ then it also satisfies it with respect to $(I', (J'_m)_{m\in\mathcal{M}})$.

Two cases remain: either u satisfies 3ii with respect to $(I', (J'_m)_{m\in\mathcal{M}})$ and not $(I, (J_m)_{m\in\mathcal{M}})$, or satisfies 3b with respect to $(I, (J_m)_{m\in\mathcal{M}})$ and not $(I', (J'_m)_{m\in\mathcal{M}})$.

We examine the two cases:

- Suppose u satisfies 3ii with respect to $(I', (J'_m)_{m\in\mathcal{M}})$ and not $(I, (J_m)_{m\in\mathcal{M}})$. Let d' be a datum such that $\mathbf{In}_{d'}(u) \notin I$. As $\mathbf{In}_{d'}(u) \in I'$, we found a word of length at most K that is in I' but not I.
- Suppose u satisfies 3b with respect to $(I, (J_m)_{m\in\mathcal{M}})$ and not $(I', (J'_m)_{m\in\mathcal{M}})$. Then there exist $m \in \mathcal{M}$ and $d' \neq d$ such that u contains a broadcast of (m, d') and $\mathbf{In}_{d'}(u) \notin J'_m$, while $\mathbf{In}_{d'}(u) \in J_m$. Furthermore, we have $|\mathbf{In}_{d'}(u)| \leq |u| \leq K$

In both cases, there exists w_ℓ with $\ell \geq j$ such that $w_\ell \sqsubseteq w$, and thus $|w_\ell| \leq |w| \leq K$.

As $|w_j| \leq |w_\ell|$, we have $|w_j| \leq K$. As $||B'|| \leq |w_{j-1}|$ and $||B'_m|| \leq |w_{j-1}|$, we obtain $|w_j| \leq \varphi(|\mathcal{R}| + (|\mathcal{M}| + 1)|w_{j-1}|)$.

We can then simply take a suitable elementary function so that $|w_{j-1}| \leq \psi(|\mathcal{R}| + |\mathcal{M}| + |w_j|)$ ◀

C.3 Main theorem

▶ **Theorem 30** (Main theorem). *SafeStrat is decidable and $\mathbf{F}_{\omega^\omega}$-complete.*

Proof. It was shown in [13] that the coverability problem is $\mathbf{F}_{\omega^\omega}$-hard. As coverability is the particular case of SafeStrat where there are no controller nodes, this immediately yields the same lower bound for SafeStrat.

Let us now show the upper bound. Let $\mathcal{G}$ a BGR. We once again apply the Length Function Theorem.

Consider a sequence of words $w_1, \ldots, w_k \in \mathcal{M}^*$ and subsets $B, (B_m)_{m\in\mathcal{M}}$ of $\{w_1, \ldots, w_k\}$ satisfying the conditions of Lemma 50.

We use a fresh letter $\# \notin \mathcal{M}$. For each w_i we define $w'_i = \#^{|\mathcal{R}|+|\mathcal{M}|} w_i \#\#$ if $w_i \in B$, and $w'_i = \#^{|\mathcal{R}|+|\mathcal{M}|} w_i \# m$ with m such that $w_i \in B_m$ otherwise.

By the second condition of Lemma 50, w'_i is well-defined for all i.

Note that the sequence $w'_1 \cdots w'_k$ is an antichain: as $\#$ does not appear in any w_i, $w'_i \sqsubseteq w'_j$ implies that $w_i \sqsubseteq w_j$, and that they both belong to B or to some common B_m. This is impossible as all those sets are antichains.

Furthermore, for all i, we have $|w'_{i+1}| \leq \psi(|\mathcal{R}|+|\mathcal{M}|+|w_i|)+|\mathcal{R}|+|\mathcal{M}|+2 \leq \psi(|w'_i|)+|w'_i|$. As $g : n \mapsto \psi(n)+n$ is a primitive recursive function, by the Length function theorem we obtain a function $f \in \mathscr{F}_{\omega^{|\mathcal{M}|}}$ such that every (g, n)-controlled bad sequence of words $w_0, w_1, ..., w_k$ has at most $f(n)$ terms.

As m_{err} is in B, $|w_0| \leq 1$, thus $|w'_0| \leq |\mathcal{R}| + |\mathcal{M}| + 3$ We therefore have $|w_i| \leq g^{(i)}(|\mathcal{R}| + |\mathcal{M}| + 3)$ for all i. As a consequence, we have $k \leq f(|\mathcal{R}| + |\mathcal{M}| + 3)$.

Our algorithm guesses a sequence of words of sorted by length $w_1, ..., w_k$ with $k \leq f(|\mathcal{R}| + |\mathcal{M}| + 3)$ such that $|w_{i+1}| \leq \psi(|w_i|)$ for all i. The algorithm then guesses subsets B and $(B_m)_{m\in\mathcal{M}}$ that cover $\{w_i \mid i \in [1, k]\}$.

It checks that Controller wins $\mathcal{IG}(\mathcal{G}, B{\uparrow}^{\mathsf{c}}, (B_m{\uparrow})_{m\in\mathcal{M}},)$. We accept if she does and reject otherwise.

This can be done in double-exponential time in $|\mathcal{R}| + k + |w_k|$, by Lemma 42. We can make this algorithm deterministic with an exponential blow-up in the time complexity. By Lemma 50, this algorithm is correct.

The time required by this algorithm is therefore $h(f(|\mathcal{R}| + |\mathcal{M}| + 3))$ with h a primitive recursive function. As $\mathscr{F}_{\omega^{|\mathcal{M}|}}$ is closed under composition with primitive recursive functions, the algorithm takes a time bounded by a function of $\mathscr{F}_{\omega^{|\mathcal{M}|}}$. As a consequence, the problem is in $\mathbf{F}_{\omega^\omega}$. ◀

D Missing proofs from Section 6

We say that a local run has *organised data* if

- if whenever a datum is received for the first time, it is greater than the initial datum and all data received previously.
- Each datum is recorded at most once in the registers.

▶ **Proposition 51.** *There is a function $h : \mathbb{N} \to \mathbb{N}$ such that for all BGR $\mathcal{G}$, if a control strategy is losing then there exists a σ-run ϱ in which every local run has length at most $h(|\mathcal{G}|)$ and has organised data in which m_{err} is broadcast.*

Proof. Let us start by defining an *execution tree* as a tree of the following form:

- There are two types of nodes, *word nodes* and *run nodes*
- The children of a word node are run nodes, and the children of a run node are word nodes.
- For all run node ν with a label u and all $d \in \mathbb{D}$ such that $\mathbf{In}_d(u) \neq \varepsilon$, ν has a child with a label w such that $\mathbf{In}_d(u) \sqsubseteq w$.
- For all word node ν labelled w, for all child ν' of ν labelled u, $w \sqsubseteq \mathbf{Out}_{sign}(u)$

Consider the following algorithm: We start with an execution tree made only of a root labelled m_{err}. We maintain a set of word nodes O, initially containing only the root. The word nodes in O are called *open*, others are called *closed*

While O is not empty, we apply the following steps:

- If there is a run node ν whose children are all closed, let ν' be its parent, labelled w. We remove every node that was added to the tree after ν' (in particular, we remove all of its descendants). Then, we remove ν from O.
- Otherwise, let B be the set of labels of open nodes, we define $I = B{\uparrow}^{\mathsf{c}}$. By Lemma 21, there exists a σ-local run u of length at most $\varphi(\mathcal{R}, B)$ such that $\mathbf{Out}_{sign}(u) \notin I$, $\mathbf{In}_d(u) \in I$ for all d and no datum is recorded twice. Let ν be an open node with a label w such that $w \sqsubseteq \mathbf{Out}_{sign}(u)$. It exists by definition of B and I. We add a child ν' to ν, labelled by u. Then, consider the set $\{\mathbf{In}_d(u) \mid d \in \mathbb{D}\} \setminus \{\varepsilon\}$, let B_u be its set of minimal elements for $\sqsubseteq$. For each $v \in B_u$ we add a child labelled v to ν', and we add all those children to O.

Note that when we remove a node we remove all nodes added after that one. As a consequence, at all times we can enumerate open nodes $O = \{\nu_0, \nu_1, \dots, \nu_k\}$ in their order of appearance, and we obtain $|w_i| \leq |u_i| \leq \varphi(\mathcal{R}, \{w_1, \dots, w_{i-1}\})$ for all i, with $w_1, \dots, w_k$ the labels of $\nu_1, \dots, \nu_k$ and $u_1, \dots, u_k$ the labels of their respective parents. Additionally, we maintain the fact that the sequence $w_1, \dots, w_k$ is a bad sequence. We can then apply the Length Function Theorem to bound k by $f(|\mathcal{R}|)$ with f a function of $\mathbf{F}_{\omega^\omega}$.

We also obtain a bound $h(|\mathcal{R}|)$ on the length of run node labels. As a consequence, the number of data appearing in each local run is bounded by that same bound, and thus the degree of the tree is at most $h(|\mathcal{R}|)$.

As every node is a leaf or an open node or the child of an open node, we get a bound $b(|\mathcal{R}|)$ on the size of the tree. As a consequence, the set of trees we see is finite. In order to show that the algorithm terminates, we simply have to show that we cannot loop.

Given the tree at some point of the algorithm, let $\nu_0, \ldots, \nu_k$ be the set of word nodes in their order of creation. For each i, let x_i be 0 if ν_i is open and 1 if it is closed. It is easy to check that the sequence $x_0 \cdots x_k$ increases at each step for the lexicographic ordering. As a result, we never see the same tree twice. The algorithm therefore terminates in at most $c(|\mathcal{R}|)$ steps.

▷ Claim 52. After each iteration, for all closed node ν labelled w, ν is a leaf and there is a σ-run in which every local run has organised data and has length at most $h(|\mathcal{R}|)$ and in which the sequence w is broadcast by some agent.

Proof. We proceed by induction on the number of iterations. This property is clearly true at the beginning as there are no closed nodes.

For the induction step, note that we never add children to closed nodes and only turn leaves into closed nodes. Hence we maintain the fact that every closed node is a leaf.

Furthermore, say we turn an open node labelled w into a closed one. We do so when it has a child ν' whose children are all closed. Let u be the label of ν' and $w_1, \ldots, w_n$ the labels of its children. By induction hypothesis, for each i we have a σ-run ϱ_i in which each local run has organised data and length at most $h(|\mathcal{R}|)$ and in which an agent broadcasts w_i.

For each d received in u, we know that there is an i such that $\mathbf{In}_d(u) \sqsubseteq w_i$. We define ϱ_d as ϱ_i where data have been renamed so that w_i is broadcast with datum d and all other data are fresh and do not appear in u.

Let $d_1, \ldots, d_m$ be the data received in u, in order of appearance. Let d_0 be the initial datum of u. For all $j \in [2, m]$, in increasing order, let A_{j-1} be such that all data appearing in ϱ_{j-1} are below A_{j-1}. Let $A_0 = d_0 + 1$. Define ϱ'_j as ϱ_{d_j} where each datum d' has been renamed into $d' + A_{j-1}$. The runs ϱ'_j use disjoint sets of data and in each ϱ'_j an agent broadcasts $\mathbf{In}_{d_j}(u)$ with datum $d'_j = d_j + A_{j-1}$. In particular we have $d_0 < d'_1 < \cdots < d'_m$. We have also maintained the fact that all local runs in those runs have organised data.

We rename each datum d_j with in u to d'_j and obtain a local run u' with organised data and $\mathbf{In}_{d_j}(u) = \mathbf{In}_{d'_j}(u')$. We can execute all runs ϱ'_j and u over disjoint sets of agents, and use the broadcasts in each ϱ'_j to match the receptions in u. This gives us a σ-run ϱ_i in which each local run has increasing data and length at most $h(|\mathcal{R}|)$ and in which an agent broadcasts w (by executing u). ◁

As the algorithm terminates, eventually the root is closed. By the claim above, we have a σ-run ϱ in which each local run has length at most $h(|\mathcal{R}|)$ and in which an agent broadcasts m_{err}. ◀

We recall Ramsey's theorem on infinite hypergraphs. Given a set S and $k \in \mathbb{N}$, we use the notation $\binom{S}{k}$ for the set of subsets of S of size k.

▶ **Theorem 53** (Ramsey's theorem on infinite hypergraphs). *Let V be an infinite set of vertices and $k \in \mathbb{N}$. Let $\mathbf{col} : \binom{V}{k} \to C$ with C a finite set of colours. Then there exists an infinite subset $V' \subseteq V$ and $c \in C$ such that $\mathbf{col}(\binom{V'}{k}) = \{c\}$.*

▶ **Theorem 31.** *There is a winning data-aware control strategy for $\mathcal{G}$ if and only if there is a winning control strategy for $\mathcal{G}$.*

Proof. The right-to-left direction is clear.

For the left-to-right direction, suppose there is a winning data-aware control strategy σ for $\mathcal{G}$. Let $K = h(|\mathcal{R}|)$ with h as defined in Proposition 51. Let R_K be the set of σ-local runs with organised data of length at most K.

We define a function $\mathbf{col} : \binom{\mathbb{D}}{K+1} \to 2^{\mathsf{R}_K}$ as follows. Let D be a set of $R+1$ data. Let $d_0, \cdots, d_R$ be the elements of $\mathbb{D}$ in increasing order. Then $\mathbf{col}(D)$ is the set of σ-local runs with organised data of length at most R such that the initial datum is d_0 the other data appearing in the run are $d_1, \ldots, d_k$ for some k. With the organised data property and those conditions, the local run is fully determined by its sequence of transitions. As a result, $|\mathbf{col}(D)| \leq |\Delta|^K$.

In a local run with at most B steps, at most $B+1$ data appear. As a result, every element of R_K has an antecedent by $\mathbf{col}$. We can now apply Theorem 53. We obtain an infinite set $\mathbb{D}' \subseteq \mathbb{D}$ of data and a set of local runs R such that $\mathbf{col}(\binom{\mathbb{D}'}{K+1}) = \{R\}$.

Let $d_0, \ldots, d_K \in \mathbb{D}'$ with $d_0 < \cdots < d_K$. Define the strategy $\sigma' : \Delta^* \to \Delta$ which, given a sequence of transitions, takes the same decision as σ over the unique local run with that sequence of transitions, organised data, and using data $\{d_0, \ldots, d_k\}$ for some k, with d_0 the initial datum.

If σ' was losing, we would have a run in which every local run has length at most K and organised data in which m_{err} is broadcast. This run is, however, also a σ-run, which is a contradiction. As a result, σ' is winning. ◀

E Missing proofs from Section 7

In this section we prove the following result.

▶ **Theorem 33.** *SafeStrat is NExpTime-complete for 1BGR.*

We start with the upper bound.

▶ **Proposition 54.** *SafeStrat is in NExpTime on 1BGR.*

For the rest of this section we fix a 1BGR $\mathcal{G} = (\mathcal{R}, Q_{\mathsf{ctrl}}, Q_{\mathsf{env}}, m_{err})$.

We will use the following criterion for the existence of positional strategies.

▶ **Proposition 55** ([17])**.** *If an objective is submixing then player P_0 has a positional optimal strategy in all games with this objective.*

The *output game* is played on $\mathcal{R}$, with players picking transitions from their respective states. It has two parameters: an invariant $(I, (J_m)_{m \in \mathcal{M}})$, and a set of record transitions $T \subseteq \Delta$. We will use the term $(I, (J_m)_{m\in\mathcal{M}}, T)$-*output game* for the output game with those parameters.

The winning condition is defined as follows:

(O1) If at some point the play is not compatible with any decomposition of $\mathcal{D}((J_m)_{m\in\mathcal{M}})$, then Controller wins.

(O2) If the output of the play is not in I then Environment wins.

(O3) If we reach a record transition then Controller wins if it is in T and Environment wins otherwise.

(O4) If the play goes on forever without any of the previous things happening then Controller wins.

▶ **Lemma 56.** *If Controller wins an output game then she has a positional winning strategy.*

The *echo game* is also played on $\mathcal{R}$, with players picking transitions from their respective states. It has as parameters an invariant $(I, (J_m)_{m\in\mathcal{M}})$, a set of record transitions $T \subseteq \Delta$ and a record transition $t = q \xrightarrow{\mathbf{rec}(m,\downarrow 1)} q'$. The play starts by taking transition t, and continues from q'.

(E1) If at some point the recent input on 1 is not in I, Controller wins.
(E2) If at some point we make a broadcast with letter m while the recent input on 1 is not in J_m, then Environment wins.
(E3) If we reach a record transition the game stops: If that transition is in T then Controller wins, otherwise Environment does.
(E4) If the play goes on forever without any of those things happening then Controller wins.

▶ **Lemma 57.** *If Controller wins an output game then she has a positional winning strategy.*

Proof. We show that Controller's objective in an output game is submixing. This proves the lemma by applying [17, Theorem 4.5].

Consider two losing plays for Controller π and π', and a third play $\bar{\pi} = \pi_0\pi'_0\pi_1\cdots$ obtained by shuffling the two. We show that $\bar{\pi}$ is also losing for Controller. As π and π' are losing for Controller, we can consider them as finite: the victory of Environment is witnessed by a finite prefix. We can cut π and π' into $\pi = \pi_0\pi_1\cdots\pi_m$ and $\pi' = \pi'_0\pi'_1\cdots\pi'_m$ so that $\bar{\pi} = \pi_0\pi'_0\pi_1\cdots\pi_m\pi'_m$ (note that π'_m can be empty).

Clearly no transition of T is seen in π or π', thus not in $\bar{\pi}$ as well. Let $\tilde{\pi}$ a prefix of $\bar{\pi}$, we show that it is compatible with some decomposition of $\mathcal{D}((J_m)_{m\in\mathcal{M}})$. Let $\tilde{m}_1, \ldots, \tilde{m}_k$ be the set of letters received along $\tilde{\pi}$, in that order. Let $\tilde{\pi} = \tilde{\pi}_0\cdots\tilde{\pi}_k$ so that for each i the first step of $\tilde{\pi}_i$ is the first reception of m_i. Let $\tilde{v}_i$ be the sequence of letters broadcast in $\tilde{\pi}_i$, for all i. Let $\tilde{\mathsf{dec}} = (\tilde{v}_0, \tilde{m}_1, \ldots, \tilde{v}_k)$. Clearly $\tilde{\pi}$ is compatible with dec.

It remains to show that $\tilde{\mathsf{dec}} \in \mathcal{D}((J_m)_{m\in\mathcal{M}})$. Let $i \in [1, k]$, we need to find a word in $\mathcal{L}_{(\tilde{v}_0,\tilde{m}_1,\ldots,\tilde{v}_{i-1})} \cap J_{m_i}$. For that, we observe that the reception of $\tilde{m}_i$ happens in either a segment from π or from π'. We assume that it is from π, the other case is symmetric. Since every prefix of π is compatible with some decomposition of $\mathcal{D}((J_m)_{m\in\mathcal{M}})$, in particular the prefix of π up to that reception of $\tilde{m}_i$ is compatible with one. Thus there exists $\mathsf{dec} = (v_0, m_1, \ldots, v_\ell)$ such that $\mathcal{L}_{\mathsf{dec}} \cap J_{\tilde{m}_i} \neq \emptyset$ and with which π is compatible. Let $w \in \mathcal{L}_{\mathsf{dec}} \cap J_{\tilde{m}_i}$, w can be obtained from $v_0\cdots v_\ell$ by adding letters from $\{m_1, \ldots, m_j\}$ to each v_j.

As this prefix of π is fully contained in $\tilde{\pi}$, we can find the same sequence of broadcast $v_0\cdots v_\ell$ in $\tilde{\pi}$. Moreover, for each j, the first reception of m_j can only be earlier in $\tilde{\pi}$ than in π, hence $\tilde{\mathsf{dec}}$ allows us to find $v_0\cdots v_\ell$ and to add the same letters at the same places. As a consequence, $w \in \mathcal{L}_{(\tilde{v}_0,\tilde{m}_1,\ldots,\tilde{v}_{i-1})}$.

It follows that every prefix of $\bar{\pi}$ is compatible with some decomposition of $\mathcal{D}((J_m)_{m\in\mathcal{M}})$. As a consequence, Controller does not win at any point in $\bar{\pi}$.

If some record transition outside of T is seen in π or π' then in $\bar{\pi}$ as well. Otherwise, it means the output of some prefix of π is not in I. As the output of that prefix must be a subword of the output of $\bar{\pi}$, and I is downward-closed, we obtain that the output of $\bar{\pi}$ is not in I.

In conclusion, Controller does not win at any point in $\bar{\pi}$ while Environment does. As a consequence, Controller's objective is submixing and thus if Controller wins she can win with a positional strategy. ◀

▶ **Lemma 58.** *If Environment wins an echo game then he has a positional winning strategy.*

Proof. We show that Environment's objective in an echo game has the submixing property, and again apply [17, Theorem 4.5].

Consider two losing plays for Environment π and π', and $\bar{\pi}$ a submixing of the two. We can cut π and π' into $\pi = \pi_0\pi_1\cdots$ and $\pi' = \pi'_0\pi'_1\cdots$ so that $\bar{\pi} = \pi_0\pi'_0\pi_1\cdots$.

At all times in $\bar{\pi}$ if we make a broadcast with letter m while the recent input is w, then that broadcast was made in π or π' with a recent input that is a subword of w. As Environment loses in π and π', and as J_m is upward-closed, $w \in J_m$.

Every record transition seen in $\bar{\pi}$ must be seen in π or π', hence must be in T.

As a consequence, Environment cannot win $\bar{\pi}$, hence Controller wins. The objective of Environment is therefore submixing, and thus if Environment wins he can win with a positional strategy. ◀

E.1 Characterisation of winning strategies

▶ **Lemma 59.** *Controller wins the $(I, (J_m)_{m\in\mathcal{M}})$-invariant game if and only if there is a set of record transitions T such that she wins the $(I, (J_m), T)$-output game and the $(I, (J_m), T, t)$-echo game for all $t \in T$.*

Proof. Suppose Controller wins the $(I, (J_m)_{m\in\mathcal{M}})$-invariant game with a strategy σ. Let T be the set of record transitions taken in a σ-play in which no player has won yet.

We start with the $(I, (J_m), T)$-output game : let Controller apply the same strategy σ in that game.

▷ Claim 60. Let π be a play such that no record transition has been seen yet.

Then π is winning for a player in the invariant game if and only if it is winning for that player in the output game.

Proof. Take a look at the winning conditions in the invariant game. Condition A and C are the same as 1 and 2. Condition B and D cannot happen: as we have not seen any record transition, $\texttt{reg}(\pi') = \{1\}$ for all prefixes π' of π. ◁

As a consequence, a σ-play can only be winning for Environment if we reach a record transition $t \notin T$ while Controller has not won. However, this means that the play obtained before reaching t is not winning for Controller in the invariant game either, by the previous claim. This contradicts the definition of T. Hence σ is winning for Controller in the $(I, (J_m), T)$-output game.

Let $t \in T$. We now show that we have a winning strategy for Controller in the $(I, (J_m), T, t)$-echo game.

▷ Claim 61. Let $t\pi_+$ be a play in the $(I, (J_m), T, t)$-echo game such that no record transition has been seen yet (apart from the first step). Let π_-t be a play ending with t in the $(I, (J_m))$-invariant game such that no player wins in it.

Then π_+ is winning for a player in the echo game if and only if $\pi_-t\pi_+$ is winning for that player in the output game.

Proof. First of all note that $\texttt{reg}(\pi_-t) = \emptyset$ as t updates the only register. As π_-t is not winning for either player, no prefix of it fulfils either A or C. We can then conclude that there is no play starting with π_-t that fulfils either of those conditions.

Furthermore, since no player wins in π_- and t updates the only register, conditions B, D are satisfied by $\pi_-t\pi_+$ if and only if they are satisfied by $t\pi_+$ if and only if $t\pi_+$ satisfies 1, 2 respectively.

This proves the claim. ◁

By definition of t there exists a play reaching t in the invariant game in which no player has won yet. Let π_- be the prefix of that play before reaching t. We define the strategy σ_E as $\sigma_E(\pi) = \sigma(\pi_- \pi)$.

Let us consider a σ_E-play $t\pi_+$ in the echo game and show that it cannot be winning for Environment.

By the claim above, a σ_E-play can only be winning for Environment if we reach a record transition $t' \notin T$ while Controller has not won. However, this means that the play π obtained before reaching t' is such that $\pi_-\pi$ is not winning for Controller in the invariant game either, by the previous claim. This contradicts the definition of T.

We have established that a winning strategy in the $(I, (J_m)_{m\in\mathcal{M}})$-invariant game yields a set of record transitions T and winning strategies in the $(I, (J_m), T)$-output game and the $(I, (J_m), T, t)$-echo game for all $t \in T$.

For the reverse direction, let us consider a set of record transitions T, σ_O a winning strategy in the $(I, (J_m), T)$-output game and, for all $t \in T$, σ_t a winning strategy in the $(I, (J_m), T, t)$-echo game.

We define a strategy σ in the invariant game as follows: If π does not contain any record transition then $\sigma(\pi) = \sigma_O(\pi)$. Otherwise, let π' be the largest suffix of π with no record transition and t the record transition just before π'. We set $\sigma(\pi) = \sigma_t(t\pi')$.

It remains to show that σ is a winning strategy in the invariant game. Suppose by contradiction that there exists a finite σ-play winning for Environment. Let π be such a σ-play of minimal size. If π contains no record transition then by the first claim it is also winning for Environment in the output game. As σ mimics σ_O while no record transition has been seen, this is a contradiction with the fact that σ_O is winning.

On the other hand, if π contains a record transition, then we can decompose it as $\pi = \pi_- t \pi_+$ with t a record transition and π_+ the maximal suffix of π with no record transition.

Then by minimality of π, no player wins in π_-. As a result, by the second claim, $t\pi_+$ is winning for Environment in the $(I, (J_m), T, t)$-echo game. This is a contradiction as by definition of π, $t\pi_+$ is a σ_t-play, and σ_t is winning for Controller.

This concludes our proof. ◀

▶ **Lemma 62.** *If there exists a winning control strategy for a BGR then there exist I such that every word in the basis of I^c is of length $\leq |\mathcal{R}|(|\mathcal{M}|+1)$ and $(J_m)_{m\in\mathcal{M}}$ in which every word in the basis has length $\leq |\mathcal{R}|$ for all m and T a set of record transitions such that Controller wins the $I, (J_m), T$-output game and the $I, (J_m), T, t$-echo game for all $t \in T$.*

Proof. Suppose there exists a winning control strategy σ, then we have some I, (J_m) such that Controller wins $I, (J_m), T$-output game and the $I, (J_m), T, t$-echo game for all $t \in T$. We can assume that the sum of the lengths in the basis of the J_m is minimal.

We remove a word w from the basis of J_m. By minimality of J_m the resulting invariant is not sufficient. Hence Environment wins one of the games. Since I has not changed but (J_m) has decreased, Controller still wins the output game. As a consequence, Environment wins the $I, (J_m), T, t$-echo game for some $t \in T$. Let σ_{echo} be a positional winning strategy for Environment in the new instance of that game. There must be a σ_{echo}-play that is losing for him in the previous instance. As we have decreased $\mathcal{L}(I, (J_m))\downarrow$, the only possibility is that there is a play in which we broadcast m while the recent input is not in J_m. As J_m is upward-closed and σ_{echo} is positional, we can cut all cycles from this play: We obtain a σ_{echo}-play whose recent input is not in J_m, of length at most $|\mathcal{R}|$. As a consequence, $|w| \leq |\mathcal{R}|$.

We have shown that all words in the basis of all J_m have length at most $|\mathcal{R}|$. Let us now bound the words in the basis of I^c.

Consider I with a basis of minimal size such that $I, (J_m)$ is a sufficient invariant for σ. We remove a word w from the basis of I^c, thus increasing I. By minimality of I^c, Environment wins one of the games. It cannot be the output game as I has increased and the J_m are the same. If it is an echo game then let σ_{echo} be a positional winning strategy for Environment in the new instance of that game. There must be a play whose recent input was previously out of $\mathcal{L}(I, (J_m))\downarrow$ but is now in it. We can once again cut cycles on that play. Once we do so, we obtain a play of length $\leq |\mathcal{R}|$ whose recent input w_{in} is in $\mathcal{L}(I, (J_m))\downarrow$ but was not previously. As a consequence, there exists $\mathsf{dec} = (v_0, m_1, \dots, v_k)$ such that $w_{in} \in \mathcal{L}_{\mathsf{dec}}\downarrow$ and for all i, $\mathcal{L}_{(v_0, m_1, \dots, v_{i-1})} \cap J_{m_i} \neq \emptyset$. As we have bounded the lengths of words in the basis of each J_m by $|\mathcal{R}|$, there exist $u_1, \dots, u_k$, all of size at most $|\mathcal{R}|$, such that $u_i \in J_{m_i}$ and $u_i \in \mathcal{L}_{(v_0, m_1, \dots, v_{i-1})}\downarrow$.

For each u_i and for w_{in} at most $|\mathcal{R}|$ letters from $v_0, \cdots, v_k$ suffice to maintain these properties. We define $v'_0, \dots, v'_k$ as the words obtained by removing all other letters. Let $\mathsf{dec}' = (v'_0, m_1, \dots, v'_k)$. We therefore have $|v'_0 \cdots v'_k| \leq |\mathcal{R}|(|\mathcal{M}| + 1)$, and $(v'_0, m_1, \dots, v'_k) \in \mathcal{D}(I, (J_m))$ and $w_{in} \in \mathcal{L}_{\mathsf{dec}'}\downarrow$.

As a consequence, we must have that $v'_0 \cdots v'_k$ is in I but was previously not. As a result, $w \sqsubseteq v'_0 \cdots v'_k$ and thus $|w| \leq |\mathcal{R}|(|\mathcal{M}| + 1)$. ◀

▶ **Theorem 33.** *SAFESTRAT is NEXPTIME-complete for 1BGR.*

Proof. We guess a positional strategy σ for Controller in the output game. We also guess a set of record transitions T, a set of words B of length $\leq |\mathcal{R}|(|\mathcal{M}| + 1)$ and a family of sets of words $(B_m)_{m \in \mathcal{M}}$, where all words have length at most $|\mathcal{R}|$.

We then try to check if Environment has a winning strategy in one of the games. For the output game, we enumerate all positional strategies for Controller. As the size of words in the basis of I is bounded by $|\mathcal{R}|(|\mathcal{M}| + 1)$, if such a strategy allows a losing play, it allows one of length at most $|\mathcal{R}|(|\mathcal{R}| + 1)(|\mathcal{M}| + 1)$. As a consequence, we can check in exponential time whether one of those strategies is winning.

For the echo games, we enumerate all positional strategies for Environment.

▷ Claim 63. We can check that a positional strategy σ_{echo} is not winning for Environment in an echo game in non-deterministic exponential time.

Proof. Let π be a play won by Controller, at all times if we make a broadcast with letter m, the recent input on 1 is in J_m. For each m broadcast in the play, we can select a sequence of at most $|\mathcal{R}|$ preceding receptions forming an element of the basis of J_m. Those elements witness the fact that the recent input is in J_m.

We can thus easily construct an NFA of exponential size recognising finite plays in which Environment does not win.

Since σ_{echo} is positional, there is an automaton with $|\mathcal{R}|$ states recognising the set of σ_{echo}-plays. We first check whether there is an infinite word whose prefixes are all accepted by the NFA. If not, we check whether the NFA accepts a play ending with a transition of T.

Finally, we project it to obtain an NFA $\mathcal{A}$ recognising the recent inputs of σ_{echo}-plays not won by Environment. We also build an exponential-size NFA $\mathcal{B}$ recognising $\mathcal{L}(I, (J_m)_{m \in \mathcal{M}})$. As shown in [1], if there is a word in $\mathcal{L}(\mathcal{A})\downarrow$ that is not in $\mathcal{L}(\mathcal{B})\downarrow$, then there is one of polynomial size in $|\mathcal{A}|$ and $|\mathcal{B}|$.

As a consequence, we can check that non-inclusion in non-deterministic exponential time.

In sum, we can check in non-deterministic exponential time that the given strategy is not winning for Environment. ◁

This lets us decide in non-deterministic exponential time if there exist $I, (J_m), T$ such that Controller wins the output game and all the echo games. As a result, SafeStrat is in NExpTime. All that is left to do is show the matching lower bound, which is done below. ◀

E.2 NExpTime-hardness of SafeStrat for 1BGR

The *exponential grid tiling problem* asks, given a set of *colours* C, a number N *in unary* and a set of *tiles* $T \subseteq C^{\{\mathtt{up},\mathtt{down},\mathtt{left},\mathtt{right}\}}$, whether there is a *tiling* of the $2^N \times 2^N$ grid, i.e., a function $\tau : [0, 2^N - 1] \times [0, 2^N - 1] \to T$ such that for all $x, y, x', y' \in [0, 2^N - 1]$,

- if $x = x'$ and $y = y' + 1$ then $\tau(x, y).\mathtt{down} = \tau(x, y).\mathtt{up}$
- if $x = x' + 1$ and $y = y'$ then $\tau(x, y).\mathtt{left} = \tau(x, y).\mathtt{right}$
- if $x = 0$ (resp. $x = 2^N - 1$, $y = 0$, $y = 2^N - 1$) then $\tau(x, y).\mathtt{left} = c_{border}$ (resp. $\mathtt{right}$,$\mathtt{down}$, $\mathtt{up}$)

This problem is NExpTime-complete [27].

▶ **Lemma 64.** *SafeStrat is NExpTime-hard on 1BGR.*

Proof. We reduce from the exponential grid tiling problem.

Let C be a set of colours containing a border colour B, let $T = \{t_1, \dots, t_k\}$ be a set of tiles and N an integer in unary. We use the alphabet of letters $\mathcal{M} = \{0, 1, \bar{0}, \bar{1}\} \cup T \cup \bar{T}$, where $\bar{T} = \{\bar{t}_1, \dots, \bar{t}_k\}$ is a copy of T.

We design a 1BGR in which Controller wins if and only if there is a valid tiling of the $2^N \times 2^N$ grid with those tiles.

Essentially, Environment may use some agents to broadcast coordinates (x, y) and $(\bar{x}, \bar{y})$ in the grid, respectively using letters $\{0, 1\}$ and $\{\bar{0}, \bar{1}\}$. Environment can also make an agent receive coordinates (x, y) (resp. $(\bar{x}, \bar{y})$), while checking that they all have the same datum. He then makes Controller choose a tile t (resp. $\bar{t}$), which is broadcast with that same identifier. A strategy for Controller amounts to two functions $\tau, \bar{\tau} : [0, 2^N - 1] \times [0, 2^N - 1] \to T$.

The agents that broadcast coordinates (x, y) and $(\bar{x}, \bar{y})$ can then receive tiles t and $\bar{t}$ with their own identifier, and check that:

- If $x = \bar{x}$ and $y = \bar{y}$ then $t = \bar{t}$
- If $x + 1 = \bar{x}$ and $y = \bar{y}$ then $t.right = \bar{t}.left$
- If $x = \bar{x}$ and $y + 1 = \bar{y}$ then $t.up = \bar{t}.down$
- If $x = 0$ (resp. $y = 0$, $x = 2^N - 1$, $y = 2^N - 1$) then $t.left = B$ (resp. *down*, *left*, *right*).

The first item forces Controller to choose $\tau = \bar{\tau}$. The other items make sure that she picks a valid tiling of the grid.

From the initial state Environment chooses between three modes:

- He can receive a sequence of $2N$ bits in $\{0, 1\}$ with the same datum and then let Controller broadcast a letter of T with that same identifier.
- He can receive a sequence of $2N$ bits in $\{\bar{0}, \bar{1}\}$ with the same datum and then let Controller broadcast a letter of $\bar{T}$ with that same identifier.
- He can broadcast a sequence of letters of the form $x_1\bar{x}_1 \cdots x_N\bar{x}_N y_1\bar{y}_1 \cdots y_N\bar{y}_N$ with $x_1, y_1, \dots, x_N, y_N \in \{0, 1\}$, all with his initial datum. He then receives one letter t' of T and one letter $\bar{t}$ of $\bar{T}$ with his initial datum. If $t = t'$ then he stops, otherwise he broadcasts m_{err}.

- He can broadcast a sequence of letters of the form $x_1\bar{x}_1 \cdots x_N\bar{x}_N y'_1\bar{y}_1 \cdots y'_N\bar{y}_N$ with $x_1, y_1, y'_1, \ldots, x_N, y_N, y'_N \in \{0,1\}$, all with his initial datum. He makes sure that $\langle y_1 \cdots y_N\rangle_2 = \langle y'_1 \cdots y'_N\rangle_2 + 1$. He then receives one letter t' of T and one letter $\bar{t}$ of $\bar{T}$ with his initial datum. If $up(t') \neq down(t)$ or $\langle y'_1 \cdots y'_N\rangle_2 = 0$ and $down(t') \neq B$ or $\langle y_1 \cdots y_N\rangle_2 = 2^N - 1$ and $up(t) \neq B$, he broadcasts m_{err}. Otherwise he stops broadcasting m_{err}.
- Similarly, he can broadcast a sequence of letters of the form $x'_1\bar{x}_1 \cdots x'_N\bar{x}_N y_1\bar{y}_1 \cdots y_N\bar{y}_N$ with $x_1, x'_1, y_1, \ldots, x_N, x'_N, y_N \in \{0,1\}$, all with his initial datum. He makes sure that $\langle x_1 \cdots x_N\rangle_2 = \langle x'_1 \cdots x'_N\rangle_2 + 1$. He then receives one letter t' of T and one letter $\bar{t}$ of $\bar{T}$ with his initial datum. If $right(t') \neq left(t)$ or $\langle x'_1 \cdots x'_N\rangle_2 = 0$ and $left(t') \neq B$ or $\langle x_1 \cdots x_N\rangle_2 = 2^N - 1$ and $right(t) \neq B$, he broadcasts m_{err}. Otherwise he stops without broadcasting m_{err}.

If there is a valid tiling, Controller can play the corresponding strategy. In order to broadcast m_{err}, Environment must make an agent a broadcast coordinates with its initial datum, and then receive two tiles that do not satisfy the conditions mentioned above. The agents that send those tiles must receive exactly $2N$ letters from a, as they are signed by its initial datum. Thus their broadcasts are the tiles of the valid tiling at those coordinates, and the agent will not be able to broadcast m_{err}, as they match all the conditions.

If there is no valid tiling, Controller's strategy will either induce two different tilings or two identical invalid ones. In both cases Environment can detect the mistake by making an agent a broadcast the coordinates corresponding to the mistake, making two agents answer with the faulty tiles, and make a broadcast m_{err} by observing the mistake. ◀