Optimally Controlling a Random Population

Hugo Gimbert
CNRS, LaBRI, Université de Bordeaux, Talence, France

Corto Mascle
Max Planck Institute for Software Systems, Kaiserslautern, Germany

Patrick Totzke
University of Liverpool, UK

Abstract

The population control problem is a parameterised problem where a controller sends messages to a whole population of identical finite-state agents, aiming to eventually move them all into a target state. The decision problem asks whether this can be achieved for *arbitrarily large* finite populations. We focus on the *randomised* version of this problem, where every agent is a copy of the same finite Markov Decision Process and non-determinism in the global action chosen by the controller is resolved independently and uniformly at random. Colcombet, Fijalkow and Ohlmann [6] showed that this problem is decidable, but without any complexity upper bound. We show that the random population control problem is in fact ExpTime-complete.

2012 ACM Subject Classification Theory of computation → Formal languages and automata theory

Keywords and phrases Controller synthesis, Parameterized verification

Digital Object Identifier 10.4230/LIPIcs.ICALP.2026.180

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Related Version *Full Version*: `https://arxiv.org/abs/2411.15181`

Funding *Patrick Totzke*: EPSRC, grant no.: EP/X042596/1.

1 Introduction

We place N tokens on the initial state of a non-deterministic finite automaton and update their positions in rounds, in each of which a Controller selects a letter a and then every token moves along an a-labelled edge out of its current state. The goal for the Controller is to eventually synchronise all tokens, gathering them into accepting states at the same time. Naturally, increasing N can only make this task more difficult. The decision question we study asks whether, for a given automaton, can the Controller succeed for all values of N?

This population control problem has been studied both in the adversarial and random settings, which differ in how agents resolve choices and what guarantees the Controller is after. In the adversarial setting, an antagonistic environment resolves all agents' choices, trying to frustrate Controller and avoid synchronisation. Bertrand et al. [3, 4] showed that the adversarial population control problem is decidable and ExpTime-complete.

In the random setting, all agents' choices are made uniformly at random and the Controller aims to synchronise the agents almost surely, with probability one.

▶ **Example 1.** Controller loses the adversarial population control game from Figure 1. For example, the opponent can win by keeping all tokens in the initial state in every round.

However, Controller does have a winning strategy for the corresponding random population control game: Regardless of the size N of the population, she plays the action a until exactly one token is in s_1, which happens eventually with probability 1. She then plays b followed by either a (if the isolated token moved to s_2) or b (if the token moved to s_3) to send that

© Hugo Gimbert, Corto Mascle, and Patrick Totzke;
licensed under Creative Commons License CC-BY 4.0

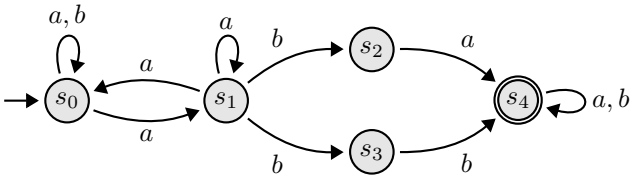

Figure 1 An automaton for which Controller wins against a random environment but not an adversarial one. The initial and target states are s_0 and s_4, respectively. Not shown, but implicitly present is a sink state, to which all states move on actions not shown (b from s_2 and a from s_3).

token to the final state. She can repeat this procedure until all tokens are in the target. Note that if instead she plays b while more than one token is in s_1 then some of them may spread simultaneously to s_2 and s_3 and she is stuck: whatever action she plays, at least one token will move to the (invisible) sink state and cannot be recovered.

The random population control problem is ExpTime-hard [14] and decidable [6]. Colcombet et al.'s decision procedure is based on two ingredients [6]:

- First, winning regions are downward-closed with respect to the natural product order on $\mathbb{N}^S$ and can therefore be finitely represented and manipulated as a union of ideals.
- The second ingredient is solving the *sequential flow problem*, which asks whether an unbounded number of tokens can be moved from one set of states to another, while respecting capacity constraints on the number of tokens occupying each transition. They show that one can compute the full corresponding pre-set of a given downward-closed set of target configurations, by a reduction to the boundedness problem of nested distance-desert automata [13].

Ultimately, this allows one to compute a representation of the winning region by iterative refinement. The main limitation of this approach is that the bounds on the representation of intermediate sets, and termination of the resulting algorithm rely on well-quasi-orders and therefore only provide a non-elementary upper bound.

Our Contribution. We show that the random population control problem is ExpTime-complete. As in [6], our upper bound relies on symbolic representations. However, instead of computing the whole winning region, we show that one can witness positive instances already with a potentially smaller, more compactly representable subset (Theorem 7). Computing this in exponential time requires solving what we call the Dynamic flow problem, which asks whether we can transfer arbitrarily many tokens, with positive probability, from an initial configuration to the set of final configurations while staying in a given set of safe configurations. It is closely related to the sequential flow problem defined in [6]. For this we rely on a recent result showing that a closely-related problem is decidable in PSPACE [10], which we adapt to our setting.

We also include the original, previously unpublished lower bound [14] (Theorem 21).

Related work. Models for biological, chemical, or computational systems with large crowds of simple finite-state components include Petri nets [16], population protocols [2], or chemical reaction networks [21].

Population control problems provide a formal framework for the design of strategies to control a large number of identical agents. This formalisation was introduced in [4], as a model for the synchronization of large populations of yeasts [23]. It fits into parameterized

verification, a line of research that aims to verify distributed protocols over arbitrarily large networks [20, 8]. While we take a discrete approach, in the sense that we have an integral number of agents, other models have been studied where the number is considered so large that it can be abstracted by a continuous probability distribution [1, 7].

Another model close to our setting is explorable automata [11], which generalise both (adversarial) population control problems and history-deterministic automata on infinite words. Closer to our setting, other models of games parameterized by the number of agents exist, typically aiming to synthesize distributed protocols for arbitrarily large networks [5, 22]. Finally, recent works bridge the adversarial setting for population control to the modern study of *history-deterministic* automata: They introduce explorability games, played on automata on infinite words. A player gives an input letter by letter, while the other player has to move tokens in the state space of the automaton so that one of them finds an accepting run whenever the input word is in the language of the automaton [11].

We omit part of the proofs due to space constraints; missing proofs can be found in the full version of the paper.

2 Preliminaries

We assume familiarity with automata theory [18] and Markov Decision Processes [17], and proceed to recall some necessary notations.

Markov decision processes. A *Markov decision process* (MDP) $\mathcal{M} = (S, \Sigma, \Delta)$ consists of a set S of states, Σ a set of actions, and a transition function $\Delta : S \times \Sigma \to \mathcal{D}ist(S)$, where $\mathcal{D}ist(S)$ denotes the set of probability distributions over S.

We consider almost-sure reachability objectives, given by a set $F \subseteq S$ of target states that Controller aims to visit. A *strategy* is a function $\sigma : S \to \mathcal{D}ist(\Sigma)$. Fixing such a strategy σ and an initial state $s \in S$ results in a Markov Chain with probability measure $\mathbb{P}_{\sigma,s}$ (see [17] for details). We call σ *winning* from state s if F is reached $\mathbb{P}_{\sigma,s}$-almost surely.

The *winning region* is the subset $W \subseteq S$ of states from which a winning strategy exists.

Random walks in winning regions. In an MDP with finitely many states, if there is an almost surely winning strategy for Controller, then there is a canonical one: play at random any action that guarantees staying in the winning region. This is formalized using *arenas* and *safe random walks*, as follows.

We call an MDP *simple* if for all $s \in S$, the set of states reachable from s is finite. A *commit* is an element of $S \times \Sigma$. An *arena* is a set $\Gamma \subseteq S \times \Sigma$ of commits such that for all $(s, a) \in \Gamma$ and $t \in S$, if $\Delta(s, a)(t) > 0$ then there exists $b \in \Sigma$ such that $(t, b) \in \Gamma$. For brevity we will write $s \in \Gamma$ instead of $\exists b \in \Sigma, (s, b) \in \Gamma$. A *path* in arena Γ is a sequence $s_0 \xrightarrow{a_1} s_1 \xrightarrow{a_2} \dots \xrightarrow{a_k} s_k \in S(\Sigma S)^*$ such that $(s_{j-1}, a_j) \in \Gamma$ and $\Delta(s_{j-1}, a_j)(s_j) > 0$ for all $j > 0$. A *safe random walk* in the arena Γ is a strategy σ defined such that $\sigma(s)$ is the uniform distribution on $\{a \in \Sigma \mid (s, a) \in \Gamma\}$. An arena is *winning* (with respect to F) if from every $s \in \Gamma$ there is a path in Γ to F. This is closely linked with winning regions, as shown in the next lemma.

▶ **Lemma 2** (Winning arenas). *Given a simple MDP and a set of target states F,*

1. *A union of winning arenas is a winning arena, and the winning region is the projection onto S of the largest winning arena.*
2. *In a winning arena, a safe random walk is a winning strategy from every state.*

Random Populations. We consider *populations* of agents (or *tokens*), described by a common finite MDP $\mathcal{M} = (S, \Sigma, \Delta)$.

Our arguments require tracking the trajectory of selected (sets of) tokens along paths and the following definitions will be convenient for this purpose.

Write T_∞ for the countably infinite set of tokens. For any finite subset $T \subseteq T_\infty$, let $\mathcal{M}^T = (S^T, \Sigma, \Delta^T)$ denote the MDP whose states (called *configurations*) are functions mapping each token of T to a state of S, and Δ is naturally lifted to S^T: $\Delta^T(\gamma, a)(\gamma') = \prod_{t \in T} \Delta(\gamma(t), a)(\gamma'(t))$ for all $\gamma, \gamma' \in S^T$ and $a \in \Sigma$. For a subset $R \subseteq S$, R^T contains those configurations where all tokens are mapped to states in R. In particular, $\{s\}^T$ contains only the configuration mapping all tokens to $s \in S$, and F^T is the set of final configurations, mapping all tokens to some state of F.

We study the following decision problem, which asks whether for every finite population T, Controller can almost surely ensure that all tokens simultaneously[1] reach a final state.

The Random population control problem
Given: an MDP $\mathcal{M} = (S, \Sigma, \Delta)$, initial state $i \in S$ and target set $F \subseteq S$.
Question: Is there a winning strategy to reach F^T from $\{i\}^T$, for all finite $T \subseteq T_\infty$?

We define $\mathcal{M}^* = \bigcup_{T \subseteq_f T_\infty} \mathcal{M}^T$ as the disjoint union of all finite $\mathcal{M}^T$. Notice that, while the state space $S^* = \bigcup_{T \subseteq_f T_\infty} S^T$ is infinite, each S^T is finite and therefore $\mathcal{M}^*$ is simple.

According to Lemma 2, a given MDP $\mathcal{M}$ is a positive instance if, and only if, there is a winning arena $A \subseteq S^* \times \Sigma$ with $\{i\}^T \subseteq A$ for all $T \subseteq_f T_\infty$. Our main decidability result relies on identifying such a witness A. By Lemma 2, the existence of such a witness A does not depend on the precise probabilities in Δ. We will omit them in all examples and just assume without loss of generality that all probability distributions are uniform.

▶ Remark 3. At this point one might wonder why we choose to define configurations as mappings from a set of tokens to states, and not simply as vectors counting the number of tokens in each state. This is due to the nature of our main proof: we use strategies in which tokens are split between a *cohort* of many tokens and small groups of tokens, isolated from each other. This isolation requirement is difficult to formulate without referring to specific token identities, and dropping those identities would make some proofs extremely hard to write (and read), especially for Lemma 16 and Theorem 7.

3 Symbolic Configurations

We first observe that the maximal winning arena W of $\mathcal{M}^{(*)}$ must be closed under renaming of tokens. That is, for any bijective $\tau : T \to T'$ and configurations $\gamma \in S^T$ and $\gamma' \in S^{T'}$, if $\gamma = \gamma' \circ \tau$ then either both γ and γ', or neither is in W. Furthermore, W must be closed under removal of tokens: if W contains $\gamma \in S^T$ and $T' \subseteq T$ then W must also contain the restriction of γ to T'. Both these can be seen by simple transfers of winning strategies.

The winning region can therefore conveniently be represented as a downward-closed set of $|S|$-dimensional vectors of integers, each of which just reflects how many tokens occupy each state. Given a configuration $\gamma \in S^T$, we write $|\gamma|$ for the vector of $\mathbb{N}^S$ such that $|\gamma|(s)$ is the number of tokens in s in γ, for all $s \in S$.

[1] Requiring all tokens to visit a final state *simultaneously* or possibly at different times is irrelevant for our purposes: The two variants are logspace inter-reducible (see the full version for a proof).

Take $\overline{\mathbb{N}} = \{0 < 1 < 2 < 3 < \ldots < \omega\}$, the non-negative integers with natural ordering extended by a maximal element ω. This ordering is lifted pointwise to $\overline{\mathbb{N}}^S$, whereby $\mathbf{v} \leq \mathbf{v}'$ if and only if $\mathbf{v}(s) \leq \mathbf{v}'(s)$ for all $s \in S$, and $(\mathbf{v}, a) \leq (\mathbf{v}', a')$ if and only if $\mathbf{v} \leq \mathbf{v}'$ and $a = a'$.

▶ **Definition 4** (Symbolic representations). *A* symbolic configuration *is a vector* $\mathbf{v} \in \overline{\mathbb{N}}^S$. *Given a symbolic configuration, the* ideal $\mathbf{v}{\downarrow}$ *is the set of all configurations* γ *such that* $|\gamma| \leq \mathbf{v}$. *For a set of symbolic configurations* $\mathbf{V}$, *we write* $\mathbf{V}{\downarrow}$ *for the set* $\bigcup_{\mathbf{v}\in\mathbf{V}} \mathbf{v}{\downarrow}$.

Similarly, a symbolic commit $(\mathbf{v}, a)$ *is a symbolic configuration* $\mathbf{v} \in \overline{\mathbb{N}}^S$ *together with an action* $a \in \Sigma$. *The* commit ideal $(\mathbf{v}, a){\downarrow}$ *is the set* $\{(\gamma, a) \mid \gamma \in \mathbf{v}{\downarrow}\}$.

We recall that the product ordering on commit ideals is a well-quasi-order. Every downward-closed set $A \subseteq \overline{\mathbb{N}}^S \times \Sigma$, including our maximal winning arena, can therefore be written as a union $A = \bigcup_{0<i<m} (\mathbf{v}_i, a_i){\downarrow}$ of finitely many incomparable ideals [19]. We call such a finite union of commit ideals a *population arena*, or simply an arena when clear from the context. Throughout this work we will make use of the notion of *K-definability*, that bounds the maximal finite constants in such ideal representations of population arenas.

▶ **Definition 5** (K-definability). *Let* $K \in \mathbb{N}$. *A set* Γ *of commits is* K-definable *if it is a finite union of commit ideals of the form* $(\mathbf{v}{\downarrow}, a)$ *with* $\mathbf{v} \in \{0, \ldots, K, \omega\}^S$. *The maximal* K*-definable subset of* Γ *is denoted by* Γ_K.

▶ **Example 6.** In the MDP from Example 1, the maximal winning arena is

$$W = ((\omega, \omega, \omega, 0, \omega), a){\downarrow} \cup ((\omega, 1, 0, \omega, \omega), b){\downarrow}$$

with states enumerated as $(s_0, \ldots, s_4)$. A safe random walk in W instructs to play, with positive probability, the action a in any configuration where no tokens are on s_3. Further, to play b with positive probability if at most one token is on s_1 and none on s_2. This set is clearly 1-definable and therefore, $W = W_1$. Its maximal 0-definable subset is

$$W_0 = ((\omega, \omega, \omega, 0, \omega), a){\downarrow} \cup ((\omega, 0, 0, \omega, \omega), b){\downarrow}$$

While W_0 is still an arena (it is a forward invariant), it is not winning because there is no path in it from initial to target configurations. This is because the action b that takes tokens to s_2 or s_3 is never played when there are tokens on s_1.

4 Solving the Random Population Control Problem

We present our main result: the Random population control problem is solvable in exponential time. According to Lemma 2, there is a maximal winning arena containing all configurations from which there is a winning strategy. As a consequence, there is a winning strategy from all initial configurations if and only if there is a winning arena which contains them all.

Thus, all positive instances are witnessed by a set of commits that is 1) an arena (forward-closed), 2) winning (every included configuration can reach a target configuration without leaving this set), and finally, 3) includes all initial configurations.

We will symbolically represent candidate sets as per Definition 4, as finite unions of commit ideals $Y = \bigcup_{0<i<m} (\mathbf{v}_i, a_i){\downarrow}$. This makes it straightforward to check conditions 1) and 3) syntactically. The two main remaining ingredients for our approach are Theorem 7, that positive instances admit polynomially-definable winning arenas; and Theorem 8, that one can check condition 2) in exponential time (and even polynomial space). We formally state these theorems below and derive an algorithm to solve the Random population control problem in exponential time. The following sections contain proofs.

In what follows, fix an MDP $\mathcal{M} = (S, \Sigma, \Delta)$, initial state $i \in S$ and target set $F \subseteq S$. Let W denote the winning arena in $\mathcal{M}^{(*)}$, and recall that for every $K \in \mathbb{N}$, W_K is its maximal K-definable subset. Let $\mathbf{i}, \mathbf{f} \in \overline{\mathbb{N}}^S$ be the symbolic configurations mapping the state i (resp. all states in F) to ω and other states to 0. Then in particular, $\mathbf{f}{\downarrow}$ is the set of configurations whose tokens are all in F. Observe that, since all those configurations are in the winning region by definition, we have $\mathbf{f}{\downarrow} \subseteq W_0$ and the answer to the Random population control problem is positive if and only if W_0 contains all configurations with all tokens on the initial state, i.e., $\mathbf{i}{\downarrow} \subseteq W_0$.

▶ **Theorem 7.** *Let W be the maximal winning arena. There is a winning arena $Y \subseteq W$ with*
- *Y contains W_0; and*
- *Y is $|S|$-definable.*

▶ Remark. Theorem 7 implies that, if we can control arbitrarily many tokens then we can do so with a safe random walk in some $|S|$-definable arena. This has no immediate consequences for the shape of the winning region W. In fact, W may not be $|S|$-definable and its ideal representation may require doubly exponentially large constants (cf. Section 7.2). Therefore, in general, Y is strictly contained in W.

Moreover, despite there existing memoryless and deterministic strategies from every configuration in the winning region [15], the $|S|$-definable winning strategies guaranteed by Theorem 7 may still require randomisation (cf. Section 7.3).

To verify that a candidate arena is winning, we need to be able to check that all configurations in it can reach a final one while staying in the arena. We show (in Section 6), how to solve the following problem[2].

The Dynamic flow problem
Given: an MDP $\mathcal{M} = (S, \Sigma, \Delta)$, a finite set of symbolic commits $\mathbf{V}$, a symbolic configuration $\mathbf{v}_0 \in \mathbf{V}$ and a set of symbolic configurations $\mathbf{F}$.
Question: Does every configuration in $\mathbf{v}_0{\downarrow}$, admit a path to $\mathbf{F}{\downarrow}$ inside $\mathbf{V}{\downarrow}$?

▶ **Theorem 8.** *The Dynamic flow problem can be solved in polynomial space in $\log(K)$ and $|S|$, where K is the largest integer appearing in the input.*

Together, Theorems 7 and 8 suggest a simple algorithm to compute a suitable winning arena Y by refining a candidate set $\mathbf{V}$ of symbolic commits until the corresponding union of ideals is a winning arena. Recall that given a set of commits Γ we use the notation $\gamma \in \Gamma$ for $\exists b \in \Sigma, (\gamma, b) \in \Gamma$. Similarly, given a set of configurations C and a set of commits Γ, we write $C \subseteq \Gamma$ instead of $\forall \gamma \in C, \exists a \in \Sigma, (\gamma, a) \in \Gamma$.

The idea is to compute a $|S|$-definable winning arena Y as guaranteed by Theorem 7 by refinement. Start with the largest such candidate set (line 1), and reduce it if it is not an arena (lines 3 and 4) or if it is not a winning one (lines 5 and 6). Computing ideal representations for reduced candidate sets, and the check in line 3, are syntactic and therefore simple. The check in line 5 is an instance of the Dynamic flow problem. Once the procedure stabilizes, $\mathbf{V}{\downarrow}$ is a winning arena that satisfies the conditions of Theorem 7. In particular, it contains all of Γ_0. The input therefore describes a positive instance if and only if $\mathbf{i} \in \mathbf{V}$, the output of the algorithm in line 8.

[2] This is a slightly different presentation of the "sequential flow problem" in [6]. They presented a solution in exponential space, which was recently improved to polynomial space [10].

Algorithm 1 Algorithm for the Random population control problem.

1: $\mathbf{V} \leftarrow \{0, \dots, |S|, \omega\}^S \times \Sigma$ ▷ start with maximal $|S|$-definable candidate
2: **repeat**
3: **if** $\exists (\mathbf{v}, a) \in \mathbf{V}, \gamma_1 \in \mathbf{v}{\downarrow}, \gamma_2 \notin \mathbf{V}{\downarrow}$ s.t. $\Delta(\gamma_1, a)(\gamma_2) > 0$ **then**
4: $\mathbf{V} \leftarrow \mathbf{V} \setminus \{(\mathbf{v}, a)\}$ ▷ reduce if not an arena
5: **if** $\exists (\mathbf{v}, a) \in \mathbf{V}, \gamma \in \mathbf{v}{\downarrow}$ s.t. there is no path from γ to $\mathbf{f}{\downarrow}$ in $\mathbf{V}{\downarrow}$ **then**
6: $\mathbf{V} \leftarrow \mathbf{V} \setminus \{(\mathbf{v}, a)\}$ ▷ reduce if not a winning arena
7: **until** $\mathbf{V}$ does not change
8: **return** $\mathbf{i} \in \mathbf{V}$

Observe that $\mathbf{V}$ is deliberately initialised as the set of *all* symbolic configurations that may appear in the decomposition of Y, and therefore only reduces during the computation.

▶ **Theorem 9.** *The Random population control problem is ExpTime-complete.*

Proof. ExpTime-hardness is shown as Theorem 21. The matching upper bound is provided by Algorithm 1: It remains to argue that it takes at most exponential time.

- $\mathbf{V}$ has $(|S|+2)^{|S|}|\Sigma|$ elements at the start, and every iteration removes at least one.
- The first condition (line 3) can be checked simply by computing the set of successors of each symbolic commit in $\mathbf{V}$. This can be straightforwardly done in exponential time in the number of states.
- Checking the second condition (line 5) is solving an instance of the Dynamic flow problem with largest constant $\leq |S|$, for each $(\mathbf{v}, a)$ in $\mathbf{V}$. This problem is solvable in polynomial space (and thus exponential time) in $|S|$ (Section 6).

Therefore, Algorithm 1 takes exponential time in the size of the input. ◀

5 Theorem 6: Small Winning Arenas

Consider now a positive instance of the Random population control problem with maximal winning arena W.

Towards proving Theorem 7, we now describe a winning strategy for Controller that remains in (and thus defines) a suitable winning sub-arena $Y \subseteq W$ which satisfies the claim of the theorem. As discussed previously, the inclusion $Y \subseteq W$ might be strict.

We consider the tokens of any configuration to be split between those that are part of the large *cohort* and a few *individuals* that are tracked separately. Increasing the number of tokens in states occupied by the cohort cannot result in configurations outside the winning region (see Definition 15). By contrast, increasing the number of individuals on a state may turn a winning configuration into a losing one.

The strategy operates in two modes. The first mode attempts to follow a "lucky" path in W that leads to a final configuration: one where at any step only a few tokens leave the cohort and those that do, subsequently never meet until they rejoin the cohort. For instance, in Example 1 we would try sending tokens one by one via $s_0 \to s_1 \to s_2 \to s_4$, while keeping all others in their current states. Formally, this means that the path stays within W_1, and therefore introduces at most $|S|$ individual tokens. The existence of such paths is guaranteed by Lemma 11. Successfully following such a path to the end has a positive probability that is bounded away from zero (for a given number of tokens).

In case the "lucky" path is exited prematurely, our strategy enters a second, recovery mode that, almost surely, brings the individual tokens back into the large cohort (reaches W_0) in order to attempt a new "lucky" path to a final configuration again. See Figure 2 for an illustration.

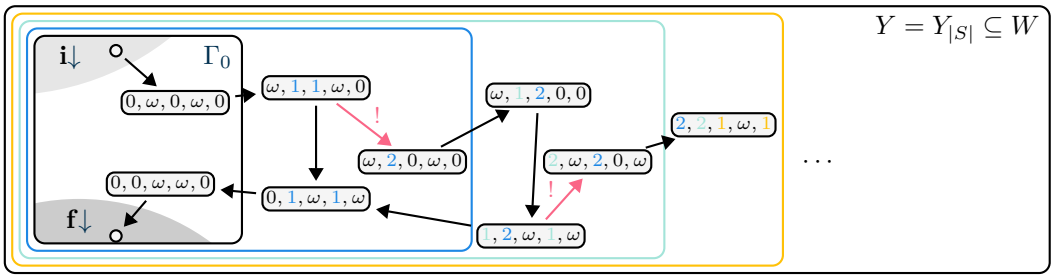

Figure 2 Controller tries to follow the black path from $\mathbf{i}\downarrow$ to $\mathbf{F}\downarrow$, along which individual tokens (in blue) may be produced temporarily yet never meet in the same state. If this path is exited (red arrow) those individuals can meet, in which case we leave W_1 as here. We then attempt to recover by following a new black path towards W_0 which can spawn new individual tokens (as here, in blue-green) and so on. Lemma 16 guarantees that tokens of different colors never meet and therefore that there are no more than $|S|$ layers. We define Y as the union of all those layers.

This must be possible because we have not left the maximal winning arena W, and therefore can still almost surely reach a final configuration in $\mathbf{F}\downarrow \subseteq W_0$. We again attempt to follow another "lucky" path towards this new target, using Lemma 11, thereby creating up to $|S|$ additional individuals along the way. Again, following this path succeeds with probability bounded away from zero.

The issue with this approach is that we may continue to be unlucky and track ever more individual tokens. However, as we show in Lemma 16, one can play as described above while ensuring that *individuals introduced at different steps never meet* before they are brought back to the cohort. As each step produces at most $|S|$ new individual tokens (necessarily on distinct states), we can see at most $|S|$ individual tokens in each state overall.

We will need some notation to formalize our argument. In particular, we extend all previous definitions to (symbolic) configurations that explicitly track finitely many individual tokens. We thus extend the symbolic representations in Definition 4 accordingly.

▶ **Definition 10.** *Let T_f be a finite set of tokens, called individuals, whereas other tokens in $T_\infty \setminus T_f$ are called the cohort tokens. Given a symbolic configuration $\mathbf{v} \in \overline{\mathbb{N}}^S$ and a configuration $\gamma_f : T_f \to S$, the ideal tracking T_f generated by $\mathbf{v}$ and γ_f is written $(\gamma_f \otimes \mathbf{v})\downarrow$ and defined as the set of configurations $\gamma : T \to S$ such that:*

- *individuals are placed on states according to γ_f, i.e. $T_f \subseteq T$ and $\forall t \in T_f, \gamma(t) = \gamma_f(t)$.*
- *the number of cohort tokens on a state is bounded from above by $\mathbf{v}$, i.e.*

$$\forall s \in S, |\{t \in T \setminus T_f, \gamma(t) = s\}| \leq \mathbf{v}(s).$$

Similarly, given an action $a \in \Sigma$, the commit ideal tracking T_f associated with $\mathbf{v}$, γ_f and a is denoted $(\gamma_f \otimes \mathbf{v}, a)\downarrow$, and is the set of commits (γ, a) with $\gamma \in (\gamma_f \otimes \mathbf{v})\downarrow$. A population arena tracking T_f is a finite union of commit ideals tracking T_f.

Let Γ be an arena and $K \in \mathbb{N}$. Then Γ_{K,T_f} denotes the union of all ideals tracking T_f of the form $(\gamma_f \otimes \mathbf{v})\downarrow$ with $\mathbf{v} \in \{0, \ldots, K, \omega\}^S$ that are included in Γ.

5.1 The Existence of Lucky Paths

We start from a fairly simple idea: Suppose that we are able to transfer arbitrarily many tokens from one state to another along paths that remain in the winning region. It may be necessary to transfer those tokens in small groups, as they have to go through a bottleneck. In this case, we might as well transfer them one by one.

▶ **Lemma 11.** *Let T_f be a finite set of tokens, $\mathbf{F}$ an ideal tracking T_f and Γ a population arena tracking T_f. If Γ is a winning arena with respect to $\mathbf{F}$, then for every configuration $\gamma_0 \in \Gamma_{0,T_f}$ there is a path in Γ_{1,T_f} from γ_0 to $\mathbf{F}$.*

The rest of Section 5.1 is dedicated to the proof of Lemma 11. Let us start by defining the necessary terminology on flows and cuts. Let $\overline{\mathbb{R}^+}$ be the set of non-negative real numbers, with an additional maximum element ω. We extend the addition by setting $\omega + r = r + \omega$ for all $r \in \overline{\mathbb{R}^+}$.

Given a finite directed graph $G = (V, E)$, a capacity function $c : V \to \overline{\mathbb{R}^+}$, and two vertices *src* and *tgt* (a source and a target), we define a *flow* from *src* to *tgt* as follows. It is a function $f : E \to \overline{\mathbb{R}^+}$ such that for all $v \in V \setminus \{src, tgt\}$, we have

$$\sum_{(v_-,v)\in E} f(v_-, v) = \sum_{(v,v_+)\in E} f(v, v_+) \leq c(v).$$

The *value* of the flow is defined as $\mathbf{val}_{G,c}(f) = \sum_{(src,v)\in E} f(src, v)$.

A *cut* is a set M of vertices such that every path from *src* to *tgt* contains a vertex of M. Its *weight* is defined as $\mathbf{weight}_{G,c}(M) = \sum_{v\in M} c(v)$.

▶ **Theorem 12** (Max flow-min cut [9])**.** *For every graph $G = (V, E)$, capacity function $c : V \to \overline{\mathbb{R}^+}$ and vertices $src, tgt \in V$,*

$$\max_{f\, flow} \mathbf{val}_{G,c}(f) = \min_{M\, cut} \mathbf{weight}_{G,c}(M)$$

▶ Remark 13. We state this theorem with the capacities on the vertices, as it is convenient for the next proof. It is more commonly stated with capacities on the edges $c : E \to \overline{\mathbb{R}^+}$. The constraints on the flow is then $\sum_{(v_-,v)\in E} f(v_-, v) = \sum_{(v_-,v)\in E} f(v_-, v)$ for all $v \in V$ and $f(e) \leq c(e)$ for all $e \in E$. A cut is then defined as a set of edges, and the theorem is stated analogously.

Another classic result is the *integer flow theorem*. It says that if all capacities in the graph are integers, then there is an integer maximal flow $f : E \to \overline{\mathbb{N}}$. This is a by-product of the Ford-Fulkerson algorithm.

▶ **Theorem 14** (Integer flow theorem [9])**.** *For every graph $G = (V, E)$, capacity function $c : V \to \overline{\mathbb{N}}$ and vertices $src, tgt \in V$, there exists a flow $f : E \to \overline{\mathbb{R}^+}$ of maximal value such that $f(u, v) \in \overline{\mathbb{N}}$ for all $(u, v) \in E$.*

With those two results in mind, we can establish Lemma 11.

Proof. As Γ is a population arena tracking T_f, by definition it is a finite union of commit ideals tracking T_f, hence we can decompose it as

$$\Gamma = \bigcup_{j=1}^{k} (\gamma_j \otimes \mathbf{v}_j, a_j)\downarrow.$$

Let $B < \omega$ be the largest constant used to define those commit ideals. That is, $B = \max(\{\mathbf{v}_j(s) \mid 1 \leq j \leq k, s \in S, \mathbf{v}_j < \omega\})$.

By definition, Γ_{0,T_f} is also a finite union of commit ideals tracking T_f. Let $(\gamma_f \otimes \mathbf{v}, a)\downarrow$ with $\mathbf{v} \in \{0, \omega\}^S$ be one of them. Let S_ω be the set of states s such that $\mathbf{v}(s) = \omega$ and let $d = |S_\omega|$. For all $N \in \mathbb{N}$, define $\gamma_f \otimes \mathbf{v}[N]$ as a configuration obtained by taking γ_f and adding N tokens on each state of S_ω. The set of configurations γ_0 which satisfy the

conclusion of the theorem is clearly stable by token renaming and token removal (because the set of paths in Γ_{1,T_f} is stable by token deletion). Thus, it is enough to prove it for configurations γ_0 of the specific form $\gamma_f \otimes \mathbf{v}[N], N \in \mathbb{N}$.

Fix $N \in \mathbb{N}$, we show that a path from $\gamma_f \otimes \mathbf{v}[N]$ to $\mathbf{F}$ in Γ_{1,T_f} exists.

The configuration $\gamma_f \otimes \mathbf{v}[B \cdot N]$ belongs to the ideal tracking T_f $(\gamma_f \otimes \mathbf{v})\downarrow$, which is a subset of Γ, hence there is a path $y_0 \xrightarrow{a_1} y_1 \xrightarrow{a_2} \cdots y_n$ within Γ with $y_0 = \gamma_f \otimes \mathbf{v}[B \cdot N]$ and $y_n \in \mathbf{F}$. Let T be the set of tokens used in y_0.

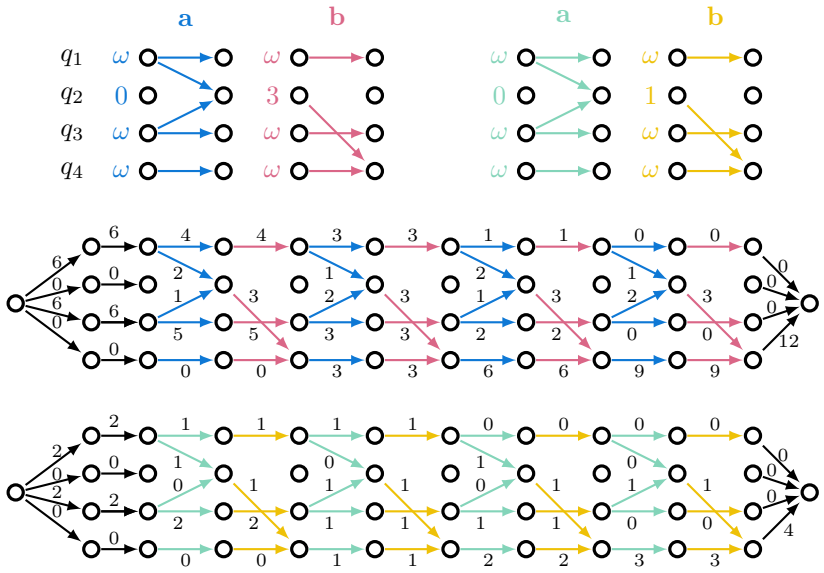

Figure 3 An illustration of the proof of Lemma 11. We do not describe Γ in full, only the two relevant ideals (which by themselves, do not form an arena). We have two actions, a and b. We can play a when there are no tokens in q_2, b when there are at most 3, and they let us transfer tokens along the indicated transitions. Those two actions let us transfer arbitrarily many tokens from q_1 and q_3 to q_4. With the notations of the proof, we have $d = 2$, $B = 3$ and $T_f = \emptyset$. Let $N = 2$. The graph G and flow given below represents the path transferring $BN = 6$ tokens from q_1 and q_3 to q_4. The corresponding flow has value $dBN = 12$. On the top right are restricted versions of our ideals, where finite positive numbers have been replaced by 1: in order to play b, we must have at most one token on q_2. By applying those constraints in G, we get a graph where the maximal flow has been divided by at most 3: by the max flow-min cut theorem and because all capacities have been divided by at most $B = 3$. We are thus guaranteed that there is a flow of value ≥ 4 in this graph (such as the one on the bottom graph), which corresponds to the transfer of $N = 2$ tokens from q_1 and q_3.

Consider the following directed graph $G = (V, E)$ and capacity function $c : V \to \overline{\mathbb{N}}$. The set of vertices is $V = S \times \{0, \ldots, n\} \cup \{src, tgt\} \cup \{r_s \mid s \in S\}$, i.e., one vertex for each state and configuration in the path, plus a source, a target, and one intermediate vertex r_s between src and each $(s, 0)$.

We define the set of edges E and the capacity function c on vertices of G as follows: for all $j \in \{0, \ldots, n-1\}$, we pick a commit ideal tracking T_f $(\gamma_j \otimes \mathbf{v}_j, a_{j+1})\downarrow$ in the ideal decomposition of Γ such that (γ_j, a_{j+1}) is in it. We also define $\gamma_n, \mathbf{v}_n$ such that $\mathbf{F} = (\gamma_n \otimes \mathbf{v}_n)\downarrow$ (they are well-defined since $\mathbf{F}$ is an ideal tracking T_f). For all $j < n$ and states s, s', there is an edge from (s, j) to $(s', j+1)$ if and only if $\Delta(s, a_{j+1})(s') > 0$. We also have edges from src to r_s, r_s to $(s, 0)$ and from (s, n) to tgt for all $s \in S$.

For every state s and index j, the vertex (s, j) is assigned capacity $c(s, j) = \mathbf{v}_j(s)$. Furthermore, for all $s \in S$, r_s has capacity $c(r_s) = B \cdot N$ if $\mathbf{v}(s) = \omega$ and 0 otherwise. Observe that by definition of B, we only assigned capacities in $\{0, \ldots, B, \omega\}$ to vertices (s, j).

The trajectories of tokens outside of T_f in the path $y_0 \xrightarrow{a_1} y_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} y_n$ define a flow $\phi : E \to \mathbb{N}$ from src to tgt in G of value dBN: for each edge $e = ((s, j-1), (s', j))$ we define $\phi(e)$ as the number of tokens of $T \setminus T_f$ going from s to s' at the jth step. We also define $\phi((src, r_s)) = \phi(r_s, (s, 0)) = BN$ for all $s \in S_\omega$ and 0 otherwise. For all $s \in S$, $\phi(((s, n), tgt))$ is the number of tokens of $T \setminus T_f$ in s at the end of the path. This flow satisfies the capacity constraints of G by definition of $(\mathbf{v}_j)_{0 \le j \le n}$.

Let us now define a new capacity function c^1 on G, as follows.

- For all $s \in S$ we set $c^1(r_s) = N$ if $s \in S_\omega$ and 0 otherwise.
- For all other vertices $v \in V \setminus \{r_s \mid s \in S\}$, we set $c^1(v) = 1$ if $c(v) \in \{1, \dots, B\}$ and $c^1(v) = c(v)$ if $c(v) \in \{0, \omega\}$.

We claim that G has an integer flow of value dN satisfying the capacity constraint c^1. Indeed, suppose the contrary, then by the integer flow theorem all flows have value $< dN$. Therefore, by the max-flow min-cut theorem there is a cut M in G such that $\mathbf{weight}_{G,c^1}(M) < dN$. As $c(v) \le B \cdot c^1(v)$ for all $v \in V$, that same cut has capacity $\mathbf{weight}_{G,c}(M) < dBN$ in G. By the max-flow min-cut theorem, this contradicts the existence of a flow of value dBN in (G, c). We obtain that (G, c^1) has a flow of value dN, which is optimal as $\{r_s \mid s \in S\}$ is a cut of value dN.

As all capacities defined by c^1 are integers, the integer flow theorem guarantees the existence of an optimal integer flow $\phi^1 : E \to \mathbb{N}$ of value dN, which in turn defines a path of length n from $\gamma_f \otimes \mathbf{v}[N]$ to $\mathbf{F}$: at step j, the tokens of T_f move in the same way as in the step $y_{j-1} \xrightarrow{a_j} y_j$, and the number of other tokens sent from state s to s' is $\phi^1((s, j-1), (s', j))$. Note that by definition of E tokens can only move from state s to s' if $\Delta(s, a_j)(s') > 0$.

Observe that the jth commit along this path belongs to the ideal $(\gamma_{j-1} \otimes \mathbf{v}_{j-1}, a_j)\downarrow$ because the flow f must respect the capacities of vertices G^1, which were derived from $\mathbf{v}_{j-1}$. The path therefore stays in Γ. In fact, by definition of c^1, the jth commit belongs to the smaller ideal $(\gamma_{j-1} \otimes \mathbf{v}_{j-1}, a_j)\downarrow$, where $\mathbf{v}^1_{j-1}$ is $\mathbf{v}_{j-1}$ where all finite positive coefficients have been replaced by 1. Consequently, it belongs to Γ_{1,T_f}. Since all $(r_s)_{s \in S_\omega}$ have capacity N, the resulting path starts with N tokens outside T_f in each state of S_ω. The path ends in $\mathbf{F}$ as the capacities of $(s, n)_{s \in S}$ constrain the final configuration to be in $(\gamma_n \otimes \mathbf{v}_n)\downarrow \subseteq \mathbf{F}$. This concludes the proof. ◀

5.2 The Isolation Lemma

In a configuration, we say that a set of states is an ω-*base* if an arbitrary amount of extra tokens could be placed on these states without exiting the arena Γ. This is formally defined as follows.

▶ **Definition 15** (ω-base and finite base). *Fix an arena Γ and $\gamma : T \to S$ a configuration in Γ over a set of tokens T. A set of states S_ω is an ω-base of γ in Γ if*

$$\gamma[S_\omega * \omega]\downarrow \subseteq \Gamma \ .$$

with $\gamma[S_\omega * \omega]$ the symbolic configuration obtained from $|\gamma|$ by mapping states of S_ω to ω.

By extension, a set of tokens $T_\omega \subseteq T$ is an ω-base of γ in Γ if the set of states occupied by those tokens in γ is. Dually, a set of tokens $T_f \subseteq T$ is a finite base of γ in Γ iff $T \setminus T_f$ is an ω-base of γ in Γ.

Note that a configuration γ may have several ω-bases and finite bases: for instance, say we have two states s_1, s_2 and the arena Γ is $(\omega, 1)\downarrow \cup (1, \omega)\downarrow$, then the configuration $[t_1 \mapsto s_1, t_2 \mapsto s_2]$ has $\{t_1\}$ and $\{t_2\}$ as ω-bases, but not $\{t_1, t_2\}$.

We say that two tokens t_1, t_2 *meet* in a configuration γ if they share the same state i.e. if $\gamma(t_1) = \gamma(t_2)$. The following lemma says that if starting from a configuration with a finite base T_f we can reduce the size of the finite base (bring back an individual token into the cohort) with a strategy that treats tokens symmetrically, then we can do so while making sure that the tokens of T_f never meet the tokens outside T_f before this goal is achieved.

Intuitively, while the tokens of T_f are all in bounded places, it means that they only meet boundedly many other tokens. Since this strategy works with an arbitrarily large cohort, we can show that in fact we can strengthen it to make sure that no token meets the ones in T_f.

▶ **Lemma 16** (Isolation Lemma). *Fix a population arena Γ and $k \in \mathbb{N}$, and assume that from all $\gamma \in \Gamma$ there is a strategy to almost surely reach a configuration with a finite base of size $< k$, without leaving Γ.*

Then for all $\gamma' \in \Gamma$ with a finite base T_f of size k, there is a strategy σ which, when starting from γ':

- *surely remains in Γ;*
- *almost surely reaches a configuration with a finite base strictly included in T_f, and*
- *guarantees that in every configuration before that, every state contains either only tokens of T_f or no token of T_f.*

Sketch of proof. This lemma is central to the proof of Theorem 7. Unfortunately we cannot include the full proof due to space constraints. The proof can be found in the full version, we sketch it here.

A crucial element in the assumption is that Γ is a population arena, i.e., it does not distinguish tokens. Since we have a winning strategy to reach a configuration with finite base of size $< k$ from everywhere in Γ, a safe random walk in Γ is a winning strategy, from all configurations of Γ. Again, this random walk treats tokens symmetrically.

Now consider an initial configuration γ' in Γ, with N tokens. On states of the ω-base, we can add many more tokens and still have a winning strategy. We add many imaginary tokens on all states of the ω-base. Then we make the following observation: as long as tokens in T_f remain outside the cohort, no token of the initial finite base T_f meets the cohort; they can only meet a few tokens at a time. If a token t of T_f meets sufficiently many tokens along a path, then some of them must have re-entered the cohort since they met. Since the random walk treats all tokens symmetrically, t also had a chance of reaching the cohort. This lets us bound the expected number of tokens t meets before reaching the cohort, independently of the number of tokens in total. As we add more imaginary tokens, the probability that one of the tokens t meets one of the real ones converges to 0.

We use these observations to show that we can obtain a probability as small as we want that any of the tokens of T_f meets tokens outside T_f before reaching the cohort. Since the set of configurations reachable from γ' is finite, this implies that this probability can be brought down to 0. As a consequence, we can make sure that a token of T_f reaches the cohort while no token of T_f meets one outside T_f beforehand. ◀

5.3 Proof of Theorem 7

The proof of Theorem 7 uses a partition $T_1, \ldots, T_d$ of the individuals T_f into non-empty groups with at most $|S|$ tokens per group. Moreover, the induction hypothesis assumes that

- individuals of different groups never meet in the arena, other than in final configurations.
- T_f is a finite base, i.e., the set of states occupied by the cohort is an ω-base.

Since every group occupies at least one state then $d \leq |S|$. The induction step is done by induction on $|S| - d$. The base case $|S| = d$ is trivial, because in that case all tokens are

individuals: the cohort is empty, and $|S|$-definability follows from the hypothesis that tokens of different groups never meet unless on F. The value of F is not fixed; the induction step uses the inductive hypothesis for $F = W_0$. The induction step is done as follows, details are in the full version.

- We try to follow a path to the target set $\mathbf{F}{\downarrow}$. We can choose this path so that we create at most $|S|$ individual tokens, by Lemma 11. These $|S|$ isolated tokens constitute a new group of individuals, denoted T_{d+1}.
- If we deviate from that path, we use Lemma 16 to define a sub-arena that lets us recover all individual tokens (i.e. bring them into the cohort) while making sure that none of them meets any token from outside their group. We then apply the induction hypothesis (for $d+1$ and F the set of configurations where tokens of T_{d+1} have been recovered) to recover those individual tokens within an $|S|$-definable sub-arena.
- Once all individuals are recovered, we apply the first step again, until we successfully follow the path to the end.

We define the desired sub-arena Y as the union of those paths and sub-arenas. Since, in any included configuration, the number of individuals residing on any one state is at most $|S|$, this arena must be $|S|$-definable. Clearly, Controller wins from any included configuration by following the strategy outlined above. Therefore, Y is a winning arena.

6 Theorem 7: The Dynamic Flow Problem

To prove Theorem 8, we rely on a result by Gimbert, Mascle and Totzke [10] about maximising the flow through directed graphs, where edge capacities are dynamically chosen by the maximiser. We recall the notation and statement of the relevant theorem.

A *capacity* is a function $\alpha : S^2 \to \mathbb{N} \cup \{\omega\}$ mapping each pair of states (s_1, s_2) to a maximal number of tokens that can be transferred from s_1 to s_2. Let $A \subseteq (\mathbb{N} \cup \{\omega\})^{S^2}$ be a finite alphabet of capacities, and $E \subseteq S^2$ a set of pairs of states. Given a word $w = \alpha_1 \dots \alpha_k \in A^*$ and a set of tokens T, a *token flow* over w is a sequence of configurations $\tau = \gamma_0 \to \cdots \to \gamma_k \in (S^T)^*$ such that, for every $i \in [1, k]$ and $s, s' \in S$, $|\{t \in T \mid \gamma_{i-1}(t) = s \wedge \gamma_i(t) = s'\}| \leq \alpha_i(s, s')$. Define the *global flow* of τ as the function $g(\tau) : S^2 \to \mathbb{N}$ counting the number of tokens moving between each pair of states: formally, for all $s, s' \in S$,

$$g(\tau)(s, s') = |\{t \in T : \gamma_0(t) = s \wedge \gamma_k(t) = s'\}|.$$

Now let $L \subseteq A^*$ be a language over A. The *maximal sequential multi-flow on E satisfying L*, denoted $\xi(E, L)$, is the maximum number of tokens that can be simultaneously transferred between every pair of states in E by a path over some word of L.

$$\xi(E, L) = \sup_{w \in L} \sup_{\pi \text{ over } w} \min_{(s,s') \in E} g(\pi)(s, s').$$

▶ **Example 17.** Consider the capacities from Figure 4, and set of edges $E = \{(s_1, s_4), (s_3, s_4)\}$. For the language $L = \{abba\}$ containing only the capacity word $abba$, we get $\xi(E, L) = 1$, witnessed by the token flow τ with global flow $g(\tau)(s_1, s_4) = g(\tau)(s_3, s_4) = 1$. Note that the maximal flow from s_2 to s_3 for the intermediate word bb cannot exceed 2. However letting $L = \{ab^n a\}$, we have $\xi(E, L) = \infty$, because for every n there is a token flow τ_n over $ab^{2n}a$ so that $g(\tau_n)(s_1, s_4) = g(\tau_n)(s_3, s_4) = n$.

As Gimbert et al. show, one can determine if such flows are unbounded, and otherwise precisely compute their finite value, in polynomial space.

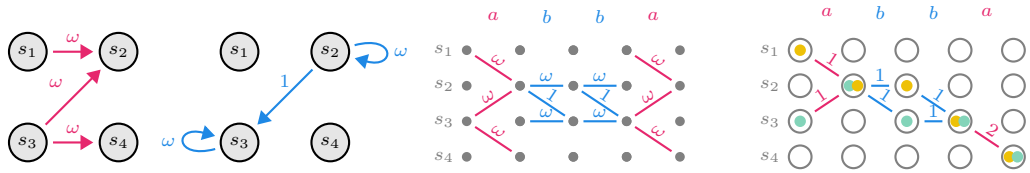

Figure 4 Two capacity constraints $a, b : S \to \mathbb{N} \cup \{\omega\}$, a capacity word, and a token flow over it.

▶ **Theorem 18** (Theorem 37 in [10]). *Let A be a set of capacities over a set of states S, with coefficients in $\{0, \dots, K, \omega\}$, and $L \subseteq A^*$ a regular language recognised by an NFA with m states. Then, either $\xi(E, L) = +\infty$, or $\xi(E, L) \leq K(2|S|)^{(170 \log_2(m)+835)|S|^{12}}$.*

This is almost what we need for deciding the Dynamic flow problem, except for two things. First, our constraints are on vertices while theirs are on edges. Second, in our initial ideal we may have both states marked ω and states bounded by finite numbers. We will translate the constraints given by ideals over configurations into capacities, thereby reinterpreting paths as token flows. This way one can use Theorem 18 to check that we can transfer arbitrarily many tokens from the states marked ω, while using the regular constraint to track the finite number of extra tokens and make sure that all token end up in F.

Let $\mathcal{M} = (S, \Sigma, \Delta)$, $\mathbf{V}$, $\mathbf{v}_0$, $\mathbf{F}$ be an instance of the Dynamic flow problem. We split $\mathbf{v}_0$ in two parts $\mathbf{v}_0 = \mathbf{v}_f + \mathbf{v}_\omega$, so that $\mathbf{v}_f \in \mathbb{N}^S$ and $\mathbf{v}_\omega \in \{0, \omega\}^S$. Vector $\mathbf{v}_\omega$ describes the set of states with arbitrarily many tokens at the start, while $\mathbf{v}_f$ describes the remaining tokens in other states. Intuitively, the Dynamic flow problem comes down to checking that we can simultaneously transfer tokens of $\mathbf{v}_f$ and arbitrarily many tokens from every state marked ω in $\mathbf{v}_0$ to $\mathbf{F}{\downarrow}$. We will construct a finite alphabet of capacities and a finite automaton $\mathcal{A}$ which guesses the movement of the remaining tokens. Its language is the set of sequences of capacities which can be used while making sure that both the cohort of tokens from unbounded states and those remaining tokens are brought safely to $\mathbf{F}$. The largest finite number appearing in the capacities is the same as in $\mathbf{V}$, and $\mathcal{A}$ has exponential size in in S and the logarithm of the largest finite number appearing in $\mathbf{V}$.

▶ **Lemma 19.** *$\mathcal{M}$, $\mathbf{V}$, $\mathbf{v}_0$, $\mathbf{F}$ is a positive instance of the Dynamic flow problem if and only if there exists $E \subseteq S^2$ such that*

1. $\xi(E, L(\mathcal{A})) = +\infty$
2. $\mathbf{v}_0^{-1}(\omega) \subseteq \{s \in S \mid \exists s' \in S, (s, s') \in E\}$.

Proof of Theorem 8. By Lemma 19, in order to check if $\mathcal{M}$, $\mathbf{V}$, $\mathbf{v}_0$, $\mathbf{F}$ is a positive instance of the Dynamic flow problem, we can enumerate sets $E \subseteq S^2$ and for each such E, check the two conditions from Lemma 19. Condition 2 can be checked easily in polynomial time (and space). For condition 1, we can rely on Theorem 18: Note that the automaton $\mathcal{A}$ as constructed above is singly exponential in the input. Consequently, the same holds for m and the quantity $M = K(2|S|)^{(170 \log_2(m)+835)|S|^{12}}$. This means that in order to check whether $\xi(E, L(\mathcal{A})) = +\infty$, it suffices to check that there is a word $w \in L(\mathcal{A})$ and a path π over w carrying $M + 1$ tokens from s to s' for all $(s, s') \in E$. These can be guessed on the fly: it suffices to memorise the current state in the automaton and the number of tokens in each state, in binary. This only requires polynomial space in $\log(K)$ and $|S|$. ◀

7 Lower bounds an a hard case

7.1 Complexity lower bound

An exponential-time lower bound for the Random population control problem can be established by reduction from countdown games, as shown by Mascle, Shirmohammadi, and Totzke in [14].

▶ **Definition 20.** *A Countdown Game is given by a directed graph $\mathcal{G} = (V, E)$, where edges carry positive integer weights, $E \subseteq (V \times \mathbb{N}_{>0} \times V)$. For an initial pair $(v, c_0) \in V \times \mathbb{N}$ of a vertex and a number, two opposing players (Player 1 and 2) alternatingly determine a sequence of such pairs as follows. In each round, from (v, c), Player 1 picks a number $d \leq c$ such that E contains at least one edge (v, d, v'); then Player 2 picks one such edge and the game continues from $(v', c - d)$. Player 1 wins the game iff the play reaches a pair in $V \times \{0\}$.*

Determining the winner of a Countdown Game, where all constants are given in binary, is ExpTime-complete [12]. We state the lower bound and a sketch of the construction.

▶ **Theorem 21.** *The Random population control problem is ExpTime-hard.*

Proof sketch. The number of turns in a Countdown Game cannot exceed the initial value of the counter, as the initial counter value decreases at each turn. Thus, if Player 2 has a winning strategy, choosing actions at random yields a positive probability of applying that strategy, hence a positive probability of winning. Therefore, Player 1 wins the initial game if and only if she wins with probability one against a randomised adversary.

The main idea for the further construction is to require Controller, who impersonates Player 1, to move tokens one-by-one from a waiting state, first into the control graph of the Countdown game, and ultimately into the target. To avoid a loss in the intermediate phase, she must win an instance of the game against a randomising opponent. This is enforced using a combination of gadgets, including two binary counters (encoded using two states per bit) that can effectively test for zero, be set to specific numbers, and that are set up so that they can decrement at the same rate. These are used to hold the global integral value n, and an auxiliary counter holding the value d chosen by Player 1. Controller is compelled to reduce them both and can only continue once the auxiliary counter is exhausted. She can only afford to safely end the simulation of the game if the first counter holds value 0. As a result, Player 1 has a winning strategy for the two-player Countdown Game if, and only if, Controller can synchronise the n-fold product of the constructed MDP for all n. ◀

7.2 A Double-exponential Cut-off

We use the following vocabulary: a state s is *marked* in a configuration γ if at least one token occupies it: $\exists t.\gamma(t) > 0$. Whenever an action a takes state s only back to itself we say that s *ignores* a. There are states `Heaven` (the target) and `Hell` which ignore all actions. For a given state s, an action a is *angelic* if it takes s only to `Heaven`, and *daemonic* if it takes s to `Hell`. An action a is *safe* in a configuration if it is not daemonic for any marked state (in any gadget).

The *cut-off* of an MDP is the minimal number of tokens against which Controller does not have an almost-sure winning strategy, In this section we show that there are MDPs whose cut-off is finite but doubly-exponential in the number of states. We reuse a construction from [3]. We simply replace the adversary with randomisation. Let us summarise it here.

Let $N \in \mathbb{N}$. We build an MDP $\mathcal{M}_N$ as follows: We have an initial state i and a final state f, plus a sink state `sink`.

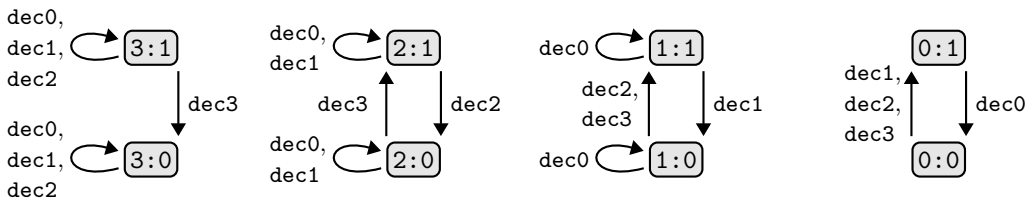

Figure 5 A (4-bit) Binary Counter. Not displayed are edges labelled by `(deci)` that make the respective actions daemonic for state `(i:0)`, and error actions `errori`, which are daemonic for `(i:0)` and `(i:1)`, for all bits $i \in \{0, 1, 2, 3\}$.

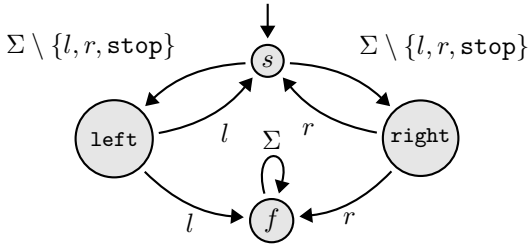

Figure 6 The splitting gadget. The action `stop` sends all tokens in s, `left` and `right` to a sink state.

- First of all we have a *splitting gadget*, as shown in Figure 6. This gadget forces Controller to play a word of $((\Sigma_N \setminus \{l, r\})\{l, r\})^*$ until all three states are empty. We call a sequence of two actions of $(\Sigma_N \setminus \{l, r\})\{l, r\}$ a *round*. Say there are 2^K tokens in `s` at the start, then there is a positive probability that they distribute evenly between `left` and `right` whenever they split from `s`. Therefore, for any strategy of Controller, there is a positive probability that after K rounds there is still at least one token in `s`.
- Second, we have a *counter gadget*, as shown in Figure 5. This one consists of $2N$ states, representing N bits, $\{(i : 0), (i : 1) \mid 0 \leq i \leq N - 1\}$. For all $0 \leq j \leq N - 1$ there is an action $\texttt{dec}_j$ that fixes all states $(k : 0), (k : 1)$ with $k > j$, sends $(j : 1)$ to $(j : 0)$, and sends each $(k : 0)$ with $k < j$ to $(k : 1)$. The tokens in states $(k : 1)$ with $k < j$ and $(j : 0)$ are all sent to `sink`. Action `stop` sends all tokens in $(i : 0)_{1 \leq i \leq N}$ to f and all tokens in $(i : 1)_{1 \leq i \leq N}$ to `sink`. Actions l, r leave the tokens in those states idle. This gadget implements a binary counter over N bits. Let $\texttt{Dec} = \{\texttt{dec}_j \mid 1 \leq j \leq N\}$. If we start with tokens in states $(i : 0)_{1 \leq i \leq N}$ and not in states $(i : 1)_{1 \leq i \leq N}$, then at all times Controller has only one action from `Dec` available, which increments the counter. Once every bit is 1, the only possible action apart from $\{l, r\}$ is `stop`. She is forced to play a sequence of actions of $\texttt{Dec}^{2^N-1}\texttt{stop}$, interleaved with some l and r.
- We add an action `start` which sends tokens from i to $\{\texttt{s}\} \cup \{0_i \mid 1 \leq i \leq N\}$, and leaves all other tokens idle. All other actions map i to `sink`.

If there are more than $2^{2^N} + N$ tokens, then, with positive probability, when playing `start` one goes to each of the $(i : 0)_{1 \leq i \leq N}$ and the rest to `s`. Then, as observed above, with positive probability the first gadget forces Controller to play at least 2^N actions from $\Sigma_N \setminus \{l, r\}$ before being able to play `stop`. However, the second gadget forces Controller to play `stop` after playing 2^N actions from $\Sigma \setminus \{l, r, \texttt{stop}\}$. Hence Controller cannot win almost surely.

If there are fewer than 2^{2^N} tokens, then Controller can play as follows: Play `start`, then alternate between playing the available $\texttt{dec}_j$ (it may be that several actions are available because some $(i:0)$ did not receive any tokens in the first step; in that case, choose the minimal j) and playing either l or r (whichever sends the most tokens to f). After 2^N steps, all tokens in the counter gadget are in the states $(i:1)_{1\leq i\leq N}$. Since at every round we send at least half of the tokens in the splitting gadget to f, after 2^N rounds the gadget is empty. Controller can then play `stop` safely, and send all tokens in the counter gadget to f.

We obtain an MDP with $2N+6$ states whose cut-off is between 2^{2^N} and $2^{2^N}+N$.

7.3 A Butterfly

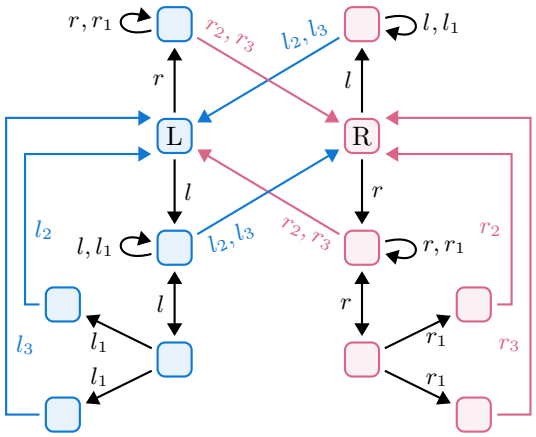

Figure 7 An automaton where Controller can synchronise any finite number of tokens but no deterministic, k-definable winning strategy exists.

Consider now the example in Figure 7. We will set it up so that initially, all tokens are randomly distributed onto states L and R, and Controller must eventually place all of them on only one side of the graph. To do so,

- add an initial action which moves tokens from an initial state to L and R and is daemonic everywhere else;
- add a fresh winning action that is angelic for all red states and daemonic for all green states; and
- add a fresh winning action that is angelic for all green states and daemonic for all red states.

Idea. Each round starts with all tokens on L and R. Controller stepwise proposes a sequence of actions, either in $l^+l_1[l_2l_3]$ or $r^+r_1[r_2r_3]$. Notice that until one side is empty, these are the only safe sequences to play. At the end of each round, all tokens (except possibly one) will switch sides. Controller can choose to isolate one token and keep it on its side, thereby getting closer to her goal of moving everyone to one side. The relevant decisions to make are

1. whether to go left or right at the start of a round
2. when to stop playing l (or r, resp.)

We say that two configurations γ_1, γ_2 are K-equivalent if for all states s, they either have the same number of tokens in s or both have at least K tokens in s. A strategy is K-deterministic if it "counts up to K". That is, it is deterministic and for all K-equivalent γ_1, γ_2, it plays the same action from both.

▶ **Lemma 22.** *The example is a positive instance of the* Random population control problem. *However, for every $K \geq 0$, every K-deterministic strategy is losing.*

Proof. Referring to the relevant decisions above, a winning strategy is to

- always pick the side with the least number of tokens (one can also pick the side at random), and
- play l (or r) until exactly one token is isolated.

Consider a K-deterministic strategy. Say we start with $2K + 2$ tokens, and $K + 1$ tokens end up on both L and R. Our strategy must play l or r, say it plays l. We have $K + 1$ tokens in the state below L and in the one above R. Playing l_2 or l_3 brings us back to the previous configuration (up to renaming tokens), and playing l_1 has no effect, thus it must play l. Once this is done, it must keep playing l until exactly one token is in the lower state: if all tokens are in the upper state we have established that it plays l, and if there are 2 or more tokens below, every move is losing except l (playing l_1 risks tokens splitting between the two left states, after which we are stuck). Therefore, the strategy must wait until one token is in the lower state, then play l_1 and then l_2 or l_3.

L now contains $K + 2$ tokens and R contains K, which is K-equivalent to $K + 1$ and $K + 1$. By repeating the same reasoning, in the next round the strategy does the same thing, which yields a configuration with $K + 1$ tokens in both L and R, meaning we are back to the initial configuration. ◀

8 Conclusion

We have shown that the Random population control problem is ExpTime-complete. We establish that it is possible to win the population MDP while staying in a part of the winning region that has low descriptive complexity, in the sense of Theorem 7. This is the key to defining an algorithm (Algorithm 1) to solve the Random population control problem using an exponential number of calls to an oracle solving the Dynamic flow problem.

These results shed new light on parameterised control and pave the way for further positive results with more ambitious objectives. For example, the tools developed in this paper may be used to address a generalised version of the problem, where infinite executions must satisfy ω-regular conditions.

A natural question that remains open is the cut-off: if there is a number n such that we cannot synchronize almost surely n tokens, how large can the smallest such n be? In the adversarial case studied in [3], it was shown that the cut-off was doubly exponential in the worst case. We conjecture that this is also the case in our setting. We show in Appendix 7.2 that the cut-off may be doubly exponential, but we are missing a matching upper bound.

References

1 S. Akshay, Blaise Genest, and Nikhil Vyas. Distribution-based objectives for markov decision processes. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018*, pages 36–45. ACM, 2018. doi:10.1145/3209108.3209185.

2 Dana Angluin, James Aspnes, Zoë Diamadi, Michael J. Fischer, and René Peralta. Computation in networks of passively mobile finite-state sensors. *Distributed Comput.*, 18(4):235–253, 2006. doi:10.1007/S00446-005-0138-3.

3 Nathalie Bertrand, Miheer Dewaskar, Blaise Genest, and Hugo Gimbert. Controlling a population. In *28th International Conference on Concurrency Theory, CONCUR 2017, September 5-8, 2017, Berlin, Germany*, volume 85 of *LIPIcs*, pages 12:1–12:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.CONCUR.2017.12.

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

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

6 Thomas Colcombet, Nathanaël Fijalkow, and Pierre Ohlmann. Controlling a random population. *Log. Methods Comput. Sci.*, 17(4), 2021. doi:10.46298/LMCS-17(4:12)2021.

7 Laurent Doyen. Stochastic games with synchronization objectives. *J. ACM*, 70(3):23:1–23:35, 2023. doi:10.1145/3588866.

8 E. Allen Emerson and Kedar S. Namjoshi. Reasoning about rings. In *Conference Record of POPL'95: 22nd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, San Francisco, California, USA, January 23-25, 1995*, pages 85–94. ACM Press, 1995. doi:10.1145/199448.199468.

9 Lester Randolph Ford and Delbert Ray Fulkerson. Maximal flow through a network. *Canadian journal of Mathematics*, 8:399–404, 1956.

10 Hugo Gimbert, Corto Mascle, and Patrick Totzke. Optimal sequential flows. In Sayan Bhattacharya, Danupon Nanongkai, Michael Benedikt, and Gabriele Puppis, editors, *53rd International Colloquium on Automata, Languages, and Programming (ICALP 2026)*, volume 374 of *LIPIcs*, pages 98:1–98:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2026. doi:10.4230/LIPIcs.ICALP.2026.98.

11 Emile Hazard, Olivier Idir, and Denis Kuperberg. Explorable parity automata. *CoRR*, abs/2410.23187, 2024. doi:10.48550/arXiv.2410.23187.

12 Marcin Jurdzinski, François Laroussinie, and Jeremy Sproston. Model checking probabilistic timed automata with one or two clocks. In *Tools and Algorithms for the Construction and Analysis of Systems, 13th International Conference, TACAS 2007, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2007 Braga, Portugal, March 24 - April 1, 2007, Proceedings*, volume 4424 of *Lecture Notes in Computer Science*, pages 170–184. Springer, 2007. doi:10.1007/978-3-540-71209-1_15.

13 Daniel Kirsten. Distance desert automata and the star height problem. *RAIRO Theor. Informatics Appl.*, 39(3):455–509, 2005. doi:10.1051/ITA:2005027.

14 Corto Mascle, Mahsa Shirmohammadi, and Patrick Totzke. Controlling a random population is ExpTime-hard. *CoRR*, abs/1909.06420, 2019. arXiv:1909.06420.

15 Donald Ornstein. On the existence of stationary optimal strategies. *Proceedings of the American Mathematical Society*, 20(2):563–569, 1969. doi:10.2307/2035700.

16 Carl Adam Petri. Kommunikation mit Automaten. Dissertation, Schriften des IIM 2, Rheinisch-Westfälisches Institut für Instrumentelle Mathematik an der Universität Bonn, 1962.

17 Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., 1st edition, 1994. doi:10.1002/9780470316887.

18 G. Rozenberg and A. Salomaa. *Handbook of Formal Languages*. Number v. 1-3 in Handbook of formal languages / G. Rozenberg; A. Salomaa. Springer, 1997.

19 Sylvain Schmitz. *Algorithmic Complexity of Well-Quasi-Orders*. Accreditation to supervise research, École normale supérieure Paris-Saclay, November 2017.

20 A. Prasad Sistla and Steven M. German. Reasoning with many processes. In *Proceedings of the Symposium on Logic in Computer Science (LICS '87), Ithaca, New York, USA, June 22-25, 1987*, pages 138–152. IEEE Computer Society, 1987.

21 David Soloveichik, Matthew Cook, Erik Winfree, and Jehoshua Bruck. Computation with finite stochastic chemical reaction networks. *Nat. Comput.*, 7(4):615–633, 2008. doi:10.1007/S11047-008-9067-Y.

22 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.

23 Jannis Uhlendorf, Agnès Miermont, Thierry Delaveau, Gilles Charvin, François Fages, Samuel Bottani, Pascal Hersen, and Gregory Batt. In silico control of biomolecular processes. *Computational Methods in Synthetic Biology*, pages 277–285, 2015.