How to Play Optimally for Regular Objectives?

Patricia Bouyer
Université Paris-Saclay, CNRS, ENS Paris-Saclay, Laboratoire Méthodes Formelles, 91190, Gif-sur-Yvette, France

Nathanaël Fijalkow
CNRS, LaBRI and Université de Bordeaux, France
University of Warsaw, Poland

Mickael Randour
F.R.S.-FNRS & UMONS – Université de Mons, Belgium

Pierre Vandenhove
F.R.S.-FNRS & UMONS – Université de Mons, Belgium
Université Paris-Saclay, CNRS, ENS Paris-Saclay, Laboratoire Méthodes Formelles, 91190, Gif-sur-Yvette, France

Abstract

This paper studies two-player zero-sum games played on graphs and makes contributions toward the following question: given an objective, how much memory is required to play optimally for that objective? We study regular objectives, where the goal of one of the two players is that eventually the sequence of colors along the play belongs to some regular language of finite words. We obtain different characterizations of the chromatic memory requirements for such objectives for both players, from which we derive complexity-theoretic statements: deciding whether there exist small memory structures sufficient to play optimally is NP-complete for both players. Some of our characterization results apply to a more general class of objectives: topologically closed and topologically open sets.

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

Keywords and phrases two-player games on graphs, strategy complexity, regular languages, finite-memory strategies, NP-completeness

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.118

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

Related Version *Full Version*: `https://arxiv.org/abs/2210.09703` [4]

Funding This work has been partially supported by the ANR Project MAVeriQ (ANR-20-CE25-0012) and by the Fonds de la Recherche Scientifique – FNRS under Grant n° T.0188.23 (PDR ControlleRS). Mickael Randour is an F.R.S.-FNRS Research Associate and a member of the TRAIL Institute. Pierre Vandenhove is an F.R.S.-FNRS Research Fellow.

Acknowledgements We thank Antonio Casares and Igor Walukiewicz for valuable discussions about this article.

1 Introduction

Games on graphs is a fundamental model in theoretical computer science for modeling systems involving competing agents. Its applications include model-checking, program verification and synthesis, control theory, and reactive synthesis: in all cases, the system specification is turned into a winning objective for a player and the goal is to construct a winning strategy. Some central results in the field state that for some objectives, there exist memoryless optimal strategies, meaning not requiring any memory. For instance, the celebrated memoryless determinacy result for (infinite) parity games is a key ingredient in the modern proof of decidability of monadic second-order logic over infinite trees by Gurevich and Harrington [16].

Memory requirements. However for many objectives, some memory is required; a central question is therefore, stated informally:

Given an objective, how much memory is required to play optimally for this objective?

The first answers to this question, at the dawn of the study of games, were memory requirements for concrete objectives, such as Rabin objectives [22]. The work of Dziembowski, Jurdziński, and Walukiewicz [13] gave a computable characterization of memory requirements for the whole class of Muller objectives. This triggered the following long-term research goal: characterizing the memory requirements for ω-regular objectives.

Regular objectives. Many results have been obtained toward this research goal; we refer to the related works section in Section 3 for further details. The most pressing open question in that direction is regular objectives, meaning the special case of ω-regular objectives concerned with finite duration: in this setting, the objective is induced by a regular language over finite words and the goal of one of the players is that *eventually* the sequence of colors along the play belongs to this language. We call these *regular reachability objectives*. The opponent's objective is then to ensure that the sequence of colors *never* belongs to the language, describing *regular safety objectives*.

A first observation is that for such a regular (reachability or safety) objective, a deterministic finite automaton recognizing the regular language provides an upper bound on the memory requirements of both players. Indeed, playing with the extra information from the automaton reduces the game to a standard reachability or safety game, for which no further memory is required to make optimal decisions. Yet, as we will see, structures smaller than the minimal automaton recognizing the language may suffice for the players.

Chromatic memory. One of the many contributions of Kopczyński [18] in the study of memory for games on graphs is the notion of chromatic memory. In this model, the memory states are updated only using the sequence of colors seen along a play, and in particular do not depend on the graph itself (as opposed to chaotic memory, which may use information from the graph in its updates). Kopczyński conjectured [18] that for ω-regular objectives, chromatic and chaotic memory requirements coincide; unfortunately, this does not hold, as recently proved by Casares [8] (i.e., there are objectives for which the number of memory states required to play optimally in all arenas differs depending on the memory model). In our study, we will see another counterexample using regular objectives.

Contributions. We study the chromatic memory requirements of both regular reachability and regular safety objectives. For both cases, we give a combinatorial characterization of the memory structures sufficient to play optimally in all arenas (of any cardinality). As a by-product of the characterization we obtain complexity-theoretic statements: given as input a deterministic finite automaton representing the objective,

- deciding whether a memory structure suffices to play optimally in all arenas can be done in polynomial time;
- deciding the existence of a sufficient memory structure with a given number of states is NP-complete.

From our characterizations it also follows that for both regular reachability and safety objectives, chromatic and chaotic memory requirements do not coincide.

We also discuss when relevant the extension of our results to the more general class of topologically open and topologically closed objectives (called respectively *general reachability objectives* and *general safety objectives* for consistency in what follows), which include the regular reachability and regular safety objectives.

Implementation. In order to test ideas and conjectures, we have implemented algorithms that automatically build a memory structure with a minimal number of states, both for regular reachability and regular safety objectives. These algorithms are based on the theoretical analysis from this paper. Our implementation[1] uses SAT solvers provided by the Python package PySAT [17].

Structure of the paper. All required definitions are provided in Section 2. Section 3 includes an in-depth discussion of related works and a technical overview of the results and proofs: Section 3.1 for safety objectives, Section 3.2 for reachability objectives, and Section 3.3 for computational complexity results. Due to length constraints, only proof sketches are provided in this version of the article; complete proofs are available in the full version [4]. In particular, proofs for safety objectives are available in [4, Section 4], for reachability objectives in [4, Section 5], and for complexity-theoretic statements in [4, Section 6].

2 Preliminaries

Let C be a non-empty alphabet of *colors*.

Arenas. We study zero-sum turn-based games on graphs with two players, called $\mathcal{P}_1$ and $\mathcal{P}_2$. Players play on *arenas*, which are tuples $\mathcal{A} = (V, V_1, V_2, E)$ where V is a non-empty set of *vertices* such that $V = V_1 \uplus V_2$ (disjoint union) and $E \subseteq V \times C \times V$ is a *set of colored edges*. If $e = (v_1, c, v_2) \in E$, we write $\mathsf{in}(e) = v_1$, $\mathsf{col}(e) = c$, and $\mathsf{out}(e) = v_2$. Vertices in V_1 are controlled by $\mathcal{P}_1$ and vertices in V_2 are controlled by $\mathcal{P}_2$. An arena is *finite* if it has finitely many vertices and edges, and is *finitely branching* if for all $v \in V$, there are finitely many edges $e \in E$ such that $\mathsf{in}(e) = v$. Unless otherwise specified, we consider arenas of any cardinality. An arena $\mathcal{A} = (V, V_1, V_2, E)$ is a *one-player arena of* $\mathcal{P}_1$ (resp. *of* $\mathcal{P}_2$) if $V_2 = \emptyset$ (resp. $V_1 = \emptyset$).

A *history* on arena $\mathcal{A} = (V, V_1, V_2, E)$ is a finite sequence $\gamma = e_1 \dots e_n \in E^*$ such that for i, $1 \leq i \leq n-1$, we have $\mathsf{out}(e_i) = \mathsf{in}(e_{i+1})$. We write $\mathsf{out}(\gamma)$ for $\mathsf{out}(e_n)$. For convenience, we assume that for all $v \in V$, there is a distinct empty history λ_v such that $\mathsf{out}(\lambda_v) = v$. For $i \in \{1, 2\}$, we write $\mathsf{Hists}_i(\mathcal{A})$ for the set of histories γ on $\mathcal{A}$ such that $\mathsf{out}(\gamma) \in V_i$. A *play* on arena $\mathcal{A}$ is an infinite sequence $\pi = e_1 e_2 \dots \in E^\omega$ such that for $i \geq 1$, $\mathsf{out}(e_i) = \mathsf{in}(e_{i+1})$; play π is *from* v if $\mathsf{in}(e_1) = v$. If $\pi = e_1 e_2 \dots \in E^\omega$ is a play (resp. $\gamma = e_1 \dots e_n \in E^*$ is a history), we write $\mathsf{col}^\omega(\pi)$ (resp. $\mathsf{col}^*(\gamma)$) for the infinite sequence $\mathsf{col}(e_1)\mathsf{col}(e_2) \dots \in C^\omega$ (resp. the finite sequence $\mathsf{col}(e_1) \dots \mathsf{col}(e_n) \in C^*$).

Objectives. *Objectives* are subsets $W \subseteq C^\omega$. Given an objective W, we write $\overline{W} = C^\omega \setminus W$ for its complement. We focus on two types of objectives, both derived from a set $A \subseteq C^*$:

- the *general reachability objective derived from* A, denoted $\mathsf{Reach}(A)$, is the objective $\bigcup_{w \in A} wC^\omega$ of infinite words that have (at least) one finite prefix in A.
- the *general safety objective derived from* A, denoted $\mathsf{Safe}(A)$, is the objective $\overline{\bigcup_{w \in A} wC^\omega}$ of infinite words that have no finite prefix in A. We have $\mathsf{Safe}(A) = \overline{\mathsf{Reach}(A)}$.

General reachability and safety objectives are respectively the *topologically open* and *topologically closed sets*, at the first level of the Borel hierarchy. When A is a regular language, we call $\mathsf{Reach}(A)$ a *regular reachability objective* and $\mathsf{Safe}(A)$ a *regular safety objective*. We

[1] Our implementation is available at `https://github.com/pvdhove/regularMemoryRequirements`.

call an objective *regular* if it is a regular reachability or a regular safety objective. Our characterizations apply to regular reachability and safety objectives, but we sometimes discuss when we may generalize our results to the general case. For computational complexity questions, we restrict our focus to regular reachability and safety objectives so that an objective can be finitely represented as an automaton. The objectives that we consider are therefore very simple both in terms of their algebraic representation (using automata representing languages of finite words) and in terms of their topology (they are at the first level of the Borel hierarchy).

A *game* is a tuple $\mathcal{G} = (\mathcal{A}, W)$ where $\mathcal{A}$ is an arena and W is an objective.

Automata. A *deterministic automaton* is a tuple $\mathcal{D} = (Q, C, q_{\mathsf{init}}, \delta, F)$ where Q is a possibly infinite set of *states*, C is a non-empty *alphabet* (usually the set of colors), $q_{\mathsf{init}} \in Q$ is an *initial state*, $\delta \colon Q \times C \to Q$ is a (complete, deterministic) *update function*, and $F \subseteq Q$ is a set of *final states*. All automata in this work are deterministic, so we sometimes omit the word *deterministic*. Automaton $\mathcal{D}$ is *finite* if Q is finite. We write $\delta^* \colon M \times C^* \to M$ for the natural extension of δ to sequences of colors. The *language recognized by* $\mathcal{D}$, denoted $\mathcal{L}(\mathcal{D})$, is the set of finite words $w \in C^*$ such that $\delta^*(q_{\mathsf{init}}, w) \in F$. For $q_1, q_2 \in Q$, we write $\Pi^{\mathcal{D}}_{q_1,q_2}$ for the language of words $w \in C^*$ such that $\delta^*(q_1, w) = q_2$. We drop the superscript $\mathcal{D}$ if the automaton considered is clear in the context. We denote the empty word by ε.

Continuations. For an objective $W \subseteq C^\omega$ and $w \in C^*$, we define the *winning continuations of* w as the set $w^{-1}W = \{w' \in C^\omega \mid ww' \in W\}$ (this set is sometimes called a *left quotient* of W in the literature). Given an objective $W \subseteq C^\omega$, its *prefix preorder* $\preceq_W \subseteq C^* \times C^*$ is defined as $w_1 \preceq_W w_2$ if $w_1^{-1}W \subseteq w_2^{-1}W$. Its *prefix equivalence* $\sim_W \subseteq C^* \times C^*$ is defined as $w_1 \sim_W w_2$ if $w_1^{-1}W = w_2^{-1}W$. We denote $\prec_W = \preceq_W \setminus \sim_W$. We drop the subscript W when there is no ambiguity on the objective. The prefix preorder is a relation that is preserved by reading colors.

▶ **Lemma 1.** *Let $W \subseteq C^\omega$ be an objective. If $w_1 \preceq w_2$, then for all $w \in C^*$, $w_1 w \preceq w_2 w$.*

Starting from a general reachability or safety objective $W \subseteq C^\omega$ derived from a set $A \in C^*$, we can associate with W its minimal automaton $\mathcal{D}_W$ that "classifies" the equivalence classes of $\sim$. Formally, $\mathcal{D}_W = (Q, C, q_{\mathsf{init}}, \delta, F)$ where $Q = \{[w]_\sim \mid w \in C^*\}$ is the set of equivalence classes of $\sim$, $q_{\mathsf{init}} = [\varepsilon]_\sim$, $\delta([w]_\sim, c) = [wc]_\sim$, and $F = \{q_{\mathsf{fin}}\}$ where $q_{\mathsf{fin}} = [w]_\sim$ for some $w \in A$ (the choice of w does not matter). The transition function δ is well-defined: $w_1 \sim w_2$ implies $w_1 c \sim w_2 c$ for all $c \in C$. Notice that the final state of such an automaton is always absorbing, i.e., for all $c \in C$, $\delta(q_{\mathsf{fin}}, c) = q_{\mathsf{fin}}$. This matches the intuition that once a word of A is seen and the reachability (resp. safety) game is won (resp. lost), it stays that way for the rest of the game.

We have that a general reachability (resp. safety) objective W is equal to $\mathsf{Reach}(\mathcal{L}(\mathcal{D}_W))$ (resp. to $\mathsf{Safe}(\mathcal{L}(\mathcal{D}_W))$) – in examples, we will sometimes start from an automaton to generate an objective. Using the well-known Myhill-Nerode theorem [20], we obtain that a general reachability or safety objective W is regular if and only if $\sim$ has finitely many equivalence classes if and only if $\mathcal{D}_W$ is finite.

When considering a minimal automaton $\mathcal{D}_W = (Q, C, q_{\mathsf{init}}, \delta, F)$, for $q \in Q$, we abusively write $q^{-1}W$ for the set $w^{-1}W$, where w is any finite word such that $\delta^*(q_{\mathsf{init}}, w) = q$ (the choice of w does not matter). We extend $\preceq$ to automaton states ($q_1 \preceq q_2$ if $q_1^{-1}W \subseteq q_2^{-1}W$).

Preorders. Let $\preceq$ be a preorder on some set B. We say that two elements $b_1, b_2 \in B$ are *comparable for* $\preceq$ if $b_1 \preceq b_2$ or $b_2 \preceq b_1$. A set $\Gamma \subseteq B$ is a *chain for* $\preceq$ (resp. *antichain for* $\preceq$) if for all $b_1, b_2 \in \Gamma$, b_1 and b_2 are (resp. are not) comparable for $\preceq$. A preorder $\preceq$ is *well-founded* if every chain for $\preceq$ contains a minimal element for $\preceq$.

Memory structures. A *(chromatic) memory structure* is a tuple $\mathcal{M} = (M, m_{\mathsf{init}}, \alpha_{\mathsf{upd}})$ where M is a possibly infinite set of *states*, $m_{\mathsf{init}} \in M$ is an *initial state*, and $\alpha_{\mathsf{upd}} \colon M \times C \to M$ is a (deterministic, complete) *update function*. It is syntactically almost the same as a deterministic automaton, except that we do not specify final states. We recover notations α^*_{upd} and Π_{m_1,m_2} (for $m_1, m_2 \in M$) from automata. We let $\mathcal{M}_{\mathsf{triv}} = (\{m_{\mathsf{init}}\}, m_{\mathsf{init}}, (m_{\mathsf{init}}, c) \mapsto m_{\mathsf{init}})$ denote the only memory structure with a single state. The *size* of a memory structure is its number of states.

Strategies. Let $\mathcal{A} = (V, V_1, V_2, E)$ be an arena and $i \in \{1, 2\}$. A *strategy of $\mathcal{P}_i$ on $\mathcal{A}$* is a function $\sigma_i \colon \mathsf{Hists}_i(\mathcal{A}) \to E$ such that for all $\gamma \in \mathsf{Hists}_i(\mathcal{A})$, $\mathsf{out}(\gamma) = \mathsf{in}(\sigma_i(\gamma))$. Given a strategy σ_i of $\mathcal{P}_i$, we say that a play $\pi = e_1 e_2 \dots$ is *consistent with σ_i* if for all finite prefixes $\gamma = e_1 \dots e_j$ of π such that $\mathsf{out}(\gamma) \in V_i$, $\sigma_i(\gamma) = e_{j+1}$. For $v \in V$, we denote by $\mathsf{Plays}(\mathcal{A}, v, \sigma_i)$ the set of plays on $\mathcal{A}$ from v that are consistent with σ_i.

For $\mathcal{M} = (M, m_{\mathsf{init}}, \alpha_{\mathsf{upd}})$ a memory structure, a strategy σ_i of $\mathcal{P}_i$ on arena $\mathcal{A}$ is *based on (memory) $\mathcal{M}$* if there exists a function $\alpha_{\mathsf{nxt}} \colon V_i \times M \to E$ such that for all $v \in V_i$, $\sigma_i(\lambda_v) = \alpha_{\mathsf{nxt}}(v, m_{\mathsf{init}})$, and for all non-empty histories $\gamma \in \mathsf{Hists}_i(\mathcal{A})$, $\sigma_i(\gamma) = \alpha_{\mathsf{nxt}}(\mathsf{out}(\gamma), \alpha^*_{\mathsf{upd}}(m_{\mathsf{init}}, \mathsf{col}^*(\gamma)))$. A strategy is *memoryless* if it is based on $\mathcal{M}_{\mathsf{triv}}$. For conciseness, we sometimes abusively assume that a strategy of $\mathcal{P}_i$ based on $\mathcal{M}$ is a function $V_i \times M \to E$.

▶ Remark 2. This *chromatic* memory model only observes the sequence of *colors* seen, and not the precise edges that are taken during a play (i.e., the current memory state is determined by the word in C^* seen, not by the history in E^*). A memory structure observing the edges is sometimes called a *chaotic* memory [18] and, as was recently shown, may allow to play optimally with fewer memory states for some objectives [8]. However, this comes at the cost of needing to specialize the transition function of the memory structure for every arena – it does not provide an *arena-independent* memory structure [5]. The chaotic memory requirements of general safety objectives are characterized in [10] while, as far as we know, the chaotic memory requirements of general and regular reachability objectives are unknown. ⌟

Optimality. Let $\mathcal{G} = (\mathcal{A} = (V, V_1, V_2, E), W)$ be a game, and $v \in V$. We say that a strategy σ_1 of $\mathcal{P}_1$ on $\mathcal{A}$ is *winning from v for W* if for all $\pi \in \mathsf{Plays}(\mathcal{A}, v, \sigma_1)$, $\mathsf{col}^\omega(\pi) \in W$.

A strategy of $\mathcal{P}_1$ is *optimal for $\mathcal{P}_1$ in $(\mathcal{A}, W)$* if it is winning from all the vertices of $\mathcal{A}$ from which $\mathcal{P}_1$ has a winning strategy. We often write *optimal for $\mathcal{P}_1$ in $\mathcal{A}$* if the objective W is clear from the context.

▶ Remark 3. We stress that this notion of optimality requires a *single* strategy to be winning from *all* the winning vertices (a property sometimes called *uniformity*). Asking for uniformity may require strategies that are more complex to implement than just requiring winning strategies from individual vertices. Still, uniformity is a common requirement (see, e.g., [14, 21]) that comes at no extra cost in many well-studied situations [13, 12]. We discuss uniformity again in Remark 6.

Note also that there is no requirement on the behavior of an optimal strategy from vertices from which no strategy is winning, as we assume that the opponent plays rationally. In particular, even if winning becomes possible due to a mistake of the opponent after starting from a non-winning vertex, an optimal strategy needs not win. ⌟

Let $\mathcal{M}$ be a memory structure and $W \subseteq C^\omega$ be an objective. We say that $\mathcal{M}$ *suffices (to play optimally) for W (resp. in finite, finitely branching, one-player arenas)* if for all (resp. finite, finitely branching, one-player) arenas $\mathcal{A}$, $\mathcal{P}_1$ has an optimal strategy based on $\mathcal{M}$ in game $(\mathcal{A}, W)$.

3 Technical overview

In this section, we start with a more in-depth discussion of the related literature. We then present our main contributions (characterization of the memory requirements of safety objectives, of reachability objectives, and the computational complexity of the related decision problems) while describing and illustrating the main concepts used in our results. Complete proofs for the three kinds of contributions are available in the full version [4].

Related works. To classify the existing literature on memory for games, we identify two axes. The first is whether they concern chaotic memory or chromatic memory. The second is how the class of objectives is defined: either in automata-theoretic terms, typically as a subclass of ω-regular languages, or in topological terms, referring to the natural topology over the set of infinite words.

The result of Dziembowski, Jurdziński, and Walukiewicz [13] applies to the whole class of Muller objectives, which specify the set of colors which appears infinitely many times. It shows that Zielonka trees [23] can be used to compute chaotic memory requirements in polynomial time. Recently, Casares [8] has shown that this characterization does not extend to chromatic memory: deciding whether there is a memory structure of size k becomes NP-complete and equivalent to minimizing *transition-based Rabin automata*. In this direction, Casares, Colcombet and Lehtinen [9] showed that computing chaotic memory requirements for Muller objectives is equivalent to minimizing good-for-games automata. A result by Bouyer, Randour, and Vandenhove [7] provides a link between the chromatic memory requirements of all ω-regular objectives (not only Muller conditions) and their representation as *transition-based parity automata*, but with less tight bounds on the minimal memory structures.

Article [6] establishes the existence of finite-memory optimal strategies from topological properties of objectives. Although general reachability and safety objectives fit into their framework, there are major differences with our work: their framework is different (they study *concurrent* games that are not played *on graphs*), and their aim is to establish the existence of finite-memory optimal strategies for many objectives, but not to understand precisely the memory requirements of some class of objectives.

Regular objectives are also mentioned in [19], where the existence of finite-memory optimal strategies is shown for Boolean combinations of objectives involving regular objectives.

In another line of works, Gimbert and Zielonka [14] gave a characterization of all payoff functions (extending objectives to a quantitative setting) for which both players have memoryless optimal strategies, implying an important lifting result: the sufficiency of memoryless strategies in finite two-player arenas is implied by the existence of memoryless optimal strategies in both players' finite one-player arenas. Bouyer et al. [5] extended this to chromatic finite memory.

The work most related to the present paper is by Colcombet, Fijalkow, and Horn [10, 11], which gives a characterization of chaotic memory requirements for general safety objectives. Their constructions strongly rely on the model of chaotic memory; indeed, as a corollary of our results, we will see that already for regular safety objectives, chromatic and chaotic memory requirements do not coincide. Our first step is to obtain a characterization of chromatic memory requirements for (general and regular) safety objectives.

3.1 Monotony and safety objectives

Let us fix an objective $W \subseteq C^\omega$. In order to play optimally for W, a memory structure $\mathcal{M}$ needs to be able to distinguish between histories that are not comparable for $\preceq_W$: indeed, if two finite words $w_1, w_2 \in C^*$ are not comparable, we can construct an arena in which the opponent chooses between playing w_1 and playing w_2, and then the correct choice has to be made between a continuation only winning after w_1, and a continuation only winning after w_2. This motivates the following definition, which we call $\mathcal{M}$-*strong-monotony*.

▶ **Definition 4** ($\mathcal{M}$-strong-monotony). *Let $W \subseteq C^\omega$ be an objective and $\mathcal{M} = (M, m_{\mathsf{init}}, \alpha_{\mathsf{upd}})$ be a memory structure. We say that W is* $\mathcal{M}$-strongly-monotone *if for all $w_1, w_2 \in C^*$, $\alpha^*_{\mathsf{upd}}(m_{\mathsf{init}}, w_1) = \alpha^*_{\mathsf{upd}}(m_{\mathsf{init}}, w_2)$ implies that w_1 and w_2 are comparable for $\preceq_W$.*

Notice also that W is $\mathcal{M}$-strongly-monotone if and only if $\overline{W}$ is $\mathcal{M}$-strongly-monotone (as being comparable for $\preceq_W$ is equivalent to being comparable for $\preceq_{\overline{W}} = \succeq_W$). Although stated differently, a property called *strong monotony* was introduced in [1] and coincides with our definition of $\mathcal{M}_{\mathsf{triv}}$-strong-monotony. We can therefore see our definition as a reformulation and a generalization to handle arbitrary memory structures, rather than only the "memoryless memory structure" $\mathcal{M}_{\mathsf{triv}}$.

The discussion above implies that for a memory $\mathcal{M}$, $\mathcal{M}$-strong-monotony is necessary for $\mathcal{M}$ to be sufficient to play optimally. Depending on the type of objective (regular or general), we specify a class of arenas in which $\mathcal{M}$-strong-monotony can already be shown to be necessary. Intuitively, regularity allows to distinguish distinct objectives with ultimately periodic words, which can be encoded into a finite arena.

▶ **Lemma 5** (Necessity of $\mathcal{M}$-strong-monotony). *Let W be an objective and $\mathcal{M}$ a memory structure.*

1. *If W is regular and $\mathcal{M}$ suffices to play optimally for W in all finite one-player arenas, then W is $\mathcal{M}$-strongly-monotone.*
2. *In the general case, if $\mathcal{M}$ suffices to play optimally for W in all finitely branching one-player arenas, then W is $\mathcal{M}$-strongly-monotone.*

Complete proofs for this section can be found in [4, Section 4].

▶ Remark 6. This is the only result relying on the "uniformity" assumption (see Remark 3). This assumption is crucial to obtain this lemma with a hypothesis about *one-player* arenas. We provide additional details about the (small) cost of requiring uniformity w.r.t. memory requirements in two-player games in [4, Section 4]. ⌟

In the case of general reachability or safety objectives, it is useful to reformulate the notion of $\mathcal{M}$-strongly-monotone objectives using chains. Given a general reachability or safety objective W, its minimal automaton $\mathcal{D}_W = (Q, C, q_{\mathsf{init}}, \delta, F)$, and a memory structure $\mathcal{M} = (M, m_{\mathsf{init}}, \alpha_{\mathsf{upd}})$, we can associate with each state $m \in M$ the set $\Gamma^W_m \subseteq Q$ of states of $\mathcal{D}_W$ that can be reached "simultaneously". Formally, for $m \in M$,

$$\Gamma^W_m = \{\delta^*(q_{\mathsf{init}}, w) \in Q \mid w \in C^*, \alpha^*_{\mathsf{upd}}(m_{\mathsf{init}}, w) = m\}.$$

We drop the superscript W if there is no ambiguity. The following property follows from the definitions.

▶ **Lemma 7.** *Let W be a general reachability or safety objective and $\mathcal{M} = (M, m_{\mathsf{init}}, \alpha_{\mathsf{upd}})$ be a memory structure. Objective W is $\mathcal{M}$-strongly-monotone if and only if for all $m \in M$, the set Γ_m is a chain for $\preceq_W$.*

Our initial definition of $\mathcal{M}$-strong-monotony required that any two finite words reaching the same state of $\mathcal{M}$ must be comparable; in this reformulation, we focus instead on the minimal automaton of W and require that states of the automaton that can be reached along with the same state of $\mathcal{M}$ are comparable.

Our first characterization states that for general safety objectives, $\mathcal{M}$-strong-monotony also implies that $\mathcal{M}$ suffices to play optimally. We state two variants of the results: in the first one, we assume that the preorder $\preceq$ induced by the objective is well-founded (which includes the regular case), and the result holds for all arenas; in the second one, we make no such assumption, but the result holds only for finitely branching arenas. We will discuss why we do not have the result with none of these hypotheses in Remark 10.

▶ **Theorem 8** (Characterization for safety). *Let W be a general safety objective, and $\mathcal{M}$ be a memory structure.*

1. *If $\preceq_W$ is well-founded (in particular, if W is regular), then $\mathcal{M}$ suffices to play optimally for W if and only if W is $\mathcal{M}$-strongly-monotone.*
2. *In the general case, $\mathcal{M}$ suffices to play optimally for W in all finitely branching arenas if and only if W is $\mathcal{M}$-strongly-monotone.*

Proof sketch. We provide an overview of the proof of Theorem 8 (complete proof in [4, Section 4]). We discuss here the sufficiency of $\mathcal{M}$-strong-monotony (the necessity is easier and was stated in Lemma 5).

Let $\mathcal{M} = (M, m_{\mathsf{init}}, \alpha_{\mathsf{upd}})$ and let $\mathcal{D}_W = (Q, C, q_{\mathsf{init}}, \delta, F)$ be the minimal automaton of W. We assume that W is $\mathcal{M}$-strongly-monotone. Let $\mathcal{A} = (V, V_1, V_2, E)$ be an arena. As per the hypotheses, we require that $\preceq$ is well-founded or that $\mathcal{A}$ is finitely branching.

We want to build a strategy σ optimal for $\mathcal{P}_1$ in $\mathcal{A}$ such that $\sigma\colon V_1 \times M \to E$ is based on $\mathcal{M}$. The key to the proof is to understand the following sets of states of $\mathcal{D}_W$ in order to know what to play in each pair $(v, m) \in V_1 \times M$. For $v \in V$, $m \in M$, we define

$$Q_{v,m} = \{q \in \Gamma_m \mid \mathcal{P}_1 \text{ has a winning strategy for objective } q^{-1}W \text{ from } v\}.$$

States in $Q_{v,m}$ are states of $\mathcal{D}_W$ that could be reached while the memory state is m, by definition of Γ_m. Moreover, $\mathcal{M}$-strong-monotony tells us that each Γ_m is a chain for $\preceq$ (Lemma 7), so each $Q_{v,m}$ is too.

For given $v \in V_1$ and $m \in M$, we distinguish three possibilities.

- If $Q_{v,m}$ is empty, then this means that there is no state of Γ_m for which $\mathcal{P}_1$ can win from v (i.e., for all $q \in \Gamma_m$, $\mathcal{P}_1$ has no winning strategy for $q^{-1}W$ from v). In this case, we can define $\sigma(v, m)$ arbitrarily, as there is no hope to win.
- If $Q_{v,m}$ has a minimum $q_{v,m}$ for $\preceq$, then this minimum represents the worst (for $\preceq$) state of Γ_m for which $\mathcal{P}_1$ still has a winning strategy. We define $\sigma(v, m)$ as the edge played by such a winning strategy. Intuitively, this is the most robust way to play as it is a winning move for the worst possible state of Γ_m for which winning is possible. Notice that a non-empty $Q_{v,m}$ always has a minimum when $\preceq$ is well-founded.
- In case $Q_{v,m}$ is non-empty and has no minimum, then $\preceq$ is not well-founded, so we work under the hypothesis that $\mathcal{A}$ is finitely branching. We can then consider, for each $q \in Q_{v,m}$, the set of edges E_q that can be taken from v to win for $q^{-1}W$. Each E_q is finite and non-empty, and they are ordered for inclusion. We can then show that their intersection is non-empty, so there is an edge that can be taken in v to win for all $q \in Q_{v,m}$. We define $\sigma(v, m)$ as such an edge.

We have now defined a strategy σ based on $\mathcal{M}$ that makes local choices that are played by winning strategies for as many states of Γ_m (where m is the current memory state) as possible. Using the fact that W is a general safety condition, one can prove that this strategy is in fact optimal. ◀

A corollary of this characterization, by comparing to the characterization for chaotic memory in [10], is that chromatic and chaotic memory requirements differ already for regular safety objectives. We provide an instructive example below. Note that this provides a new simple kind of counterexample to Kopczyński's conjecture [18], which Casares [8] had already falsified with a Muller objective.

▶ **Example 9.** Let $C = \{a, b, c, d\}$. We consider the regular language recognized by the finite automaton $\mathcal{D}$ depicted in Figure 1 (left). It accepts the finite words that first see both a and b (in any order, possibly interspersed with c's and d's), and then see both c and d (in any order, possibly interspersed with a's and b's). This language can be described by the regular expression $C^*(aC^*b \mid bC^*a)C^*(cC^*d \mid dC^*c)C^*$. We write W for the induced regular safety objective: $W = \mathsf{Safe}(\mathcal{L}(\mathcal{D}))$.

The main claim is that the chaotic memory requirements for W are two states, which is easily obtained from the existing characterization [10] (this is the size of a maximal antichain for $\preceq$), while the chromatic requirements for W are three states. We depict a memory structure $\mathcal{M}$ with three states which makes W $\mathcal{M}$-strongly-monotone in Figure 1 (right). To check that W is indeed $\mathcal{M}$-strongly-monotone, we have to check that there is no pair of words $w_1, w_2 \in C^*$ such that w_1 and w_2 reach the same state of $\mathcal{M}$, but reach non-comparable states in $\mathcal{D}$. The only two pairs of non-comparable states in $\mathcal{D}$ are q_a and q_b, and q_c and q_d (besides these, states are ordered for $\preceq$ from right to left). We can check that for this choice of $\mathcal{M}$, $\Gamma_{m_1} = \{q_{\mathsf{init}}, q_a\}$, $\Gamma_{m_2} = \{q_b, q_{ab}, q_d, q_{cd}\}$, $\Gamma_{m_3} = \{q_b, q_{ab}, q_c, q_{cd}\}$. As these are all chains for $\preceq$, we have that W is $\mathcal{M}$-strongly-monotone.

It is not possible to find a chromatic memory structure $\mathcal{M}$ with *two* states which makes W $\mathcal{M}$-strongly-monotone (this can be checked by trying to assign transitions to two states while distinguishing non-comparable states, and observing that all cases fail). ⌟

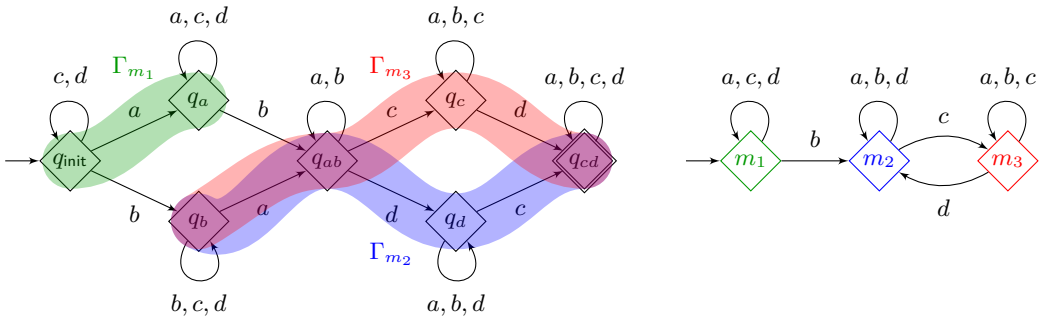

Figure 1 Example 9: automaton $\mathcal{D}$ (left) and a minimal memory structure $\mathcal{M}$ (right) such that $\mathsf{Reach}(\mathcal{L}(\mathcal{D}))$ and $\mathsf{Safe}(\mathcal{L}(\mathcal{D}))$ are $\mathcal{M}$-strongly-monotone. In figures, diamonds are used to depict automaton states and memory states, and accepting states are depicted with a double border.

To conclude this section, we discuss why, with neither the well-foundedness hypothesis nor the finitely branching hypothesis from Theorem 8, we cannot expect such a characterization.

▶ Remark 10. If the prefix preorder of an objective W is not well-founded, then there is an infinite decreasing sequence of finite words $w_1 \succ w_2 \succ \dots$ in C^*. This means that for all $i \geq 1$, there is $w'_i \in C^\omega$ such that $w_i w'_i \in W$, but for $j > i$, $w_j w'_i \notin W$. We can then build the infinitely branching arena depicted in Figure 2 in which $\mathcal{P}_2$ first chooses a word w_j, and $\mathcal{P}_1$ can win by playing a word w'_i with $i \geq j$. This requires infinite memory, even if W is $\mathcal{M}_{\mathsf{triv}}$-strongly-monotone. ⌟

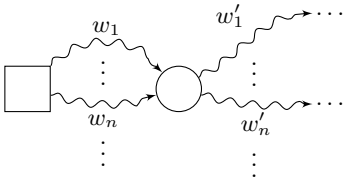

Figure 2 Infinite branching arena in which $\mathcal{P}_1$ needs memory beyond the $\mathcal{M}$-strong-monotony property in Remark 10. In figures, circles (resp. squares) represent arena vertices controlled by $\mathcal{P}_1$ (resp. $\mathcal{P}_2$), i.e., in V_1 (resp. V_2). Squiggly arrows indicate a sequence of edges.

3.2 Capturing progress and reachability objectives

To play optimally for general and regular reachability objectives with a memory $\mathcal{M}$, $\mathcal{M}$-strong-monotony is necessary (Lemma 5) but not enough: the following example shows that the memory structure must keep track of *progress*.

▶ **Example 11.** Let $C = \{a, b\}$. We consider the regular language $b^*a^+bC^*$ of words that have to see at least one a, followed by at least one b. This language is recognized by the finite automaton $\mathcal{D}$ in Figure 3 (left). We write W for the induced regular reachability objective: $W = \mathsf{Reach}(\mathcal{L}(\mathcal{D}))$.

In the arena in Figure 3 (center), $\mathcal{P}_1$ may win by starting a play with ab, but not without memory. The intuition is that playing a first makes some progress (it reaches an automaton state with more winning continuations), but is not sufficient to win, even if repeated. Therefore, in our memory structures, if a word makes some progress but without guaranteeing the win when repeated, we want the memory state to change upon reading that word. The memory structure in Figure 3 (right) is sufficient for W; in particular, seeing the first a, which makes progress from q_{init} to q_a, changes the memory state. ⌟

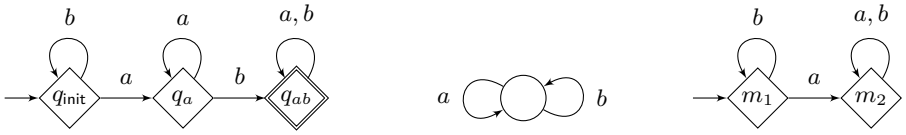

Figure 3 Example 11: automaton $\mathcal{D}$ (left), an arena requiring memory for $\mathsf{Reach}(\mathcal{L}(\mathcal{D}))$ (center), and a minimal sufficient memory structure (right).

We formalize this intuition in the following definition, which is a generalization of the *progress-consistency* property [3]. Notation Π_{m_1,m_2}, representing the finite words read from memory state m_1 to memory state m_2, was defined in Section 2.

▶ **Definition 12** ($\mathcal{M}$-progress-consistency)**.** *Let W be an objective and $\mathcal{M} = (M, m_{\mathsf{init}}, \alpha_{\mathsf{upd}})$ be a memory structure. We say that W is* $\mathcal{M}$-progress-consistent *if for all $m \in M$, for all $w_1 \in \Pi_{m_{\mathsf{init}},m}$, for all $w_2 \in \Pi_{m,m}$,*

$$w_1 \prec w_1w_2 \implies w_1(w_2)^\omega \in W.$$

Intuitively, this says that if it is possible to come back to the same memory state while reading a "word that makes progress" (i.e., that improves our situation by putting us in a position with more winning continuations), then repeating this word infinitely often from that point onward must be winning. The notion of $\mathcal{M}_{\mathsf{triv}}$-progress-consistency corresponds to the previous definition of *progress-consistency* [3].

The discussion above shows that $\mathcal{M}$-progress-consistency is necessary for a memory structure $\mathcal{M}$ to be sufficient to play optimally. As for $\mathcal{M}$-strong-monotony, we distinguish the regular case from the general case.

▶ **Lemma 13** (Necessity of $\mathcal{M}$-progress-consistency). *Let W be an objective and $\mathcal{M}$ a memory structure.*

1. *If W is regular and $\mathcal{M}$ suffices to play optimally for W in all finite one-player arenas, then W is $\mathcal{M}$-progress-consistent.*
2. *In the general case, if $\mathcal{M}$ suffices to play optimally for W in all finitely branching one-player arenas, then W is $\mathcal{M}$-progress-consistent.*

Complete proofs for this section can be found in [4, Section 5]. The following example should help the reader form the right intuition about $\mathcal{M}$-progress-consistency.

▶ **Example 14.** Let $C = \{a, b\}$. We consider the regular language of words containing *ababa* as a (non-necessarily contiguous) subword, recognized by the finite automaton $\mathcal{D}$ in Figure 4 (left). We consider the memory structure $\mathcal{M}$ remembering whether a or b was last seen, depicted in Figure 4 (right). The regular reachability objective $W = \mathsf{Reach}(\mathcal{L}(\mathcal{D}))$ is $\mathcal{M}$-progress-consistent. Indeed, let us first consider $m = m_b$ in the definition of $\mathcal{M}$-progress-consistency. A finite word w_1 reaching m_b in $\mathcal{M}$ necessarily reaches q_{init}, q_{ab}, or q_{abab} in Q (excluding the final state from the reasoning, as no progress is possible from it). After w_1, words w_2 that both (i) make progress ($w_1 \prec w_1w_2$) and (ii) are a cycle on m_b necessarily see both a and b. Therefore, $w_1(w_2)^\omega$ is always a winning word. The same reasoning holds for $m = m_a$. Notice that the memory states from the memory structure do not carry enough information to ascertain when a word of the language has been seen (i.e., when the game is won).

The upcoming Theorem 16 implies that $\mathcal{M}$ suffices to play optimally for $\mathcal{P}_1$. ⌟

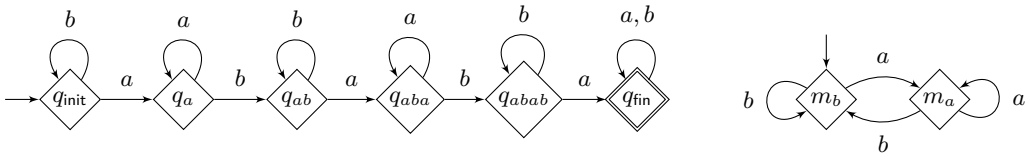

Figure 4 Example 14: automaton $\mathcal{D}$ (left) and memory structure $\mathcal{M}$ (right).

This need to capture *progress* was not necessary to understand the memory requirements of safety objectives, which may be explained by the following reasoning.

▶ Remark 15. Unlike general reachability objectives, all general safety objectives are $\mathcal{M}_{\mathsf{triv}}$-progress-consistent. Here is a proof of this statement. Let $W \subseteq C^\omega$ be a general safety objective. Let $w_1, w_2 \in \Pi^{\mathcal{M}_{\mathsf{triv}}}_{m_{\mathsf{init}}, m_{\mathsf{init}}} = C^*$ be such that $w_1 \prec w_1w_2$. This implies that w_1w_2, and therefore w_1, have a non-empty set of winning continuations. Assume by contradiction that $w_1(w_2)^\omega \notin W$. As W is a general safety objective, there is a smallest $n \geq 1$ such that $w_1(w_2)^n$ has no winning continuation. Hence, $w_1(w_2)^{n-1}$ still has some winning continuations, so $w_1(w_2)^n \prec w_1(w_2)^{n-1}$. This is a contradiction, as $w_1 \prec w_1w_2$ implies that $w_1(w_2)^{n-1} \preceq w_1w_2(w_2)^{n-1} = w_1(w_2)^n$ by Lemma 1. This property is, at least intuitively, a reason hinting that the memory requirements of safety objectives are lower and easier to understand than those for their complement reachability objective. ⌟

We have now discussed two necessary properties for a memory $\mathcal{M}$ to be sufficient to play optimally for an objective. For regular reachability objectives, it appears that the conjunction of these two properties is also sufficient.

▶ **Theorem 16** (Characterization for reachability). *Let W be a regular reachability objective and $\mathcal{M}$ be a finite memory structure. Memory $\mathcal{M}$ suffices to play optimally for W if and only if W is $\mathcal{M}$-strongly-monotone and $\mathcal{M}$-progress-consistent.*

Proof sketch. We provide an overview of the proof of Theorem 16 (complete proof in [4, Section 5]). We discuss here the sufficiency of $\mathcal{M}$-strong-monotony and $\mathcal{M}$-progress-consistency (their necessity is easier and was stated in Lemmas 5 and 13).

Let $\mathcal{D}_W = (Q, C, q_{\mathsf{init}}, \delta, F)$ be the minimal automaton of W (which is finite as W is regular), and $\mathcal{M} = (M, m_{\mathsf{init}}, \alpha_{\mathsf{upd}})$. We assume that W is $\mathcal{M}$-strongly-monotone and $\mathcal{M}$-progress-consistent. Let $\mathcal{A} = (V, V_1, V_2, E)$ be a (possibly infinite) arena. We construct an optimal strategy based on memory $\mathcal{M}$, using the same idea as in the proof for safety objectives (Theorem 8): we once again consider a strategy based on $\mathcal{M}$ making choices that are "locally optimal". We then show, thanks to our hypotheses ($\mathcal{M}$-strong-monotony and $\mathcal{M}$-progress-consistency), that this strategy must be optimal.

For $v \in V_1$, $m \in M$, we define

$$q_{v,m} = \min_{\preceq}\{q \in \Gamma_m \mid \mathcal{P}_1 \text{ has a winning strategy for objective } q^{-1}W \text{ from } v\}.$$

Every set Γ_m is a chain using $\mathcal{M}$-strong-monotony (Lemma 7) and is finite since $\mathcal{D}_W$ is finite. Hence, the minimum $q_{v,m}$ exists (except when the set is empty, but that means that the game cannot be won anymore – we ignore this case). Let $\sigma_{v,m}$ be a strategy winning for $q_{v,m}^{-1}W$ from v.

Now, just like for the proof for safety, we want to define $\sigma(v, m)$ as the first edge taken by $q_{v,m}$ from v – we play locally reasonable edges played by good strategies and hope that this creates a "globally" optimal strategy. However, this does not work in general, as any choice for the strategies $\sigma_{v,m}$ may not be good: indeed, such strategies may be winning, but may make unnecessary moves delaying the achievement of the objective. For instance, in the arena of Figure 3, a strategy playing bab^ω is winning for $q_{\mathsf{init}}^{-1}W$, but not as fast as possible (it takes three moves to create a word in $\mathcal{L}(\mathcal{D})$, while it is possible to do it in two moves). If, by imitating the first move of this strategy, we define $\sigma(v, m_1) = (v, b, v)$, we then get stuck and σ plays the losing word b^ω.

A way to remedy this is by formally defining the "time" taken by a strategy to guarantee a win, and choosing strategies $\sigma_{v,m}$ that win in the least time. When considering infinite two-player arenas, this time has to be defined using ordinals. If we do this, it is possible (though still quite involved) to show that σ defined as above is indeed optimal, thanks to $\mathcal{M}$-progress-consistency. ◀

▶ Remark 17. Unlike safety objectives, our characterization is only shown to hold for *regular* reachability objectives. We discuss in [4, Section 5] why our proof technique does not apply to general reachability objectives (even with $\preceq$ well-founded and finite branching of the arenas). ⌟

For objectives beyond reachability and safety, $\mathcal{M}$-strong-monotony and $\mathcal{M}$-progress-consistency may not imply the sufficiency of $\mathcal{M}$ to play optimally. For instance, with $C = \{a, b\}$, let us consider the objective

$$W = \{w \in C^\omega \mid a \text{ and } b \text{ are both seen infinitely often}\},$$

which is ω-regular (it can be recognized by a *deterministic Büchi automaton* with two states), but is not a general reachability nor safety objective. Objective W is $\mathcal{M}_{\mathsf{triv}}$-strongly-monotone and $\mathcal{M}_{\mathsf{triv}}$-progress-consistent, but $\mathcal{M}_{\mathsf{triv}}$ does not suffice to play optimally.

Lift for regular objectives. As a by-product of our results, we observe that for regular objectives, our characterizations deal with arbitrary arenas of any cardinality, but the properties used in the characterizations are already necessary in *finite one-player* arenas. This means that strategy-wise, to accomplish a regular objective, all the complexity already appears in finite graphs with no opponent. For the specific class of regular objectives that we study, this strengthens so-called *one-to-two-player lifts* from the literature [14, 5].

▶ **Theorem 18** (Finite-to-infinite, one-to-two-player lift). *Let W be a regular (reachability or safety) objective and $\mathcal{M}$ be a finite memory structure. Memory $\mathcal{M}$ suffices to play optimally for W (in all arenas) if and only if $\mathcal{M}$ suffices to play optimally for W in finite one-player arenas.*

Proof. The implication from left-to-right holds as this is the same property quantified over fewer arenas. We argue the other implication for each case.

For regular safety objectives W, we showed that if $\mathcal{M}$ suffices in finite one-player arenas, then W is $\mathcal{M}$-strongly-monotone (by Lemma 5 as W is regular), which implies that $\mathcal{M}$ suffices in all arenas (by Theorem 8 as W is a safety condition with a well-founded preorder).

For regular reachability objectives W, we showed that if $\mathcal{M}$ suffices in finite one-player arenas, then W is $\mathcal{M}$-strongly-monotone and $\mathcal{M}$-progress-consistent (by Lemmas 5 and 13 as W is regular), which implies that $\mathcal{M}$ suffices in all arenas (by Theorem 16 as W is a regular reachability objective). ◀

3.3 The complexity of finding small memory structures

We finally discuss the computational complexity of finding small memory structures for regular objectives. We formalize the question as two decision problems: given a regular reachability or safety objective, how much memory is required to play optimally for this objective?

Memory-Safe

Input: A finite automaton $\mathcal{D}$ inducing the regular safety objective $W = \mathsf{Safe}(\mathcal{L}(\mathcal{D}))$ and an integer $k \in \mathbb{N}$.

Question: Does there exist a memory structure $\mathcal{M}$ of size at most k which suffices to play optimally for W?

Memory-Reach

Input: A finite automaton $\mathcal{D}$ inducing the regular reachability objective $W = \mathsf{Reach}(\mathcal{L}(\mathcal{D}))$ and an integer $k \in \mathbb{N}$.

Question: Does there exist a memory structure $\mathcal{M}$ of size at most k which suffices to play optimally for W?

It follows from our characterizations (Theorems 8 and 16) that Memory-Safe is equivalent to asking whether there is a memory structure $\mathcal{M}$ of size at most k such that $\mathsf{Safe}(\mathcal{L}(\mathcal{D}))$ is $\mathcal{M}$-strongly-monotone, and Memory-Reach whether there is a memory structure $\mathcal{M}$ of size at most k such that $\mathsf{Reach}(\mathcal{L}(\mathcal{D}))$ is $\mathcal{M}$-strongly-monotone and $\mathcal{M}$-progress-consistent.

▶ Remark 19. The way k is encoded (in binary or in unary) has no impact on the complexity. Indeed, the input consists of the number k together with a (deterministic) automaton describing the objective. Since the automaton is an upper bound on the memory requirements (for both Memory-Safe and Memory-Reach), the problem is non-trivial only when k is smaller than the size of the automaton. Therefore, the size of the input is dominated by the size of the automaton in the non-trivial cases. ⌟

▶ **Theorem 20** (Complexity of Memory-Safe and Memory-Reach). *Both Memory-Safe and Memory-Reach are NP-complete.*

For NP-hardness, we construct a reduction from the Hamiltonian cycle problem which works for both Memory-Safe and Memory-Reach. Complete proofs for this section can be found in [4, Section 6].

Our main insight is to reformulate the notion of $\mathcal{M}$-strong-monotony (NP-membership of Memory-Safe follows from this reformulation). Let $W = \mathsf{Safe}(\mathcal{L}(\mathcal{D}))$ be a regular objective and $\mathcal{M} = (M, m_{\mathsf{init}}, \alpha_{\mathsf{upd}})$ be a memory structure. In Example 9, we have seen how to go from a memory structure $\mathcal{M}$ such that W is $\mathcal{M}$-strongly-monotone to a covering of the states of $\mathcal{D}$ by chains of states. We formulate exactly the requirements for such coverings in order to have a point of view equivalent to $\mathcal{M}$-strong-monotony. For $\Gamma \subseteq Q$ a set of automaton states and $c \in C$ a color, we define $\delta(\Gamma, c) = \{\delta(q, c) \mid q \in \Gamma\}$.

▶ **Definition 21** (Monotone decomposition). *Let $\mathcal{D} = (Q, C, q_{\mathsf{init}}, \delta, F)$ be an automaton. We say that the sets $\Gamma_1, \dots, \Gamma_k \subseteq Q$ form a* monotone decomposition of $\mathcal{D}$ *if*
(a) $Q = \bigcup_{i=1}^{k} \Gamma_i$,
(b) *for all $c \in C$, for all $i \in \{1, \dots, k\}$, there is $j \in \{1, \dots, k\}$ such that $\delta(\Gamma_i, c) \subseteq \Gamma_j$, and*
(c) *for all $i \in \{1, \dots, k\}$, Γ_i is a chain for $\preceq$.*

Note that the sets Γ_i do not have to be disjoint (as was illustrated in Example 9). If we only consider requirements (a) and (b) of this definition, we recover the definition of an *admissible decomposition*, which can be used to quotient an automaton [15]. Here, we add the additional requirement (c) that each set of states is a chain for $\preceq$. Note that there always exists an admissible decomposition with just one set (by taking $\Gamma_1 = Q$), but finding a small *monotone* decomposition may not be so easy. This point of view in terms of monotone decompositions turns out to be equivalent to our initial point of view in terms of $\mathcal{M}$-strong-monotony in the following sense.

▶ **Lemma 22.** *Let $\mathcal{D}$ be an automaton and W be equal to $\mathsf{Safe}(\mathcal{L}(\mathcal{D}))$ or $\mathsf{Reach}(\mathcal{L}(\mathcal{D}))$. Automaton $\mathcal{D}$ admits a monotone decomposition with k sets if and only if W is $\mathcal{M}$-strongly-monotone for some memory structure $\mathcal{M}$ of size k.*

It is instructive to reformulate the characterization of *chaotic* memory requirements from [10]: the original phrasing was that the number of memory states necessary and sufficient to play optimally for the safety objective W is the size of the largest antichain of $\preceq_W$. Using our terminology and Dilworth's theorem, it is equivalent to the smallest number of chains required to cover all states; that is, decompositions satisfying (a) and (c) in Definition 21, but not necessarily (b). Hence, it is smaller in general.

We finish this section with an overview of the proof of Theorem 20 (complete proofs in [4, Section 6]).

Proof sketch of Theorem 20. We first discuss membership in NP, and then NP-hardness.

Membership in NP. Problem Memory-Safe with inputs $\mathcal{D}$ and k was shown in Theorem 8 to be equivalent to the existence of a memory structure $\mathcal{M}$ of size k such that W is $\mathcal{M}$-strongly-monotone. This second problem is itself equivalent by Lemma 22 to the existence of a monotone decomposition of $\mathcal{D}$ with k sets. A monotone decomposition is a polynomial-size witness, and checking whether k sets of states form a monotone decomposition is done in polynomial time by checking the three conditions in the definition. This shows NP-membership of Memory-Safe.

For Memory-Reach, we use memory structures as polynomial-size witnesses. Let $\mathcal{D}$ be a finite automaton and $k \in \mathbb{N}$. Given a memory structure $\mathcal{M}$ of size k, we want to decide in polynomial time whether objective $\mathsf{Reach}(\mathcal{L}(\mathcal{D}))$ is $\mathcal{M}$-strongly-monotone and $\mathcal{M}$-progress-consistent. We have discussed how to check $\mathcal{M}$-strong-monotony in polynomial time through monotone decompositions. Checking $\mathcal{M}$-progress-consistency in polynomial time is slightly more involved and is described in [4, Section 6]: we reduce $\mathcal{M}$-progress-consistency to checking a polynomial number of emptiness queries of intersections of regular languages recognized by deterministic finite automata.

NP-hardness. As mentioned above, we prove NP-hardness of Memory-Safe using a reduction from the Hamiltonian cycle problem. The proof also applies to Memory-Reach, but we move this discussion to [4, Section 6]. In the following, a *(directed) graph* is a tuple $G = (V, E)$ with $E \subseteq V \times V$. A *Hamiltonian cycle of* G is a sequence $(u_1, \dots, u_n)$ in which each state of V appears exactly once, $(u_i, u_{i+1}) \in E$ for all i, $1 \leq i < n$, and $(u_n, u_1) \in E$.

We start from a directed graph $G = (V, E)$ and we intend to build an automaton $\mathcal{D}_G$ such that G has a Hamiltonian cycle if and only if $\mathcal{D}_G$ has a monotone decomposition with k sets, for a well-chosen k. We write $|V| = n$ and $|E| = m$. We assume $m \geq n$, otherwise G cannot have a Hamiltonian cycle. We define $\mathsf{Automaton}(G)$ as the automaton $(Q, \Sigma, \delta, q_{\mathsf{init}}, F)$ with $Q = V \uplus E$, $\Sigma = \{\mathsf{in}, \mathsf{out}\}$, and transitions such that for $v \in V$, $\delta(v, \mathsf{in}) = \delta(v, \mathsf{out}) = v$, and for $e = (v_1, v_2) \in E$, $\delta(e, \mathsf{in}) = v_1$ and $\delta(e, \mathsf{out}) = v_2$. We ignore q_{init} and F at the moment. This definition is inspired from a reduction in [2] (although the rest of the proof is different).

We also consider the *cycle graph with* n *vertices* $C_n = (V_C, E_C)$, with $V_C = \{v_1^C, \dots, v_n^C\}$ and $E_C = \{e_1^C, \dots, e_n^C\}$ such that $e_i^C = (v_i^C, v_{i+1}^C)$ for $1 \leq i < n$ and $e_n = (v_n^C, v_1^C)$. We now consider an automaton $\mathcal{D}_G = (Q, \Sigma, \delta, q_{\mathsf{init}}, F)$ based on the disjoint union $\mathsf{Automaton}(C_n) \uplus \mathsf{Automaton}(G)$ along with three extra states q_{init}, $\bot$, and $\top$. We illustrate this part of the construction in Figure 5.

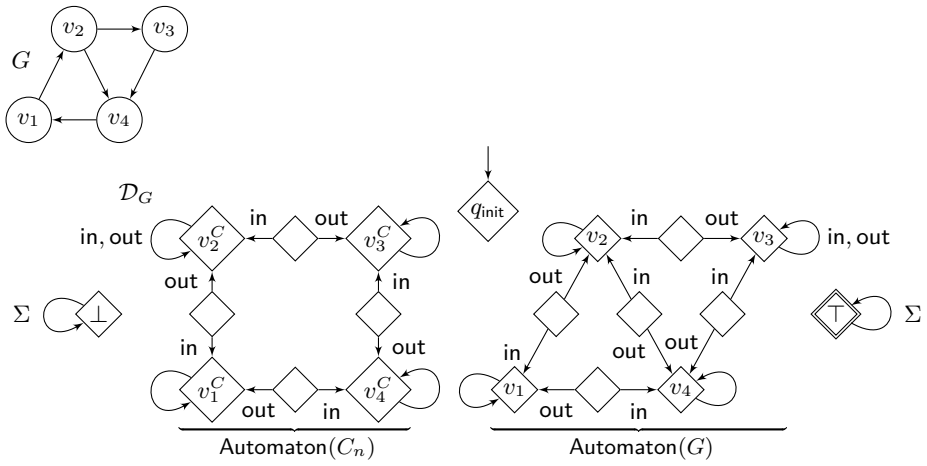

Figure 5 Illustration of automaton $\mathcal{D}_G$ starting from a graph G with four vertices. This is only a part of the full construction in [4, Section 6] to give an overview of the proof.

What is now missing is a way to induce an interesting ordering $\preceq$ – intuitively, we want $\bot$ to be the smallest state, $\top$ to be the largest, and all automaton states corresponding to vertices (resp. edges) of $\mathsf{Automaton}(C_n)$ to be smaller than all automaton states corresponding to vertices (resp. edges) of $\mathsf{Automaton}(G)$, while making all other pairs of states non-comparable. We can get this ordering by adding a letter to Σ for each state of $\mathcal{D}_G$ and defining the right transitions from q_{init} and to $\bot$ and $\top$.

In this way, we have that chains of $\mathcal{D}_G$ for $\preceq$ have at most 4 states, and chains with four states contain either $\bot$, $\top$, a vertex of C_n and a vertex of G, or $\bot$, $\top$, an edge of C_n and an edge of G. Moreover, the largest antichain of $\mathcal{D}_G$ for $\preceq$ has $n+m+1$ elements and is achieved by $V \cup E \cup \{q_{\mathsf{init}}\}$. By a counting argument, it is then possible to cover all states with $n+m+1$ chains if only if every vertex (resp. edge) of C_n is in a chain with one vertex (resp. edge) of G. A covering with $n+m+1$ chains therefore induces a bijection between V_C and V and an injection from E_C to E. To form a monotone decomposition, these chains still have to satisfy condition b from Definition 21. If it is possible to find $n+m+1$ such chains, we can show by reading in and out from chains containing edges that the cycle on C_n transfers to a Hamiltonian cycle on G. Reciprocally, if G has a Hamiltonian cycle, then we can find a natural correspondence between vertices (resp. edges) of C_n and vertices (resp. edges) of G that allows to define a monotone decomposition with $n+m+1$ sets. We have that G has a Hamiltonian cycle if and only if $\mathcal{D}_G$ has a monotone decomposition in $k = n+m+1$ sets. ◀

4 Conclusion

We have characterized the minimal memory structures sufficient to play optimally for regular reachability and safety objectives. In doing so, we were able to prove that related decision problems about regular objectives were NP-complete. Our characterizations were encoded into a SAT solver that automatically generates a minimal memory structure given a finite automaton as an input (link in Section 1).

This article can be seen as one step toward understanding more generally the (chromatic or chaotic) memory requirements of all ω-regular objectives, as well as synthesizing minimal memory structures for them. The chaotic memory requirements of regular reachability objectives are still unknown, as well as the chromatic memory requirements of larger classes of ω-regular objectives (such as, e.g., the objectives recognized by deterministic Büchi automata).

References

1 Alessandro Bianco, Marco Faella, Fabio Mogavero, and Aniello Murano. Exploring the boundary of half-positionality. *Annals of Mathematics and Artificial Intelligence*, 62(1-2):55–77, 2011. doi:10.1007/s10472-011-9250-1.

2 Kellogg S. Booth. Isomorphism testing for graphs, semigroups, and finite automata are polynomially equivalent problems. *SIAM Journal on Computing*, 7(3):273–279, 1978. doi:10.1137/0207023.

3 Patricia Bouyer, Antonio Casares, Mickael Randour, and Pierre Vandenhove. Half-positional objectives recognized by deterministic Büchi automata. In Bartek Klin, Sławomir Lasota, and Anca Muscholl, editors, *Proceedings of the 33rd International Conference on Concurrency Theory, CONCUR 2022, Warsaw, Poland, September 12–16, 2022*, volume 243 of *LIPIcs*, pages 20:1–20:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.CONCUR.2022.20.

4 Patricia Bouyer, Nathanaël Fijalkow, Mickael Randour, and Pierre Vandenhove. How to play optimally for regular objectives? *CoRR*, abs/2210.09703, 2022. doi:10.48550/arXiv.2210.09703.

5 Patricia Bouyer, Stéphane Le Roux, Youssouf Oualhadj, Mickael Randour, and Pierre Vandenhove. Games where you can play optimally with arena-independent finite memory. *Logical Methods in Computer Science*, 18(1), 2022. doi:10.46298/lmcs-18(1:11)2022.

6 Patricia Bouyer, Stéphane Le Roux, and Nathan Thomasset. Finite-memory strategies in two-player infinite games. In Florin Manea and Alex Simpson, editors, *Proceedings of the 30th EACSL Annual Conference on Computer Science Logic, CSL 2022, Göttingen, Germany, February 14–19, 2022*, volume 216 of *LIPIcs*, pages 8:1–8:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.CSL.2022.8.

7 Patricia Bouyer, Mickael Randour, and Pierre Vandenhove. Characterizing omega-regularity through finite-memory determinacy of games on infinite graphs. *TheoretiCS*, 2:1–48, 2023. doi:10.46298/theoretics.23.1.

8 Antonio Casares. On the minimisation of transition-based Rabin automata and the chromatic memory requirements of Muller conditions. In Florin Manea and Alex Simpson, editors, *Proceedings of the 30th EACSL Annual Conference on Computer Science Logic, CSL 2022, Göttingen, Germany, February 14–19, 2022*, volume 216 of *LIPIcs*, pages 12:1–12:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.CSL.2022.12.

9 Antonio Casares, Thomas Colcombet, and Karoliina Lehtinen. On the size of good-for-games Rabin automata and its link with the memory in Muller games. In Mikołaj Bojańczyk, Emanuela Merelli, and David P. Woodruff, editors, *Proceedings of the 49th International Colloquium on Automata, Languages, and Programming, ICALP 2022, Paris, France, July 4–8, 2022*, volume 229 of *LIPIcs*, pages 117:1–117:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ICALP.2022.117.

10 Thomas Colcombet, Nathanaël Fijalkow, and Florian Horn. Playing safe. In Venkatesh Raman and S. P. Suresh, editors, *Proceedings of the 34th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2014, New Delhi, India, December 15–17, 2014*, volume 29 of *LIPIcs*, pages 379–390. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2014. doi:10.4230/LIPIcs.FSTTCS.2014.379.

11 Thomas Colcombet, Nathanaël Fijalkow, and Florian Horn. Playing safe, ten years later. *CoRR*, abs/2212.12024, 2022. doi:10.48550/arXiv.2212.12024.

12 Thomas Colcombet and Damian Niwiński. On the positional determinacy of edge-labeled games. *Theoretical Computer Science*, 352(1-3):190–196, 2006. doi:10.1016/j.tcs.2005.10.046.

13 Stefan Dziembowski, Marcin Jurdziński, and Igor Walukiewicz. How much memory is needed to win infinite games? In *Proceedings of the 12th Annual IEEE Symposium on Logic in Computer Science, LICS 1997, Warsaw, Poland, June 29 – July 2, 1997*, pages 99–110. IEEE Computer Society, 1997. doi:10.1109/LICS.1997.614939.

14 Hugo Gimbert and Wieslaw Zielonka. Games where you can play optimally without any memory. In Martín Abadi and Luca de Alfaro, editors, *Proceedings of the 16th International Conference on Concurrency Theory, CONCUR 2005, San Francisco, CA, USA, August 23–26, 2005*, volume 3653 of *Lecture Notes in Computer Science*, pages 428–442. Springer, 2005. doi:10.1007/11539452_33.

15 Abraham Ginzburg and Michael Yoeli. Products of automata and the problem of covering. *Transactions of the American Mathematical Society*, 116:253–266, 1965. URL: http://www.jstor.org/stable/1994117.

16 Yuri Gurevich and Leo Harrington. Trees, automata, and games. In Harry R. Lewis, Barbara B. Simons, Walter A. Burkhard, and Lawrence H. Landweber, editors, *Proceedings of the 14th Annual ACM Symposium on Theory of Computing, STOC 1982, San Francisco, CA, USA, May 5–7, 1982*, pages 60–65. ACM, 1982. doi:10.1145/800070.802177.

17 Alexey Ignatiev, António Morgado, and João Marques-Silva. PySAT: A Python toolkit for prototyping with SAT oracles. In Olaf Beyersdorff and Christoph M. Wintersteiger, editors, *Proceedings of the 21st International Conference on the Theory and Applications of Satisfiability Testing, SAT 2018, Held as Part of FloC 2018, Oxford, UK, July 9–12, 2018*, volume 10929 of *Lecture Notes in Computer Science*, pages 428–437. Springer, 2018. doi:10.1007/978-3-319-94144-8_26.

18 Eryk Kopczyński. *Half-positional Determinacy of Infinite Games*. PhD thesis, Warsaw University, 2008.

19 Stéphane Le Roux, Arno Pauly, and Mickael Randour. Extending finite-memory determinacy by Boolean combination of winning conditions. In Sumit Ganguly and Paritosh K. Pandya, editors, *Proceedings of the 38th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2018, Ahmedabad, India, December 11–13, 2018*, volume 122 of *LIPIcs*, pages 38:1–38:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.FSTTCS.2018.38.

20 Anil Nerode. Linear automaton transformations. *Proceedings of the American Mathematical Society*, 9(4):541–544, 1958. doi:10.2307/2033204.

21 Pierre Ohlmann. Characterizing positionality in games of infinite duration over infinite graphs. *TheoretiCS*, 2, 2023. doi:10.46298/theoretics.23.3.

22 Michael O. Rabin. Decidability of second-order theories and automata on infinite trees. *Transactions of the American Mathematical Society*, 141:1–35, 1969. doi:10.2307/1995086.

23 Wiesław Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theoretical Computer Science*, 200(1-2):135–183, 1998. doi:10.1016/S0304-3975(98)00009-7.