Decoupled Planning for Multiple Omega-Regular Objectives

Guy Avni[1], Thomas A. Henzinger[2], Kaushik Mallik[3], Suman Sadhukhan[4], and K. S. Thejaswini[5(✉)]

[1] University of Haifa, Haifa, Israel
gavni@cs.haifa.ac.il
[2] Institute of Science and Technology Austria, Klosterneuburg, Austria
tah@ista.ac.at
[3] IMDEA Software Institute, Pozuelo de Alarcón, Spain
kaushik.mallik@imdea.org
[4] Clausthal University of Technology, Clausthal-Zellerfeld, Germany
suman.sadhukhan@tu-clausthal.de
[5] Université Libre de Bruxelles, Brussels, Belgium
thejaswini.raghavan@ulb.be

Abstract. We study the problem of generating paths on a graph that satisfy a collection of ω-regular objectives. We propose a decoupled framework in which each objective is assigned to an independent agent that selects a *local* policy, while a scheduler—oblivious to the graph and objective—dynamically composes these policies into a single path. We ask when such a composition satisfies all objectives, assuming their conjunction is realizable. The framework enables modular policy design but raises fundamental compositional challenges. We show that even extremely fair deterministic schedulers do not ensure correctness, and that *stochastic* schedulers, while necessary, are insufficient without coordination. For safety objectives, we demonstrate that fully decentralized implementations are impossible, and we introduce a protocol for synchronizing on maximal safe actions. For non-safety objectives, we introduce *conventions*—simple, a priori restrictions agreed upon before the graph or objectives are revealed—that guarantee satisfaction of all objectives when followed by all agents. We characterize minimally restrictive conventions for major subclasses of ω-regular objectives. In particular, Büchi objectives admit universal composition of finite-memory policies without scheduler communication; co-Büchi objectives require only knowledge of whether the agent was scheduled; and parity objectives additionally require knowledge of which agent was scheduled.

Keywords: ω-regular · Modular Synthesis · Runtime Composition

1 Introduction

The graph traversal problem takes as input a finite directed graph and a temporal specification over sequences of vertices, and the goal is to compute a traversal

E. Darulova et al. (Eds.): CAV 2026, LNCS 16682, pp. 237–257, 2026.
https://doi.org/10.1007/978-3-032-32519-8_13

policy for selecting the next vertex based on the history of the past vertices, such that the resulting infinite path fulfills the given specification. A primary application of graph traversal is robotic path planning [15], and other applications include parsing natural languages using stochastic grammars [19], and informational search with online learning [18]. Often, the specification requires simultaneous fulfillment of multiple, competing objectives; for example, a robot may need to simultaneously patrol a given location *and* visit the charging station intermittently. The traditional approach to solve such problems is to compute a single, monolithic policy for the conjunction of all objectives.

In this paper, we introduce a *decoupled* framework for graph traversals. Consider a collection of objectives $\varphi_1, \ldots, \varphi_N$ that a path needs to satisfy. We describe the framework as an interaction among N agents. For $1 \leq i \leq N$, Agent i is only responsible for satisfying objective φ_i and does so by selecting a *policy* that achieves φ_i. At runtime, a fixed *scheduler* composes the policies by scheduling one of the agents to choose the next vertex at each step. Our goal is to ensure that the path generated by the composition satisfies all objectives (either deterministically or *almost surely*) while achieving maximal decoupling by keeping the agents' choices and scheduler as independent as possible. Before we make these goals more concrete and discuss the features that they lead to, we first illustrate the framework and its challenges.

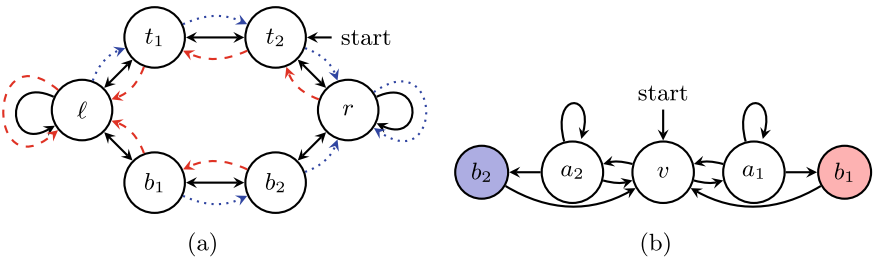

Fig. 1. (a) Reachability objectives given by $T_1 = \{\ell\}$ and $T_2 = \{r\}$, and policies π_1 dashed red and π_2 dotted blue. (b) A graph with two Büchi objectives given by accepting vertices $T_1 = \{b_1\}$ and $T_2 = \{b_2\}$.

Example 1. Consider a coffee-serving robot in an office modeled as in Fig. 1(a). The objective φ_1 requires "reach and serve coffee at ℓ" and the objective φ_2 requires "reach r". Suppose Agent 1 and Agent 2 choose policies π_1 and π_2 that always take the shortest paths towards their respective goal states. Clearly, if the robot follows only π_1 or only π_2, respectively, φ_1 and φ_2 are satisfied. We describe two schedulers for composing the policies at runtime.

First, consider a *deterministic* scheduler $\sigma_{\mathtt{d}}$ that alternates between π_1 and π_2: at odd time steps, the robot follows π_1, and at even time steps, it follows π_2. The composition does not satisfy either objective. For example, the path starting from t_2 forever alternates between t_1 and t_2. Notably, $\sigma_{\mathtt{d}}$ fulfills the *extreme*

fairness property [24], because each agent is given control infinitely often. The example shows that in order to guarantee that all objectives are satisfied, our framework requires schedulers with an even stronger form of fairness.

Now consider a *stochastic* scheduler σ_{s} that, at each point in time, chooses which policy to follow uniformly at random. This scheduler fulfills a stronger notion of fairness, called *probabilistic fairness* [11] (also related to α-*fairness* [25]). The composition of π_1 and π_2 using σ_{s} now generates a random path that almost surely visits each of ℓ, t_1, t_2, and r infinitely often, thus both objectives are satisfied almost surely. □

We postpone the justification of our design choices of schedulers and policies until the end of the introduction.

The advantages of our new framework stem from its modularity; it enables a separation of concerns in designing policies and allows changing, deleting, or adding any number of policies without any extra computational overhead, offering a lightweight, plug-and-play composition operation. We point out specific advantages over the monolithic approach: (i) *parallel design*: each policy is designed independently, which enables, for example, designed on separate CPUs to increase scalability, or by different vendors increasing flexible design; (ii) *robustness*: when an objective changes, only the relevant module needs to change and the other modules can stay fixed; (iii) *iterative design*: a realistic design procedure often requires addition of objectives to the original objectives, which can be simply added as a new module; (iv) *dynamic changes*: our framework allows deleting or adding objectives to a system that has already been deployed (a "patch").

The first step of our decoupled synthesis addresses the safety components of the individual objectives. Given a graph G and an objective φ, we can use standard procedure to decompose φ into safety and liveness components, writing $\varphi = \varphi^{\mathsf{safe}} \cap \varphi^{\mathsf{live}}$, where φ^{safe} requires some unsafe vertices to be avoided at all time, and φ^{live} requires some good event to eventually occur. We show that φ^{safe} cannot, in general, be guaranteed under stochastic scheduling without additional communication. To this end, we propose a protocol in which agents communicate their sets of safe actions at each step, and the scheduled agent is required to select an action from the intersection of these sets. Under mild assumptions, we prove that this protocol suffices to ensure satisfaction of each safety objective at all times. Once safety is enforced, we remove all unsafe edges from the graph and exclusively focus on the liveness objectives in the resulting subgraph. For clarity of exposition, we assume this graph is strongly connected, although this assumption is not essential and will be relaxed later.

We first study the existence of a stochastic scheduler that composes *any* collection of policies in a way that the composition almost-surely satisfies all liveness objectives. Somewhat surprisingly, we show that, already for Büchi objectives in strongly-connected graphs, such "universal" schedulers does not exist. We sketch the intuition in the following example, and details can be found in Theorem 3.

Example 2. Consider the graph depicted in Fig. 1(b). We describe two policies π_1 and π_2 respectively for the Büchi objectives φ_1 and φ_2 given by target vertices

$\{b_1\}$ and $\{b_2\}$. Consider an exponentially increasing sequence, e.g., $n_0 = 1$ and $n_j = 2 \cdot n_{j-1}$, for $j \geq 1$. Define π_1 as follows, and π_2 is dual. When not at a_1, proceed towards b_2, and when at a_1 at time t, if $t = n_j$, for $j \geq 0$, proceed to b_1, otherwise stay at a_1. That is, π_1 spends exponentially increasing time at a_1; visits to b_1 become exponentially rare. Clearly, if π_i operates alone, it satisfies φ_i, for $i \in \{1, 2\}$. Consider a scheduler that at each point in time, chooses which policy acts uniformly at random. Intuitively, the probability that a random walk visits a_i, for $i \in \{1, 2\}$, precisely at a time n_j, for some $j \geq 1$, is exponentially decreasing, which implies that the probability of visiting b_i infinitely often tends to a constant, thus $\varphi_1 \wedge \varphi_2$ is *not* satisfied almost surely. In Theorem 3, we generalize the construction to any stochastic scheduler. □

Example 2 shows that if the agents are allowed to choose any unrestricted policies, there exist situations when the composition will violate the individual objectives. We submit that if all the agents are made to choose policies according to some suitable *convention*, the composition will be guaranteed to satisfy all objectives, almost surely. For example, for Büchi objectives, we show that the convention of choosing only finite-memory policies guarantee the fulfillment of all individual objectives. We formalize conventions as mappings from classes of objectives (Büchi, co-Büchi, etc.) to restrictions on policies (finite-memory, etc.). Importantly, the restrictions need to be finalized *before* seeing the specific graph and objectives at hand, setting our framework apart from the joint design of coordinated policies. This way, the modularity requirement is achieved: agents can independently design their policies based on the respective conventions, and addition, modification, or removal of policies do not affect the rest of the policies.

Conventions can be intuitively explained using a simple metaphor, whose inspiration comes from the work of Lewis [22]. Suppose two friends want to meet, but they are unable to talk to each other and arrange the location. Convention dictates that each of them should go to the place where they most often meet, and if both parties follow the same convention, they will successfully meet. In essence, convention is a way of inducing coordination when direct communication is not possible. Coming back to our framework, conventions help agents achieve their objectives without directly synchronizing their policies with each other.

Our goal is to establish minimally restrictive conventions for different classes of ω-regular objectives.

While the convention for Büchi objectives was merely to use finite-memory policies, the same does not suffice anymore for co-Büchi objectives, as we illustrate using the following example.

Example 3. Consider the graph that is depicted in Fig. 1(a) and two co-Büchi objectives: $\mathrm{FG}\,\ell$, requiring eventually staying in ℓ, and $\mathrm{FG}\,(\ell \vee r)$, requiring eventually staying in either ℓ or r. Observe that for $i \in \{1, 2\}$, following the finite-memory policy π_i (depicted also in Fig. 1(a) in dashed red and dotted blue) results in a path that satisfies the respective objective, but a random composition of the two visits both t_1 and t_2 infinitely often, almost surely, and thus violates both objectives. □

The challenge behind co-Büchi convention is that in order to satisfy a conjunction of co-Büchi objectives, all policies—that are oblivious to each other's target vertices—must eventually converge into a common cycle that is good for everyone. We describe a convention for co-Büchi objectives in strongly-connected graphs. Given a co-Büchi objective φ_i, Agent i chooses a policy π_i that operates as follows. (1) Randomly choose a lasso-shaped path $\rho^i = \rho_i \cdot C_i^\omega$, where C_i is a cycle that satisfies φ_i. This is Agent i's guess of the common path everyone is trying to follow. (2) If scheduled at vertex v: choose the successor v' of v on ρ^i. (3) If not scheduled: observe the next vertex u, and if $u \neq v'$, a conflict is observed: return to Step (1) and sample a new path.

Example 4. Consider again the objectives over the graph depicted in Fig. 1(a) $\varphi_1 = \mathrm{FG}\,\ell$ and $\varphi_2 = \mathrm{FG}\,(\ell \vee r)$, the initial position is t_1, and let the choice of lasso paths be $\rho^1 = t_1\ell^\omega$ and $\rho^2 = t_1t_2r^\omega$. Observe that ρ^i satisfies φ_i for each i. Suppose that π_1 is scheduled and proceeds to ℓ. Thus, π_2 observes a conflict (it expects the next vertex to be t_2), and randomly chooses an alternative lasso path that satisfies φ_2. Correctness follows from a measure-theoretic argument: If there exists a common path that satisfies all co-Büchi objectives, then there also exists a lasso-shaped path that does the same. Random scheduling implies that eventually and almost surely the policies will stabilize in a lasso-shaped path where no policies will change their guesses anymore. □

We develop sufficient conventions for major classes of ω-regular objectives, including safety, Büchi, co-Büchi, and parity, and consider path planning problems on both strongly connected and general graphs. As the objectives become more complex, the conventions become more demanding, as expected. Although our conventions are simple to describe, their correctness proofs involve intricate measure-theoretic arguments. With this, we create the foundation of a first-of-its-kind modular design framework for ω-regular decision-making.

Design Choices. We discuss the design choices of our framework. *(i) Independent policy choice.* In all our conventions, an agent chooses a policy without knowing the objectives of the other agents. This enables robustness: when objective φ_j changes, Agent i does not need to change their policy. *(ii) Minimal runtime coordination.* For Büchi objectives, conventions choose traditional policies (later called *path-aware policies*) that use no runtime coordination (see π_1 and π_2 in Example 1). This means that agents can apply off-the-shelf tools to design policies, and different agents can even apply different techniques, e.g., reinforcement learning or formal techniques. For co-Büchi objectives, policies know which time steps they are scheduled in. Importantly, such policies are *not* aware of the number of agents, meaning that additional objectives can be added (offline or at runtime) without changing the policy. *(iii) Graph-independent schedulers.* To motivate this, we return to Example 1. Intuitively, composition using the deterministic scheduler σ_{d} failed because none of the agents were getting enough time to reach their target. A naïve fix simply schedules each policy *three* times in a row. While this does solve the problem at hand, it is an ad hoc solution that will break if either graph or objectives change. Alternatively, one could consider a

nondeterministic scheduler and construct policies that fulfill their objectives no matter how the nondeterminism is resolved. Unfortunately, this is too restrictive from the policy design standpoint: in Example 1, given the policies π_1 and π_2, a nondeterministic scheduler can act like an adversary and find an interleaving of the policies (like the one of σ_{d}) such that the path keeps cycling between t_1 and t_2 and never visits the targets. Defeating such an adversarial scheduler is possible, but would require the policies to take joint actions (like π_1 and π_2 making a clockwise traversal of the graph in Example 1), which would preclude the key aspect of modular design. *(iv) Simple schedulers.* We see it as a feature of our framework that our schedulers are simple and can thus run on devices with limited processing capabilities like robots or drones.

Related Work

Computing policies for multi-objective decision-making problems is a well-studied problem, both for probabilistic [6,7,10] and non-probabilistic [26] systems. Almost all works produce a single, monolithic policy that fulfills all objectives.

Decoupling has been considered in other fields with the same motivation as us; a framework that is modular, robust, and separates design concerns. For example, decoupling exploration from exploitation has been studied in RL [27], *behavioral programming* is a decoupled approach to programming [16], decoupling has been applied in control [14], and game theory [17].

We compare with two approaches for decoupled design that are closest to our work. The first kind "over-approximates" the local policies using the so-called *strategy templates* [2], which abstractly represent sets of policies that would fulfil the objectives.[1] When these local strategy templates are composed, we obtain a strategy template for the conjunction of all objectives, from which a concrete policy for the overall problem can be extracted. Unfortunately, the composition of strategy templates is computationally intensive, which creates substantial computational overhead and can impact the runtime performance of the composed policy.

The second kind of modular design uses auctions as a scheduler, and is called *auction-based scheduling* (ABS) [5]: Each policy starts with a budget, and at each step, an auction is held, where the policies use their available budgets to bid for the privilege of selecting the next action. *Bidding policies* for ABS are found by independently solving a two-player zero-sum *bidding game* [3,21]. We point to two key advantages of our approach over ABS. First, our approach allows an arbitrary number of objectives whereas ABS for multiple objectives is currently not possible due to the dependency on multi-player bidding games, which, to best of our knowledge, have not yet been studied. Second, ABS is a sound but incomplete method: it is possible that there is a path that satisfies both objectives, but there is no decoupled solution for ABS. In fact, most instances

[1] Strategy templates are originally developed on turn-based game graphs, which are generalizations of the plain graphs that we use in this work.

with a pair of co-Büchi objectives cannot be decoupled using ABS. On the other hand, our framework is guaranteed to successfully decouple any collection of parity objectives on strongly-connected graphs, assuming, of course, that they have an intersection.

In several graph games, "mixed" or *randomised policies* are used either out of necessity for winning against the opponent [12] or to reduce the memory requirements of the policies [9]. Our randomised scheduling of local policies can be viewed as randomised policies, where the randomisation over actions is facilitated by the scheduler. However, there is an important difference: usually, randomised policies are designed centrally for the entire objective, whereas our goal is to design isolated local policies independently from the scheduler.

The classical distributed view of reactive synthesis [13,20,23] uses a fundamentally different model. It is assumed that the actions chosen by each local policy affect a separate set of variables, so that all local policies can be deployed in parallel. In other words, classical distribution synthesis does not require scheduling of policies, and asks how to design local policies whose parallel composition fulfills all objectives.

All missing proofs can be found in the extended version of the paper [4].

2 Preliminaries

We use $\mathbb{N}$ to denote the set of natural numbers $\{0, 1, 2, \dots\}$. For two natural numbers i, j with $i < j$, we use $[i; j]$ to denote the set $\{i, i+1, \dots, j\}$ consisting of natural numbers that are at least i and at most j. Sometimes, for a natural number i, we use $[i]$ to denote the set $[1; i]$.

Graphs, and Paths. A graph is an ordered triple $\mathcal{G} = (V, E, v_{\mathsf{init}})$, where V is a finite set of vertices, $E \subseteq V \times V$ is a set of directed edges, and $v_{\mathsf{init}} \in V$ is the initial vertex. A path in $\mathcal{G}$ is a (finite or infinite) sequence of vertices $v_0 v_1 \dots$ such that $v_0 = v_{\mathsf{init}}$ and $(v_i, v_{i+1}) \in E$ for every valid index i. We write $Paths_{\mathsf{fin}}(\mathcal{G})$ and $Paths_{\mathsf{inf}}(\mathcal{G})$ for the sets of finite and infinite paths of $\mathcal{G}$, respectively. By convention, $Paths_{\mathsf{fin}}(\mathcal{G})$ includes the path of length 0.

Objectives. An *objective* φ in $\mathcal{G}$ is a set of infinite paths in $\mathcal{G}$. We represent objectives defined using predicates over the vertices of the given graph. We consider the following *families* of objectives.

Reachability is specified by a set of *target* vertices $T \subseteq V$, and is the set of every path that eventually visits at least one vertex in T.

Safety is specified by a set of *safe* vertices $S \subseteq V$, and is the set of every path always remaining within the vertices in S.

Parity is specified by a *colouring* function $\kappa : V \mapsto \mathbb{N}$, which associates each vertex a natural number called its *colour*. An infinite path is in the parity objective iff the largest colour that appears infinitely often is even.

Büchi is a special case of parity in which vertices are only coloured by 1 and 2, where the latter are called *Büchi* vertices. A path is in the Büchi objective iff Büchi vertices are visited infinitely often.

Co-Büchi is a special case of parity in which vertices are only coloured by 0 and 1, where the latter are called *co-Büchi* vertices or "bad" vertices. A path is in the co-Büchi objective iff it visits co-Büchi vertices only finitely often.

We will divide objectives into two complementary families, namely safety and liveness, defined below. While safety can be defined using safe vertices as above, the following definition uses an alternate, semantic variant.

Definition 1 (Safety versus liveness objectives). *Let $\mathcal{G}$ be a given graph. A given objective φ in $\mathcal{G}$ is called* liveness *if every finite path in $\mathcal{G}$ can be extended into an infinite path in φ. Dually, a given objective φ in $\mathcal{G}$ is called* safety *if every infinite path $v_0 v_1 \ldots \notin \varphi$ has a finite prefix $\rho = v_0 \ldots v_i$ such that for every $j \geq i$, all infinite extensions of the path $v_0 \ldots v_j$ is not in φ.*

Almost-Sure Satisfaction. Let $\mathcal{D}(V)$ denote the set of all probability distributions over V, and given $d \in \mathcal{D}(V)$, we will write $\mathsf{Supp}(d)$ to denote the support of d. Consider a function $f : \mathit{Paths}_{\mathsf{fin}}(\mathcal{G}) \to \mathcal{D}(V)$ that, given a finite path ρ ending at a vertex v, assigns a probability distribution whose support is contained in the set of successors of v. We extend f to infinite paths as follows. For a finite path $\rho = v_0 \ldots v_k$, the cylinder set spanned by ρ, denoted $\mathrm{Cyl}(\rho)$, is the set of all infinite paths whose prefix is ρ. Define the probability distribution over cylinder sets based on f as $\Pr^f(\mathrm{Cyl}(\rho)) = \prod_{i=0}^{k-1} f(v_0 \ldots v_i)(v_{i+1})$. The function $\Pr^f$ is a pre-measure over the set $\mathit{Paths}_{\mathsf{inf}}(\mathcal{G})$, which extends to a *unique* probability measure over $\mathit{Paths}_{\mathsf{inf}}(\mathcal{G})$ by applying the Carathéodory's extension theorem [8]. For simplicity, we use the notation $\Pr^f$ to also represent the probability measure. We say that f *almost surely* satisfies an objective φ if $\Pr^f(\varphi) = 1$.

3 The Decoupled Planning Framework

We develop a decoupled framework for generating a path that satisfies a collection $\varphi_1, \ldots, \varphi_N$ of objectives. The framework has two main ingredients, N individual *agents* and *a scheduler*. Each Agent i, for $i \in [1; N]$, is solely interested in fulfilling the objective φ_i. To this end, the Agent i chooses a policy π_i, and the policies of all agents are composed at runtime by the scheduler.

This section is devoted to formalizing this framework. We start with schedulers.

Definition 2 (Schedulers: general, fair, deterministic). *A* scheduler *over $[N]$ is a function $\sigma : [N]^* \to \mathcal{D}([N])$. The scheduler σ is called* fair *if there exists $\epsilon > 0$ such that for every $u \in [N]^*$ and for every $n \in [N]$, $\Pr(n = \sigma(u)) > \epsilon$. A scheduler is called* deterministic *if it always chooses Dirac distributions.*

We reiterate the advantages of stochastic schedulers mentioned in the introduction. First, they are completely agnostic of the underlying graph and the objectives, while not obstructing policies to fulfill their objectives. Moreover, stochastic schedulers are lightweight and easy to implement. In fact such schedulers are often used in concurrent programs, where multiple programs (analogous

to our policies) that are competing for a resource (being scheduled, analogous to our policies being able to execute their actions) are allocated the resource randomly. Henceforth, schedulers will by default be stochastic.

Semantics of Schedulers. Every sequence from $[N]^* \cup [N]^\omega$ is called a *schedule* which is either finite or infinite. Every scheduler σ induces a probability measure over the infinite schedules as follows. Given a finite sequence $\theta = p_0 \dots p_k \in [N]^*$, we define $\Pr^\sigma(\mathrm{Cyl}(\theta)) = \prod_{i=0}^{k-1} \sigma(p_0 \dots p_{k-1})$, which is a pre-measure that extends to a unique measure—also denoted as $\Pr^\sigma$—over the set $[N]^\omega$ by applying Carathéodory's extension theorem [8]. □

We formalize the second ingredient of our framework, the policies that the agents choose, where we define policy types with increasing knowledge of scheduling choices. We introduce some notation first. Given two alphabets A and B, and a pair of finite words of equal length $w_A = a_0 \dots a_k \in A^*$ and $w_B = b_0 \dots b_k \in B^*$, we define their element-wise cross-product as: $w_A \otimes w_B := (a_0, b_0) \dots (a_k, b_k) \in (A \times B)^*$. Furthermore, we lift this cross-product to sets of sequences: given $W_A \subseteq A^*$ and $W_B \subseteq B^*$, define $W_A \otimes W_B := \{w_A \otimes w_B \mid w_A \in W_A, w_B \in W_B\}$.

Definition 3 (Policies augmented with scheduling information). *Let σ be a scheduler over $[N]$.*

- **Path-aware** *policies coincide with the traditional definition of policies and have no scheduling knowledge, they are of the form $\pi\colon Paths_{\mathsf{fin}}(\mathcal{G}) \to \mathcal{D}(V)$. We denote the set of path-aware policies by Π^{Path}.*
- **Scheduled-aware** *policies keep track of which past time points they have been scheduled, and all choices made by it in the past. A scheduled-aware policy is a* partial *function of the form $\pi\colon Paths_{\mathsf{fin}}(\mathcal{G}) \otimes \{\top, \bot\}^* \otimes V^* \to \mathcal{D}(V)$, mapping the path $\rho = v_0 \dots v_k \in Paths_{\mathsf{fin}}(\mathcal{G})$, the history $\theta = q_0 \dots q_{k-1} \in \{\top, \bot\}^*$ of time points in which it was scheduled, where "$\top$" means "scheduled" and "$\bot$" means not scheduled, and the history of choices $\gamma = w_1 \dots w_k \in V^*$ made by it, to the distribution $\pi(\rho, \theta, \gamma)$. The distribution $\pi(\rho, \theta, \gamma)$ is defined iff ρ, θ, and γ are consistent, i.e., for every $i \in [k]$ with $q_{i-1} = \top$, $v_i = w_i$. The set of all full scheduled-aware policies will be denoted as Π^{S}.*
- **Full history-aware** *policies enrich scheduled-aware policies by additionally including the indices of scheduled agents during times it was unscheduled (instead of just keeping track of "$\bot$" as scheduled-aware policies do). Without loss of generality, assume that the indices of the other policies are $[2; N]$. A full history-aware policy is a* partial *functions of the form $\pi\colon Paths_{\mathsf{fin}}(\mathcal{G}) \otimes \{\top, 2, \dots, N\}^* \otimes V^* \to \mathcal{D}(V)$, mapping the path $\rho \in Paths_{\mathsf{fin}}(\mathcal{G})$, the history of scheduling decisions $\theta \in \{\top, 2, \dots, N\}^*$, and the history of choices $\gamma \in V^*$ made by π, to the distribution $\pi(\rho, \theta, \gamma)$. The distribution $\pi(\rho, \theta, \gamma)$ is defined under the same conditions as for scheduled-aware policies. The set of all full history-aware policies with N agents will be denoted as Π_N^{FH}.*

For each policy, regardless of its type, for every given path $\rho v \in Paths_{\mathsf{fin}}(\mathcal{G})$, the support of the output distribution over vertices must be a subset of $\mathsf{Supp}(v)$.

Remark 1 (Why keep track of past choices?). One may naïvely think that it is redundant that policies in Π^{S} and Π_N^{FH} keep track of their own past choices, alongside the whole path history, because the old histories could be used to simulate the old choices on demand. This will not work since the policies are stochastic, and executing them on the same history will generate different outcomes on different instances with positive probability.

Remark 2 (Comparing decoupling strength). Decoupling is stronger when using a family of policies with weaker scheduling information. Since path-aware policies coincide with the traditional definition of policies, an agent can use any off-the-shelf tool to construct a policy for their objective. This is no longer the case with scheduled-aware and full history-aware policies. Yet, an appealing feature of scheduled-aware policies is their independence of how many policies are interacting. This enables, for example, "patching" a deployed system by removing or adding an objective at runtime. Indeed, even when N changes, no update is needed to an individual policy. This is no longer the case with full history-aware policies. We point out that decoupling with full history-aware policies still provides modularity: as long as the number of objectives is fixed, changes to objectives at runtime is possible by changing only the relevant policies.

We define the *composition* of a scheduler and a collection of full history-aware policies. This generates a probability distribution over the infinite paths generated by a random interleaving of the individual local policies.

Definition 4 (Composition). *Suppose we are given a graph $\mathcal{G}$, a scheduler σ for a set of N agents, and a set of policies $\pi_1, \ldots, \pi_N$ where for every $i \in [N]$, $\pi_i \in \Pi_N^{\mathrm{FH}} \cup \Pi^{\mathrm{S}} \cup \Pi^{\mathrm{Path}}$. The composition is the tuple $\mathcal{I} = (\mathcal{G}, \pi_i, \ldots, \pi_N, \sigma)$.*

Each composition defines a probability distribution over the paths generated, which we computed step-by-step below. Such a distribution depends on the type of scheduler, hence we define it carefully below.

Semantics of Composition. At each time point i, when the current path is $v_0 \ldots v_i \in \mathit{Paths}_{\mathsf{fin}}(\mathcal{G})$, each Agent j randomly selects the next vertex w_{i+1}^j using its policy, giving us a tuple $(w_{i+1}^1, \ldots, w_{i+1}^N)$ of selections. Out of this tuple, the choice of the scheduled agent p_i is used to extend the current path to $v_0 \ldots v_{i+1}$, where $v_{i+1} = w_{i+1}^{p_i}$, and the process continues. For a given fixed schedule $p_0 \ldots p_{k-1} \in [N]^*$, we define the probability of the sequence of selections being $(w_1^1, \ldots, w_1^N) \ldots (w_k^1, \ldots, w_k^N)$ given as:

$$\Pr^{\mathcal{I}}\left((w_1^1, \ldots, w_1^N) \ldots (w_k^1, \ldots, w_k^N) \mid (p_0, \ldots p_{k-1})\right) \coloneqq \prod_{j=1}^{N} \prod_{i=0}^{k-1} D(w_{i+1}^j), \qquad (1)$$

where $D(w_{i+1}^j)$ represents the probability that at time $i+1$, the policy π_j selected the vertex w_{i+1}^j, whose value is defined based on the policy type as below. In the above, the path generated by the schedule $p_0, \ldots p_{k-1}$ from the sequence

$(w_1^1, \ldots, w_1^N) \ldots (w_k^1, \ldots, w_k^N)$ is $v_0 v_1 \ldots v_k$ where $v_0 = v_{\mathsf{init}}$ and for each $i \in [k]$, $v_i = w_i^{p_{i-1}}$. The set of all policies are $\{\pi_j\}_{j \in [N]}$. Then,

$$D(w_{i+1}^j) = \begin{cases} \pi_j\left(v_0, \ldots v_i\right) & \pi_j \in \Pi^{\mathrm{Path}}, \\ \pi_j\left((v_0 \ldots v_i), (q_0 \ldots q_{i-1}), (w_1^j \ldots w_i^j)\right), \forall t \,.\, q_t = \top \Leftrightarrow p_t = j & \pi_j \in \Pi^{\mathrm{S}}, \\ \pi_j\left((v_0 \ldots v_i), (p_0 \ldots p_{i-1}), (w_1^j \ldots w_i^j)\right) & \pi_j \in \Pi_N^{\mathrm{FH}}. \end{cases}$$

Suppose we are given a finite path $\rho = v_0 \ldots v_k \in \mathit{Paths}_{\mathsf{fin}}(\mathcal{G})$ and a schedule $\theta = p_0 \ldots p_{k-1} \in [N]^*$. We introduce the set $\Sigma_{v_0,\rho,\theta} \subseteq V^{N \times k}$ representing the set of all sequences of vertices generated by all agents such that the schedule θ would produce the path ρ starting at the initial vertex v_0. Formally, $\Sigma_{v_0,\rho,\theta} \coloneqq \left\{(v_1^1, \ldots, v_1^N) \ldots (v_k^1, \ldots, v_k^N) \mid \forall i \in [k] \,.\, v_i = v_i^{p_{i-1}}\right\}$. Now we combine $\Pr^{\mathcal{I}}(\rho \mid \cdot)$ with the probability measure $\Pr^{\sigma}$ over sequences of policy indices, and write $\Pr^{\mathcal{I}}(\mathrm{Cyl}(\rho))$ as

$$\sum_{\substack{\theta \in [N]^{k-1} \\ (v_1^1, \ldots, v_1^N) \ldots (v_k^1, \ldots, v_k^N) \in \Sigma_{v_0,\rho,\theta}}} \Pr^{\mathcal{I}}((v_1^1, \ldots, v_1^N) \ldots (v_k^1, \ldots, v_k^N) \mid \theta) \cdot \Pr^{\sigma}(\theta), \quad (2)$$

which is a pre-measure and leads to a unique measure—also written $\Pr^{\mathcal{I}}$—over the set of all infinite paths by applying the Carathéodory's extension theorem [8]. We will write $\mathcal{I} \models_{\mathsf{a.s.}} \varphi$ iff $\Pr^{\mathcal{I}}(\varphi) = 1$. □

Following is the decoupling problem that we undertake.

Problem statement (informal): decoupled planning

Fix a fair, stochastic scheduler σ. Suppose $\mathcal{G}$ is a given graph and $\varphi_1, \ldots, \varphi_N$ are the objectives. Independently design the policies $\pi_1, \ldots, \pi_N$ in a way that $\mathcal{I}(\mathcal{G}, \pi_1, \ldots, \pi_N, \sigma) \models_{\mathsf{a.s.}} \varphi_1 \wedge \ldots \wedge \varphi_N$.

Towards this goal, we start by stating why a fair *deterministic* scheduler would not suffice in achieving the desired decoupling, which formalizes the informal description in Example 1.

Theorem 1 (No deterministic scheduler can be universal). *For every deterministic scheduler σ, there exist a graph $\mathcal{G}$ and reachability objectives φ_1 and φ_2 with $(\varphi_1 \cap \varphi_2) \neq \emptyset$ and deterministic policies π_1^σ and π_2^σ that respectively satisfy φ_1 and φ_2, but $(\mathcal{G}, \pi_1^\sigma, \pi_2^\sigma, \sigma) \not\models \varphi_1 \wedge \varphi_2$.*

The high level idea (slightly over-simplified) of the proof appears in Example 1, and a rigorous proof can be found in the full version of the paper. This limitation of (extremely) fair deterministic schedulers inspired us to use stochastic schedulers with the stronger probabilistic fairness [11] properties.

4 Building Support for Safety Components of ω-Regular Objectives

We start by demonstrating the challenge of incorporating safety.

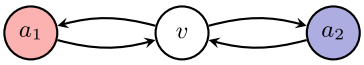

Fig. 2. A graph with a safety objective $G(\neg a_1)$ and a Büchi objective GFa_2. (Color figure online)

Example 5. Consider the graph depicted in Fig. 2, and two objectives, the safety objective φ_1 requires "avoid the Red vertex a_1" and a Büchi objective φ_2 requires "visit the Blue vertex a_2 infinitely often". Suppose that Agent 2 chooses the policy π_2 that alternates between selecting a_1 and a_2 at v. Note that if π_2 is always scheduled, then φ_2 is satisfied. Observe that no matter which policy Agent 1 chooses, any stochastic scheduler that schedules π_2 at v with probability at least $\epsilon > 0$ leads to a random walk that visits a_1 almost surely and thus violates φ_1. Finally, note that the path $(v, a_1)^\omega$ satisfies $\varphi_1 \wedge \varphi_2$. □

As demonstrated in Examples 1 and 4, liveness objectives allow for flexible decoupling. Safety, however, is rigid. Indeed, it is well known that almost-sure and sure satisfaction coincide for safety [12]. That is, in order to satisfy safety, a composition needs to satisfy it deterministically on *every* path. This requires communication, which we establish by augmenting the policies with a safety-preserving component defined as follows.

Maximally Permissive Policies. A *maximally permissive policy* for a safety specification φ is $\chi : V \to 2^V$. Intuitively, for a vertex v, the set $\chi(v)$ represents safe continuations from v. More formally, a path $\rho = v_0, v_1, \ldots$ such that $v_{i+1} \in \chi(v_i)$ satisfies φ. Maximality of χ means that for every v and every $u \notin \chi(v)$, there is a continuation from u that violates φ. In order to construct χ, we compute the set of vertices from which φ_i can be fulfilled, which we call the *winning region* and denote it by W. Then, for every $v \in W$, we define $\chi(v) = \{u : E(v, u) \wedge u \in W\}$. Observe that χ is maximal; proceeding to $u \notin \chi(v)$ violates φ.

Shielded Composition. We augment the composition defined in Definition 4 with maximally-permissive policies. Consider policies $\pi_1, \ldots, \pi_N$ and maximally-permissive policies $\chi_1, \ldots, \chi_N$. Whenever Agent $i \in [N]$ is scheduled at vertex v, every other Agent $j \neq i$ sends $\chi_j(v)$ to Agent i. Agent i selects a vertex $\pi_i(v) \in \bigcap_{1 \leq j \leq N} \chi_j(v)$ that is safe for all. We will prove that, under mild conditions on the graph and objectives, there is always such a safe vertex to choose.

Decomposition of ω-Regular Specifications. Consider an ω-regular objective φ_i for Agent i. It is possible to decompose φ_i into a safety and a liveness component

$\varphi_i^{\mathsf{safe}}$ and $\varphi_i^{\mathsf{live}}$, such that $\varphi_i = \varphi_i^{\mathsf{safe}} \cap \varphi_i^{\mathsf{live}}$ [1]. Let W_i be the maximal winning region, W_i^c the complement of W_i, and $\mathit{Reach}(W_i^c)$ the objective that contains all paths eventually reaching W_i^c. Define $\varphi_i^{\mathsf{live}} := \varphi_i \cup \mathit{Reach}(W_i^c)$. Observe that $\varphi_i^{\mathsf{live}}$ is a liveness objective, since no matter what prefix we have seen, we can either stay in W_i and fulfil φ_i, or reach W_i^c. Clearly $\varphi_i = \varphi_i^{\mathsf{safe}} \cap \varphi_i^{\mathsf{live}}$, and moreover, $\varphi_i^{\mathsf{live}}$ can be expressed as a parity objective (possibly with additional colours).

We describe how the agents act. Let π_i be a policy for Agent i and let χ_i be a maximally permissive policy. As before, when Agent i is scheduled following path ρ that ends at v, every other Agent j sends $\chi_j(v)$ and Agent i selects $\pi_i(\rho) \in \bigcap_j \chi_j(v)$. We call this a *shielded* composition, and denote it by $(\mathcal{G}, \pi_1, \ldots, \pi_n, \sigma)^{\mathsf{shld}}$. We omit the maximally permissive policies for brevity.

Mutual Safety Closed. Consider an objective $\varphi \subseteq V^\omega$, we define $\mathit{Pref}(\varphi) := \{u \in V^\star \mid \exists v \in V^\omega \,.\, uv \in \varphi\}$ to be the set of all finite prefixes of φ. We call a pair of objectives φ_1 and φ_2 *mutually safety-closed*, if every path ρ in $\mathcal{G}$ fulfilling $\rho \in \mathit{Pref}(\varphi_1) \cap \mathit{Pref}(\varphi_2)$ also fulfils $\rho \in \mathit{Pref}(\varphi_1 \cap \varphi_2)$. By transitivity, we can generalize mutual safety-closure to arbitrary number of objectives.

The mutually safety closed assumption guarantees that every prefix of a path has an extension that satisfies each objective. This ensures that a path that makes a "wrong step" can always recover.

Assumption 1. *The objectives $\{\varphi_1, \ldots, \varphi_N\}$ are mutually safety-closed.*

Let W_i be the maximal set of vertices in $\mathcal{G}$ from which the agent i has a policy to fulfil φ_i. In other words, $\varphi_i^{\mathsf{safe}} = \mathit{Safe}(W_i)$. Define $W := \cap_{i \in [N]} W_i$. By assumption, W is nonempty, since $v_{\mathsf{init}} \in W_i$ for all i. Let $\mathcal{G}[W]$ be the subgraph obtained by removing all the vertices $V \setminus W$ from $\mathcal{G}$, and by removing all the edges that involve these vertices. If Assumption 1 holds, then it can be shown that the subgraph $\mathcal{G}[W]$ is deadlock-free, and therefore every path can be extended into an infinite path.

We are now ready to present the main theorem of this section, which suggests the following decoupling framework for ω-regular objectives. The shielded composition that is described above keeps the path in the *strongly-connected* region $G[W]$ that is safe for all agents. This leaves us with liveness, which the policies $\pi_1, \ldots, \pi_N$ take care of. Constructing $\pi_1, \ldots, \pi_N$ is not trivial. In subsequent sections, we develop such constructions while assuming that the graph is strongly connected and all objectives are liveness objectives.

Theorem 2 (Compositional enforcement of the safety components). *Suppose Assumption 1 holds true. Then for every fair scheduler σ and every set of local policies $\pi_1, \ldots, \pi_N$ the following holds:*

$$\left[(\mathcal{G}[W], \pi_1, \ldots, \pi_N, \sigma) \models_{\mathsf{a.s.}} \bigcap_{i=1}^{N} \varphi_i^{\mathsf{live}}\right] \implies \left[(\mathcal{G}, \pi_1, \ldots, \pi_N, \sigma)^{\mathsf{shld}} \models_{\mathsf{a.s.}} \bigcap_{i=1}^{N} \varphi_i\right].$$

5 Decoupling Liveness Objectives via *Conventions*

Unlike safety objectives, we show that liveness objectives *can* be decomposed without any direct communications among agents. However, this does not rule out the need for coordination. In fact, already for Büchi objectives on strongly connected graphs, without coordination the agents may not be able to fulfill their objectives. This is somewhat surprising, and the intuition was sketched in Example 2. We formally state this result in the following theorem.

Theorem 3 (No stochastic scheduler for uncoordinated Büchi policies). *There is a graph $\mathcal{G}$ such that for any stochastic scheduler σ, there are Büchi objectives φ_1 and φ_2 with $(\varphi_1 \cap \varphi_2) \neq \emptyset$ and deterministic policies π_1^σ and π_2^σ that respectively satisfy φ_1 and φ_2, but the composition $(\mathcal{G}, \pi_1^\sigma, \pi_2^\sigma, \sigma) \not\models_{\mathsf{a.s.}} \varphi_1 \cap \varphi_2$.*

We remark that the proof above also shows that even for a reachability objective, despite using Büchi strategies that ensure that every prefix satisfies the Büchi objective, the probability of reaching an accepting state can be bounded by 1/2 (union bound).

Corollary 1 (No stochastic scheduler for uncoordinated reachability objectives). *There exists a graph $\mathcal{G}$ such that for every scheduler σ, there exists reachability objectives φ_1 and φ_2 with $(\varphi_1 \cap \varphi_2) \neq \emptyset$ and deterministic policies π_1^σ and π_2^σ that respectively satisfy φ_1 and φ_2, but $(\mathcal{G}, \pi_1^\sigma, \pi_2^\sigma, \sigma) \not\models_{\mathsf{a.s.}} \varphi_1 \cap \varphi_2$.*

These results motivate us to seek mechanisms that would induce the required coordination among agents, without them requiring to establish direct communication during the design and deployment of their policies. To this end, we formalize *conventions*, which are rules that agents should follow while selecting their policies, and it is guaranteed if every agent follow these conventions, the composition will fulfill all objectives. Figure 3 illustrates how conventions are used.

Definition 5 (Conventions). *Consider a family of objectives α, where we focus on $\alpha \in \{\text{Büchi}, \text{co-Büchi}, \text{parity}\}$. A convention $Conv_\alpha$ for the family α is a mapping that takes a graph $\mathcal{G}$, a scheduler σ, and an objective φ, and outputs a collection of policies of type β that the agent can choose from. We focus on $\beta \in \left\{\widetilde{\Pi}_N^{\mathrm{FH}}, \widetilde{\Pi}^{\mathrm{S}}, \widetilde{\Pi}^{\mathrm{Path}}\right\}$ and design a convention $Conv_\alpha$ such that for every $\mathcal{G}, \sigma$, and $\varphi \in \alpha$, we have $Conv_\alpha(\mathcal{G}, \sigma, \varphi) \subseteq \beta$.*

- *Soundness: A convention $Conv_\alpha$ is sound for classes of graphs and schedules, if for every graph $\mathcal{G}$ and schedule σ from the respective class, and collection of objectives $\varphi_1, \ldots, \varphi_N \in \alpha$, if for every $i \in \{1, \ldots, N\}$ and $\pi_i \in Conv_\alpha(\mathcal{G}, \sigma, \varphi_i)$, we have $(\mathcal{G}, \pi_i, \ldots, \pi_N, \sigma) \models_{\mathsf{a.s.}} \varphi$.*
- *A partial order on conventions: Consider two conventions $Conv_\alpha$ and $Conv'_\alpha$. We define $Conv_\alpha \preceq Conv'_\alpha$ if $Conv_\alpha$ uses policies with weaker scheduling information or if $Conv_\alpha$ is less restrictive than $Conv'_\alpha$, that is for every $\mathcal{G}, \sigma, \varphi$, we have $Conv'_\alpha(\mathcal{G}, \sigma, \varphi) \subseteq Conv_\alpha(\mathcal{G}, \sigma, \varphi)$.*

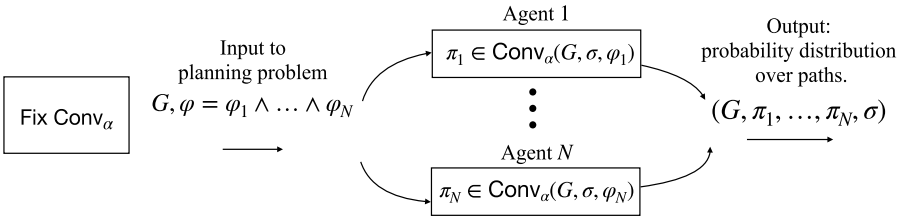

Fig. 3. Decoupling planning using a convention Conv_α. Observe that Conv_α is independent of the graph, scheduler, or specific objectives. Agent i independently chooses $\pi_i \in \mathrm{Conv}_\alpha(\mathcal{G}, \sigma, \varphi_i)$ that agrees with the convention. Soundness of Conv_α implies that the output almost-surely satisfies φ.

We now present the problem that we will study.

Problem statement: decoupled planning of liveness objectives

We focus on strongly-connected graphs and fair stochastic schedulers. For a family of objectives $\alpha \in \{\text{Büchi}, \text{co-Büchi}, \text{parity}\}$. Find a sound and minimal convention Conv_α.

Our focus is on finding sound conventions and we do not show optimality, i.e., provide lower bounds to show inexistence of better conventions. We argue that finding conventions that suffice for decoupling is a nontrivial and challenging problem since conventions are fixed before the graph, scheduler, and objectives are known. Moreover, developing conventions has practical applications whereas showing their inexistence does not.

6 Convention for Decoupling Büchi Objectives

We show that decoupling is possible for Büchi objectives with mildly restricted path-aware policies; that is, Büchi admits a strong form of decoupling.

Definition 6 (Policies with uniform bounded hitting-time). *Let $\mathcal{G} = (V, E, v_0)$ be a graph and let $\pi \in \Pi^P$ be a stochastic policy on $\mathcal{G}$. Fix a subset $B \subseteq V$ and a finite path $\alpha = v_0 v_1 \ldots v_k \in Paths_{\mathsf{fin}}(\mathcal{G})$. The* hitting time random variable *of B under π from α is defined by*

$$\tau_\pi(\alpha)(\rho) \;:=\; \min\{\, t \geq 1 \mid v_{k+t} \in B \text{ in the extension } \rho \in \mathrm{Ext}_\pi(\alpha) \,\},$$

with the convention $\tau_\pi(\alpha)(\rho) = \infty$ if no such t exists. The expected hitting time *of B from α under π is then $\mathbb{E}_\pi[\tau_\pi(\alpha)] = \int_{\mathrm{Ext}_\pi(\alpha)} \tau_\pi(\alpha)(\rho)\, d\Pr^\pi(\rho)$. We say that π has* uniformly bounded expected hitting time *for B if there exists a finite constant L such that $\sup_{\alpha \in Paths_{\mathsf{fin}}(\mathcal{G})} \mathbb{E}_\pi[\tau_\pi(\alpha)] \leq L$.*

The following theorem proves implementation of any such policies with a stochastic scheduler is good for all. The proof can be found in the full version.

Theorem 4. *Consider a strongly connected graph $\mathcal{G}$, policies $\pi_1, \ldots, \pi_N$ with uniform bounded hitting-time for Büchi objectives $\varphi_1, \ldots, \varphi_N$, and a stochastic scheduler σ. Then, $(\mathcal{G}, \pi_1, \ldots, \pi_N, \sigma) \models_{\text{a.s.}} \bigcap_{1 \leq i \leq N} \varphi_i$.*

7 Convention for Decoupling Co-Büchi Objectives

We start by showing that restricting to dense path aware policies does not suffice for co-Büchi objectives.

Theorem 5. *There is a graph, two co-Büchi objectives φ_1 and φ_2 with $(\varphi_1 \cap \varphi_2) \neq \emptyset$ and deterministic memoryless policies π_1^σ and π_2^σ that respectively satisfy φ_1 and φ_2, but for any scheduler σ, we have $(\mathcal{G}, \pi_1^\sigma, \pi_2^\sigma, \sigma) \not\models_{\text{a.s.}} \varphi_1 \cap \varphi_2$.*

Proof (sketch). Consider the graph $\mathcal{G}$ depicted in Fig. 4. For $i \in \{1, 2\}$, the "bad" co-Büchi vertex for objective φ_i is v_{1i}. Note that $(\varphi_1 \cap \varphi_2) \neq \emptyset$ since v_0^ω satisfies both objectives. We define $\pi_i(v_1) = v_{1,i}$, otherwise the policies are identical. Observe that π_i satisfies φ_i. However, for any stochastic scheduler, all vertices will be almost-surely visited, which violates both objectives.

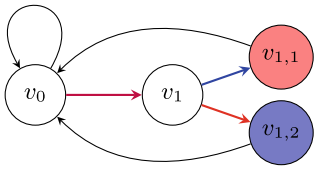

Fig. 4. A graph with two co-Büchi objectives given by bad states $v_{1,1}$ (Red) and $v_{1,2}$ (Blue). Red edges are chosen by π_1 and Blue edges by π_2. (Color figure online)

7.1 A Convention for Co-Büchi Objectives

Consider a strongly-connected graph $\mathcal{G}$, a stochastic scheduler σ, and a co-Büchi objective φ given by a set of "bad" states B. Let $\mathfrak{C}_B$ denote the set of "good" simple cycles in $\mathcal{G}$ that are disjoint from B. A policy $\pi \in \text{Conv}_{\text{co-Büchi}}(\mathcal{G}, \sigma, \varphi)$ proceeds as follows. It randomly chooses a lasso path $\rho \cdot C^\omega$, where ρ is a simple path and $C \in \mathfrak{C}_B$. If π is scheduled in a turn, if C has not yet been reached, it chooses the next vertex on ρ, and otherwise advances on C. Suppose that π is not scheduled, and let π' be the scheduled policy. Let v be the vertex that π would have chosen if it was scheduled and let v' be the vertex that π' chooses. If $v = v'$, we proceed to the next turn. If $v \neq v'$, then π' realizes it is in *conflict* with π: the lasso that π chose differs from the choice of π'. Then, π' randomly chooses a different lasso path $\rho' \cdot C'^\omega$ with $C' \in \mathfrak{C}_B$. Correctness will be derived from the property of co-Büchi objectives that if there is a path that satisfies all

objectives, then there is a lasso $\rho \cdot C^\omega$ with $C \in \mathfrak{C}_B$ that is good for all. The idea is that almost surely, all policies will eventually choose the same lasso path leading to the generated path "stabilizing" on a cycle that is good for all.

We describe π formally. The memory states of π are $M = Paths_{\mathsf{fin}}(\mathcal{G}) \times \mathfrak{C}_B$, where a state $\langle \rho, C \rangle \in M$ means that π chooses cycle C and the path ρ leading to it. The interaction with a scheduler and the other policies generates a random sequence, and π's *local* view of the sequence is $\langle m_0, v_0 \rangle, \langle m_1, v_1 \rangle, \ldots$, where for $i \geq 0$, each $m_i = \langle \eta_i, C_i \rangle \in M$ is a memory state and $v_i \in V$ is a location on $\mathcal{G}$. The policy maintains the invariant that the location in $\mathcal{G}$ matches the first vertex of η, namely $\eta_i[0] = v_i$. We use $\top$ and $\bot$ to respectively denote that π is and is not scheduled. The definition of $\langle m_{i+1}, v_{i+1} \rangle = \pi(m_i, v_i, \top)$ is deterministic and depends on whether the cycle has been reached: (1) C_i has not been reached: then we define $v_{i+1} = \eta[1]$ and $m_{i+1} = \langle \eta[1:], C_i \rangle$, and (2) C_i has been reached: then $\eta_i = \epsilon$ and we define $m_{i+1} = m_i$ and v_{i+1} is the successor of v_i on C. We proceed to define $\pi(m_i, v_i, \bot)$. Let $\langle m, v \rangle = \pi(m_i, v_i, \top)$ be π's choice if it was scheduled, and let v_{i+1} be the vertex chosen by some other policy. If $v = v_{i+1}$, then the update is as above, $m_{i+1} = m$. If $v_{i+1} \neq v$, then π randomly selects $C_{i+1} \in \mathfrak{C}_B$ and a simple path η_{i+1} from v_{i+1} to C_{i+1}.

To prove the following theorem, we construct a Markov chain that simulates the global state of the policies and show that its Bottom Strongly Connected Components (BSCCs) are states in which the policies are in consensus regarding the choice of a lasso path that is good for all. The proof then follows from the property of Markov chains that a random walk reaches a BSCC almost surely.

Theorem 6. *Let $B_1, \ldots, B_N$ be a collection of co-Büchi objectives on a graph $\mathcal{G}$. For every choice of policies that satisfies the co-Büchi condition $\pi_i \in \chi_{B_i}$, for $i \in [N]$, and a fair scheduler σ, we have $(\mathcal{G}, \pi_1, \ldots, \pi_N, \sigma) \models_{\mathsf{a.s.}} \bigwedge_{1 \leq i \leq N} \varphi_i$.*

8 Convention for Decoupling Parity Objectives

We generalize the ideas for decoupling co-Büchi objectives to parity objectives. The construction requires full-history aware policies when $N > 2$. Fix a graph $\mathcal{G}$ and stochastic scheduler σ.

The co-Büchi convention relies on the property that if there is a path that satisfies all objectives, there is such a lasso path. This is no longer the case for parity objectives. For example, consider the graph depicted in Fig. 1(a) and objectives $\varphi_1 = (GF\ell) \wedge (G\neg b_1)$ and $\varphi_2 = (GFr) \wedge (G\neg b_1)$, then a path that satisfies both must cycle between r and ℓ via t_1 and t_2. This is not a simple cycle since both outgoing edges of t_1 and t_2 are traversed infinitely often. We overcome this using the following lemma whose proof can be found in the full version of the paper.

Lemma 1. *Consider parity objectives $\kappa_1, \ldots, \kappa_N : V \to \mathbb{N}$ with $\bigwedge_{i \in [1;N]} \kappa_i \neq \emptyset$. Then, there are memoryless policies $\langle \theta_1, \ldots, \theta_N \rangle : V \to V$ such that $(\mathcal{G}, \theta_1, \ldots, \theta_N, \sigma) \models_{\mathsf{a.s.}} \bigwedge_{i \in [1;N]} \kappa_i$*

For example, Fig. 1(a) depicts two memoryless policies π_1 and π_2 for which we have $(\mathcal{G}, \pi_1, \pi_2, \sigma) \models_{\text{a.s.}} \varphi_1 \wedge \varphi_2$ since the generated random walk almost surely visits each of the vertices ℓ, t_1, t_2, r infinitely often.

Parity Convention. Consider a parity objective given by a colouring function $\kappa_i : V \to \mathbb{N}$. A policy $\pi_i \in \text{Conv}_{\text{Parity}}(\mathcal{G}, \sigma, \kappa)$ randomly chooses an N-tuple of *internal* memoryless policies $\theta_1, \dots \theta_N$ such that $(\mathcal{G}, \theta_1, \dots \theta_N, \sigma) \models_{\text{a.s.}} \kappa_i$. Intuitively, when scheduled, π_i follows θ_i, and θ_j, for $j \neq i$, constitutes a guess of the internal policy that π_j follows. At vertex v, if π_i is scheduled, it proceed to $\theta_i(v)$ and if π_j is scheduled and chooses $u \neq \theta_i(v)$, then π_i observes a conflict and randomly chooses a new N-tuple of internal memoryless policies. In order to observe a conflict, π_i requires knowledge of who is scheduled and their choice, thus it is a full-history aware policy.

We describe π_i formally. Its memory states are $M = (V^V)^N$. The interaction with a scheduler and the other policies generates a random sequence, and π_i's *local* view of the sequence is $\langle m_0, v_0 \rangle, \langle m_1, v_1 \rangle, \dots$, where at time $j \geq 0$, the location in $\mathcal{G}$ is $v_j \in V$ and the memory state is $m_j = \langle \theta_{j,1}, \dots \theta_{j,N} \rangle \in M$. If π_i is scheduled at time j, we define $v_{j+1} = \theta_{j,i}(v_j)$ and the memory state is unchanged, namely $m_{j+1} = m_j$. Suppose that $\pi_{i'}$ is scheduled and chooses u, then we define $u = v_{j+1}$. If $u = \theta_{j,i}(v_j)$, then there is no conflict and $m_{j+1} = m_j$. Otherwise, $u \neq \theta_{j,i}(v_j)$, policy π_i realizes a conflict and randomly chooses $m_{j+1} = \langle \theta_{j+1,1}, \dots \theta_{j+1,N} \rangle \in M$.

The following theorem is obtained by reasoning about the Markov chain that simulates the global state of the policies, carefully identifying consensus global states that are trivially good for all, and showing that they form the BSSCs of the Markov chain. The proof is available in Appendix ??.

Theorem 7. *Consider a strongly-connected graph $\mathcal{G}$, a stochastic scheduler σ, and a collection of N parity objectives $\kappa_1, \dots, \kappa_N$ having $\bigcap_{1 \leq i \leq N} \kappa_i \neq \emptyset$. For every collection of policies $\pi_1, \dots, \pi_N$ that satisfy the convention, i.e., $\pi_i \in Conv_{parity}(\mathcal{G}, \sigma, \kappa_i)$, for $i \in \{1, \dots, N\}$, we have*

$$(\mathcal{G}, \pi_1, \dots, \pi_N, \sigma) \models_{\text{a.s.}} \bigwedge_{1 \leq i \leq N} \kappa_i.$$

9 Discussion

We established sufficient conditions under which local policies for different ω-regular objectives can be composed under a common scheduler to almost-surely satisfy all objectives, showing that Büchi, co-Büchi, and parity objectives require progressively richer scheduler awareness.

Although our focus is on multi-parity objectives, the framework and the existence results for universal local policies naturally extend to any class of specifications implementable via finite-memory strategies, provided the memory bound is common knowledge among all modules. In this setting, each module may guess a finite-memory strategy per agent (rather than a memoryless one) whose bounded memory size is shared across modules.

Interestingly, while our analysis in Sect. 8 shows that full-history awareness is necessary in general, there are special cases that require less information. For instance, when there are only two agents with parity objectives, each can infer the scheduler's choice from the observed transition: if the transition corresponds to its own proposed action, the agent was scheduled; otherwise, the other agent was. Hence, no explicit communication from the scheduler is needed in this case.

For parity objectives, the memory size of each agent typically scales with the number of agents, as each agent must maintain a guess of a cycle for every other agent (including itself). However, if an agent has additional knowledge—e.g., that some others have Büchi or co-Büchi objectives—this requirement can be reduced. When a parity objective is combined with a (co-)Büchi objective, satisfiability can be achieved via a single simple cycle, eliminating the need for additional memory components.

Although we establish the existence of such universal schedulers, our work does not address their convergence properties. In particular, the expected convergence time of the presented protocol is likely to be exponential in the size of the underlying system. Improving this convergence time remains is not discussed in this paper. We conjecture that techniques from reinforcement learning or stochastic approximation could be adapted to accelerate convergence while preserving correctness guarantees, and is left for future work.

Acknowledgements. This work is funded by the following grants: European Research Council under Grant No.: ERC-2020-AdG 101020093, ISF grant no. 1679/21, grant RYC2024-049116, MICIU/AEI/10.13039/501100011033, the ESF+, and Volkswagen Foundation within its Momentum framework under project no. 9C283.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Alpern, B., Schneider, F.B.: Recognizing safety and liveness. Distrib. Comput. **2**(3), 117–126 (1987). https://doi.org/10.1007/BF01782772
2. Anand, A., Nayak, S.P., Schmuck, A.K.: Synthesizing permissive winning strategy templates for parity games. In: Enea, C., Lal, A. (eds.) CAV 2023. LNCS, vol. 13964, pp. 436–458. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-37706-8_22
3. Avni, G., Henzinger, T.A., Chonev, V.: Infinite-duration bidding games. J. ACM **66**(4), 31:1–31:29 (2019). https://doi.org/10.1145/3340295
4. Avni, G., Henzinger, T.A., Mallik, K., Sadhukhan, S., Thejaswini, K.S.: Decoupled planning for multiple omega-regular objectives. CoRR arXiv:2605.13185 (2026). https://doi.org/10.48550/arXiv.2605.13185
5. Avni, G., Mallik, K., Sadhukhan, S.: Auction-based scheduling. In: Finkbeiner, B., Kovács, L. (eds.) TACAS 2024. LNCS, vol. 14572, pp. 153–172. Springer, Cham (2024). https://doi.org/10.1007/978-3-031-57256-2_8
6. Basset, N., Kwiatkowska, M., Topcu, U., Wiltsche, C.: Strategy synthesis for stochastic games with multiple long-run objectives. In: Baier, C., Tinelli, C. (eds.)

TACAS 2015. LNCS, vol. 9035, pp. 256–271. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46681-0_22
7. Basset, N., Kwiatkowska, M., Wiltsche, C.: Compositional strategy synthesis for stochastic games with multiple objectives. Inf. Comput. **261**, 536–587 (2018). https://doi.org/10.1016/j.ic.2017.09.010
8. Billingsley, P.: Probability and Measure, 3rd edn. Wiley, New York (1995). https://doi.org/10.1017/S0013091500004521
9. Chatterjee, K., De Alfaro, L., Henzinger, T.A.: Trading memory for randomness. In: 2004 Proceedings of the First International Conference on the Quantitative Evaluation of Systems, QEST 2004, pp. 206–217. IEEE (2004). https://doi.org/10.1109/QEST.2004.1348035
10. Chatterjee, K., Piterman, N.: Combinations of qualitative winning for stochastic parity games. In: 30th International Conference on Concurrency Theory, CONCUR, pp. 6:1–6:17. LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2019). https://doi.org/10.4230/LIPICS.CONCUR.2019.6
11. De Alfaro, L.: Formal verification of probabilistic systems. Stanford university (1998)
12. De Alfaro, L., Henzinger, T.A.: Concurrent omega-regular games. In: Proceedings Fifteenth Annual IEEE Symposium on Logic in Computer Science (Cat. No. 99CB36332), pp. 141–154. IEEE (2000). https://doi.org/10.1109/LICS.2000.855763
13. Finkbeiner, B., Schewe, S.: Uniform distributed synthesis. In: 20th Annual IEEE Symposium on Logic in Computer Science (LICS 2005), pp. 321–330. IEEE (2005). https://doi.org/10.1109/LICS.2005.53
14. Guenov, M., Barker, S.: Application of axiomatic design and design structure matrix to the decomposition of engineering systems. Syst. Eng. **8**, 29–40 (2005). https://doi.org/10.1002/sys.20015
15. Guruji, A.K., Agarwal, H., Parsediya, D.: Time-efficient A* algorithm for robot path planning. Procedia Technol. **23**, 144–149 (2016). https://doi.org/10.1016/J.PROTCY.2016.03.010
16. Harel, D., Marron, A., Weiss, G.: Behavioral programming. Commun. ACM **55**(7), 90–100 (2012). https://doi.org/10.1145/2209249.2209270
17. Hart, S., Mas-Colell, A.: Uncoupled dynamics do not lead to Nash equilibrium. Am. Econ. Rev. **93**(5), 1830–1836 (2003). https://doi.org/10.1257/000282803322655581
18. Kagan, E., Ben-Gal, I.: A group testing algorithm with online informational learning. IIE Trans. **46**(2), 164–184 (2014). https://doi.org/10.1080/0740817X.2013.803639
19. Klein, D., Manning, C.D.: A* parsing: fast exact viterbi parse selection. In: Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics, pp. 119–126 (2003). https://doi.org/10.3115/1073445.1073461
20. Kupermann, O., Varfi, M.: Synthesizing distributed systems. In: Proceedings 16th Annual IEEE Symposium on Logic in Computer Science, pp. 389–398. IEEE (2001). https://doi.org/10.1109/LICS.2001.932514
21. Lazarus, A.J., Loeb, D.E., Propp, J.G., Stromquist, W.R., Ullman, D.H.: Combinatorial games under auction play. Games Econom. Behav. **27**(2), 229–264 (1999). https://doi.org/10.1006/game.1998.0676
22. Lewis, D.: Convention: A Philosophical Study. Wiley (2008). https://doi.org/10.1002/9780470693711

23. Pneuli, A., Rosner, R.: Distributed reactive systems are hard to synthesize. In: Proceedings [1990] 31st Annual Symposium on Foundations of Computer Science, pp. 746–757. IEEE (1990). https://doi.org/10.1109/FSCS.1990.89597
24. Pnueli, A.: On the extremely fair treatment of probabilistic algorithms. In: Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing, pp. 278–290 (1983). https://doi.org/10.1145/800061.808757
25. Pnueli, A., Zuck, L.D.: Probabilistic verification. Inf. Comput. **103**(1), 1–29 (1993). https://doi.org/10.1006/inco.1993.1012
26. Roijers, D.M., Vamplew, P., Whiteson, S., Dazeley, R.: A survey of multi-objective sequential decision-making. J. Artif. Intell. Res. **48**, 67–113 (2013). https://doi.org/10.1613/jair.3987
27. Schäfer, L., Christianos, F., Hanna, J.P., Albrecht, S.V.: Decoupled reinforcement learning to stabilise intrinsically-motivated exploration. In: Proceedings of the 21st AAMAS, pp. 1146–1154. IFAAMAS (2022). https://doi.org/10.65109/EYJJ1710