Department of Computer Science

PERMISSIVE CONTROLLER SYNTHESIS FOR PROBABILISTIC SYSTEMS

Klaus Dräger
Vojtěch Forejt
Marta Kwiatkowska
David Parker
Mateusz Ujma

RR-14-01

Department of Computer Science, University of Oxford
Wolfson Building, Parks Road, Oxford OX1 3QD

Permissive Controller Synthesis for Probabilistic Systems

Klaus Dräger[3], Vojtěch Forejt[1], Marta Kwiatkowska[1],
David Parker[2], and Mateusz Ujma[1]

[1] Department of Computer Science, University of Oxford, UK
[2] School of Computer Science, University of Birmingham, UK
[3] EECS, Queen Mary, University of London, UK

Abstract. We propose novel controller synthesis techniques for probabilistic systems modelled using stochastic two-player games: one player acts as a controller, the second represents its environment, and probability is used to capture uncertainty arising due to, for example, unreliable sensors or faulty system components. Our aim is to generate robust controllers that are resilient to unexpected system changes at runtime, and flexible enough to be adapted if additional constraints need to be imposed. We develop a *permissive* controller synthesis framework, which generates *multi-strategies* for the controller, offering a choice of control actions to take at each time step. We formalise the notion of permissiveness using penalties, which are incurred each time a possible control action is blocked by a multi-strategy. Permissive controller synthesis aims to generate a multi-strategy that minimises these penalties, whilst guaranteeing the satisfaction of a specified system property. We establish several key results about the optimality of multi-strategies and the complexity of synthesising them. Then, we develop methods to perform permissive controller synthesis using mixed integer linear programming and illustrate their effectiveness on a selection of case studies.

1 Introduction

Probabilistic model checking is used to automatically verify systems with stochastic behaviour. Systems are modelled as, for example, Markov chains, Markov decision processes, or stochastic games, and analysed algorithmically to verify quantitative properties specified in temporal logic. Applications include checking the safe operation of fault-prone systems ("the brakes fail to deploy with probability at most 10^{-6}") and establishing guarantees on the performance of, for example, randomised communication protocols ("the expected time to establish connectivity between two devices never exceeds 1.5 seconds").

A closely related problem is that of *controller synthesis*. This entails constructing a model of some entity that can be controlled (e.g., a robot, a vehicle or a machine) and its environment, formally specifying the desired behaviour of the system, and then generating, through an analysis of the model, a controller that will guarantee the required behaviour. In many applications of controller synthesis, a model of the system is inherently probabilistic. For example, a robot's sensors and actuators may be unreliable, resulting in uncertainty when detecting and responding to its current state; or messages sent wirelessly to a vehicle may fail to be delivered with some probability.

In such cases, the same techniques that underly probabilistic model checking can be used for controller synthesis. For, example, we can model the system as a Markov decision process (MDP), specify a property ϕ in a probabilistic temporal logic such as PCTL and LTL, and then apply probabilistic model checking. This yields an optimal *strategy* (policy) for the MDP, which instructs the controller as to which action should be taken in each state of the model in order to guarantee that ϕ will be satisfied. This approach has been successfully applied in a variety of application domains, to synthesise, for example: control strategies for robots [21], power management strategies for hardware [16], and efficient PIN guessing attacks against hardware security modules [27].

Another important dimension of the controller synthesis problem is the presence of uncontrollable or adversarial aspects of the environment. We can take account of this by phrasing the system model as a *game* between two players, one representing the controller and the other the environment. Examples of this approach include controller synthesis for surveillance cameras [23], autonomous vehicles [11] or real-time systems [1]. In our setting, we use (turn-based) stochastic two-player games, which can be seen as a generalisation of MDPs where decisions are made by two distinct players. Probabilistic model checking of such a game yields a strategy for the controller player which guarantees satisfaction of a property ϕ, regardless of the actions of the environment player.

In this paper, we tackle the problem of synthesising *robust* and *flexible* controllers, which are resilient to unexpected changes in the system at runtime. For example, one or more of the actions that the controller can choose at runtime might unexpectedly become unavailable, or additional constraints may be imposed on the system that make some actions preferable to others. One motivation for our work is its applicability to model-driven runtime control of adaptive systems [5], which uses probabilistic model checking in an online fashion to adapt or reconfigure a system at runtime in order to guarantee the satisfaction of certain formally specified performance or reliability requirements.

We develop novel, *permissive* controller synthesis techniques for systems modelled as stochastic two-player games. Rather than generating *strategies*, which specify a single action to take at each time-step, we synthesise *multi-strategies*, which specify multiple possible actions. As in classical controller synthesis, generation of a multi-strategy is driven by a formally specified quantitative property: we focus on probabilistic reachability and expected total reward properties. The property must be guaranteed to hold, whichever of the specified actions are taken and regardless of the behaviour of the environment. Simultaneously, we aim to synthesise multi-strategies that are as *permissive* as possible, which we quantify by assigning *penalties* to actions. These are incurred when a multi-strategy blocks (does not make available) a given action. Actions can be assigned different penalty values to indicate the relative importance of allowing them. Permissive controller synthesis amounts to finding a multi-strategy whose total incurred penalty is minimal, or below some given threshold.

We formalise the permissive controller synthesis problem and then establish several key theoretical results. In particular, we show that randomised multi-strategies are strictly more powerful than deterministic ones, and we prove that the permissive controller synthesis problem is NP-hard for either class. We also establish upper bounds, showing that the problem is in NP and PSPACE for the deterministic and randomised cases, respectively.

Next, we propose practical methods for synthesising multi-strategies using mixed integer linear programming (MILP) [25]. We give an exact encoding for deterministic multi-strategies and an approximation scheme (with adaptable precision) for the randomised case. For the latter, we prove several additional results that allow us to reduce the search space of multi-strategies. The MILP solution process works incrementally, yielding increasingly permissive multi-strategies, and can thus be terminated early if required. This is well suited to scenarios where time is limited, such as online analysis for runtime control, as discussed above, or "anytime verification" [26]. Finally, we implement our techniques and evaluate their effectiveness on a range of case studies.

This paper is an extended version, with proofs, of [13].

Related work. Permissive strategies in *non*-stochastic games were first studied in [2] for parity objectives, but permissivity was defined solely by comparing enabled actions. Bouyer et al. [3] showed that optimally permissive memoryless strategies exist for reachability objectives and expected penalties, contrasting with our (stochastic) setting, where they may not. The work in [3] also studies penalties given as mean-payoff and discounted reward functions, and [4] extends the results to the setting of parity games. None of [2,3,4] consider stochastic games or even randomised strategies, and they provide purely theoretical results.

As in our work, Kumar and Garg [20] consider control of stochastic systems by dynamically disabling events; however, rather than stochastic games, their models are essentially Markov chains, which the possibility of selectively disabling branches turns into MDPs. Finally, although tackling a rather different problem (counterexample generation), [28] is related in that it also uses MILP to solve probabilistic verification problems.

2 Preliminaries

We denote by $\mathit{Dist}(X)$ the set of discrete probability distributions over a set X. A *Dirac* distribution is one that assigns probability 1 to some $s \in X$. The *support* of a distribution $d \in \mathit{Dist}(X)$ is defined as $\mathit{supp}(d) \overset{\text{def}}{=} \{x \in X \mid d(x) > 0\}$.

Stochastic games. In this paper, we use *turn-based stochastic two-player games*, which we often refer to simply as *stochastic games*. A stochastic game takes the form $\mathsf{G} = \langle S_\Diamond, S_\Box, \overline{s}, A, \delta \rangle$, where $S \overset{\text{def}}{=} S_\Diamond \cup S_\Box$ is a finite set of states, each associated with player $\Diamond$ or $\Box$, $\overline{s} \in S$ is an initial state, A is a finite set of actions, and $\delta : S \times A \to \mathit{Dist}(S)$ is a (partial) probabilistic transition function. An MDP is a stochastic game with $S_\Box = \emptyset$. Each state s of a stochastic game G has a set of *enabled* actions, given by $A(s) \overset{\text{def}}{=} \{a \in A \mid \delta(s, a) \text{ is defined}\}$. The unique player $\circ$ such that $s \in S_\circ$ picks the action $a \in A(s)$ to be taken in state s. Then, the next state is determined randomly according to the distribution $\delta(s,a)$, i.e., a transition to state s' occurs with probability $\delta(s,a)(s')$. A *path* is a (finite or infinite) sequence $\omega = s_0 a_0 s_1 a_1 \dots$ of such transitions through G. We denote by IPath_s (FPath_s) the set of all infinite (finite) paths starting in s. We omit the subscript s when s is the initial state $\overline{s}$.

A *strategy* $\sigma : \mathit{FPath} \to \mathit{Dist}(A)$ for player $\circ \in \{\Diamond, \Box\}$ of G is a resolution of the choices of actions in each state from $S_\circ$, based on the execution so far. In standard fashion [19], a pair of strategies σ and π for $\Diamond$ and $\Box$ induces, for any state s, a probability measure $\mathit{Pr}^{\sigma,\pi}_{\mathsf{G},s}$ over IPath_s. A strategy σ is *deterministic* if $\sigma(\omega)$ is a Dirac distribution for all ω, and *randomised* if not. In this work, we focus purely on *memoryless* strategies, where $\sigma(\omega)$ depends only on the last state of ω, treating the strategy as a function $\sigma : S_\circ \to \mathit{Dist}(A)$. The case of history-dependent strategies is an interesting topic for future research. We write $\Sigma^\circ_{\mathsf{G}}$ for the set of all (memoryless) player $\circ$ strategies in G.

Properties and rewards. In order to synthesise controllers, we need a formal description of their required properties. In this paper, we use two common classes of properties: *probabilistic reachability* and *expected total reward*, which we will express in an extended version of the temporal logic PCTL [18].

For probabilistic reachability, we write properties of the form $\phi = \mathtt{P}_{\bowtie p}[\,\mathtt{F}\ g\,]$, where $\bowtie\ \in \{\leqslant, \geqslant\}$, $p \in [0,1]$ and $g \subseteq S$ is a set of target states, meaning that the probability of reaching a state in g satisfies the bound $\bowtie p$. Formally, for a specific pair of strategies $\sigma \in \Sigma^{\Diamond}_{\mathsf{G}}, \pi \in \Sigma^{\Box}_{\mathsf{G}}$ for G, the probability of reaching g under σ and π is $\mathit{Pr}^{\sigma,\pi}_{\mathsf{G},\overline{s}}(\mathtt{F}\ g) \overset{\text{def}}{=} \mathit{Pr}^{\sigma,\pi}_{\mathsf{G},\overline{s}}(\{s_0 a_0 s_1 a_1 \dots \in \mathit{IPath}_{\overline{s}} \mid s_i \in g \text{ for some } i\})$. We say that ϕ is satisfied under σ and π, denoted $\mathsf{G}, \sigma, \pi \models \phi$, if $\mathit{Pr}^{\sigma,\pi}_{\mathsf{G},\overline{s}}(\mathtt{F}\ g) \bowtie p$.

For rewards, we augment stochastic games with *reward structures*, which are functions of the form $r : S \times A \to \mathbb{R}_{\geqslant 0}$ mapping state-action pairs to non-negative reals. In practice, we often use these to represent "costs" (e.g. elapsed time or energy consumption), despite the terminology "rewards".

The *total reward* for reward structure r along an infinite path $\omega = s_0 a_0 s_1 a_1 \dots$ is $r(\omega) \overset{\text{def}}{=} \sum_{j=0}^{\infty} r(s_j, a_j)$. For strategies $\sigma \in \Sigma^{\Diamond}_{\mathsf{G}}$ and $\pi \in \Sigma^{\Box}_{\mathsf{G}}$, the *expected total reward* is defined as $E^{\sigma,\pi}_{\mathsf{G},\overline{s}}(r) \overset{\text{def}}{=} \int_{\omega \in \mathit{IPath}_{\overline{s}}} r(\omega)\, d\mathit{Pr}^{\sigma,\pi}_{\mathsf{G},\overline{s}}$. For technical reasons, we will always assume that the maximum possible reward $\sup_{\sigma,\pi} E^{\sigma,\pi}_{\mathsf{G},s}(r)$ is finite (which can be checked with an analysis of the game's underlying graph). An expected reward property is written $\phi = \mathtt{R}^r_{\bowtie b}[\,\mathtt{C}\,]$ (where $\mathtt{C}$ stands for *cumulative*), meaning that the expected total reward for r satisfies $\bowtie b$. We say that ϕ is satisfied under strategies σ and π, denoted $\mathsf{G}, \sigma, \pi \models \phi$, if $E^{\sigma,\pi}_{\mathsf{G},\overline{s}}(r) \bowtie b$.

In fact, probabilistic reachability can be easily reduced to expected total rewards. Thus, in the techniques presented in this paper, we focus purely on expected total reward.

Controller synthesis. To perform controller synthesis, we model the system as a stochastic game $\mathsf{G} = \langle S_\Diamond, S_\Box, \overline{s}, A, \delta \rangle$, where player $\Diamond$ represents the controller and player $\Box$ represents the environment. A specification of the required behaviour of the system is a property ϕ, either a probabilistic reachability property $\mathtt{P}_{\bowtie p}[\,\mathtt{F}\ t\,]$ or an expected total reward property $\mathtt{R}^r_{\bowtie b}[\,\mathtt{C}\,]$.

Definition 1 (Sound strategy). *A strategy $\sigma \in \Sigma_{\mathsf{G}}^{\Diamond}$ for player $\Diamond$ in stochastic game* G *is* sound *for a property ϕ if $\mathsf{G}, \sigma, \pi \models \phi$ for any strategy $\pi \in \Sigma_{\mathsf{G}}^{\Box}$.*

The classical *controller synthesis* problem asks whether there is a sound strategy. We can determine whether this is the case by computing the optimal strategy for player $\Diamond$ in game G [12,15]. This problem is known to be in NP $\cap$ co-NP, but, in practice, methods such as value or policy iteration can be used efficiently.

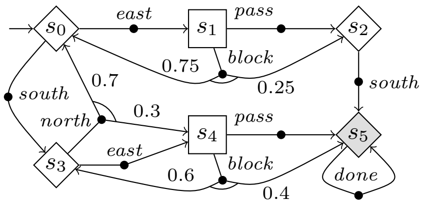


Fig. 1. A stochastic game G for Ex. 1.

Example 1. Fig. 1 shows a stochastic game G, with controller and environment player states drawn as diamonds and squares, respectively. It models the control of a robot moving between 4 locations (s_0, s_2, s_3, s_5). When moving east ($s_0 \to s_2$ or $s_3 \to s_5$), it may be impeded by a second robot, depending on the position of the latter. If it is blocked, there is a chance that it does not successfully move to the next location. We use a reward structure *moves*, which assigns 1 to the controller actions *north*, *east*, *south*, and define property $\phi = \mathtt{R}_{\leqslant 5}^{moves}[\,\mathtt{C}\,]$, meaning that the expected number of moves to reach s_5 is at most 5. A sound strategy (found by minimising *moves*) chooses *south* in s_0 and *east* in s_3, yielding an expected number of moves of 3.5.

3 Permissive Controller Synthesis

We now define a framework for *permissive controller synthesis*, which generalises classical controller synthesis by producing *multi-strategies* that offer the controller flexibility about which actions to take in each state.

3.1 Multi-Strategies

Multi-strategies generalise the notion of strategies, as defined in Section 2.

Definition 2 (Multi-strategy). *A (memoryless)* multi-strategy *for a game $\mathsf{G}{=}\langle S_\Diamond, S_\Box, \overline{s}, A, \delta\rangle$ is a function $\theta{:}S_\Diamond{\to}Dist(2^A)$ with $\theta(s)(\emptyset) = 0$ for all $s \in S_\Diamond$.*

As for strategies, a multi-strategy θ is deterministic if θ always returns a Dirac distribution, and randomised otherwise. We write $\Theta_{\mathsf{G}}^{det}$ and $\Theta_{\mathsf{G}}^{rand}$ for the sets of all deterministic and randomised multi-strategies in G, respectively.

A deterministic multi-strategy θ chooses a set of *allowed actions* in each state $s \in S_\Diamond$, i.e., those in the unique set $B \subseteq A$ for which $\theta(s)(B) = 1$. The remaining actions $A(s) \setminus B$ are said to be *blocked* in s. In contrast to classical controller synthesis, where a strategy σ can be seen as providing instructions about precisely which action to take in each state, in permissive controller synthesis a multi-strategy provides multiple actions, any of which can be taken. A randomised multi-strategy generalises this by selecting a set of allowed actions in state s randomly, according to distribution $\theta(s)$.

We say that a controller strategy σ *complies* with multi-strategy θ if it picks actions that are allowed by θ. Formally (taking into account the possibility of randomisation), σ complies with θ if, for any state s and non-empty subset $B \subseteq A(s)$, there is a distribution $d_{s,B} \in \mathit{Dist}(B)$ such that, for all $a \in A(s)$, $\sigma(s)(a) = \sum_{B \ni a} \theta(s)(B) d_{s,B}(a)$.

Now, we can define the notion of a *sound* multi-strategy, i.e., one that is guaranteed to satisfy a property ϕ when complied with.

Definition 3 (Sound multi-strategy). *A multi-strategy θ for game G is* sound *for a property ϕ if any strategy σ that complies with θ is sound for ϕ.*

Example 2. We return to the stochastic game from Ex. 1 (see Fig. 1) and re-use the property $\phi = \mathtt{R}^{moves}_{\leqslant 5}[\,\mathtt{C}\,]$. The strategy that picks *south* in s_0 and *east* in s_3 results in an expected reward of 3.5 (i.e., 3.5 moves on average to reach s_5). The strategy that picks *east* in s_0 and *south* in s_2 yields expected reward 5. Thus a (deterministic) *multi-strategy* θ that picks $\{\mathit{south}, \mathit{east}\}$ in s_0, $\{\mathit{south}\}$ in s_2 and $\{\mathit{east}\}$ in s_3 is sound for ϕ since the expected reward is always at most 5.

3.2 Penalties and Permissivity

The motivation for multi-strategies is to offer flexibility in the actions to be taken, while still satisfying a particular property ϕ. Generally, we want a multi-strategy θ to be as *permissive* as possible, i.e. to impose as few restrictions as possible on actions to be taken. We formalise the notion of permissivity by assigning *penalties* to actions in the model, which we then use to quantify the extent to which actions are blocked by θ. Penalties provide expressivity in the way that we quantify permissivity: if it is more preferable that certain actions are allowed than others, then these can be assigned higher penalty values.

A *penalty scheme* is a pair (ψ, t), comprising a *penalty function* $\psi : S_\Diamond \times A \to \mathbb{R}_{\geqslant 0}$ and a *penalty type* $t \in \{\mathit{sta}, \mathit{dyn}\}$. The function ψ represents the impact of blocking each action in each controller state of the game. The type t dictates how penalties for individual actions are combined to quantify the permissiveness of a specific multi-strategy. For *static penalties* ($t = \mathit{sta}$), we simply sum penalties across all states of the model. For *dynamic penalties* ($t = \mathit{dyn}$), we take into account the likelihood that blocked actions would actually have been available, by using the *expected sum* of penalty values.

More precisely, for a penalty scheme (ψ, t) and a multi-strategy θ, we define the resulting penalty for θ, denoted $\mathit{pen}_t(\psi, \theta)$ as follows. First, we define the *local* penalty for θ at state s as $\mathit{pen}_{loc}(\psi, \theta, s) = \sum_{B \subseteq A(s)} \sum_{a \notin B} \theta(s, B) \psi(s, a)$. If θ is deterministic, $\mathit{pen}_{loc}(\psi, \theta, s)$ is simply the sum of the penalties of actions that are blocked by θ in s. If θ is randomised, $\mathit{pen}_{loc}(\psi, \theta, s)$ gives the expected penalty value in s, i.e. the sum of penalties weighted by the probability with which θ blocks them in s.

Now, for the static case, we sum the local penalties over all states, i.e. we put $\mathit{pen}_{sta}(\psi, \theta) = \sum_{s \in S_\Diamond} \mathit{pen}_{loc}(\psi, \theta, s)$. For the dynamic case, we use the (worst-case) expected sum of local penalties. We define an auxiliary reward structure

ψ' given by the local penalties: $\psi'(s,a) = pen_{loc}(\psi,\theta,s)$ for all $a \in A(s)$. Then:

$$pen_{dyn}(\psi,\theta) = \sup\{E^{\sigma,\pi}_{\mathsf{G},\overline{s}}(\psi') \mid \sigma \in \Sigma^{\Diamond}_{\mathsf{G}}, \pi \in \Sigma^{\Box}_{\mathsf{G}} \text{ and } \sigma \text{ complies with } \theta\}.$$

3.3 Permissive Controller Synthesis

We can now formally define the central problem studied in this paper.

Definition 4 (Permissive controller synthesis). *Consider a game* G*, a class of multi-strategies* $\star \in \{det, rand\}$*, a property* ϕ*, a penalty scheme* (ψ, t) *and a threshold* $c \in \mathbb{Q}_{\geqslant 0}$*. The* permissive controller synthesis *problem asks: does there exist a multi-strategy* $\theta \in \Theta^{\star}_{\mathsf{G}}$ *that is sound for* ϕ *and satisfies* $pen_t(\psi,\theta) \leqslant c$*?*

Alternatively, in a more quantitative fashion, we can aim to synthesise (if it exists) an *optimally permissive* sound multi-strategy.

Definition 5 (Optimally permissive). *Let* G*,* $\star$*,* ϕ *and* (ψ,t) *be as in Defn. 4. A sound multi-strategy* $\hat{\theta} \in \Theta^{\star}_{\mathsf{G}}$ *is* optimally permissive *if its penalty* $pen_t(\psi,\hat{\theta})$ *equals* $\inf\{pen_t(\psi,\theta) \mid \theta \in \Theta^{\star}_{\mathsf{G}}$ *and* θ *is sound for* $\phi\}$*.*

Example 3. We return to Ex. 2 and consider a static penalty scheme (ψ, sta) assigning 1 to the actions *north*, *east*, *south* (in any state). The deterministic multi-strategy θ from Ex. 2 is optimally permissive for $\phi = \mathtt{R}^{moves}_{\leqslant 5}[\,\mathtt{C}\,]$, with penalty 1 (just *north* in s_3 is blocked). If we instead use $\phi' = \mathtt{R}^{moves}_{\leqslant 16}[\,\mathtt{C}\,]$, the multi-strategy θ' that extends θ by also allowing *north* is now sound and optimally permissive, with penalty 0. Alternatively, the randomised multi-strategy θ'' that picks 0.7:$\{north\}$+0.3:$\{north, east\}$ in s_3 is sound for ϕ with penalty just 0.7.

Next, we establish several fundamental results about the permissive controller synthesis problem. Proofs can be found in the appendix.

Optimality. Recall that two key parameters of the problem are the type of multi-strategy sought (deterministic or randomised) and the type of penalty scheme used (static or dynamic). We first note that *randomised* multi-strategies are strictly more powerful than deterministic ones, i.e. they can be more permissive (yield a lower penalty) for the same property ϕ.

Theorem 1. *The answer to a permissive controller synthesis problem (for either a* static *or* dynamic *penalty scheme) can be "no" for* deterministic *multi-strategies, but "yes" for* randomised *ones.*

This is why we explicitly distinguish between classes of multi-strategies when defining permissive controller synthesis. This situation contrasts with classical controller synthesis, where deterministic strategies are optimal for the same classes of properties ϕ. Intuitively, randomisation is more powerful in this case because of the trade-off between rewards and penalties: similar results exist in, for example, multi-objective controller synthesis on MDPs [14].

Second, we observe that, for the case of static penalties, the optimal penalty value for a given property (the infimum of achievable values) may not actually be achievable by any randomised multi-strategy.

Theorem 2. *For permissive controller synthesis using a* static *penalty scheme, an optimally permissive* randomised *multi-strategy does not always exist.*

If, on the other hand, we restrict our attention to deterministic strategies, then an optimally permissive multi-strategy *does* always exist (since the set of deterministic, memoryless multi-strategies is finite). For randomised multi-strategies with dynamic penalties, the question remains open.

Complexity. Next, we present complexity results for the different variants of the permissive controller synthesis problem. We begin with lower bounds.

Theorem 3. *The permissive controller synthesis problem is NP-hard, for either* static *or* dynamic *penalties, and* deterministic *or* randomised *multi-strategies.*

We prove NP-hardness by reduction from the Knapsack problem, where weights of items are represented by penalties, and their values are expressed in terms of rewards to be achieved. The most delicate part is the proof for randomised strategies, where we need to ensure that the multi-strategy cannot benefit from picking certain actions (corresponding to items being put to the Knapsack) with probability other than 0 or 1. See Appx. A.3 for details. For upper bounds, we have the following.

Theorem 4. *The permissive controller synthesis problem for* deterministic *(resp.* randomised*) strategies is in NP (resp. PSPACE) for* dynamic/static *penalties.*

For deterministic multi-strategies it is straightforward to show NP membership in both the dynamic and static penalty case, since we can guess a multi-strategy satisfying the required conditions and check its correctness in polynomial time. For randomised multi-strategies, with some technical effort we can encode existence of the required multi-strategy as a formula of the existential fragment of the theory of real arithmetic, solvable with polynomial space [7]. See Appx. A.4.

A natural question is whether the PSPACE upper bound for randomised multi-strategies can be improved. We show that this is likely to be difficult, by giving a reduction from the square-root-sum problem. We use a variant of the problem that asks, for positive rationals x_1,. . . ,x_n and y, whether $\sum_{i=1}^{n} \sqrt{x_i} \leqslant y$. This problem is known to be in PSPACE, but establishing a better complexity bound is a long-standing open problem in computational geometry [17].

Theorem 5. *There is a reduction from the square-root-sum problem to the permissive controller synthesis problem with* randomised *multi-strategies, for both* static *and* dynamic *penalties.*

4 MILP-Based Synthesis of Multi-Strategies

We now consider practical methods for synthesising multi-strategies that are sound for a property ϕ and optimally permissive for some penalty scheme. Our methods use mixed integer linear programming (MILP), which optimises an objective function subject to linear constraints that mix both real and integer variables. A variety of efficient, off-the-shelf MILP solvers exists.

An important feature of the MILP solvers we use is that they work incrementally, producing a sequence of increasingly good solutions. Here, that means generating a series of sound multi-strategies that are increasingly permissive. In practice, when resources are constrained, it may be acceptable to stop early and accept a multi-strategy that is sound but not necessarily optimally permissive.

4.1 Deterministic Multi-Strategies

We first consider synthesis of *deterministic* multi-strategies. Here, and in the rest of this section, we assume that the property ϕ is of the form $\mathtt{R}^r_{\geqslant b}[\,\mathtt{C}\,]$. Upper bounds on expected rewards ($\phi = \mathtt{R}^r_{\leqslant b}[\,\mathtt{C}\,]$) can be handled by negating rewards and converting to a lower bound. For the purposes of encoding into MILP, we rescale r and b such that $\sup_{\sigma,\pi} E^{\sigma,\pi}_{\mathsf{G},s}(r) < 1$ for all s, and rescale every (non-zero) penalty such that $\psi(s,a) \geqslant 1$ for all s and $a \in A(s)$.

Static penalties. Fig. 2 shows an encoding into MILP of the problem of finding an optimally permissive deterministic multi-strategy for property $\phi = \mathtt{R}^r_{\geqslant b}[\,\mathtt{C}\,]$ and a *static* penalty scheme (ψ, sta). The encoding uses 5 types of variables: $y_{s,a} \in \{0,1\}$, $x_s \in \mathbb{R}_{\geqslant 0}$, $\alpha_s \in \{0,1\}$, $\beta_{s,a,t} \in \{0,1\}$ and $\gamma_t \in [0,1]$, where $s, t \in S$ and $a \in A$. So the worst-case size of the MILP problem is $\mathcal{O}(|A|\cdot|S|^2\cdot\kappa)$, where κ stands for the longest encoding of a number used.

Variables $y_{s,a}$ encode a multi-strategy θ: $y_{s,a}{=}1$ iff θ allows action a in s (constraint (2) enforces at least one action per state). Variables x_s represent the worst-case expected total reward (for r) from state s, under any controller strategy complying with θ and under any environment strategy. This is captured by constraints (3)–(4) (which amounts to minimising the reward in an MDP). Constraint (1) imposes the required bound of b on the reward from $\overline{s}$.

The objective function minimises the static penalty (the sum of all local penalties) minus the expected reward in the initial state. The latter acts as a tie-breaker between solutions with equal penalties (but, thanks to rescaling, is always dominated by the penalties and therefore does not affect optimality).

As an additional technicality, we need to ensure that the values of x_s are the *least* solution of the defining inequalities, to deal with the possibility of zero reward loops [24]. To achieve this, we use an approach similar to the one taken in [28]. It is sufficient to ensure that $x_s = 0$ whenever the minimum expected reward from s achievable under θ is 0, which is the case if and only if, starting from s, it is possible to avoid ever taking an action with positive reward.

In our encoding, $\alpha_s = 1$ if x_s is positive (constraint (5)). The binary variables $\beta_{s,a,t} = 1$ represent, for each such s and each action a allowed in s, a choice of successor $t \in supp(\delta(s,a))$ (constraint (6)). The variables γ_s then represent a ranking function: if $r(s,a) = 0$, then $\gamma_s > \gamma_{t(s,a)}$ (constraint (8)). If a positive reward could be avoided starting from s, there would in particular be an infinite sequence $s_0, a_1, s_1, \ldots$ with $s_0 = s$ and, for all i, $s_{i+1} = t(s_i, a_i)$ and $r(s_i, a_i) = 0$, and therefore $\gamma_{s_i} > \gamma_{s_{i+1}}$. Since S is finite, this sequence would have to enter a loop, leading to a contradiction.

Minimise: $-\,x_{\overline{s}}\; + \sum_{s\in S_\Diamond} \sum_{a\in A(s)} (1-y_{s,a})\cdot\psi(s,a)$ subject to:

$$x_{\overline{s}} \geqslant b \tag{1}$$

$$1 \leqslant \sum_{a\in A(s)} y_{s,a} \qquad \text{for all } s\in S_\Diamond \tag{2}$$

$$x_s \leqslant \sum_{t\in S} \delta(s,a)(t)\cdot x_t + r(s,a) + (1-y_{s,a}) \qquad \text{for all } s\in S_\Diamond,\ a\in A(s) \tag{3}$$

$$x_s \leqslant \sum_{t\in S} \delta(s,a)(t)\cdot x_t \qquad \text{for all } s\in S_\Box,\ a\in A(s) \tag{4}$$

$$x_s \leqslant \alpha_s \qquad \text{for all } s\in S \tag{5}$$

$$y_{s,a} = (1-\alpha_s) + \sum_{t\in supp(\delta(s,a))} \beta_{s,a,t} \qquad \text{for all } s\in S, a\in A(s) \tag{6}$$

$$y_{s,a} = 1 \qquad \text{for all } s\in S_\Box, a\in A(s) \tag{7}$$

$$\gamma_t < \gamma_s + (1-\beta_{s,a,t}) + r(s,a) \qquad \text{for all } (s,a,t)\in supp(\delta) \tag{8}$$

Fig. 2. MILP encoding for deterministic multi-strategies with static penalties.

Minimise: $z_{\overline{s}}$ subject to $(1),\ldots,(7)$ and:

$$\ell_s = \sum_{a\in A(s)} \psi(s,a)\cdot(1-y_{s,a}) \qquad \text{for all } s\in S_\Diamond \tag{9}$$

$$z_s \geqslant \sum_{t\in S} \delta(s,a)(t)\cdot z_t + \ell_s - c\cdot(1-y_{s,a}) \qquad \text{for all } s\in S_\Diamond,\ a\in A(s) \tag{10}$$

$$z_s \geqslant \sum_{t\in S} \delta(s,a)(t)\cdot z_t \qquad \text{for all } s\in S_\Box,\ a\in A(s) \tag{11}$$

Fig. 3. MILP encoding for deterministic multi-strategies with dynamic penalties.

Dynamic penalties. Next, we show how to compute an optimally permissive sound multi-strategy for a *dynamic* penalty scheme (ψ, dyn). This case is more subtle since the optimal penalty can be infinite. Hence, our solution proceeds in two steps as follows. Initially, we determine if there is *some* sound multi-strategy. For this, we just need to check for the existence of a sound strategy, using standard algorithms for solution of stochastic games [12,15].

If there is no sound multi-strategy, we are done. If there *is*, we use the MILP problem in Fig. 3 to determine the penalty for an optimally permissive sound multi-strategy. This MILP encoding extends the one in Fig. 2 for static penalties, adding variables ℓ_s and z_s, representing the local and the expected penalty in state s, and three extra sets of constraints. Equations (9) and (10) define the expected penalty in controller states, which is the sum of penalties for all disabled actions and those in the successor states, multiplied by their transition probability. The behaviour of environment states is captured by Equation (11), where we only maximise the penalty, without incurring any penalty locally.

The constant c in (10) is chosen to be no lower than any *finite* penalty achievable by a deterministic multi-strategy, a possible value being $\sum_{i=0}^{\infty}(1-p^{|S|})^i\cdot p^{|S|}\cdot i\cdot|S|\cdot pen_{\max}$, where p is the smallest non-zero probability assigned by δ, and $pen_{\max}$ is the maximal local penalty over all states. If the MILP problem has a solution, this is the optimal dynamic penalty over all sound multi-strategies. If not, no deterministic sound multi-strategy has finite penalty and the optimal penalty is ∞ (recall that we established there is *some* sound multi-strategy). In

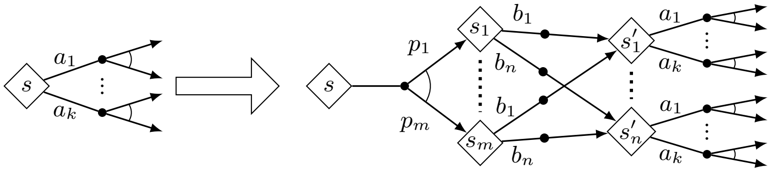


Fig. 4. Transformed game for approximating randomised multi-strategies (Section 4.2).

practice, we might choose a lower value of c than the one above, resulting in a multi-strategy that is sound, but possibly not optimally permissive.

4.2 Approximating Randomised Multi-Strategies

As shown in Section 3, randomised multi-strategies can outperform deterministic ones. The MILP encodings in Fig.s 2 and 3, though, cannot be adapted to the randomised case, since this would need non-linear constraints.

Instead, in this section, we propose an *approximation* which finds the optimal randomised multi-strategy θ in which each probability $\theta(s, B)$ is a multiple of $\frac{1}{M}$ for a given *granularity* M. Any such multi-strategy can then be simulated by a deterministic one on a transformed game, allowing synthesis to be carried out using the MILP-based methods described in the previous section.

The transformed game is illustrated in Fig. 4. For each controller state s, we add two layers of states: *gadgets* s'_j (for $1 \leqslant j \leqslant n$) representing the subsets $B \subseteq A(s)$ with $\theta(s, B) > 0$, and *selectors* s_i (for $1 \leqslant i \leqslant m$), which distribute probability among the gadgets. The s_i are reached from s via a transition using fixed probabilities $p_1, \ldots, p_m$ which need to be chosen appropriately (see below). For efficiency, we want to minimise the number of gadgets n and selectors m for each state s. We now present several results used to achieve this.

First, note that, if $|A(s)| = k$, a randomised multi-strategy chooses probabilities for all $n = 2^k - 1$ non-empty subsets of $A(s)$. Below, we show that it suffices to consider randomised multi-strategies whose support in each state has just two subsets, allowing us to reduce the number of gadgets from $n = 2^k - 1$ to $n = 2$, resulting in a smaller MILP problem to solve for multi-strategy synthesis.

Theorem 6. *1. For a (static or dynamic) penalty scheme (ψ, t) and any sound multi-strategy θ we can construct another sound multi-strategy θ' such that $pen_t(\psi, \theta) \geqslant pen_t(\psi, \theta')$ and $|supp(\theta'(s))| \leqslant 2$ for any $s \in S_\Diamond$.*

2. Furthermore, for static penalties, we can construct θ' such that, for each state $s \in S_\Diamond$, if $supp(\theta'(s)) = \{B_1, B_2\}$, then either $B_1 \subseteq B_2$ or $B_1 \subseteq B_2$.

Part 2 of Theorem 6 states that, for static penalties, we can further reduce the possible multi-strategies that we need to consider. This, however, does not extend to dynamic penalties (see Appx. A.8).

Lastly, we define the probabilities $p_1, \ldots, p_m$ on the transitions to selectors in Fig. 4. We let $m = \lfloor 1 + \log_2 M \rfloor$ and $p_i = \frac{l_i}{M}$, where $l_1 \ldots, l_m \in \mathbb{N}$ are defined recursively as follows: $l_1 = \lceil \frac{M}{2} \rceil$ and $l_i = \lceil \frac{M - (l_1 + \cdots + l_{i-1})}{2} \rceil$ for $2 \leqslant i \leqslant m$. Assuming $n = 2$, as discussed above, this allows us to encode any probability distribution $(\frac{l}{M}, \frac{M-l}{M})$ between two subsets B_1 and B_2.

Name [param.s]	Param. values	States	Ctrl. states	Property	Penalty	Time (s)
cloud	5	8,841	2,177	$\mathtt{P}_{\geqslant 0.9999}[\,\mathtt{F}\ \mathit{deployed}\,]$	0.001	9.08
[vm]	6	34,953	8,705	$\mathtt{P}_{\geqslant 0.999}[\,\mathtt{F}\ \mathit{deployed}\,]$	0.01	72.44
android [r, s]	1, 48	2,305	997	$\mathtt{R}^{\mathit{time}}_{\leqslant 10000}[\,\mathtt{C}\,]$	0.0009	0.58
	2, 48	9,100	3,718		0.0011	10.64
	3, 48	23,137	9,025		0.0013	17.34
mdsm	3	62,245	9,173	$\mathtt{P}_{\leqslant 0.1}[\,\mathtt{F}\ \mathit{deviated}\,]$	52	50.97
[N]	3	62,245	9,173	$\mathtt{P}_{\leqslant 0.01}[\,\mathtt{F}\ \mathit{deviated}\,]$	186	15.84
investor	5,10	10,868	3,344	$\mathtt{R}^{\mathit{profit}}_{\geqslant 4.98}[\,\mathtt{C}\,]$	1	3.32
[$vinit, vmax$]	10, 15	21,593	6,644	$\mathtt{R}^{\mathit{profit}}_{\geqslant 8.99}[\,\mathtt{C}\,]$	1	18.99
team-form	3	12,476	2,023	$\mathtt{P}_{\geqslant 0.9999}[\,\mathtt{F}\ \mathit{done}_1\,]$	0.8980	0.12
[N]	4	96,666	13,793		0.704	2.26
cdmsn [N]	3	1240	604	$\mathtt{P}_{\geqslant 0.9999}[\,\mathtt{F}\ \mathit{prefer}_1]$	2	0.46

Table 1. Experimental results for synthesising optimal deterministic multi-strategies.

The following result states that, by varying the granularity M, we can get arbitrarily close to the optimal penalty for a randomised multi-strategy and, for the case of static penalties, defines a suitable choice of M.

Theorem 7. *Let θ be a sound multi-strategy. For any $\varepsilon > 0$, there is an M and a sound multi-strategy θ' of granularity M satisfying $pen_t(\psi, \theta') - pen_t(\psi, \theta) \leqslant \varepsilon$. Moreover, for static penalties it suffices to take $M = \lceil \sum_{s \in S, a \in A(s)} \frac{\psi(s,a)}{\varepsilon} \rceil$.*

5 Experimental Results

We have implemented our techniques within PRISM-games [9], an extension of the PRISM model checker for performing model checking and strategy synthesis on stochastic games. PRISM-games can thus already be used for (classical) controller synthesis problems on stochastic games. To this, we add the ability to synthesise multi-strategies using the MILP-based method described in Section 4. Our implementation currently uses CPLEX to solve MILP problems. It also supports SCIP and lp_solve, but in our experiments (run on a PC with a 1.7GHz i7 Core processor and 4GB RAM) these were slower in all cases.

We investigated the applicability and performance of our approach on a variety of case studies, some of which are existing benchmark examples and some of which were developed for this work. These are described in detail below and the files used can be found online [29].

Deterministic multi-strategy synthesis. We first discuss the generation of optimal *deterministic* multi-strategies, the results of which are summarised in Table 1. In each row, we first give details of the model: the case study, any parameters used, the number of states ($|S|$) and of controller states ($|S_\Diamond|$). Then, we show the property ϕ used, the penalty value of the optimal multi-strategy and the time to generate it. Below, we give further details for each case study, illustrating the variety of ways that permissive controller synthesis can be used.

cloud: We adapt a PRISM model from [6] to synthesise deployments of services across virtual machines (VMs) in a cloud infrastructure. Our property ϕ specifies that, with high probability, services are deployed to a preferred subset of VMs, and we then assign unit (dynamic) penalties to all actions corresponding to deployment on this subset. The resulting multi-strategy has very low expected penalty (see Table 1) indicating that the goal ϕ can be achieved whilst the controller experiences reduced flexibility only on executions with low probability.

android: We apply permissive controller synthesis to a model created for runtime control of an Android application that provides real-time stock monitoring (see [29] for details). We extend the application to use multiple data sources and synthesise a multi-strategy which specifies an efficient runtime selection of data sources (ϕ bounds the total expected response time). We use static penalties, assigning higher values to actions that select the two most efficient data sources at each time point and synthesise a multi-strategy that always provides a choice of at least two sources (in case one becomes unavailable), while preserving ϕ.

mdsm: Microgrid demand-side management (MDSM) is a randomised scheme for managing local energy usage. A stochastic game analysis [8] previously showed it is beneficial for users to selfishly deviate from the protocol, by ignoring a random back-off mechanism designed to reduce load at busy times. We synthesise a multi-strategy for a (potentially selfish) user, with the goal (ϕ) of bounding the probability of deviation (at either 0.1 or 0.01). The resulting multi-strategy could be used to modify the protocol, restricting the behaviour of this user to reduce selfish behaviour. To make the multi-strategy as permissive as possible, restrictions are only introduced where necessary to ensure ϕ. We also guide where restrictions are made by assigning (static) penalties at certain times of the day.

investor: This example [22] synthesises strategies for a futures market investor, who chooses when to reserve shares, operating in a (malicious) market which can periodically ban him from investing. We generate a multi-strategy that achieves 90% of the maximum expected profit (obtainable by a single strategy) and assign (static) unit penalties to all actions, showing that, after an immediate share purchase, the investor can choose his actions freely and still meet the 90% target.

team-form: This example [10] synthesises strategies for forming teams of agents in order to complete a set of collaborative tasks. Our goal (ϕ) is to guarantee that a particular task is completed with high probability (0.9999). We use (dynamic) unit penalties on all actions of the first agent and synthesise a multi-strategy representing several possibilities for this agent while still achieving the goal.

cdmsn: Lastly, we apply permissive controller synthesis to a model of a protocol for collective decision making in sensor networks (CDMSN) [8]. We synthesise strategies for nodes in the network such that consensus is achieved with high probability (0.9999). We use (static) penalties inversely proportional to the energy associated with each action a node can perform to ensure that the multi-strategy favours more efficient solutions.

Analysis. Unsurprisingly, permissive controller synthesis is slightly more costly to execute than (classical) controller synthesis. But we successfully synthesised

Name†	Par-am.s	States	Ctrl. states	Property	Pen. (det.)	Pen. (randomised)		
						M=100	M=200	M=300
android	1,1	49	10	$\mathtt{P}_{\geqslant 0.9999}[\,\mathtt{F}\ \mathit{done}]$	1.01	0.91	0.905	0.903
	1,10	481	112	$\mathtt{P}_{\geqslant 0.999}[\,\mathtt{F}\ \mathit{done}]$	19.13	18.14*	17.73*	17.58*
cloud	5	8,841	2,177	$\mathtt{P}_{\geqslant 0.9999}[\,\mathtt{F}\ \mathit{deployed}]$	1	0.91	0.905	0.906*
investor	5,10	10,868	3,344	$\mathtt{R}^{\mathit{profit}}_{\geqslant 4.98}[\,\mathtt{C}\,]$	1	1*	1*	0.996*
team-form	3	12,476	2,023	$\mathtt{P}_{\geqslant 0.9999}[\,\mathtt{F}\ \mathit{done}_1\,]$	264	263.96*	263.95*	263.94*

† See Table 1 for parameter names.

* Sound but possibly non-optimal multi-strategy obtained after 5 minute MILP time-out.

Table 2. Experimental results for approximating optimal randomised multi-strategies.

deterministic multi-strategies for a wide range of models and properties, with model sizes ranging up to approximately 100,000 states. The performance and scalability of our method is affected (as usual) by the state space size. But, in particular, it is affected by the number of actions in controller states, since these result in integer MILP variables, which are the most expensive part of the solution. Performance is also sensitive to the penalty scheme used: for example, states with all penalties equal to zero can be dealt with more efficiently.

Randomised multi-strategy synthesis. Finally, Table 2 presents results for approximating optimal *randomised* multi-strategies on several models from Table 1. We show the (static) penalty values for the generated multi-strategies for 3 different levels of precision (i.e. granularities M; see Section 4.2) and compare them to those of the deterministic multi-strategies for the same models.

The MILP encodings for randomised multi-strategies are larger than deterministic ones and thus slower to solve, so we impose a time-out of 5 minutes. We are able to generate a sound multi-strategy for all the examples; in some cases it is optimally permissive, in others it is not (denoted by a * in Table 2). As would be expected, we generally observe smaller penalties with increasing values of M. In the instance where this is not true (*cloud*, M=300), we attribute this to the size of the MILP problem, which grows with M. For all examples, we built randomised multi-strategies with smaller penalties than the deterministic ones.

6 Conclusions

We have presented a framework for permissive controller synthesis on stochastic two-player games, based on generation of multi-strategies that guarantee a specified objective and are optimally permissive with respect to a penalty function. We proved several key properties, developed MILP-based synthesis methods and evaluated them on a set of case studies. Topics for future work include synthesis for more expressive temporal logics and using history-dependent multi-strategies.

Acknowledgements. The authors are part supported by ERC Advanced Grant VERIWARE and EPSRC projects EP/K038575/1 and EP/F001096/1.

References

1. G. Behrmann, A. Cougnard, A. David, E. Fleury, K. Larsen, and D. Lime. UPPAAL-Tiga: Time for playing games! In *Proc. CAV'07*, volume 4590, 2007.
2. J. Bernet, D. Janin, and I. Walukiewicz. Permissive strategies: from parity games to safety games. *ITA*, 36(3):261–275, 2002.
3. P. Bouyer, M. Duflot, N. Markey, and G. Renault. Measuring permissivity in finite games. In *Proc. CONCUR'09*, pages 196–210, 2009.
4. P. Bouyer, N. Markey, J. Olschewski, and M. Ummels. Measuring permissiveness in parity games: Mean-payoff parity games revisited. In *Proc. ATVA'11*, 2011.
5. R. Calinescu, C. Ghezzi, M. Kwiatkowska, and R. Mirandola. Self-adaptive software needs quantitative verification at runtime. *CACM*, 55(9):69–77, 2012.
6. R. Calinescu, K. Johnson, and S. Kikuchi. Compositional reverification of probabilistic safety properties for large-scale complex IT systems. In *LSCITS*, 2012.
7. J. Canny. Some algebraic and geometric computations in PSPACE. In *Proc, STOC'88*, pages 460–467, New York, NY, USA, 1988. ACM.
8. T. Chen, V. Forejt, M. Kwiatkowska, D. Parker, and A. Simaitis. Automatic verification of competitive stochastic systems. In *Proc. TACAS'12*, 2012.
9. T. Chen, V. Forejt, M. Kwiatkowska, D. Parker, and A. Simaitis. PRISM-games: A model checker for stochastic multi-player games. In *Proc. TACAS'13*, 2013.
10. T. Chen, M. Kwiatkowska, D. Parker, and A. Simaitis. Verifying team formation protocols with probabilistic model checking. In *Proc. CLIMA'11*, 2011.
11. T. Chen, M. Kwiatkowska, A. Simaitis, C. Wiltsche. Synthesis for multi-objective stochastic games: An application to autonomous urban driving. In *QEST'13*, 2013.
12. A. Condon. On algorithms for simple stochastic games. *Advances in computational complexity theory, DIMACS Series*, 13:51–73, 1993.
13. K. Draeger, V. Forejt, M. Kwiatkowska, D. Parker, and M. Ujma. Permissive controller synthesis for probabilistic systems. In *Proc. TACAS'14*, 2014.
14. K. Etessami, M. Kwiatkowska, M. Vardi, and M. Yannakakis. Multi-objective model checking of Markov decision processes. *LMCS*, 4(4):1–21, 2008.
15. J. Filar and K. Vrieze. *Competitive Markov Decision Processes.* Springer, 1997.
16. V. Forejt, M. Kwiatkowska, G. Norman, D. Parker, and H. Qu. Quantitative multi-objective verification for probabilistic systems. In *Proc. TACAS'11*, 2011.
17. M. R. Garey, R. L. Graham, and D. S. Johnson. Some np-complete geometric problems. In *STOC '76*, pages 10–22, New York, NY, USA, 1976. ACM.
18. H. Hansson and B. Jonsson. A logic for reasoning about time and reliability. *Formal Aspects of Computing*, 6(5):512–535, 1994.
19. J. Kemeny, J. Snell, and A. Knapp. *Denumerable Markov Chains.* Springer, 1976.
20. R. Kumar and V. Garg. Control of stochastic discrete event systems modeled by probabilistic languages. *IEEE Trans. Automatic Control*, 46(4):593–606, 2001.
21. M. Lahijanian, J. Wasniewski, S. Andersson, and C. Belta. Motion planning and control from temporal logic specifications with probabilistic satisfaction guarantees. In *Proc. ICRA'10*, pages 3227–3232, 2010.
22. A. McIver and C. Morgan. Results on the quantitative mu-calculus qMu. *ACM Transactions on Computational Logic*, 8(1), 2007.
23. N. Ozay, U. Topcu, R. Murray, and T. Wongpiromsarn. Distributed synthesis of control protocols for smart camera networks. In *Proc. ICCPS'11*, 2011.
24. M. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming.* John Wiley and Sons, 1994.
25. A. Schrijver. *Theory of Linear and Integer Programming.* John Wiley & Sons, 1998.

26. N. Shankar. A tool bus for anytime verification. In *Usable Verification*, 2010.
27. G. Steel. Formal analysis of PIN block attacks. *TCS*, 367(1-2):257–270, 2006.
28. R. Wimmer, N. Jansen, E. Ábrahám, B. Becker, and J.-P. Katoen. Minimal critical subsystems for discrete-time Markov models. In *Proc. TACAS'12*. 2012. Extended version available as technical report SFB/TR 14 AVACS 88.
29. `http://www.prismmodelchecker.org/files/tacas14pcs/`.

A Appendix

A.1 Proof of Theorem 1 (Randomisation is Required)

Theorem 1. The answer to a permissive controller synthesis problem (for either a *static* or *dynamic* penalty scheme) can be "no" for *deterministic* multi-strategies, but "yes" for *randomised* ones.

Proof. Consider an MDP with states s, t_1 and t_2, and actions a_1 and a_2, where $\delta(s, a_i)(t_i) = 1$ for $i \in \{1, 2\}$, and t_1, t_2 have self-loops only. Let r be a reward structure assigning 1 to (s, a_1) and 0 to all other state-action pairs, and ψ be a penalty function assigning 1 to (s, a_2) and 0 elsewhere. We then ask whether there is a multi-strategy satisfying $\phi = \mathtt{R}^r_{\geqslant 0.5}[\,\mathtt{C}\,]$ and with penalty at most 0.5.

Considering either static or dynamic penalties, the randomised multi-strategy θ that chooses distribution $\{0.5{:}a_1, 0.5{:}a_2\}$ in s is sound and yields penalty 0.5. However, there is no such deterministic multi-strategy.

A.2 Proof of Theorem 2 (Optimal Strategies)

Theorem 2. For permissive controller synthesis using a *static* penalty scheme, an optimally permissive *randomised* multi-strategy does not always exist.

Proof. Consider a game with states s and t, and actions a and b, where we define $\delta(s, a)(s) = 1$ and $\delta(s, b)(t) = 1$, and t has just a self-loop. The reward structure r assigns 1 to (s, b) and 0 to all other state-action pairs. The penalty function ψ assigns 1 to (s, a) and 0 elsewhere.

Now observe that any multi-strategy which blocks the action a with probability $\varepsilon > 0$ and does not block any other actions incurs penalty ε and is sound for $\mathtt{R}^r_{\geqslant 1}[\,\mathtt{C}\,]$ since any strategy which complies with the multi-strategy satisfies that the action b is taken eventually. Thus the infimum of achievable penalties is 0. However, the multi-strategy that incurs penalty 0, i.e. does not block any actions, is not sound for $\mathtt{R}^r_{\geqslant 1}[\,\mathtt{C}\,]$.

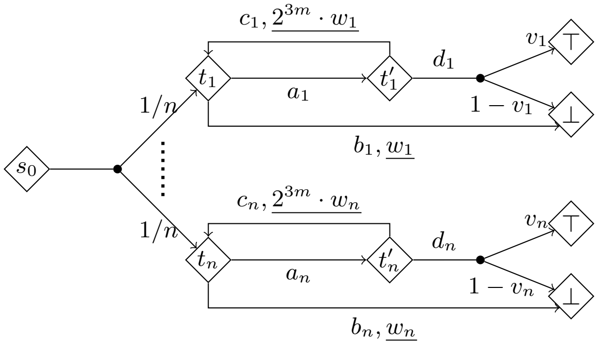
A.3 Proof of Theorem 3 (NP-hardness)

Theorem 3. The permissive controller synthesis problem is NP-hard, for either *static* or *dynamic* penalties, and *deterministic* or *randomised* multi-strategies.

Proof. We start with the case of randomised multi-strategies and static penalties which is the most delicate. Then we analyse the case of randomised multi-strategies and dynamic penalties, and finally show that this case can easily be modified for the remaining two combinations.

Randomised multi-strategies and static penalties. We give a reduction from the Knapsack problem. Let n be the number of items, each of which can either be or not be put in a knapsack, let v_i and w_i be the value and the weight of item i, respectively, and let V and W be the bounds on the value and weight of the items to be picked. We assume that $v_i \leqslant 1$ for every $1 \leqslant i \leqslant n$, and that all numbers v_i and w_i are given as fractions with denominator q.

Let us construct the following MDP, where m is chosen such that $n2^{-m} \leqslant \frac{1}{q}$ and $2^{-m} \cdot W \leqslant \frac{1}{q}$.



We define a reward structure r such that every path reaching $\top$ is assigned cumulative reward 1. The penalties are as given by the underlined expressions.

We show that there is a multi-strategy θ sound for the property $\mathtt{R}^{r}_{\geqslant V/n}[\,\mathtt{C}\,]$ such that $\mathit{pen}_{sta}(\psi, \theta) \leqslant W + 2^{-m} \cdot W$ if and only if the answer to the Knapsack problem is "yes".

In the direction $\Leftarrow$, let $I \subseteq \{1, \ldots, n\}$ be the set of items put in the knapsack. It suffices to define the multi-strategy θ by

- $\theta(t_i')(\{c_i, d_i\}) = 1 - 2^{-4m}$, $\theta(t_i')(\{d_i\}) = 2^{-4m}$, $\theta(t_i)(\{a_i\}) = 1$ for $i \in I$,
- $\theta(t_i')(\{c_i, d_i\}) = 1$, $\theta(t_1)(\{a_i, b_i\}) = 1$ for $i \not\in I$.

In the direction $\Rightarrow$, let us have a multi-strategy θ satisfying the assumptions. Let $P(s \to s')$ denote the lower bound on the probability of reaching s' from s under a strategy which complies with the multi-strategy θ. Denote by $I \subseteq \{1, \ldots, n\}$ the indices i such that $P(t_i \to \top) \geqslant 2^{-m}$.

Let $\beta_i = \theta(t_i)(\{a_i\})$ and $\alpha_i = \theta(t_1')(\{d_i\})$. Observe that:

$$y_i = \beta_i \cdot \sum_{j=0}^{\infty} ((1 - \alpha_i) \cdot \beta_i)^j \cdot \alpha_i \cdot v_i = \frac{\alpha_i \beta_i v_i}{1 - (1 - \alpha_i)\beta_i} = \frac{\alpha_i \beta_i v_i}{1 - \beta_i + \alpha_i \beta_i}$$

because the optimal strategy σ will pick b_i and c_i whenever they are available. Note that for $i \in I$, $\alpha_i \geqslant 2^{-m}(1 - \beta_i)$, since otherwise we have:

$$\frac{\alpha_i \beta_i v_i}{1 - \beta_i + \alpha_i \beta_i} < \frac{\alpha_i \beta_i}{1 - \beta_i + \alpha_i \beta_i} < \frac{2^{-m}(1 - \beta_i)\beta_i}{1 - \beta_i + 2^{-m}(1 - \beta_i)\beta_i} < \frac{2^{-m}\beta_i}{1 + 2^{-m}\beta_i} \leqslant 2^{-m}$$

Hence, $\alpha_i \geqslant 2^{-m}(1 - \beta_i)$, and so:

$$\mathit{pen}_{loc}(\psi, \theta, t_i) + \mathit{pen}_{loc}(\psi, \theta, t_i') = \beta_i w_i + \alpha_i 2^{3m} w_i \geqslant \beta_i w_i + 2^{-m}(1 - \beta_i) 2^{3m} w_i \geqslant w_i$$

We have:

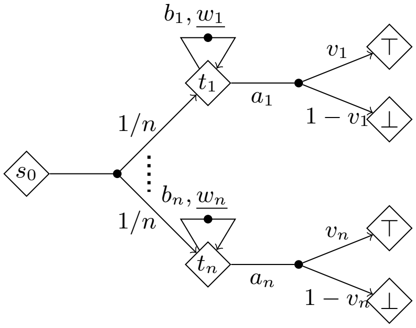
$$\sum_{i\in I} w_i \leqslant \sum_{i\in I} \big(pen_{loc}(\psi,\theta,t_i) + pen_{loc}(\psi,\theta,t_i')\big) \leqslant W + 2^{-m}\cdot W$$

and because $\sum_{i\in I} w_i$ and W are fractions with denominator q, by the choice of m, we can infer that $\sum_{i\in I} w_i \leqslant W$. Similarly:

$$\sum_{i\in I} \frac{1}{n} v_i \geqslant \sum_{i\in I} \frac{1}{n} P(t_i \to \top) \geqslant \Big(\frac{1}{n}\sum_{i=1}^{n} P(t_i \to \top)\Big) - \frac{1}{n}2^{-m}n \geqslant \frac{1}{n}V - 2^{-m}$$

and again, because $\sum_{i\in I} v_i$ and V are fractions with denominator q, by the choice of m we can infer that $\sum_{i\in I} v_i \geqslant V$. Hence in the instance of the knapsack problem it suffices to pick exactly items from I to satisfy the restrictions.

Randomised multi-strategies with dynamic penalties. The proof is analogous to the proof above, we only need to modify the MDP and the computations. For an instance of the Knapsack problem given as before, we construct the following MDP:



We claim that there is a multi-strategy θ sound for the property $\mathtt{R}^{r}_{\geqslant V/n}[\,\mathtt{C}\,]$ such that $pen_{dyn}(\psi,\theta) \leqslant \frac{1}{n}W$ if and only if the answer to the Knapsack problem is "yes".

In the direction ⇐, for $I \subseteq \{1,\dots,n\}$ the set of items in the knapsack we define θ by $\theta(t_i)(\{a_i\}) = 1$ for $i \in I$ and by allowing all actions in every other state.

In the direction ⇒, let us have a multi-strategy θ satisfying the assumptions. Let $P(s \to s')$ denote the lower bound on the probability of reaching s' from s under a strategy which complies with the multi-strategy θ. Denote $I \subseteq \{1,\dots,n\}$ the indices i such that $\theta(t_i)(\{a_i\}) > 0$. Observe that $P(t_i \to \top) = v_i$ if $i \in I$ and $P(t_i \to \top) = 0$ otherwise. Hence:

$$\sum_{i\in I} \frac{1}{n} v_i = \sum_{i\in I} \frac{1}{n} P(t_i \to \top) = \frac{1}{n}\sum_{i=1}^{n} P(t_i \to \top) \geqslant \frac{1}{n}V$$

And for the penalty, denoting $x_i := \theta(t_i)(\{a_i\})$, we get:

$$\frac{1}{n}W \geqslant \mathit{pen}_{\mathit{dyn}}(\psi,\theta) = \frac{1}{n}\sum_{i=0}^{n}\sum_{j=0}^{\infty}(1-x_i)^j x_i w_i = \frac{1}{n}\sum_{i\in I}\sum_{j=0}^{\infty}(1-x_i)^j x_i w_i = \frac{1}{n}\sum_{i\in I} w_i \tag{12}$$

because the strategy that maximises the penalty will pick b_i whenever it is available. Hence in the instance of the knapsack problem it suffices to pick exactly items from I to satisfy the restrictions.

Deterministic multi-strategies and dynamic penalties. The proof is identical to the proof for randomised multi-strategies and dynamic penalties above: observe that the multi-strategy constructed there from an instance of Knapsack is in fact deterministic.

Deterministic multi-strategies and static penalties. The proof is obtained by a small modification of the proof for randomised multi-strategies and dynamic penalties above. Instead of requiring $\mathit{pen}_{\mathit{dyn}}(\psi,\theta) \leqslant \frac{1}{n}W$ we require $\mathit{pen}_{\mathit{sta}}(\psi,\theta) \leqslant W$, and Equation 12 changes to:

$$W \geqslant \mathit{pen}_{\mathit{sta}}(\psi,\theta) = \sum_{i=0}^{n} x_i w_i = \sum_{i\in I} w_i\,.$$

A.4 Proof of Theorem 4 (Upper Bounds)

Theorem 4. The permissive controller synthesis problem for *deterministic* (resp. *randomised*) strategies is in NP (resp. PSPACE) for *dynamic*/*static* penalties.

Proof. We consider the two cases separately.

Deterministic multi-strategies. We start by showing NP membership for deterministic multi-strategies. If the answer to the problem is "yes", then there is a witnessing deterministic multi-strategy, which is of polynomial size. We can guess such a strategy nondeterministically and then in polynomial time verify that the guess is correct. The fact that the multi-strategy is sound and that it achieves the required dynamic penalty can be verified using standard algorithms for computing expected total reward in MDPs, static penalties can be checked by summing up the local penalties.

Randomised multi-strategies. Now we show that the permissive controller synthesis problem is in PSPACE if we restrict to randomised multi-strategies and static penalties. For dynamic penalties the proof is similar.

The proof proceeds by constructing a polynomial-size closed formula Ψ of the existential fragment of $(\mathbb{R},+,\cdot,\leqslant)$ such that Ψ is true if and only if there is a multi-strategy ensuring the required penalty and reward. Because determining the validity of a closed formula of the existential fragment of $(\mathbb{R},+,\cdot,\leqslant)$ is in PSPACE [7], we obtain the desired result.

For the rest of this section, fix an instance of the permissive controller problem as in Defn. 4, with static penalties. We say that a multi-strategy is *winning* if it satisfies the conditions on θ in Defn. 4.

For numbers $\boldsymbol{p} = (p_s)_{s\in S}$ where $0 \leqslant p_s \leqslant 1$ for every $s \in S$, let us consider a game $\mathsf{G}_{\boldsymbol{p}}$ which is obtained from G by applying the transformation from Section 4.2 for approximating randomised multi-strategies (see also Fig. 4) where we fix $n = 2$ and substitute the numbers p_1 and p_2 in the gadget created for s with numbers p_s and $1 - p_s$. We claim that there is a randomised winning multi-strategy in G if and only if there exists a vector $\boldsymbol{p}$ such that there is a deterministic winning multi-strategy in $\mathsf{G}_{\boldsymbol{p}}$. The proof proceeds by establishing a direct correspondence between randomised multi-strategies in G and games $\mathsf{G}_{\boldsymbol{p}}$ and deterministic multi-strategies in them.

Further, let $\Psi[\mathsf{G}_{\boldsymbol{p}}]$ denote the conjunction of the constraints 1-8 from Fig. 2 for the game $\mathsf{G}_{\boldsymbol{p}}$, together with the constraints:

$$\sum_{s\in S_\Diamond} \Big(p_s \cdot \big(\alpha(s,1,1) + \alpha(s,1,2)\big) + (1-p_s)\cdot\big(\alpha(s,2,1)+\alpha(s,2,2)\big)\Big) \leqslant c$$

$$y_{s_i,b_j} \cdot \sum_{a\in A(s'_j)} (1 - y_{s'_j,a})\cdot\psi(s'_j,a) = \alpha(s,i,j) \quad \text{for all } s \in S,\ i,j \in \{1,2\}$$

$$0 \leqslant p_s \leqslant 1 \quad \text{for all } s \in S$$

We get that $\Psi[\mathsf{G}_{\boldsymbol{p}}]$ is satisfiable if and only if there is a deterministic winning multi-strategy.

Note that the formulae $\Psi[\mathsf{G}_{\boldsymbol{p}}]$ for different $\boldsymbol{p}$ differ only at positions where the numbers of $\boldsymbol{p}$ are substituted. Hence, we can create a formula Ψ' which is obtained from $\Psi[G_{\boldsymbol{p}}]$ where each p_s is treated as a variable. From the above we get that the formula Ψ' is satisfiable if and only if there is a randomised winning multi-strategy, and hence we finish the proof by putting $\Psi \equiv \exists \boldsymbol{x}\Psi'$ where $\boldsymbol{x}$ are all variables of Ψ'.

A.5 Proof of Theorem 5 (Square-root-sum Reduction)

Theorem 5. There is a reduction from the square-root-sum problem to the permissive controller synthesis problem with *randomised* multi-strategies, for both *static* and *dynamic* penalties.

Proof. Let $x_1,\ldots,x_n$ and y be numbers giving the instance of the square-root-sum problem, i.e. we aim to determine whether $\sum_{i=1}^n \sqrt{x_i} \leqslant y$. We construct the game from Fig. 5.

The penalties are as given by the underlined numbers, and the rewards $1/x_i$ are awarded under the actions b_i.

Static penalties. We first give the proof for static penalties. We claim that there is a multi-strategy θ sound for the property $\mathtt{R}^r_{\geqslant 1}[\mathtt{C}]$ such that $pen_{sta}(\psi,\theta) \leqslant y$ if and only if $\sum_{i=1}^n \sqrt{x_i} \leqslant y$.

In the direction $\Leftarrow$ let us define a multi-strategy θ by $\theta(t'_i)(\{c'_i\}) = \theta(\bar{t}_i)(\{\bar{c}_i\}) = \sqrt{x_i}$ and $\theta(t'_i)(\{a'_i,c'_i\}) = \theta(\bar{t}_i)(\{\bar{a}_i,\bar{c}_i\}) = 1 - \sqrt{x_i}$, and allowing all actions in all remaining states. We then have: $pen_{sta}(\psi,\theta) = \sum_{i=1}^n 2\cdot\sqrt{x_i}$ and the reward

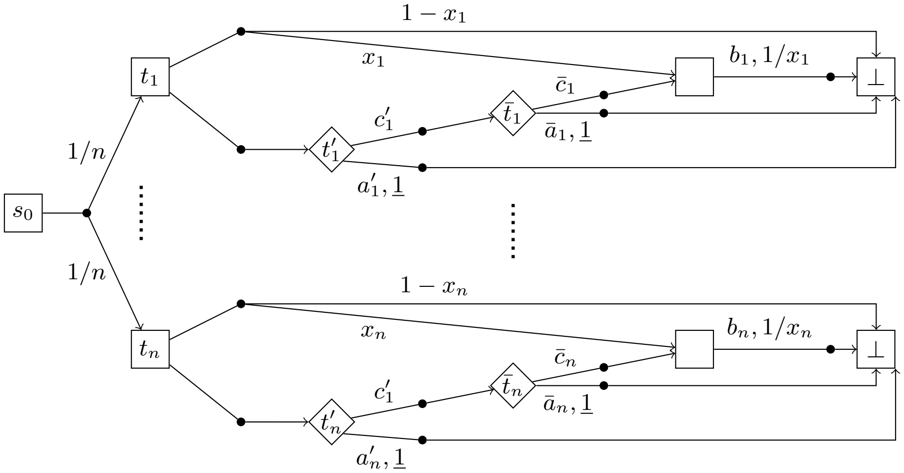


Fig. 5. The game for the proof of Theorem 5.

achieved is:

$$\frac{1}{n}\sum_{i=1}^{n}\min\{x_i\cdot\frac{1}{x_i},\sqrt{x_i}\cdot\sqrt{x_i}\frac{1}{x_i}\} \quad = \quad 1.$$

In the direction ⇒, let θ be an arbitrary multi-strategy sound for the property $\mathtt{R}^r_{\geqslant 1}[\mathtt{C}]$ satisfying $pen_{sta}(\psi,\theta) \leqslant 2\cdot y$. Let $z'_i = \theta(t'_i)(\{c'_i\})$ and $\bar{z}_i = \theta(\bar{t}_i)(\{\bar{c}_i\})$. The reward achieved is:

$$\frac{1}{n}\sum_{i=1}^{n}\min\{x_i\cdot\frac{1}{x_i},z'_i\cdot\bar{z}_i\frac{1}{x_i}\} = \frac{1}{n}\sum_{i=1}^{n}\min\{1,z'_i\cdot\bar{z}_i\frac{1}{x_i}\}$$

which is greater or equal to 1 if and only if $z'_i\cdot\bar{z}_i \geqslant a_i$ for every i. We show that $z'_i+\bar{z}_i \geqslant 2\cdot\sqrt{x_i}$: If both z'_i and $\bar{z}_i$ are greater than $\sqrt{x_i}$, we are done. The case $z'_i,\bar{z}_i \leqslant \sqrt{x_i}$ cannot take place. As for the remaining case, w.l.o.g., suppose that $z'_i = \sqrt{x_i}+p$ and $\bar{z}_i = \sqrt{x_i}-q$ for some non-negative p and q. Then $(\sqrt{x_i}+p)\cdot(\sqrt{x_i}-q) = x_i+(p-q)\sqrt{x_i}-pq$, and for this to be at least x_i we necessarily have $p \geqslant q$, and so $z'_i+\bar{z}_i = \sqrt{x_i}+p+\sqrt{x_i}-q \geqslant 2\cdot\sqrt{x_i}$. Hence, we get that:

$$\sum_{i=1}^{n}2\cdot\sqrt{x_i} \leqslant \sum_{i=1}^{n}\left(z'_i+\bar{z}_i\right) = pen_{sta}(\psi,\theta) \leqslant 2\cdot y.$$

Dynamic penalties. We now proceed with dynamic penalties, where the analysis is similar. Let us use the same game as before, but in addition assume that the penalty assigned to actions c'_i and $\bar{c}'_i$ is equal to 1. We claim that there is a multi-strategy θ sound for the property $\mathtt{R}^r_{\geqslant 1}[\mathtt{C}]$ such that $pen_{dyn}(\psi,\theta) \leqslant 2\cdot y/n$ if and only if $\sum_{i=1}^{n}\sqrt{x_i} \leqslant y$.

In the direction $\Leftarrow$ let us define a multi-strategy θ as before, and obtain $pen_{dyn}(\psi, \theta) = \frac{1}{n} \sum_{i=1}^{n} 2 \cdot \sqrt{y_i}$.

In the direction $\Rightarrow$, let θ be an arbitrary multi-strategy sound for the property $\mathtt{R}^{r}_{\geqslant 1}[\,\mathtt{C}\,]$ satisfying $pen_{dyn}(\psi, \theta) \leqslant 2 \cdot y/n$. Let $z'_i = \theta(t'_i)(\{c'_i\})$, $\bar{z}_i = \theta(\bar{t}_i)(\{\bar{c}_i\})$, $u'_i = \theta(t'_i)(\{a'_i\})$, and $\bar{u}_i = \theta(\bar{t}_i)(\{\bar{a}_i\})$.

As before we can show that $z'_i + \bar{z}_i \geqslant 2 \cdot \sqrt{x_i}$, and so:

$$\frac{1}{n} \sum_{i=1}^{n} 2 \cdot \sqrt{x_i} \leqslant \frac{1}{n} \sum_{i=1}^{n} \left(z'_i + \bar{z}_i\right) \leqslant \frac{1}{n} \sum_{i=1}^{n} \left((z'_i + u'_i) + (1 - u'_i) \cdot (\bar{z}_i + \bar{u}_i)\right)$$
$$= pen_{dyn}(\psi, \theta) \leqslant 2 \cdot y/n.$$

A.6 Proof of Theorem 6.1 (Randomisation on 2 Sets)

Theorem 6.1. For a (static or dynamic) penalty scheme (ψ, t) and any sound multi-strategy θ we can construct another sound multi-strategy θ' such that $pen_t(\psi, \theta) \geqslant pen_t(\psi, \theta')$ and $|supp(\theta'(s))| \leqslant 2$ for any $s \in S_\Diamond$.

Proof. Let θ be a multi-strategy allowing $n > 2$ different sets $A_1, \ldots, A_n$ with non-zero probabilities $\lambda_1, \ldots, \lambda_n$ in $s_1 \in S_\Diamond$. Let p_i and r_i, where $i \in \{1, ..., n\}$, be the penalties and rewards from θ after allowing A_i against an optimal opponent strategy, and p_λ, r_λ the resulting penalty and reward in s_1. The rewards are given in a standard manner as the least fixpoint of linear equations:

- $r_\lambda = \sum_{i=1}^{n} \lambda_i r_i$,
- $r_i = c_i \cdot r_\lambda + d_i$ (where c_i, d_i represent the behaviour in the rest of the system, c_i being the probability of returning to s_1),

and the penalties are $p_\lambda = \sum_{i=1}^{n} \lambda_i p_i$, either with constant p_i (in the case of static penalties) or as the least fixpoint of an analogous system of equations.

Let $S_0 \subseteq S$ be those states from which the opponent can ensure a return to s_1 without accumulating any reward, and for each A_i, let $B_i \subseteq A_i$ contain those actions a with $r(s_1, a) = 0$ and $supp(\delta(s_1, a)) \subseteq S_0$. While constructing θ', we need to take care that for at least one of the A_i in $supp(\theta'(s_1))$, B_i is empty, since otherwise the expected reward drops to 0. One helpful fact is that B_i is empty whenever $r_i > r_\lambda$ (since any $a \in B_i$ could have been used by the opponent to force $r_i \leqslant r_\lambda$).

For each tuple $\mu = (\mu_1, \ldots, \mu_m) \in \mathbb{R}^n$, let $t(\mu) = (r_\mu, p_\mu)$ with $r_\mu = \mu_1 r_1 + \cdots + \mu_n r_n$ and $p_\mu = \mu_1 p_1 + \cdots + \mu_n p_n$. Then the set $T = \{t(\mu) \mid 0 \leqslant \mu_i \leqslant 1, \mu_1 + \cdots + \mu_n = 1\}$ is a bounded convex polygon, with vertices given by images $t(e_i)$ of unit vectors (i.e. Dirac distributions) $e_i = (0, \ldots, 0, 1, 0, \ldots, 0)$, and containing the original point $t(\lambda) = (r_\lambda, p_\lambda)$.

We distinguish several cases, depending on the shape of T:

1. T has non-empty interior. Let $(r_1, p_1), \ldots, (r_m, p_m)$ be its corners in counterclockwise order, each with an associated non-empty set $I_j = \{i \mid t(e_i) = (r_j, p_j)\}$ of vectors. Any given edge $((r_j, p_j)(r_{j+1}, p_{j+1}))$ does not contain

the next corner (r_{j+2}, p_{j+2}) (taking indices modulo m), and since all λ_i are positive, it also does not contain $t(\lambda)$, which is therefore in the interior of T. The idea now is to pick the point (r, p') directly below $t(\lambda) = (r, p)$ on the boundary of T, represent it as a convex combination of adjacent corners $(r, p') = \alpha(r_j, p_j) + (1-\alpha)(r_{j+1}, p_{j+1})$, and let $\theta'(s_1)$ be $\alpha A_u + (1-\alpha)A_v$ for some $u \in I_j, v \in I_{j+1}$. Some care must be taken if (r, p') happens to be a corner (r_j, p_j), as all A_u may have non-empty B_u in this case. However, we then can (since $p_j < p$) instead choose $\varepsilon A_v + (1-\varepsilon)A_u$ for some $v \in I_{j+1}$ and $\varepsilon > 0$ small enough such that $\varepsilon p_{j+1} + (1-\varepsilon)p_j \leqslant p$. Since $t(\lambda)$ is above (r_j, p_j) and inside T, r_{j+1} must be greater than r_j, and therefore $B_v = \emptyset$.

2. T is a vertical line segment, i.e. it is the convex hull of two extreme points (r, p_0) and (r, p_1) with $p_0 < p_1$. In case $r = 0$, we can simply always allow some A_i with $i \in I_0$, minimising the penalty and still achieving reward 0.
   If $r > 0$, there must be at least one A_i with $B_i = \emptyset$. Since all λ_i are positive, $t(\lambda)$ lies inside the line segment, and in particular $p > p_0$. We can therefore choose $\theta'(s_1) = \varepsilon A_i + (1-\varepsilon)A_j$ with the above i and some $j \in I_0$, for an $\varepsilon > 0$ small enough that $\varepsilon p_1 + (1-\varepsilon)p_0 \leqslant p$.
3. T is a non-vertical line segment, i.e. it is the convex hull of two extreme points (r_0, p_0) and (r_1, p_1) with $r_0 < r_1$. Since all λ_i are positive, $t(\lambda)$ is not one of the extreme points, i.e. $t(\lambda) = \alpha(r_0, p_0) + (1-\alpha)(r_1, p_1)$ with $0 < \alpha < 1$. We can therefore choose $\theta'(s_1) = \alpha A_i + (1-\alpha)A_j$ with $i \in I_0, j \in I_1$. Again, since $r_1 > r$, B_j is empty.
4. T consists of a single point (r, p). This can be treated like the second case: either $r = 0$, and we can allow any A_i, or $r > 0$, and there is some A_i with $B_i = \emptyset$, which we can allow.

We now want to show that the reward and penalty of the updated multi-strategy are indeed no worse than before. The rewards $r'_\mu, r'_1, \ldots, r'_n$ are given by the least fixpoint of equations:

- $r'_\mu = \mu r'_u + (1-\mu)r'_v$,
- $r'_i = c'_i \cdot r' + d'_i$ (where c'_i, d'_i represent the behaviour in the rest of the system, taking into account a possibly different optimal opponent strategy).

We then have:

$$\begin{aligned}
r' - r &= (\mu r'_u + (1-\mu)r'_v) - \textstyle\sum_i \lambda_i r_i \\
&= (\mu r'_u + (1-\mu)r'_v) - (\mu r_u + (1-\mu)r_v) + (\mu r_u + (1-\mu)r_v) - \textstyle\sum_i \lambda_i r_i \\
&\geqslant (\mu r'_u + (1-\mu)r'_v) - (\mu r_u + (1-\mu)r_v) \\
&\qquad \text{(by the choice of } \mu, u, v) \\
&= (\mu(c'_u r' + d'_u) + (1-\mu)(c'_v r' + d'_v)) - (\mu(c_u r + d_u) + (1-\mu)(c_v r + d_v)) \\
&\geqslant (\mu(c'_u r' + d'_u) + (1-\mu)(c'_v r' + d'_v)) - (\mu(c'_u r + d'_u) + (1-\mu)(c'_v r + d'_v)) \\
&\qquad (c_i r + d_i \leqslant c'_i r + d'_i \text{ due to optimality of original opponent strategy}) \\
&= (\mu c'_u + (1-\mu)c'_v)(r' - r),
\end{aligned}$$

i.e. $(1-\mu c'_u-(1-\mu)c'_v)(r'-r)\geqslant 0$. By finiteness of rewards and the choice of $\theta(s_1)$, at least one of the return probabilities c'_u, c'_v is less than 1, and thus so is $\mu c'_u+(1-\mu)c'_v$, therefore $r'\geqslant r$.

For static penalties, the fact that the new multi-strategy is no worse than the old one is straightforward from the choice of $\theta'(s_1)$, while dynamic penalties are handled analogously to rewards.

A.7 Proof of Theorem 6.2 (Subset Ordering of Sets)

Theorem 6.2. For static penalties, we can construct θ' such that, for each state $s\in S_\Diamond$, if $\mathit{supp}(\theta'(s))=\{B_1,B_2\}$, then either $B_1\subseteq B_2$ or $B_1\subseteq B_2$.

Proof. To simplify the presentation of this proof, we first establish a useful reduction from the problem of finding a sound multi-strategy in a game $\mathsf{G}=\langle S_\Diamond,S_\Box,\overline{s},A,\delta\rangle$ to the classical controller synthesis problem (of finding a *single* strategy) on an *induced stochastic game* G^i.

Induced game. For game G, the induced game G^i is built by adding intermediate states (s,X) indicating that the multi-strategy has chosen to allow the set of actions $X\subseteq A$. More precisely, $\mathsf{G}^i=\langle S^i_\Diamond,S^i_\Box,\overline{s},A^i,\delta^i\rangle$ where:

- $S^i_\Box=S_\Box\cup\{(s,X)|s\in S_\Diamond\wedge\emptyset\subsetneq X\subseteq A(s)\}$ and $S^i_\Diamond=S_\Diamond$;
- $A^i=A\cup 2^A$ and $\delta^i(s,a)$ is defined as follows:
  - if $s\in S^i_\Diamond$ and $X\subseteq A$, then $\delta^i(s,X)$ is the Dirac distribution on (s,X);
  - if $s=(s',X)\in S^i_\Box$ and $a\in X$, then we define $\delta^i(s,a)=\delta(s',a)$.
  - if $s\in S_\Box$, then $\delta^i(s,a)=\delta(s,a)$.

For a reward structure r on G, we define a corresponding reward structure r^i on G^i by $r^i((s',X),a)=r(s',a)$ for $a\in X\subseteq A(s')$ and $r^i(s',a)=r(s',a)$ for $s'\in S_\Box$.

Lemma 1. *There is a multi-strategy θ in G sound for property $\mathtt{R}^r_{\geqslant b}[\mathtt{C}]$ and satisfying $pen_{sta}(\psi,\theta)\leqslant c$ if and only if there is a strategy σ for player $\Diamond$ in G^i that is sound for $\mathtt{R}^{r^i}_{\geqslant b}[\mathtt{C}]$ and satisfies $\sum_{s\in S_\Diamond}\sum_{B\subseteq 2^A}\sigma(s,B)\psi(s,A(s)\setminus B)\leqslant c$.*

Now we prove Theorem 6.2. In what follows, given a state t and action a we use $E^{\sigma,\pi}_{\mathsf{G},t,a}(r)$ to denote $\sum_{s\in S}\delta(t,a)(s)\cdot E^{\sigma,\pi}_{\mathsf{G},s}(r)$.

Let σ be a strategy in the induced game, and fix s such that σ takes two different actions B and C with probability $p_B>0$ and $p_C>0$ where $B\nsubseteq C$ and $C\nsubseteq B$, and their union with probability p_{BC}. Let π be an optimal strategy for player $\Box$, and let b and c be actions chosen by π in (s,B) and (s,C), respectively. Without loss of generality we assume that π chooses b or c in $(s,B\cup C)$: observe that if there was a better action $b'\in B$ available in $(s,B\cup C)$, we would get that b is not an optimal action in (s,B), and similarly for $c'\in C$. In particular, if $\pi(s,B\cup C)=b$, then $E^{\sigma,\pi}_{\mathsf{G},(s,B)}(r)=E^{\sigma,\pi}_{\mathsf{G},(s,B\cup C}(r)$, and if $\pi(s,B\cup C)=c$, then $E^{\sigma,\pi}_{\mathsf{G},(s,C)}(r)=E^{\sigma,\pi}_{\mathsf{G},(s,B\cup C)}(r)$.

Suppose $E^{\sigma,\pi}_{\mathsf{G},(s,B)}(r) \leqslant E^{\sigma,\pi}_{\mathsf{G},(s,C)}(r)$, we define σ' by modifying σ and picking $B \cup C$ with probability $p_{BC} + p_B$, and B with probability 0. Please note that this operation cannot increase the local penalty in a state, therefore the overall static penalty also does not increase. Subsequently, we argue that the π above is also optimal against σ'. We show that for all t we have $E^{\sigma,\pi}_{\mathsf{G},t}(r) = E^{\sigma',\pi}_{\mathsf{G},t}(r)$. For a sequence $s_0 \cdot a_0 \cdot s_1 \cdot a_1 \cdots s_n$ of states and actions of G, let $\Theta(s_0 \cdot a_0 \cdot s_1 \cdot a_1 \cdots s_n)$ be the set of all paths in G^i which are of the form:

$$s_0 \cdot w_0 \cdot s_1 \cdot w_1 \cdots s_n,$$

where $w_i = a_i$ if $s_i \in S_\Box$, and $w_i = X_i \cdot (s_i, X_i) \cdot a_i$ for some $X_i \subseteq A$ otherwise. By an induction on n one can show that the probability of $\Theta(s_0 \cdot a_0 \cdot s_1 \cdot a_1 \cdots s_n)$ is equal under (σ, π) and (σ', π), for all $s_0 \cdot a_0 \cdot s_1 \cdot a_1 \cdots s_n$. The base step is trivial, for the induction step we use the fact that $\pi((s,B)) = \pi((s, B \cup C))$ and the re-definition of σ'.

From above we immediately get that for all states t and actions a we have $E^{\sigma,\pi}_{\mathsf{G},t,a}(r) = E^{\sigma',\pi}_{\mathsf{G},t,a}(r)$. Hence for every state t, the action $\pi(t)$ satisfies $E^{\sigma',\pi}_{\mathsf{G},t,\pi(t)}(r) \leqslant \min_{a \in A(t)} E^{\sigma',\pi}_{\mathsf{G},t,a}(r)$ since:

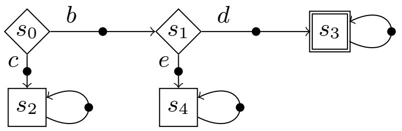
$$E^{\sigma',\pi}_{\mathsf{G},t,\pi(t)}(r) = E^{\sigma,\pi}_{\mathsf{G},t,\pi(t)}(r) \leqslant \min_{a \in A(t)} E^{\sigma,\pi}_{\mathsf{G},t,a}(r) = \min_{a \in A(t)} E^{\sigma',\pi}_{\mathsf{G},t,a}(r)\,.$$

This means that by changing its decision in one state π cannot improve, and so it is optimal (this follows from the correctness of the policy iteration algorithm for MDPs which obtains an optimal controller by changing decision in one state at a time).

For $E^{\sigma,\pi}_{\mathsf{G},(s,B)}(r) > E^{\sigma,\pi}_{\mathsf{G},(s,C)}(r)$, we proceed similarly by choosing $B \cup C$ with probability $p_{BC} + p_C$, and C with probability 0.

A.8 Counterexample to Theorem 6.2 for Dynamic Penalties

As mentioned in Section 4.2, Theorem 6.2 does not hold for the case of dynamic penalties. This is because, in this case, increasing the probability of allowing an action can lead to an increased penalty if one of the successor states has a high expected penalty. An example is shown in Fig. 6, for which we want to reach the goal state s_3 with probability at least 0.5.



action	penalty
b	0
c	1
d	0
e	1

Fig. 6. Counterexample for Theorem 6.2 in case of dynamic penalties.

This implies $\theta(s_0,\{b\}){\cdot}\theta(s_1,\{d\}){\geqslant}0.5$, and so $\theta(s_0,\{b\}){>}0, \theta(s_1,\{d\}){>}0$. If θ satisfies the condition of Theorem 6.2, then $\theta(s_0,\{c\}) = \theta(s_1,\{e\}) = 0$, so an opponent can always use b, forcing an expected penalty of $\theta(s_0,\{b\}) + \theta(s_1,\{d\})$, for a minimal value of $\sqrt{2}$. However, the sound multi-strategy θ with $\theta(s_0,\{b\}){=}\theta(s_0,\{c\}){=}0.5$ and $\theta(s_1,\{d\}){=}1$ achieves a dynamic penalty of just 1.

A.9 Proof of Theorem 7 (Approximations)

Theorem 7. Let θ be a sound multi-strategy. For any $\varepsilon > 0$, there is an M and a sound multi-strategy θ' of granularity M satisfying $pen_t(\psi,\theta') - pen_t(\psi,\theta) \leqslant \varepsilon$. Moreover, for static penalties it suffices to take $M = \lceil \sum_{s\in S, a\in A(s)} \frac{\psi(s,a)}{\varepsilon} \rceil$.

Proof. We deal with the cases of static and dynamic penalties separately.

Static penalties. Let $s \in S$ and $\theta(s)(A_{s,0}) = q_0$, $\theta(s)(A_{s,1}) = q_1$ for $A_{s,0} \subseteq A_{s,1} \subseteq A(s)$. Modify θ by rounding q_0 up and q_1 down to the nearest multiple of $\frac{1}{M}$. The result is again sound (we increase the probability of the smaller set with a possibly higher reward), and the penalty changes by at most $\frac{1}{M}\sum_{a\in A(s)} \psi(s,a)$. Repeat for all s.

Dynamic penalties. Intuitively, the claim follows since by making small changes to the multi-strategy, while not (dis)allowing any new actions, we only cause small changes to the reward and penalty.

Let θ be a multi-strategy and, for $s \in S$, denote $R^\theta(s)$ the optimal reward in s under θ. We have that $R^\theta(s)$ is the solution to the least fixpoint of the equations:

$$R^\theta(s) = \sum\nolimits_{B\in supp(\theta(s))} \theta(s)(B) \min_{a\in B} R^\theta_a(s),$$

$$R^\theta_a(s) = r(s,a) + \sum\nolimits_{t\in S} \delta(s,a)(t) \cdot R^\theta(t).$$

Fix a state t. By Theorem 6.1, we can assume (or replace θ such that) $supp(\theta(t)) = \{A_{t,1}, A_{t,2}\}$ with $\min_{a\in A_{t,1}} R^\theta_a(t) \geqslant \min_{a\in A_{t,2}} R^\theta_a(t)$. We have that $R^{\theta_x}(t) \geqslant R^\theta(t)$ for any multi-strategy θ_x defined by $\theta_x(t)(A_{t,1}) = \theta(t)(A_{t,1}) + x$ and $\theta_x(t)(A_{t,2}) = \theta(t)(A_{t,2}) - x$ for some $x \geqslant 0$, and $\theta_x(s) = \theta(s)$ for all $s \neq t$. Thus, by increasing the probability of allowing $A_{t,1}$ in t the soundness of the multi-strategy is preserved.

Further, for any strategy σ' compliant with θ_x and any π, the penalty when starting in t is equal to:

$$E^{\sigma',\pi}_{\mathsf{G},t}(\psi) = pen_{loc}(\psi,\theta_x,t) + \sum_{a\in A} \lambda'(t)(a) \sum_{t'\in S} \delta(t,a)(t') \cdot (y^{\sigma',\pi}_{t',t} + z^{\sigma',\pi}_{t',t} \cdot E^{\sigma',\pi}_{\mathsf{G},t}(\psi))$$

where $\lambda' = \sigma' \cup \pi$, $y^{\sigma',\pi}_{t',t}$ is the expected penalty incurred when starting from t' before a visit to t, and $z^{\sigma',\pi}_{t',t}$ is the probability of reaching t from t' under σ' and π.

There is a strategy σ compliant with θ which differs from σ' only on t, where $\sum_{a\in A}|\sigma'(s,a)-\sigma(s,a)|\leqslant x$. We have, for any π:

$$\begin{aligned}
E^{\sigma,\pi}_{\mathsf{G},t}(\psi) &= pen_{loc}(\psi,\theta,t)+\sum_{a\in A}\lambda(t,a)\sum_{s\in S}\delta(t,a)(s)\cdot(y^{\sigma,\pi}_{s,t}+z^{\sigma,\pi}_{s,t}\cdot E^{\sigma,\pi}_{\mathsf{G},t}(\psi))\\
&\geqslant pen_{loc}(\psi,\theta_x,t)-x+\sum_{a\in A}(\lambda'(t,a)-x)\sum_{s\in S}\delta(t,a)(s)\cdot(y^{\sigma',\pi}_{s,t}+z^{\sigma',\pi}_{s,t}\cdot E^{\sigma,\pi}_{\mathsf{G},t}(\psi))
\end{aligned}$$

where $\lambda=\sigma\cup\pi$ and the rest is as above.

Thus:

$$\begin{aligned}
E^{\sigma',\pi}_{\mathsf{G},t}(\psi) &= \frac{pen_{loc}(\psi,\theta_x,t)+\sum_{a\in A}\lambda'(t)(a)\sum_{t'\in S}\delta(t,a)(t')\cdot y^{\sigma',\pi}_{t',t}}{1-\sum_{a\in A}\lambda'(t)(a)\sum_{t'\in S}\delta(t,a)(t')\cdot z^{\sigma',\pi}_{t',t}}\\
E^{\sigma,\pi}_{\mathsf{G},t}(\psi) &\geqslant \frac{pen_{loc}(\psi,\theta_x,t)-x+\sum_{a\in A}(\lambda'(t)(a)-x)\sum_{t'\in S}\delta(t,a)(t')\cdot y^{\sigma',\pi}_{t',t}}{1-\sum_{a\in A}(\lambda'(t)(a)-x)\sum_{t'\in S}\delta(t,a)(t')\cdot z^{\sigma',\pi}_{t',t}}
\end{aligned}$$

and so $E^{\sigma',\pi}_{\mathsf{G},t}(\psi)-E^{\sigma,\pi}_{\mathsf{G},t}(\psi)$ goes to 0 as x goes to 0. Hence, $pen_{dyn}(\psi,\theta_x)-pen_{dyn}(\psi,\theta)$ goes to 0 as x goes to 0.

The above gives us that for any error bound ε and a fixed state s there is x such that we can modify the decision of θ in s by x, not violate the soundness property and increase penalty by at most $\varepsilon/|S|$. We thus need to pick M such that $1/M\leqslant x$. To finish the proof, we repeat this procedure for every state s.