PERMISSIVE CONTROLLER SYNTHESIS FOR PROBABILISTIC SYSTEMS

KLAUS DRÄGER, VOJTĚCH FOREJT, MARTA KWIATKOWSKA, DAVID PARKER, AND MATEUSZ UJMA

EECS, Queen Mary, University of London, UK
e-mail address: k.draeger@qmul.ac.uk

Department of Computer Science, University of Oxford, UK
e-mail address: vojtech.forejt@cs.ox.ac.uk

Department of Computer Science, University of Oxford, UK
e-mail address: marta.kwiatkowska@cs.ox.ac.uk

School of Computer Science, University of Birmingham, UK
e-mail address: d.a.parker@cs.bham.ac.uk

Department of Computer Science, University of Oxford, UK
e-mail address: mateusz.ujma@cs.ox.ac.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 permissivity using penalties, which are incurred each time a possible control action is disallowed 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

LOGICAL METHODS
IN COMPUTER SCIENCE

DOI:10.2168/LMCS-???

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 or 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 [22], power management strategies for hardware [16], and efficient PIN guessing attacks against hardware security modules [29].

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 [24], 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 disallows (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) [27]. 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" [28]. Finally, we implement our techniques and evaluate their effectiveness on a range of case studies.

This paper is an extended version of [13], containing complete proofs, optimisations for MILP encodings and experiments comparing performance under two different MILP solvers.

1.1. **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. [26] studies games where the aim of one opponent is to ensure properties of systems against an opponent who can modify the system on-the-fly by removing some transitions.

Finally, although tackling a rather different problem (counterexample generation), [31] 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 $\mathit{supp}(d) = \{x \in X \mid d(x) > 0\}$.

2.1. **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 $S = S_\Diamond \cup S_\Box$ 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 such that the distributions assigned by δ only select elements of S with rational probabilities.

An MDP is a stochastic game in which either $S_{\Diamond}$ or $S_{\Box}$ is empty. Each state s of a stochastic game G has a set of *enabled* actions, given by $A(s) = \{a \in A \,|\, \delta(s,a) \text{ is defined}\}$. The unique player $\circ \in \{\Diamond, \Box\}$ such that $s \in S_{\circ}$ picks an 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* through G is a (finite or infinite) sequence $\omega = s_0a_0s_1a_1\ldots$, where $s_i \in S$, $a_i \in A(s_i)$ and $\delta(s_i,a_i)(s_{i+1}) > 0$ for all i. We denote by $IPath_s$ the set of all infinite paths starting in state s. For a player $\circ \in \{\Diamond, \Box\}$, we denote by $FPath^{\circ}$ the set of all finite paths starting in any state and ending in a state from $S_{\circ}$.

A *strategy* $\sigma : FPath^{\circ} \to Dist(A)$ for player $\circ$ of G is a resolution of the choices of actions in each state from $S_{\circ}$ based on the execution so far, such that only enabled actions in a state are chosen with non-zero probability. In standard fashion [19], a pair of strategies σ and π for $\Diamond$ and $\Box$ induces, for any state s, a probability measure $Pr^{\sigma,\pi}_{\mathsf{G},s}$ 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, in which case we define the strategy as a function $\sigma : S_{\circ} \to Dist(A)$. We write $\Sigma^{\circ}_{\mathsf{G}}$ for the set of all (memoryless) player $\circ$ strategies in G.

2.2. **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

$$Pr^{\sigma,\pi}_{\mathsf{G},\overline{s}}(\mathtt{F}\ g) = Pr^{\sigma,\pi}_{\mathsf{G},\overline{s}}(\{s_0a_0s_1a_1\cdots \in IPath_{\overline{s}} \,|\, s_i \in g \text{ for some } i\}).$$

We say that ϕ is satisfied under σ and π, denoted $\mathsf{G}, \sigma, \pi \models \phi$, if $Pr^{\sigma,\pi}_{\mathsf{G},\overline{s}}(\mathtt{F}\ g) \bowtie p$.

We also augment stochastic games with *reward structures*, which are functions of the form $r : S \times A \to \mathbb{Q}_{\geqslant 0}$ mapping state-action pairs to non-negative rationals. In practice, we often use these to represent "costs" (e.g. elapsed time or energy consumption), despite the terminology "rewards". The restriction to non-negative rewards allows us to avoid problems with non-uniqueness of total rewards, which would require special treatment [30].

Rewards are accumulated along a path and, for strategies $\sigma \in \Sigma^{\Diamond}_{\mathsf{G}}$ and $\pi \in \Sigma^{\Box}_{\mathsf{G}}$, the *expected total reward* is defined as:

$$E^{\sigma,\pi}_{\mathsf{G},\overline{s}}(r) = \int_{\omega = s_0a_0s_1a_1\cdots \in IPath_{\overline{s}}} \sum_{j=0}^{\infty} r(s_j,a_j)\, dPr^{\sigma,\pi}_{\mathsf{G},\overline{s}}.$$

For technical reasons, we will always assume 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); similar assumptions are commonly introduced [25, Section 7]. In our proofs, we will also use $E^{\sigma,\pi}_{\mathsf{G},\overline{s}}(r{\downarrow}s)$ for the expected total reward accumulated before the first visit to s, defined by:

$$E^{\sigma,\pi}_{\mathsf{G},\overline{s}}(r{\downarrow}s) = \int_{\omega = s_0a_0s_1a_1\ldots \in IPath_{\overline{s}}} \sum_{i=0}^{fst(s,\omega)-1} r(s_i,a_i)\, dPr^{\sigma,\pi}_{\mathsf{G},\overline{s}}$$

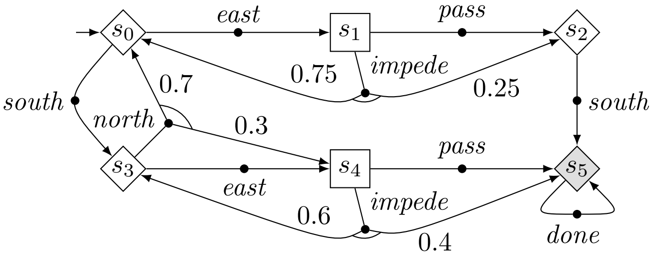


Figure 1: A stochastic game G used as a running example (see Ex. 1).

where $fst(s,\omega)$ is $\min\{i \mid s_i = s\}$ if $\omega = s_0a_0s_1a_1\dots$ contains s_i, and ∞ otherwise.

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 easily be reduced to expected total reward (by replacing any outgoing transitions from states in the target set with a single transition to a sink state labelled with a reward of 1). Thus, in the techniques presented in this paper, we focus purely on expected total reward.

2.3. **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 $\mathtt{P}_{\bowtie p}[\,\mathtt{F}\ g\,]$ or an expected total reward property $\mathtt{R}^r_{\bowtie b}[\,\mathtt{C}\,]$.

Definition 2.1 (Sound strategy)**.** A strategy $\sigma \in \Sigma^\Diamond_\mathsf{G}$ for player $\Diamond$ in stochastic game G is *sound* for a property ϕ if $\mathsf{G},\sigma,\pi \models \phi$ for any strategy $\pi \in \Sigma^\Box_\mathsf{G}$.

To simplify notation, we will consistently use σ and π to refer to strategies of player $\Diamond$ and $\Box$, respectively, and will not always state explicitly that $\sigma \in \Sigma^\Diamond_\mathsf{G}$ and $\pi \in \Sigma^\Box_\mathsf{G}$. Notice that, in Defn. 2.1, strategies σ and π are both memoryless. We could equivalently allow π to range over history-dependent strategies since, for the properties ϕ considered in this paper (probabilistic reachability and expected total reward), the existence of a history-dependent counter-strategy π for which $\mathsf{G},\sigma,\pi \not\models \phi$ implies the existence of a memoryless one.

The classical *controller synthesis* problem asks whether there exists a sound strategy for game G and property ϕ. We can determine whether this is the case by computing the optimal strategy for player $\Diamond$ in 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.

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 ($s_0{\rightarrow}s_2$ or $s_3{\rightarrow}s_5$), it may be impeded by a second robot, depending on the position of the latter. If it is impeded, 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 $\phi = \mathtt{R}^{moves}_{\leqslant 5}[\,\mathtt{C}\,]$, meaning that the expected number of moves to reach s_5 is at most 5 (notice that s_5 is the only state from which all subsequent transitions have reward zero). A sound strategy for ϕ in G (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. They will always be defined for player $\Diamond$ of a game.

Definition 3.1 (Multi-strategy)**.** Let $\mathsf{G} = \langle S_\Diamond, S_\Box, \overline{s}, A, \delta\rangle$ be a stochastic game. A (memoryless) *multi-strategy* for G is a function $\theta : S_\Diamond \to \mathit{Dist}(2^A)$ 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 $\Theta^{det}_{\mathsf{G}}$ and $\Theta^{rand}_{\mathsf{G}}$ 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 $\theta(s)(B) = 1$. When θ is deterministic, we will often abuse notation and write $a \in \theta(s)$ for the actions $a \in B$. The remaining actions $A(s) \setminus B$ are said to be *disallowed* 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 (allows) 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, denoted $\sigma \lhd \theta$, if it picks actions that are allowed by θ. Formally (taking into account the possibility of randomisation), we define this as follows.

Definition 3.2 (Compliant strategy)**.** Let θ be a multi-strategy and σ a strategy for a game G. We say that σ is *compliant* (or that it *complies*) with θ, written $\sigma \lhd \theta$, if, for any state $s \in S_\Diamond$ and non-empty subset $B \subseteq A(s)$, there is a distribution $d_{s,B} \in \mathit{Dist}(B)$ such that, for all $a \in A(s)$, $\sigma(s)(a) = \sum_{B \ni a} \theta(s)(B) \cdot d_{s,B}(a)$.

Example 2. Let us explain the technical definition of a compliant strategy on the game from Ex. 1 (see Fig. 1). Consider a randomised multi-strategy θ that, in s_0, picks $\{\mathit{east}, \mathit{south}\}$ with probability 0.5, $\{\mathit{south}\}$ with probability 0.3, and $\{\mathit{east}\}$ with probability 0.2. A compliant strategy then needs to, for some number $0 \leqslant x \leqslant 1$, pick *south* with probability $0.3 + 0.5 \cdot x$ and *east* with probability $0.2 + 0.5 \cdot (1 - x)$. The number x corresponds to the probability $d_{s_0,\{\mathit{east},\mathit{south}\}}(\mathit{south})$ in the formal definition above.

Hence, a strategy σ that picks *east* and *south* with equal probability 0.5 satisfies the requirements of compliance in state s_0, as witnessed by selecting $x = 0.4$, or, in other words, the distribution $d_{s_0,\{\mathit{east},\mathit{south}\}}$ assigning 0.4 and 0.6 to *south* and *east*, respectively. On the other hand, a strategy that picks *east* with probability 0.8 cannot be compliant with θ.

Each multi-strategy determines a set of compliant strategies, and our aim is to design multi-strategies which allow as many actions as possible, but at the same time ensure that any compliant strategy satisfies some specified property. We define the notion of a *sound* multi-strategy, i.e., one that is guaranteed to satisfy a property ϕ when complied with.

Definition 3.3 (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 3. We return again to the stochastic game from Ex. 1 (see Fig. 1) and re-use the property $\phi = \mathtt{R}^{moves}_{\leqslant 5}[\,\mathtt{C}\,]$. A 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). A strategy that picks *east* in s_0 and *south* in s_2 yields expected reward 5. Thus, a (deterministic) *multi-strategy* θ that picks $\{south, east\}$ in s_0, $\{south\}$ in s_2 and $\{east\}$ in s_3 is sound for ϕ since the expected reward is always at most 5.

3.2. **Penalties and Permissivity.** The motivation behind synthesising 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 disallowed 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, τ), comprising a *penalty function* $\psi : S_\Diamond \times A \to \mathbb{Q}_{\geqslant 0}$ and a *penalty type* $\tau \in \{sta, dyn\}$. The function ψ represents the impact of disallowing each action in each controller state of the game. The type τ dictates how penalties for individual actions are combined to quantify the permissivity of a specific multi-strategy. For *static penalties* ($\tau = sta$), we simply sum penalties across all states of the model. For *dynamic penalties* ($\tau = dyn$), we take into account the likelihood that disallowed actions would actually have been available, by using the *expected sum* of penalty values.

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

Now, for the static case, we sum the local penalties over all states, i.e., we put:

$$pen_{sta}(\psi, \theta) = \sum\nolimits_{s \in S_\Diamond} pen_{loc}(\psi, \theta, s).$$

For the dynamic case, we use the (worst-case) expected sum of local penalties. We define an auxiliary reward structure ψ^θ_{rew} given by the local penalties: $\psi^\theta_{rew}(s, a) = pen_{loc}(\psi, \theta, s)$ for all $s \in S_\Diamond$ and $a \in A(s)$, and $\psi^\theta_{rew}(s, a) = 0$ for all $s \in S_\Box$ and $a \in A(s)$. Then:

$$pen_{dyn}(\psi, \theta, s) = \sup\{E^{\sigma,\pi}_{\mathsf{G},s}(\psi^\theta_{rew}) \mid \sigma \in \Sigma^\Diamond_{\mathsf{G}}, \pi \in \Sigma^\Box_{\mathsf{G}} \text{ and } \sigma \text{ complies with } \theta\}.$$

We use $pen_{dyn}(\psi, \theta) = pen_{dyn}(\psi, \theta, \overline{s})$ to reference the dynamic penalty in the initial state.

3.3. **Permissive Controller Synthesis.** We can now formally define the central problem studied in this paper.

Definition 3.4 (Permissive controller synthesis)**.** Consider a game G, a class of multi-strategies $\star \in \{det, rand\}$, a property ϕ, a penalty scheme (ψ, τ) 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 $pen_{\tau}(\psi, \theta) \leqslant c$?

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

Definition 3.5 (Optimally permissive)**.** Let G, $\star$, ϕ and (ψ, τ) be as in Defn. 3.4. A sound multi-strategy $\hat{\theta} \in \Theta^{\star}_{\mathsf{G}}$ is *optimally permissive* if its penalty $pen_{\tau}(\psi, \hat{\theta})$ equals the infimum $\inf\{pen_{\tau}(\psi, \theta) \,|\, \theta \in \Theta^{\star}_{\mathsf{G}}$ and θ is sound for $\phi\}$.

Example 4. We return to Ex. 3 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. 3 is optimally permissive for $\phi = \mathtt{R}^{moves}_{\leqslant 5}[\,\mathtt{C}\,]$, with penalty 1 (just *north* in s_3 is disallowed). 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 $\{north\}$ with probability 0.7 and $\{north, east\}$ with probability 0.3 in s_3 is sound for ϕ with penalty just 0.7.

It is important to point out that penalties will typically be used for *relative comparisons* of multi-strategies. If two multi-strategies θ and θ' incur penalties x and x' with $x < x'$, then the interpretation is that θ is better than θ'; there is not necessarily any intuitive meaning assigned to the values x and x' themselves. Accordingly, when modelling a system, the penalties of actions should be chosen to reflect the actions' relative importance. This is different from rewards, which usually correspond to a specific measure of the system.

Next, we establish several fundamental results about the permissive controller synthesis problem. Proofs that are particularly technical are postponed to the appendix and we only highlight the key ideas in the main body of the paper.

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) whilst satisfying the same property ϕ.

Theorem 3.6. *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 $\phi = \mathtt{R}^{r}_{\geqslant 0.5}[\,\mathtt{C}\,]$ 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. □

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].

Next, 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 3.7. *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 disallows the action a with probability $\varepsilon > 0$ and allows all 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 leads to action b being taken eventually. Thus, the infimum of achievable penalties is 0. However, the multi-strategy that incurs penalty 0, i.e. allows all actions, is not sound for $\mathtt{R}^{r}_{\geqslant 1}[\,\mathtt{C}\,]$. □

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 3.8. *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 into the Knapsack) with probability other than 0 or 1. See Appx. A.1 for details. For upper bounds, we have the following.

Theorem 3.9. *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.2. 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.

Theorem 3.10. *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.*

We use a variant of the problem that asks, given positive rationals $x_1,\ldots,x_n$ 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]. See Appx. A.3 for details.

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 computational resources are constrained, it may be acceptable to stop early and accept a multi-strategy that is sound but not necessarily optimally permissive.

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 $a \in A(s)$.

We begin by discussing the synthesis of deterministic multi-strategies, first for static penalties and then for dynamic penalties. Subsequently, we present an approach to synthesising approximations to optimal randomised multi-strategies. In each case, we describe encodings into MILP problems and prove their correctness. We conclude this section with a brief discussion of ways to optimise the MILP encodings. Then, in Section 5, we investigate the practical applicability of our techniques.

4.1. Deterministic Multi-Strategies with 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: $y_{s,a} \in \{0,1\}$, $x_s \in [0,1]$, $\alpha_s \in \{0,1\}$, $\beta_{s,a,t} \in \{0,1\}$ and $\gamma_t \in [0,1]$, where $s,t \in S$ and $a \in A$. 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 as follows: $y_{s,a}$ has value 1 iff θ allows action a in $s \in S_\Diamond$ (constraint (4.2) enforces at least one allowed 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 (4.3)–(4.4) (which are analogous to the linear constraints used when minimising the reward in an MDP). Constraint (4.1) puts 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

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

$$x_{\overline{s}} \geqslant b \tag{4.1}$$

$$1 \leqslant \sum\nolimits_{a\in A(s)} y_{s,a} \qquad \text{for all } s\in S_{\Diamond} \tag{4.2}$$

$$x_s \leqslant \sum\nolimits_{t\in S} \delta(s,a)(t)\cdot x_t + r(s,a) + (1-y_{s,a}) \qquad \text{for all } s\in S_{\Diamond},\ a\in A(s) \tag{4.3}$$

$$x_s \leqslant \sum\nolimits_{t\in S} \delta(s,a)(t)\cdot x_t + r(s,a) \qquad \text{for all } s\in S_{\Box},\ a\in A(s) \tag{4.4}$$

$$x_s \leqslant \alpha_s \qquad \text{for all } s\in S \tag{4.5}$$

$$y_{s,a} = (1-\alpha_s) + \sum\nolimits_{t\in \mathit{supp}(\delta(s,a))} \beta_{s,a,t} \qquad \text{for all } s\in S, a\in A(s) \tag{4.6}$$

$$y_{s,a} = 1 \qquad \text{for all } s\in S_{\Box}, a\in A(s) \tag{4.7}$$

$$\gamma_t < \gamma_s + (1-\beta_{s,a,t}) + r(s,a) \qquad \text{for all } s,\ a,\ t \text{ with } t\in \mathit{supp}(\delta(s,a)) \tag{4.8}$$

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

$$\text{Minimise: } z_{\overline{s}} \text{ subject to (4.1),...,(4.8) and:}$$

$$\ell_s = \sum\nolimits_{a\in A(s)} \psi(s,a)\cdot(1-y_{s,a}) \qquad \text{for all } s\in S_{\Diamond} \tag{4.9}$$

$$z_s \geqslant \sum\nolimits_{t\in S} \delta(s,a)(t)\cdot z_t + \ell_s - c\cdot(1-y_{s,a}) \qquad \text{for all } s\in S_{\Diamond},\ a\in A(s) \tag{4.10}$$

$$z_s \geqslant \sum\nolimits_{t\in S} \delta(s,a)(t)\cdot z_t \qquad \text{for all } s\in S_{\Box},\ a\in A(s) \tag{4.11}$$

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

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 the values of x_s are the *least* solution of the defining inequalities, to deal with the possibility of zero reward loops. To achieve this, we use an approach similar to the one taken in [31]. It is sufficient to ensure that $x_s = 0$ whenever the minimum expected reward from s achievable under θ is 0, which is true 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 (4.5)). The binary variables $\beta_{s,a,t} = 1$ represent, for each such s and each action a allowed in s, a choice of successor $t = t(s,a) \in \mathit{supp}(\delta(s,a))$ (constraint (4.6)). The variables γ_s then represent a ranking function: if $r(s,a) = 0$, then $\gamma_s > \gamma_{t(s,a)}$ (constraint (4.8)). 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, either (i) $x_{s_i} > x_{s_{i+1}}$, or (ii) $x_{s_i} = x_{s_{i+1}}$, $s_{i+1} = t(s_i, a_i)$ and $r(s_i, a_i) = 0$, and therefore $\gamma_{s_i} > \gamma_{s_{i+1}}$. This means that the sequence $(x_{s_0}, \gamma_{s_0}), (x_{s_1}, \gamma_{s_1}), \ldots$ is (strictly) decreasing w.r.t. the lexicographical order, but at the same time S is finite, and so this sequence would have to enter a loop, which is a contradiction.

Correctness. Before proving the correctness of the encoding (stated in Theorem 4.2, below), we prove the following auxiliary lemma that characterises the reward achieved under a multi-strategy in terms of a solution of a set of inequalities.

Lemma 4.1. *Let $\mathsf{G} = \langle S_\Diamond, S_\Box, \overline{s}, A, \delta\rangle$ be a stochastic game, $\phi = \mathtt{R}^r_{\geqslant b}[\,\mathtt{C}\,]$ a property, (ψ, sta) a* static *penalty scheme and θ a deterministic multi-strategy. Consider the inequalities:*

$$\begin{array}{lll} x_s & \leqslant & \min_{a\in\theta(s)} \sum_{s'\in S} \delta(s,a)(s')x_{s'} + r(s,a) \quad \textit{for } s\in S_\Diamond \\ x_s & \leqslant & \min_{a\in A(s)} \sum_{s'\in S} \delta(s,a)(s')x_{s'} + r(s,a) \quad \textit{for } s\in S_\Box. \end{array}$$

Then the following hold:

- *$\bar{x}_s = \inf_{\sigma\lhd\theta,\pi} E^{\sigma,\pi}_{\mathsf{G},s}(r)$ is a solution to the above inequalities.*
- *A solution $\bar{x}_s$ to the above inequalities satisfies $\bar{x}_s \leqslant \inf_{\sigma\lhd\theta,\pi} E^{\sigma,\pi}_{\mathsf{G},s}(r)$ for all s whenever the following condition holds: for every s with $\bar{x}_s > 0$, every $\sigma \lhd \theta$ and every π there is a path $\omega = s_0a_0\dots s_na_n$ starting in s that satisfies $Pr^{\sigma,\pi}_{\mathsf{G},s}(\omega) > 0$ and $r(s_n, a_n) > 0$.*

Proof. The game G, together with θ, determines a Markov decision process $\mathsf{G}^\theta = \langle\emptyset, S_\Diamond \cup S_\Box, \overline{s}, A, \delta'\rangle$ in which the choices disallowed by θ are removed, i.e. $\delta'(s,a)$ is equal to $\delta(s,a)$ for every $s \in S_\Box$ and every $s\in S_\Diamond$ with $a \in \theta(s)$, and is undefined for any other combination of s and a. We have:

$$\inf_{\sigma\lhd\theta,\pi} E^{\sigma,\pi}_{\mathsf{G},s}(r) = \inf_{\bar{\sigma}} E^{\bar{\sigma}}_{\mathsf{G}^\theta,s}(r)$$

since, for any strategy pair $\sigma \lhd \theta$ and π in G, there is a strategy $\bar{\sigma}$ in G^θ which is defined, for every finite path ω of G^θ ending in t, by $\bar{\sigma}(\omega) = \sigma(\omega)$ or $\bar{\sigma}(\omega) = \pi(\omega)$, depending on whether $t \in S_\Diamond$ or $t \in S_\Box$, and which satisfies $E^{\sigma,\pi}_{\mathsf{G},s}(r) = E^{\bar{\sigma}}_{\mathsf{G}^\theta,s}(r)$. Similarly, a strategy $\bar{\sigma}$ for G^θ induces a compliant strategy σ and a strategy π defined for every finite path ω of G ending in $S_\Diamond$ (resp. $S_\Box$) by $\sigma(\omega) = \bar{\sigma}(\omega)$ (resp. $\pi(\omega) = \bar{\sigma}(\omega)$).

The rest is then the following simple application of results from the theory of Markov decision processes. The first item of the lemma follows from [25, Theorem 7.1.3], which gives a characterisation of values in MDPs in terms of Bellman equations; the inequalities in the lemma are in fact a relaxation of these equations. For the second part of the lemma, observe that if, $\inf_{\sigma\lhd\theta,\pi} E^{\sigma,\pi}_{\mathsf{G},s}(r)$ is infinite, then the claim holds trivially. Otherwise, from the assumption on the existence of ω we have that, under any compliant strategy, there is a path $\omega' = s_0a_0s_1\dots s_n$ of length at most $|S|$ in G^θ such that $\inf_{\sigma\lhd\theta,\pi} E^{\sigma,\pi}_{\mathsf{G},s_n}(r) = 0$ (otherwise the reward would be infinite) and so $\bar{x}_{s_n} = 0$. We can thus apply [25, Proposition 7.3.4], which states that a solution to our inequalities gives optimal values whenever under any strategy the probability of reaching a state s with $x_s = 0$ is 1. Note that the result of [25] applies for maximisation of reward in "negative models"; our problem can be easily reduced to this setting by multiplying the rewards by -1 and looking for maximising (instead of minimising) strategies. □

Theorem 4.2. *Let G be a game, $\phi = \mathtt{R}^r_{\geqslant b}[\,\mathtt{C}\,]$ a property and (ψ, sta) a* static *penalty scheme. There is a sound multi-strategy in G for ϕ with penalty p if and only if there is an optimal assignment to the MILP instance from Fig. 2 which satisfies $p = \sum_{s\in S_\Diamond}\sum_{a\in A(s)}(1-y_{s,a})\cdot\psi(s,a)$.*

Proof. We prove that every multi-strategy θ induces a satisfying assignment to the variables such that the static penalty under θ is $\sum_{s\in S_\Diamond}\sum_{a\in A(s)}(1-y_{s,a})\cdot\psi(s,a)$, and vice versa. The theorem then follows from the rescaling of rewards and penalties that we performed.

We start by proving that, given a sound multi-strategy θ, we can construct a satisfying assignment $\{\bar{y}_{s,a}, \bar{x}_s, \bar{\alpha}_s, \bar{\beta}_{s,a,t}, \bar{\gamma}_t\}_{s,t\in S, a\in A}$ to the constraints from Fig. 2. For $s \in S_\Diamond$ and $a \in A(s)$ we set $\bar{y}_{s,a} = 1$ if $a \in \theta(s)$, and otherwise we set $\bar{y}_{s,a} = 0$. This gives satisfaction of contraint (4.2). For $s \in S_\Box$ and $a \in A(s)$ we set $\bar{y}_{s,a} = 1$, ensuring satisfaction of (4.7). We then put $\bar{x}_s = \inf_{\sigma \lhd \theta, \pi} E^{\sigma,\pi}_{\mathsf{G},s}(r)$. By the first part of Lemma 4.1 we get that constraints (4.1), (4.3) (for $a \in \theta(s)$) and (4.4) are satisfied. Constraint (4.3) for $a \notin \theta(s)$ is satisfied because in this case $\bar{y}_{s,a} = 0$, and so the right-hand side is always at least 1.

We further set $\bar{\alpha}_s = 1$ if $x_s > 0$ and $\bar{\alpha}_s = 0$ if $x_s = 0$, thus satisfying constraint (4.5). For a state s, let d_s be the maximum distance to a positive reward. Formally, the values d_s are defined inductively by putting $d_s = 0$ for any state s such that we have $r(s,a) > 0$ for all $a \in A(s)$, and then for any other state s:

$$d_s = 1 + \min_{a \in \theta(s), r(s,a)=0} \max_{\delta(s,a)(t)>0} d_t \quad \text{if } s \in S_\Diamond$$

$$d_s = 1 + \min_{a \in A(s), r(s,a)=0} \max_{\delta(s,a)(t)>0} d_t \quad \text{if } s \in S_\Box$$

Put $d_s = \bot$ if d_s was not defined by the above. For s such that $d_s \neq \bot$, we put $\bar{\gamma}_s = d_s/|S|$, and for every a we choose t such that $d_t < d_s$, and set $\bar{\beta}_{s,a,t} = 1$, leaving $\bar{\beta}_{s,a,t} = 0$ for all other t. For s such that $d_s = \bot$ we define $\bar{\gamma}_s = 0$ and for all a and t put $\bar{\beta}_{s,a,t} = 0$. This ensures the satisfaction of the remaining constraints.

In the opposite direction, assume that we are given a satisfying assignment. Firstly, we create a game G' from G by making any states s with $\bar{x}_s = 0$ sink states (i.e. imposing a self-loop with no penalty on s and removing all other transitions). Any sound multi-strategy θ for ϕ in G' directly gives a sound multi-strategy θ' for ϕ in G defined by $\theta'(s) = \theta(s)$ for states $s \in S_\Diamond$ with $x_s > 0$, and otherwise letting θ allow all available actions.

We construct θ for G' by putting $\theta(s) = \{a \in A(s) \mid \bar{y}_{s,a} = 1\}$ for all $s \in S_\Diamond$ with $\bar{x}_s > 0$, and by allowing the self-loop in the states $s \in S_\Diamond$ with $x_s = 0$; note that $\theta(s)$ is non-empty by constraint (4.2). First, by definition, the multi-strategy yields the penalty $\sum_{s \in S_\Diamond} \sum_{a \in A(s)} (1 - \bar{y}_{s,a}) \cdot \psi(s,a)$. Next, we will show that θ satisfies the assumption of the second part of Lemma 4.1, from which we get that:

$$\inf_{\sigma \lhd \theta, \pi} E^{\sigma,\pi}_{\mathsf{G}',s}(r) \geqslant \bar{x}_s$$

which, together with constraint (4.1) being satisfied, gives us the desired result.

Consider any s such that $\inf_{\sigma \lhd \theta, \pi} E^{\sigma,\pi}_{\mathsf{G}',s}(r) > 0$. Then we have $\bar{x}_s > 0$ (by the definition of G'). Let us fix any $\sigma \lhd \theta$ and any π, and let $s_0 = s$. We show that there is a path ω satisfying the assumption of the lemma. We build $\omega = s_0 \ldots s_n a_n$ inductively, to satisfy: (i) $r(s_n, a_n) > 0$, (ii) $\bar{x}_{s_i} \geqslant \bar{x}_{s_{i-1}}$ for all i, and (iii) for any sub-path $s_i a_i \ldots s_j$ with $\bar{x}_{s_i} = \bar{x}_{s_j}$ we have that $\bar{\gamma}_{s_k} < \bar{\gamma}_{s_{k-1}}$ for all $i+1 \leqslant k \leqslant j$.

Assume we have defined a prefix $s_0 a_0 \ldots s_i$ to satisfy conditions (ii) and (iii). We put a_i to be the action picked by σ (or π) in s_i. If $r(s_i, a_i) > 0$, we are done. Otherwise, we pick s_{i+1} as follows:

- If there is $s' \in \mathit{supp}(\delta(s_i, a_i))$ with $\bar{x}_{s'} > \bar{x}_s$, then we put $s_{i+1} = s'$. Such a choice again satisfies (ii) and (iii) by definition.
- If we have $\bar{x}_{s'} = \bar{x}_s$ for all $s' \in \mathit{supp}(\delta(s_i, a_i))$, then any choice will satisfy (ii). To satisfy the other conditions, we pick s_{i+1} so that $\bar{\beta}_{s_i, a_i, s_{i+1}} = 1$ is true. We argue that such an s_{i+1} can be chosen. We have $\bar{x}_{s_i} > 0$ and so $\bar{\alpha}_s = 1$ by constraint (4.5). We also have $\bar{y}_{s,a} = 1$: for $s \in S_\Diamond$ this follows from the definition of θ, for $s \in S_\Box$ from constraint (4.7). Hence, since constraint (4.6) is satisfied, there must be s_{i+1} such that $\bar{\beta}_{s_i, a, s_{i+1}} = 1$. Then, we apply constraint (4.8) (for $s = s_i$, $t = s_{i+1}$ and

$a = a_i$) and, since the last two summands on the right-hand side are 0, we get $\bar{\gamma}_{s_{i+1}} < \bar{\gamma}_{s_i}$, thus satisfying (iii).

Note that the above construction must terminate after at most $|S|$ steps since, due to conditions (ii) and (iii), no state repeats on ω. Because the only way of terminating is satisfaction of (i), we are done. □

4.2. **Deterministic Multi-Strategies with Dynamic Penalties.** Next, we show how to compute a sound and optimally permissive *deterministic* 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. Otherwise, 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. First, (4.9) and (4.10) 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 probabilities. The behaviour of environment states is then captured by constraint (4.11), where we only maximise the penalty, without incurring any penalty locally.

The constant c in (4.10) is chosen to be no lower than any *finite* penalty achievable by a deterministic multi-strategy, a possible value being:

$$\sum_{i=0}^{\infty}(1 - p^{|S|})^i \cdot p^{|S|} \cdot i \cdot |S| \cdot pen_{\max} \tag{4.12}$$

where p is the smallest non-zero probability assigned by δ, and $pen_{\max}$ is the maximal local penalty over all states. To see that (4.12) indeed gives a safe bound on c (i.e. it is lower than any finite penalty achievable), observe that for the penalty to be finite under a deterministic multi-strategy, for every state s there must be a path of length at most $|S|$ to a state from which no penalty will be incurred. This path has probability at least $p^{|S|}$, and since the penalty accumulated along a path of length $i \cdot |S|$ is at most $i \cdot |S| \cdot pen_{\max}$, the properties of (4.12) follow easily.

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 a finite penalty and the optimal penalty is ∞ (recall that we already established there is *some* sound multi-strategy). In 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.

Correctness. Formally, correctness of the MILP encoding for the case of dynamic penalties is captured by the following theorem.

Theorem 4.3. *Let* $\mathtt{G}$ *be a game,* $\phi = \mathtt{R}^r_{\geqslant b}[\,\mathtt{C}\,]$ *a property and* (ψ, dyn) *a* dynamic *penalty scheme. Assume there is a sound multi-strategy for* ϕ*. The MILP formulation from Fig. 3 satisfies: (a) there is no solution if and only if the optimally permissive deterministic multi-strategy yields infinite penalty; and (b) there is a solution* $\bar{z}_{\bar{s}}$ *if and only if an optimally permissive deterministic multi-strategy yields penalty* $\bar{z}_{\bar{s}}$*.*

Proof. We show that any sound multi-strategy with finite penalty $\bar{z}_{\bar{s}}$ gives rise to a satisfying assignment with the objective value $\bar{z}_{\bar{s}}$, and vice versa. Then, (b) follows directly, and (a) follows by the assumption that there is *some* sound multi-strategy.

Let us prove that for any sound multi-strategy θ we can construct a satisfying assignment to the constraints. For constraints (4.1) to (4.8), the construction works exactly the same as in the proof of Theorem 4.2. For the newly added variables, i.e. z_s and ℓ_s, we put $\bar{\ell}_s = pen_{loc}(\psi, \theta, s)$, ensuring satisfaction of constraint (4.9), and:

$$\bar{z}_s = \sup_{\sigma \lhd \theta, \pi} E^{\sigma,\pi}_{\mathsf{G},s}(\psi^{\theta}_{rew})$$

which, together with [25, Section 7.2.7, Equation 7.2.17] (giving characterisation of optimal reward in terms of a linear program), ensures that constraints (4.10) and 4.11 are satisfied.

In the opposite direction, given a satisfying assignment we construct θ for G' exactly as in the proof of Theorem 4.2. As before, we can argue that constraints (4.1) to (4.8) are satisfied under any sound multi-strategy. We now need to argue that the multi-strategy satisfies $pen_{dyn}(\psi, \theta, s) \geqslant \bar{z}_{\bar{s}}$. It is easy to see that $pen_{loc}(\psi, \theta, s) = \bar{\ell}_s$. Moreover, by [25, Section 7.2.7, Equation 7.2.17] the penalty is the least solution to the inequalities:

$$z'_s \geqslant \max_{a \in \theta(s)} \sum_{s' \in S} \delta(s,a)(s) \cdot z'_{s'} + \bar{\ell}_s \qquad \text{for all } s \in S_\Diamond \tag{4.13}$$

$$z'_s \geqslant \max_{a \in A(s)} \sum_{s' \in S} \delta(s,a)(s) \cdot z'_{s'} \qquad \text{for all } s \in S_\Box \tag{4.14}$$

We can replace (4.13) with:

$$z'_s \geqslant \max_{a \in A(s)} \sum_{s' \in S} \delta(s,a)(s) \cdot z'_{s'} + \bar{\ell}_s - c \cdot (1 - \bar{y}_{s,a}) \tag{4.15}$$

since for $a \in \theta(s)$ we have $c \cdot (1 - \bar{y}_{s,a}) = 0$ and otherwise $c \cdot (1 - \bar{y}_{s,a})$ is greater than $\sum_{s' \in S} \delta(s,a)(s) \cdot z'_{s'} + \bar{\ell}_s$ in the least solution to (4.13) and (4.14), by the definition of c. Finally, it suffices to observe that the set of solutions to (4.14) and (4.15) is the same as the set of solutions to (4.10) and (4.11). □

4.3. **Approximating Randomised Multi-Strategies.** In Section 3, we showed that 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 (intuitively, we would need to multiply expected total rewards by probabilities of actions being allowed under a multi-strategy, and both these quantities are unknowns in our formalisation). 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. Before giving the definition of the transformed game, we show that we can simplify our problem by restricting to multi-strategies which in any state select at most two actions with non-zero probability.

Theorem 4.4. *Let G be a game, $\phi = \mathtt{R}^r_{\geqslant b}[\,\mathtt{C}\,]$ a property, and (ψ, τ) a (static or dynamic) penalty scheme. For any sound multi-strategy θ we can construct another sound multi-strategy θ' such that $pen_\tau(\psi, \theta) \geqslant pen_\tau(\psi, \theta')$ and $|supp(\theta'(s))| \leqslant 2$ for any $s \in S_\Diamond$.*

Proof. If the (dynamic) penalty under θ is infinite, then the solution is straightforward: we can simply take θ' which, in every state, allows a single action so that the reward is maximised. This restrictive multi-strategy enforces a strategy that maximises the reward (so it performs at least as well as any other multi-strategy), and at the same time it cannot yield the dynamic penalty worse than θ, as the dynamic penalty under θ is already infinite. From now on, we will assume that the penalty is finite.

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 $s_1 \in S_\Diamond$. We construct a multi-strategy θ' that in s_1 allows only two of the sets $A_1, \ldots, A_n$ with non-zero probability, and in other states behaves like θ.

We first prove the case of dynamic penalties and then describe the differences for static penalties. Supposing that $\inf_{\sigma \lhd \theta, \pi} E^{\sigma,\pi}_{\mathsf{G},s_1}(r) \leqslant \inf_{\sigma \lhd \theta', \pi} E^{\sigma,\pi}_{\mathsf{G},s_1}(r)$, we have that the total reward is:

$$\begin{aligned}
\inf_{\sigma \lhd \theta, \pi} E^{\sigma,\pi}_{\mathsf{G},\overline{s}}(r) &= \inf_{\sigma \lhd \theta, \pi} \Big(E^{\sigma,\pi}_{\mathsf{G},\overline{s}}(r{\downarrow}s_1) + Pr^{\sigma,\pi}_{\mathsf{G},\overline{s}}(\mathtt{F}\ s_1) \cdot E^{\sigma,\pi}_{\mathsf{G},s_1}(r) \Big) \\
&= \inf_{\sigma \lhd \theta, \pi} \Big(E^{\sigma,\pi}_{\mathsf{G},\overline{s}}(r{\downarrow}s_1) + Pr^{\sigma,\pi}_{\mathsf{G},\overline{s}}(\mathtt{F}\ s_1) \cdot \inf_{\sigma' \lhd \theta, \pi'} E^{\sigma',\pi'}_{\mathsf{G},s_1}(r) \Big) \\
&\leqslant \inf_{\sigma \lhd \theta, \pi} \Big(E^{\sigma,\pi}_{\mathsf{G},\overline{s}}(r{\downarrow}s_1) + Pr^{\sigma,\pi}_{\mathsf{G},\overline{s}}(\mathtt{F}\ s_1) \cdot \inf_{\sigma' \lhd \theta', \pi'} E^{\sigma',\pi'}_{\mathsf{G},s_1}(r) \Big) \\
&= \inf_{\sigma \lhd \theta', \pi} \Big(E^{\sigma,\pi}_{\mathsf{G},\overline{s}}(r{\downarrow}s_1) + Pr^{\sigma,\pi}_{\mathsf{G},\overline{s}}(\mathtt{F}\ s_1) \cdot \inf_{\sigma' \lhd \theta', \pi'} E^{\sigma',\pi'}_{\mathsf{G},s_1}(r) \Big) \qquad (*) \\
&= \inf_{\sigma \lhd \theta', \pi} E^{\sigma,\pi}_{\mathsf{G},\overline{s}}(r)
\end{aligned}$$

where the equation $(*)$ above follows by the fact that, up to the first time s_1 is reached, θ and θ' allow the same actions. Hence, it suffices to define θ' so that $\inf_{\sigma \lhd \theta, \pi} E^{\sigma,\pi}_{\mathsf{G},s_1}(r) \leqslant \inf_{\sigma \lhd \theta', \pi} E^{\sigma,\pi}_{\mathsf{G},s_1}(r)$. Similarly, for the penalties, it is enough to ensure $\sup_{\sigma \lhd \theta, \pi} E^{\sigma,\pi}_{\mathsf{G},s_1}(\psi^{\theta}_{rew}) \geqslant \sup_{\sigma \lhd \theta', \pi} E^{\sigma,\pi}_{\mathsf{G},s_1}(\psi^{\theta'}_{rew})$.

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, i.e.:

$$\begin{aligned}
P_i &= \sum_{a \notin A_i} \psi(s_1, a) + \sup_{\sigma \lhd \theta, \pi} \max_{a \in A_i} \sum_{s' \in S} \delta(s_1, a)(s') \cdot E^{\sigma,\pi}_{\mathsf{G},s'}(\psi^{\theta}_{rew}) \\
R_i &= \inf_{\sigma \lhd \theta, \pi} \min_{a \in A_i} (r(s_1, a) + \sum_{s' \in S} \delta(s_1, a)(s') \cdot E^{\sigma,\pi}_{\mathsf{G},s'}(r))
\end{aligned}$$

We also define $R = \inf_{\sigma \lhd \theta, \pi} E^{\sigma,\pi}_{\mathsf{G},s_1}(r)$ and $P = \sup_{\sigma \lhd \theta, \pi} E^{\sigma,\pi}_{\mathsf{G},s_1}(\psi^{\theta}_{rew})$ and have $R = \sum_{i=1}^{n} \lambda_i R_i$ and $P = \sum_{i=1}^{n} \lambda_i P_i$.

Let $S_0 \subseteq S$ be those states for which there are $\sigma \lhd \theta$ and π ensuring a return to s_1 without accumulating any reward. Formally, S_0 contains all states s_0 which satisfy $Pr^{\sigma,\pi}_{\mathsf{G},s_0}(\mathtt{F}\ \{s_1\}) = 1$ and $E^{\sigma,\pi}_{\mathsf{G},s_0}(r{\downarrow}s_1)$ for some $\sigma \lhd \theta$ and π. We say that A_i is *progressing* if for all $a \in A_i$ we have $r(s_1, a) > 0$ or $supp(\delta(s_1, a)) \not\subseteq S_0$. We note that A_i is progressing whenever $R_i > R$ (since any a violating the condition above could have been used by the opponent to force $R_i \leqslant R$).

For each tuple $\mu = (\mu_1, \ldots, \mu_n) \in \mathbb{R}^n$, let $R^{\mu} = \mu_1 R_1 + \cdots + \mu_n R_n$ and $P^{\mu} = \mu_1 P_1 + \cdots + \mu_n P_n$. Then the set $T = \{(R^{\mu}, P^{\mu}) \mid 0 \leqslant \mu_i \leqslant 1, \mu_1 + \cdots + \mu_n = 1\}$ is a bounded convex polygon, with vertices given by images (R^{e_i}, P^{e_i}) of unit vectors (i.e., Dirac distributions) $e_i = (0, \ldots, 0, 1, 0, \ldots, 0)$, and containing $(R^{\lambda}, P^{\lambda}) = (R, P)$. To each vertex (R_j, P_j) we associate the (non-empty) set $I_j = \{i \mid (R^{e_i}, P^{e_i}) = (R_j, P_j)\}$ of indices.

We will find $\alpha \in (0,1)$ and $1 \leqslant u, v \leqslant n$ such that one of A_u or A_v is progressing, and define the multi-strategy θ' to pick A_u and A_v with probabilities α and $1-\alpha$, respectively. We distinguish several cases, depending on the shape of T:

(1) T has non-empty interior. Let $(R_1, P_1), \dots, (R_m, P_m)$ be its vertices in the anticlockwise order. Since all λ_i are positive, (R, P) is in the interior of T. Now consider the point (R, P') directly below (R, P) on the boundary of T, i.e. $P' = \min\{P'' \mid (R, P'') \in T\}$. If (R, P') is not a vertex, it is a convex combination of adjacent vertices $(R, P') = \alpha(R_j, P_j) + (1-\alpha)(R_{j+1}, P_{j+1})$, and we pick such α and $u \in I_j$ and $v \in I_{j+1}$. If (R, P') happens to be a vertex (R_j, P_j) we can (since $P_j < P$) instead choose sufficiently small $\alpha > 0$ so that $R \geqslant \alpha R_j + (1-\alpha)R_{j+1}$ and $P \leqslant \alpha P_j + (1-\alpha)P_{j+1}$ and again pick $u \in I_j$ and $v \in I_{j+1}$. In either case, we necessarily have $R_{j+1} > R$ (by ordering of the vertices in the anticlockwise order and since (R, P) is in the interior of T), and so A_v is progressing.

(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 progressing A_u. Since all λ_i are positive, (R, P) lies inside the line segment, and in particular $P > P_0$. We can therefore choose some v and $\alpha \in (0,1)$ such that $P \leqslant \alpha \cdot P_u + (1-\alpha) \cdot P_v$.

(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, (R, P) is not one of the extreme points, i.e. $(R, P) = \alpha(R_0, P_0) + (1-\alpha)(R_1, P_1)$ with $0 < \alpha < 1$. We can therefore choose $u \in I_0, v \in I_1$. Again, since $R_1 > R$, A_v is progressing.

(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 combination, or $R > 0$, and there is some progressing A_u, and we then pick arbitrary v and α.

We now want to show that the reward of the updated multi-strategy is indeed no worse than before. For $i \in \{u, v\}$ we define:

$$R'_i = \inf_{\sigma \triangleleft \theta', \pi} \min_{a \in A_i} (r(s_1, a) + \sum_{s' \in S} \delta(s_1, a)(s') \cdot E^{\sigma,\pi}_{\mathsf{G},s'}(r))$$

and we define $R' = \alpha R'_u + (1-\alpha) R'_v$. Pick an action a (resp. a') that realises the minimum and strategies σ and π (resp. σ' and π') that realise the infimum in the definition of R_i (resp. R'_i). (Such strategies indeed exist). Define:

$$c_i = \sum_{s'} \delta(s_1, a)(s') \cdot Pr^{\sigma,\pi}_{\mathsf{G},s'}(\mathtt{F}\ s_1) \qquad c'_i = \sum_{s'} \delta(s_1, a')(s') \cdot Pr^{\sigma,\pi}_{\mathsf{G},s'}(\mathtt{F}\ s_1)$$

$$d_i = r(s, a) + \sum_{s'} \delta(s_1, a)(s') \cdot E^{\sigma,\pi}_{\mathsf{G},s'}(r{\downarrow}s_1) \quad d'_i = r(s, a') + \sum_{s'} \delta(s_1, a')(s') \cdot E^{\sigma,\pi}_{\mathsf{G},s'}(r{\downarrow}s_1)$$

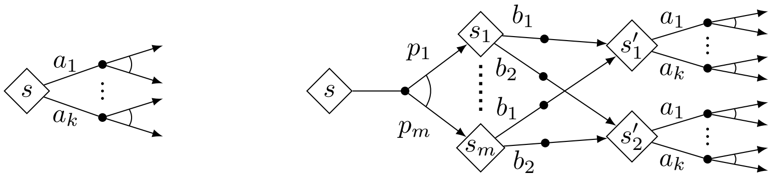


Figure 4: A node in the original game G (left), and the corresponding nodes in the transformed game G' for approximating randomised multi-strategies (right, see Section 4.3).

We have $R_i = c_i \cdot R + d_i$ for every $1 \leqslant i \leqslant n$, and $R'_i = c'_i \cdot R' + d'_i$ for all $i \in \{u, v\}$. Then:

$$\begin{aligned}
R' - R &= (\alpha R'_u + (1-\alpha)R'_v) - \textstyle\sum_i \lambda_i R_i \\
&= (\alpha R'_u + (1-\alpha)R'_v) - (\alpha R_u + (1-\alpha)R_v) + (\alpha R_u + (1-\alpha)R_v) - \textstyle\sum_i \lambda_i R_i \\
&\geqslant (\alpha R'_u + (1-\alpha)R'_v) - (\alpha R_u + (1-\alpha)R_v) \\
&\qquad \text{(by the choice of } \alpha, u, v) \\
&= (\alpha(c'_u R' + d'_u) + (1-\alpha)(c'_v R' + d'_v)) - (\alpha(c_u R + d_u) + (1-\alpha)(c_v R + d_v)) \\
&\geqslant (\alpha(c'_u R' + d'_u) + (1-\alpha)(c'_v R' + d'_v)) - (\alpha(c'_u R + d'_u) + (1-\alpha)(c'_v R + d'_v)) \\
&\qquad (c_i R + d_i \leqslant c'_i R + d'_i \text{ by optimality of actions/strategies defining } c_i \text{ and } d_i) \\
&= (\alpha c'_u + (1-\alpha)c'_v)(R' - R),
\end{aligned}$$

i.e., $(1 - \alpha c'_u - (1-\alpha)c'_v)(R' - R) \geqslant 0$. 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 $\alpha c'_u + (1-\alpha)c'_v$, therefore $R' \geqslant R$.

We can show that the penalty under θ' is at most as big as the penalty under θ in exactly the same way (note that in addition using $\psi^{\theta'}_{rew}$ instead of ψ^{θ}_{rew} for c', d', R' and R'_i). For static penalties, the proof that the new multi-strategy is no worse than the old one is straightforward from the choice of $\theta'(s_1)$. □

The result just proved allows us to simplify the game construction that we use to map between (discretised) randomised multi-strategies and deterministic ones. Let the original game be G and the transformed game be G'. The transformation is illustrated in Fig. 4. The left-hand side shows a controller state $s \in S_\Diamond$ in the original game G (i.e., the one for which we are seeking randomised multi-strategies). For each such state, we add the two layers of states illustrated on the right-hand side of the figure: *gadgets* s'_1, s'_2 representing the two 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 two gadgets. Two new actions, b_1 and b_2, are also added to label the transitions between selectors s_i and gadgets s'_1, s'_2.

The selectors s_i are reached from s via a transition using fixed probabilities $p_1, \ldots, p_m$ which need to be chosen appropriately. For efficiency, we want to minimise the number of selectors m for each state s. 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 $2 \leqslant i \leqslant m$. For example, for M=10, we have $m = 4$ and $l_1, \ldots, l_4 = 5, 3, 1, 1$, so $p_1, \ldots, p_4 = \frac{5}{10}, \frac{3}{10}, \frac{1}{10}, \frac{1}{10}$.

We are now able to find optimal discretised randomised multi-strategies in G by finding optimal deterministic multi-strategies in G'. This connection will be formalised in Lemma 4.5 below. But we first point out that, for the case of *static* penalties, a small transformation to the MILP encoding (see Fig. 2) used to solve game G' is required. The problem is that the definition of static penalties on G' does not precisely capture the static penalties of the original game G. In this, case we adapt Fig. 2 as follows. For each state s, action $a \in A(s)$ and $i \in \{1,\dots,n\}$, we add a binary variable $y'_{s_i,a}$ and constraints $y'_{s_i,a} \geqslant y_{s'_j,a} - (1 - y_{s_i,b_j})$ for $j \in \{1,2\}$. We then change the objective function that we minimise to:

$$-x_{\overline{s}} + \sum\nolimits_{s\in S_\Diamond} \sum\nolimits_{a\in A(s)} \sum\nolimits_{i=1}^{m} p_i \cdot (1 - y'_{s_i,a}) \cdot \psi(s,a)$$

Lemma 4.5. *Let G be a game, $\phi = \mathtt{R}^r_{\geqslant b}[\,\mathtt{C}\,]$ a property, (ψ,τ) a (static or dynamic) penalty scheme, and let G' be the game transformed as described above. The following two claims are equivalent:*

1. *There is a sound multi-strategy θ in G with $pen_{dyn}(\psi,\theta) = x$ (or, for static penalties, $pen_{sta}(\psi,\theta) = x$), and θ only uses probabilities that are multiples of $\frac{1}{M}$.*
2. *There is a sound deterministic multi-strategy θ' in G' and $pen_{dyn}(\psi,\theta) = x$ (or, for static penalties, $\sum_{s\in S_\Diamond}\sum_{i=1}^{m} p_i \cdot \psi'(i) = x$ where $\psi'(i)$ is $\sum_{a\in A(s)\setminus(\theta'(s'_1)\cup\theta'(s'_2))}\psi(s,a)$ if $\theta'(s_i) = \{b_1,b_2\}$, and otherwise $\psi'(i)$ is $\sum_{a\in A(s)\setminus\theta'(s'_j)}\psi(s,a)$ where j is the (unique) number with $b_j \in \theta'(s_i)$).*

Proof. Firstly, observe that for any integer $0 \leqslant k \leqslant M$ there is a set $I_k \subseteq \{1,\dots,m\}$ with $\sum_{j\in I_k} l_j = k$. The opposite direction also holds.

Let θ be a multi-strategy in G. By Theorem 4.4 we can assume that $|supp(\theta(s))| \leqslant 2$ for any s. We create θ' as follows. For every state $s \in S_\Diamond$ with $\{A_1, A_2\} = supp(\theta(s))$, we set $\theta'(s'_1)(A_1) = 1$ and $\theta'(s'_2)(A_2) = 1$. Then, supposing $\theta(s)(A) = \frac{k}{M}$, we let $\theta'(s_i)(\{b_1\}) = 1$ whenever $i \in I_k$, and $\theta'(s_i)(\{b_2\}) = 1$ whenever $i \notin I_k$. If $\theta(s)$ is a singleton set, the construction is analogous. Clearly, the property for static penalties is preserved. For any memoryless $\sigma' \lhd \theta'$ there is a memoryless strategy $\sigma \lhd \theta$ that is given by $\sigma(s)(a) = \frac{k}{M} \cdot \sigma(s'_1)(a) + (1 - \frac{k}{M}) \cdot \sigma(s'_1)(a)$ for any a, and conversely for any $\sigma \lhd \theta$ we can define $\sigma' \lhd \theta'$ by putting $\sigma'(s'_1) = d_{s,A_1}$ and $\sigma'(s'_2) = d_{s,A_2}$ for all s, where d_{s,A_1} and d_{s,A_2} are distributions witnessing that σ is compliant with θ. It is easy to see that both σ and σ' in either of the above yield the same reward and dynamic penalty.

In the other direction, we define θ from θ' for all $s \in S_\Diamond$ as follows. Let A_1 and A_2 be the sets allowed by θ' in s'_1 and s'_2 respectively. If $A_1 = A_2$, then $\theta(s)$ allows this set with probability 1. Otherwise $\theta(s)$ allows the set $A_1 \cup A_2$ with probability $\sum_{i:\theta(s_i)=\{b_1,b_2\}} p_i$, the set A_1 with probability $\sum_{i:\theta(s_i)=\{b_1\}} p_i$ and the set A_2 with probability $\sum_{i:\theta(s_i)=\{b_2\}} p_i$. The correctness can be proved similarly to above. □

Our next goal is to show 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, define a suitable choice of M. This will be formalised in Theorem 4.7 shortly. First, we need to establish the following intermediate result, stating that, in the static case, in addition to Theorem 4.4 we can require the action subsets allowed by a multi-strategy to be ordered with respect to the subset relation.

Theorem 4.6. *Let G be a game, $\phi = \mathtt{R}^r_{\geqslant b}[\,\mathtt{C}\,]$ a property and (ψ, sta) a* static *penalty scheme. For any sound multi-strategy θ we can construct another sound multi-strategy θ'*

such that $pen_{sta}(\psi,\theta) \geqslant pen_{sta}(\psi,\theta')$ *and for each* $s \in S_\Diamond$*, if* $supp(\theta'(s))=\{B,C\}$*, then either* $B \subseteq C$ *or* $C \subseteq B$*.*

Proof. Let θ be a multi-strategy and fix s_1 such that θ takes two different actions B and C with probability $p \in (0,1)$ and $1-p$ where $B \not\subseteq C$ and $C \not\subseteq B$. If $\inf_{\sigma\lhd\theta,\pi} E^{\sigma,\pi}_{\mathsf{G},s_1}(r) = 0$, then we can in fact allow deterministically the single set $A(s_1)$ and we are done. Hence, suppose that the reward accumulated from s_1 is non-zero.

Suppose, w.l.o.g., that:

$$\min_{a\in B} r(s_1,a) + \sum_{s'\in S} \delta(s_1,a)(s') \cdot \inf_{\sigma\lhd\theta,\pi} E^{\sigma,\pi}_{\mathsf{G},s'}(r) \leqslant \min_{a\in C} r(s_1,a) + \sum_{s'\in S} \delta(s_1,a)(s') \cdot \inf_{\sigma\lhd\theta,\pi} E^{\sigma,\pi}_{\mathsf{G},s'}(r) \tag{4.16}$$

It must be the case that, for some $D \in \{B,C\}$, we have:

$$\min_{a\in D} r(s_1,a) + \inf_{\sigma\lhd\theta,\pi} \sum_{s'\in S} \delta(s_1,a)(s') \cdot E^{\sigma,\pi}_{\mathsf{G},s'}(r{\downarrow}s_1) > 0 \tag{4.17}$$

(otherwise the minimal reward accumulated from s_1 is 0 since there is a compliant strategy that keeps returning to s_1 without ever accumulating any reward), and if the inequality in (4.16) is strict, then (4.17) holds for $D = C$. W.l.o.g., suppose that the above property holds for C. We define θ' by modifying θ and picking $B \cup C$ with probability p, C with $(1-p)$, and B with probability 0.

Under θ, the minimal reward achievable by some compliant strategy is given as the least solution to the following equations [25, Theorem 7.3.3] (as before, the notation of [25] requires "negative" models):

$$x_s = \sum_{A\in supp(\theta(s))} \theta(s)(A) \cdot \min_{a\in A} \sum_{s'\in S} r(s,a) + \delta(s,a)(s') \cdot x_{s'} \qquad \text{for } s \in S_\Diamond$$

$$x_s = \min_{a\in A(s)} \sum_{s'\in S} r(s,a) + \delta(s,a)(s') \cdot x_{s'} \qquad \text{for } s \in S_\Box$$

The minimal rewards x'_s achievable under θ' are defined analogously. In particular, for the equation with s_1 on the left-hand side we have:

$$x_{s_1} = p \cdot \min_{a\in B} r(s_1,a) + \sum_{s'\in S} \delta(s_1,a)(s') \cdot x_{s'} + (1-p) \cdot \min_{a\in C} r(s_1,a) + \sum_{s'\in S} \delta(s_1,a)(s') \cdot x_{s'}$$

$$x'_{s_1} = p \cdot \min_{a\in B\cup C} r(s_1,a) + \sum_{s'\in S} \delta(s_1,a)(s') \cdot x'_{s'} + (1-p) \cdot \min_{a\in C} r(s_1,a) + \sum_{s'\in S} \delta(s_1,a)(s') \cdot x'_{s'}$$

We show that the least solution $\bar{x}$ to x is also the least solution to x'.

First, note that $\bar{x}$ is clearly a solution to any equation with $s \neq s_1$ on the left-hand side since these equations remain unchanged in both sets of equations. As for the equation with s_1, we have $\min_{a\in B} \sum_{s'} r(s_1,a)+\delta(s_1,a)(s')\cdot\bar{x}_{s'} \leqslant \min_{a\in C} \sum_{s'} r(s_1,a)+\delta(s_1,a)(s')\cdot\bar{x}_{s'}$, and so necessarily $\min_{a\in B} \sum_{s'} r(s_1,a)+\delta(s_1,a)(s')\cdot\bar{x}_{s'} = \min_{a\in B\cup C} \sum_{s'} r(s_1,a)+\delta(s_1,a)(s')\cdot\bar{x}_{s'}$.

To see that $\bar{x}$ is the *least* solution to x', we show that (i) for all s, if $\inf_{\sigma\lhd\theta',\pi} E^{\sigma,\pi}_{\mathsf{G},s}(r) = 0$ then $\bar{x}_s = 0$; and (ii) there is a unique fixpoint satisfying $\bar{x}_s = 0$ for all s such that $\inf_{\sigma\lhd\theta',\pi} E^{\sigma,\pi}_{\mathsf{G},s}(r) = 0$.

For (i), suppose $\bar{x}_s > 0$. Let σ' be a strategy compliant with θ', and π an arbitrary strategy. Suppose $Pr^{\sigma',\pi}_{\mathsf{G},s}(\mathtt{F}\ s_1) = 0$, then there is a strategy σ compliant with θ which behaves exactly as σ' when starting from s, and by our assumption on the properties of $\bar{x}_s$

we get that $E^{\sigma,\pi}_{\mathsf{G},s}(r) > 0$ and so $E^{\sigma',\pi}_{\mathsf{G},s}(r) > 0$. Now suppose that $Pr^{\sigma',\pi}_{\mathsf{G},s}(\mathtt{F}\ s_1) > 0$. For this case, by condition (4.17), the fact that it holds for $D = C$ and by defining θ' so that it picks C with nonzero probability we get that the reward under any strategy compliant with θ' is non-zero when starting in s_1, and so $E^{\sigma',\pi}_{\mathsf{G},s}(r) > 0$.

Point (ii) can be obtained by an application of [25, Proposition 7.3.4]. □

We can now return to the issue of how to vary the granularity M to get sufficiently close to the optimal penalty. We formalise this as follows.

Theorem 4.7. *Let* G *be a game,* $\phi = \mathtt{R}^r_{\geqslant b}[\,\mathtt{C}\,]$ *a property, and* (ψ, τ) *a (static or dynamic) penalty scheme. 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* $pen_\tau(\psi, \theta') - pen_\tau(\psi, \theta) \leqslant \varepsilon$*. 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. By Theorem 4.4, we can assume $supp(\theta(t)) = \{A_1, A_2\}$ for any state $t \in S_\Diamond$.

We deal with the cases of static and dynamic penalties separately. For static penalties, let $t \in S_\Diamond$ and $\theta(t)(A_1) = q$, $\theta(t)(A_2) = 1 - q$ for $A_1 \subseteq A_2 \subseteq A(t)$. Modify θ by rounding q up to the number q' which is the nearest multiple of $\frac{1}{M}$. The resulting multi-strategy θ' is again sound, since any strategy compliant with θ' is also compliant with θ: the witnessing distributions (see Definition 3.2) d_{t,A_1} and d_{t,A_2} for θ are obtained from the distributions d'_{t,A_1} and d'_{t,A_2} for θ' by setting $d_{t,A_1}(a) = d'_{t,A_1}(a)$ for all $a \in A_1$ and $d_{t,A_2}(a) = \frac{q'-q}{1-q} \cdot d'_{t,A_1}(a) + \frac{1-q'}{1-q} \cdot d'_{t,A_2}(a)$ for all $a \in A_2$; note that both d_{t,A_1} and d_{t,A_2} are indeed probability distributions. Further, the penalty in θ' changes by at most $\frac{1}{M} \sum_{a \in A(s)} \psi(t, a)$. To obtain the result we repeat the above for all t.

Now let us consider 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 $t \in S_\Diamond$ a state. W.l.o.g., suppose:

$$\inf_{\sigma \lhd \theta, \pi} \min_{a \in A_1} r(s,a) + \sum_{s'} \delta(s,a)(s') \cdot E^{\sigma,\pi}_{\mathsf{G},s'}(r) \geqslant \inf_{\sigma \lhd \theta, \pi} \min_{a \in A_2} r(s,a) + \sum_{s'} \delta(s,a)(s') \cdot E^{\sigma,\pi}_{\mathsf{G},s'}(r)$$

For $0 < x < \theta(t)(A_2)$, we define a multi-strategy θ_x by $\theta_x(t)(A_1) = \theta(t)(A_1) + x$ and $\theta_x(t)(A_2) = \theta(t)(A_2) - x$, and $\theta_x(s) = \theta(s)$ for all $s \neq t$. We will show that $\inf_{\sigma \lhd \theta_x, \pi} E^{\sigma,\pi}_{\mathsf{G},\overline{s}}(r) \geqslant \inf_{\sigma \lhd \theta, \pi} E^{\sigma,\pi}_{\mathsf{G},\overline{s}}(r)$. Consider the following functional $F_x : (S \to \mathbb{R}) \to (S \to \mathbb{R})$, constructed for the multi-strategy θ_x

$$F_x(f)(s) = \sum_{A \in supp(\theta(s))} \theta_x(s) \cdot \min_{a \in A} \sum_{s' \in S} r(s,a) + \delta(s,a)(s') \cdot f(s') \qquad \text{for } s \in S_\Diamond$$

$$F_x(f)(s) = \min_{a \in A(s)} \sum_{s' \in S} r(s,a) + \delta(s,a)(s') \cdot f(s') \qquad \text{for } s \in S_\Box$$

Let f be the function assigning $\inf_{\sigma \lhd \theta, \pi} E^{\sigma,\pi}_{\mathsf{G},t}(r)$ to s. Observe that $f(s) = 0$ whenever $\inf_{\sigma \lhd \theta_x, \pi} E^{\sigma,\pi}_{\mathsf{G},t}(r)$; this follows since $x < \min\{\theta(t)(A_1), \theta(t)(A_2)\}$ and so both θ and θ_x allow the same actions with non-zero probability. Also, $F_x(f)(s) \geqslant f(s)$: for $s \neq t$ in fact $F_x(f)(s) = f(s)$ because the corresponding functional F for θ coincides with F_x on s; for $s = t$, we have $F_x(f)(s) \geqslant f(s)$ since $\min_{a \in A_1} r(s,a) + \sum_{s'} \delta(s,a)(s') \cdot f(s') \geqslant \min_{a \in A_2} r(s,a) + \sum_{s'} \delta(s,a)(s') \cdot f(s')$ by the properties of A_1 and A_2 and since x is non-negative. Hence, we can apply [25, Proposition 7.3.4] and obtain that θ_x ensures at least the same reward as θ.

Thus, by increasing the probability of allowing A_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, i.e. $E^{\sigma',\pi}_{\mathsf{G},t}(\psi^{\theta_x}_{rew})$, is equal to:

$$pen_{loc}(\psi,\theta_x,t) + \sum_{a\in A}\xi'(t)(a)\sum_{t'\in S}\delta(t,a)(t')\cdot\big(E^{\sigma',\pi}_{\mathsf{G},t'}(\psi^{\theta_x}_{rew}{\downarrow}t) + Pr^{\sigma',\pi}_{\mathsf{G},t'}(\mathtt{F}\ t)\cdot E^{\sigma',\pi}_{\mathsf{G},t}(\psi^{\theta_x}_{rew})\big)$$

for $\xi' = \sigma' \cup \pi$. 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:

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

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

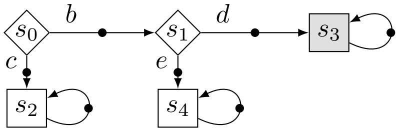
Thus:

$$\begin{aligned}
E^{\sigma',\pi}_{\mathsf{G},t}(\psi^{\theta_x}_{rew}) &= \frac{pen_{loc}(\psi,\theta_x,t) + \sum_{a\in A}\xi'(t)(a)\sum_{t'\in S}\delta(t,a)(t')\cdot E^{\sigma',\pi}_{\mathsf{G},t'}(\psi^{\theta_x}_{rew}{\downarrow}t)}{1 - \sum_{a\in A}\xi'(t)(a)\sum_{t'\in S}\delta(t,a)(t')\cdot Pr^{\sigma',\pi}_{\mathsf{G},t'}(\mathtt{F}\ t)}\\
E^{\sigma,\pi}_{\mathsf{G},t}(\psi^{\theta}_{rew}) &\geqslant \frac{pen_{loc}(\psi,\theta,t) - x + \sum_{a\in A}(\xi'(t)(a) - x)\sum_{t'\in S}\delta(t,a)(t')\cdot E^{\sigma',\pi}_{\mathsf{G},t'}(\psi^{\theta}_{rew}{\downarrow}t)}{1 - \sum_{a\in A}(\xi'(t)(a) - x)\sum_{t'\in S}\delta(t,a)(t')\cdot Pr^{\sigma',\pi}_{\mathsf{G},t'}(\mathtt{F}\ t)}
\end{aligned}$$

and so $E^{\sigma',\pi}_{\mathsf{G},t}(\psi^{\theta_x}_{rew}) - E^{\sigma,\pi}_{\mathsf{G},t}(\psi^{\theta}_{rew})$ 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 an x such that we can modify the decision of θ in s by x, not violate the soundness property and increase the 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. □

For the sake of completeness, we also show that Theorem 4.6 does *not* extend to 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. 5, 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

Figure 5: Counterexample for Theorem 4.6 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 4.6, 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.

4.4. **Optimisations.** We conclude this section on MILP-based multi-strategy synthesis by presenting some optimisations that can be applied to our methods. The general idea is to add additional constraints to the MILP problems that will reduce the search space to be explored by a solver. We present two different optimisations, targeting different aspects of our encodings: (i) possible variable values; and (ii) penalty structures.

Bounds on variable values. In our encodings, for the variables x_s, we only specified very general lower and upper bounds that would constrain their value. Narrowing down the set of values that a variable may take can significantly reduce the search space and thus the solution time required by an MILP solver. One possibility that works well in practice is to bound the values of the variables by the minimal and maximal expected reward achievable from the given state, i.e., add the constraints:

$$\inf_{\sigma,\pi} E^{\sigma,\pi}_{\mathsf{G},s}(r) \leqslant x_s \leqslant \sup_{\sigma,\pi} E^{\sigma,\pi}_{\mathsf{G},s}(r) \qquad\qquad \text{for all } s \in S$$

where both the infima and suprema above are constants obtained by applying standard probabilistic model checking algorithms.

Actions with zero penalties. Our second optimisation exploits the case where an action has zero penalty assigned to it. Intuitively, this action could always be disabled without harming the overall penalty of the multi-strategy. On the other hand, enabling an action with zero penalty might be the only way to satisfy the property and therefore we cannot disable all such actions. However, it is enough to allow at most one action that has zero penalty. For simplicity of the presentation, we assume $Z_s = \{a \in A(s) \,|\, \psi(s,a) = 0\}$; then formally we add the constraints $\sum_{a \in Z_s} y_{s,a} \leqslant 1$ for all $s \in S_\Diamond$.

5. Experimental Results

We have implemented our techniques within PRISM-games [9], an extension of the PRISM model checker [21] 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 [32] or Gurobi [33] to solve MILP problems. 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 [34]. Our experiments were run on a PC with a 2.8GHz Xeon processor and 32GB of RAM, running Fedora 14.

5.1. **Deterministic Multi-strategy Synthesis.** We first discuss the generation of optimal *deterministic* multi-strategies, the results of which are presented in Tab.s 1 and 2. Tab. 1 summarises the models and properties considered. For each model, we give: the case study name, any parameters used, the number of states ($|S|$) and of controller states ($|S_\Diamond|$), and the property used. The final column gives, for comparison purposes, the time required for performing classical (single) strategy synthesis on each model and property ϕ.

Name [parameters]	Parameter values	States	Controller states	Property	Strat. synth. time (s)
cloud	5	8,841	2,177	$\mathtt{P}_{\geqslant 0.9999}[\,\mathtt{F}\ \mathit{deployed}\,]$	0.04
[vm]	6	34,953	8,705	$\mathtt{P}_{\geqslant 0.999}[\,\mathtt{F}\ \mathit{deployed}\,]$	0.10
android [r, s]	1, 48	2,305	997	$\mathtt{R}^{\mathit{time}}_{\leqslant 10000}[\,\mathtt{C}\,]$	0.16
	2, 48	9,100	3,718		0.57
	3, 48	23,137	9,025		0.93
mdsm	3	62,245	9,173	$\mathtt{P}_{\leqslant 0.1}[\,\mathtt{F}\ \mathit{deviated}\,]$	5.64
[N]	3	62,245	9,173	$\mathtt{P}_{\leqslant 0.01}[\,\mathtt{F}\ \mathit{deviated}\,]$	
investor	5,10	10,868	3,344	$\mathtt{R}^{\mathit{profit}}_{\geqslant 4.98}[\,\mathtt{C}\,]$	0.87
[$vinit, vmax$]	10, 15	21,593	6,644	$\mathtt{R}^{\mathit{profit}}_{\geqslant 8.99}[\,\mathtt{C}\,]$	2.20
team-form	3	12,476	2,023	$\mathtt{P}_{\geqslant 0.9999}[\,\mathtt{F}\ \mathit{done}_1\,]$	0.20
[N]	4	96,666	13,793		0.83
cdmsn [N]	3	1240	604	$\mathtt{P}_{\geqslant 0.9999}[\,\mathtt{F}\ \mathit{prefer}_1]$	0.05

Table 1: Details of the models and properties used for experiments with deterministic multi-strategies, and execution times for *single* strategy synthesis.

Name [parameters]	Parameter values	Penalty	Multi-strategy synthesis time (s)				
			CPLEX				Gurobi
			No-opts	Bounds	Zero	Both	Both
cloud	5	0.001	2.5	3.35	13.04	10.36	1.45
[vm]	6	0.01	62.45	*	63.59	25.37	4.73
android [r, s]	1, 48	0.0009	1.07	0.66	1.04	0.48	0.53
	2, 48	0.0011	28.56	8.41	28.48	8.42	3.6
	3, 48	0.0013	*	13.41	*	13.30	47.62
mdsm	3	52	28.06	36.28	27.88	33.72	19.40
[N]	3	186	11.89	11.57	11.88	11.56	12.27
investor	5,10	1	68.64	131.38	68.90	131.36	12.02
[$vinit, vmax$]	10, 15	1	*	*	*	*	208.95
team-form	3	0.890	0.15	0.26	0.15	0.26	0.80
[N]	4	0.890	249.36	249.49	186.41	184.50	3.84
cdmsn [N]	3	2	0.57	0.62	0.62	0.61	1.65

* No optimal solution to MILP problem within 5 minute time-out.

Table 2: Experimental results for synthesising optimal deterministic multi-strategies.

In Tab. 2, we show, for each different model, the penalty value of the optimal multi-strategy and the time to generate it. We report several different times, each for different combinations of the optimisations described in Section 4.4 (either no optimisations, one or both). For the last result, we give times for both MILP solvers: CPLEX and Gurobi.

Case studies. Now, we move on to give further details for each case study, illustrating the variety of ways that permissive controller synthesis can be used. Subsequently, we will discuss the performance and scalability of our approach.

cloud: We adapt a PRISM model from [6] to synthesise deployment 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 Tab. 2) 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. 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 [23] synthesises strategies for a futures market investor, who chooses when to reserve shares, operating in a (malicious) market which can periodically ban him/her 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/her 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.

Performance and scalability. Unsurprisingly, permissive controller synthesis is more costly to execute than classical controller synthesis – this is clearly seen by comparing the times in the rightmost column of Tab. 1 with the times in Tab. 2. However, we successfully synthesised 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. In particular, it is also affected

Name [parameters]	Parameters values	States	Controller states	Property
android $[r, s]$	1,1 1,10	49 481	10 112	$\mathtt{P}_{\geqslant 0.9999}[\,\mathtt{F}\ done]$ $\mathtt{P}_{\geqslant 0.999}[\,\mathtt{F}\ done]$
cloud $[vm]$	5	8,841	2,177	$\mathtt{P}_{\geqslant 0.9999}[\,\mathtt{F}\ deployed]$
investor $[vinit, vmax]$	5,10	10,868	3,344	$\mathtt{R}^{profit}_{\geqslant 4.98}[\,\mathtt{C}\,]$
team-form $[N]$	3	12,476	2,023	$\mathtt{P}_{\geqslant 0.9999}[\,\mathtt{F}\ done_1\,]$
cdmsn $[N]$	3	1240	604	$\mathtt{P}_{\geqslant 0.9999}[\,\mathtt{F}\ prefer_1]$

Table 3: Details of models and properties for approximating optimal randomised multi-strategies.

by the number of actions in controller states, since these result in integer MILP variables, which are the most expensive part of the solution.

The performance optimisations presented in Section 4.4 often allowed us to obtain an optimal multi-strategy quicker. In many cases, it proved beneficial to apply both optimisations at the same time. In the best case (*android*, r=3, s=48), an order of magnitude gain was observed. We reported a slowdown after applying optimisation in the case of *investor*. We attribute this to the fact that adding bounds on variable value can make finding the initial solution of the MILP problem harder, causing a slowdown of the overall solution process.

Both solvers were run using their off-the-shelf configuration and Gurobi proved to be a more efficient solver. In the case of CPLEX, we also observed worse numerical stability, causing it to return a sub-optimal multi-strategy as optimal. In the case of Gurobi, we did not see any such behaviour.

5.2. **Randomised multi-strategy synthesis.** Next, we report the results for approximating optimal *randomised* multi-strategies. Tab. 3 summarises the models and properties used. In Tab. 4, we report the effects on state space size caused by adding the approximation gadgets to the games. We picked three different granularities $M = 100, M = 200$ and $M = 300$; for higher values of M we did not observe improvements in the penalties of the generated multi-strategies. Finally, in Tab. 5, we show the penalties obtained by the randomised multi-strategies that were generated. We compare the (static) penalty value of the randomised multi-strategies to the value obtained by optimal 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 used Gurobi as the MILP solver in every case.

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 Tab. 5). As would be expected, often, larger values of M give smaller penalties. In some cases, this is not true, which we attribute to the size of the MILP problem (which grows with M). For all examples, we built randomised multi-strategies with smaller or equal penalties than the deterministic ones.

Name [parameters]	Parameters values	States	Controller states	States		
				M=100	M=200	M=300
android $[r, s]$	1,1 1,10	49 481	10 112	90 1629	94 1741	98 1853
cloud $[vm]$	5	8,841	2,177	29447	32686	35233
investor $[vinit, vmax]$	5,10	10,868	3,344	33440	35948	38456
team-form $[N]$	3	12,476	2,023	31928	33716	35504
cdmsn $[N]$	3	1240	604	3625	3890	4155

Table 4: State space growth for approximating optimal randomised multi-strategies.

Name [parameters]	Parameters values	States	Controller states	Pen. (det.)	Penalty (randomised)		
					M=100	M=200	M=300
android $[r, s]$	1,1 1,10	49 481	10 112	1.01 19.13	0.91 12.27*	0.905 9.12*	0.903 17.18*
cloud $[vm]$	5	8,841	2,177	1	0.91*	0.905*	0.91*
investor $[vinit, vmax]$	5,10	10,868	3,344	1	1*	1*	1*
team-form $[N]$	3	12,476	2,023	264	263.96*	263.95*	263.95*
cdmsn $[N]$	3	1240	604	2	0.38*	1.9*	0.5*

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

Table 5: Experimental results for approximating optimal randomised multi-strategies.

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.

In this paper, we have imposed several restrictions on permissive controller synthesis. Firstly, we focused on properties expressed in terms of expected total reward (which also subsumes the case of probabilistic reachability). A possible topic for future work would be to consider more expressive temporal logics or parity objectives. The results might also be generalised so that both positive and negative rewards can be used, for example by using the techniques of [30]. We also restricted our attention to memoryless multi-strategies, rather than the more general class of history-dependent multi-strategies. Extending our theory to the latter case and exploring the additional power brought by history-dependent multi-strategies is another interesting direction of future work.

Acknowledgements. The authors are part supported by ERC Advanced Grant VERIWARE and EPSRC projects EP/K038575/1, EP/F001096/1 and EP/M023656/1. Vojtěch Forejt is also affiliated with Faculty of Informatics, Masaryk University, Czech Republic.

References

[1] G. Behrmann, A. Cougnard, A. David, E. Fleury, K. Larsen, and D. Lime. UPPAAL-Tiga: Time for playing games! In *Proc. 19th International Conference on Computer Aided Verification (CAV'07)*, volume 4590 of *LNCS*, pages 121–125. Springer, 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*, pages 135–149, 2011.

[5] R. Calinescu, C. Ghezzi, M. Kwiatkowska, and R. Mirandola. Self-adaptive software needs quantitative verification at runtime. *Communications of the ACM*, 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 *Large-Scale Complex IT Systems – Development, Operation and Management*, pages 303–329, 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 C. Flanagan and B. König, editors, *Proc. 18th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'12)*, volume 7214 of *LNCS*, pages 315–330. Springer, 2012.

[9] T. Chen, V. Forejt, M. Kwiatkowska, D. Parker, and A. Simaitis. PRISM-games: A model checker for stochastic multi-player games. In N. Piterman and S. Smolka, editors, *Proc. 19th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'13)*, volume 7795 of *LNCS*, pages 185–191. Springer, 2013.

[10] T. Chen, M. Kwiatkowska, D. Parker, and A. Simaitis. Verifying team formation protocols with probabilistic model checking. In *Proc. 12th International Workshop on Computational Logic in Multi-Agent Systems (CLIMA XII 2011)*, volume 6814 of *LNCS*, pages 190–297. Springer, 2011.

[11] T. Chen, M. Kwiatkowska, A. Simaitis, and C. Wiltsche. Synthesis for multi-objective stochastic games: An application to autonomous urban driving. In *Proc. 10th International Conference on Quantitative Evaluation of SysTems (QEST'13)*, volume 8054 of *LNCS*, pages 322–337. Springer, 2013.

[12] A. Condon. On algorithms for simple stochastic games. *Advances in computational complexity theory, DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 13:51–73, 1993.

[13] K. Draeger, V. Forejt, M. Kwiatkowska, D. Parker, and M. Ujma. Permissive controller synthesis for probabilistic systems. In *Proc. 20th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'14)*, volume 8413 of *LNCS*, pages 531–546. Springer, 2014.

[14] K. Etessami, M. Kwiatkowska, M. Vardi, and M. Yannakakis. Multi-objective model checking of Markov decision processes. *Logical Methods in Computer Science*, 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 P. Abdulla and K. Leino, editors, *Proc. 17th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'11)*, volume 6605 of *LNCS*, pages 112–127. Springer, 2011.

[17] M. R. Garey, R. L. Graham, and D. S. Johnson. Some NP-complete geometric problems. In *Proceedings of the Eighth Annual ACM Symposium on Theory of Computing*, 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-Verlag, 2nd edition, 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. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: Verification of probabilistic real-time systems. In G. Gopalakrishnan and S. Qadeer, editors, *Proc. 23rd International Conference on Computer Aided Verification (CAV'11)*, volume 6806 of *LNCS*, pages 585–591. Springer, 2011.

[22] M. Lahijanian, J. Wasniewski, S. B. Andersson, and C. Belta. Motion planning and control from temporal logic specifications with probabilistic satisfaction guarantees. In *Proc. 2010 IEEE International Conference on Robotics and Automation*, pages 3227–3232, 2010.

[23] A. McIver and C. Morgan. Results on the quantitative mu-calculus qMu. *ACM Transactions on Computational Logic*, 8(1), 2007.

[24] N. Ozay, U. Topcu, R. Murray, and T. Wongpiromsarn. Distributed synthesis of control protocols for smart camera networks. In *Proc. IEEE/ACM International Conference on Cyber-Physical Systems (ICCPS'11)*, 2011.

[25] M. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley and Sons, 1994.

[26] F. Radmacher. *Games on Dynamic Networks: Routing and Connectivity*. PhD thesis, RWTH Aachen University, 2012.

[27] A. Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, New York, NY, USA, 1998.

[28] N. Shankar. A tool bus for anytime verification. In *Usable Verification*, 2010.

[29] G. Steel. Formal analysis of PIN block attacks. *Theoretical Computer Science*, 367(1-2):257–270, 2006.

[30] F. Thuijsman and O. Vrieze. The bad match; a total reward stochastic game. *Operations-Research-Spektrum*, 9(2):93–99, 1987.

[31] R. Wimmer, N. Jansen, E. Ábrahám, B. Becker, and J.-P. Katoen. Minimal critical subsystems for discrete-time Markov models. In C. Flanagan and B. Knig, editors, *Proc. TACAS'12*, volume 7214 of *Lecture Notes in Computer Science*, pages 299–314. Springer Berlin Heidelberg, 2012. Extended version available as technical report SFB/TR 14 AVACS 88.

[32] `http://www.ilog.com/products/cplex/`.

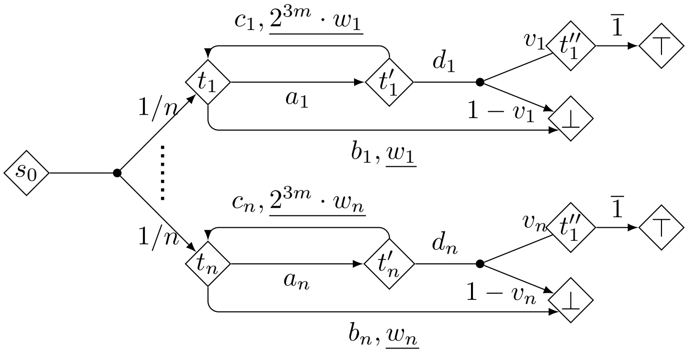
[33] `http://www.gurobi.com/`.

[34] `http://www.prismmodelchecker.org/files/tacas14pcs/`.

Appendix A. Appendix

A.1. **Proof of Theorem 3.8 (NP-hardness).** 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 $2^{-m} < \frac{1}{q}$ and $2^{-m} \cdot W \leqslant \frac{1}{q}$.



The rewards and penalties are as given by the overlined and underlined expressions, and set to 0 where not present. In particular, note that for any state s different from $\top$ the probability of reaching $\top$ from s is the same as the expected total reward from s.

We show that there is a multi-strategy θ sound for the property $\mathtt{R}^r_{\geqslant V/n}[\mathtt{C}]$ such that $pen_{sta}(\psi, \theta) \leqslant W + 2^{-m} \cdot W$ 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 $i \notin I$.

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 $\alpha_i = \theta(t_1')(\{d_i\})$. We will show that for any $i \in I$ we have $\alpha_i \geqslant 2^{-m}(1 - \beta_i)$. When $\beta_i = 1$, this obviously holds. For $\beta_i < 1$, assume that $\alpha_i \geqslant 2^{-m}(1 - \beta_i)$. Because the optimal strategy σ will pick b_i and c_i whenever they are available, we have:

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

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

which is a contradiction with $i \in I$. Hence, $\alpha_i \geqslant 2^{-m}(1-\beta_i)$ and so:

$$pen_{sta}(\psi,\theta,t_i) + pen_{sta}(\psi,\theta,t_i') = \beta_i w_i + \alpha_i 2^{3m} w_i \geqslant \beta_i w_i + 2^{-m}(1-\beta_i)2^{3m} w_i \geqslant w_i$$

We have:

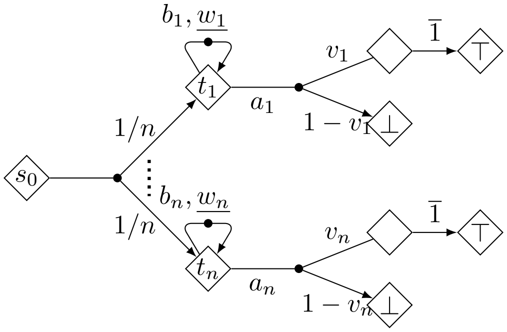
$$\sum_{i\in I} w_i \leqslant \sum_{i\in I} \big(pen_{sta}(\psi,\theta,t_i) + pen_{sta}(\psi,\theta,t_i')\big) \leqslant W + 2^{-m}\cdot W$$

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 $\Leftarrow$, 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 $\Rightarrow$, 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 $I \subseteq \{1,\dots,n\}$ 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:

$$\frac{1}{n}W \geqslant pen_{dyn}(\psi,\theta) = \frac{1}{n}\sum_{i=0}^{n}\sum_{j=0}^{\infty}(1-x_i)^j x_i w_i = \frac{1}{n}\sum_{i\in I}\sum_{j=0}^{\infty}(1-x_i)^j x_i w_i = \frac{1}{n}\sum_{i\in I} w_i \quad \text{(A.1)}$$

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 (A.1) 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.2. **Proof of Theorem 3.9 (Upper Bounds).** We consider the two cases of deterministic and randomised multi-strategies separately, showing that they are in NP and PSPACE, respectively. To simplify readability, the proofs make use of constructions that appear later in the main paper than Theorem 3.9. Note that those constructions do not build on the theorem and so there is no cyclic dependency.

Deterministic multi-strategies. For deterministic multi-strategies, it suffices to observe that the problem can be solved by verifying an MILP instance constructed in polynomial time (and, additionally, in the case of dynamic penalties: a polynomial-time identification of the infinite-penalty case – see Theorem 4.2 and Theorem 4.3). Since the problem of solving an MILP instance is in NP, the result follows.

Randomised multi-strategies. We now show that the permissive controller synthesis problem is in PSPACE for randomised multi-strategies and static penalties. The proof for dynamic penalties 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.

We do not construct the formula Ψ explicitly, but only sketch the main idea. Recall that in Section 4.3 we presented a reduction that allows us to *approximate* the existence of a multi-strategy using the construction described on page 15 and in Fig. 4. Note that if we *knew* the probabilities with which the required multi-strategy θ chooses some sets in a state s, we could use these probabilities instead of the numbers $p_1, \ldots, p_m$ in Fig. 4. In fact, by Theorem 4.4 we would only need $n = 2$, i.e. two numbers per state. Now knowing these numbers p_1^s, p_2^s for each state, we can construct, in polynomial time a polynomial-size instance (disregarding the size of representation of the numbers p_1^s, p_2^s) of an MILP problem such that the optimal solution for the problem is the optimal reward/penalty under the multi-strategy θ. Of course, we do *not* know the numbers p_1^s, p_2^s a priori, and so we cannot use MILP for the solution. Instead, we treat those numbers as variables in the formula Ψ which is build from the constraints in the MILP problem, by adding existential quantification for p_1^s, p_2^s for all $s \in S$ and all other variables from the MILP problem, requiring p_1^s, p_2^s to be from $(0, 1)$ and $p_1^s = 1 - p_2^s$, and adding the restriction on the total reward. Note that in the case of dynamic penalties we need to treat optimal infinite penalty separately, similarly to the construction on page 14.

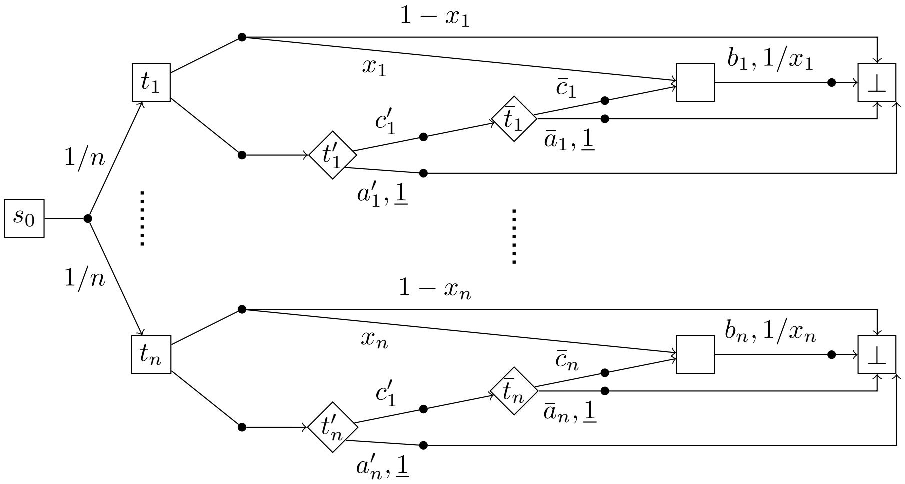


Figure 6: The game for the proof of Theorem 3.10.

A.3. **Proof of Theorem 3.10 (Square-root-sum Reduction).** 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. 6.

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 $pen_{sta}(\psi, \theta) \leqslant 2 \cdot y$ 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 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 $\Rightarrow$, 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 $z_i' \cdot \bar{z}_i \geqslant x_i$ for every i. We show that $z_i' + \bar{z}_i \geqslant 2 \cdot \sqrt{x_i}$, by analysing the possible cases: If both z_i' and $\bar{z}_i$ are greater than $\sqrt{x_i}$, we are done. The case $z_i', \bar{z}_i < \sqrt{x_i}$ 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 $x_i \leqslant (\sqrt{x_i} + p) \cdot (\sqrt{x_i} - q) = x_i + (p - q)\sqrt{x_i} - pq$, 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 $\sum_{i=1}^{n} 2 \cdot \sqrt{x_i} \leqslant \sum_{i=1}^{n} \left(z_i' + \bar{z}_i\right) = pen_{sta}(\psi, \theta) \leqslant 2 \cdot y$.

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 $\mathit{pen}_{\mathit{dyn}}(\psi,\theta) \leqslant 2 \cdot y/n$ 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 $\mathit{pen}_{\mathit{dyn}}(\psi,\theta) = \frac{1}{n}\sum_{i=1}^{n} 2 \cdot \sqrt{y_i}$.

In the direction $\Rightarrow$, let θ be an arbitrary multi-strategy sound for the property $\mathtt{R}^{r}_{\geqslant 1}[\,\mathtt{C}\,]$ satisfying $\mathit{pen}_{\mathit{dyn}}(\psi,\theta) \leqslant 2 \cdot y/n$. 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 $\bar{u}_i = \theta(\bar{t}_i)(\{\bar{a}_i\})$.

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

$$\frac{1}{n}\sum_{i=1}^{n} 2 \cdot \sqrt{x_i} \leqslant \frac{1}{n}\sum_{i=1}^{n}\big(z_i' + \bar{z}_i\big) \leqslant \frac{1}{n}\sum_{i=1}^{n}\big((z_i' + u_i') + (1-u_i')\cdot(\bar{z}_i + \bar{u}_i)\big)$$

$$= \mathit{pen}_{\mathit{dyn}}(\psi,\theta) \leqslant 2 \cdot y/n.$$