Monotonic Safety for Scalable and Data-Efficient Probabilistic Safety Analysis

Matthew Cleaveland
University of Pennsylvania
Philadelphia, USA
mcleav@seas.upenn.edu

Ivan Ruchkin
University of Pennsylvania
Philadelphia, USA
iruchkin@seas.upenn.edu

Oleg Sokolsky
University of Pennsylvania
Philadelphia, USA
sokolsky@seas.upenn.edu

Insup Lee
University of Pennsylvania
Philadelphia, USA
lee@seas.upenn.edu

ABSTRACT

Autonomous systems with machine learning-based perception can exhibit unpredictable behaviors that are difficult to quantify, let alone verify. Such behaviors are convenient to capture in probabilistic models, but probabilistic model checking of such models is difficult to scale — largely due to the non-determinism added to models as a prerequisite for provable conservatism. Statistical model checking (SMC) has been proposed to address the scalability issue. However it requires large amounts of data to account for the aforementioned non-determinism, which in turn limits its scalability. This work introduces a general technique for reduction of non-determinism based on assumptions of "monotonic safety", which define a partial order between system states in terms of their probabilities of being safe. We exploit these assumptions to remove non-determinism from controller/plant models to drastically speed up probabilistic model checking and statistical model checking while providing provably conservative estimates as long as the safety is indeed monotonic. Our experiments demonstrate model-checking speed-ups of an order of magnitude while maintaining acceptable accuracy and require much less data for accurate estimates when running SMC — even when monotonic safety does not perfectly hold and provable conservatism is not achieved.

KEYWORDS

probabilistic & statistical model checking, monotonic safety

1 INTRODUCTION

Model checking is a well-established way to assure a safety-critical CPS, such as a self-driving car, by searching for violations of a safety property in the reachable states of a system model. Recent advances in perception based on machine learning (ML) have challenged model checking with unpredictable, difficult-to-model behaviors [3, 13, 31]. The uncertainties of ML-based perception and the environment where it is deployed call for probabilistic model checking (PMC) [14, 21, 29], which computes a probability that a property holds (e.g., the chance of no collisions). However, unpredictable outputs and qualitative perception errors [5], such as false negatives, can introduce complex behaviors which lead to scalability issues when performing PMC for CPS with ML components.

A common way of addressing these scalability issues is with abstractions, which simplify system models while preserving their properties of interest. For example, instead of treating a system's state space as continuous, one can grid it up into intervals, thus shrinking the size of the model and reducing the run time of probabilistic model checking. However, many abstraction techniques rely on non-determinism to ensure conservatism (e.g., one state grid cell can transition to multiple next grid cells on a given input). The model checker must then account for these non-deterministic choices, which then leads to new scalability issues.

SMC offers an alternative to PMC. SMC samples traces of a probabilistic model and uses statistical methods to obtain guarantees about the model's behavior. In the case of probabilistic models with non-determinism, this is done by employing lightweight scheduler sampling (LSS) [24]. However, LSS needs to separately sample both the probabilistic and non-deterministic behaviors of the system in question, which makes it very data hungry in the presence of large amounts of non-determinism.

To improve the scalability of PMC and data efficiency of LSS, this paper introduces a novel way to reduce non-determinism in the intermediate abstractions of models. This reduction is based on *assumptions of monotonic safety* (MoS) that capture our intuition about which states have a higher chance of safety than others. Encoded as partial orders, these assumptions draw on domain-specific knowledge to characterize broad safety principles in a particular system. For instance, in an obstacle-avoidance scenario, being further away from an obstacle is generally safer than being closer to it. MoS assumptions can be used to simplify models. These simplifications intend to maintain conservative safety estimates and, when MoS assumptions hold, are provably conservative. Specifically, we simplify models using MoS-driven *trimming* of non-deterministic transitions leading to MoS-safer states. This reduction in non-determinism speeds up PMC and improves the data efficiency of LSS.

Ultimately, MoS assumptions are heuristic and may not strictly hold in all situations: in rare cases it is worth being a little closer to the obstacle so that the perception is more accurate. Typically, it is not immediately clear where MoS holds or how to efficiently compute that MoS holds. However, we show empirically that MoS assumptions do not need to strictly hold in every state to be useful, as long as they hold most of the time.

We instantiate our MoS assumptions and MoS-driven trimming on two simulated case studies: automated braking for an autonomous vehicle and flow control of water tanks. Our MoS-driven trimming produces abstractions which estimate the safety probability of the system and are compared with standard conservative abstractions. Experiments show that our abstractions improve the scalability of PMC by an order of magnitude and the data efficiency of LSS by at least an order of magnitude. Moreover, we observe that our abstractions remain empirically conservative, even when not provably so: they do not overestimate the chance of safety in practice, even though MoS assumptions only hold in most of the states.

This paper makes three contributions: (i) a novel notion of MoS assumptions and their use for transition trimming, (ii) conservatism proofs for transition trimming under MoS assumptions, (iii) an application of MoS-trimmed abstractions to emergency braking systems and flow control in water tanks, resulting in high-performing instances of abstractions and quantifications of the MoS assumptions.

The rest of the paper is organized as follows. We introduce two complementary motivating systems in Section 2 and give the necessary formal background in Section 3. Section 4 introduces MoS assumptions. Section 5 explains how we exploit MoS assumptions to trim non-determinism from models. We describe the results of our two case studies in Section 6. The paper ends with related work (Section 7) and a conclusion (Section 8).

2 MOTIVATING SYSTEMS

Throughout the paper, we consider two systems with complementary complexities. The first one, *emergency braking*, has a stateless multi-output controller, a stateful binary-output perception, and a total-order MoS in each state dimension. The second one, *water tanks*, has a stateful binary-output controller, a stateless multi-output perception, and a U-shaped MoS assumption.

2.1 First System: Emergency Braking

Consider an autonomous car approaching a stationary obstacle. The car is equipped with a controller which issues braking commands on detection of the obstacle and an ML-based perception system which detects obstacles. The safety goal of the car is to fully stop before hitting the obstacle. Following are the car dynamics, controller, and perception used in the paper.

A discrete kinematics model with time step τ represents the velocity (v) and position (d, same as the distance to the obstacle) of the car at time t:

$$d[t] = d[t-\tau] - \tau \times v[t-\tau], \quad v[t] = v[t-\tau] - \tau \times b[t-\tau], \tag{1}$$

where $b[t]$ is the braking command (a.k.a. the "braking power", BP) at time t.

We use an Advanced Emergency Braking System (AEBS) [23] which uses two metrics to determine the BP: time to collide (TTC) and warning index (WI). The TTC is the amount of time until a collision if the current velocity is maintained. The WI represents how safe the car would be in the hands of a human driver (positive is safe, negative is unsafe).

$$TTC = \frac{d}{v}, \quad WI = \frac{d - d_{br}}{v \cdot T_h}, \quad d_{br} = v \cdot T_s + \frac{u \cdot v^2}{2a_{max}}, \tag{2}$$

where d_{br} is the braking-critical distance, T_s is the system response delay (negligible), T_h is the average driver reaction time (set to 2s), u is the friction scaling coefficient (taken as 1), and a_{max} is the maximum deceleration of the car.

Upon detecting the obstacle, the AEBS chooses one of three BPs: no braking (BP of 0), light braking (B_1), and maximum braking ($B_2 = a_{max}$). If the obstacle is not detected, no braking occurs. The BP value, b, is determined by WI and TTC crossing either none, one, or both of the fixed thresholds C_1 and C_2:

$$\begin{aligned} WI > C_1 \land TTC > C_2 &\implies b = 0 \\ (WI \leq C_1 \land TTC > C_2) \lor (WI > C_1 \land TTC \leq C_2) &\implies b = B_1 \\ WI \leq C_1 \land TTC \leq C_2 &\implies b = B_2 \end{aligned}$$

To detect the obstacle, the car uses the deep neural network YoloNetv3 [25], as it can run at high frequencies [27]. We say a low-level detection of the obstacle occurs when Yolo detects the obstacle. To reduce noise, we apply a majority vote filter to the low-level detections, so the AEBS receives a high-level detection with the distance to the obstacle if at least 2 of the past 3 Yolo outputs were low-level detections. We model Yolo probabilistically based on two observations. First, Yolo's chance of detection is higher when the car is closer to the obstacle. Second, consecutive low-level detections correlate with each other due to weather conditions and similar car positions. Our perception model conditions the detection chance on the distance from the obstacle and the recent detection history.

The safety property of interest, ψ_{nocol}, is the absence of a collision, specified in linear temporal logic (LTL) [26] as

$$\psi_{nocol} := \Box \ (d > L), \tag{3}$$

where L is the minimum allowed distance to the obstacle and is taken to be 5m. Note that in this model the car eventually stops or collides, so for any initial condition there is an upper bound on the number of time steps.

2.2 Second System: Water Tanks

Consider a system consisting of J water tanks, each of size TS, draining over time and a controller that maintains some water level in each tank. With $w_i[t]$ as the water level in the i^{th} tank at time t, the discrete time dynamics for the water level in the tank at the next time step is given by:

$$w_i[t+1] = w_i[t] - out_i[t] + in_i[t], \tag{4}$$

where $in_i[t]$ and $out_i[t]$ are the amounts of water entering ("inflow") and leaving ("outflow") respectively the i^{th} tank at time t. The inflow is determined by the controller and the outflow is a constant determined by the environment.

Each tank is equipped with ML-based perception to report its current perceived water level, $\hat{w}$, which is a random function of the true current water level, w. In the simulated system (playing the role of the reality), $\hat{w}$ is determined as a positively-skewed high-variance mixture-of-gaussians error added to w. The gaussians have higher variance for water levels in the middle of the tank and smaller variance near the edges of the tank. In addition, with constant probability the perception can output $\hat{w}$ =0 or $\hat{w}$ =TS (to account for a large range of perception errors).

We model perception probabilistically as a categorical distribution of perception error, e. The possible error values are integers within some EW-sized error window,

$$-EW, \ldots, -1, 0, 1, \ldots, EW,$$

plus two special values to capture unexpected outputs of ML-based perception (e.g., due to water reflections or deep network peculiarities): FU to indicate that a spurious reading of the tank being full occurred (then $\hat{w} = TS$), and EM to indicate that a spurious reading of the tank being empty occurred (then $\hat{w} = 0$). Hence, this perception model has $2EW + 3$ parameters.

The controller has a single source of water to fill one tank at a time (or none at all) based on the perceived water levels. Then this tank receives a constant value $in > 0$ of water, whereas the other tanks receive 0 water. The controller only starts filling a tank that has water below the lower decision threshold LT and stops filling after the water reaches the upper threshold UT. After filling a tank or taking one step of not filling, if the controller perceives multiple tanks below LT, it starts filling the one with the least perceived water (or, if equal, with the lowest ID $1 \dots J$).

The safety property, ψ_{flow}, is that each tank must never run out of water (i.e., $w = 0$) or overflow (i.e., $w = TS$). This property can be expressed as the following time-bounded LTL formula:

$$\psi_{flow} := \square_T \ (0 < w_1 < TS \wedge \cdots \wedge 0 < w_J < TS) \qquad (5)$$

where T is a fixed number of time steps within which the tanks should be kept is the water level resulting in overflow of tanks.

3 BACKGROUND, SYSTEM MODELS, AND LSS

In the following Definitions 3.1 to 3.3, borrowed from Kwiatkowska et al. [22], we use $Dist(S)$ to refer to the set of probability distributions over a set S, η_s as the distribution with all its probability mass on $s \in S$, and $\mu_1 \times \mu_2$ to be the product distribution of μ_1 and μ_2.

Definition 3.1. A *probabilistic automaton* (PA) is a tuple $\mathsf{M} = (S, \bar{s}, \alpha, \delta, L)$, where S is a finite set of states, $\bar{s} \in S$ is the initial state, α is an alphabet of action labels, $(S, \alpha, Dist(S)) \in \delta$ is a probabilistic transition relation, and $L : S \rightarrow 2^{AP}$ is a labeling function from states to sets of atomic propositions from the set AP.

If $(s, a, \mu) \in \delta$ then the PA can make a transition in state s with action label a and move based on distribution μ to state s' with probability $\mu(s')$, which is denoted by $s \xrightarrow{a} \mu$. If $(s, a, \eta_{s'}) \in \delta$ then we say the PA can transition from state s to state s' via action a. A state s is terminal if no elements of δ contain s. A path in M is a finite/infinite sequence of transitions $\pi = s_0 \xrightarrow{a_0, \mu_0} s_1 \xrightarrow{a_1, \mu_1} \dots$ with $s_0 = \bar{s}$ and $\mu_i(s_{i+1}) > 0$. A set of paths is denoted as Π. We use $\mathsf{M}(s)$ to denote the PA M with initial state s.

Reasoning about PAs also requires the notion of *schedulers*, which resolve the non-determinism during an execution of a PA. For our purposes, a scheduler σ maps each state of the PA to an available action label in that state. We use $\Pi^{\sigma}_{\mathsf{M}}$ for the set of all paths through M when controlled by scheduler σ and Sch_{M} for the set of all schedulers for M. Finally, given a scheduler σ, we define a probability space Pr^{σ}_{M} over the set of paths $\Pi^{\sigma}_{\mathsf{M}}$ in the standard manner.

Given PAs M_1 and M_2, we define parallel composition as:

Definition 3.2. The *parallel composition* of PAs $\mathsf{M}_1 = (S_1, \bar{s}_1, \alpha_1, \delta_1, L_1)$ and $\mathsf{M}_2 = (S_2, \bar{s}_2, \alpha_2, \delta_2, L_2)$ is given by the PA $\mathsf{M}_1 || \mathsf{M}_2 = (S_1 \times S_2, (\bar{s}_1, \bar{s}_2), \alpha_1 \cup \alpha_2, \delta, L)$, where $L(s_1, s_2) = L_1(s_1) \cup L_2(s_2)$ and δ is such that $(s_1, s_2) \xrightarrow{a} \mu_1 \times \mu_2$ iff one of the following holds: (i) $s_1 \xrightarrow{a} \mu_1, s_2 \xrightarrow{a} \mu_2$ and $a \in \alpha_1 \cap \alpha_2$, (ii) $s_1 \xrightarrow{a} \mu_1, \mu_2 = \eta_{s_2}$ and $a \in (\alpha_1 \setminus \alpha_2)$, (iii) $\mu_1 = \eta_{s_1}, s_2 \xrightarrow{a} \mu_2$ and $a \in (\alpha_2 \setminus \alpha_1)$.

In this paper, we are concerned with probabilities of safety properties, which we state using LTL formulas over state labels.

Definition 3.3. For LTL formula ψ, PA M, and scheduler $\sigma \in Sch_{\mathsf{M}}$, the *probability of* ψ *holding* is:

$$Pr^{\sigma}_{\mathsf{M}}(\psi) := Pr^{\sigma}_{\mathsf{M}}\{\pi \in \Pi^{\sigma}_{\mathsf{M}} \mid \pi \models \psi\}$$

where $\pi \models \psi$ indicates that the path π satisfies ψ in the standard LTL semantics [26]. We specifically consider LTL safety properties, which are LTL specifications that can be falsified by a finite trace though a model. Both ψ_{nocol} and ψ_{flow} are LTL safety properties.

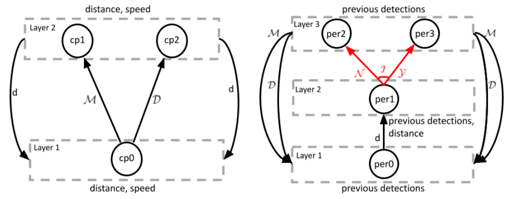


Figure 1: The models of controller/plant $\mathsf{M}_{\mathbf{cpl}}$ (left) and perception $\mathsf{M}_{\mathbf{per}}$ (right) in the emergency braking system. The dashed grey boxes indicate layers of states, with one state for each pair of distances and speeds. The red transition is probabilistic.

Probabilistically verifying an LTL formula ψ against M requires checking that the probability of satisfying ψ meets a probability bound for all schedulers. This involves computing the minimum or maximum probability of satisfying ψ over all schedulers:

$$Pr^{min}_{\mathsf{M}}(\psi) := \inf_{\sigma \in Sch_{\mathsf{M}}} Pr^{\sigma}_{\mathsf{M}}(\psi)$$
$$Pr^{max}_{\mathsf{M}}(\psi) := \sup_{\sigma \in Sch_{\mathsf{M}}} Pr^{\sigma}_{\mathsf{M}}(\psi)$$

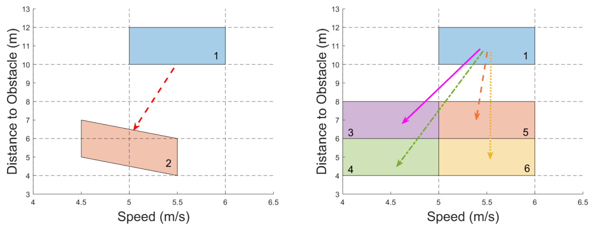
We call σ a min scheduler of M if $Pr^{\sigma}_{\mathsf{M}}(\psi) = Pr^{min}_{\mathsf{M}}(\psi)$. We use Sch^{min}_{M} to denote the set of min schedulers of M.

3.1 Probabilistic Automata for Motivating Systems

We now describe how we construct the baseline PAs for the systems described in Section 2. We always start with two models: a non-probabilistic, discrete time, infinite state controller-plant model M_{cpl} and a probabilistic perception model M_{per}. Our goal is to abstract them as PAs and compose them so that we can analyze their safety properties. As an illustration, we discuss our models for the emergency braking system, shown in Figure 1. M_{per} probabilistically generates either detections or non-detections of the obstacle and sends them to M_{cpl}, which then selects a braking command in response to the detection/non-detection and updates its (continuous) distance and speed values accordingly. M_{cpl} then sends the distance to the obstacle back to M_{per}, which it uses to alter its detection/non-detection probabilities. We now describe how we construct M_{per} as a PA before describing how to form a standard PA abstraction of M_{cpl}. Note that M_{cpl} is not a PA since it has an infinite number of states.

M_{per} has three actions — $\mathcal{D}$ (detection), $\mathcal{M}$ (non-detection), *distance=d* (d), all of which are synchronized with M_{cpl}. Colloquially, we say that M_{per} transmits actions $\mathcal{D}$ and $\mathcal{M}$ to M_{cpl} and M_{cpl} transmits parameterized action d to M_{per}. M_{per} conditions its detection probabilities on both the distance d and its last W low-level readings and uses a majority-vote filter over its last N_F low-level readings to determine its output of either $\mathcal{D}$ or $\mathcal{M}$, as described in Section 2. The choice of low-level detection is the only probabilistic transition in M_{per} (and the whole system). M_{per} waits for M_{cpl} to transmit its distance. Then it makes an internal detection and checks if at least half of the previous N_F low-level readings were detections. If so, it transmits $\mathcal{D}$. If not, it transmits $\mathcal{M}$. Either way, it then waits for M_{cpl} to transmit the new distance d.

We now describe M_{cpl} and then create PA abstraction M_{int}. Let $S \subset \mathbb{R}^n$ be the state space of M_{cpl}, $\bar{s}$ be the initial state, $K = \{\mathcal{D}, \mathcal{M}\}$ be the set of M_{per} inputs, and $f : S \times K \rightarrow S$ be the discrete



(a) **(b)**

Figure 2: In Figure 2a a set of distance and speed states (box 1) are grouped into a single entity and the set of possible next states (polygon 2) is computed. In Figure 2b the set of next states is used to compute the new transitions from box 1, which go to boxes 3-6.

transition relation (i.e. if M_{cpl} is in state $s \in S$ and receives high-level detection $k \in K$ from M_{per}, then at the next time point it will be in state $f(s,k)$).

The state space of M_{int}, denoted S', is formed by dividing S into a discrete, finite set of regions using equally sized intervals[1] (see the squares formed by the dashed lines in Figure 2a). So every $s'_1 \in S'$ has a corresponding region $S_1 \subset S$. The initial state $\bar{s}'$ is the state in S' which contains $\bar{s}$ in its set of corresponding states. The set of M_{per} inputs $K = \{\mathcal{D}, \mathcal{M}\}$ is unchanged. The transition relation δ' is formed as follows:

$$(s'_1, (k \times \alpha'), \eta_{s'_2}) \in \delta' \text{ if} \qquad (6)$$
$$\exists s'_2 \in S',\ \exists s_1 \in S_1, s_2 \in S_2,\ s.t. f(s_1,k) = s_2$$

In other words, M_{int} has a transition from s'_1 to s'_2 (in M_{int}) if at least one state in S_1 has a transition to a state in S_2 (in M_{cpl}). For an illustration of this, see Figure 2. Note that δ' is non-deterministic due to the perception values K and the reachability analysis over the intervals of states in M_{cpl}. The latter of two types of non-determinism is captured by the extra action labels $\alpha' \in A'$. Each instance of the reachability non-determinism gets its own unique action label from A' (see the different arrows in Figure 2b) and the perception non-determinism will be resolved when the model gets composed with M_{per}. Whenever M_{int} makes a transition, it transmits the distance range corresponding to its next state s'_2 for M_{per} to synchronize on. The discrete states $s'_i \in S'$ inherit any atomic labels $l \in L$ from the states in S_i, leading to labelling function L'. Finally, we convert M_{int} into a PA using the tuple notation from definition Definition 3.1: $M_{int} = \{S', s'_0, K \times A', \delta', L'\}$.

We now form our baseline model, which we also denote as M_{int}, by taking $M_{per} || M_{int}$. Note that the resulting model has non-determinism in its transition relation, which our MoS assumptions will aim to remove.

For the water tank system, the M_{per} and M_{int} models are constructed from equations in Section 2 in a similar fashion: M_{per} encodes the perception probabilistically and M_{int} is a PA based interval abstraction of M_{cpl} encoding the controller and plant non-probabilistically. M_{int} communicates the true water levels $(w_1 \dots w_J)$ to M_{per} and receives the perceived water levels $(\hat{w}_1 \dots \hat{w}_J)$.

[1]The size of these intervals is a modeling hyperparameter. We explore how it affects the abstractions in Section 6.

The execution terminates either when the safety property is violated or time T is reached.

3.2 Statistical Model Checking for PAs

This section formalizes the process of running LSS on PA M to approximate $Pr^{min}_{M}(\psi)$. First, a set of schedulers are sampled uniformly from Sch_M:

$$\sigma_1, \dots, \sigma_n \sim Unif(Sch_M)$$

Then the probability of M under scheduler σ_i, which we denote as $p_i := Pr^{\sigma_i}_{M}(\psi)$, is computed via sampling traces from the fully probabilistic model of M when scheduler σ_i is fixed. Finally, the smallest probability is returned: $Pr^{LSS,n}_{M}(\psi) = min(p_1, \dots, p_n)$.

Since LSS relies on sampling schedulers, its output is a random variable. So consecutive runs of LSS can give different results. In addition, running LSS requires n iterations of standard statistical analysis of M with fixed schedulers. Reducing n lowers the run times of LSS, but will also give less conservative results on average. The ideal choice of n depends both on the computational resources at hand and the estimated size of the set of schedulers $|Sch_M|$. LSS will never output a smaller value than PMC, so $Pr^{LSS,n}_{M}(\psi) \geq Pr^{min}_{M}(\psi)$. However, as $n \to \infty$, $P\left(Pr^{LSS,n}_{M}(\psi) = Pr^{min}_{M}(\psi)\right) \to 1$, which is to say that with enough samples LSS will eventually find the smallest satisfaction probability. In practice $|Sch_M|$ will be extremely large, thus requiring very large n to obtain reasonable safety estimates.

Remark. *For a complete description of LSS, see [11, 24].*

4 ASSUMPTIONS OF MONOTONIC SAFETY

This section starts by formalizing assumptions of *monotonic safety* (MoS) for both PMC and LSS[2] and stating the intuitive MoS assumptions for the motivating systems. Then we observe that MoS does not always hold in practice, exemplified in our motivating systems.

Definition 4.1 (Assumption of Monotonic Safety for PMC). An *MoS assumption for PMC* is a partial order, $\geq^{PMC}_{MoS}$, over the states S with respect to a given safety property ψ and a PA M. It stipulates that if two states $s_1, s_2 \in S$ are ordered, $s_1 \geq^{PMC}_{MoS} s_2$, then for every min scheduler σ of M, the probability of M satisfying ψ when using scheduler σ and starting from state s_1 is greater or equal to the probability of M satisfying ψ when using scheduler σ and starting from state s_2:

$$s_1 \geq^{PMC}_{MoS} s_2 \implies \forall \sigma \in Sch^{min}_{M} \ \ Pr^{\sigma}_{M(s_1)}(\psi) \geq Pr^{\sigma}_{M(s_2)}(\psi)$$

Definition 4.2 (Assumption of Monotonic Safety for LSS). An *MoS assumption for LSS* is a partial order, $\geq^{LSS}_{MoS}$, over the states S with respect to a given safety property ψ and PA M. It stipulates that if two states $s_1, s_2 \in S$ are ordered, $s_1 \geq^{LSS}_{MoS} s_2$, then for every scheduler σ of M, the probability of M satisfying ψ when using scheduler σ and starting from state s_1 is greater or equal to the probability of M satisfying ψ when using scheduler σ and starting from state s_2:

$$s_1 \geq^{LSS}_{MoS} s_2 \implies \forall \sigma \in Sch_M \ \ Pr^{\sigma}_{M(s_1)}(\psi) \geq Pr^{\sigma}_{M(s_2)}(\psi)$$

[2]The two techniques require different MoS assumptions because they handle non-determinism differently.

These definitions are very general. The partial orders could relate exactly two states in S, or they could order some subset of states in S, or they could even be a full ordering over states in S.

Such assumptions are to be made by the engineers familiar with the domain and the system at hand. By formalizing an MoS assumption, an engineer aims to integrate their high-level knowledge of the system's operation into the formal analyses of this system. Several assumptions can be made for a given system, and if they do not contradict each other they can be combined into a single overarching partial order.

One way to find candidates for MoS assumptions is to examine the robustness, in terms of Signal Temporal Logic (STL) [12], of the safety property ψ. Intuitively, if state s_1 is more STL-robust with respect to ψ than state s_2, then it may have a higher chance of remaining safe in the subsequent execution. So one can formulate an MoS partial order based on the direction of STL robustness. However, MoS assumptions can draw on substantially broader intuitions than what can be gleaned from the safety property.

For our case studies, we consider two intuitive MoS assumptions for our emergency braking system and its safety property ψ_{nocol}, and one intuitive MoS rule for our water tank system and its safety property ψ_{flow}. In all three rules, consider model M with different initial states s_1 and s_2:

- Distance MoS $\succeq_d$: *"Increasing the distance to the obstacle does not reduce the safety chance".* That is, if M_1 starts further away, i.e., $d_{s_1} \geq d_{s_2}$, then it is safer than M_2: $Pr_{\mathsf{M}(s_1)}(\psi_{nocol}) \geq Pr_{\mathsf{M}(s_2)}(\psi_{nocol})$.
- Speed MoS $\succeq_v$:*"Decreasing the velocity does not reduce the safety chance".* That is, if M_1 starts slower, i.e., $v_{s_1} \leq v_{s_2}$, then it is safer than M_2: $Pr_{\mathsf{M}(s_1)}(\psi_{nocol}) \geq Pr_{\mathsf{M}(s_2)}(\psi_{nocol})$.
- Water level MoS $\succeq_w$: *"Moving the water level in any tank towards its middle does not reduce safety chance".* This partial order has two branches, for each half of the tank. First, if $w_{s_2} \geq w_{s_1} \geq \frac{TS}{2}$, then $P_{\mathsf{M}(s_1)}(\psi_{flow}) \geq P_{\mathsf{M}(s_2)}(\psi_{flow})$. Second, if $w_{s_2} \leq w_{s_1} \leq \frac{TS}{2}$, then $P_{\mathsf{M}(s_1)}(\psi_{flow}) \geq P_{\mathsf{M}(s_2)}(\psi_{flow})$.

Note that assumptions $\succeq_d$ and $\succeq_w$ order states in the higher-robustness direction of ψ_{nocol} and ψ_{flow} respectively. But the second AEBS assumption, $\succeq_v$, is an example of a more general rule not tied to ψ_{nocol}. It was discovered by observing that higher speeds lead to greater distance losses, and if greater distances are to be considered safer, so should be the lower speeds. Before describing how to exploit MoS assumptions to reduce non-determinism in PAs in Section 5, we present some simple counter examples showing that $\succeq_d$, $\succeq_v$, and $\succeq_w$ do not always holds.

4.1 Violations of Monotonic Safety in the Motivating Systems

For our systems, the introduced MoS assumptions rules appear natural and intuitive: to avoid a collision, being further away and driving slower seems safer; to avoid overflowing and underflowing, it appears safer to keep the tanks half-full. As we will experimentally show in Section 6, these intuitions are mostly correct. However, our assumptions may not hold in some pairs of states, which we now demonstrate on simplified models. The proofs of these counterexamples are provided in Appendix A.1 and Appendix A.2.

When discussing MoS scenarios in concrete systems, we consider *state shifts* encoded as changes to states with a state vector Δ. AEBS has two-dimensional states (d, v) and, hence, two types of shifts corresponding to $\succeq_d$ and $\succeq_v$: $(\Delta_d > 0, 0)$ and $(0, \Delta_v < 0)$. Each water tank is unidimensional, so we consider scalar shifts Δ_w, such that $\Delta_w > 0$ iff $w \leq \frac{TS}{2}$; otherwise, $\Delta_w < 0$.

Definition 4.3. An AEBS system has *distance-independent perception* if the detection chance is constant: $\forall d, Pr_{det}(d) = q$. Otherwise, it is *distance-dependent.*

Definition 4.4. An AEBS system has *one BP* if the controller bounds in Section 2 are infinite: $C_1 = C_2 = +\infty$. Otherwise, it has *multiple BPs.*

With two counterexamples for AEBS, we demonstrate that multiple BPs or distance-dependent perception alone are sufficient to violate assumptions $\succeq_d$ and $\succeq_v$. In both AEBS counterexamples, we set the time step $\tau = 1s$ and the filter window size $N_F = 1$.

The first counterexample shows that starting closer to the obstacle may be safer due to higher detection chances at closer distances.

Counterexample 1. *Assumption $\succeq_d$ does not hold in the AEBS with one BP and distance-dependent perception for shift $\Delta = (1m, 0)$ when $B = 10m/s^2, v_0 = 11m/s, d_0 = 13m$, and $Pr_{det}(d) = 1 - \lceil d \rceil /20$.*

The second counterexample shows that going at a faster speed may be beneficial because it leads to closer distances, which result in stronger braking.

Counterexample 2. *Assumption $\succeq_v$ does not hold in the AEBS with multiple BPs and distance-independent perception for shift $\Delta = (0, -1m/s)$ when $B(d) = [10m/s^2 \text{ if } d \leq 11m; 3m/s^2 \text{ otherwise}]$, $v_0 = 9m/s$, $d_0 = 20m$, and $Pr_{det} = 0.5$.*

Intuitively, the above counterexamples arise when, due to distance or speed shifting, the model crosses a boundary into a different braking mode or a detection bin with a different probability.

In the water tank system, the safest state is almost never $\frac{TS}{2}$ because of the asymmetry in *in* vs. *out*, controller choices, perception errors. Therefore, it is straightforward to upset the safety balance by picking disproportionate values for inflow and outflow.

Definition 4.5. A water tank system has *random perception* if $EW = 0$ and $\forall i, \hat{w}_i \in \{EM, FU\}$.

Counterexample 3. *Assumption $\succeq_w$ does not hold for a water tank system with $J = 1, TS = 100, out = 3, in = 40, T = 4, Pr(\hat{w} = 0) = 0.4$, and $Pr(\hat{w} = TS) = 0.6, w = 10$, and $\Delta_w = 30$.*

5 MOS-BASED TRIMMING

This section describes how we use the MoS assumptions from Section 4 to remove non-determinism from PAs to speed up PMC and LSS. The two trimming procedures are slightly different, due to the differences in how PMC and LSS handle non-determinism. Finally, recall that our MoS trimmmings are only applied to the non-probabilistic transitions of the M_{int} models.

5.1 MoS Abstraction for Model Checking

We describe how to remove non-determinism from a single state as follows:

Definition 5.1 (Single State PMC Transition Trimming). Let PA $\mathsf{M} = \{S, \bar{s}, \alpha, \delta, L\}$ with MoS assumption $\geq_{MoS}^{PMC}$ be given. Consider $s \in S$ and actions $\alpha_1, \alpha_2 \in \alpha$. If there exist states s_1 and s_2 such that $(s, \alpha_1, \eta_{s_1}), (s, \alpha_2, \eta_{s_2}) \in \delta$ and $s_1 \geq_{MoS}^{PMC} s_2$, then we return PA M', which is identical to M except that it does not have $(s, \alpha_1, \eta_{s_1})$ in its transition relation.

Remark. *Another way to think about this trimming is that we remove every scheduler that picks α_1 in state s from $Sch_{\mathsf{M}'}$.*

We now prove that under the MoS assumption from Definition 4.1 each application of the trimming procedure in Definition 5.1 is conservative. First, we introduce the following lemma which states that the probability of M satisfying safety property ψ can be decomposed by reasoning about the paths of M which do and do not pass through some state s. The proof can be found in Appendix A.3.

Lemma 5.2 (Satisfaction Probability by Trace Decomposition). *Let M be a PA. Assume that there are no infinite paths through M. Let ψ be an LTL safety property. Let σ be a scheduler of M and let s be a state of M. Then*

$$\begin{aligned} Pr_{\mathsf{M}}^{\sigma}(\psi) &= Pr_{\mathsf{M}}^{\sigma}(\psi \& \neg \diamond s) + Pr_{\mathsf{M}}^{\sigma}(\psi \& \diamond s) \\ &= Pr_{\mathsf{M}}^{\sigma}(\psi \& \neg \diamond s) + Pr_{\mathsf{M}}^{\sigma}(\diamond s) Pr_{\mathsf{M}(s)}^{\sigma}(\psi) \end{aligned}$$

We now state the single state trimming conservatism result and its proof.

Theorem 5.3 (PMC MoS and PMC Transition Trimming Implies PMC Conservatism). *Let $\mathsf{M} = \{S, \bar{s}, \alpha, \delta, L\}$ be a PA with MoS assumption $\geq_{MoS}^{PMC}$. Assume that there are no infinite paths through M. Let ψ be an LTL safety property. If we trim M according to the procedure detailed in Definition 5.1 for state s to arrive at model M', then $Pr_{\mathsf{M}}^{min}(\psi) = Pr_{\mathsf{M}'}^{min}(\psi)$.*

Proof. Assume that state s has non-deterministic transitions to state s_1 via action α_1 and state s_2 via action α_2:

$$(s, \alpha_1, \eta_{s_1}), (s, \alpha_2, \eta_{s_2}) \in \delta$$

In addition, assume that $s_1 \geq_{MoS}^{PMC} s_2$. Now consider PA M' that is formed by removing $(s, \alpha_1, \eta_{s_1})$ from δ. To prove that the trimming is conservative, we just need to show that at least one min scheduler of M for ψ is contained in the set of schedulers for M'.

Let σ be a min scheduler of M for ψ. So $Pr_{\mathsf{M}}^{\sigma}(\psi) = Pr_{\mathsf{M}}^{min}(\psi)$. We now show that either σ or some other min scheduler is contained in the set of schedulers for M' by contradiction. Assume that $\sigma \notin Sch_{\mathsf{M}'}$. By the strong MoS assumption, σ chooses action α_1 in state s. Now consider scheduler σ' which picks action α_2 in state s and is otherwise identical to σ. By Lemma 5.2, we have that:

$$\begin{aligned} Pr_{\mathsf{M}}^{\sigma}(\psi) &= Pr_{\mathsf{M}}^{\sigma}(\psi \& \neg \diamond s) + Pr_{\mathsf{M}}^{\sigma}(\diamond s) Pr_{\mathsf{M}(s)}^{\sigma}(\psi) \\ Pr_{\mathsf{M}}^{\sigma'}(\psi) &= Pr_{\mathsf{M}}^{\sigma'}(\psi \& \neg \diamond s) + Pr_{\mathsf{M}}^{\sigma'}(\diamond s) Pr_{\mathsf{M}(s)}^{\sigma'}(\psi) \end{aligned}$$

Note that σ, σ' differ only in state s and M has no self loops since it only has finite paths. So:

$$Pr_{\mathsf{M}}^{\sigma}(\psi \& \neg \diamond s) = Pr_{\mathsf{M}}^{\sigma'}(\psi \& \neg \diamond s) \text{ and } Pr_{\mathsf{M}}^{\sigma}(\diamond s) = Pr_{\mathsf{M}}^{\sigma'}(\diamond s)$$

For the final two terms, we unroll one step of each of the models from state s under their respective schedulers. Scheduler σ takes M to state s_1 and σ' takes M to state s_2. Since σ, σ' differ only in state s and M cannot loop back to s from s_2, it follows that $Pr_{\mathsf{M}(s_2)}^{\sigma'}(\psi) = Pr_{\mathsf{M}(s_2)}^{\sigma}(\psi)$.

Since $s_1 \geq_{MoS}^{PMC} s_2$ we have that:

$$Pr_{\mathsf{M}(s_1)}^{\sigma}(\psi) \geq Pr_{\mathsf{M}(s_2)}^{\sigma}(\psi) = Pr_{\mathsf{M}(s_2)}^{\sigma'}(\psi)$$

It follows that $Pr_{\mathsf{M}}^{\sigma}(\psi) \geq Pr_{\mathsf{M}}^{\sigma'}(\psi)$. If the inequality is strict then we have a contradiction, as σ is not a min scheduler of M and if the terms are equal then both σ and σ' are both min schedulers and the MoS trimming preserves at least one.

□

The model-wide PMC MoS trimming procedure applies the trimming procedure in Definition 5.1 to all states of the model.

Corollary 5.4. *The model-wide PMC MoS trimming procedure is conservative under Definition 4.1.*

5.2 MoS Abstraction for Statistical Model Checking

We describe how to remove non-determinism from a single state for LSS as follows:

Definition 5.5 (Single State LSS Transition Trimming). Let $\mathsf{M} = \{S, \bar{s}, \alpha, \delta, L\}$ be a PA with MoS assumption $\geq_{MoS}^{LSS}$. Consider $s \in S$. Let $\alpha_1, \ldots, \alpha_d$ be the actions enabled in s. So

$$(s, \alpha_1, \eta_{s_1}), \ldots, (s, \alpha_d, \eta_{s_d}) \in \delta$$

for some $s_1, \ldots, s_d \in S$. If

$$s_1 \geq_{MoS}^{LSS} s_d, \ldots s_{d-1} \geq_{MoS}^{LSS} s_d$$

then we return PA M' which is identical to M except that it does not have $(s, \alpha_1, \eta_{s_2}), \ldots, (s, \alpha_{d-1}, \eta_{s_{n-1}})$ in its transition relation.

Remark. *Another way to think about this trimming is that we remove every scheduler that picks $\alpha_1, \ldots, \alpha_{d-1}$ in state s from $Sch_{\mathsf{M}'}$.*

We now prove that under the MoS assumption from Definition 4.2 each application of the trimming procedure in Definition 5.5 is conservative. Since the LSS outputs are random variables, when we say that the results are conservative we mean that the LSS results on the untrimmed model have first order stochastic dominance (FSD) over those of the trimmed model. This is a formal way of defining that we expect M to return larger safety chances than M'.

Definition 5.6 (First-order Stochastic Dominance (FSD)). We say that random variable A has FSD over random variable B if $F_A(x) \leq F_B(x)\ \forall x$, where $F_A(x)$ and $F_B(x)$ are the cumulative density functions (CDFs) of A and B at value x. We denote this as $A \geq_{FSD} B$.

Remark. *One way to think about this definition is that $A \geq_{FSD} B$ if the probability mass of A lies to the right of that of B.*

We now state the single state conservatism result and give its proof.

Theorem 5.7 (LSS MoS and LSS MoS Trimming Implies LSS Conservatism). *Let M be a PA with MoS assumption $\geq_{MoS}^{LSS}$. Assume that there are no infinite paths through M. Let ψ be an LTL safety property. If we trim M according to the procedure detailed in Definition 5.5 for state s to arrive at model M', then $Pr_{\mathsf{M}}^{LSS,n}(\psi) \geq_{FSD} Pr_{\mathsf{M}'}^{LSS,n}(\psi)$.*

Proof. Let s be a state in M. Assume that state s has non-deterministic transitions to states $s_1, s_2, \ldots, s_d$

$$(s, \alpha_1, \eta_{s_1}), \ldots, (s, \alpha_d, \eta_{s_d}) \in \delta$$

and that:

$$s_1 \geq^{PMC}_{MoS} s_d, \ldots, s_{d-1} \geq^{PMC}_{MoS} s_d$$

Now consider PA M′ formed by removing $(s, \alpha_1, \eta_{s_1}), \ldots, (s, \alpha_d, \eta_{s_{d-1}})$ from δ.

Consider scheduler $\sigma \in Sch_{\mathsf{M}}$. Now we generate its corresponding scheduler in M′, called σ', by requiring σ' to take action α_d to state s_d whenever it is in state s. Effectively, we are saying that of the d choices that σ could make in state s, σ' takes α_d. So each $\sigma' \in Sch_{\mathsf{M}'}$ has d corresponding schedulers in Sch_{M}. It follows that $|Sch_{\mathsf{M}}| = d * |Sch_{\mathsf{M}'}|$.

Now we show that $Pr^{\sigma}_{\mathsf{M}}(\psi) \geq Pr^{\sigma'}_{\mathsf{M}'}(\psi)$. By Lemma 5.2, we have:

$$Pr^{\sigma}_{\mathsf{M}}(\psi) = Pr^{\sigma}_{\mathsf{M}}(\psi \& \neg \diamond s) + Pr^{\sigma}_{\mathsf{M}}(\diamond s) Pr^{\sigma}_{\mathsf{M}(s)}(\psi)$$
$$Pr^{\sigma'}_{\mathsf{M}'}(\psi) = Pr^{\sigma'}_{\mathsf{M}'}(\psi \& \neg \diamond s) + Pr^{\sigma'}_{\mathsf{M}'}(\diamond s) Pr^{\sigma'}_{\mathsf{M}'(s)}(\psi)$$

Note that σ, σ' differ only in state s and M has no self loops since it only has finite paths. So:

$$Pr^{\sigma}_{\mathsf{M}}(\psi \& \neg \diamond s) = Pr^{\sigma'}_{\mathsf{M}'}(\psi \& \neg \diamond s) \text{ and } Pr^{\sigma}_{\mathsf{M}}(\diamond s) = Pr^{\sigma'}_{\mathsf{M}'}(\diamond s)$$

For the final two terms, we unroll one step of the models from state s. This one step takes model M to one of the following states: $\{s_1, s_2, \ldots, s_d\}$ (determined by scheduler σ) and model M′ to state s_d (since M′ doesn't have non-deterministic transitions to $s_1, \ldots, s_{d-1}$). By $\geq^{LSS}_{MoS}$, $Pr^{\sigma}_{\mathsf{M}(s_d)}(\psi) \leq Pr^{\sigma}_{\mathsf{M}(s_i)}(\psi), \quad i = 1, \ldots, d-1$. Finally, since M and M′ and σ and σ' differ only in state s and M cannot loop back to s from $s_2, \ldots, s_{d-1}$, it follows that $Pr^{\sigma}_{\mathsf{M}(s_d)}(\psi) = Pr^{\sigma'}_{\mathsf{M}'(s_d)}(\psi)$. So $Pr^{\sigma}_{\mathsf{M}}(\psi) \geq Pr^{\sigma'}_{\mathsf{M}'}(\psi)$.

Now draw some $\sigma_1 \sim Unif(Sch_{\mathsf{M}})$. Let σ'_1 be the corresponding scheduler in $sch_{\mathsf{M}'}$. It follows that $\sigma'_1 \sim Unif(sch_{\mathsf{M}'})$:

$$P(\sigma'_1 = \sigma') = P(\sigma_1 \text{ corresponds to } \sigma') = \frac{1}{|Sch_{\mathsf{M}}|} d = \frac{1}{|Sch_{\mathsf{M}'}|}$$

Now we apply LSS simultaneously to M and M′.

For M, we pick n schedulers uniformly $\sigma_1, \ldots, \sigma_n \sim Unif(Sch_{\mathsf{M}})$, compute each scheduler's safety probability $p_i := Pr^{\sigma_i}_{\mathsf{M}}(\psi)$ and return the smallest safety probability: $min_{i=1,\ldots,n}(p_i)$.

For M′, we convert each σ_i into its corresponding $\sigma'_i \in Sch_{\mathsf{M}'}$, compute its safety probability $p'_i := Pr^{\sigma'_i}_{\mathsf{M}'}(\psi)$ and return the minimum one: $min_{i=1,\ldots,n}(p'_i)$.

From earlier, we know that $p_i \geq p'_i \; \forall i = 1, \ldots, n$. It follows that $min_{i=1,\ldots,n}(p_i) \geq min_{i=1,\ldots,n}(p'_i)$. So with this shared sample space of schedulers we always get a larger value from running LSS on M than M′. Thus, the probability mass of $Pr^{LSS}_{\mathsf{M}}(\psi)$ lies to the right of that of $Pr^{LSS}_{\mathsf{M}'}(\psi)$. Thus, $Pr^{LSS}_{\mathsf{M}}(\psi) \geq_{FSD} Pr^{LSS}_{\mathsf{M}'}(\psi)$. □

The model-wide LSS MoS trimming procedure works by applying the trimming procedure in Definition 5.5 to all states of the model.

Corollary 5.8. *The model-wide LSS MoS trimming procedure is conservative under Definition 4.2.*

5.2.1 LSS Data Efficiency. As LSS is a sampling based method, it has a trade-off between sample count and accuracy. Sampling more schedulers produces more accurate LSS results. Intuitively, the larger the set of schedulers for a PA M, the more schedulers LSS should sample to get a reasonable probability estimate of $Pr^{min}_{\mathsf{M}}(\psi)$. Thus, reducing the number of schedulers when constructing M′ allows LSS to sample fewer schedulers. In our evaluation, we show that our trimmings allow for better LSS results while using 10x fewer schedulers than without trimming.

6 EVALUATION AND RESULTS

Our experimental evaluation had two goals: compare the scalability/data efficiency of our abstractions and explore the extent to which Definition 4.2 holds on small models[3]. We perform evaluation on simulations of the two case studies described in Section 2. For PMC we use the PRISM model checker [20] and for LSS we use the MODEST model checker [2]. The code and models used can be found at: `https://github.com/earnedkibbles58/ICCPS_2022_MoS`.

6.1 Setup: Data Collection and Modeling

6.1.1 AEBS. First, we setup the AEBS controller in the self-driving car simulator CARLA [9], using a red Toyota Prius as the obstacle. We performed two data collections: (i) constant-v runs to gather image-distance data to construct PA M_{per}, (ii) AEBS-controlled runs to approximate the true value of Pr_{nocol}.

For the former collection, we ran 500 simulations starting from 200m and obtained 180500 pairs (d, ©). Model M_{per} contains the probabilities of detection conditioned on the 10m-wide distance bin and the past 3 low-level outcomes. For the latter collection, we ran 300 simulations of emergency braking starting at 160m and 20m/s, with YoloNet perception and a 3-wide majority filter on top. 20 runs out of 300 resulted in a crash, yielding a $[0.04, 0.1]$ 95% confidence interval (CI) for the true collision chance (i.e., $1 - Pr_{nocol}$).

6.1.2 Water Tank. To derive a probabilistic model of the simulated perception error with $EW = 6$, at each water level (from 1 to TS) we ran 100 trials of the perception and recorded the counts of perception errors into 15 bins, 13 of which were of width 1, representing errors from -6 to 6. Errors above 6 or below -6 got their own bins as well, which represented the perception giving an output of 0 or TS, respectively. To summarize, the model of M_{per} was a categorical distribution over the perception error with the following bins:

$$\{0, wl - 6, wl - 5, \ldots, wl + 5, wl + 6, TS\}$$

6.2 PMC Scalability and Accuracy

For each case study, we consider two different models: (i) M_{int} from Section 3.1, (ii) $\mathsf{M}^{PMC}_{\text{tt}}$, which we form by applying the MoS-driven trimming procedures from Definition 5.1.

6.2.1 AEBS. One experiment evaluates the accuracy-scalability tradeoffs for a fixed initial state (160m, 20m/s) and differing distance interval sizes of the models. Figure 3 shows that $\mathsf{M}^{PMC}_{\text{tt}}$ outperforms M_{int} in terms of run time while maintaining empirical conservatism. This shows that our MoS-enabled trimming shows significant speed ups while maintaining empirical conservatism, as

[3]Large models have too many schedulers to enumerate them.

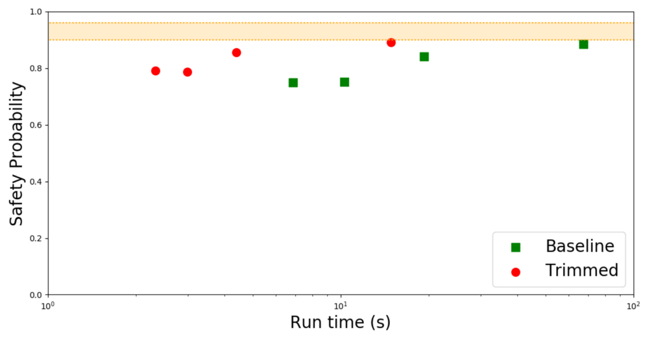


Figure 3: AEBS run times and safety probabilities for M_{int} (green squares) and $\mathsf{M}_{\text{tt}}^{PMC}$ (red dots), and 95% CI for the simulated collision chance (orange band). Each dot represents a different interval size for M_{int}.

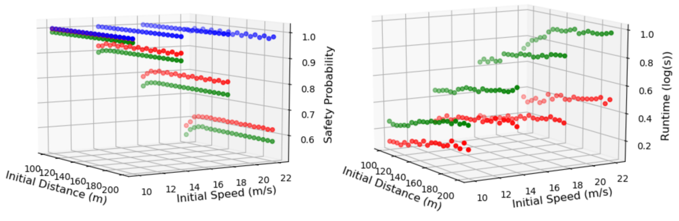


Figure 4: Comparison of PMC safety probabilities (left) and run times (right) for models across different initial distances and velocities of M_{int} (green), $\mathsf{M}_{\text{tt}}^{PMC}$ (red) and $\mathsf{M}_{\text{ttneg}}^{PMC}$ (blue) models of the AEBS example.

it returns a lower safety chance than the simulations, for a range of model hyperparameters.

In the other experiment we fixed the distance interval size and varied the initial conditions, shown in Figure 4. First, M_{int} scales very poorly. It takes an order of magnitude longer to run than $\mathsf{M}_{\text{tt}}^{PMC}$. Second, $\mathsf{M}_{\text{tt}}^{PMC}$ returns similar safety probabilities to M_{int}. Third, we computed a model using the negations of $\succeq_d$ and $\succeq_v$, which we denote as $\mathsf{M}_{\text{ttneg}}^{PMC}$. This model returns very large safety probabilities. This is further evidence that $\succeq_d$ and $\succeq_v$ are fairly accurate, as taking their negations produces overly safe results.

6.2.2 Water Tank. We first fixed the initial water levels to be 50 (exactly half full) and varied the water level interval size of the models. Figure 5 shows that the $\mathsf{M}_{\text{tt}}^{PMC}$ outperforms M_{int} in terms of run time while returning similar safety probabilities over a range of hyperparameters.

We then fixed the water level interval size and varied the initial conditions, shown in Figure 6. First, M_{int} scales very poorly, with several models taking upwards of an hour to verify. $\mathsf{M}_{\text{tt}}^{PMC}$ speeds this by up to 2 orders of magnitude. Second, $\mathsf{M}_{\text{tt}}^{PMC}$ returns very similar safety probabilities to M_{int}. Third, $\mathsf{M}_{\text{ttneg}}$ returns really high safety probabilities, which further indicates that $\succeq_w$ is an accurate assumption.

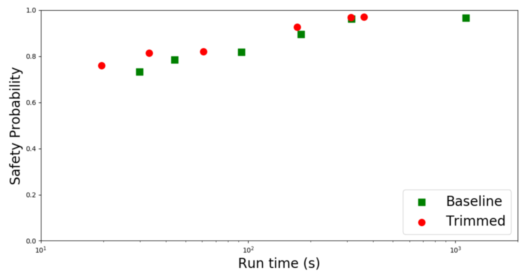


Figure 5: Water tank PMC run times and overflow/underflow probabilities for M_{int} (green square) and $\mathsf{M}_{\text{tt}}^{PMC}$ (red dot). Each dot is for a set of values of a model's hyperparameters.

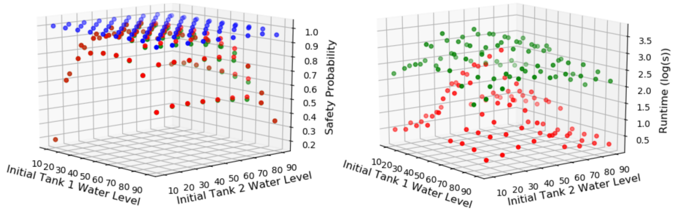


Figure 6: Comparison of PMC safety probabilities (left) and run times (right) of M_{int} (green), $\mathsf{M}_{\text{tt}}^{PMC}$ (red) and $\mathsf{M}_{\text{ttneg}}^{PMC}$ (blue) models for the water tank example.

Initial Condition (d,v)	M_{int} and 10 Schedulers	$\mathsf{M}_{\text{tt}}^{LSS}$ and 10 Schedulers	$\mathsf{M}_{\text{tt}}^{LSS}$ and 1 Scheduler
(130,14)	0.991	0.919	0.962
(130,18)	0.962	0.802	0.831
(160,14)	0.993	0.919	0.943
(160,18)	0.966	0.802	0.846

Table 1: LSS Safety Probabilities for the AEBS system.

6.3 LSS Data Efficiency

For each case study, we consider two different models: (i) M_{int} from Section 3.1, (ii) $\mathsf{M}_{\text{tt}}^{LSS}$, which we form by applying the MoS-driven trimming procedure from Definition 5.5.

We kept the same distance and water level interval sizes as before and ran LSS on a smaller range of initial conditions for both abstractions. The safety probability of each scheduler was computed to an absolute error of 0.05 with 0.8 confidence. We ran LSS using 10 scheduler samples on both M_{int} and $\mathsf{M}_{\text{tt}}^{LSS}$. In addition, we ran LSS using 1 scheduler sample on $\mathsf{M}_{\text{tt}}^{LSS}$. For all cases, we ran 10 different trials of LSS and averaged the results. The AEBS results are shown in Table 1 and Table 2 and the water tank results are shown in Figure 7. The $\mathsf{M}_{\text{tt}}^{LSS}$ models give more conservative safety probabilities with 10x fewer schedulers and, hence, 10x smaller run times. This is because all the extra non-determinism in the M_{int} models dilutes the scheduler sampling. Note that $\mathsf{M}_{\text{tt}}^{LSS}$ took around the same time as M_{int} when using an equal number of samples.

Initial Condition (d,v)	M_{int} and 10 Schedulers	$\mathsf{M}_{\text{tt}}^{LSS}$ and 10 Schedulers	$\mathsf{M}_{\text{tt}}^{LSS}$ and 1 Scheduler
(130,14)	171	201	15
(130,18)	603	179	11
(160,14)	203	250	17
(160,18)	608	228	12

Table 2: LSS Run Times (sec) for the AEBS system.

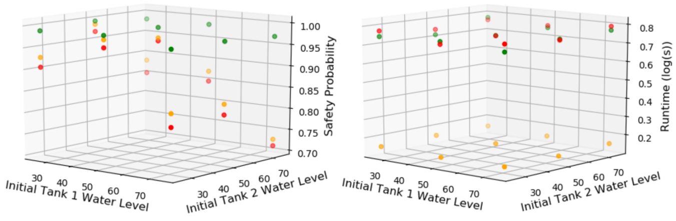


Figure 7: LSS safety probabilities (left) and run times (right) of water tank models across different initial water levels for M_{int} and 10 schedulers (green), $\mathsf{M}_{\text{tt}}^{LSS}$ and 10 schedulers (red) and $\mathsf{M}_{\text{tt}}^{LSS}$ and 1 scheduler (orange).

6.4 Validating MoS Assumptions

We generated small M_{int} models for the AEBS and water tanks model and enumerated every scheduler of both. We computed the proportion of schedulers for which Definition 4.2 held for each pair of states involved in the MoS trimming, thus quantifying the extent to which our MoS assumptions held on these models.

6.4.1 AEBS. We kept the same discretization parameters as before and used an initial distance of 9m and speed of 1.2m/s. M_{int} had 11664 schedulers and a minimum safety chance of 0.986. $\mathsf{M}_{\text{tt}}^{PMC}$ had a safety chance of 0.990, so our MoS-trimming is not conservative and the conjunction of $\succeq_d$ and $\succeq_v$ does not hold. However, we want to examine how close this conjunction is to holding. So we computed each state pair involved in the MoS-trimming procedure detailed in Section 5.1, of which there were 17. Consider one such pair (s_1, s_2), with $s_1 \succeq_{MoS}^{PMC} s_2$. We then computed the safety probability of the model when starting in both s_1 and s_2 for every scheduler of the model and computed the proportion of schedulers for which s_1 gave a higher safety chance than s_2. This amounts to computing $P_{\sigma\sim Sch_{\mathsf{M}}}\left(Pr^{\sigma}_{\mathsf{M}(s_1)}(\psi_{nocol}) \geq Pr^{\sigma}_{\mathsf{M}(s_2)}(\psi_{nocol})\right)$.[4] We denote this probability as p_{s_1,s_2}.

We applied this procedure to each of the 17 trimmed state pairs. For 15 of them, $p_{s_1,s_2} = 1$. The other two had values of 0.5 and 0.656, respectively. So $\succeq_d$ and $\succeq_v$ hold perfectly for the majority of the state pairs. But for the pairs where it didn't hold it was not very close to holding. This is most likely due to those two state pairs being near the AEBS control boundaries and echoes the intuitions from the counterexamples, which is that $\succeq_d$ and $\succeq_v$ do not hold near control and perception boundaries.

6.4.2 Water Tank. We used a model of a single water tank with a max water level of 30 and the same T and water level interval size as before. M_{int} had 1024 schedulers and a minimum safety chance of 0.261. $\mathsf{M}_{\text{tt}}^{PMC}$ had a safety chance of 0.303. So our MoS-trimming is not conservative and $\succeq_w$ does not hold. However, we want to examine how close $\succeq_w$ is to holding. So we computed each state pair involved in the MoS-trimming procedure from Section 5.1 used to convert M_{int} into $\mathsf{M}_{\text{tt}}^{PMC}$, of which there were 5. Consider one such pair (s_1, s_2), with $s_1 \succeq_{MoS}^{PMC} s_2$. We then computed the safety probability of the model when starting in both s_1 and s_2 for every scheduler of the model and calculated the proportion of schedulers for which s_1 gave a higher safety chance than s_2. This amounts to computing $P_{\sigma\sim Sch_{\mathsf{M}}}\left(Pr^{\sigma}_{\mathsf{M}(s_1)}(\psi_{flow}) \geq Pr^{\sigma}_{\mathsf{M}(s_2)}(\psi_{flow})\right)$.

[4]If $\succeq_w$ held, then this proportion would be 1.

State s_1	State s_2	p_{s_1,s_2}
[5, 10]	[0, 5]	1.0
[10, 15]	[5, 10]	0.934
[15, 20]	[10, 15]	0.813
[15, 20]	[20, 25]	0.881
[20, 25]	[25, 30]	1.0

Table 3: Probabilities of $\succeq_w$ holding over trimmed state pairs of M_{int}.

We applied this process for each of the 5 state pairs and the results are shown in Table 3. This shows that although our MoS-trimming is not conservative, our MoS assumption $\succeq_w$ holds for most schedulers and state pairs. We see that MoS holds over more schedulers near the extremes of the water tank. This also demonstrates why the MoS-trimming is so conservative for LSS, since most of the trimmed schedulers are in fact overly optimistic.

7 RELATED WORK

Many works have applied statistical analysis methods to probabilistic systems with non-determinism. As previously mentioned, [24] introduced LSS and improved upon it with smart sampling [11]. These LSS techniques have been implemented in the MODEST model checker [16]. These methods have been applied to the analysis of timed probabilistic models in [6, 17] and rare event detection [1]. Reinforcement learning techniques have also been employed to find min schedulers of markov decision processes [18].

Previous works have addressed compositional reasoning for probabilistic models, such as extending notions of simulation and bisimulation to PAs [15, 29, 30]. These relations are preserved under parallel composition, but they are too fine-grained for compositional verification. Another approach uses the notion of trace distribution inclusion (distributions over traces of automata actions) for compositional reasoning [28], but this relation is not preserved under parallel composition. Other works target more restrictive modeling formalisms than PAs, such as reactive modules [7] and probabilistic I/O systems [4]. Another work [14] automatically generates assumptions for composition. However, it forbids the composed model from having nondeterministic transitions, which does not work with the abstractions presented in this paper.

The most conceptually similar topic to our MoS assumptions is *sensitivity analysis* of dynamical systems, which aims to quantify how changes in system initial conditions affect system traces [8].

Reasoning about the safety of autonomous vehicles has been studied from other perspectives as well. The Responsibility-Sensitive Safety (RSS) model [19] uses physics-based safety analysis for provable safety in different driving scenarios for autonomous vehicles.

RSS does not consider ML-based perception models in the closed loop of their autonomous systems. Finally, researchers have investigated the use of falsification to find perception outputs that cause unsafe system behaviors [10].

8 CONCLUSION

This paper introduced an abstraction approach for scalable probabilistic model checking and data efficient lightweight scheduler sampling of probabilistic automata. Specifically, we use intuitive assumptions of monotonic safety to remove non-determinism from PAs. When such assumptions hold, they lead to provably conservative abstractions. Even when these assumptions do not perfectly hold, MoS-based simplifications remain empirically conservative. We demonstrated these MoS-driven trimmings on case studies of a self-driving car and a water tank. Future work includes analytically bounding conservatism loss from MoS violations, extending the MoS trimming techniques to distributions of states, and exploring how our notion of MoS relate to sensitivity analysis for dynamical systems. More generally, we see the development of abstractions suitable for LSS as a promising area to explore.

ACKNOWLEDGMENTS

The authors thank Ramneet Kaur for helping to develop the AEBS case study. This work was supported in part by ARO W911NF-20-1-0080, AFRL and DARPA FA8750-18-C-0090, and ONR N00014-20-1-2744. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the Air Force Research Laboratory (AFRL), the Army Research Office (ARO), the Defense Advanced Research Projects Agency (DARPA), the Office of Naval Research (ONR) or the Department of Defense, or the United States Government.

REFERENCES

[1] Carlos Budde, Arnd Hartmanns, Pedro D'Argenio, and Sean Sedwards. 2020. An efficient statistical model checker for nondeterminism and rare events. *International Journal on Software Tools for Technology Transfer* 22 (12 2020), 1–22.
[2] Carlos E. Budde, Pedro R. D'Argenio, Arnd Hartmanns, and Sean Sedwards. 2018. A Statistical Model Checker for Nondeterminism and Rare Events. In *Tools and Algorithms for the Construction and Analysis of Systems*, Dirk Beyer and Marieke Huisman (Eds.). Springer International Publishing, Cham, 340–358.
[3] Simon Burton, Lydia Gauerhof, and Christian Heinzemann. 2017. Making the Case for Safety of Machine Learning in Highly Automated Driving. In *Computer Safety, Reliability, and Security (Lecture Notes in Computer Science)*. Springer International Publishing, Cham, 5–16.
[4] Ling Cheung, Nancy Lynch, Roberto Segala, and Frits Vaandrager. 2005. Switched Probabilistic I/O Automata. In *Theoretical Aspects of Computing - ICTAC 2004*, Zhiming Liu and Keijiro Araki (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 494–510.
[5] Ajay Chhokra, Nagabhushan Mahadevan, Abhishek Dubey, and Gabor Karsai. 2020. Qualitative Fault Modeling in Safety Critical Cyber Physical Systems. In *Proceedings of the 12th System Analysis and Modelling Conference* (Virtual Event, Canada) *(SAM '20)*. Association for Computing Machinery, New York, NY, USA, 128–137.
[6] Pedro D'Argenio, Arnd Hartmanns, Axel Legay, and Sean Sedwards. 2016. Statistical Approximation of Optimal Schedulers for Probabilistic Timed Automata. 99–114.
[7] Luca de Alfaro, Thomas A. Henzinger, and Ranjit Jhala. 2001. Compositional Methods for Probabilistic Systems. In *CONCUR 2001 — Concurrency Theory*, Kim G. Larsen and Mogens Nielsen (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 351–365.
[8] Alexandre Donzé and Oded Maler. 2007. Systematic simulation using sensitivity analysis. In *International Workshop on Hybrid Systems: Computation and Control*. Springer, 174–189.
[9] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. 2017. CARLA: An Open Urban Driving Simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*. 1–16.
[10] Tommaso Dreossi, Alexandre Donze, and Sanjit A Seshia. 2017. Compositional Falsification of Cyber-Physical Systems with Machine Learning Components. (2017).
[11] Pedro D'Argenio, Axel Legay, Sean Sedwards, and Louis-Marie Traonouez. 2015. Smart sampling for lightweight verification of Markov decision processes. *International Journal on Software Tools for Technology Transfer* 17, 4 (2015), 469–484.
[12] Georgios E Fainekos and George J Pappas. 2006. Robustness of temporal logic specifications. In *Formal approaches to software testing and runtime verification*. Springer, 178–192.
[13] J. M. Faria. 2017. Non-determinism and Failure Modes in Machine Learning. In *2017 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*. 310–316.
[14] L. Feng, T. Han, M. Kwiatkowska, and D. Parker. 2011. Learning-based Compositional Verification for Synchronous Probabilistic Systems. In *Proc. 9th International Symposium on Automated Technology for Verification and Analysis (ATVA'11) (LNCS, Vol. 6996)*. Springer, 511–521.
[15] Daniel Gebler and Simone Tini. 2013. Compositionality of Approximate Bisimulation for Probabilistic Systems. *Electronic Proceedings in Theoretical Computer Science* 120 (Jul 2013), 32–46.
[16] Arnd Hartmanns and Holger Hermanns. 2014. The Modest Toolset: An integrated environment for quantitative modelling and verification. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 593–598.
[17] Arnd Hartmanns, Sean Sedwards, and Pedro R. D'Argenio. 2017. Efficient simulation-based verification of probabilistic timed automata. In *2017 Winter Simulation Conference (WSC)*. 1419–1430.
[18] David Henriques, João G. Martins, Paolo Zuliani, André Platzer, and Edmund M. Clarke. 2012. Statistical Model Checking for Markov Decision Processes. In *2012 Ninth International Conference on Quantitative Evaluation of Systems*. 84–93.
[19] Philip Koopman, Beth Osyk, and Jack Weast. 2019. Autonomous vehicles meet the physical world: RSS, variability, uncertainty, and proving safety. In *International Conference on Computer Safety, Reliability, and Security*. Springer, 245–253.
[20] M. Kwiatkowska, G. Norman, and D. Parker. 2011. PRISM 4.0: Verification of Probabilistic Real-time Systems. In *Proc. 23rd International Conference on Computer Aided Verification (CAV'11) (LNCS, Vol. 6806)*, G. Gopalakrishnan and S. Qadeer (Eds.). Springer, 585–591.
[21] Marta Kwiatkowska, Gethin Norman, David Parker, and Hongyang Qu. 2010. Assume-Guarantee verification for probabilistic systems. In *Proceedings of the 16th international conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'10)*. Springer-Verlag, Paphos, Cyprus, 23–37.
[22] Marta Kwiatkowska, Gethin Norman, David Parker, and Hongyang Qu. 2013. Compositional probabilistic verification through multi-objective model checking. *Information and Computation* 232 (2013), 38 – 65.
[23] Taeyoung Lee, Kyongsu Yi, Jangseop Kim, and Jaewan Lee. 2011. Development and Evaluations of Advanced Emergency Braking System Algorithm for the Commercial Vehicle.
[24] Axel Legay, Sean Sedwards, and Louis-Marie Traonouez. 2014. Scalable verification of Markov decision processes. In *International Conference on Software Engineering and Formal Methods*. Springer, 350–362.
[25] Chunsheng Liu, Yu Guo, Shuang Li, and Faliang Chang. 2019. ACF based region proposal extraction for YOLOv3 network towards high-performance cyclist detection in high resolution images. *Sensors* 19, 12 (2019), 2671.
[26] Amir Pnueli. 1977. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science, 1977*. 46–57.
[27] Joseph Redmon and Anelia Angelova. 2015. Real-time grasp detection using convolutional neural networks. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 1316–1322.
[28] Roberto Segala. 1995. A Compositional Trace-Based Semantics for Probabilistic Automata. In *Proceedings of the 6th International Conference on Concurrency Theory (CONCUR '95)*. Springer-Verlag, Berlin, Heidelberg, 234–248.
[29] R. Segala. 1996. *Modeling and Verification of Randomized Distributed Real -Time Systems.* Technical Report. USA.
[30] Roberto Segala and Nancy A. Lynch. 1994. Probabilistic Simulations for Probabilistic Processes. In *Proceedings of the Concurrency Theory (CONCUR '94)*. Springer-Verlag, Berlin, Heidelberg, 481–496.
[31] Roman Yampolskiy. 2018. Predicting future AI failures from historic examples. *Foresight* 21 (2018).

A APPENDIX

A.1 Counterexamples to Monotonic Safety in AEBS

In all counterexamples, we set $\tau = 1s$ and window size $N_F = 1$. We adopt the notation $Pr(\psi_{nocol} \mid d, s)$ as a shorthand for $Pr_{\mathsf{M}}(d, s)(\psi_{nocol})$.

Counterexample 1. *Assumption $\succeq_d$ does not hold in the AEBS with one BP and distance-dependent perception for shift $\Delta = (1m, 0)$ when $B = 10m/s^2, v_0 = 11m/s, d_0 = 13m$, and $Pr_{det}(d) = 1 - \lceil d \rceil/20$.*

Proof. We set $\bar{s}_b = (13, 11)$ and $\bar{s}_a = (14, 11)$. Then we calculate as follows.

$$Pr_{\mathsf{M}(\bar{s}_a)}(\psi_{nocol}) =$$
$$Pr_{det}(14) * Pr(\psi_{nocol}|3,1) + (1 - Pr_{det}(14)) * \underbrace{Pr(\psi_{nocol}|3,11)}_{=0} =$$
$$Pr_{det}(14) * (Pr_{det}(3) * \underbrace{Pr(\psi_{nocol}|2,0)}_{=1}$$
$$+ (1 - Pr_{det}(3)) * \underbrace{Pr(\psi_{nocol}|2,1))}_{=Pr_{det}(2)} =$$
$$Pr_{det}(14) * (Pr_{det}(3) + (1 - Pr_{det}(3)) * Pr_{det}(2)) =$$
$$Pr_{det}(14) * (Pr_{det}(3) + Pr_{det}(2) - Pr_{det}(3) * Pr_{det}(2)).$$

$$Pr_{\mathsf{M}(\bar{s}_b)}(\psi_{nocol}) =$$
$$Pr_{det}(13) * Pr(safe|2,1) + (1 - Pr_{det}(13)) * \underbrace{Pr(safe|2,11)}_{=0} =$$
$$Pr_{det}(13) * (Pr_{det}(2) * \underbrace{Pr(safe|1,0)}_{=1} +$$
$$(1 - Pr_{det}(2)) * \underbrace{Pr(safe|1,1))}_{=0} =$$
$$Pr_{det}(13) * Pr_{det}(2).$$

If we set $Pr_{det}(d) = 1 - \lceil d \rceil/20$ (i.e., the detection probability grows linearly from 0% at 20 meters to 100% at 0 meters), then the above falsifies MoS:

$$Pr_{\mathsf{M}(\bar{s}_b)}(\psi_{nocol}) = 0.315 > 0.2955 = Pr_{\mathsf{M}(\bar{s}_a)}(\psi_{nocol})$$

P.S. This inequality is reversed for less steep detection curves like $1 - \lceil d \rceil/40$. □

The second counterexample shows that going at a faster speed may be beneficial because it leads to closer distances, which result in stronger braking.

Counterexample 2. *Assumption $\succeq_v$ does not hold in the AEBS with multiple BPs and distance-independent perception for shift $\Delta = (0, -1m/s)$ when $B(d) = [10m/s^2$ if $d \leq 11m; 3m/s^2$ otherwise$]$, $v_0 = 9m/s, d_0 = 20m$, and $Pr_{det} = 0.5$.*

Proof. We set $\bar{s}_b = (20, 9)$ and $\bar{s}_a = (20, 8)$. Then we calculate as follows:

$$Pr_{\mathsf{M}(\bar{s}_a)}(\psi_{nocol}) =$$
$$Pr_{det} * Pr(\psi_{nocol}|11,6) + (1 - Pr_{det}) * Pr(\psi_{nocol}|11,9) =$$
$$Pr_{det} * (Pr_{det} * \underbrace{Pr(\psi_{nocol}|5,0)}_{=1} + (1 - Pr_{det}) * \underbrace{Pr(\psi_{nocol}|5,6))}_{=0} +$$
$$(1 - Pr_{det}) * (Pr_{det} * \underbrace{Pr(\psi_{nocol}|2,0)}_{=1}$$
$$+ (1 - Pr_{det}) * \underbrace{Pr(\psi_{nocol}|2,9)}_{=0} =$$
$$(Pr_{det})^2 + (1 - Pr_{det}) * Pr_{det} = Pr_{det}$$

$$Pr_{\mathsf{M}(\bar{s}_b)}(\psi_{nocol}) =$$
$$Pr_{det} * Pr(\psi_{nocol}|12,5) + (1 - Pr_{det}) * Pr(\psi_{nocol}|12,8) =$$
$$Pr_{det} * (Pr_{det} * Pr(\psi_{nocol}|7,2) + (1 - Pr_{det}) * \underbrace{Pr(\psi_{nocol}|7,5))}_{=Pr_{det}} +$$
$$(1 - Pr_{det}) * (Pr_{det} * \underbrace{Pr(\psi_{nocol}|4,5)}_{=0}$$
$$+ (1 - Pr_{det}) * \underbrace{Pr(\psi_{nocol}|4,8)}_{=0} = \cdots =$$
$$Pr^2_{det}(1 + 2Pr_{det} - 3Pr^2_{det} + Pr^3_{det})$$

It can be shown that for any $0 < Pr_{det} < 1$, $Pr_{det} > Pr^2_{det}(1 + 2Pr_{det} - 3Pr^2_{det} + Pr^3_{det})$. For example, for $Pr_{det} = 0.5$, $0.5 > 0.34375$. Thus, the above falsifies MoS:

$$Pr_{\mathsf{M}(\bar{s}_b)}(\psi_{nocol}) > Pr_{\mathsf{M}(\bar{s}_a)}(\psi_{nocol})$$

□

A.2 Counterexamples to Monotonic Safety in Water Tanks

Counterexample 3. *Assumption $\succeq_w$ does not hold for a water tank system with $J = 1, TS = 100, out = 3, in = 40, T = 4, Pr(\hat{w} = 0) = 0.4$, and $Pr(\hat{w} = TS) = 0.6, w = 10$, and $\Delta_w = 30$.*

Proof. For state $\bar{s}_b = 10$, we compute the chance of underflow as $0.6^4 = 0.1296$, and the chance of overflow is the chance of 3 or 4 successes in Binomial distribution $Bi(4, 0.4)$, which is 0.1792. So $Pr_{\mathsf{M}(\bar{s}_b)}(\psi_{flow}) = 0.6912$. For state $\bar{s}_a = 40$, the chance of underflow is 0. The chance of overflow is the chance of 2+ successes in Binomial distribution from the binomial distribution $Bi(4, 0.4)$, so $Pr_{\mathsf{M}(\bar{s}_a)}(\psi_{flow}) = 0.4752$. Thus, $\bar{s}_a \succeq_w \bar{s}_b$, but $Pr_{\mathsf{M}(\bar{s}_a)}(\psi_{flow}) < Pr_{\mathsf{M}(\bar{s}_b)}(\psi_{flow})$. □

A.3 Proof of Trace Decomposition Lemma

Lemma A.1 (Satisfaction Probability by Trace Decomposition). *Let M be a PA. Assume that there are no infinite paths through M. Let σ be a scheduler of M. Then*

$$Pr^{\sigma}_{\mathsf{M}}(\psi) = Pr^{\sigma}_{\mathsf{M}}(\psi \& \neg \diamond s) + Pr^{\sigma}_{\mathsf{M}}(\diamond s) Pr^{\sigma}_{\mathsf{M}(s)}(\psi)$$

Proof. We can split up the traces through M by taking the ones that don't pass through state s and the ones that do, as these are two disjoint sets. So

$$Pr^{\sigma}_{\mathsf{M}}(\psi) = Pr^{\sigma}_{\mathsf{M}}(\psi \& \neg \diamond s) + Pr^{\sigma}_{\mathsf{M}}(\psi \& \diamond s) \tag{7}$$

Now we examine the $Pr^{\sigma}_{\mathsf{M}}(\psi \& \diamond s)$ term. Note that ψ is a safety property, so it can be expressed as $\square\neg\rho$. First, we decompose this term using the chain rule for probabilities:

$$Pr^{\sigma}_{\mathsf{M}}(\psi \& \diamond s) = Pr^{\sigma}_{\mathsf{M}}(\diamond s) Pr^{\sigma}_{\mathsf{M}}(\psi | \diamond s) \tag{8}$$

We now focus on the $Pr^{\sigma}_{\mathsf{M}}(\psi | \diamond s)$. This asks for the measure of the traces which satisfy ψ among the traces which pass through s. But by the construction of M every state which falsifies ψ (i.e. every state for which ρ is true) is a sink state, so it is impossible for a trace through M to falsify ψ and then pass through s. Thus we have that:

$$Pr^{\sigma}_{\mathsf{M}}(\psi | \diamond s) = Pr^{\sigma}_{\mathsf{M}(s)}(\psi) \tag{9}$$

Substitute Equation (8) and Equation (9) into (7) to complete the proof.

□