Nonlinear Analysis: Hybrid Systems 23 (2017) 230–253

Contents lists available at ScienceDirect

Nonlinear Analysis: Hybrid Systems

journal homepage: www.elsevier.com/locate/nahs

Temporal logic control for stochastic linear systems using abstraction refinement of probabilistic games☆

Mária Svoreňová [a,b,*], Jan Křetínský [c], Martin Chmelík [d], Krishnendu Chatterjee [d], Ivana Černá [a], Calin Belta [e]

[a] Faculty of Informatics, Masaryk University, Botanická 68a, 60200 Brno, Czech Republic
[b] Department of Computer Science, University of Oxford, Wolfson Building, Parks Road, Oxford, OX1 3QD, United Kingdom
[c] Institut für Informatik, Technische Universität München, Boltzmannstr. 3, 85748 Garching bei München, Germany
[d] IST Austria, Am Campus 1, A-3400 Klosterneuburg, Austria
[e] Department of Mechanical Engineering, Boston University, 110 Cummington Mall, 02215 Boston, MA, USA

ARTICLE INFO

Article history:
Available online 1 June 2016

Keywords:
Control
Linear stochastic system
Temporal logic
Abstraction refinement
Games

ABSTRACT

We consider the problem of computing the set of initial states of a dynamical system such that there exists a control strategy to ensure that the trajectories satisfy a temporal logic specification with probability 1 (almost-surely). We focus on discrete-time, stochastic linear dynamics and specifications given as formulas of the Generalized Reactivity(1) fragment of Linear Temporal Logic over linear predicates in the states of the system. We propose a solution based on iterative abstraction-refinement, and turn-based 2-player probabilistic games. While the theoretical guarantee of our algorithm after any finite number of iterations is only a partial solution, we show that if our algorithm terminates, then the result is the set of all satisfying initial states. Moreover, for any (partial) solution our algorithm synthesizes witness control strategies to ensure almost-sure satisfaction of the temporal logic specification. While the proposed algorithm guarantees progress and soundness in every iteration, it is computationally demanding. We offer an alternative, more efficient solution for the reachability properties that decomposes the problem into a series of smaller problems of the same type. All algorithms are demonstrated on an illustrative case study.

1. Introduction

The formal verification problem, in which the goal is to check whether behaviors of a finite model satisfy a correctness specification, received a lot of attention during the past thirty years [1,2]. In contrast, in the synthesis problem the goal is to synthesize or control a finite system from a temporal logic specification. While the synthesis problem also has a long

☆ This work was partially supported by Czech Science Foundation grant 15-17564S, People Programme (Marie Curie Actions) of the European Union's Seventh Framework Programme (FP7/2007–2013) under REA grant agreement 291734, European Research Council (ERC) grant 267989 (QUAREM) and Start grant (279307: Graph Games), Austrian Science Fund (FWF) grants S11402-N23 (RiSE), P23499-N23 and S11407-N23 (RiSE), Czech Ministry of Education Youth and Sports grant LH11065, Czech Science Foundation Grant No.15-08772S, and National Science Foundation (NSF) grants CMMI-1400167 and CNS-1035588.

* Corresponding author.
E-mail addresses: maria.svorenova@cs.ox.ac.uk (M. Svoreňová), jan.kretinsky@tum.de (J. Křetínský), martin.chmelik@ist.ac.at (M. Chmelík), kchatterjee@ist.ac.at (K. Chatterjee), cerna@muni.cz (I. Černá), cbelta@bu.edu (C. Belta).

http://dx.doi.org/10.1016/j.nahs.2016.04.006
1751-570X/

tradition [3–5], it has gained significant attention in formal methods only recently. For example, these techniques are being deployed in control and path planning. Model checking techniques can be adapted to synthesize (optimal) controllers for deterministic finite systems [6,7], Büchi and Rabin games can be reformulated as control strategies for nondeterministic systems [8,9], and probabilistic games can be used to compute controllers for finite probabilistic systems such as Markov decision processes [10,11].

With the widespread integration of physical and digital components in cyber physical systems, and the safety and security requirements in such systems, there is an increased need for the development of formal methods techniques for systems with infinite state spaces, normally modeled as difference or differential equations. Most of the works in the area use partitions and simulation/bisimulation relations to construct a finite abstraction of the system, followed by verification or control of the abstraction. Existing results showed that such approaches are feasible for discrete and continuous time linear systems [12,13]. With some added conservatism, more complicated dynamics and stochastic dynamics can also be handled [14,15].

In this work, we focus on the problem of finding the set of initial states of a dynamic system from which a given constraint can be satisfied, and synthesizing the corresponding witness control strategies. In particular, we consider discrete-time continuous-domain linear stochastic dynamics with constraints given as formulas of the Generalized Reactivity(1) (GR(1)) [16] fragment of Linear Temporal Logic (LTL) over linear predicates in the states of the system. The GR(1) fragment offers polynomial computational complexity as compared to the doubly exponential one of general LTL, while being expressive enough to describe most of the usually considered temporal properties [16]. We require the formula to be satisfied almost-surely, *i.e.,* with probability 1. Note that almost-sure analysis is very different from robust analysis often considered in control of systems with disturbances, where the formula must hold under all possible disturbances. To illustrate the difference, consider a simple example of a faulty messaging protocol that, on an attempt to send a message, fails or succeeds, both with probability $\frac{1}{2}$. In robust analysis, it is not true that the protocol eventually sends a message as it may always fail. However, in almost-sure analysis, it is true that a message is eventually sent with probability 1 since the event of successfully sending a message has non-zero probability in every attempt. In other words, the uncertainty of a system is interpreted as an adversarial choice in robust analysis, whereas in almost-sure analysis, it is interpreted as a probabilistic choice. The almost-sure satisfaction is therefore the strongest probability guarantee one can achieve while accounting for the stochasticity of the dynamics. Moreover, this type of analysis is resistant to changes in the underlying distribution of disturbances as long as the set of disturbances with non-zero probability of occurrence does not change. In the example above, the answers to both robust and almost-sure questions remain the same if both failure and success occur with non-zero probability in every step. This makes the almost-sure analysis a powerful tool particularly in cases, where the distribution over disturbances is not precisely known or where it is approximated from data, which is typically the case in applications. Finally, the computational techniques used in almost-sure analysis for stochastic systems can be simplified comparing to the general, quantitative probabilistic analysis due to the above fact that the exact probabilities of possible disturbances can be abstracted away. The importance of almost-sure analysis of stochastic systems was advocated already in [17].

In our proposed approach, we iteratively construct and refine a discrete abstraction of the system and solve the synthesis problem for the abstract model. The discrete model considered in this work is a turn-based 2-player probabilistic game, also called $2\frac{1}{2}$-player game [18]. Every iteration of our algorithm produces a partial solution given as a partition of the state space into three categories. The first is a set of satisfying initial states together with corresponding witness strategies. The second is a set of non-satisfying initial states, *i.e.,* those from which the system cannot be controlled to satisfy the specification with probability 1. Finally, some parts of the state space may remain undecided due to coarse abstraction. As the abstraction gets more precise, more states are being decided with every iteration of the algorithm. The designed solution is partially correct. That means, we guarantee soundness, *i.e.,* almost sure satisfaction of the formula by all controlled trajectories starting in the satisfying initial set and non-existence of a satisfying control strategy for non-satisfying initial states. On the other hand, completeness is only ensured if the algorithm terminates. In comparison, use of a weaker abstraction model such as 2 player game would lead to the robust analysis discussed above and in this case, there would be no soundness guarantee on the non-satisfying initial states and no completeness guarantees. We provide a practical implementation of the algorithm that ends after a predefined number of iterations.

The original results of this work appeared in [19]. In addition to [19], this paper includes an improved implementation and the theoretical results are extended as follows. While the proposed abstraction-refinement algorithm guarantees soundness and progress in every iteration, it suffers from high computational complexity. To address this issue, we offer a detailed analysis of the problem for the class of reachability properties, *i.e.,* a subclass of GR(1) properties, where the goal is to reach a specific target region within the state space of the system with probability 1. We present three algorithms that can be used to partially or fully solve the problem. Whereas for GR(1) properties, strategies with memory might be necessary for satisfaction, for reachability properties strategies without memory are sufficient. Thus the computation can be simplified by omitting automata construction and employing computationally less demanding polytopic computations over the state and control space of the system. First, we discuss a simple, fast algorithm that can be used to compute the set of all satisfying initial states $\mathcal{X}_{\text{init}}$, but cannot provide any information about the corresponding satisfying strategies. Second, we extend this algorithm to provide a partial information about the satisfying strategies. The two algorithms operate directly on the linear stochastic system and use polytopic operators only. The first algorithm does not involve any construction, analysis or refinement of an abstract model and the second algorithm only constructs (but does not solve) the game once in order to provide the information about satisfying control strategies. Finally, we combine the latter algorithm with the abstraction-

refinement algorithm to obtain a framework that besides computing the set of satisfying initial states also computes the satisfying control strategies. Intuitively, the second algorithm above is used to decompose the original control problem into a series of smaller control problems of the same type that are then solved using the abstraction-refinement algorithm. All presented algorithms are demonstrated and compared on an illustrative example. Theoretically, the combined algorithm has the same computational complexity as the abstraction-refinement. However, we demonstrate that it can lead to much reduced computation times.

The main contribution of our work is twofold. First, we present a sound abstraction-refinement using a $2\frac{1}{2}$-player game for a stochastic system with continuous state and control space. Second, we compute the control strategies for such a system that almost-surely satisfy complex temporal constraints, possibly over infinite time. Abstraction-refinement exists for discrete systems and is often referred to as counterexample-guided abstraction-refinement (CEGAR). It has been developed for both non-deterministic and probabilistic systems [20,21]. In particular, [11,22] use $2\frac{1}{2}$-player games as the abstraction model of the original discrete system. While abstraction-refinement has been also considered for some classes of timed and hybrid systems [23–25], these works consider systems with discrete, finite space of control inputs and analyze only simple temporal properties such as reachability. To the best of our knowledge, the approach presented in this paper is the first attempt to construct abstraction-refinement of stochastic systems with infinite, continuous state and control spaces in the form of $2\frac{1}{2}$-player games. The game theoretic solutions are necessary to determine what needs to be refined, and the dynamics of the linear stochastic systems guide the refinement steps. Thus both game theoretic aspects and the dynamics of the system play a crucial role in the refinement step.

This paper is closely related to [8,26–28]. Our computation of the abstraction is inspired from [8], which introduces a technique for abstraction of discrete-time piece-wise affine systems, *i.e.,* systems with continuous state and control space without stochasticity. The authors in [26] extend the technique to abstraction-refinement and control strategy synthesis for temporal specifications over finite time. An uncontrolled version of the abstraction problem for a stochastic system is considered in [27], where the abstract model is a Markov set-chain. The authors approximate the transition probabilities and evaluate the corresponding error. The abstraction can be constructed with arbitrary precision by parameter tuning. Controllable stochastic dynamics with finite control space is considered in [28], and the authors design an abstraction-refinement, where the abstract model is a bounded-parameter Markov decision process. Upper and lower bounds on transition probabilities are then used to synthesize control strategies that satisfy specifications over finite time in the form of probabilistic Computation Tree Logic formulas. Comparing to [27,28], we extend the results by considering continuous control space and complex temporal specification over (possibly) infinite time. We analyze qualitative, *i.e.,* almost-sure, rather than quantitative properties of the system which eliminate the need to approximate the transition probabilities. The abstraction is instead constructed using polytopic operators and does not contain any error. This consequently allows us to prove soundness of the designed abstraction-refinement.

Finally, linear systems have been studied extensively with respect to the restricted class of reachability properties. Non-stochastic linear systems are investigated, *e.g.,* in [29–31], where the last reference also contains an overview of existing methods. The disturbances in the system are specified by a bounded set rather than a distribution as in our case, and robust analysis is used to approximate the reachable sets. Furthermore, typically no control strategies are computed. Recent results for reachability analysis of linear stochastic systems include [32,33] that focus on control strategy synthesis for quantitative reachability, approximating the transition probabilities and evaluating errors. Other works such as [34,35] quantitatively analyze the system but do not construct control strategies.

The rest of the paper is organized as follows. We give some preliminaries in Section 2 before we formulate the problem and outline the approach in Section 3. The abstraction-refinement algorithm is presented in Section 4 and demonstrated on a case study in Section 5. We follow with the detailed analysis of the reachability problem in Section 6 and use the same case study as above to demonstrate the usability of the proposed algorithms in Section 7. We conclude with final remarks in Section 8. Finally, Appendix A contains the description of the games algorithm used in the abstraction-refinement and Appendix B contains algorithms for computation of all polytopic operators used in this work.

2. Notation and preliminaries

For a non-empty set S, let S^ω, S^* and S^+ denote the set of all infinite, finite and non-empty finite sequences of elements of S, respectively. For $\sigma \in S^+$ and $\rho \in S^\omega$, we use $|\sigma|$ to denote the length of σ, and $\sigma(n)$ and $\rho(n)$ to denote the nth element, for $1 \leq n \leq |\sigma|$ and $n \geq 1$, respectively. For two sets $S_1 \subseteq S^*$, $S_2 \subseteq S^* \cup S^\omega$, we use $S_1 \cdot S_2 = \{s_1 \cdot s_2 \mid s_1 \in S_1, s_2 \in S_2\}$ to denote their concatenation. Finally, for a finite set S, $|S|$ is the cardinality of S, $\mathcal{D}(S)$ is the set of all probability distributions over S and $\{s \in S \mid d(s) > 0\}$ is the support set of $d \in \mathcal{D}(S)$.

2.1. Polytopes

A (convex) polytope $\mathcal{P} \subset \mathbb{R}^N$ is defined as the convex hull of a finite set $X = \{x_i\}_{i \in I} \subset \mathbb{R}^N$, *i.e.,*

$$\mathcal{P} = \text{hull}(X) = \left\{ \sum_{i \in I} \lambda_i x_i \mid \forall i : \lambda_i \in [0, 1], \sum_{i \in I} \lambda_i = 1 \right\}. \tag{1}$$

We use $V(\mathcal{P})$ to denote the vertices of $\mathcal{P}$ that is the minimum set of vectors in $\mathbb{R}^N$ for which $\mathcal{P} = \text{hull}(V(\mathcal{P}))$. Alternatively, a polytope can be defined as an intersection of a finite number of half-spaces in $\mathbb{R}^N$, *i.e.,*

$$\mathcal{P} = \{x \in \mathbb{R}^N \mid H_{\mathcal{P}} x \leq K_{\mathcal{P}}\}, \tag{2}$$

where $H_{\mathcal{P}}, K_{\mathcal{P}}$ are matrices of appropriate sizes. Forms in Eqs. (1) and (2) are referred to as the V-representation and H-representation of polytope $\mathcal{P}$, respectively. A polytope $\mathcal{P} \subset \mathbb{R}^N$ is called full-dimensional if it has at least $N+1$ vertices. In this work, we consider all polytopes to be full-dimensional, *i.e.,* if a polytope is not full-dimensional, we consider it empty.

2.2. Automata and specifications

Definition 1 (*ω-Automata*)**.** A deterministic ω-automaton with Büchi implication (aka one-pair Streett) acceptance condition is a tuple $\mathcal{A} = (Q, \Sigma, \delta, q_0, (E, F))$, where Q is a non-empty finite set of states, Σ is a finite alphabet, $\delta\colon Q \times \Sigma \to Q$ is a deterministic transition function, $q_0 \in Q$ is an initial state, and $(E, F) \subseteq Q \times Q$ defines an acceptance condition.

Given an automaton, every word $w \in \Sigma^\omega$ over the alphabet Σ induces a run which is an infinite sequence of states $q_0 q_1 \ldots \in Q^\omega$, such that $q_{i+1} = \delta(q_i, w(i))$ for all $i \geq 0$. Given a run r, let $\text{Inf}(r)$ denote the set of states that appear infinitely often in r. Given a Büchi implication acceptance condition (E, F), a run r is accepting, if $\text{Inf}(r) \cap E \neq \emptyset$ implies $\text{Inf}(r) \cap F \neq \emptyset$, *i.e.,* if the set E is visited infinitely often, then the set F is visited infinitely often. The Büchi acceptance condition is a special case of Büchi implication acceptance condition where $E = Q$, *i.e.,* we require F to be visited infinitely often. The language of an automaton is the set of words that induce an accepting run.

Definition 2 (*GR(1) Formulae*)**.** A GR(1) formula φ is a particular type of an LTL formula over alphabet Σ of the form

$$\varphi = \left(\bigwedge_{i=1}^{m} \varphi_i\right) \implies \left(\bigwedge_{j=1}^{n} \varphi_j\right), \tag{3}$$

where each φ_i, φ_j is an LTL formula that can be represented by a deterministic ω-automaton with Büchi acceptance condition.

The above definition of GR(1) is the extended version of the standard General Reactivity(1) fragment introduced in [16]. The advantage of using GR(1) instead of full LTL as specification language is that realizability for LTL is 2EXPTIME-complete [5], whereas for GR(1) it is only cubic in the size of the formula [16]. Given a finite number of deterministic ω-automata with Büchi acceptance conditions, we can construct a deterministic ω-automaton with Büchi acceptance condition that accepts the intersection of the languages of the given automata [2]. Thus a GR(1) formula can be converted to a deterministic ω-automaton with a Büchi implication acceptance condition.

2.3. Games

In this work, we consider the following probabilistic games that generalize Markov decision processes (MDPs).

Definition 3 (*2½-Player Games*)**.** A two-player turn-based probabilistic game, or 2½-player game, is a tuple $\mathcal{G} = (S_1, S_2, Act, \delta)$, where S_1 and S_2 are disjoint finite sets of states for Player 1 and Player 2, respectively, Act is a finite set of actions for the players, and $\delta\colon (S_1 \cup S_2) \times Act \to \mathcal{D}(S_1 \cup S_2)$, is a probabilistic transition function.

Let $S = S_1 \cup S_2$. A play of a 2½-player game $\mathcal{G}$ is a sequence $g \in S^\omega$ such that for all $n \geq 1$ there exists $a \in Act$ such that $\delta(g(n), a)(g(n+1)) > 0$. A finite play is a finite prefix of a play of $\mathcal{G}$. A Player 1 strategy for $\mathcal{G}$ is a function $C^1_{\mathcal{G}}\colon S^* \cdot S_1 \to Act$ that determines the Player 1 action to be applied after any finite prefix of a play ending in a Player 1 state, and strategies for Player 2 are defined analogously. If there exists an implementation of a strategy that uses finite memory, *e.g.,* a finite-state transducer, the strategy is called finite-memory. If there exists an implementation that uses only one memory element, it is called memoryless. Given a Player 1 and Player 2 strategy, and a starting state, there exists a unique probability measure over sets of plays.

Given a game $\mathcal{G}$, an acceptance condition defines the set of accepting plays. We consider GR(1) formulae and Büchi implication over S as accepting conditions for $\mathcal{G}$. The almost-sure winning set, denoted as $\text{Almost}^{\mathcal{G}}(\varphi)$ for a GR(1) formula and $\text{Almost}^{\mathcal{G}}((E, F))$ for a Büchi implication condition, is the set of states such that Player 1 has a strategy to ensure the objective with probability 1 irrespective of the strategy of Player 2. Formally, $\text{Almost}^{\mathcal{G}}(\varphi) = \{s \in S \mid \exists C^1_{\mathcal{G}}\, \forall C^2_{\mathcal{G}}$ the probability to satisfy φ using the two strategies and starting from state s is 1$\}$, and $\text{Almost}^{\mathcal{G}}((E, F))$ is defined similarly. The almost-sure winning set $\text{Almost}^{\mathcal{G}}((E, F))$ for Büchi implication acceptance condition can be solved in quadratic time [36,18]. In this work, we use more intuitive, cubic time algorithm described in detail in Appendix A. Moreover, in the states of the set $\text{Almost}^{\mathcal{G}}((E, F))$, there always exist witness strategies, called almost-sure winning strategies, that are memoryless and indeed pure, *i.e.,* not randomized, as defined above. This follows from the fact that the Büchi implication condition can be seen as a special case of a more general parity acceptance condition [18]. In Section 4, we show how to compute the almost-sure winning set $\text{Almost}^{\mathcal{G}}(\varphi)$ for a GR(1) formula φ.

In this work, we also consider the following cooperative interpretation of 2½-player games which is an MDP or so called 1½-player game.

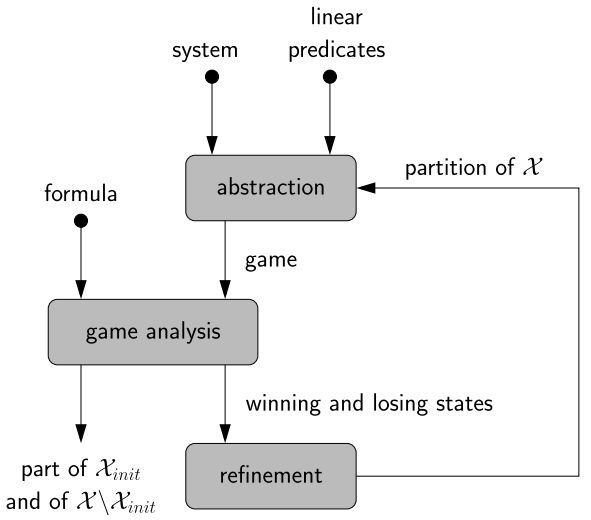


Fig. 1. Graphical representation of the proposed solution to Problem 1. Linear stochastic system is abstracted using a $2^1/_2$-player game by partitioning the state and control spaces using polytopic operations. The game is solved to identify those partition elements of the state space that belong to $\mathcal{X}_{\text{init}}$, *i.e.,* from which the given temporal formula can be almost-surely satisfied, as well as those that lie outside $\mathcal{X}_{\text{init}}$, *i.e.,* from which it can never be almost-surely satisfied. The remaining, unidentified partition elements may still contribute to $\mathcal{X}_{\text{init}}$ and are further refined, resulting in a new state space partition.

Definition 4 ($1^1/_2$*-Player Games*)**.** An MDP or $1^1/_2$-player game, is a tuple $\mathcal{G} = (S, Act, \delta)$, where S, Act are non-empty finite sets of states and actions, and $\delta : S \times Act \to \mathcal{D}(S)$ is a probabilistic transition function.

Given a $2^1/_2$-player game $\mathcal{G}$, the $1^1/_2$-player interpretation, where the players cooperate, is called $\mathcal{G}^{\text{coop}}$ with $S = S_1 \cup S_2$. The almost-sure winning set in $\mathcal{G}^{\text{coop}}$ for a GR(1) formula φ is then defined as $\text{Almost}^{\mathcal{G}^{\text{coop}}}(\varphi) = \{s \in S \mid \exists C^1_{\mathcal{G}}\, \exists C^2_{\mathcal{G}}$ such that the probability to satisfy φ using the two strategies and starting from state s is 1$\}$, analogously for a Büchi implication acceptance condition.

3. Problem formulation

In this work, we assume we are given a linear stochastic system $\mathcal{T}$ defined as

$$\mathcal{T} : x_{t+1} = Ax_t + Bu_t + w_t, \tag{4}$$

where $x_t \in \mathcal{X} \subset \mathbb{R}^N$, $u_t \in \mathcal{U} \subset \mathbb{R}^M$, $\mathcal{X}$, $\mathcal{U}$ are polytopes in the corresponding Euclidean spaces called the state space and control space, respectively, $w_t \in \mathcal{W} \subset \mathbb{R}^N$ is the value at time t of a random vector with values in polytope $\mathcal{W}$. The random vector has positive density on all values in $\mathcal{W}$. Finally, A and B are matrices of appropriate sizes.

The system $\mathcal{T}$ evolves in traces. A trace of a linear stochastic system $\mathcal{T}$ is an infinite sequence $\rho \in \mathcal{X}^\omega$ such that for every $n \geq 1$, we have $\rho(n+1) = A\rho(n) + Bu + w$ for some $u \in \mathcal{U}$, $w \in \mathcal{W}$. A finite trace $\sigma \in \mathcal{X}^+$ of $\mathcal{T}$ is then a finite prefix of a trace. A linear stochastic system $\mathcal{T}$ can be controlled using control strategies, where a control strategy is a function $C_{\mathcal{T}} : \mathcal{X}^+ \to \mathcal{U}$.

To formulate specifications over the linear stochastic system $\mathcal{T}$, we assume we are given a set Π of linear predicates over the state space $\mathcal{X}$ of $\mathcal{T}$:

$$\Pi = \{\pi_k \mid \pi_k : c_k^T x \leq d_k, c_k \in \mathbb{R}^N, d_k \in \mathbb{R}, k \in K\}, \tag{5}$$

where K is a finite index set. Every trace of the system generates a word over 2^Π, and every GR(1) specification formulated over the alphabet Π can be interpreted over these words.

Problem 1. Given a linear stochastic system $\mathcal{T}$ (Eq. (4)), a finite set of linear predicates Π (Eq. (5)) and a GR(1) formula φ over alphabet Π, find the set $\mathcal{X}_{\text{init}}$ of states $x \in \mathcal{X}$ for which there exists a control strategy $C_{\mathcal{T}}$ such that the probability that a trace starting in state x using $C_{\mathcal{T}}$ satisfies φ is 1, and find the corresponding strategies for $x \in \mathcal{X}_{\text{init}}$.

Approach overview. In Section 4, we propose a solution for Problem 1 that can be summarized as follows. First, we abstract the linear stochastic system $\mathcal{T}$ using a $2^1/_2$-player game based on the partition of the state space $\mathcal{X}$ given by linear predicates Π. The game is built only using polytopic operations on the state space and control space. We analyze the game and identify those partition elements of the state space $\mathcal{X}$ that provably belong to the solution set $\mathcal{X}_{\text{init}}$, as well as those that do not contain any state from $\mathcal{X}_{\text{init}}$. The remaining parts of the state space still have the potential to contribute to the set $\mathcal{X}_{\text{init}}$ but are not decided yet due to coarse abstraction. In the next step, the partition of state space $\mathcal{X}$ is refined using deep

analysis of the constructed game. Given the new partition, we build a new game and repeat the analysis. The approach can be graphically represented as shown in Fig. 1.

We prove that the result of every iteration is a partial solution to Problem 1. In other words, the computed set of satisfying initial states and the set of non-satisfying initial states are correct. Moreover, they are improved or maintained with every iteration as the abstraction gets more precise. This allows us to efficiently use the proposed algorithm for a fixed number of iterations. Finally, we prove that if the algorithm terminates then the result is indeed the solution to Problem 1. We demonstrate the proposed algorithm on a case study in Section 5.

The main difficulty of the approach is the abstraction-refinement of the $2\frac{1}{2}$-player game. Abstraction-refinement has been considered for discrete systems [11,20–22], and also for some classes of hybrid systems [24,25]. However, in all of these approaches, even if the original system is considered to be probabilistic, the distributions are assumed to be discrete and given, and are not abstracted away during the refinement. The key challenge is the extension of abstraction-refinement approach to continuous stochastic systems, where the transition probabilities in the abstract discrete model need to be abstracted. We show that by exploiting the nature of the considered dynamic systems we can develop an abstraction-refinement approach for our problem, see Remark 1.

Reachability analysis. While the solution to Problem 1 presented in the following section is sound and every iteration results in a partial solution, it suffers from high computational complexity. Intuitively, every iteration of the abstraction-refinement algorithm runs in time exponential with respect to the size of the current state space partition and the partition is non-trivially refined at the end of every iteration. This fact can also be observed in the case study in Section 5. To address this issue, in Section 6 we offer a detailed analysis of Problem 1 for the class of reachability properties, where the goal is to eventually reach a given target region within the state space $\mathcal{X}$. The corresponding reformulation of Problem 1 is as follows.

Problem 2. Given a linear stochastic system $\mathcal{T}$ (Eq. (4)) and a set of target polytopes $\{\mathcal{X}_k\}_{k\in K}$ such that $\mathcal{X}_k \subseteq \mathcal{X}$ for every $k \in K$, find the set $\mathcal{X}_{\text{init}}$ of states $x \in \mathcal{X}$ for which there exists a control strategy $C_{\mathcal{T}}$ such that the probability that a trace starting in state x using $C_{\mathcal{T}}$ reaches a target polytope is 1, and find the corresponding strategies for $x \in \mathcal{X}_{\text{init}}$.

In Section 6, we present several algorithms that can be used to partially or fully solve Problem 2. First, we present a simple algorithm that can be used to compute the set of satisfying initial states in a fast manner. The algorithm operates directly on the linear stochastic system using polytopic operators only and involves no construction, analysis or refinement of an abstract model. While being fast, the algorithm provides no information about the corresponding satisfying control strategies. Second, we extend this algorithm to provide partial information about the strategies by maintaining a specific state space partition that can be used to rule out control inputs that cannot appear in any satisfying control strategy for states in chosen partition elements. The algorithm is again based on polytopic operators only and the abstract game is constructed only once, for the final state space partition, in order to provide the above information on satisfying control strategies. The game however does not need to be solved.

Finally, we combine the latter algorithm with the abstraction-refinement algorithm from Section 4. The resulting framework first computes the set of all satisfying initial states in a fast manner together with a specific state space partition. Namely, the state space $\mathcal{X}$ is divided into so called layers according to the minimum number of time steps needed to reach the target region with non-zero probability. The abstraction-refinement algorithm is then used to compute control strategies to move between these layers towards the target region. These control strategies can be computed in parallel. Theoretically, the computational complexity of this combined algorithm is the same as for the abstraction-refinement algorithm. However, we demonstrate on case studies in Section 7 that it can lead to significantly reduced running times.

4. Solution

In this section, we describe the proposed solution to Problem 1 in detail and present necessary proofs. We start with the abstraction procedure that consists of two steps. The linear stochastic system $\mathcal{T}$ is first abstracted using a non-deterministic transition system which is then extended to a $2\frac{1}{2}$-player game. The game analysis section then describes how to identify parts of the solution to Problem 1. The procedure for refinement is presented last. Finally, we prove all properties of the proposed solution.

Let $\mathcal{X}_{\text{out}}$ be the set of all states outside of the state space $\mathcal{X}$ that can be reached within one step in system $\mathcal{T}$, *i.e.*, $\mathcal{X}_{\text{out}}$ is the set $\text{Post}(\mathcal{X}, \mathcal{U}) \setminus \mathcal{X}$, where Post is the posterior operator defined in Table 1. Note that $\mathcal{X}_{\text{out}}$ is generally not a polytope, but it can be represented as a finite set of polytopes $\{\mathcal{X}_{i_{\text{out}}}\}_{i_{\text{out}}\in I_{\text{out}}}$, or $\{\mathcal{X}_{i_{\text{out}}}\}$ for short. All polytopic operators that are used in this section are formally defined in Table 1 and their computation is described in detail in Appendix B.

4.1. Abstraction

The abstraction consists of two steps. First, the linear stochastic system is abstracted using a non-deterministic transition system which is then extended to a $2\frac{1}{2}$-player game.

Definition 5 (*NTS*)**.** A non-deterministic transition system (NTS) is a tuple $\mathcal{N} = (S, Act, \delta)$, where S is a non-empty finite set of states, Act is a non-empty finite set of actions, and $\delta : S \times Act \to 2^S$ is a non-deterministic transition function.

Table 1
Definitions of polytopic operators Post (posterior), Pre (predecessor), PreR (robust predecessor), PreP (precise predecessor), Attr (attractor) and AttrR (robust attractor), where $\mathcal{X}' \subseteq \mathcal{X}$, $\mathcal{U}' \subseteq \mathcal{U}$ are polytopes, and $\{\mathcal{X}_j\}_{j\in J}$ is a set of polytopes in $\mathcal{X}$. The algorithms to compute all the operators are listed in Appendix B. Operators Post, Pre and AttrR can be computed in time polynomial in the number of vertices of all input polytopes and the size of the index set J, and operators PreR, PreP and Attr can be computed in time polynomial in the number of vertices of all input polytopes and exponential in the size of the index set J.

$$
\begin{aligned}
\mathrm{Post}(\mathcal{X}', \mathcal{U}') &= \{x \in \mathbb{R}^N \mid \exists x' \in \mathcal{X}', \exists u \in \mathcal{U}', \exists w \in \mathcal{W} : x = Ax' + Bu + w\}\\
\mathrm{Pre}(\mathcal{X}', \mathcal{U}', \{\mathcal{X}_j\}_{j\in J}) &= \{x \in \mathcal{X}' \mid \exists u \in \mathcal{U}' : \mathrm{Post}(x, u) \cap \textstyle\bigcup_{j\in J} \mathcal{X}_j \text{ is non-empty}\}\\
\mathrm{PreR}(\mathcal{X}', \mathcal{U}', \{\mathcal{X}_j\}_{j\in J}) &= \{x \in \mathcal{X}' \mid \exists u \in \mathcal{U}' : \mathrm{Post}(x, u) \subseteq \textstyle\bigcup_{j\in J} \mathcal{X}_j\}\\
\mathrm{PreP}(\mathcal{X}', \mathcal{U}', \{\mathcal{X}_j\}_{j\in J}) &= \{x \in \mathcal{X}' \mid \exists u \in \mathcal{U}' : \mathrm{Post}(x, u) \subseteq \textstyle\bigcup_{j\in J} \mathcal{X}_j \text{ and } \forall j \in J : \mathrm{Post}(x, u) \cap \mathcal{X}_j \text{ is non-empty}\}\\
\mathrm{Attr}(\mathcal{X}', \mathcal{U}', \{\mathcal{X}_j\}_{j\in J}) &= \{x \in \mathcal{X}' \mid \forall u \in \mathcal{U}' : \mathrm{Post}(x, u) \cap \textstyle\bigcup_{j\in J} \mathcal{X}_j \text{ is non-empty}\}\\
\mathrm{AttrR}(\mathcal{X}', \mathcal{U}', \{\mathcal{X}_j\}_{j\in J}) &= \{x \in \mathcal{X}' \mid \forall u \in \mathcal{U}' : \mathrm{Post}(x, u) \subseteq \textstyle\bigcup_{j\in J} \mathcal{X}_j\}
\end{aligned}
$$

NTS construction. In order to build an NTS abstraction for $\mathcal{T}$, we assume we are given a partition $\{\mathcal{X}_i\}_{i\in I}$, or $\{\mathcal{X}_i\}$ for short, of the state space $\mathcal{X}$. Initially, the partition is given by the set of linear predicates Π, *i.e.,* it is the partition given by the equivalence relation $\sim_\Pi$ defined as

$$x \sim_\Pi x' \iff \forall k \in K : \left(c_k^T x \le d_k \Leftrightarrow c_k^T x' \le d_k\right).$$

In the later iterations of the algorithm, the partition is given by the refinement procedure. The construction below builds on the approach from [8].

We use $\mathcal{N}_{\{\mathcal{X}_i\}} = (S_\mathcal{N}, Act_\mathcal{N}, \delta_\mathcal{N})$ to denote the NTS corresponding to partition $\{\mathcal{X}_i\}$ defined as follows. The states of $\mathcal{N}_{\{\mathcal{X}_i\}}$ are given by the partition of the state space $\mathcal{X}$ and the outer part $\mathcal{X}_{\text{out}}$, *i.e.,* $S_\mathcal{N} = \{\mathcal{X}_i\} \cup \{\mathcal{X}_{i_{\text{out}}}\}$. Let $\mathcal{X}_i \in \{\mathcal{X}_i\} \subset S_\mathcal{N}$ be a state of the NTS, a polytope in $\mathcal{X}$. We use $\sim_i$ to denote the equivalence relation on $\mathcal{U}$ such that $u \sim_i u'$ if for every state $\mathcal{X}_j \in \{\mathcal{X}_i\} \cup \{\mathcal{X}_{i_{\text{out}}}\}$, it holds that $\mathrm{Post}(\mathcal{X}_i, u) \cap \mathcal{X}_j$ is non-empty if and only if $\mathrm{Post}(\mathcal{X}_i, u') \cap \mathcal{X}_j$ is non-empty. Intuitively, two control inputs are equivalent with respect to $\mathcal{X}_i$, if from $\mathcal{X}_i$ the system $\mathcal{T}$ can transit to the same set of partition elements of $\mathcal{X}$ and $\mathcal{X}_{\text{out}}$. The partition $\mathcal{U}/\sim_i$ is then the set of all actions of the NTS $\mathcal{N}_{\{\mathcal{X}_i\}}$ that are allowed in state $\mathcal{X}_i$. We use $\mathcal{U}_i^J$ to denote the union of those partition elements from $\mathcal{U}/\sim_i$ that contain control inputs that lead the system from $\mathcal{X}_i$ to polytopes $\mathcal{X}_j, j \in J \subseteq I \cup I_{\text{out}}$, *i.e.,*

$$\mathcal{U}_i^J = \{u \in \mathcal{U} \mid \forall j \in J : \mathrm{Post}(\mathcal{X}_i, u) \cap \mathcal{X}_j \text{ is non-empty and } \forall j \notin J : \mathrm{Post}(\mathcal{X}_i, u) \cap \mathcal{X}_j \text{ is empty}\}. \tag{6}$$

The set $\mathcal{U}_i^J$ can be computed using only polytopic computations as described in Appendix B.1. For a state $\mathcal{X}_i \in \{\mathcal{X}_i\} \subset S_\mathcal{N}$ and action $\mathcal{U}_{i'}^J \in Act_\mathcal{N}$, we let

$$\delta_\mathcal{N}(\mathcal{X}_i, \mathcal{U}_{i'}^J) = \begin{cases} \{\mathcal{X}_j \mid j \in J\} & \text{if } i = i',\\ \emptyset & \text{otherwise.}\end{cases}$$

For states $\mathcal{X}_{i_{\text{out}}} \in \{\mathcal{X}_{i_{\text{out}}}\} \subset S_\mathcal{N}$, no actions or transitions are defined.

From NTS to game. Since the NTS does not capture the probabilistic aspect of the linear stochastic system, we build a 2½-player game on top of the NTS. Let $\mathcal{X}_i$ be a polytope within the state space $\mathcal{X}$ of $\mathcal{T}$, a state of $\mathcal{N}_{\{\mathcal{X}_i\}}$. When $\mathcal{T}$ is in a particular state $x \in \mathcal{X}_i$ and a control input $u \in \mathcal{U}_i^J$ is to be applied, we can compute the probability distribution over the set $\{\mathcal{X}_j\}_{j\in J}$ that determines the probability of the next state of $\mathcal{T}$ being in $\mathcal{X}_j, j \in J$, using the distribution of the random vector for uncertainty. The evolution of the system can thus be seen as a game, where Player 1 acts in states $\mathcal{X}_i \in S_\mathcal{N}$ of the NTS and chooses actions from $Act_\mathcal{N}$, and Player 2 determines the exact state within the polytope $\mathcal{X}_i$ and thus chooses the probability distribution according to which a transition in $\mathcal{T}$ is made. This intuitive game construction implies that Player 2 has a possibly infinite number of actions. On the other hand, in Problem 1 we are interested in satisfying the GR(1) specification with probability 1 and in the theory of finite discrete probabilistic models, it is a well-studied phenomenon that in almost-sure analysis, the exact probabilities in admissible probability distributions of the model are not relevant. It is only important to know supports of such distributions, see *e.g.,* [2], where almost-sure reachability is computed only using graph analysis, omitting the exact transition probabilities. For intuition, recall the example of a faulty messaging protocol from Section 1. As long as the probability of success remains non-zero in every step, the message is eventually sent with probability 1. That means that in our case we do not need to consider all possible probability distributions as actions for Player 2, but it is enough to consider that Player 2 chooses support for the probability distribution that will be used to make a transition. For a polytope $\mathcal{X}_i \in S_\mathcal{N}$ and $\mathcal{U}_i^J \in Act_\mathcal{N}$, we use $\mathrm{Supp}(\mathcal{X}_i, \mathcal{U}_i^J)$ to denote the set of all subsets $J' \subseteq J$ for which there exist $x \in \mathcal{X}_i, u \in \mathcal{U}_i^J$ such that the next state $x' = Ax + Bu + w$ of $\mathcal{T}$ belongs to $\mathcal{X}_j, j \in J'$ with non-zero probability and with zero probability to $\mathcal{X}_j, j \notin J'$, *i.e.,*

$$\mathrm{Supp}(\mathcal{X}_i, \mathcal{U}_i^J) = \{J' \subseteq J \mid \mathrm{PreP}(\mathcal{X}_i, \mathcal{U}_i^J, \{\mathcal{X}_j\}_{j\in J'}) \text{ is non-empty}\}, \tag{7}$$

where PreP is the precise predecessor operator from Table 1.

Game construction. Given the NTS $\mathcal{N}_{\{\mathcal{X}_i\}}$, the 2½-player game $\mathcal{G}_{\{\mathcal{X}_i\}} = (S_1, S_2, Act, \delta_\mathcal{G})$ is defined as follows. Player 1 states $S_1 = \{\mathcal{X}_i\} \cup \{\mathcal{X}_{i_{\text{out}}}\}$ are the states $S_\mathcal{N}$ of the NTS and Player 1 actions are the actions $Act_\mathcal{N}$ of $\mathcal{N}_{\{\mathcal{X}_i\}}$. Player 2 states are given

by the choice of an action in a Player 1 state, *i.e.,* $S_2 = \{\mathcal{X}_i\} \times \{\mathcal{U}_i^J\}$. The Player 2 actions available in a state $(\mathcal{X}_i, \mathcal{U}_i^J)$ are the elements of the set $\mathrm{Supp}(\mathcal{X}_i, \mathcal{U}_i^J)$ defined in Eq. (7). For Player 1, the transition probability function $\delta_{\mathcal{G}}$ defines non-zero probability transitions only for triples of the form $\mathcal{X}_i, \mathcal{U}_i^J, (\mathcal{X}_i, \mathcal{U}_i^J)$ and for such it holds $\delta_{\mathcal{G}}(\mathcal{X}_i, \mathcal{U}_i^J)((\mathcal{X}_i, \mathcal{U}_i^J)) = 1$. For Player 2, the function $\delta_{\mathcal{G}}$ defines the following transitions:

$$\delta_{\mathcal{G}}\big((\mathcal{X}_i, \mathcal{U}_i^J), J'\big)\big(\mathcal{X}_j\big) = \begin{cases} \dfrac{1}{|J'|} & \text{if } J' \in \mathrm{Supp}(\mathcal{X}_i, \mathcal{U}_i^J) \\ & \text{and } j \in J', \\ 0 & \text{otherwise.} \end{cases}$$

The definition reflects the fact that once Player 2 chooses the support, the exact transition probabilities are irrelevant and without loss of generality, we can consider them to be uniform.

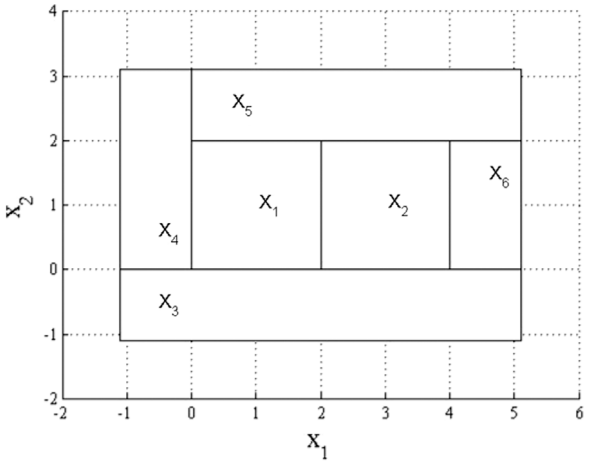


Fig. 2. Partition of state space $\mathcal{X}$ of system $\mathcal{T}$ in Example 1 given by linear predicates Π. Polytopes $\mathcal{X}_3, \ldots, \mathcal{X}_6$ form the set $\mathcal{X}_{\text{out}}$.

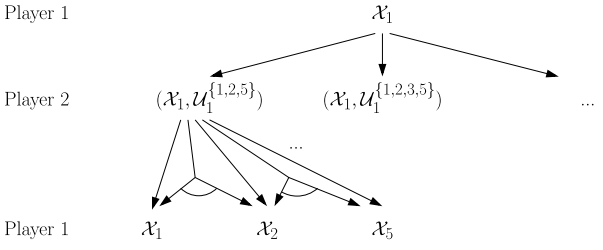


Fig. 3. Part of the transition function of the game $\mathcal{G}_{\{\mathcal{X}_i\}}$ constructed in Example 1.

Example 1 (*Illustrative Example, Part I*)**.** Let $\mathcal{T}$ be a linear stochastic system of the form given in Eq. (4), where A and B are identity matrices of size 2, the state space is $\mathcal{X} = \{x \in \mathbb{R}^2 \mid 0 \leq x(1) \leq 4, 0 \leq x(2) \leq 2\}$, the control space is $\mathcal{U} = \{u \in \mathbb{R}^2 \mid -1 \leq u(1), u(2) \leq 1\}$, and the random vector takes values in polytope $\mathcal{W} = \{w \in \mathbb{R}^2 \mid -0.1 \leq w(1), w(2) \leq 0.1\}$. Let Π contain a single linear predicate $\pi_1 \colon x(1) \leq 2$. In Fig. 2, polytopes $\mathcal{X}_1$ and $\mathcal{X}_2$ form the partition of $\mathcal{X}$ given by Π, and polytopes $\mathcal{X}_3$, $\mathcal{X}_4$, $\mathcal{X}_5$, $\mathcal{X}_6$ form the rest of the one step reachable set of system $\mathcal{T}$, *i.e.,* $\mathcal{X}_{\text{out}}$. The game $\mathcal{G}_{\{\mathcal{X}_i\}}$ given by this partition has 6 states and 18 actions. In Fig. 3, we visualize part of the transition function as follows. In Player 1 state $\mathcal{X}_1$, if Player 1 chooses, *e.g.,* action $\mathcal{U}_1^{\{1,2,5\}}$ that leads from $\mathcal{X}_1$ to polytopes $\mathcal{X}_1$, $\mathcal{X}_2$, $\mathcal{X}_5$, the game is in Player 2 state $(\mathcal{X}_1, \mathcal{U}_1^{\{1,2,5\}})$ with probability 1. Actions of Player 2 are the available supports of the action over the set $\{\mathcal{X}_1, \mathcal{X}_2, \mathcal{X}_5\}$, which are in this case all non-empty subsets. If Player 2 chooses, *e.g.,* support $\{\mathcal{X}_1, \mathcal{X}_2\}$, the game is in Player 1 state $\mathcal{X}_1$ or $\mathcal{X}_2$ with equal probability 0.5.

The following proposition proves that the game $\mathcal{G}_{\{\mathcal{X}_i\}}$ simulates the linear stochastic system $\mathcal{T}$.

Proposition 1. *Let ρ be a trace of the linear stochastic system $\mathcal{T}$. Then there exists a play g of the game $\mathcal{G}_{\{\mathcal{X}_i\}}$ such that $\rho(n) \in g(2n-1)$ for every $n \geq 1$.*

Proof. The play g is defined as follows. The states $g(2n-1) = \mathcal{X}_i$ such that $\rho(n) \in \mathcal{X}_i$. The states $g(2n) = (\mathcal{X}_i, \mathcal{U}_i^J)$ such that there exist $u \in \mathcal{U}_i^J$, $w \in \mathcal{W}$ for which $\rho(n+1) = A\rho(n) + Bu + w$. □

On the other hand, since $\mathcal{G}_{\{\mathcal{X}_i\}}$ is only an abstraction of the system $\mathcal{T}$, it may contain plays that do not correspond to any trace of the system.

4.2. Game analysis

Let $\mathcal{G}_{\{\mathcal{X}_i\}}$ be the $2^1/_2$-player game constructed for the linear stochastic system $\mathcal{T}$ and partition $\{\mathcal{X}_i\}$ of its state space using the procedure from Section 4.1. In this section, we identify partition elements from $\{\mathcal{X}_i\}$ which are part of the solution set of initial states $\mathcal{X}_{\text{init}}$ as well as those that do not contain any satisfying initial states from $\mathcal{X}$.

Computing satisfying states. First, we compute the almost-sure winning set S_{yes} in game $\mathcal{G}_{\{\mathcal{X}_i\}}$ with respect to the GR(1) formula φ from Problem 1, *i.e.,*

$$S_{\text{yes}} = \text{Almost}^{\mathcal{G}_{\{\mathcal{X}_i\}}}(\varphi). \tag{8}$$

We proceed as follows. Let $\mathcal{A} = (Q, 2^{\Pi}, \delta_{\mathcal{A}}, q_0, (E, F))$ be a deterministic ω-automaton with Büchi implication acceptance condition for the GR(1) formula φ constructed as described in Section 2.2. We consider the $2^1/_2$-player game $\mathcal{P}_{\{\mathcal{X}_i\}} = (S_1^{\mathcal{P}}, S_2^{\mathcal{P}}, Act, \delta_{\mathcal{P}})$ that is the synchronous product of $\mathcal{G}_{\{\mathcal{X}_i\}}$ and $\mathcal{A}$, *i.e.,* $S_1^{\mathcal{P}} = S_1 \times Q$, $S_2^{\mathcal{P}} = S_2 \times Q$, and for every $(\mathcal{X}_i, q) \in S_1^{\mathcal{P}}$ and $\mathcal{U}_i^J \in Act$ we have

$$\delta_{\mathcal{P}}\big((\mathcal{X}_i, q), \mathcal{U}_i^J\big)\big((\mathcal{X}_i, \mathcal{U}_i^J), q'\big)\big) = \begin{cases} \delta_{\mathcal{G}}\big(\mathcal{X}_i, \mathcal{U}_i^J\big)\big((\mathcal{X}_i, \mathcal{U}_i^J)\big) & \text{if } \delta_{\mathcal{A}}(q, \Pi(\mathcal{X}_i)) = q', \\ 0 & \text{otherwise,} \end{cases}$$

where $\Pi(\mathcal{X}_i)$ is the set of all linear predicates from Π that are true on polytope $\mathcal{X}_i$, and similarly, for all $((\mathcal{X}_i, \mathcal{U}_i^J), q) \in S_2^{\mathcal{P}}$ and $J' \in Act$ we have

$$\delta_{\mathcal{P}}\big(((\mathcal{X}_i, \mathcal{U}_i^J), q), J'\big)\big((\mathcal{X}_j, q')\big) = \begin{cases} \delta_{\mathcal{G}}\big((\mathcal{X}_i, \mathcal{U}_i^J), J'\big)\big(\mathcal{X}_j\big) & \text{if } q = q', \\ 0 & \text{otherwise.} \end{cases}$$

When constructing the product game, we only consider those states from $S_1 \times Q$ and $S_2 \times Q$ that are reachable from some $(\mathcal{X}_i, q_0)$, where q_0 is the initial state of the automaton $\mathcal{A}$. Finally, we consider Büchi implication acceptance condition $(E^{\mathcal{P}}, F^{\mathcal{P}})$, where $E^{\mathcal{P}} = (S_1^{\mathcal{P}} \cup S_2^{\mathcal{P}}) \times E$ and $F^{\mathcal{P}} = (S_1^{\mathcal{P}} \cup S_2^{\mathcal{P}}) \times F$.

Proposition 2. *The set S_{yes} defined in Eq. (8) consists of all $\mathcal{X}_i \in S_1$ for which $(\mathcal{X}_i, q_0) \in S_{\text{yes}}^{\mathcal{P}}$, where the set*

$$S_{\text{yes}}^{\mathcal{P}} = \text{Almost}^{\mathcal{P}_{\{\mathcal{X}_i\}}}((E^{\mathcal{P}}, F^{\mathcal{P}})) \tag{9}$$

can be computed using algorithm in Appendix A.

Proof. Follows directly from the construction of the game $\mathcal{P}_{\{\mathcal{X}_i\}}$ above and the results of [37]. □

The next proposition proves that the polytopes from S_{yes} are part of the solution to Problem 1.

Proposition 3. *For every $\mathcal{X}_i \in S_{\text{yes}}$, there exists a finite-memory strategy $C_{\mathcal{T}}$ for $\mathcal{T}$ such that every trace of $\mathcal{T}$ under strategy $C_{\mathcal{T}}$ that starts in any $x \in \mathcal{X}_i$ satisfies φ with probability* 1.

Proof. Let $\mathcal{X}_i \in S_{\text{yes}}$ and let $C_{\mathcal{G}_{\{\mathcal{X}_i\}}}$ be a finite-memory almost-sure winning strategy for Player 1 from state $\mathcal{X}_i$ in game $\mathcal{G}_{\{\mathcal{X}_i\}}$, see Section 2.3. Let $C_{\mathcal{T}}$ be a strategy for $\mathcal{T}$ defined as follows. For a finite trace $\sigma_{\mathcal{T}}$, let $C_{\mathcal{T}}(\sigma_{\mathcal{T}}) = u$, where $u \in C_{\mathcal{G}_{\{\mathcal{X}_i\}}}(\sigma_{\mathcal{G}_{\{\mathcal{X}_i\}}})$, where $\sigma_{\mathcal{G}_{\{\mathcal{X}_i\}}}$ is finite play such that $\sigma_{\mathcal{T}}(n) \in \sigma_{\mathcal{G}_{\{\mathcal{X}_i\}}}(2n)$ for every $1 \leq n \leq |\sigma_{\mathcal{T}}|$. Since $C_{\mathcal{G}_{\{\mathcal{X}_i\}}}$ for game $\mathcal{G}_{\{\mathcal{X}_i\}}$ is almost-sure winning from state $\mathcal{X}_i$ with respect to φ, *i.e.,* every play that starts in $\mathcal{X}_i$ almost-surely satisfies φ, the analogous property holds for $C_{\mathcal{T}}$ and traces in $\mathcal{T}$. □

Computing non-satisfying states. Next, we consider the set S_{no} of Player 1 states in game $\mathcal{G}_{\{\mathcal{X}_i\}}$ defined as follows:

$$S_{\text{no}} = S_1 \setminus \text{Almost}^{\mathcal{G}^{\text{coop}}_{\{\mathcal{X}_i\}}}(\varphi). \tag{10}$$

Intuitively, S_{no} is the set of states, where even if Player 2 cooperates with Player 1, φ can still not be satisfied with probability 1.

Proposition 4. *The set S_{no} defined in Eq. (10) consists of all $\mathcal{X}_i \in S_1$ for which $(\mathcal{X}_i, q_0) \in S_{\text{no}}^{\mathcal{P}}$, where*

$$S_{\text{no}}^{\mathcal{P}} = S_1^{\mathcal{P}} \setminus \text{Almost}^{\mathcal{P}^{\text{coop}}_{\{\mathcal{X}_i\}}}((E^{\mathcal{P}}, F^{\mathcal{P}})). \tag{11}$$

Proof. Follows directly from the construction of the product game $\mathcal{P}_{\{\mathcal{X}_i\}}$. □

We prove that no state $x \in \mathcal{X}_i$ for $\mathcal{X}_i \in S_{\text{no}}$ is part of the solution to Problem 1.

Proposition 5. *For every $\mathcal{X}_i \in S_{\text{no}}$ and $x \in \mathcal{X}_i$, there does not exist a strategy $C_{\mathcal{T}}$ for $\mathcal{T}$ such that every trace of $\mathcal{T}$ under $C_{\mathcal{T}}$ starting in x satisfies φ with probability* 1.

Proof. Intuitively, from the construction of the game $\mathcal{G}_{\{\mathcal{X}_i\}}$ in Section 4.1, Player 2 represents the unknown precise state of the system $\mathcal{T}$ in the abstraction, *i.e.,* he makes the choice of a state inside each polytope $\mathcal{X}_i$ at each step. Therefore, if φ cannot be almost-surely satisfied from $\mathcal{X}_i$ in the game even if the two players cooperate, in $\mathcal{T}$ it translates to the fact that φ cannot be almost-surely satisfied from any $x \in \mathcal{X}_i$ even if we consider strategies that can moreover change inside each $\mathcal{X}_i$ arbitrarily at any moment. □

Undecided states. Finally, consider the set

$$S_? = S_1 \setminus (S_{\text{yes}} \cup S_{\text{no}}). \tag{12}$$

These are the polytopes within the state space of $\mathcal{T}$ that have not been decided as satisfying or non-satisfying due to coarse abstraction. Alternatively, from Propositions 2 and 4, and Eq. (12), we can define the set $S_?$ as the set of all $\mathcal{X}_i \in S_1$, for which $(\mathcal{X}_i, q_0) \in S_?^{\mathcal{P}}$, where

$$S_?^{\mathcal{P}} = S_1^{\mathcal{P}} \setminus (S_{\text{yes}}^{\mathcal{P}} \cup S_{\text{no}}^{\mathcal{P}}). \tag{13}$$

Proposition 6. *For every $\mathcal{X}_i \in S_?$ it holds that the product game $\mathcal{P}_{\{\mathcal{X}_i\}}$ can be won cooperatively starting from the Player* 1 *state $(\mathcal{X}_i, q_0)$. Analogously, for every $(\mathcal{X}_i, q) \in S_?^{\mathcal{P}}$ it holds that the product game $\mathcal{P}_{\{\mathcal{X}_i\}}$ can be won cooperatively starting from $(\mathcal{X}_i, q)$.*

Proof. The proposition follows directly from Eqs. (12) and (13), and Propositions 2 and 4. □

Example 2 (*Illustrative Example, Part II*)**.** Recall the linear stochastic system $\mathcal{T}$ from Example 1 and consider GR(1) formula $\mathbf{F}\neg\pi_1$ over the set Π that requires to eventually reach a state $x \in \mathcal{X}$ such that $x(1) \geq 2$. The deterministic ω-automaton for the formula has only two states, q_0 and q_1. The automaton remains in the initial state q_0 until polytope $\mathcal{X}_2$ is visited in $\mathcal{T}$. Then it transits to state q_1 and remains there forever. The Büchi implication condition (E, F) is $E = \{q_0\}, F = \{q_1\}$. The solution of the game $\mathcal{G}_{\{\mathcal{X}_i\}}$ constructed in Example 1 with respect to the above formula is depicted in Fig. 4.

If the set S_{no} contains all Player 1 states of the game $\mathcal{G}_{\{\mathcal{X}_i\}}$, the GR(1) formula φ cannot be satisfied in the system $\mathcal{T}$ and our algorithm terminates. If set $S_?$ is empty, the algorithm terminates and returns the union of all polytopes from S_{yes} as the solution to Problem 1. The corresponding satisfying strategies are synthesized as described in the proof of Proposition 3. Otherwise, we continue the algorithm by computing a refined partition of the state space $\mathcal{X}$ as described in the next section.

4.3. Refinement

Refinement is a heuristic that constructs a new partition of $\mathcal{X}$, a subpartition of $\{\mathcal{X}_i\}$, that is used in the next iteration of the overall algorithm. We design two refinement procedures, called positive and negative, that aim to enlarge the combined volume of polytopes in the set S_{yes} and S_{no}, respectively, or equivalently, to reduce the combined volume of polytopes in the set $S_?$. Based on Propositions 2 and 4, both procedures are formulated over the product game $\mathcal{P}_{\{\mathcal{X}_i\}}$ and reach their respective goals through refining polytopes $\mathcal{X}_i$ for which $(\mathcal{X}_i, q) \in S_?^{\mathcal{P}}$ for some $q \in Q$.

In this section, we use J_{yes}^q to denote the set of all indices $i \in I$ for which $(\mathcal{X}_i, q) \in S_{\text{yes}}^{\mathcal{P}}$, and $J_?^q, J_{\text{no}}^q$ are defined analogously. In the two refinement procedures, every polytope $\mathcal{X}_i$ can be partitioned into a set of polytopes in iterative manner, as $(\mathcal{X}_i, q) \in S_?^{\mathcal{P}}$ can hold for multiple $q \in Q$. Therefore, given a partition of $\mathcal{X}_i$, the refinement of $\mathcal{X}_i$ according to a polytope $\mathcal{B}$ refers to the partition of $\mathcal{X}_i$ that contains all intersections and differences of elements of the original partition of $\mathcal{X}_i$ and polytope $\mathcal{B}$.

Positive refinement. In the positive refinement, we explore the following property of states in $S_?^{\mathcal{P}}$. In Proposition 6, we proved that the product game $\mathcal{P}_{\{\mathcal{X}_i\}}$ can be won cooperatively from every $(\mathcal{X}_i, q) \in S_?^{\mathcal{P}}$. It follows that there exist a Player 1 action $\mathcal{U}_i^J$ and Player 2 action J' such that after their application in $(\mathcal{X}_i, q)$, the game is not in a losing state with probability 1. We can graphically represent this property as follows:

$$(\mathcal{X}_i, q) \xrightarrow{\mathcal{U}_i^J} ((\mathcal{X}_i, \mathcal{U}_i^J), q') \xrightarrow{J'} \begin{cases} (\mathcal{X}_{j_1}, q') \\ \vdots \\ (\mathcal{X}_{j_n}, q') \end{cases} \tag{14}$$

where an arrow $a \xrightarrow{b}$ represents the uniform probability distribution $\delta_{\mathcal{P}}(a, b)$, and $\{j_1, \ldots, j_n\} = J' \subseteq J_{\text{yes}}^{q'} \cup J_?^{q'}$. Note that from the construction of the product game $\mathcal{P}_{\{\mathcal{X}_i\}}$ in Section 2.3 it follows that q' is given uniquely over all actions $\mathcal{U}_i^J$. The following design ensures that every polytope $\mathcal{X}_i$ is refined at least once for every state $(\mathcal{X}_i, q) \in S_?^{\mathcal{P}}, q \in Q$.

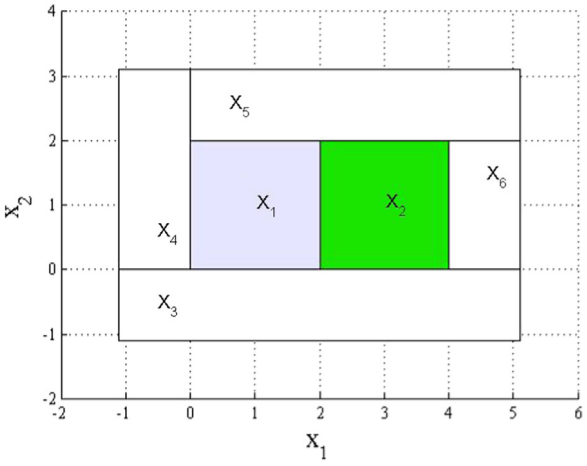


Fig. 4. Solution of the game in Example 2. The polytopes, *i.e.,* Player 1 states, that belong to sets S_{yes}, S_{no}, $S_?$ are shown in green, white and light blue, respectively. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Let $(\mathcal{X}_i, q) \in S_?^{\mathcal{P}}$. The positive refinement first refines $\mathcal{X}_i$ according to the robust predecessor

$$\text{PreR}(\mathcal{X}_i, \mathcal{U}, \{\mathcal{X}_j\}_{j \in J_{\text{yes}}^{q'}}). \tag{15}$$

That means, we find all states $x \in \mathcal{X}_i$ for which there exists any control input under which the system $\mathcal{T}$ evolves from x to a state $x' \in \mathcal{X}_j, j \in J_{\text{yes}}^{q'}$.

Next, the positive refinement considers three cases. First, assume that from $(\mathcal{X}_i, q)$, the two players can cooperatively reach a winning state of the product game in two steps with probability 1, and let $\mathcal{U}_i^J$ and J' be Player 1 and Player 2 actions, respectively, that accomplish that, *i.e.,* in Eq. (14), $\{j_1, \ldots, j_n\} = J' \subseteq J_{\text{yes}}^{q'}$. For every such $\mathcal{U}_i^J, J'$, we find an (arbitrary) partition $\{\mathcal{U}_y\}_{y \in Y}$ of the polytope $\mathcal{U}_i^J$ and we partition $\mathcal{X}_i$ according to the robust attractors

$$\text{AttrR}(\mathcal{X}_i, \mathcal{U}_y, \{\mathcal{X}_j\}_{j \in J_{\text{yes}}^{q'}}). \tag{16}$$

Intuitively, the above set contains all $x \in \mathcal{X}_i$ such that under every control input $u \in \mathcal{U}_y$, $\mathcal{T}$ evolves from x to a state $x' \in \mathcal{X}_j, j \in J_{\text{yes}}^{q'}$. Note that the robust attractor sets partition the robust predecessor set from Eq. (15), as every state x that belongs to one of the robust attractor sets must lie in the robust predecessor set as well. In the next iteration of the overall algorithm, the partition elements given by the robust attractor sets will belong to the set $S_{\text{yes}}^{\mathcal{P}}$. In the second case, assume that the two players can reach a winning state of the product game cooperatively in two steps, but only with probability $0 < p < 1$, while the probability of reaching a losing state is 0. Let $\mathcal{U}_i^J, J'$ be Player 1 and Player 2 actions, respectively, that maximize p, *i.e.,* in Eq. (14), there exists $m < n$ such that $\{j_1, \ldots, j_m\} = J' \cap J_{\text{yes}}^{q'}$, $\{j_{m+1}, \ldots, j_n\} = J' \cap J_?^{q'}$ and $p = \frac{m}{n}$ is maximal. Similarly as in the first case, we refine the polytope $\mathcal{X}_i$ according to the robust attractor sets as in Eq. (16), but we compute the sets with respect to the set of indices $J_{\text{yes}}^{q'} \cup \{j_{m+1}, \ldots, j_n\}$. Finally, assume that $(\mathcal{X}_i, q)$ does not belong to any of the above two categories. As argued at the beginning of this section, there still exist Player 1 and Player 2 actions $\mathcal{U}_i^J$ and J', respectively, such that in Eq. (14), $\{j_1, \ldots, j_n\} = J' \subseteq J_?^{q'}$. Again, we refine the polytope $\mathcal{X}_i$ according to the robust attractor sets as in Eq. (16), where the sets are computed with respect to the set of indices $J_?^{q'}$.

Example 3 (*Illustrative Example, Part III*)**.** We demonstrate a part of the positive refinement for the game in Example 2. Consider polytope $\mathcal{X}_1 \in S_?$. It follows from the form of the ω-automaton in Example 2 that $\mathcal{X}_1$ appears in $S_?^{\mathcal{P}}$ only in pair with q_0, *i.e.,* $(\mathcal{X}_1, q_0) \in S_?^{\mathcal{P}}$. Note that for state $(\mathcal{X}_1, q_0)$, every successor state is of the form $((\mathcal{X}_1, \mathcal{U}_1^J), q_0)$, *i.e.,* $q' = q_0$. First, polytope $\mathcal{X}_1$ is refined with respect to the robust predecessor

$$\text{PreR}(\mathcal{X}_1, \mathcal{U}, \{\mathcal{X}_2\}),$$

since $J_{\text{yes}}^{q_0} = \{\mathcal{X}_2\}$ because $(\mathcal{X}_2, q_0) \in S_{\text{yes}}^{\mathcal{P}}$ is a winning state of the product game. The robust predecessor set is depicted in Fig. 5 in cyan. Next, we decide which of the three cases described in the positive refinement procedure above applies to state $(\mathcal{X}_1, q_0)$. Consider for example Player 1 action $\mathcal{U}_1^{\{1,2,5\}}$ and Player 2 action $\{2\}$, as shown in Fig. 3. It holds that

$$(\mathcal{X}_1, q_0) \xrightarrow{\mathcal{U}_1^{\{1,2,5\}}} ((\mathcal{X}_1, \mathcal{U}_1^{\{1,2,5\}}), q_0) \xrightarrow{\{2\}} (\mathcal{X}_2, q_0),$$

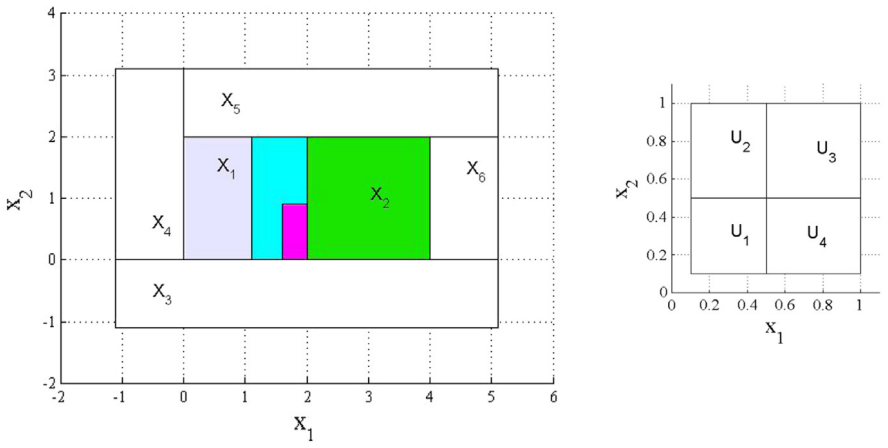


Fig. 5. Part of the positive refinement for the system in Example 3. Polytope $\mathcal{X}_1$ is first refined according to the robust predecessor as in Eq. (15), the robust predecessor is shown in cyan. Next, we consider the polytope of control inputs $\mathcal{U}_1^{\{1,2,5\}}$ and its partition as depicted on the right. The robust predecessor of $\mathcal{U}_3$ is then shown in magenta. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

and $(\mathcal{X}_2, q_0) \in S^{\mathcal{P}}_{\text{yes}}$ is a winning state of the product game. Therefore, the state $(\mathcal{X}_1, q_0)$ is of the first type. To further refine polytope $\mathcal{X}_1$, we first partition the polytope

$$\mathcal{U}_1^{\{1,2,5\}} = \{u \in \mathcal{U} \mid 0.1 \leq u(1), u(2) \leq 1\},$$

e.g., into 4 parts as shown in Fig. 5 on the right. The robust attractor

$$\text{AttrR}(\mathcal{X}_1, \mathcal{U}_3, \{\mathcal{X}_2\})$$

for one of the polytopes $\mathcal{U}_3$ is depicted in magenta in Fig. 5. This polytope will be recognized as a satisfying initial polytope in the next iteration, since starting in any x within the robust predecessor, system $\mathcal{T}$ as defined in Example 1 evolves from x under every control input from $\mathcal{U}_3$ to polytope $\mathcal{X}_2$.

Negative refinement. In the negative refinement, we consider all Player 1 states $(\mathcal{X}_i, q) \in S^{\mathcal{P}}_?$ such that if Player 2 does not cooperate, but rather plays against Player 1, the game is lost with non-zero probability. In other words, for every Player 1 $\mathcal{U}_i^j$, there exists a Player 2 action J' such that in Eq. (14), there exists an index $j \in J'$ such that $(\mathcal{X}_j, q') \in S^{\mathcal{P}}_{\text{no}}$. In this case, we refine polytope $\mathcal{X}_i$ according to the attractor set

$$\text{Attr}(\mathcal{X}_i, \mathcal{U}, \{\mathcal{X}_j\}_{j \in J^{q'}_{\text{no}}}).$$

Intuitively, the attractor set contains all states $x \in \mathcal{X}_i$ such that by applying any control input $u \in \mathcal{U}$, system $\mathcal{T}$ evolves from x to a state in $\mathcal{X}_j$ for some $j \in J^{q'}_{\text{no}}$ with non-zero probability. In the next iteration of the algorithm, the partition elements given by the attractor set will belong to the set $S^{\mathcal{P}}_{\text{no}}$.

Remark 1. We remark that both game theoretic aspects as well as the linear stochastic dynamics play an important role in the refinement step. The game theoretic results compute the undecided states, and thereby determine what parts of the state space need to be refined and which actions need to be considered in the refinement. The linear stochastic dynamics allow us to perform the refinement itself using polytopic operators.

4.4. Correctness and complexity

We prove that the algorithm presented in this section provides a partially correct solution to Problem 1.

For $n \in \mathbb{N}$, let $S^n_{\text{yes}}, S^n_{\text{no}}$ be the sets from Eqs. (8) and (10), respectively, computed in the nth iteration of the algorithm presented above. We use $\mathcal{X}^n_{\text{yes}}, \mathcal{X}^n_{\text{no}} \subseteq \mathcal{X}$ to denote the union of polytopes from S^n_{yes} and S^n_{no}, respectively.

Theorem 1 (*Progress*). *For every $n \in \mathbb{N}$, it holds that $\mathcal{X}^n_{\text{yes}} \subseteq \mathcal{X}^{n+1}_{\text{yes}}$ and $\mathcal{X}^n_{\text{no}} \subseteq \mathcal{X}^{n+1}_{\text{no}}$.*

Proof. Follows from Propositions 2 and 4, and the fact that the partition of the state space $\mathcal{X}$ used in $n+1$th iteration is a subpartition of the one used in nth iteration. □

Theorem 2 (Soundness). *For every $n \in \mathbb{N}$, it holds that $\mathcal{X}^n_{\text{yes}} \subseteq \mathcal{X}_{\text{init}}$ and $\mathcal{X}^n_{\text{no}} \subseteq \mathcal{X} \setminus \mathcal{X}_{\text{init}}$.*

Proof. Follows directly from Propositions 3 and 5. □

Theorem 3 (Partial Correctness). *If the algorithm from Section 4 terminates, after nth iteration, then* $\mathcal{X}_{\text{init}} = \mathcal{X}^n_{\text{yes}}$ *is the solution of Problem 1 and the corresponding satisfying strategies for every* $x \in \mathcal{X}_{\text{init}}$ *are given by the winning strategies in the* $2\frac{1}{2}$*-player game from the last iteration.*

Proof. Follows directly from the condition of the algorithm termination and from Theorems 1 and 2. □

It is important to note that if instead of a $2\frac{1}{2}$-player game a weaker abstraction model such as a 2-player game, *i.e.,* the NTS from Section 4.1, was used, our approach would not be sound. Namely, some states of $\mathcal{X}$ might be wrongfully identified as non-satisfying initial states based on behavior that has zero probability in the original stochastic system. In such a case, even after termination, the resulting set would only be a subset of $\mathcal{X}_{\text{init}}$. Therefore, the approach with 2-player games is not complete. The use of a 2-player game as the abstraction model leads to robust analysis of the stochastic system rather than almost-sure analysis. We also discussed the difference between robust and almost-sure analysis in Section 1. The $2\frac{1}{2}$-player game is needed to account for both the non-determinism introduced by the abstraction and for the stochasticity of the system to be able to recognize (non-satisfying) behavior of zero probability.

Note that there exist linear stochastic systems for which our algorithm does not terminate, *i.e.,* there does not exist a finite partition of the systems' state space over which Problem 1 can be solved for a given GR(1) formula.

Example 4 (Non-Termination). Let $\mathcal{T}$ be a linear stochastic system of the form given in Eq. (4), where A and B are identity matrices of size 2, state space $\mathcal{X} = \{x \in \mathbb{R}^2 \mid 0 \leq x(1), x(2) \leq 3\}$, control space $\mathcal{U} = \{u \in \mathbb{R}^2 \mid -1.5 \leq u(1), u(2) \leq 1.5\}$ and the random vector takes values in polytope $\mathcal{W} = \{w \in \mathbb{R}^2 \mid -0.5 \leq w(1), w(2) \leq 0.5\}$. Let Π contain four linear predicates that partition the state space into a grid of three by three equally sized square polytopes. Assume that the aim is to eventually reach the polytope $\mathcal{X}_f$, where $1 \leq x(1), x(2) \leq 2$. In this case, the maximal set $\mathcal{X}_{\text{init}}$ of states from which $\mathcal{X}_f$ can be reached with probability 1 is the whole state space $\mathcal{X}$, as for any $x \in \mathcal{X}$, there exists exactly one control input $u = (1.5, 1.5) - x \in \mathcal{U}$ that leads the system $\mathcal{T}$ from x to a state in $\mathcal{X}_5$ with probability 1. Since the control input is different for every $x \in \mathcal{X}$, there does not exist any finite state space partition, which could be used to solve Problem 1.

Complexity analysis. Finally, let us analyze the computational complexity of the designed algorithm. In the abstraction part, the $2\frac{1}{2}$-player game $\mathcal{G}_{\{\mathcal{X}_i\}}$ requires to first compute the set of actions for every state $\mathcal{X}_i$, $i \in I$, in time in $\mathcal{O}(2^{|I|})$ using algorithm in Appendix B.1. For every action $\mathcal{U}_i^j$, the set of valid supports $J' \subseteq J$ is then computed in time in $\mathcal{O}(2^{|J|})$, see Appendix B. Overall, the abstraction runs in time in $\mathcal{O}(2^{2\cdot|I|})$. The game is then analyzed using the algorithm described in Appendix A in time in $\mathcal{O}(|I|^3)$. Finally, the refinement procedure iteratively refines every polytope $\mathcal{X}_i$ at most $|Q| \times |\{\mathcal{U}_i^j\}|$ times, where $\{\mathcal{U}_i^j\}$ denotes the set of all actions of $\mathcal{X}_i$. For every $q \in Q$ such that $(\mathcal{X}_i, q) \in S_?^{\mathcal{P}}$, $\mathcal{X}_i$ is first refined using the robust predecessor operator in time exponential in $|J^{q'}_{\text{yes}}|$. Then $\mathcal{X}_i$ is refined $|Y|$ times using the robust attractor operator in polynomial time. Negative refinement is performed again for every $q \in Q$ such that $(\mathcal{X}_i, q) \in S_?^{\mathcal{P}}$, using the attractor operator in polynomial time. Overall, the refinement runs in time in $\mathcal{O}(|Q| \cdot 2^{|I|})$ and the size of the state space partition grows linearly with respect to the number of Player 1 states and actions of the game.

As the game construction is the most expensive part of the overall algorithm, the refinement procedure is designed in a way that extends both sets $S_{\text{yes}}, S_{\text{no}}$ as much as possible and thus speeds up convergence and minimizes the number of iterations of the overall algorithm.

5. Case study: abstraction-refinement

We demonstrate the designed framework on a discrete-time double integrator dynamics with uncertainties. Let $\mathcal{T}$ be a linear stochastic system of the form given in Eq. (4), where

$$A = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \qquad B = \begin{pmatrix} 0.5 \\ 1 \end{pmatrix}. \tag{17}$$

The state space is $\mathcal{X} = \{x \in \mathbb{R}^2 \mid -5 \leq x(1) \leq 5, -3 \leq x(2) \leq 3\}$ and the control space is $\mathcal{U} = \{u \in \mathbb{R} \mid -1 \leq u \leq 1\}$. The random vector, or uncertainty, takes values within polytope $\mathcal{W} = \{w \in \mathbb{R}^2 \mid -0.1 \leq w(1), w(2) \leq 0.1\}$. The set Π consists of 4 linear predicates $\pi_1 : x(1) \leq -1, \pi_2 : x(1) \leq 1, \pi_3 : x(2) \leq -1, \pi_4 : x(2) \leq 1$. The initial partition of the state space $\mathcal{X}$ together with the rest of the one-step reachable set $\mathcal{X}_{\text{out}}$ is depicted in Fig. 6 on top left.

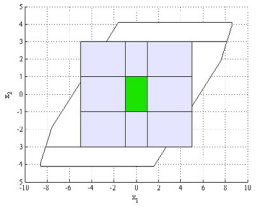
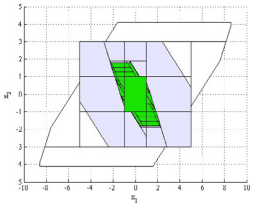
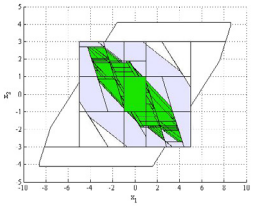
We consider GR(1) formula

$$\mathbf{F}(\neg\pi_1 \wedge \pi_2 \wedge \neg\pi_3 \wedge \pi_4) \tag{18}$$

that requires the system to eventually reach a state, where both variables of the system have values in interval $(-1, 1)$. On top left of Fig. 6, the corresponding area is shown in green.

We implemented the abstraction-refinement algorithm from Section 4 in Matlab and the game algorithm from Appendix A that is used in the game analysis phase was implemented in Java. As discussed in Section 4.4, the size of the game in one iteration of the algorithm grows linearly with respect to the number of Player 1 states and actions in the previous iteration. This leads to rapid increase of running times between iterations. To partially overcome this computational

Initial partition		First iteration		Second iteration	
time:	3 sec.	time:	2 min 40 sec	time:	16 min 24 sec
game size:	13 / 25	game size:	67 / 479	game size:	156 / 1405

Third iteration		Fourth iteration		Fifth iteration	
time:	31 min 8 sec	time:	50 min 28 sec	time:	63 min 10 sec
game size:	238 / 2272	game size:	292 / 3550	game size:	342 / 4828

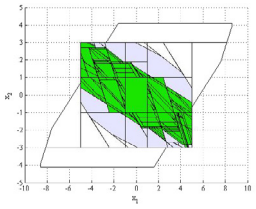
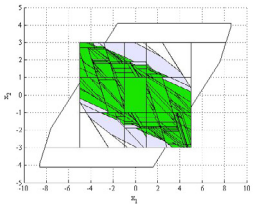
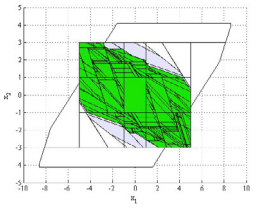




Fig. 6. Running times and results of a sample simulation of the abstraction-refinement algorithm from Section 4 for the double integrator example. We depict the results for the initial partition and the next five iterations, where polytopes from sets S_{yes}, $S_?$, S_{no} are shown in green, light blue and white, respectively. The size of the game indicates the number of states and actions. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

overhead, we implemented the following techniques on top of the refinement heuristic from Section 4.3. In positive refinement, when partitioning a polytope $\mathcal{X}_i \in S_?$ using robust attractors, instead of considering all suitable actions of $\mathcal{X}_i$ as described in Section 4.3, we only partition $\mathcal{X}_i$ with respect to three such actions chosen at random. Also, the refinement is implemented in a way that the minimum size of all partition elements in the resulting state space partition is at least 0.04.

Using the implementation, we computed the initial game and the following five iterations of the abstraction-refinement algorithm on a dual-core Intel i7 processor with 8 GB of RAM. The results together with running times are summarized in Fig. 6. In every iteration, a satisfying strategy for a state x in the partial solution is constructed as described in the proof of Proposition 3.

6. Reachability analysis

As discussed in Section 4.4, every iteration of the abstraction-refinement algorithm for solving Problem 1 takes time exponential in the size of the current state space partition in both abstraction and refinement phase. The case study in Section 5 demonstrates this rapid increase in running times in between iterations for a reachability property. It is important to note that in the case of reachability, the computation is simplified comparing to the general GR(1) specification as one can skip the construction of the corresponding ω-automaton and the product game and perform the game analysis and refinement directly on the game itself. Hence, for a more complex temporal property, the computation times would increase even more drastically.

In this section, we address this issue and propose an alternative solution to Problem 1 for the class of reachability properties, formulated as Problem 2. First, we present a simple algorithm that can be used to compute the set of all satisfying initial states $\mathcal{X}_{\text{init}}$, but cannot provide any information about the corresponding satisfying strategies. Second, we extend this algorithm to provide a partial information about the satisfying strategies. Finally, we combine the latter algorithm with the abstraction-refinement algorithm from Section 4 to obtain a full solution to Problem 2.

6.1. Computing satisfying initial states only

To compute the set of satisfying initial states, we consider an algorithm that is an extension of the reachability algorithm for Markov decision processes [2] to linear stochastic systems. Intuitively, the algorithm finds the set $\mathcal{X}_{\text{init}}$ using two fixed-point computations. The first one computes the set of all states that can reach the given target polytopes with non-zero

Algorithm 1 Computing the set $\mathcal{X}_{\text{init}} \subseteq \mathcal{X}$ of states from which a set of target polytopes $\{\mathcal{X}_k\}_{k\in K}$ in $\mathcal{X}$ can be reached with probability 1.

Input: linear stochastic system $\mathcal{T}$, target polytopes $\{\mathcal{X}_k\}_{k\in K}$
1: $\mathcal{X}_{>0} \leftarrow \bigcup_{k\in K} \mathcal{X}_k$
2: **while** $\mathcal{X}_{>0}$ is not a fixed point **do**
3: $\quad \mathcal{X}_{>0} \leftarrow \mathcal{X}_{>0} \cup \text{Pre}(\mathcal{X}, \mathcal{U}, \mathcal{X}_{>0})$
4: **end while**
5: $\mathcal{X}_{=0,\text{attr}} \leftarrow \mathcal{X}_{\text{out}} \cup \mathcal{X} \setminus \mathcal{X}_{>0}$
6: **while** $\mathcal{X}_{=0,\text{attr}}$ is not a fixed point **do**
7: $\quad \mathcal{X}_{=0,\text{attr}} \leftarrow \mathcal{X}_{=0,\text{attr}} \cup \text{Attr}(\mathcal{X}, \mathcal{U}, \mathcal{X}_{=0,\text{attr}})$
8: **end while**
9: $\mathcal{X}_{\text{init}} \leftarrow \mathcal{X} \setminus \mathcal{X}_{=0,\text{attr}}$
Output: $\mathcal{X}_{\text{init}}$

probability using the predecessor operator. As a result, the remaining states of the state space $\mathcal{X}$ have zero probability of reaching the target polytopes. The second fixed-point computation finds the attractor of this set, *i.e.,* all states that have non-zero probability, under each control input from $\mathcal{U}$, of ever transiting to a state from which the target polytopes cannot be reached. Finally, the complement of the attractor is the desired set $\mathcal{X}_{\text{init}}$. The algorithm is summarized in Algorithm 1.

Corollary 1. *If the while loop on line* 2 *of Algorithm* 1 *terminates, then $\mathcal{X}_{>0}$ is the set of all $x \in \mathcal{X}$ for which there exists a control strategy C such that the probability that a trace starting in state x using C reaches a target polytope is greater than* 0.

Corollary 2. *If the while loop on line* 6 *of Algorithm* 1 *terminates, then $\mathcal{X}_{=0,\text{attr}}$ is the set that consists of $\mathcal{X}_{\text{out}}$ and all $x \in \mathcal{X}$ for which it holds that a trace starting in x under arbitrary strategy has a non-zero probability of reaching a state from which the probability of reaching the target polytopes under arbitrary strategy is* 0.

Theorem 4 (Partial Correctness)**.** *If Algorithm* 1 *terminates, then the resulting set $\mathcal{X}_{\text{init}}$ is the solution of Problem* 2.

Note that Algorithm 1 operates directly on the linear stochastic system. It performs polytopic operations only, and it involves neither abstraction nor building a product with an automaton or refinement. Both predecessor and attractor operators can be computed in time polynomial in the number of vertices of the considered polytopes and the attractor operator is exponential in the size of the index set K, see Appendix B. While computationally efficient, the algorithm has two serious drawbacks.

Firstly, Algorithm 1 computes the set of satisfying initial states of the system, but no satisfying strategies. In extreme cases, every state may need to use a different control input in order to reach polytopes computed during the fixed-point computations, as in Example 4. In order to extract a finite satisfying strategy (if there is one), these polytopes have to be partitioned to smaller polytopes so that a fixed input can be used in all states of the new polytope. This partitioning is exactly the refinement procedure that the abstraction-refinement algorithm from Section 4 performs when applied to reachability.

Secondly, unlike the abstraction-refinement algorithm from Section 4, Algorithm 1 cannot be used for more complex properties than reachability. For more complex formulas, the game needs to be constructed and the product of the game with the automaton for the formula needs to be considered, since a winning strategy may require memory and pure polytopic methods can only provide memoryless strategies.

6.2. Computing satisfying initial states and layers

We extend Algorithm 1 to compute not only the set of satisfying initial states $\mathcal{X}_{\text{init}}$, but also a partition of the state space $\mathcal{X}$ that can provide a partial information about the satisfying control strategies for states in $\mathcal{X}_{\text{init}}$. The algorithm is summarized in Algorithm 2 and it differs from Algorithm 1 only in lines 2, 5 and 10 that maintain the state space partition. Initially, the state space partition is any partition of $\mathcal{X}$ that respects the set of target polytopes $\{\mathcal{X}_k\}_{k\in K}$. It is then refined in every iteration of the two fixed-point computations with respect to the resulting predecessor or attractor set.

Just like Algorithm 1, Algorithm 2 operates directly on the linear stochastic system and it uses polytopic operators only. It also suffers from the same two serious drawbacks, *i.e.,* it does not compute the satisfying control strategies and can only be used for reachability properties. However, in comparison with Algorithm 1, it can provide at least a partial information on the properties of satisfying strategies. Namely, it is able to identify for a state $x \in \mathcal{X}_{\text{init}}$ some of the control inputs that cannot be used in any satisfying strategy starting from x. The control inputs are identified as follows.

Proposition 7. *Let $\mathcal{X}_{\text{init}}$ be the set and $\{\mathcal{X}_i\}_{i\in I}$ be the state space partition resulting from Algorithm* 2*. Let $\mathcal{G}_{\{\mathcal{X}_i\}}$ be the corresponding* $2\frac{1}{2}$*-player game constructed according to Section* 4.1*. Consider a Player* 1 *state $\mathcal{X}_i \subseteq \mathcal{X}_{\text{init}}$ and a Player* 1 *action $\mathcal{U}_i^J$ such that for every Player* 2 *action $J' \in \text{Supp}(\mathcal{X}_i, \mathcal{U}_i^J)$ there exists $j \in J'$ such that the Player* 1 *state $\mathcal{X}_j \not\subseteq \mathcal{X}_{\text{init}}$. Then it holds that any strategy C that applies a control input $u \in \mathcal{U}_i^J$ in states $x \in \mathcal{X}_i$ is not a satisfying strategy, i.e., the probability of reaching the target polytopes is* <1.

Algorithm 2 Computing the set $\mathcal{X}_{\text{init}} \subseteq \mathcal{X}$ of states from which a set of target polytopes $\{\mathcal{X}_k\}_{k\in K}$ in $\mathcal{X}$ can be reached with probability 1, and a partition of $\mathcal{X}_{\text{init}}$ according to the minimum number of steps to reach the target.

Input: linear stochastic system $\mathcal{T}$, target polytopes $\{\mathcal{X}_k\}_{k\in K}$
1: $\mathcal{X}_{>0} \leftarrow \bigcup_{k\in K} \mathcal{X}_k$
2: $\{\mathcal{X}_i\}_{i\in I}$ is an arbitrary partition of $\mathcal{X}$ that respects the target polytopes
3: **while** $\mathcal{X}_{>0}$ is not a fixed point **do**
4: $\quad \mathcal{X}_{>0} \leftarrow \mathcal{X}_{>0} \cup \text{Pre}(\mathcal{X}, \mathcal{U}, \mathcal{X}_{>0})$
5: $\quad$ refine $\{\mathcal{X}_i\}_{i\in I}$ according to $\mathcal{X}_{>0}$
6: **end while**
7: $\mathcal{X}_{=0,\text{attr}} \leftarrow \mathcal{X}_{\text{out}} \cup \mathcal{X} \setminus \mathcal{X}_{>0}$
8: **while** $\mathcal{X}_{=0,\text{attr}}$ is not a fixed point **do**
9: $\quad \mathcal{X}_{=0,\text{attr}} \leftarrow \mathcal{X}_{=0,\text{attr}} \cup \text{Attr}(\mathcal{X}, \mathcal{U}, \mathcal{X}_{=0,\text{attr}})$
10: $\quad$ refine $\{\mathcal{X}_i\}_{i\in I}$ according to $\mathcal{X}_{=0,\text{attr}}$
11: **end while**
12: $\mathcal{X}_{\text{init}} \leftarrow \mathcal{X} \setminus \mathcal{X}_{=0,\text{attr}}$
Output: $\mathcal{X}_{\text{init}}$ and $\{\mathcal{X}_i\}_{i\in I}$

Proof. From Eq. (6) it follows that for every $u \in \mathcal{U}_i^l$ there exists $x \in \mathcal{X}_i$ such that after applying control input u in state x the system $\mathcal{T}$ is with non-zero probability in a state in $\mathcal{X}_j$. As $\mathcal{X}_j \not\subseteq \mathcal{X}_{\text{init}}$, the target polytopes can no longer be reached with probability 1. □

We use the following notation for the partition $\{\mathcal{X}_i\}_{i\in I}$ resulting from Algorithm 2. For $n \geq 0$, let $\mathcal{X}_{>0}^n$ denote the set $\mathcal{X}_{>0}$ after nth iteration of the while loop on line 3 of Algorithm 2. It holds that $\mathcal{X}_{>0}^0$ is the union of the target polytopes and if $\mathcal{X}_{\text{init}} \not\subseteq \mathcal{X}_{>0}^n$ then $\mathcal{X}_{>0}^n \subset \mathcal{X}_{>0}^{n+1}$. For $n \geq 1$, we refer to the set

$$\mathcal{L}_n = (\mathcal{X}_{>0}^n \setminus \mathcal{X}_{>0}^{n-1}) \setminus \mathcal{X}_{=0,\text{attr}} \tag{19}$$

as the nth layer of the partition $\{\mathcal{X}_i\}_{i\in I}$ and set $\mathcal{L}_0 = \bigcup_{k\in K} \mathcal{X}_k$.

Proposition 8. *For every $n \geq 1$ and every $x \in \mathcal{L}_n$, there exists a control input $u_1 \in \mathcal{U}$ such that*

$$\text{Post}(x, u_1) \cap \mathcal{X}_{>0}^{n-1} \neq \emptyset \tag{20}$$

and there exists a control input $u_2 \in \mathcal{U}$ such that

$$\text{Post}(x, u_2) \cap \mathcal{X}_{=0,\text{attr}} = \emptyset. \tag{21}$$

Proof. The property in Eq. (20) is given by the fact that $x \in \mathcal{X}_{>0}^n = \mathcal{X}_{>0}^{n-1} \cup \text{Pre}(\mathcal{X}, \mathcal{U}, \mathcal{X}_{>0}^{n-1})$ and property in Eq. (21) follows from the fact that $x \notin \mathcal{X}_{=0,\text{attr}}$. □

6.3. The combined algorithm for Problem 2

In this section, we present an alternative solution to Problem 2 and argue that in general, it performs better than the abstraction-refinement proposed in Section 4.

The algorithm combines Algorithm 2 with the abstraction-refinement algorithm as follows. First, Algorithm 2 is used to compute the set $\mathcal{X}_{\text{init}}$ and its partition into layers. A satisfying control strategy for a state $x \in \mathcal{X}_{\text{init}}$ is then constructed as follows. Let $n \geq 1$ be such that x belongs to the nth layer, *i.e.*, $x \in \mathcal{L}_n$. We use the abstraction-refinement algorithm from Section 4 to solve Problem 2 for the linear stochastic system $\mathcal{T}$ and $\{\mathcal{L}_k\}_{0\leq k\leq n-1}$ as the set of target polytopes. The initial state space partition is the coarsest partition of $\mathcal{X}$ that respects the nth layer and the union of target polytopes. Note that we can terminate the abstraction-refinement algorithm as soon as all states in the nth layer have been recognized as satisfying initial states and the corresponding strategy has been computed that leads from the nth layer to lower layers with probability 1. Apply this strategy starting from x until a state x' in mth layer is reached, $m < n$. Restart the above procedure with the state x'.

Theorem 5 (Partial Correctness). *If the combined algorithm presented above terminates, then $\mathcal{X}_{\text{init}}$ is the solution of Problem 2 for the class of reachability properties and the computed strategies are satisfying.*

The computational complexity of the combined algorithm above is in the worst case the same as the complexity of the abstraction-refinement algorithm. However, the combined algorithm divides the original instance of Problem 2 into a series of instances of Problem 2 over smaller state spaces. These can be solved in parallel and the computation can be terminated early, before convergence of the abstraction-refinement algorithm. Therefore, it is reasonable to conclude that the combined

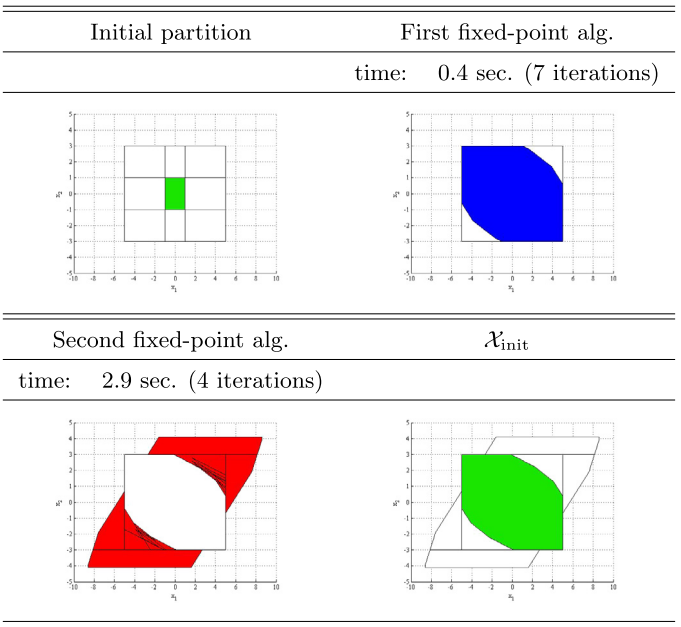


Fig. 7. Simulation of Algorithm 1 for the double integrator example. We depict the initial partition of $\mathcal{X}$ according to Π with the polytope we aim to reach in green. The results for the two fixed-point computations are presented next with running times and the number of iterations needed to find the fixed points. Finally, we depict the computed set of satisfying initial states $\mathcal{X}_{\text{init}}$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

algorithm is more efficient in practice than the abstraction-refinement. We demonstrate in the following section that it indeed can lead to significantly reduced running times.

7. Case study: reachability analysis and comparison

In this section, we demonstrate the usability of the three algorithms from Section 6 on the case study of a double integrator introduced in Section 5 and compare their efficiency with the abstraction-refinement algorithm from Section 4. The algorithms were implemented in Matlab and simulated on a dual-core Intel i7 processor with 8 GB of RAM.

The results obtained for Algorithm 1 are summarized in Fig. 7. The set $\mathcal{X}_{\text{init}}$ was computed fast but it is a single polytope that does not provide any information about the corresponding satisfying strategies.

The results obtained for Algorithm 2 are presented in Fig. 8. As Algorithm 2 is a simple extension of Algorithm 1, the number of iterations for each of the two fixed-point computations is the same. However, maintaining the state space partition requires additional time. The resulting set of satisfying initial states $\mathcal{X}_{\text{init}}$ is partitioned into 7 layers as explained in Section 6.2. Given the partition, we constructed the game with 110 states and 1617 actions in 10 min 41 s. Out of the 110 states, *i.e.,* elements $\mathcal{X}_i$ of the state space partition $\{\mathcal{X}_i\}_{i\in I}$, 57 form the satisfying initial set $\mathcal{X}_{\text{init}}$. The maximum number of actions ruled out for such a state $\mathcal{X}_i$ was 17, on average 6 actions per state can be ruled out. The remaining 53 polytopes contain only non-satisfying initial states from $\mathcal{X}$.

Finally, the results obtained for the combined algorithm from Section 6.3 are presented in Fig. 10. First, Algorithm 2 was used to compute the set $\mathcal{X}_{\text{init}}$ together with its partition into 7 layers. For every layer $2 \leq n \leq 7$, we used the abstraction-refinement algorithm to find the control strategies that lead from nth layer to the lower layers with probability 1. The algorithm was terminated once the size of every partition element within layer i that was recognized as undecided, *i.e.,* element of the set $S_?$, was below 0.04 and we used the heuristics described in Section 5 to avoid computational overhead during refinement phase. Note that the computations for individual layers can be done in parallel.

Comparing these results to the results from Section 5, where the abstraction-refinement algorithm was used to solve the full original problem, the overall running time of the combined algorithm is comparable to the one of the abstraction-refinement. The main reason is that to compute the strategies leading from layer 2 to layer 1, a large part of the state space $\mathcal{X}$ needed to be recognized as satisfying initial states, in 4 iterations of the abstraction-refinement, before (almost) all states of layer 2 were recognized as satisfying initial states.

On the other hand, consider the modification of the double integrator example from Section 5, where the set Π consists of 4 linear predicates $\pi_1 : x(1) + x(2) \leq -1, \pi_2 : x(1) + x(2) \leq 1, \pi_3 : x(2) \leq -1, \pi_4 : x(2) \leq 1$. For convenience, we depict the results we obtained for the modified double integrator using Algorithm 2 in Fig. 9. Note that comparing to the original double integrator example, the shape of the target region is in this case more in correspondence with the dynamics of the linear stochastic system. Next, we simulated the abstraction-refinement algorithm for the modified double integrator

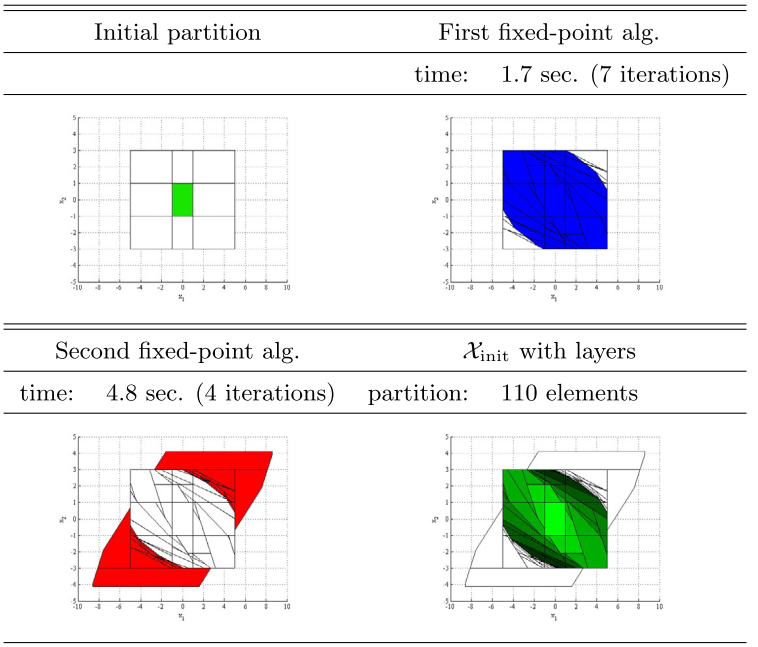


Fig. 8. Simulation of Algorithm 2 for the double integrator example. We depict the initial partition of $\mathcal{X}$ according to Π with the polytope we aim to reach in green. The results for the two fixed-point computations are presented next together with running times and the number of iterations needed to find the fixed points. Finally, we depict the computed set of satisfying initial states $\mathcal{X}_{\text{init}}$ with 7 layers shown in different shades of green. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

example which leads to running times and game sizes comparable to the ones obtained for the original double integrator in Section 5:

initial partition : 5sec. game had 13 states and 37 actions,
iteration 1 : 7 min. 31 sec., game had 78 states and 663 actions,
iteration 2 : 19 min. 5 sec., game had 158 states and 1581 actions,
iteration 3 : 27 min. 12 sec., game had 223 states and 2612 actions,
iteration 4 : 40 min. 36 sec., game had 282 states and 3403 actions,
iteration 5 : 65 min. 17 sec., game had 346 states and 4828 actions.

Finally, the results obtained using the combined algorithm are summarized in Fig. 11. In this case, the maximum number of iterations of the abstraction-refinement algorithm needed for each layer was only 2 leading to overall computation time considerably lower than the one of the abstraction-refinement.

Overall, we conclude that the combined algorithm presented in Section 6.3 provides an alternative solution to Problem 2 for the class of reachability properties that can lead to significant decrease of running time if the set of target polytopes has favorable shape with respect to the dynamics of the system. Otherwise, it provides results in time comparable to the abstraction-refinement algorithm from Section 4 which is applicable to a wider class of temporal properties.

8. Conclusion and future work

In this work, we consider the problem of computing the set of initial states of a linear stochastic system such that there exists a control strategy to ensure a GR(1) specification over states of the system. The solution is based on iterative abstraction-refinement using a $2\frac{1}{2}$-player game. Every iteration of the algorithm provides a partial solution given as a set of satisfying initial states with the satisfying strategies, and a set of non-satisfying initial states. While the proposed algorithm guarantees progress and soundness in every iteration, it is computationally demanding.

For the class of reachability properties, two more efficient algorithms are presented that compute the set of satisfying initial states in a fast manner but provide no or only partial information about the corresponding satisfying strategies, respectively. Finally, an algorithm combining the latter algorithm with the abstraction-refinement algorithm is considered. While the theoretical computational complexity remains the same as for the abstraction-refinement, it can lead to a considerable speed up as demonstrated on a case study.

As for future work, the abstraction-refinement approach could be considered for a more general class of temporal properties such as LTL or general ω-regular properties. In fact, the algorithm can be applied in a straightforward way in these cases, provided that an appropriate automaton model and game solving algorithm are used.

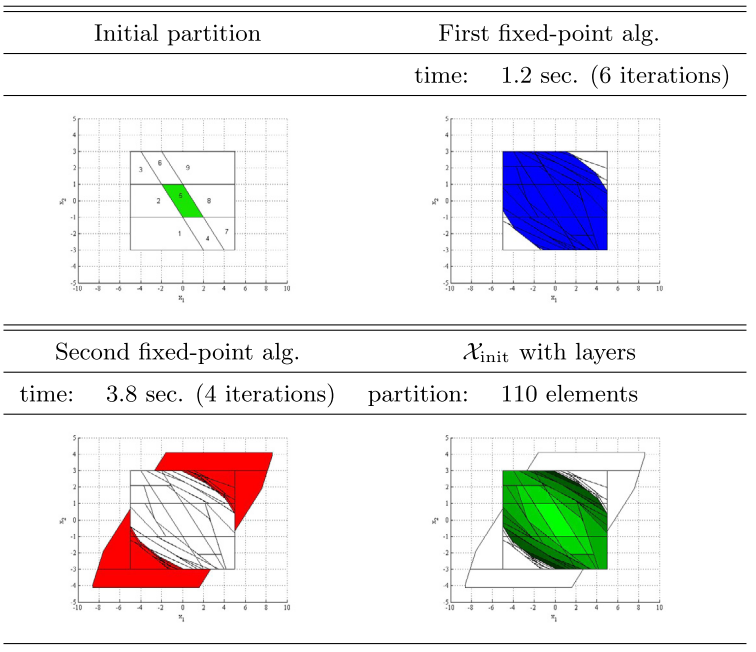


Fig. 9. Simulation of Algorithm 2 for the double integrator example with modified linear predicates. We depict the initial partition of $\mathcal{X}$ according to Π, with the polytope we aim to reach in green. The results for the two fixed-point computations are presented next together with running times and the number of iterations needed to find the fixed points. Finally, we depict the computed set of satisfying initial states $\mathcal{X}_{\text{init}}$ with 6 layers shown in different shades of green. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

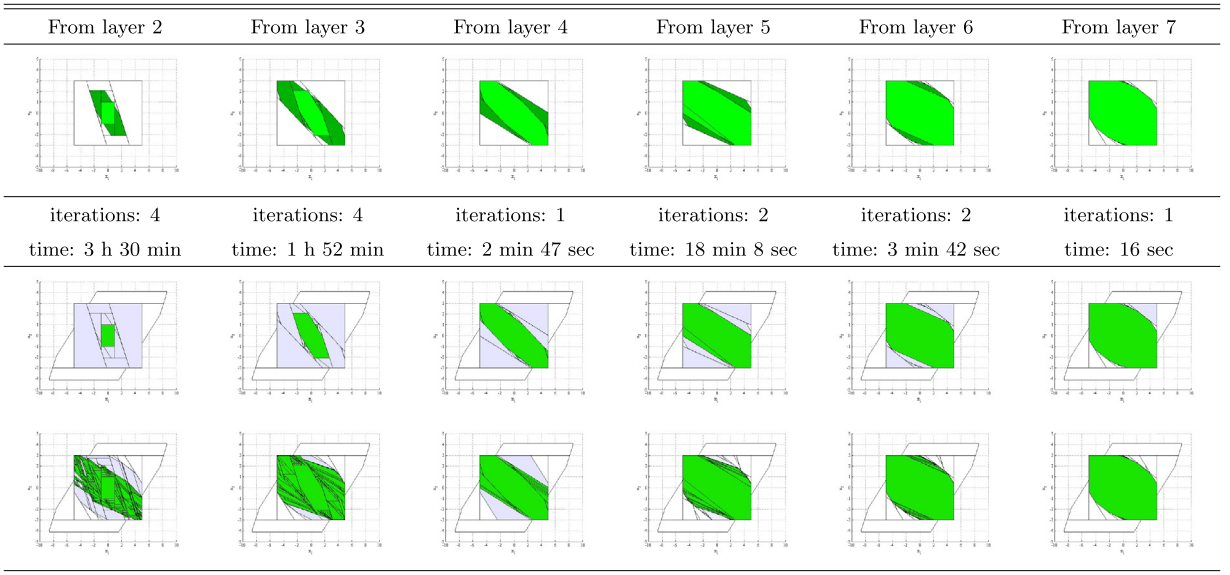


Fig. 10. Simulation of the combined algorithm from Section 6 for the double integrator example. We depict the results obtained from the use of the abstraction-refinement algorithm to compute the control strategies that lead from one layer to the lower layers. For every layer 2 to 7, on the top we depict the (coarsest) initial state space partition, where the target polytope that is the union of all lower layers is shown in light green and the current layer is in dark green. We applied the abstraction-refinement algorithm until all states in the current layer were recognized as satisfying initial states and the corresponding strategies have been computed. We depict the state space partition after the first and the last iteration, where polytopes from sets S_{yes}, $S_{?}$, S_{no} are shown in green, light blue and white, respectively. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Furthermore, while we can prove progress between individual iterations of the abstraction-refinement algorithm, see Theorem 1, convergence and termination of the whole procedure are not guaranteed. An interesting direction for future work is to study in more detail the conditions for convergence and techniques for efficient computation of the limit. For example,

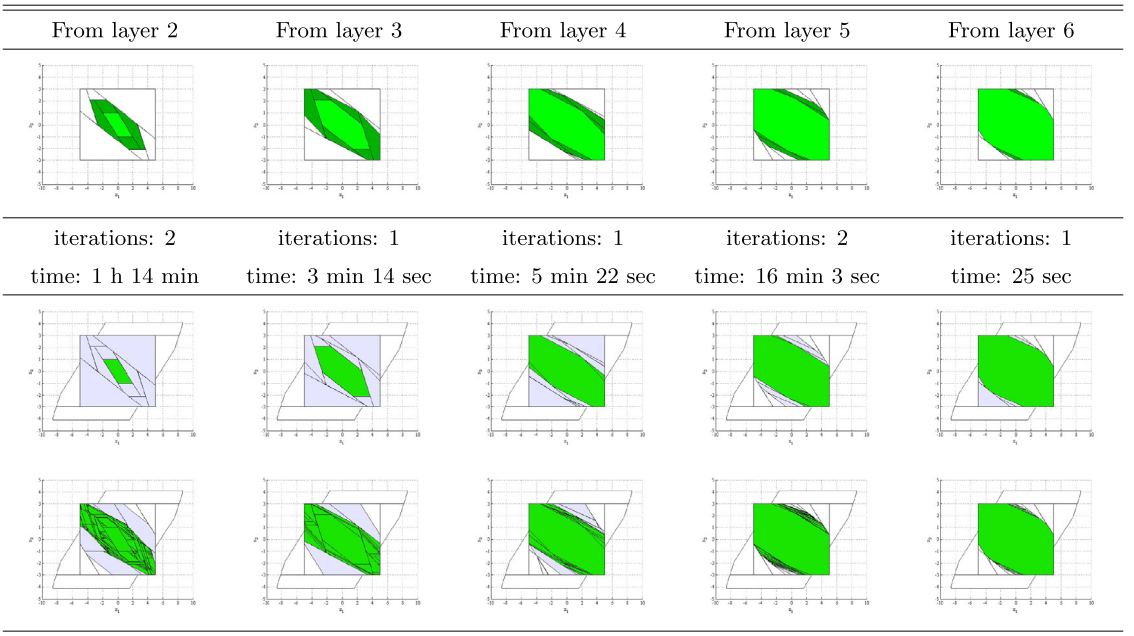


Fig. 11. Simulation of the combined algorithm from Section 6 for the double integrator example with modified linear predicates. We depict the results obtained from the use of the abstraction-refinement algorithm to compute the control strategies that lead from one layer to the lower layers. For every layer 2 to 6, on the top we depict the (coarsest) initial state space partition, where the target polytope that is the union of all lower layers is shown in light green and the current layer is in dark green. We applied the abstraction-refinement algorithm until all states in the current layer were recognized as satisfying initial states and the corresponding strategies have been computed. We depict the state space partition after the first and the last iteration, where polytopes from sets S_{yes}, $S_?$, S_{no} are shown in green, light blue and white, respectively. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

the authors in [38] study a general class of value improvement algorithms for discrete probabilistic models such as Markov chains, Markov decision processes or $2^1/_2$-player games, and show that under certain assumptions, such an algorithm is guaranteed to converge but not necessarily in finite time. Nevertheless, the limit value can still be computed in a finite number of iterations by terminating the algorithm as soon as the improvement is below a pre-computed threshold and then using simple rounding. It would be very interesting to investigate whether an analogous approach can be used in our scenario too. That means, does there exist a parameter such that if the improvement of the set of satisfying initial states and non-satisfying initial states is bounded by the parameter, the algorithm can be terminated and the remaining unidentified states decided using a simple rule?

Appendix A. Solving a $2^1/_2$-player game

Here we present an algorithm to solve the almost-sure winning problem for a $2^1/_2$-player game $\mathcal{G} = (S_1, S_2, Act, \delta)$ with a Büchi implication condition (E, F), where $E, F \subseteq S$. The optimal solution is a rather involved, quadratic time algorithm that can be found in [36]. In this work, we use a more intuitive, cubic time algorithm presented in Algorithm 3, whose correctness follows from [37]. The algorithm is a simple iterative fixed-point algorithm that uses three types of local predecessor operator over the set of states of the game.

Consider sets X, Y, Z such that $Y \subseteq Z \subseteq X \subseteq S$. Given a state $s \in S$ and an action $a \in Act$, we denote by $\text{Succ}(s, a) = \text{Supp}(\delta(s, a))$ the set of possible successors of the state and the action. We define conditions on state action pairs as follows:

$$\begin{aligned}
\mathrm{C}_1(X) &= \{(s, a) \mid \text{Succ}(s, a) \subseteq X\},\\
\mathrm{C}_2(X, Y) &= \{(s, a) \mid \text{Succ}(s, a) \subseteq X \text{ and } \text{Succ}(s, a) \cap Y \neq \emptyset\},\\
\mathrm{C}_3(Z, X, Y) &= \{(s, a) \mid (\text{Succ}(s, a) \subseteq Z) \text{ or } (\text{Succ}(s, a) \subseteq X \text{ and } \text{Succ}(s, a) \cap Y \neq \emptyset)\}.
\end{aligned}$$

The first condition ensures that given the state and action the next state is in U with probability 1, the second condition ensures that the next state is in X with probability 1 and in Y with positive probability. The third condition is the disjunction of the first two. The three predecessor operators are defined as the set of Player 1, or Player 2 states, where there exists, or for all, respectively, actions, the condition for the predecessor operator is satisfied. The three respective predecessor operators,

Algorithm 3 Algorithm for $\mathsf{Almost}^{\mathcal{G}}(\varphi)$

Input: game $\mathcal{G}$, accepting condition (E, F), $D = S \setminus (E \cup F)$

```
Set: X, Y, Z, X̄, Ȳ, Z̄;
X̄ ← S    Z̄ ← S    Ȳ ← ∅                                   ▷ Initialization
do
    X ← X̄
    do
        Y ← Ȳ
        do
            Z ← Z̄
            Z̄ ← (F ∩ Pre1(X)) ∪ (E ∩ Pre2(X, Y)) ∪ (D ∩ Pre3(Z, X, Y))
        while Z ≠ Z̄
        Ȳ ← Z    Z̄ ← S
    while Y ≠ Ȳ
    X̄ ← Y    Ȳ ← ∅
while X ≠ X̄
```

Output: X

namely, Pre_1, Pre_2, and Pre_3 are defined as follows:

$$\begin{aligned}
\mathsf{Pre}_1(X) &= \{s \in S_1 \mid \exists a \in Act.\ (s,a) \in \mathsf{C}_1(X)\} \cup \{s \in S_2 \mid \forall a \in Act.\ (s,a) \in \mathsf{C}_1(X)\},\\
\mathsf{Pre}_2(X,Y) &= \{s \in S_1 \mid \exists a \in Act.\ (s,a) \in \mathsf{C}_2(X,Y)\} \cup \{s \in S_2 \mid \forall a \in Act.\ (s,a) \in \mathsf{C}_2(X,Y)\},\\
\mathsf{Pre}_3(Z,X,Y) &= \{s \in S_1 \mid \exists a \in Act.\ (s,a) \in \mathsf{C}_3(Z,X,Y)\} \cup \{s \in S_2 \mid \forall a \in Act.\ (s,a) \in \mathsf{C}_3(Z,X,Y)\}.
\end{aligned}$$

Once the almost-sure winning set $\mathsf{Almost}^{\mathcal{G}}((E,F))$ is computed using Algorithm 3, an almost-sure winning strategy for Player 1 can be constructed as follows. For every almost-sure winning Player 1 state the strategy applies, in a round-robin fashion, all actions that remain within the set $\mathsf{Almost}^{\mathcal{G}}((E,F))$ with probability 1. The strategy is pure, but not memoryless. A pure memoryless strategy for Player 1 can be constructed using deeper analysis of the game [18].

Appendix B. Polytopic operators

In this section, we describe in detail the computation of all polytopic operators introduced in Section 4 and used in our solution to Problem 1.

B.1. Action polytopes

First, we describe how to compute the action polytopes $\mathcal{U}_i^J$ for every polytope $\mathcal{X}_i \in \{\mathcal{X}_i\}_{i\in I}$, formally defined in Eq. (6).

For a polytope $\mathcal{X}' \subset \mathbb{R}^N$, we use $\mathcal{U}^{\mathcal{X}_i \to \mathcal{X}'}$ to denote the set of all control inputs from $\mathcal{U}$ under which the system $\mathcal{T}$ can evolve from a state in $\mathcal{X}_i$ to a state in $\mathcal{X}'$ with non-zero probability, *i.e.,*

$$\mathcal{U}^{\mathcal{X}_i \to \mathcal{X}'} = \{u \in \mathcal{U} \mid \mathrm{Post}(\mathcal{X}_i, u) \cap \mathcal{X}' \text{ is non-empty}\}. \tag{B.1}$$

The following proposition states that $\mathcal{U}^{\mathcal{X}_i \to \mathcal{X}'}$ can be computed from the V-representations of $\mathcal{X}_i$, $\mathcal{X}'$ and $\mathcal{W}$.

Proposition 9. *Let H, K be the matrices from the H-representation of the following polytope:*

$$\{y \in \mathbb{R}^N \mid \exists x \in \mathcal{X}_i, \exists w \in \mathcal{W} : Ax + y + w \in \mathcal{X}'\}, \tag{B.2}$$

which can be computed as the convex hull

$$\mathrm{hull}(\{v_{\mathcal{X}'} - (Av_{\mathcal{X}_i} + v_{\mathcal{W}}) \mid v_{\mathcal{X}'} \in V(\mathcal{X}'), v_{\mathcal{X}_i} \in V(\mathcal{X}_i), v_{\mathcal{W}} \in V(\mathcal{W})\}). \tag{B.3}$$

Then the set $\mathcal{U}^{\mathcal{X}_i \to \mathcal{X}'}$ defined in Eq. (B.1) is the polytope with the following H-representation:

$$\mathcal{U}^{\mathcal{X}_i \to \mathcal{X}'} = \{u \in \mathcal{U} \mid HBu \leq K\}. \tag{B.4}$$

Proof. To fact that the set in Eq. (B.2) is a polytope with the V-representation given in Eq. (B.3) can be easily shown as follows. Let $y \in \mathbb{R}^N$ be such that there exist $x \in \mathcal{X}_i$, $w \in \mathcal{W}$, $x' \in \mathcal{X}'$ for which $Ax + y + w = x'$, *i.e.,* $y = x' - (Ax + w)$.

By representing x', x and w as an affine combination of the respective vertices in $V(\mathcal{X}')$, $V(\mathcal{X}_i)$ and $V(\mathcal{W})$, we obtain the V-representation in Eq. (B.3). Next, let H, K be the matrices from the H-representation of the set in Eq. (B.2). Then the definition of set $\mathcal{U}^{\mathcal{X}_i \to \mathcal{X}'}$ in Eq. (B.1) can be written as

$$\mathcal{U}^{\mathcal{X}_i \to \mathcal{X}'} = \{u \in \mathcal{U} \mid \exists x \in \mathcal{X}_i, \exists w \in \mathcal{W} : Ax + Bu + w \in \mathcal{X}'\},$$

that leads to H-representation in Eq. (B.4). □

Corollary 3. *Let $J \subseteq I \cup I_{\text{out}}$. The set $\mathcal{U}_i^J$ from Eq. (6) can be computed as follows:*

$$\mathcal{U}_i^J = \bigcap_{j \in J} \mathcal{U}^{\mathcal{X}_i \to \mathcal{X}_j} \setminus \bigcup_{j' \notin J} \mathcal{U}^{\mathcal{X}_i \to \mathcal{X}_{j'}}. \tag{B.5}$$

Proof. Follows directly from Eqs. (6) and (B.1). □

Note that $\mathcal{U}_i^J$ is generally not a polytope but can be represented as a finite union of polytopes.

B.2. Posterior

The posterior operator $\text{Post}(\mathcal{X}', \mathcal{U}')$, formally defined in Table 1, can be easily computed using Minkowski sum as

$$\begin{aligned}\text{Post}(\mathcal{X}', \mathcal{U}') &= A\mathcal{X}' + B\mathcal{U}' + \mathcal{W} \\ &= \text{hull}(\{Av_{\mathcal{X}'} + Bv_{\mathcal{U}'} + v_{\mathcal{W}} \mid v_{\mathcal{X}'} \in V(\mathcal{X}'), v_{\mathcal{U}'} \in V(\mathcal{U}'), v_{\mathcal{W}} \in V(\mathcal{W})\}).\end{aligned}$$

B.3. Predecessor

The predecessor operator $\text{Pre}(\mathcal{X}', \mathcal{U}', \{\mathcal{X}_j\}_{j \in J})$, formally defined in Table 1, can be computed as follows. First, note that

$$\text{Pre}(\mathcal{X}', \mathcal{U}', \{\mathcal{X}_j\}_{j \in J}) = \bigcup_{j \in J} \text{Pre}(\mathcal{X}', \mathcal{U}', \mathcal{X}_j).$$

Proposition 10. *Let H, K be the matrices from the H-representation of the following polytope:*

$$\{y \in \mathbb{R}^N \mid \exists u \in \mathcal{U}', \exists w \in \mathcal{W} : y + Bu + w \in \mathcal{X}_j\},$$

which can be computed as the convex hull

$$\text{hull}(\{v_{\mathcal{X}_j} - (Bv_{\mathcal{U}'} + v_{\mathcal{W}}) \mid v_{\mathcal{X}_j} \in V(\mathcal{X}_j), v_{\mathcal{U}'} \in V(\mathcal{U}'), v_{\mathcal{W}} \in V(\mathcal{W})\}).$$

Then the set $\text{Pre}(\mathcal{X}', \mathcal{U}', \mathcal{X}_j)$ is the polytope with the following H-representation:

$$\text{Pre}(\mathcal{X}', \mathcal{U}', \mathcal{X}_j) = \{x \in \mathcal{X}' \mid HAx \leq K\}.$$

Proof. The proof is analogous to the one of Proposition 9. □

B.4. Robust and precise predecessor

From definitions of the robust and precise predecessor operators in Table 1 it follows that

$$\text{PreR}(\mathcal{X}', \mathcal{U}', \{\mathcal{X}_j\}_{j \in J}) = \bigcup_{J' \subseteq J\, J' \neq \emptyset} \text{PreP}(\mathcal{X}', \mathcal{U}', \{\mathcal{X}_j\}_{j \in J'}).$$

Below we describe the computation of the precise predecessor $\text{PreP}(\mathcal{X}', \mathcal{U}', \{\mathcal{X}_j\}_{j \in J'})$ for any $J' \subseteq J$.

Let $\mathcal{Z}$ denote the polytope, or finite union of polytopes, $\mathcal{Z} = A\mathcal{X}' + B\mathcal{U}'$, where $+$ denotes the Minkowski sum. For a polytope $\mathcal{P} \subset \mathbb{R}^N$, we define set

$$\mathcal{Z}(\mathcal{P}) = \{z \in \mathcal{Z} \mid (z + \mathcal{W}) \cap \mathcal{P} \text{ is non-empty}\}. \tag{B.6}$$

For a set of polytopes $\{\mathcal{P}\}$, $\mathcal{Z}(\{\mathcal{P}\})$ can be computed as the union of all $\mathcal{Z}(\mathcal{P})$ for every polytope $\mathcal{P}$ in the set $\{\mathcal{P}\}$.

Proposition 11. *The set from Eq. (B.6) is the following polytope, or finite union of polytopes:*

$$\mathcal{Z}(\mathcal{P}) = \textit{hull}(\{v_{\mathcal{P}} - v_{\mathcal{W}} \mid v_{\mathcal{P}} \in V(\mathcal{P}), v_{\mathcal{W}} \in V(\mathcal{W})\}) \cap \mathcal{Z}. \tag{B.7}$$

Proof. The proof is carried out in a similar way as the first part of proof of Proposition 9. □

For $J' \subseteq J$, we use $\mathcal{Z}(J')$ to denote the set

$$\mathcal{Z}(J') = \bigcap_{j \in J'} \mathcal{Z}(\mathcal{X}_j) \setminus \left(\bigcup_{j \in J \setminus J'} \mathcal{Z}(\mathcal{X}_j) \cup \mathcal{Z}(\mathcal{X}_{\neg J}) \right), \tag{B.8}$$

where $\mathcal{Z}(\mathcal{X}_{\neg J}) = \mathcal{Z}((\mathcal{X} \cup \mathcal{X}_{\text{out}}) \setminus \bigcup_{j \in J} \mathcal{X}_j)$.

Proposition 12. *Let $\mathcal{U}' = \{\mathcal{U}_{l_1}\}_{l_1 \in L_1}$, $J \subseteq J'$ and let $\mathcal{Z}(J') = \{\mathcal{Z}_{l_2}\}_{l_2 \in L_2}$. Then the precise predecessor can be written as*

$$\text{PreP}(\mathcal{X}', \mathcal{U}', \{\mathcal{X}_j\}_{j \in J'}) = \bigcup_{l_1 \in L_1} \bigcup_{l_2 \in L_2} \{x \in \mathcal{X}' \mid \exists u \in \mathcal{U}_{l_1} : Ax + Bu \in \mathcal{Z}_{l_2}\}. \tag{B.9}$$

Let $l_1 \in L_1$, $l_2 \in L_2$ and let H, K be the matrices from the H-representation of the following polytope:

$$\{y \in \mathbb{R}^N \mid \exists u \in \mathcal{U}_{l_1} : y + Bu \in \mathcal{Z}_{l_2}\}, \tag{B.10}$$

which can be computed as the convex hull

$$\text{hull}(\{v_{\mathcal{Z}_{l_2}} - B v_{\mathcal{U}_{l_1}} \mid v_{\mathcal{Z}_{l_2}} \in V(\mathcal{Z}_{l_2}), v_{\mathcal{U}_{l_1}} \in V(\mathcal{U}_{l_1})\}). \tag{B.11}$$

Then the set on the right-hand side of Eq. (B.9), for l_1, l_2, is a polytope with the following H-representation:

$$\{x \in \mathcal{X}' \mid HAx \leq K\}. \tag{B.12}$$

Proof. From the definition of the set $\mathcal{Z}(J')$ in Eq. (B.8), $z \in \mathcal{Z}(J')$ iff $z + \mathcal{W}$ intersects all $\mathcal{X}_j$ for $j \in J'$ and $z + \mathcal{W} \subseteq \bigcup_{j \in J'} \mathcal{X}_j$. Moreover, every $z \in \mathcal{Z}$ can be written as $z = Ax + Bu$ and therefore $z + \mathcal{W} = \text{Post}(x, u)$. This proves Eq. (B.9). The rest of the proof is carried out in a way similar to the proof of Proposition 9. □

B.5. Attractor

The attractor operator $\text{Attr}(\mathcal{X}', \mathcal{U}', \{\mathcal{X}_j\}_{j \in J})$ from Table 1 can be computed using the robust predecessor operator, since it holds that

$$\begin{aligned}
\text{Attr}(\mathcal{X}', \mathcal{U}', \{\mathcal{X}_j\}_{j \in J}) &= \left\{ x \in \mathcal{X}' \mid \forall u \in \mathcal{U}' : \text{Post}(x, u) \cap \bigcup_{j \in J} \mathcal{X}_j \text{ is non-empty} \right\} \\
&= \mathcal{X}' \setminus \left\{ x \in \mathcal{X}' \mid \exists u \in \mathcal{U}' : \text{Post}(x, u) \subseteq (\mathcal{X} \cup \mathcal{X}_{\text{out}}) \setminus \bigcup_{j \in J} \mathcal{X}_j \right\} \\
&= \mathcal{X}' \setminus \text{PreR}\left(\mathcal{X}', \mathcal{U}', (\mathcal{X} \cup \mathcal{X}_{\text{out}}) \setminus \bigcup_{j \in J} \mathcal{X}_j \right).
\end{aligned}$$

B.6. Robust attractor

The robust attractor operator $\text{AttrR}(\mathcal{X}', \mathcal{U}', \{\mathcal{X}_j\}_{j \in J})$ from Table 1 can be computed using the predecessor operator, since it holds that

$$\begin{aligned}
\text{AttrR}(\mathcal{X}', \mathcal{U}', \{\mathcal{X}_j\}_{j \in J}) &= \left\{ x \in \mathcal{X}' \mid \forall u \in \mathcal{U}' : \text{Post}(x, u) \subseteq \bigcup_{j \in J} \mathcal{X}_j \right\} \\
&= \mathcal{X}' \setminus \left\{ x \in \mathcal{X}' \mid \exists u \in \mathcal{U}' : \text{Post}(x, u) \cap (\mathcal{X} \cup \mathcal{X}_{\text{out}}) \setminus \bigcup_{j \in J} \mathcal{X}_j \text{ is non-empty} \right\} \\
&= \mathcal{X}' \setminus \text{Pre}\left(\mathcal{X}', \mathcal{U}', (\mathcal{X} \cup \mathcal{X}_{\text{out}}) \setminus \bigcup_{j \in J} \mathcal{X}_j \right).
\end{aligned}$$

References

[1] E.M.M. Clarke, D. Peled, O. Grumberg, Model Checking, MIT Press, 1999.
[2] C. Baier, J.P. Katoen, Principles of Model Checking, The MIT Press, 2008.

[3] A. Church, Applications of recursive arithmetic to the problem of circuit synthesis, in: Summaries of the Summer Institute of Symbolic Logic, Vol. I, Cornell Univ., 1957, pp. 3–50.
[4] J.R. Büchi, L.H. Landweber, Solving sequential conditions by finite-state strategies, Trans. Amer. Math. Soc. 138 (1969) 367–378.
[5] A. Pnueli, R. Rosner, On the synthesis of a reactive module, in: Proc. of POPL, 1989, pp. 179–190.
[6] M. Svorenova, I. Cerna, C. Belta, Optimal receding horizon control for finite deterministic systems with temporal logic constraints, in: Proc. of IEEE ACC, 2013, pp. 4399–4404.
[7] M. Guo, K.H. Johansson, D.V. Dimarogonas, Revising motion planning under linear temporal logic specifications in partially known workspaces, in: Proc. of IEEE ICRA, 2013, pp. 5025–5032.
[8] B. Yordanov, J. Tumova, I. Cerna, J. Barnat, C. Belta, Temporal logic control of discrete-time piecewise affine systems, IEEE Trans. Automat. Control 57 (2012) 1491–1504.
[9] U. Topcu, N. Ozay, J. Liu, R.M. Murray, Synthesizing robust discrete controllers under modeling uncertainty, in: Proc. of ACM HSCC, 2012, pp. 85–94.
[10] M. Svorenova, I. Cerna, C. Belta, Optimal control of MDPs with temporal logic constraints, in: Proc. of IEEE CDC, 2013, pp. 3938–3943.
[11] M. Kattenbelt, M. Kwiatkowska, G. Norman, D. Parker, A game-based abstraction-refinement framework for Markov decision processes, Form. Methods Syst. Des. 36 (3) (2010) 246–280.
[12] P. Tabuada, G. Pappas, Model checking LTL over controllable linear systems is decidable, in: Proc. of HSCC, in: LNCS, vol. 2623, 2003, pp. 498–513.
[13] E.A. Gol, X. Ding, M. Lazar, C. Belta, Finite bisimulations for switched linear systems, in: Proc. of IEEE CDC, 2012, pp. 7632–7637.
[14] A. Girard, Synthesis using approximately bisimilar abstractions: state-feedback controllers for safety specifications, in: Proc. of HSCC, 2010, pp. 111–120.
[15] A.A. Julius, A. Girard, G.J. Pappas, Approximate bisimulation for a class of stochastic hybrid systems, in: Proc. of IEEE ACC, 2006, pp. 4724–4729.
[16] N. Piterman, A. Pnueli, Y. Sa'ar, Synthesis of reactive(1) designs, in: Proc. of VMCAI, in: LNCS, vol. 3855, 2006, pp. 364–380.
[17] Costas Courcoubetis, Mihalis Yannakakis, The complexity of probabilistic verification, J. ACM 42 (4) (1995) 857–907.
[18] K. Chatterjee, T.A. Henzinger, A survey of stochastic ω-regular games, J. Comput. System Sci. 78 (2) (2012) 394–413.
[19] M. Svorenova, J. Kretinsky, M. Chmelik, K. Chatterjee, I. Cerna, C. Belta, Temporal logic control for stochastic linear systems using abstraction refinement of probabilistic games, in: Proc. of HSCC, 2015, pp. 259–268.
[20] T.A. Henzinger, R. Jhala, R. Majumdar, Counterexample-guided control, in: ICALP, 2003.
[21] K. Chatterjee, M. Chmelik, P. Daca, CEGAR for qualitative analysis of probabilistic systems, in: CAV, 2014, pp. 473–490.
[22] K. Chatterjee, T.A. Henzinger, R. Jhala, R. Majumdar, Counterexample-guided planning, in: UAI, 2005.
[23] Marta Kwiatkowska, Gethin Norman, David Parker, Stochastic games for verification of probabilistic timed automata, in: Proc. of FORMATS, 2009, pp. 212–227.
[24] E.M. Hahn, G. Norman, D. Parker, B. Wachter, L. Zhang, Game-based abstraction and controller synthesis for probabilistic hybrid systems, in: Proc. of QEST, 2011, pp. 69–78.
[25] P. Nilsson, N. Ozay, Incremental synthesis of switching protocols via abstraction refinement, in: Proc. of IEEE CDC, 2014, pp. 6246–6253.
[26] E.A. Gol, M. Lazar, C. Belta, Language-guided controller synthesis for discrete-time linear systems, in: Proc. of HSCC, 2012, pp. 95–104.
[27] A. Abate, A. D'Innocenzo, M.D. Di Benedetto, Approximate abstractions of stochastic hybrid systems, IEEE Trans. Automat. Control 56 (11) (2011) 2688–2694.
[28] M. Lahijanian, S.B. Andersson, C. Belta, Formal verification and synthesis for discrete-time stochastic systems, IEEE Trans. Automat. Control 60 (8) (2015) 2031–2045.
[29] Antoine Girard, Colas Guernic, Oded Maler, Efficient computation of reachable sets of linear time-invariant systems with inputs, in: Proc. of HSCC, 2006, pp. 257–271.
[30] Colas Le Guernic, Antoine Girard, Reachability analysis of linear systems using support functions, Nonlinear Anal. Hybrid Syst. 4 (2) (2010) 250–262.
[31] John N. Maidens, Shahab Kaynama, Ian M. Mitchell, Meeko M.K. Oishi, Guy A. Dumont, Lagrangian methods for approximating the viability kernel in high-dimensional systems, Automatica 49 (7) (2013) 2017–2029.
[32] Leonhard Asselborn, Olaf Stursberg, Probabilistic control of uncertain linear systems using stochastic reachability, in: Proc. of IFAC ROCOND, Vol. 48, 2015, pp. 167–173.
[33] Mark Cannon, Qifeng Cheng, Basil Kouvaritakis, Saa V. Rakovi, Stochastic tube {MPC} with state estimation, Automatica 48 (3) (2012) 536–541.
[34] M. Althoff, O. Stursberg, M. Buss, Safety assessment for stochastic linear systems using enclosing hulls of probability density functions, in: Proc. of ECC, 2009, pp. 625–630.
[35] D. Dueri, B. Acikmese, M. Baldwin, R.S. Erwin, Finite-horizon controllability and reachability for deterministic and stochastic linear control systems with convex constraints, in: Proc. of ACC, 2014, pp. 5016–5023.
[36] K. Chatterjee, Stochastic ω-regular games (Ph.D. thesis), UC Berkeley, 2007.
[37] K. Chatterjee, L. de Alfaro, T.A. Henzinger, Qualitative concurrent parity games, ACM Trans. Comput. Log. 12 (4) (2011).
[38] Krishnendu Chatterjee, Thomas A. Henzinger, Value iteration, in: 25 Years of Model Checking—History, Achievements, Perspectives, 2008, pp. 107–138.