Automated Verification and Control of Infinite State Stochastic Systems

by

Đorđe Žikelić

November, 2023

A thesis submitted to the
Graduate School
of the
Institute of Science and Technology Austria
in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy

Committee in charge:

Caroline J. Muller, Chair
Krishnendu Chatterjee
Petr Novotný
Thomas A. Henzinger
Andreas Podelski

The thesis of Đorđe Žikelić, titled *Automated Verification and Control of Infinite State Stochastic Systems*, is approved by:

Supervisor: Krishnendu Chatterjee, ISTA, Klosterneuburg, Austria

Signature: _____

Co-supervisor: Petr Novotný, Masaryk University, Brno, Czech Republic

Signature: _____

Committee Member: Thomas A. Henzinger, ISTA, Klosterneuburg, Austria

Signature: _____

Committee Member: Andreas Podelski, University of Freiburg, Freiburg, Germany

Signature: _____

Defense Chair: Caroline J. Muller, ISTA, Klosterneuburg, Austria

Signature: _____

Signed page is on file

ISTA Thesis, ISSN: 2663-337X

ISBN: 978-3-99078-036-7

I hereby declare that this thesis is my own work and that it does not contain other people's work without this being so stated; this thesis does not contain my previous work without this being stated, and the bibliography contains all the literature that I used in writing the dissertation.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee, and that this thesis has not been submitted for a higher degree to any other university or institution.

I certify that any republication of materials presented in this thesis has been approved by the relevant publishers and co-authors.

Signature: _____

Đorđe Žikelić
November, 2023

Signed page is on file

Abstract

Stochastic systems provide a formal framework for modelling and quantifying uncertainty in systems and have been widely adopted in many application domains. Formal verification and control of finite state stochastic systems, a subfield of formal methods also known as probabilistic model checking, is well studied. In contrast, formal verification and control of infinite state stochastic systems have received comparatively less attention. However, infinite state stochastic systems commonly arise in practice. For instance, probabilistic models that contain continuous probability distributions such as normal or uniform, or stochastic dynamical systems which are a classical model for control under uncertainty, both give rise to infinite state systems.

The goal of this thesis is to contribute to laying theoretical and algorithmic foundations of fully automated formal verification and control of infinite state stochastic systems, with a particular focus on systems that may be executed over a long or infinite time. We consider formal verification of infinite state stochastic systems in the setting of static analysis of probabilistic programs and formal control in the setting of controller synthesis in stochastic dynamical systems. For both problems, we present some of the first fully automated methods for probabilistic (a.k.a. quantitative) reachability and safety analysis applicable to infinite time horizon systems. We also advance the state of the art of probability 1 (a.k.a. qualitative) reachability analysis for both problems. Finally, for formal controller synthesis in stochastic dynamical systems, we present a novel framework for learning neural network control policies in stochastic dynamical systems with formal guarantees on correctness with respect to quantitative reachability, safety or reach-avoid specifications.

Acknowledgements

I would like to begin by thanking my advisor Krishnendu Chatterjee. He provided me with an opportunity to develop my research interests, gave me a great amount of freedom to pursue my ideas and believed in my ability to execute them, while constantly being available to provide invaluable research guidance. All of that in a very enjoyable working environment. I am also deeply grateful for all the career advice, job interview preparation tips and help with other aspects of professional life. The same amount of gratitude goes to Petr Novotný, who was equally supportive and available to provide guidance throughout my doctoral studies. Even though we were based in different places, he ensured that it did not present much of an obstacle and we regularly visited each other in Brno and in Vienna. It was a privilege to be advised by Krish and Petr.

I owe a similar level of gratitude to Thomas A. Henzinger, who had a significant role in my PhD studies. Tom was extremely supportive of my work and was always available to provide research guidance and career advice. His support had instrumental role in my decision to pursue an academic career following the doctoral studies. Furthermore, I would like to thank Andreas Podelski for being a part of my thesis committee, Caroline J. Muller for chairing my thesis defense and Edouard Hannezo for chairing my qualifying exam.

During my doctoral studies, I was fortunate to meet and collaborate with many talented researchers. Two people that I worked with the most and that I would particularly like to thank are Mathias Lechner and Guy Avni. Mathias and I have spent countless hours brainstorming, writing papers, chasing deadlines and have become great friends in the process. Of all the collaborations, my interaction with Mathias has had the most significant impact on my development as a researcher. My collaboration with Guy started while I was working on a rotation project in Tom's group. Ever since, I have learned a lot from Guy, both about game theory as well as non-scientific aspects of academic life. I am grateful to Mathias and Guy and hope that we continue working together in the future.

I also thank all my other collaborators, working with whom has been an absolute pleasure: S Akshay, Ali Asadi, Amir Kafshdar Goharshady, Ehsan Kafshdar Goharshady,

Ismaël Jecker, Mehrdad Karrabi, Tobias Meggendorfer, Andreas Pavlogiannis, Jakub Svoboda, Josef Tkadlec and Abhinav Verma. During my PhD, I had the opportunity to supervise three very talented interns: Matin Ansaripour, Amirali Ebrahim-zadeh and Roodabeh Safavi. I am also grateful to all other members of Krish's and Tom's groups with whom I have shared a very enjoyable working environment over the past five years. Finally, I would like to thank Ksenja Harpprecht and Elisabeth Hacker for all the administrative support and especially for ensuring that my research travel organization was always a smooth experience.

During my PhD studies, I completed two applied scientist internships with the Prime Video Automated Reasoning team at Amazon in London, UK. Both internships were amazing experiences from which I have learned a lot. I would like to thank Pauline Bolignano, Bor-Yuh Evan Chang and Franco Raimondi for mentoring and managing me (or both) and all the members of the Prime Video Automated Reasoning team for a truly amazing and welcoming environment.

I also thank Viktor Kunčak and Anna Lukina for inviting me to visit their groups at EPFL and TU Delft, respectively.

During my time at ISTA, I have made some good friends without whom these five years in Austria would not have been such an enjoyable experience. Thank you Guille, Alex, Catalin, Linda, Sarath, Sebastiano, Marko, David, Peđa, Dario, Mariano, Elizabeth, Martin, Bryan and Suyash. As my friends and colleagues in Vienna may have noticed, and I doubt they haven't, I maintain it a priority to regularly and frequently visit my home town Belgrade. And while there are many dear people back at home that I am highly fond of, special thanks go to Ivan, Miloš, Miloš, Marko, Nenad, Luka, Danilo, Vlada, Miloš, Isidora and Đorđe. They have all known me for many years and neither time nor physical distance has affected our friendships. Thank you everyone.

In the hope that my friends in Vienna will not be offended by this statement, the most important person that I have met here is Michelle. I moved to Vienna with the intention to do a great PhD, but I've also found something more important. She has been a constant source of support and kindness. It is only appropriate that we also graduate together.

Finally, I would like to thank my parents Snežana and Nenad for their unconditional love, support and care. This thesis is dedicated to them.

Funding sources. This work was supported in part by the ERC CoG 863818 (FoRM-SMArt) and the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie Grant Agreement No. 665385.

About the Author

Đorđe Žikelić obtained bachelor's and master's degrees in mathematics at the University of Cambridge before joining ISTA in September 2018. His research focuses on developing algorithms for formally verifying correctness of software and combines ideas from formal methods, programming languages and machine learning research as well as from probability theory. His main research interests include probabilistic verification, static program analysis, AI safety, learning-based control and games on graphs. His work has been published at premier venues in formal methods (CAV, FM), programming languages (PLDI, POPL), artificial intelligence (AAAI), machine learning (NeurIPS) and algorithms (SODA) communities. He was a finalist for the Meta PhD Research Fellowship in the programming languages category in 2022.

List of Collaborators and Publications

1. Chapter 3 is based on "Krishnendu Chatterjee, Ehsan Kafshdar Goharshady, Petr Novotný, Jiří Zárevúcky, Đorđe Žikelić.[†] *On Lexicographic Proof Rules for Probabilistic Termination*. In Formal Methods - 24th International Symposium, **FM 2021**"

2. Chapter 3 is based on "Krishnendu Chatterjee, Ehsan Kafshdar Goharshady, Petr Novotný, Jiří Zárevúcky, Đorđe Žikelić.[†] *On Lexicographic Proof Rules for Probabilistic Termination*. In Formal Aspects of Computing, **FAC 2023**"

3. Chapter 4 is based on "Krishnendu Chatterjee, Petr Novotný, Đorđe Žikelić.[†] *Stochastic Invariants for Probabilistic Termination*. In 44th ACM SIGPLAN Symposium on Principles of Programming Languages, **POPL 2017**"[‡]

4. Chapter 4 is based on "Krishnendu Chatterjee, Amir Kafshdar Goharshady, Tobias Meggendorfer, Đorđe Žikelić.[†] *Sound and Complete Certificates for Quantitative Termination Analysis of Probabilistic Programs*. In Computer Aided Verification - 34th International Conference, **CAV 2022**"

5. Chapter 5 is based on "Krishnendu Chatterjee, Ehsan Kafshdar Goharshady, Petr Novotný, Đorđe Žikelić.[†] *Proving Non-termination by Program Reversal*. In 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation, **PLDI 2021**"

6. Chapter 6 is based on "Mathias Lechner*, Đorđe Žikelić* Krishnendu Chatterjee, Thomas A. Henzinger. *Stability Verification in Stochastic Control Systems via Neural Network Supermartingales*. In Thirty-Sixth AAAI Conference on Artificial Intelligence, **AAAI 2022**"

[†]Authors ordered alphabetically.

[‡]This work was completed prior to the beginning of the author's PhD studies, however it is a result of an internship project at ISTA that preceeded PhD studies. Chapter 4 contains results of a part of this publication that introduces stochastic invariants and proves their soundness for quantitative termination analysis. The rest of Chapter 4 is based on the publication in item 4.

*Equal contribution.

7. Chapter 6 is based on "Đorđe Žikelić*, Mathias Lechner*, Krishnendu Chatterjee, Thomas A. Henzinger. *Learning Stabilizing Policies in Stochastic Control Systems*. In ICLR 2022 Workshop on Socially Responsible Machine Learning, **SRML 2022**"

8. Chapter 7 is based on "Đorđe Žikelić*, Mathias Lechner*, Thomas A. Henzinger, Krishnendu Chatterjee. *Learning Control Policies for Stochastic Systems with Reach-avoid Guaranteess*. In Thirty-Seventh AAAI Conference on Artificial Intelligence, **AAAI 2023**"

The following list contains other works published during the PhD period. While they do not constitute the main body of this thesis, the contributions of some of these works will be discussed in Chapter 8.

9. S. Akshay, Krishnendu Chatterjee, Tobias Meggendorfer, Đorđe Žikelić†. *MDPs as Distribution Transformers: Affine Invariant Synthesis for Safety Objectives*. In Computer Aided Verification - 35th International Conference, **CAV 2023**

10. Krishnendu Chatterjee, Thomas A. Henzinger, Mathias Lechner, Đorđe Žikelić†. *A Learner-verifier Framework for Neural Network Controllers and Certificates of Stochastic Systems*. In Tools and Algorithms for the Construction and Analysis of Systems, **TACAS 2023** (invited paper)

11. Guy Avni, Ismaël Jecker, Đorđe Žikelić†. *Bidding Graph Games with Partially-observable Budgets*. In Thirty-Seventh AAAI Conference on Artificial Intelligence, **AAAI 2023**

12. Mathias Lechner, Đorđe Žikelić, Krishnendu Chatterjee, Thomas A. Henzinger, Daniela Rus. *Quantization-aware Interval Bound Propagation for Training Certifiably Robust Quantized Neural Networks*. In Thirty-Seventh AAAI Conference on Artificial Intelligence, **AAAI 2023**

13. Ali Ahmadi, Krishnendu Chatterjee, Amir Kafshdar Goharshady, Tobias Meggendorfer, Roodabeh Safavi, Đorđe Žikelić†. *Algorithms and Hardness Results for Computing Cores of Markov Chains*. In 42nd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, **FSTTCS 2022**

14. Krishnendu Chatterjee, Jakub Svoboda, Đorđe Žikelić, Andreas Pavlogiannis, Josef Tkadlec. *Social Balance on Networks: Local Minima and Best-edge Dynamics*. In Physical Review E, **PRE 2022**

15. Đorđe Žikelić, Bor-Yuh Evan Chang, Pauline Bolignano, Franco Raimondi. *Differential Cost Analysis with Simultaneous Potentials and Anti-potentials*. In 43rd ACM SIGPLAN International Conference on Programming Language Design and Implementation, **PLDI 2022**

16. Mathias Lechner*, Đorđe Žikelić,* Krishnendu Chatterjee, Thomas A. Henzinger. *Infinite Time Horizon Safety of Bayesian Neural Networks*. In Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems, **NeurIPS 2021**

17. Guy Avni, Thomas A. Henzinger, Đorđe Žikelić†. *Bidding Mechanisms in Graph Games*. In Journal of Computer and System Sciences, **JCSS 2021**

18. Thomas A. Henzinger, Mathias Lechner, Đorđe Žikelić†. *Scalable Verification of Quantized Neural Networks*. In Thirty-Fifth AAAI Conference on Artificial Intelligence, **AAAI 2021**

19. Guy Avni, Ismaël Jecker, Đorđe Žikelić†. *Infinite-duration All-pay Bidding Games*. In 2021 ACM-SIAM Symposium on Discrete Algorithms, **SODA 2021**

20. Guy Avni, Thomas A. Henzinger, Đorđe Žikelić†. *Bidding Mechanisms in Graph Games*. In 44th International Symposium on Mathematical Foundations of Computer Science, **MFCS 2019**

Table of Contents

List of Figures

List of Tables

List of Algorithms

CHAPTER 1

Introduction

"So Einstein was wrong when he said, "God does not play dice." Consideration of black holes suggests, not only that God does play dice, but that he sometimes confuses us by throwing them where they can't be seen."
Stephen Hawking

1.1 Prologue

Modern software systems are complex – they may contain many lines of code, be implemented by multiple developers and be modified over time. This makes it extremely hard to detect bugs and it has become practically infeasible to write test suites that ensure total coverage of all system behaviors and all corner cases. Moreover, while testing is able to detect bugs, in general it is not able to guarantee their absence. As such, testing alone is not sufficient to provide the desired level of trustworthiness and software reliability that is necessary for the deployment in safety-critical scenarios. Furthermore, recent years have seen tremendous success of artificial intelligence (AI) and machine learning (ML) technologies which have naturally fueled the desire to deploy them in general software development. However, the lack of explainability of AI and ML technologies remains the key challenge for their deployment in safety-critical domains.

Formal verification and program analysis. Formal verification is concerned with formally reasoning about programs and software systems, towards analysing their properties, detecting bugs or proving correctness with respect to well-defined specifications [Flo67, Hoa69, Dij76]. Given a program and a formal specification, the goal of formal verification is to design mathematically rigorous yet fully automated and scalable methods for proving or disproving that the program satisfies the given

specification. If a program is formally verified to be correct, then a programmer need not worry about untested corner cases that may lead to incorrect behavior because their absence is guaranteed. Such rigorous guarantees enhance reliability and trustworthiness of programs and software systems. It is therefore not surprising that formal verification and program analysis have for long time been active research fields [JM09], and formal methods and tools have found wide industrial adoption. For instance, Amazon [NRZ+15], Meta [CDD+15] and Microsoft [BCLR04] are all developing and using formal verification and program analysis tools.

Formal controller synthesis. While formal verification is concerned with formally reasoning about a given system, the goal of formal synthesis is to automatically construct a system that satisfies a given specification [PR89]. The key advantage of formal synthesis is that it yields correct-by-construction systems that are reliable and trustworthy by design. In addition to formal verification and program analysis, in this thesis we will also be interested in formal controller synthesis, which is concerned with synthesizing controllers for dynamical systems. A controller is a reactive system which continuously interacts with the dynamical system, receives inputs and produces control outputs [BS04, Ber12]. Recent years have seen significant demand for formal controller synthesis methods for safety-critical autonomous systems such as self-driving cars, aircraft systems and medical devices.

Stochastic systems. Stochastic (or probabilistic) systems are systems that exhibit uncertain behaviour and that use probability to formally model and quantify uncertainty. They have been adopted in many application domains as adequate formal models for uncertain systems whose exact semantics are unknown or too complex to analyse, but patterns in system behaviour or data can be observed. Moreover, stochastic uncertainty is not always a consequence of our lack of understanding of exact semantics and can also be a design choice. For instance, randomised algorithms use coin tossing or random value sampling to solve certain tasks more efficiently. Below we list some applications in which stochastic systems naturally arise:

1. *Randomised algorithms.* As mentioned above, randomised algorithms [MR95] use probability to yield more efficient solutions to certain algorithmic problems. For instance, selecting a pivot for the Hoare's quicksort algorithm [Hoa62] uniformly at random yields a randomised sorting algorithm which has expected runtime complexity $\mathcal{O}(n \log n)$ to sort an array of n elements. In contrast, deterministic pivot selection yields an algorithm whose worst-case runtime complexity is $\mathcal{O}(n^2)$.
2. *Distributed systems.* Distributed systems consist of networked components which exchange messages and coordinate their executions towards achieving a common goal. Uncertainty in distributed systems may arise at many levels, for instance

in the form of message loss or unknown distribution of packet traffic at routers in the Internet. In the analysis of distributed systems, stochastic networks are standard formal models [KY14]. Furthermore, randomisation is often a design choice in distributed system protocols, e.g. for leader election [GvRB00].

3. *Motion planning and control.* Planning and control tasks are solved with respect to a specified model of the environment. The model is typically inferred from observed data or noisy measurements and may need to take into account disturbances that contribute to the evolution of the system. Stochastic control [BS96] is a subfield of control theory that considers controller synthesis in stochastic environment models, where stochasticity is used to formally model data uncertainty, measurement noise or environment disturbances.

4. *Machine learning.* Machine learning can be viewed as a probabilistic inference problem, where the goal is to infer a model that best matches observed data [Gha15, Mur12]. In supervised learning, one can use probability to quantify the confidence in model's prediction. In reinforcement learning (RL), the goal is to learn a policy for sequential decision making that maximises expected reward in a Markov decision process (MDP), whose semantics are stochastic [SB18].

Formal analysis of stochastic systems. Classical formal verification and synthesis techniques are boolean in nature and aim for absolute correctness guarantees – they want to ensure that every system execution satisfies the formal specification. One could of course apply such boolean reasoning to stochastic systems by analysing every execution individually and trying to provide worst-case guarantees on correctness. However, such reasoning would be highly conservative and would lead to refutation of many systems that would in practice be regarded as correct. For instance, if in analysing software systems we take into account failures in unreliable hardware, then no software system could be verified to be absolutely correct as the worst-case scenario entails hardware failure. The key issue with such a boolean analysis is that it does not take into account that hardware failure happens with very low probability. Hence, stochastic systems demand more fine-grained analyses that reason about the *probability* with which some formal specification is satisfied. Formal analyses for stochastic systems can be classified into *qualitative* and *quantitative* [BK08]:

- *Qualitative analysis.* Qualitative analysis is concerned with proving that the set of stochastic system executions that satisfy the formal specification has probability 0 or 1. In other words, up to some set of executions of probability 0, either none or all executions of the stochastic system need to satisfy the formal specification.

- *Quantitative analysis.* Quantitative analysis takes as input a probability threshold $p \in [0, 1]$ and is concerned with proving that the set of stochastic system

executions that satisfy the formal specification has probability at least p. This is a more fine-grained analysis that provides useful information even if a stochastic system does not satisfy the formal specification with probability 0 or 1.

In this thesis, we will study qualitative and quantitative analyses for *reachability* and *safety* specifications. Given a stochastic system and a set of states S, we say that a system execution satisfies the reachability specification for S if it contains a state in S. Dually, we say that a system execution satisfies the safety specification for S if it does not contain any state in S and therefore avoids it. Reachability and safety are the most fundamental liveness and safety specifications, respectively [BK08].

Finite and infinite state stochastic system analysis. Formal analysis of stochastic systems has received significant attention within the formal methods community. However, prior work has predominantly focused on formal verification and controller synthesis for *finite state* stochastic systems, an area that we call *probabilistic model checking*. There has been a significant amount of work on probabilistic model checking and today we known how to efficiently and fully automatically reason about a large class of properties in finite state MDPs. We refer the reader to [Kat16] for an overview of developments in probabilistic model checking and to the state of the art probabilistic model checking tools such as PRISM [KNP11] and Storm [HJK+22] which implement many of these analyses. In comparison, automated formal analysis of *infinite state* stochastic systems is much less explored and presents one of the major challenges in formal methods. Infinite state stochastic systems commonly arise in practice – continuous probability distributions such as normal or uniform are standard in probabilistic modelling and stochastic planning and control problems over continuous state spaces give rise to infinite state stochastic systems. However, the key challenge for their automated analysis is that the formal verification problem for any non-trivial property in infinite state stochastic systems specified via a Turing-complete language is undecidable, by Rice's theorem [Ric53]. In contrast, many problems in probabilistic model checking can be automatically and efficiently decided. Thus, one cannot directly reduce the formal verification and controller synthesis problems for infinite state stochastic systems to probabilistic model checking. Instead, these problems require new approaches.

1.2 Infinite State Stochastic System Analysis

This thesis contributes to laying theoretical and algorithmic foundations of fully automated formal verification and controller synthesis for discrete-time* infinite state stochastic systems, with respect to reachability and safety specifications. Analogously

*We emphasise that this thesis only considers *discrete-time* stochastic systems and in what follows we assume that all stochastic systems evolve over discrete time. Formal analysis of continuous-time

to probabilistic model checking, (measurable) Markov chains and MDPs are classical objects of study in *theoretical* analysis of infinite state stochastic systems. However, they may in general be defined in terms of transition functions that are not computable. Since we are interested in both *theory and automation*, this thesis focuses on formal analysis of two classes of infinite state stochastic systems that are most commonly used in probabilistic modelling and control theory applications:

1. *Probabilistic programs.* We consider probabilistic programs as a model for formal verification of infinite state stochastic systems. A probabilistic program (PP) is a classical imperative or functional program extended with (1) the ability to sample values from probability distributions and assign it to program variables, and (2) the ability to condition program executions on observed variable values [GHNR14]. Probability distributions appearing in sampling instructions may be both discrete (such as Bernoulli or Poisson) or continuous (such as normal or uniform), hence PPs give rise to infinite state stochastic systems. They provide an expressive framework for formally specifying a rich class probabilistic models, including probabilistic graphical models [KF09] and all discrete probability distributions for which events are semi-computable [Ica17, Kam19]. PPs have been used in a multitude of applications including randomized algorithms [MR95], stochastic networks [BK08, KNP11, FKM$^+$16], cryptography [BGB09], security [BGG$^+$16, BGHP16] and machine learning [Gha15, vdMPYW18]. Today, there is a large variety of probabilistic programming languages such as Anglican [TvdMYW16], Church [GMR$^+$08], Edward [TKD$^+$16], Pyro [BCJ$^+$19] or WebPPL [GS14].

   The expressiveness and wide adoption of PPs makes them a particularly appealing model for formal analysis, and program analysis of PPs has become a very active area in formal methods and programming languages research. In particular, instead of designing different verification algorithms for each application domain in which stochastic systems may arise, one can first translate a stochastic system of interest into a PP and then focus on the analysis of the PP. Many works on static analysis of PPs, inclusive of all results on PP analysis in this thesis, extend PPs with programming constructs for non-deterministic choices. Similarly to the static analysis of non-probabilistic programs, this extension allows modelling unknown program inputs but also over-approximation of PP parts that are too complex for static analysis [CC77, MM05].

2. *Discrete-time stochastic dynamical systems.* We consider discrete-time stochastic dynamical systems as a model for formal controller synthesis in infinite state

stochastic systems has also been considered in the literature in the setting of continuous-time Markov chains [BHHK03] or continuous-time stochastic dynamical systems [KKDD01], but these models are not the topic of this thesis.

stochastic systems. Discrete-time stochastic dynamical systems [BS96, Ber12] present a standard model for solving control problems in continuous state spaces and over discrete time, in the presence of stochastic disturbances or when the model of the environment is inferred from observed data or noisy measurements. The goal of stochastic control is to synthesise a controller under which the stochastic dynamical systems satisfies some formal specification of interest.

1.3 Prior Work, Challenges and Thesis Goal

In what follows, we overview prior work on reachability and safety analysis of infinite state stochastic systems and identify challenges that this thesis aims to address. We note that this is not a complete list of related work. Rather, it only serves towards placing contributions of this thesis within the broader perspective. We will later provide a detailed overview of related work for each chapter in this thesis.

Prior work on probabilistic program analysis. Formal analysis of PPs goes back to the seminal work of Kozen on PP semantics [Koz81]. Since then, there has been much work on developing probabilistic extensions of Hoare logic and Dijkstra's weakest precondition calculus. Probabilistic propositional dynamic logic (PPDL), a modal logic for reasoning about PPs with discrete probability distributions, was developed in [Koz85]. PPDL was further studied and extended to PPs with non-determinism, giving rise to weakest pre-expectation calculus [MMS96, MM05]. Weakest pre-expectation calculus can be viewed as a generalization of Dijkstra's weakest precondition calculus to the setting of PPs with non-determinism. It has been extensively studied, with many developments on reasoning about reachability, safety and other properties [KKMO18, OKKM16, OGJ+18, KK17, ABH+21, BKKM21] and it provides a powerful theoretical framework for PP analysis. However, automation of these calculi typically requires user input in the presence of unbounded loops or continuous probability distributions.

Recent years have seen the development of several methods for fully automated PP analysis. For PPs with bounded loops, exact inference methods perform weighted model counting [HdBM20] or symbolic execution and integration [GMV16, GSV20] in order to compute exact expected values of functions or probabilities of events upon PP execution. Approximate inference methods include [CMMV16, CDM17, HDM22]. The works [SCG13, BO21, BOZ22] consider PPs with linear arithmetic and use symbolic execution and integration to compute bounds on reachability and safety probability. These are not restricted to bounded loops, however they only reason about terminating program executions and do not consider non-determinism. Abstract interpretation for PPs was studied in [Mon00, Mon01], but it remains unclear how to perform the widening operation in the presence of loops and automation is only briefly discussed. Automated

methods for the analysis of prob-solvable loops, i.e. loops whose body contains a sequence of probabilistic assignments but no conditional branching or nested loops, that are based on recurrence solving have been proposed in [BKS19, MBKK21a, MBKK21b].

Finally, *martingale-based methods* are conceptually most related to the work in this thesis. Their name is inspired by their usage of martingale processes from probability theory [Wil91]. Prior work on martingale-based methods for PPs has focused on proving probability 1 (a.k.a. almost sure) termination/reachability or proving bounds on termination/reachability time. The central concept in these approaches is a ranking supermartingale (RSM), which generalizes ranking functions for termination analysis in non-probabilistic programs. We say that a PP execution is terminating, if it reaches a terminal state and does not execute forever. By letting the PP terminate upon reaching some target set of states, one can always reduce the reachability specification to the termination specification, thus we treat these two specifications as equivalent. RSMs were introduced in [CS13], extended to PPs with non-determinism in [CFNH18] and were extensively studied since [CFG16, ACN18, MMKK18, HFCG19, AGR21, CH20, KO21]. However, martingales have not been used for quantitative reachability or safety analysis in PPs. To the best of our knowledge, no method prior to this thesis provides quantitative reachability and safety guarantees in a fully automated fashion for PPs that are *not almost-surely terminating*, and that can hence model *infinite time-horizon* systems.

Prior work on stochastic control. Automated controller synthesis for *deterministic* dynamical systems with formal reachability and safety guarantees has been extensively studied. Formal guarantees are typically achieved by synthesizing a control policy together with a certificate function which formally proves the desired property. Classical automated methods consider *polynomial systems* and utilize semidefinite programming (SDP) to synthesize polynomial policies and certificate functions [HG05, Par00, JWFT$^+$03]. However, dynamical systems appearing in practice are often not polynomial. To alleviate this problem, classical methods consider polynomial approximations of system dynamics, but this introduces approximation error that accumulates over time and allows only finite and typically short time horizon guarantees. A promising approach to enable synthesis of controllers and certificate functions for non-polynomial systems while not imposing time horizon restrictions is a *learning-based approach*. These methods leverage advances in deep reinforcement learning by learning and formally verifying a control policy together with a safety certificate, both parametrized as neural networks [CRG19, AAGP21, PAA21]. See [DGF23] for a survey. Another approach to formal controller synthesis is provided by abstraction-based methods [Tab09]. Rather than utilizing certificate functions, these methods approximate the controller synthesis problem using a simpler problem of computing winning strategies in abstract finite-state systems, for which known algorithms can be used.

In comparison, automated controller synthesis for *stochastic* dynamical systems with

formal reachability and safety guarantees is a harder problem and remains a challenging research area. An SDP method for jointly synthesizing a polynomial policy and a stochastic barrier function (SBF) in polynomial systems with quantitative safety guarantees was proposed in [PJP04, PJP07]. SBF is a martingale-based certificate that can be used to prove probabilistic safety over the infinite time horizon. Methods that first compute polynomial approximations and then use SDP to provide finite time horizon safety guarantees were proposed in [ST12, SDC21]. Concurrently to this thesis, an SDP method with reach-avoid guarantees for infinite-time horizon systems has been developed in [XLZF21], but this method is restricted to polynomial systems. Other methods use dynamic programming [APLS08] or Hamilton-Jacobi (HJ) reachability analysis [BCHT17] for finite time horizon stochastic systems.

Another approach to formal controller synthesis in stochastic dynamical systems is provided by abstraction-based methods [ADB11]. Similarly to their deterministic analogues, these methods approximate the controller synthesis problem using abstract finite-state MDPs or stochastic games for which known probabilistic model checking algorithms can be used, however the abstraction requires additional care and novel mathematical developments [TMKA17]. There has been considerable amount of work on abstraction-based controller synthesis in stochastic dynamical systems, however most existing methods are applicable to finite time horizon systems due to accumulation of the approximation error [SGA15, LKSZ20, CA19, VGO19]. Recently, a few abstraction-based controller synthesis methods for infinite-time horizon stochastic dynamical systems with affine dynamics [HS21], control affine dynamics [DHC22] as well as for infinite-time horizon *switched* stochastic systems in which control input space is finite [MMSS24, DHC22] have been proposed. To the best of our knowledge, no prior method provides formal reachability and safety guarantees for *infinite-time horizon and non-polynomial* stochastic dynamical systems with *continuous* control input spaces.

Challenges and thesis goal. While these works present significant advances, several challenges in formal reachability and safety analysis of infinite state stochastic systems still remain. In particular, most existing methods are either not fully automated or are restricted to finite-time horizon systems or terminating executions. Exceptions include automated methods for probability 1 reachability proving in PPs and for controller synthesis in polynomial, control affine or switched stochastic systems. However, we currently lack automated methods for quantitative reachability and safety analysis in infinite-time horizon systems whose dynamics are continuous but non-polynomial. The goal of this thesis is to address these challenges and to advance and enable

fully automated reachability and safety analysis
in infinite-time horizon stochastic systems.

More concretely, our goal is to further advance existing analyses such as probability 1 reachability as well as to contribute to laying theoretical and algorithmic foundations of quantitative reachability and safety analyses that have not been possible before. Furthermore, we aim for fully automated methods that are applicable to PPs that need not be almost-surely terminating and to non-polynomial stochastic dynamical systems with continuous dynamics and control inputs. To achieve the latter, following recent developments in deterministic dynamical system control that leverage advances in deep reinforcement learning, our goal is to present a neurosymbolic control framework for learning and formally verifying neural controllers in stochastic dynamical systems.

Martingale-based approach. To solve these challenges, we follow a *martingale-based approach*. As surveyed above, prior work has showed that martingale based certificates can be used to formally certify properties in stochastic systems. In particular, ranking supermartingales (RSMs) have been used to formally certify probability 1 termination and more generally reachability in PPs, and stochastic barrier functions (SBFs) have been used to formally certify quantitative safety in stochastic dynamical systems. This hints that martingales provide a tool to formally reason about both reachability and safety. However, the usage of martingales for formally reasoning about quantitative reachability or conjuction of quantitative reachability and safety has not been explored.

The *main theoretical contribution* of this thesis is the design of novel martingale-based formal certificates for reasoning about quantitative reachability, safety and reach-avoidance in infinite state stochastic systems. Our results are theoretical in nature and thus allow their instantiation both in the setting of PPs as well as stochastic dynamical systems. On the other hand, the *main algorithmic contribution* of this thesis is full automation of the computation of our novel martingale-based certificates, both in PPs and in stochastic dynamical systems. While the theoretical analyses of these two models are similar, the automation algorithms are fundamentally different. In PPs, we show that one can automate the computation of our novel martingale-based certificates by building upon the existing methods for RSM synthesis in PPs. In contrast, in stochastic dynamical systems, we develop a completely new neurosymbolic framework for learning and formally verifying neural control policies together with neural martingale certificates.

1.4 Thesis Outline and Contributions

Chapter 2 provides mathematical preliminaries and formally defines the models of PPs and stochastic dynamical systems that we consider. The rest of this thesis can be divided into two parts. First, Chapters 3, 4 and 5 study program analysis of PPs. Then, Chapters 6 and 7 study formal controller synthesis for stochastic dynamical systems. In both parts, each chapter considers formal analysis with respect to a different property,

hence we try to keep each chapter as self contained as possible so that each chapter can be read independently. The only exception is Chapter 7 in which the learning-based framework builds on Chapter 6, so Chapter 6 should be read first. Contributions in each chapter of this thesis can be summarized as follows:

- Chapter 3 considers the *probability* 1 *(a.k.a. almost-sure) termination/reachability* problem in PPs. While ranking supermartingales (RSMs) present a prominent approach to proving almost-sure termination, they are not compositional and are hard to synthesize in PPs with complex control-flow structure. A lexicographic ranking supermartingale (LexRSM) [ACN18] presents a multi-dimensional extension of RSMs, that generalizes lexicographic ranking functions for termination proving in non-probabilistic programs. However, LexRSMs have a limitation that impedes their automation – all of their components have to be non-negative in all reachable states. As we show, this might result in LexRSM not existing even for simple terminating PPs. Our contributions are as follows:
  - *Theory.* We introduce a *generalized lexicographic ranking supermartingale (GLexRSM)* which allows some of its components to be negative. This standard feature of lexicographic ranking functions was hitherto not known to be sound in the probabilistic setting, and the soundness proof requires highly non-trivial analysis.
  - *Automation.* We present a polynomial time algorithm for automated synthesis of linear GLexRSMs in linear arithmetic PPs. The algorithm is applicable to linear arithmetic PPs that may contain nested loops, conditional branching, non-determinism and sampling instructions from discrete and continuous probability distributions. Our results also yield the first automated method that operates over linear arithmetic and can prove almost-sure termination in PPs that contain sampling instructions from double-sided unbounded support probability distributions, e.g. normal distributions. We demonstrate the applicability of our synthesis procedure on PPs for which no existing linear arithmetic method could prove almost-sure termination.
- Chapter 4 considers the *quantitative termination/reachability and safety* problems in PPs. To the best of our knowledge, it presents the first fully automated method for quantitative reachability and safety analysis in PPs that may not be almost-surely terminating. Our contributions are as follows:
  - *Theory.* We introduce the notion of a *stochastic invariant (SI)*, which generalizes the classical notion of program invariants to the setting of PPs. We show that SIs can be used to formulate sound and complete formal certificates for quantitative termination/reachability and for quantitative

safety analysis in PPs. The completeness proof is based on deep results from probability and martingale theory.

- *Automation.* We present a PSPACE algorithm for automated synthesis of our SI-based formal certificates in polynomial arithmetic PPs. The algorithm is sound and relatively complete, the latter meaning that it is guaranteed to compute a formal certificate whenever a polynomial instance of the formal certificate instance exists. The algorithm is applicable to polynomial arithmetic PPs that need not be almost-surely terminating and that may contain nested loops, conditional branching, non-determinism and sampling instructions from discrete and continuous probability distributions. We experimentally evaluate a prototype implementation of the algorithm and demonstrate its effectiveness on a number of classical PP benchmarks.

- Chapter 5 considers the *non-termination* problem in *both non-probabilistic programs and in PPs*. Given a non-probabilistic program (resp. PP) with non-determinism, the goal of the non-termination problem is to find *one scheduler* under which the program is not terminating (resp. almost-surely terminating). Note that this is fundamentally different from the safety problem, where the goal is to prove that the program is not terminating (resp. almost-surely terminating) under *every scheduler*. While termination proving is well-studied, the problem of detecting non-termination bugs has received much less attention. To that end, in this chapter we first study the problem of proving non-termination in *non-probabilistic programs*, and then show how these results can be extended to PPs as well. Our contributions are as follows:

  - *Theory.* We present a new formal certificate for non-termination proving in programs with non-determinism. Our novel certificate is rather simple but efficient. It relies on a purely syntactic reversal of the program's transition system and combines forward and backward reasoning in order to certify non-termination. To the best of our knowledge, no prior method combines forward and backward analysis to proving non-termination. The method can be easily extended to disproving almost-sure termination in PPs in which all probability distributions have countable support.

  - *Automation.* We present an algorithm for automated synthesis of our non-termination formal certificate. Our algorithm provides the first method for non-termination proving that offers a combination of the following features: it handles programs with non-determinism, provides relative completeness guarantees and supports programs with polynomial arithmetic. Experimental evaluation of our prototype tool RevTerm shows that, despite its simplicity and stronger theoretical guarantees, RevTerm outperforms all non-

termination proving tools that competed in the TermComp'19 competition, both in the number of proved non-terminations and in runtime.

- Chapter 6 considers the *controller synthesis with probability* 1 *reachability guarantees* problem in stochastic dynamical systems. We propose the first learning-based approach for learning and formally verifying neural controllers for almost-sure reachability in stochastic dynamical systems. Our contributions are as follows:
  - *Theory.* Drawing insight from the results on proving almost-sure termination/reachability in PPs, we show that RSMs provide a formal certificate for probability 1 reachability in stochastic dynamical systems. We also show that RSMs can be used to formally prove upper bounds on reachability time.
  - *Automation.* We present a learner-verifier framework which jointly learns and formally verifies a control policy and an RSM, both parametrized as neural networks. This is the first method for formal neural controller synthesis in stochastic dynamical systems. Our method also yields a verification procedure for formally verifying almost-sure reachability under a given neural network control policy. The method is applicable to infinite-time horizon systems with non-polynomial dynamics, thus overcoming the limitations of prior work. Our method only requires that the state space of the system is compact and that the dynamics function is Lipschitz continuous. We experimentally evaluate our method on several non-linear RL environments.
- Chapter 7 considers the *controller synthesis with quantitative reachability and safety guarantees* problem in stochastic dynamical systems. Building on our results in Chapter 6, we propose the first learning-based approach for learning and formally verifying neural controllers for quantitative reachability, safety and reach-avoidance in stochastic dynamical systems. Our contributions are as follows:
  - *Theory.* We introduce a *reach-avoid supermartingale (RASM)*, a martingale-based formal certificate function for proving quantitative reachability, safety and reach-avoidance in stochastic dynamical systems. Our RASMs unify and significantly generalize RSMs for proving probability 1 reachability and stochastic barrier functions (SBFs) for proving quantitative safety.
  - *Automation.* We present a learner-verifier framework which jointly learns and formally verifies a control policy and an RASM, both parametrized as neural networks. This is the first method for formal neural controller synthesis in stochastic dynamical systems that provides quantitative reachability, safety and reach-avoidance guarantees. Our method also yields a verification procedure for formally verifying satisfaction of quantitative specifications under a given neural network control policy. The method is

applicable to infinite-time horizon systems with non-polynomial dynamics, thus overcoming the limitations of prior work. As in Chapter 6, our method only requires that the state space of the system is compact and that the dynamics function is Lipschitz continuous. We experimentally evaluate our method on several nonlinear RL environments.

We conclude this thesis with Chapter 8. In this chapter, in Section 8.1 we first discuss the applicability of theoretical concepts and automation techniques introduced in this thesis to problems stretching beyond PP analysis and control of stochastic dynamical systems. This discussion is based on the works of the author published during the PhD period but which do not constitute a part of thesis. These include applications to static differential cost analysis in programs (Section 8.1.1), formal verification of Bayesian neural networks (Section 8.1.2), formal verification of distributional safety properties in finite-state MDPs (Section 8.1.3) and formal analysis of bidding games on graphs (Section 8.1.4). We then present concluding remarks and discuss directions for future work in Section 8.2.

CHAPTER 2

Preliminaries

2.1 Mathematical Preliminaries

We start by fixing the notation and defining the basic notions from probability and measure theory that will be used throughout this thesis. We postpone introducing more advanced probability and martingale theory concepts to later sections, after we formally define our models for PPs and stochastic dynamical systems. This will allows us to illustrate these advanced concepts on examples relevant for the topic of this thesis.

Sets. We use $\mathbb{N}$, $\mathbb{N}_0$, $\mathbb{Z}$, $\mathbb{Z}^{\geq 0}$, $\mathbb{Q}$, $\mathbb{Q}^{\geq 0}$, $\mathbb{R}$, $\mathbb{R}^{\geq 0}$ to denote the sets of all natural numbers, natural numbers extended with 0, integers, nonnegative integers, rational numbers, nonnegative rational numbers, reals, and nonnegative reals, respectively.

Vectors. We use boldface symbols to denote vectors. For a set S possibly being any of the sets defined above, a natural number $n \in \mathbb{N}$ and an n-dimensional vector $\mathbf{x} \in S^n$, we use $\mathbf{x}[i]$ to denote the i-th component of $\mathbf{x}$ for each $1 \leq i \leq n$. Furthermore, for a scalar value $a \in S$, we write $\mathbf{x}[i \leftarrow a]$ for an n-dimensional vector $\mathbf{y} \in S^n$ with $\mathbf{y}[i] = a$ and $\mathbf{y}[j] = \mathbf{x}[j]$ for each $1 \leq j \leq n$ with $j \neq i$.

Probability space. A *probability space* is a triple $(\Omega, \mathcal{F}, \mathbb{P})$, where Ω is a non-empty set called *sample space*, $\mathcal{F}$ is a σ-algebra over Ω (which is a collection of subsets of Ω that contains the empty set $\emptyset$ and is closed under complementation and countable union), and $\mathbb{P}$ is a *probability measure* over $\mathcal{F}$, i.e. a function $\mathbb{P} : \mathcal{F} \to [0, 1]$ that satisfies the following three properties: (1) $\mathbb{P}[\emptyset] = 0$, (2) $\mathbb{P}[\Omega \backslash A] = 1 - \mathbb{P}[A]$ for each $A \in \mathcal{F}$, and (3) $\mathbb{P}[\cup_{i=0}^{\infty} A_i] = \sum_{i=0}^{\infty} \mathbb{P}[A_i]$ for any sequence $(A_i)_{i=0}^{\infty}$ of pairwise disjoint sets in $\mathcal{F}$. An element of $\mathcal{F}$ is said to be an *event*.

$$
\begin{aligned}
\langle stmt \rangle &::= \langle assgn \rangle \mid \text{'}\textbf{skip}\text{'} \mid \langle stmt \rangle \text{ ';' } \langle stmt \rangle \\
&\mid \text{'}\textbf{if}\text{'} \langle bexpr \rangle \text{ '}\textbf{then}\text{'} \langle stmt \rangle \text{ '}\textbf{else}\text{'} \langle stmt \rangle \text{ '}\textbf{fi}\text{'} \\
&\mid \text{'}\textbf{while}\text{'} \langle predicate \rangle \text{ '}\textbf{do}\text{'} \langle stmt \rangle \text{ '}\textbf{od}\text{'} \\
\langle assgn \rangle &::= \langle pvar \rangle \text{ ':='} \langle expr \rangle \mid \langle pvar \rangle \text{ ':=} \textbf{ndet(}\langle dom \rangle\textbf{)}\text{'} \\
&\mid \langle pvar \rangle \text{ ':=} \textbf{sample(}\langle dist \rangle\textbf{)}\text{'} \\
\langle expr \rangle &::= \langle constant \rangle \mid \langle pvar \rangle \mid \langle expr \rangle \text{ '}\cdot\text{' } \langle expr \rangle \\
&\mid \langle expr \rangle \text{ '}+\text{' } \langle expr \rangle \mid \langle expr \rangle \text{ '}-\text{' } \langle expr \rangle \\
&\mid \langle expr \rangle \text{ '}/\text{' } \langle expr \rangle \mid f(\langle expr \rangle) \\
\langle dom \rangle &::= \text{'}\textbf{Real}\text{'} \mid \text{'}\textbf{Real}[\langle constant \rangle, \langle constant \rangle]\text{'} \\
&\mid \text{'}\textbf{Real}(\langle constant \rangle, \langle constant \rangle]\text{'} \\
&\mid \text{'}\textbf{Real}[\langle constant \rangle, \langle constant \rangle)\text{'} \\
&\mid \text{'}\textbf{Real}(\langle constant \rangle, \langle constant \rangle)\text{'} \\
\langle bexpr \rangle &::= \langle predicate \rangle \mid \star \mid \text{ '}\textbf{prob}\text{(p)'} \\
\langle predicate \rangle &::= \langle literal \rangle \mid \neg \langle literal \rangle \\
&\mid \langle predicate \rangle \text{ '}\textbf{and}\text{' } \langle predicate \rangle \\
&\mid \langle predicate \rangle \text{ '}\textbf{or}\text{' } \langle predicate \rangle \\
\langle literal \rangle &::= \langle expr \rangle \text{ '}\bowtie\text{' } \langle expr \rangle \\
\text{'}\bowtie\text{'} &::= \text{'}\geq\text{'} \mid \text{'}>\text{'} \mid \text{'}<\text{'} \mid \text{'}\leq\text{'} \mid \text{'}=\text{'}
\end{aligned}
$$

Figure 2.1: Syntax grammar of our PPs.

Random variables. Given a probability space $(\Omega, \mathcal{F}, \mathbb{P})$, a *random variable* is an $\mathcal{F}$-measurable function $X : \Omega \to \mathbb{R} \cup \{\pm\infty\}$, i.e. for each $a \in \mathbb{R} \cup \{\pm\infty\}$ we have that $\{\omega \in \Omega \mid X(\omega) \leq a\} \in \mathcal{F}$. We use $\mathbb{E}[X]$ to denote the *expected value* of X (for the formal definition of expected value, see [Wil91]). A *(discrete-time) stochastic process* is a sequence of random variables $(X_i)_{i=0}^{\infty}$ in $(\Omega, \mathcal{F}, \mathbb{P})$.

2.2 Probabilistic Programs

We now define the syntax and semantics of probabilistic programs that we consider in this thesis. We follow standard notation and definitions in probabilistic program analysis [CGMZ22b].

$x = 0$
ℓ_{init} : **while** $x \geq 0$ **do**
ℓ_1 : $r_1 := Uniform([-1, 0.5])$
ℓ_2 : $x := x + r_1$
ℓ_3 : **if** $x \geq 100$ **then**
ℓ_4 : **if** $\star$ **then**
ℓ_5 : $r_2 := Uniform([-1, 2])$
ℓ_6 : $x := x + r_2$
fi fi od
ℓ_{out} :

Figure 2.2: An example PP with non-determinism.

2.2.1 Program Syntax

We consider imperative arithmetic probabilistic programs (PPs) with non-determinism. Our PPs allow standard programming constructs such as conditional branching, while-loops and variable assignments. They also allow two probabilistic constructs – probabilistic branching which is indicated in the syntax by a command '**if prob(**p**) then** ...' with $p \in [0, 1]$ a real constant, and sampling instructions of the form $x := d$ where d is a probability distribution. Sampling instructions may contain both discrete (e.g. Bernoulli or Poisson) and continuous (e.g. uniform or normal) probability distributions.

We also allow constructs for (demonic) non-determinism. We allow non-deterministic branching indicated in the syntax by '**if** $\star$ **then** ...', and non-deterministic assignments indicated by an instruction of the form $x :=$ **ndet(**$[a, b]$**)**, where $a, b \in \mathbb{R} \cup \{\pm\infty\}$ with $[a, b]$ being a (possibly unbounded) real interval from which the new variable value is chosen non-deterministically. We also allow one or both sides of the interval to be open. The complete syntax of our programs is presented in Figure 2.1.

Example 2.2.1. *We present an example PP with non-determinism in Figure 2.2. This PP is taken from [CGMZ22b] and we will consider it again in Chapter 4. The PP consists of a single loop which contains sampling instructions, conditional branching and non-deterministic branching. Whenever* r_1 *and* r_2 *are evaluated, their values are sampled from the uniform distributions* $Uniform([-1, 0.5])$ *and* $Uniform([-1, 2])$*, independently from previous samples. The command* ***if*** $\star$ ***then*** *denotes non-deterministic branching. The labels* ℓ_{init}*,* $\ell_1, \ldots, \ell_6$ *and* ℓ_{out} *denote locations in the program's probabilistic control-flow graph which we will later use as a formal model for PPs.*

Variables, expressions and predicates. Variables in our PPs are real-valued. Given a finite set of variables V, a *variable valuation* of V is a vector $\mathbf{x} \in \mathbb{R}^{|V|}$.

Arithmetic *expressions* in our PPs are built from constants, program variables and standard Borel-measurable [Bil95] arithmetic operators. We also allow sampling instructions to appear on right-hand sides of variable assignments as linear terms. An expression with no such terms is called *sampling-free*. We allow sampling from both discrete and continuous distributions. We denote by $\mathcal{D}$ the set of distributions appearing in the program with each $d \in \mathcal{D}$ assumed to be *integrable*, i.e. $\mathbb{E}_{X\sim d}[|X|] < \infty$. We write $X \sim d$ to denote that X is a random variable with probability distribution d. This is to ensure that expected value of each d over any measurable set is well-defined and finite.

A *predicate* over a set of variables V is a Boolean combination of *atomic predicates* of the form $E \leq E'$, where E, E' are sampling-free expressions whose all variables are from V. We denote by $\mathbf{x} \models \Psi$ the fact that the predicate Ψ is satisfied by substituting values of $\mathbf{x}$ for the corresponding variables in Ψ.

Linear and polynomial PPs. While all theoretical results on PPs in this thesis apply to general PPs in which arithmetic expressions may contain arbitrary Borel-measurable arithmetic operators, the algorithms for automating our PP analyses will be restricted either to affine (or linear) arithmetic PPs or to polynomial arithmetic PPs. We say that a PP is *affine (or linear)* if all arithmetic expressions appearing in it are affine expressions over program variables. Similarly, we say that a PP is *polynomial* if all arithmetic expressions appearing in it are polynomial expressions over program variables.

Conditioning. Note that our syntax in Figure 2.1 does not contain constructs for conditioning. Many PP languages support conditioning constructs indicated in the syntax by an instruction of the form **observe**(ϕ), where ϕ is a predicate over PP variables [BGG$^+$13, GAB$^+$13]. Intuitively, whenever such an instruction is executed, the PP "observes" whether the current valuation of program variables satisfies the predicate ϕ and rejects the execution if the predicate is not satisfied. A PP execution is said to be accepted if it is not rejected by any **observe**-instruction. The semantics of the PP are then defined only over accepted executions. There are two reasons behind omitting conditioning constructs from our PPs:

1. For PPs without non-determinism, it was shown in [OGJ$^+$18, Section 7.2] that reachability and safety analyses in PPs that contain **observe**-instructions can be reduced to reachability and safety analyses in semantically equivalent PPs that do not contain **observe**-instructions, assuming that the probability of accepted PP executions is strictly greater than 0.

2. For PPs with non-determinism, it is unclear how to formally define their semantics when extended with conditioning constructs, see [OGJ$^+$18, Section 6].

Hence, since for PPs without non-determinism we can remove **observe**-instructions and for PPs with non-determinism semantical issues arise if we try to extend them with conditioning constructs, we assume that our PPs are conditioning-free.

2.2.2 Program Semantics

Probabilistic control-flow graphs (pCFGs). We model our PPs via probabilistic control-flow graphs (pCFGs) [CNZ17, ACN18, CFNH18]. Each PP can be straightforwardly translated to an equivalent pCFG, see [CFNH18]. A *probabilistic control-flow graph (pCFG)* is a tuple $\mathcal{C} = (L, V, \ell_{init}, \mathbf{x}_{init}, \mapsto, G, Pr, Up)$, where:

- L is a finite set of *locations*, partitioned into locations of *conditional branching* L_C, *probabilistic branching* L_P, *non-deterministic branching* L_N and *assignment* L_A.
- $V = \{x_1, \ldots, x_{|V|}\}$ is a finite set of *program variables*;
- ℓ_{init} is the *initial program location*;
- $\mathbf{x}_{init} \in \mathbb{R}^{|V|}$ is the initial variable valuation;
- $\mapsto \subseteq L \times L$ is a finite set of *transitions*. For each transition $\tau = (\ell, \ell')$, we say that ℓ is its *source location* and ℓ' its *target location*;
- G is a map assigning to each transition $\tau = (\ell, \ell') \in \mapsto$ with $\ell \in L_C$ a *guard* $G(\tau)$, which is a logical formula over V specifying whether τ can be executed;
- Pr is a map assigning to each transition $\tau = (\ell, \ell') \in \mapsto$ with $\ell \in L_P$ a *probability* $Pr(\tau) \in [0, 1]$. We require $\sum_{\tau=(\ell,\_)} Pr(\tau) = 1$ for each $\ell \in L_P$;
- Up is a map assigning to each transition $\tau = (\ell, \ell') \in \mapsto$ with $\ell \in L_A$ an *update* $Up(\tau) = (j, u)$ where $j \in \{1, \ldots, |V|\}$ is a *target variable index* and u is an *update element* which can be:
  - the bottom element $u = \bot$, denoting no update;
  - a Borel-measurable expression $u : \mathbb{R}^{|V|} \to \mathbb{R}$, denoting a deterministic variable assignment;
  - a probability distribution $u = d$, denoting that the new variable value is sampled according to d. We assume that d is integrable, i.e. that $\mathbb{E}_{X\sim d}[|X|] < \infty$ where $X \sim d$ denotes a random variable distributed according to d;
  - an interval $u = [a, b] \subseteq \mathbb{R} \cup \{\pm\infty\}$, denoting a non-deterministic update. We also allow one or both sides of the interval to be open.

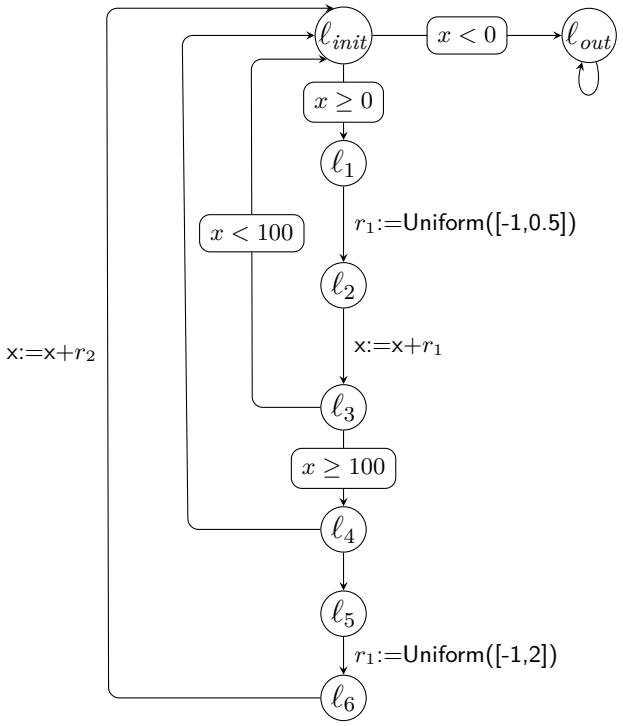

Figure 2.3: pCFG for the PP in Figure 2.2.

We assume the existence of the special *terminal location* denoted by ℓ_{out}. This location only has a self-loop with a trivial guard and identity update as an outgoing transition. We also require that each location has at least one outgoing transition, and that each $\ell \in L_A$ has a unique outgoing transition. For each location $\ell \in L_C$, we assume that the disjunction of guards of all transitions outgoing from ℓ is equivalent to $true$, i.e. $\bigvee_{\tau=(l,\_)} G(\tau) \equiv true$. This is done to ensure that it is always possible to execute at least one transition, and is without loss of generality as we may introduce an additional transition from ℓ to ℓ_{out}.

Example 2.2.2. *The pCFG for the program in Figure 2.2 is provided in Figure 2.3.*

States, paths and runs. A *state* in a pCFG $\mathcal{C}$ is a tuple $(\ell, \mathbf{x})$, where ℓ is a location in $\mathcal{C}$ and $\mathbf{x} \in \mathbb{R}^{|V|}$ is a variable valuation of V. We say that a transition $\tau = (\ell, \ell')$ is *enabled* at a state $(\ell, \mathbf{x})$ if $\mathbf{x} \models G(\tau)$. We say that a state $(\ell', \mathbf{x}')$ is a *successor* of $(\ell, \mathbf{x})$, if there exists an enabled transition $\tau = (\ell, \ell')$ in $\mathcal{C}$ such that $(\ell', \mathbf{x}')$ can be reached from $(\ell, \mathbf{x})$ by executing τ, i.e. we can obtain $\mathbf{x}'$ by applying the updates of τ to $\mathbf{x}$, if any. A *finite path* in $\mathcal{C}$ is a sequence $(\ell_0, \mathbf{x}_0), (\ell_1, \mathbf{x}_1), \ldots, (\ell_k, \mathbf{x}_k)$ of states with $(\ell_0, \mathbf{x}_0) = (\ell_{init}, \mathbf{x}_{init})$ and with $(\ell_{i+1}, \mathbf{x}_{i+1})$ being a successor of $(\ell_i, \mathbf{x}_i)$ for each

$0 \leq i \leq k-1$. A state $(\ell, \mathbf{x})$ is *reachable* in $\mathcal{C}$ if there exists a finite path in $\mathcal{C}$ that ends in $(\ell, \mathbf{x})$. A *run* (or *execution*) in $\mathcal{C}$ is an infinite sequence of states where each finite prefix is a finite path. We use $State_{\mathcal{C}}$, $Fpath_{\mathcal{C}}$, $Run_{\mathcal{C}}$, $\text{Reach}_{\mathcal{C}}$ to denote the set of all states, finite paths, runs and reachable states in $\mathcal{C}$, respectively. Finally, we use $State_{term}$ to denote the set $\{(\ell_{out}, \mathbf{x}) \mid \mathbf{x} \in \mathbb{R}^{|V|}\}$ of terminal states.

Schedulers. The behavior of a pCFG may be captured by defining a probability space over the set of all runs in the pCFG. For this to be done, however, we need to resolve non-determinism and this is achieved via the standard notion of a scheduler. A *scheduler* in a pCFG $\mathcal{C}$ is a map σ which to each finite path $\rho \in Fpath_{\mathcal{C}}$ assigns a probability distribution $\sigma(\rho)$ over successor states of the last state in ρ. Since we deal with programs operating over real-valued variables, the set $Fpath_{\mathcal{C}}$ may be uncountable. To that end, we impose an additional *measurability* assumption on schedulers, in order to ensure that the semantics of probabilistic programs with non-determinism is defined in a mathematically sound way. The restriction to measurable schedulers is standard [NK07, NSK09] hence we omit the formal definition.

Semantics of pCFGs. A pCFG $\mathcal{C}$ together with a scheduler σ define a stochastic process taking values in the set of states of $\mathcal{C}$, whose trajectories correspond to runs in $\mathcal{C}$. The process evolves as follows: we start in the initial state $(\ell_{init}, \mathbf{x}_{init})$ and inductively extended the path. Suppose that, at time step i, the path produced so far is ρ_i and its last state is $(\ell_i, \mathbf{x}_i)$. Depending on the type of the location ℓ_i, the next state $(\ell_{i+1}, \mathbf{x}_{i+1})$ is chosen as follows:

- If $\ell_i \in L_C$, let $\tau = (\ell_i, \ell')$ be the unique transition enabled at $(\ell_i, \mathbf{x}_i)$. Then $(\ell_{i+1}, \mathbf{x}_{i+1}) = (\ell', \mathbf{x}_i)$;
- If $\ell_i \in L_P$, sample $\tau = (\ell_i, \ell')$ from the set of all transitions outgoing from ℓ_i according to the distribution defined by Pr at ℓ_i. Then $(\ell_{i+1}, \mathbf{x}_{i+1}) = (\ell', \mathbf{x}_i)$;
- If $\ell_i \in L_N$, sample $\tau = (\ell_i, \ell')$ from the set of all transitions outgoing from ℓ_i according to the distribution $\sigma(\rho_i)$. Then $(\ell_{i+1}, \mathbf{x}_{i+1}) = (\ell', \mathbf{x}_i)$;
- If $\ell_i \in L_A$, let $\tau = (\ell_i, \ell')$ be the unique transition outgoing from ℓ_i and let $Up(\tau) = (j, u)$. Then:
  - If $u = \perp$, then $(\ell_{i+1}, \mathbf{x}_{i+1}) = (\ell', \mathbf{x}_i)$;
  - If $u : \mathbb{R}^{|V|} \to \mathbb{R}$ is a Borel-measurable expression, then $(\ell_{i+1}, \mathbf{x}_{i+1}) = (\ell', \mathbf{x}_i[x_j \leftarrow u(\mathbf{x}_i)])$;
  - If $u = d$ is a probability distribution, then sample X according to u and $(\ell_{i+1}, \mathbf{x}_{i+1}) = (\ell', \mathbf{x}_i[x_j \leftarrow X])$;

– If $u = [a, b]$ is a real interval, then sample X according to $\sigma(\rho_i)$ and $(\ell_{i+1}, \mathbf{x}_{i+1}) = (\ell', \mathbf{x}_i[x_j \leftarrow X])$.

Formally, a pCFG $\mathcal{C}$ and a scheduler σ define a probability space $(Run_{\mathcal{C}}, \mathcal{F}_{\mathcal{C}}, \mathbb{P}^{\sigma}_{(\ell_{init}, \mathbf{x}_{init})})$ over the set of all runs in $\mathcal{C}$, and a stochastic process $\mathcal{C}^{\sigma} = \{\mathbf{C}^{\sigma}_i\}_{i=0}^{\infty}$ in this space such that for each run $\rho \in Run_{\mathcal{C}}$ we have that $\mathbf{C}^{\sigma}_i(\rho)$ is the i-th configuration along the run ρ. The sigma-algebra $\mathcal{F}_{\mathcal{C}}$ is the smallest (with respect to set inclusion) sigma-algebra under which all the functions $\mathbf{C}^{\sigma}_i$, for all $i \geq 0$, are $\mathcal{F}_{\mathcal{C}}$-measurable, i.e. for each $\mathbf{C}^{\sigma}_i$ and each Borel-measurable set $B \in \mathcal{B}(\mathbb{R}^{|V|})$ it holds that $\{\rho \mid \mathbf{C}^{\sigma}_i(\rho) = (\ell, \mathbf{x}) \text{ with } \mathbf{x} \in B\} \in \mathcal{F}_{\mathcal{C}}$. The formal construction of $\mathbb{P}^{\sigma}_{(\ell_{init}, \mathbf{x}_{init})}$ proceeds via the standard *cylinder construction* [ADD00, Theorem 2.7.2]. We denote by $\mathbb{E}^{\sigma}_{(\ell_{init}, \mathbf{x}_{init})}$ the expectation operator in the probability space $(Run_{\mathcal{C}}, \mathcal{F}_{\mathcal{C}}, \mathbb{P}^{\sigma}_{(\ell_{init}, \mathbf{x}_{init})})$.

2.2.3 Reachability, Safety and Termination Analysis

We now formally define the qualitative and quantitative reachability and safety analysis problems in PPs. We then define the termination analysis problems in PPs and show that reachability analysis can be reduced to termination analysis and vice versa. While these reductions will not be helpful for designing more efficient formal analyses, termination analysis in more commonly studied in PP analysis literature and our goal here is to emphasize that these are the same problems.

Consider a pCFG $\mathcal{C} = (L, V, \ell_{init}, \mathbf{x}_{init}, \mapsto, G, Pr, Up)$ associated to some PP and a set S of states in $\mathcal{C}$. Reachability analysis with respect to S reasons about the infimum probability of a random program run *reaching* a state in S. Similarly, safety analysis with respect to S reasons about the infimum probability of a random program run *not reaching* a state in S. In both cases, the infimum is taken over all possible schedulers that can be used to resolve non-determinism in $\mathcal{C}$. Note, however, that these definitions are not yet formal. Since our PPs may contain continuous probability distributions, their state spaces may be uncountable and reachability and safety probabilities with respect to an arbitrary set of states S need not be mathematically well-defined. To that end, we need to impose the necessary measurability assumptions on S and we achieve this by considering sets of states induced by predicate functions.

Predicate functions. A *predicate function* in $\mathcal{C}$ is a map S that to every location $\ell \in L$ assigns a logical formula $S(\ell)$ over program variables in V. We require that $\{\mathbf{x} \mid \mathbf{x} \models S(\ell)\} \subseteq \mathbb{R}^{|V|}$ is a Borel-measurable set for each location $\ell \in L$. Note that each predicate function naturally defines a set of states $(\ell, \mathbf{x})$ in $\mathcal{C}$ for which $\mathbf{x} \models S(\ell)$. With a slight abuse of notation, we also use S to refer to this set of states.

Reachability analysis. Given a predicate function S in $\mathcal{C}$, we let Reach(S) be the set of all runs in $\mathcal{C}$ that reach a state in S, i.e.

$$\text{Reach}(S) = \Big\{\rho \in \mathit{Run}_{\mathcal{C}} \mid \rho \text{ reaches } (\ell, \mathbf{x}) \text{ with } \mathbf{x} \models S(\ell)\Big\}.$$

Then, the reachability analysis problems with respect to S are defined as follows:

- *Qualitative reachability.* Prove that $\inf_\sigma \mathbb{P}^\sigma[\text{Reach}(S)] = 1$.
- *Quantitative reachability.* Given $p \in [0, 1]$, prove that $\inf_\sigma \mathbb{P}^\sigma[\text{Reach}(S)] \geq p$.

We say that $\inf_\sigma \mathbb{P}^\sigma[\text{Reach}(S)]$ is the *reachability probability* of S in $\mathcal{C}$.

Safety analysis. Given a predicate function S in $\mathcal{C}$, we let Safe(S) be the set of all runs in $\mathcal{C}$ that do not reach a state in S, i.e.

$$\text{Safe}(S) = \Big\{\rho \in \mathit{Run}_{\mathcal{C}} \mid \rho \text{ does not reach } (\ell, \mathbf{x}) \text{ with } \mathbf{x} \models S(\ell)\Big\} = \mathit{Run}_{\mathcal{C}} \backslash \text{Reach}(S).$$

Then, the safety analysis problems with respect to S are defined as follows:

- *Qualitative safety.* Prove that $\inf_\sigma \mathbb{P}^\sigma[\text{Safe}(S)] = 1$.
- *Quantitative safety.* Given $p \in [0, 1]$, prove that $\inf_\sigma \mathbb{P}^\sigma[\text{Safe}(S)] \geq p$.

We say that $\inf_\sigma \mathbb{P}^\sigma[\text{Safe}(S)]$ is the *safety probability* of S in $\mathcal{C}$.

Termination analysis. A state $(\ell, \mathbf{x})$ in $\mathcal{C}$ is said to be a *terminal state* if $\ell = \ell_{out}$. A run $\rho \in \mathit{Run}_{\mathcal{C}}$ is said to be *terminating* if it reaches some terminal state in $\mathcal{C}$. We use $\mathit{Term} \subseteq \mathit{Run}_{\mathcal{C}}$ to denote the set of all terminating runs in $\mathit{Run}_{\mathcal{C}}$. Then, the termination analysis problems with respect to S are defined as follows:

- *Qualitative termination.* Prove that $\inf_\sigma \mathbb{P}^\sigma[\mathit{Term}] = 1$.
- *Quantitative termination.* Given $p \in [0, 1]$, prove that $\inf_\sigma \mathbb{P}^\sigma[\mathit{Term}] \geq p$.

We say that $\inf_\sigma \mathbb{P}^\sigma[\mathit{Term}]$ is the *termination probability* of $\mathcal{C}$.

From reachability to termination. Note that termination analysis is just an instance of reachability analysis with respect to the set of all terminal states in $\mathcal{C}$. On the other hand, given a predicate function S, one can trivially reduce reachability analysis with respect to S to termination analysis in a modified pCFG. This is achieved by simply introducing a new terminal location ℓ'_{out} and letting the pCFG transition to ℓ'_{out} upon reaching a state in S. Hence, while in what follows we will focus on termination analysis to follow the convention in PP analysis literature, all our results on termination analysis are immediately applicable to reachability analysis as well.

2.3 Stochastic Dynamical Systems

We now define the model for stochastic dynamical systems considered in this thesis, as well as reachability, safety and reach-avoidance analysis problems. We follow standard terminology and definitions in stochastic dynamical systems [LZCH22, ZLHC23].

2.3.1 Formal Definitions

Stochastic dynamical systems. A discrete-time *stochastic dynamical system* is defined by an equation of the form

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t, \omega_t),\ \mathbf{x}_0 \in \mathcal{X}_0.$$

The function $f : \mathcal{X} \times \mathcal{U} \times \mathcal{N} \to \mathcal{X}$ defines system dynamics, where

- $\mathcal{X} \subseteq \mathbb{R}^m$ is a Borel-measurable *system state space*,
- $\mathcal{U} \subseteq \mathbb{R}^n$ is a Borel-measurable *control action space*, and
- $\mathcal{N} \subseteq \mathbb{R}^p$ is a Borel-measurable *stochastic disturbance space*.

We use $t \in \mathbb{N}_0$ to denote the time index, $\mathbf{x}_t \in \mathcal{X}$ the state of the system, $\mathbf{u}_t \in \mathcal{U}$ the action and $\omega_t \in \mathcal{N}$ the stochastic disturbance vector at time t. The set $\mathcal{X}_0 \subseteq \mathcal{X}$ is the set of initial states. The action $\mathbf{u}_t$ is chosen according to a *control policy* $\pi : \mathcal{X} \to \mathcal{U}$, i.e. $\mathbf{u}_t = \pi(\mathbf{x}_t)$. The stochastic disturbance vector ω_t is sampled according to a specified probability distribution d over $\mathcal{N}$. We assume that the dynamics function f and the control policy π are continuous functions.

Semantics. A sequence $(\mathbf{x}_t, \mathbf{u}_t, \omega_t)_{t\in\mathbb{N}_0}$ of state-action-disturbance triples is a trajectory of the system, if for each $t \in \mathbb{N}_0$ we have $\mathbf{u}_t = \pi(\mathbf{x}_t)$, $\omega_t \in supp(d)$ and $\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t, \omega_t)$. For each initial state $\mathbf{x}_0 \in \mathcal{X}$ and continuous control policy $\pi : \mathcal{X} \to \mathcal{U}$, the system induces a Markov process which gives rise to the probability

space over the set of all trajectories that start in $\mathbf{x}_0$ [Put94]. We denote the probability measure and the expectation in this probability space by $\mathbb{P}^{\pi}_{\mathbf{x}_0}$ and $\mathbb{E}^{\pi}_{\mathbf{x}_0}$. When π and $\mathbf{x}_0$ are clear from the context, we omit them and write $\mathbb{P}$ and $\mathbb{E}$, respectively.

2.3.2 Reachability, Safety and Reach-avoidance Analysis

Consider a stochastic dynamical system defined as above. We now formally define controller synthesis problems under reachability, safety and reach-avoidance specifications.

Reachability analysis. Let $\mathcal{X}_t \subseteq \mathcal{X}$ be a Borel-measurable *target set*. Let

$$\mathsf{Reach}(\mathcal{X}_t) = \left\{ (\mathbf{x}_t, \mathbf{u}_t, \omega_t)_{t \in \mathbb{N}_0} \mid \exists t \in \mathbb{N}_0 . \, \mathbf{x}_t \in \mathcal{X}_t \right\}$$

be the set of all trajectories that reach the target set $\mathcal{X}_t$. The reachability controller synthesis problems with respect to $\mathcal{X}_t$ are defined as follows:

- *Qualitative reachability.* Synthesize a control policy π such that, for every initial state $\mathbf{x}_0 \in \mathcal{X}_0$, we have $\mathbb{P}^{\pi}_{\mathbf{x}_0}[\mathsf{Reach}(\mathcal{X}_t)] = 1$.
- *Quantitative reachability.* Given $p \in [0, 1]$, synthesize a control policy π such that, for every initial state $\mathbf{x}_0 \in \mathcal{X}_0$, we have $\mathbb{P}^{\pi}_{\mathbf{x}_0}[\mathsf{Reach}(\mathcal{X}_t)] \geq p$.

Safety analysis. Let $\mathcal{X}_u \subseteq \mathcal{X}$ be a Borel-measurable *unsafe set*. Let

$$\mathsf{Safe}(\mathcal{X}_u) = \left\{ (\mathbf{x}_t, \mathbf{u}_t, \omega_t)_{t \in \mathbb{N}_0} \mid \forall t \in \mathbb{N}_0 . \, \mathbf{x}_t \notin \mathcal{X}_u \right\}$$

be the set of all trajectories that do not visit the unsafe set $\mathcal{X}_u$. The safety controller synthesis problems with respect to $\mathcal{X}_u$ are defined as follows:

- *Qualitative safety.* Synthesize a control policy π such that, for every initial state $\mathbf{x}_0 \in \mathcal{X}_0$, we have $\mathbb{P}^{\pi}_{\mathbf{x}_0}[\mathsf{Safe}(\mathcal{X}_u)] = 1$.
- *Quantitative safety.* Given $p \in [0, 1]$, synthesize a control policy π such that, for every initial state $\mathbf{x}_0 \in \mathcal{X}_0$, we have $\mathbb{P}^{\pi}_{\mathbf{x}_0}[\mathsf{Safe}(\mathcal{X}_u)] \geq p$.

Reach-avoidance analysis. Let $\mathcal{X}_t, \mathcal{X}_u \subseteq \mathcal{X}$ be disjoint Borel-measurable sets, called the *target set* and the *unsafe set*, respectively. Let

$$\mathsf{ReachAvoid}(\mathcal{X}_t, \mathcal{X}_u) = \left\{ (\mathbf{x}_t, \mathbf{u}_t, \omega_t)_{t \in \mathbb{N}_0} \mid \exists t \in \mathbb{N}_0 . \, \mathbf{x}_t \in \mathcal{X}_t \wedge (\forall t' \leq t . \, \mathbf{x}_{t'} \notin \mathcal{X}_u) \right\}$$

be the set of all trajectories that reach $\mathcal{X}_t$ without reaching $\mathcal{X}_u$. The reach-avoid controller synthesis problems with respect to $\mathcal{X}_t$ and $\mathcal{X}_u$ are defined as follows:

- *Qualitative reach-avoidance.* Synthesize a control policy π such that, for every initial state $\mathbf{x}_0 \in \mathcal{X}_0$, we have $\mathbb{P}^{\pi}_{\mathbf{x}_0}[\text{ReachAvoid}(\mathcal{X}_t, \mathcal{X}_u))] = 1$.
- *Quantitative reach-avoidance.* Given $p \in [0, 1]$, synthesize a control policy π such that, for every initial state $\mathbf{x}_0 \in \mathcal{X}_0$, we have $\mathbb{P}^{\pi}_{\mathbf{x}_0}[\text{ReachAvoid}(\mathcal{X}_t, \mathcal{X}_u)] \geq p$.

2.4 Martingale Theory

We now provide definitions and results from probability and martingale theory that lie at the core of our formal reasoning about infinite state stochastic systems. We first define conditional expectation, which is a technical concept necessary to define (super)martingales. We then define stopping times, which are random variables that represent times at which some probabilistic events happen. Finally, we define (super)martingale processes. All concepts are illustrated on examples to help develop intuition before using these concepts for stochastic system analysis.

Conditional expectation. Let X be a random variable in a probability space $(\Omega, \mathcal{F}, \mathbb{P})$. Let $\mathcal{F}' \subseteq \mathcal{F}$ be a sub-σ-algebra. Intuitively, conditional expectation of X given $\mathcal{F}'$ is an $\mathcal{F}'$-measurable random variable that behaves like X whenever its expected value is evaluated over an event in $\mathcal{F}'$. Formally, a *conditional expectation* of X given $\mathcal{F}'$ is a random variable Y such that

- Y is $\mathcal{F}'$-measurable, and
- for each $A \in \mathcal{F}'$, we have $\mathbb{E}[X \cdot \mathbb{I}(A)] = \mathbb{E}[Y \cdot \mathbb{I}(A)]$,

where $\mathbb{I}(A) : \Omega \to \{0, 1\}$ is an *indicator function* of A defined via $\mathbb{I}(A)(\omega) = 1$ if $\omega \in A$, and $\mathbb{I}(A)(\omega) = 0$ if $\omega \notin A$. It is know that conditional expectation of X given $\mathcal{F}'$ is guaranteed to exist whenever X is real-valued and nonnegative or real-valued and integrable [Wil91]. Moreover, conditional expectation is almost-surely unique whenever it exists, meaning that $\mathbb{P}[Y = Y'] = 1$ for any two conditional expectations Y and Y' of X given $\mathcal{F}'$. Therefore, one may pick one such random variable as a canonical conditional expectation and denote it by $\mathbb{E}[X \mid \mathcal{F}']$.

Stopping time. A *filtration* in a probability space $(\Omega, \mathcal{F}, \mathbb{P})$ is a sequence $(\mathcal{F}_i)_{i=0}^{\infty}$ of sub-σ-algebras of $\mathcal{F}$ which is increasing under set inclusion, so that $\mathcal{F}_i \subseteq \mathcal{F}_{i+1}$ for each $i \in \mathbb{N}_0$. A *stopping time* with respect to the filtration $(\mathcal{F}_i)_{i=0}^{\infty}$ is a random variable $T : \Omega \to \mathbb{N}_0 \cup \{\infty\}$ such that $\{\omega \in \Omega \mid T(\omega) \leq i\} \in \mathcal{F}_i$ for each $i \in \mathbb{N}_0$. Intuitively, a stopping time describes at which time step should a process be stopped, and the condition $\{\omega \in \Omega \mid T(\omega) \leq i\} \in \mathcal{F}_i$ says that the decision to stop at time i is based solely on the information available up to time i.

Example 2.4.1 (Canonical filtration and reachability time)**.** *Let $\mathcal{C}$ be a pCFG associated to a PP, σ be a scheduler in $\mathcal{C}$ and S be a predicate function in $\mathcal{C}$. Consider the probability space $(\Omega_{\mathcal{C}}, \mathcal{F}_{\mathcal{C}}, \mathbb{P}^{\sigma})$ of all runs in $\mathcal{C}$. The* canonical filtration *$(\mathcal{R}_i)_{i=0}^{\infty}$ in this probability space is defined by letting each sub-sigma-algebra $\mathcal{R}_i$ of $\mathcal{F}_{\mathcal{C}}$ contain all sets $A \in \mathcal{F}_{\mathcal{C}}$ of runs in Ω whose finite path prefix of length i satisfies some property. An important example of a stopping time with respect to $(\mathcal{R}_i)_{i=0}^{\infty}$ in $(\Omega_{\mathcal{C}}, \mathcal{F}_{\mathcal{C}}, \mathbb{P}^{\sigma})$ is the* reachability time *T_S of S. For a run $\rho = (\ell_i, \mathbf{x}_i)_{i=0}^{\infty}$ in $\mathcal{C}$, the reachability time of S is defined as the first hitting time of the set of states in S,i.e. $T_S(\rho) = \inf\{t \in \mathbb{N}_0 \mid \mathbf{x}_t \models S(\ell_t)\}$. If S is the set of all terminal states in $\mathcal{C}$, this gives rise to the* termination time *$TimeTerm$.*

Supermartingales. Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space and $(\mathcal{F}_i)_{i=0}^{\infty}$ be a filtration in it. A *supermartingale* with respect to $(\mathcal{F}_i)_{i=0}^{\infty}$ is a stochastic process $(X_i)_{i=0}^{\infty}$ such that

- each X_i is $\mathcal{F}_i$-measurable,
- each conditional expectation $\mathbb{E}[X_{i+1} \mid \mathcal{F}_i]$ exists, and
- $\mathbb{E}[X_{i+1} \mid \mathcal{F}_i](\omega) \leq X_i(\omega)$ hold for each $\omega \in \Omega$ and $i \geq 0$.

Intuitively, the last condition says that the expected value of X_{i+1} given the value of X_i has to decrease. This condition is formalized by using conditional expectation. If in the last condition we have equality for each $\omega \in \Omega$ and $i \geq 0$, we then say that $(X_i)_{i=0}^{\infty}$ is a *martingale*.

Example 2.4.2 (Random walk)**.** *Consider a biased 1-dimensional random walk over the set of integers $\mathbb{Z}$. The random walk starts at 0 and then in each time step mover one step to the left with probability $2/3$ and one step to the right with probability $1/3$. Formally, this random walk defines a stochastic process $(X_i)_{i=0}^{\infty}$ with $X_0 = 0$ and*

$$X_{i+1} = \begin{cases} X_i - 1, & \text{with probability } 2/3, \\ X_i + 1, & \text{with probability } 1/3, \end{cases}$$

for each $i \in \mathbb{N}$. It is easy to verify from the definition of conditional expectation that $\mathbb{E}[X_{i+1} \mid X_i]$ exists for each $i \in \mathbb{N}_0$ and that it almost-surely equals

$$\mathbb{E}[X_{i+1} \mid X_i] = 2/3 \cdot (X_i - 1) + 1/3 \cdot (X_i + 1) = X_i - 1/3 \leq X_i.$$

Hence, the stochastic process $(X_i)_{i=0}^{\infty}$ is a supermartingale.

2.5 Fixed-point Theory

We conclude these preliminaries by providing an overview of basic notions from fixed point theory, that will be used in Section 4.

Partial order. Let $\mathcal{L}$ be a set. A binary relation $\sqsubseteq$ on $\mathcal{L}$ is said to be a *partial order* if it satisfies the following three properties:

- *Reflexivity:* $x \sqsubseteq x$ for each $x \in \mathcal{L}$,
- *Antisymmetry:* $x \sqsubseteq y \wedge y \sqsubseteq x \Rightarrow x = y$ for each $x, y \in \mathcal{L}$, and
- *Transitivity:* $x \sqsubseteq y \wedge y \sqsubseteq z \Rightarrow x \sqsubseteq z$ for each $x, y, z \in \mathcal{L}$.

Suprema and infima. Given a set $\mathcal{L}$, a partial order $\sqsubseteq$ over it and a subset $K \subseteq \mathcal{L}$, an element $u \in \mathcal{L}$ is said to be an *upper bound* of K if $k \sqsubseteq u$ holds for all $k \in K$. Similarly, $l \in \mathcal{L}$ is said to be a *lower bound* for K if $l \sqsubseteq k$ holds for all $k \in K$. The *supremum* $\sqcup K$ of K is a smallest upper bound of K with respect to $\sqsubseteq$, i.e. an upper bound such that for any other upper bound u of K we have $\sqcup K \sqsubseteq u$. Similarly, the *infimum* $\sqcap K$ of K is a largest lower bound of K with respect to $\sqsubseteq$, i.e. a lower bound such that for any other lower bound l of K we have $l \sqsubseteq \sqcap K$.

Lattice. A partial order $(\mathcal{L}, \sqsubseteq)$ is said to be a *lattice* if $\mathcal{L}$ is non-empty and the supremum $x \sqcup y$ and the infimum $x \sqcap y$ exist for every two elements $\{x, y\} \subseteq \mathcal{L}$. A lattice is said to be ω-*complete* if for any ascending chain $x_1 \sqsubseteq x_2 \sqsubseteq \ldots$ in $\mathcal{L}$ there exists the supremum of the chain $\sqcup_{i=1}^{\infty} x_i$.

Monotone and ω-continuous functions. Given a partial order $(\mathcal{L}, \sqsubseteq)$, a function $f : \mathcal{L} \to \mathcal{L}$ is *monotone* if for every $x_1 \sqsubseteq x_2$ in $\mathcal{L}$ we have $f(k_1) \sqsubseteq f(k_2)$. Given an ω-complete lattice $(\mathcal{L}, \sqsubseteq)$, a function $f : \mathcal{L} \to \mathcal{L}$ is ω-*continuous* if for every ascending chain $x_1 \sqsubseteq x_2 \sqsubseteq \ldots$ in $\mathcal{L}$ we have $f(\bigsqcup\{x_n\}_{i=0}^{\infty}) = \bigsqcup\{f(x_i)\}_{i=0}^{\infty}$.

Fixed Points. Given an ω-complete lattice $(\mathcal{L}, \sqsubseteq)$ and a function $f : \mathcal{L} \to \mathcal{L}$, an element $x \in \mathcal{L}$ is a *pre-fixed point* if $f(x) \sqsubseteq x$, a *post-fixed point* if $f(x) \sqsupseteq x$ and a *fixed point* if $f(x) = x$. The *least fixed point* of f, denoted by $\mathrm{lfp} f$, is the fixed point that is smaller than any other fixed point. Analogously, the *greatest fixed point* of f, denoted by $\mathrm{gfp} f$, is the fixed point that is larger than any other fixed point.

CHAPTER 3

Lexicographic Methods for Almost-sure Termination Analysis in PPs

This section is based on the following publications:

- Krishnendu Chatterjee, Ehsan Kafshdar Goharshady, Petr Novotný, Jiří Zárevúcky, Đorđe Žikelić.[†] *On Lexicographic Proof Rules for Probabilistic Termination*. In Formal Methods - 24th International Symposium, **FM 2021**
- Krishnendu Chatterjee, Ehsan Kafshdar Goharshady, Petr Novotný, Jiří Zárevúcky, Đorđe Žikelić.[†] *On Lexicographic Proof Rules for Probabilistic Termination*. In Formal Aspects of Computing, **FAC 2023**

3.1 Introduction

Qualitative termination/reachability analysis. In this section, we consider the qualitative (a.k.a. almost-sure) termination/reachability analysis problem in PPs. Recall, in Section 2.2.3 we showed that reachability analysis in PPs can be reduced to termination analysis in PPs. While ranking supermartingales (RSMs) [CS13] present a prominent approach to proving almost-sure termination, they are not compositional and are hard to synthesize in PPs with complex control-flow structure. Lexicographic ranking supermartingales (LexRSMs) [ACN18] present a multi-dimensional extension

[†]Authors ordered alphabetically.

of RSMs, that generalize lexicographic ranking functions for termination proving in non-probabilistic programs. The goal of this section is to advance the state of the art of lexicographic proof rules for qualitative termination analysis in PPs.

Lexicographic proof rules for non-probabilistic programs. For non-probabilistic programs, the *termination* problem asks whether a given program *always* terminates. While the problem is well-known to be undecidable over Turing-complete programs, many sound automated techniques that work well for practical programs have been developed [CPR06, CPR11]. Such techniques typically seek a suitable *certificate* of termination. Particularly relevant certificate is a ranking function (RF) [Flo67, BMS05a, CS01, SG91, PR04a, PR04b] mapping program states into a well-founded domain, forcing a strict decrease of the function value in every step. The basic ranking functions are 1-dimensional, which is often insufficient for complex control-flow structures and does not allow for compositional reasoning. A lexicographic ranking function (LexRF) is multi-dimensional extension of RFs that provide an effective approach to termination analysis [CSZ13, ADFG10, GMR15, BCI$^+$16, BMS05a, BCF13]. The literature typically restricts to linear LexRFs for linear-arithmetic (LA) programs, as LA reasoning can be more efficiently automated compared to non-linear arithmetic.

Lexicographic proof rules for PPs. For PPs, the *almost-sure (a.s.)* termination problem asks whether a given PP terminates with probability 1. One way of proving a.s. termination is via *ranking supermartingales (RSMs)*, a probabilistic analogue of ranking functions named so due to the connection with (super)martingale stochastic processes [Wil91]. There is a rich body of work on 1-dimensional RSMs, while the work [ACN18] introduces lexicographic RSMs. In PPs, a transition τ available in some state s yields a probability distribution over the successor states. The conditions defining RSMs are formulated in terms of the expectation operator $\mathbb{E}^\tau$ of this distribution. In particular, *lexicographic ranking supermartingales* (LexRSMs) of [ACN18] are functions f mapping program states to $\mathbb{R}^d$, such that for each transition τ there exists a component $1 \leq i \leq d$, satisfying, for any reachable state s at which τ is enabled, the following conditions *P-RANK* and *S-NNEG* (with f_i the i-component of f and $s \models G(\tau)$ denoting the fact that s satisfies the guard of τ):

1. *P-RANK*$(f,\tau) \equiv s \models G(\tau) \Rightarrow \Big(\mathbb{E}^\tau[f_i(s')] \leq f_i(s) - 1$ and $\mathbb{E}^\tau[f_j(s')] \leq f_j(s)$ for all $1 \leq j < i\Big)$.
2. *S-NNEG*$(f,\tau) \equiv s \models G(\tau) \Rightarrow \Big(f_j(s) \geq 0$ for all $1 \leq j \leq d\Big)$.

(We use the standard primed notation from program analysis, i.e. s' is the probabilistically chosen successor of s when performing τ.) The *P-RANK* condition enforces an expected

```
ℓ0 : while y ≥ 0 do
       x := y;
ℓ1 :   while x ≥ 0 do
         x := x − 1 + Norm(0, 1)
       od;
       y := y − 1
     od
ℓout :
```

(a)

```
ℓ0 : while x ≥ 0 do
     if y ≥ 0 then
       y := y + Uniform[−7, 1]
     else
       x := x + Uniform[−7, 1];
ℓ1 :   y := y + Uniform[−7, 1]
     fi od
ℓout :
```

(b)

Figure 3.1: Motivating examples. $Norm(\mu, \sigma)$ samples from the normal distribution with mean μ and std. deviation σ. $Uniform[a, b]$ samples uniformly from the interval $[a, b]$. Location labels are the "ℓ_i": one location per loop head and one additional location in (b) so as to have one assignment per transition (a technical requirement for our approach). In addition, the location label "ℓ_{out}" denotes the terminal location.

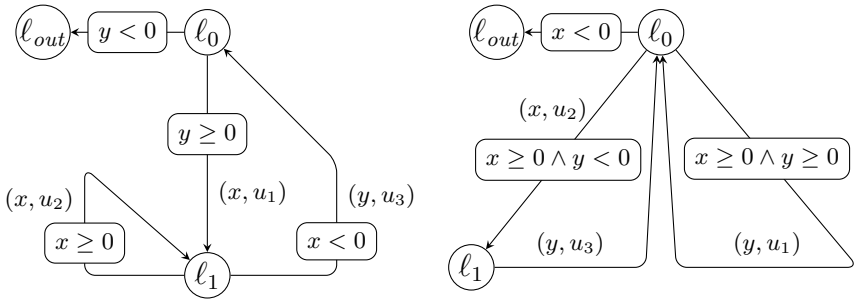

Figure 3.2: The pCFGs of the programs presented in Figure 3.1 (see Section 2.2 for a formal definition of pCFGs). Guards are shown in the rounded boxes, (absence of a box means that the guard is $true$). The update tuples are shown using variable aliases instead of indexes for better readability. On the left, we have $u_1 = y, u_2 = x - 1 + Norm(0, 1)$, and $u_3 = y - 1$. On the right, we have $u_1 = y + Uniform[-7, 1], u_2 = x + Uniform[-7, 1]$, and $u_3 = y + Uniform[-7, 1]$

decrease in lexicographic ordering, while *S-NNEG* stands for "strong non-negativity". Proving the soundness of LexRSMs for proving a.s. termination is highly non-trivial and requires reasoning about complex stochastic processes [ACN18]. Apart from soundness, [ACN18] also discusses compositionality aspects of LexRSMs in PPs and presents an algorithm for the synthesis of linear LexRSMs in linear PPs.

Limitations of LexRSMs. While LexRSMs improved the applicability of a.s. termination proving, their usage is impeded by the *restrictiveness of strong non-negativity* due to which a linear LexRSM might not exist even for simple a.s. terminating programs. This is a serious drawback from the automation perspective, since even if such a program admits a non-linear LexRSM, efficient automated tools that restrict to linear-arithmetic reasoning would not be able to find it.
Consider the program in Figure 3.1a. By employing simple random-walk arguments, we can manually prove that the program terminates a.s. A linear LexRSM proving this needs to have a component containing a positive multiple of x at the head of the inner while-loop (ℓ_1). However, due to the sampling from the normal distribution, which has unbounded support, the value of x inside the inner loop cannot be bounded from below. Hence, the program does not admit a linear LexRSM. In general, the existing 1-dimensional variants of ranking supermartingales [CS13, HFCG19] as well as LexRSMs with strong non-negativity do not handle well programs with unbounded-support distributions, as they all require their components to be nonnegative in all reachable states at which some transition is enabled. The strong nonnegativity condition is too restrictive for automated methods that reason over linear arithmetic, and existing methods that reason over linear arithmetic cannot prove even that the inner loop of the program in Figure 3.1a (also shown in Fig. 3.3) terminates a.s.
Now consider the program in Figure 3.1b. It can be again shown that this PP terminates a.s.; however, this cannot be witnessed by a linear LexRSM: to rank the "if-branch" transition, there must be a component with a positive multiple of y in ℓ_0. But y can become arbitrarily negative within the else branch, and cannot be bounded from below by a linear function of x.

Contributions. The contributions of this chapter are as follows:

1. *Generalized Lexicographic RSMs.* In the non-probabilistic setting, strong non-negativity can be relaxed to *partial non-negativity* (*P-NNEG*), where only the components which are to the left of the "ranking component" i (inclusive) need to be non-negative (Ben-Amram–Genaim RFs [BAG15]). We show that in the probabilistic setting, the same relaxation is possible under additional *expected leftward non-negativity* constraint *EXP-NNEG*. Formally, we say that f is a *generalized lexicographic ranking supermartingale* (GLexRSM) if for any transition τ there is $1 \leq i \leq d$ such that for any reachable state s at which τ is enabled we have *P-RANK*$(f,\tau) \wedge$ *P-NNEG*$(f,\tau) \wedge$ *EXP-NNEG*(f,τ), where

$$\begin{aligned} \textit{P-NNEG}(f,\tau) \quad &\equiv \quad s \models G(\tau) \Rightarrow \Big(f_j(s) \geq 0 \text{ for all } 1 \leq j \leq i\Big) \\ \textit{EXP-NNEG}(f,\tau) \quad &\equiv \quad s \models G(\tau) \Rightarrow \Big(\mathbb{E}^\tau[f_j(s') \cdot \mathbb{I}_{<j}(s')] \geq 0 \text{ for all } 1 \leq j \leq i\Big), \end{aligned}$$

with $\mathbb{I}_{<j}$ the indicator function of the set of all states in which a transition ranked by a component $< j$ is enabled. We first formulate GLexRSMs as an abstract proof rule for general stochastic processes. We then instantiate them into the setting of PPs and define *GLexRSM maps,* which we prove to be sound for proving a.s. termination. These results are general and *not specific* to linear-arithmetic PPs.

2. *Polynomial Algorithms for Linear GLexRSMs.* We present two algorithms:

   a) For linear arithmetic PPs in which sampling instructions use bounded-support distributions and non-deterministic variable update instructions use bounded-support intervals, we show that the problem $\textsc{LinGLexPP}$ of deciding whether a given PP with a given set of *linear invariants* admits a linear GLexRSM is decidable in polynomial time. Also, our algorithm computes the witnessing linear GLexRSM whenever it exists. In particular, our approach proves the a.s. termination of the program in Fig. 3.1b.

   b) Building on results of item 1, we construct a sound polynomial-time algorithm for a.s. termination proving in PPs that *do perform* sampling from *unbounded-support* distributions. In particular, the algorithm proves a.s. termination for our motivating example in Fig. 3.1a.

3. *First linear-arithmetic martingale-based method for unbounded-support distributions.* Finally, while the focus of our work is on relaxing the restrictive strong non-negativity assumption of LexRSMs, we remark that our theoretical and algorithmic results also yield the first automated method that operates over *linear arithmetic* and can prove a.s. termination in PPs in which termination depends on sampling instructions from *double-sided unbounded support* probability distributions. For instance, termination behavior of the simple loop in Fig. 3.3 is determined by values sampled from the standard normal distribution. Any linear arithmetic martingale-based certificate admitted by this program would need to have a vanishing linear coefficient of the sampled variable for the certificate to be non-negative, hence this program does not admit any martingale-based certificate that imposes the strong non-negativity condition. To the best of our knowledge, the only exception are descent supermartingale maps (DSMs) of [HFCG19] which replace the strong non-negativity assumption by the bounded difference condition that requires bounded maximal change in value at every program state. However, the program in Fig. 3.3 does not admit a linear arithmetic DSM either, as the sampled value from the standard normal distribution is unbounded therefore the linear coefficient of the sampled variable in the DSM would need to be 0. Since our method relaxes the strong non-negativity assumption while not introducing the bounded difference condition, it presents the first method that operates over linear arithmetic and can prove a.s. termination of the PP presented in Fig. 3.3.

ℓ_0 : **while** $x \geq 0$ **do**
$x := x - 1 + Norm(0, 1)$
od
ℓ_{out} :

Figure 3.3: A simple loop that terminates a.s. and involves sampling from the standard normal distribution which has unbounded support. However, no existing martingale-based method that reasons over linear arithmetic can prove its a.s. termination.

Chapter organization. This chapter is split in two parts: the first one is "abstract", with the definition and soundness proof of abstract GLexRSMs (Section 3.2). We also present an example showing that "GLexRSMs" without the expected leftward non-negativity constraint are not sound. The second part covers application to PPs: a GLexRSM-based proof rule for a.s. termination (Section 3.3) and our algorithms for a.s. termination analysis in PPs (Section 3.4). We overview related work in Section 3.5. Finally, Section 3.6 contains full proofs of results presented in earlier sections that are deferred to this section in order to enhance readability.

3.2 Generalized Lexicographic Ranking Supermartingales

In this section, we introduce the notion of *generalized lexicographic ranking supermartingales (GLexRSMs)* – an abstract concept that is not necessarily connected to PPs, but which is crucial for the soundness of our new proof rule for a.s. termination. The following section heavily relies on preliminary background in probability and martingale theory, see Sections 2.1 and 2.4.

Defintion 3.2.1 (Generalized Lexicographic Ranking Supermartingale)**.** *Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space and let $(\mathcal{F}_t)_{t=0}^{\infty}$ be a filtration of $\mathcal{F}$. Suppose that T is a stopping time w.r.t. $\mathcal{F}$. An n-dimensional real valued stochastic process $(\mathbf{X}_t)_{t=0}^{\infty}$ is a* generalized lexicographic ranking supermartingale for T *(GLexRSM) if:*

1. *For each $t \in \mathbb{N}_0$ and $1 \leq j \leq n$, the random variable $\mathbf{X}_t[j]$ is $\mathcal{F}_t$-measurable.*
2. *For each $t \in \mathbb{N}_0$, $1 \leq j \leq n$, and $A \in \mathcal{F}_{t+1}$, the conditional expectation $\mathbb{E}[\mathbf{X}_{t+1}[j] \cdot \mathbb{I}(A) \mid \mathcal{F}_t]$ exists.*
3. *For each $t \in \mathbb{N}_0$, there exists a partition of the set $\{T > t\}$ into n subsets $L_1^t, \ldots, L_n^t$, all of them $\mathcal{F}_t$-measurable (i.e., belonging to $\mathcal{F}_t$), such that for each $1 \leq j \leq n$*

- $\mathbb{E}[\mathbf{X}_{t+1}[j] \mid \mathcal{F}_t](\omega) \leq \mathbf{X}_t[j](\omega)$ *for each* $\omega \in \cup_{j'=j}^{n} L_{j'}^t$,
- $\mathbb{E}[\mathbf{X}_{t+1}[j] \mid \mathcal{F}_t](\omega) \leq \mathbf{X}_t[j](\omega) - 1$ *for each* $\omega \in L_j^t$,
- $\mathbf{X}_t[j](\omega) \geq 0$ *for each* $\omega \in \cup_{j'=j}^{n} L_{j'}^t$,
- $\mathbb{E}[\mathbf{X}_{t+1}[j] \cdot \mathbb{I}(\cup_{j'=0}^{j-1} L_{j'}^{t+1}) \mid \mathcal{F}_t](\omega) \geq 0$ *for each* $\omega \in \cup_{j'=j}^{n} L_{j'}^t$, *with* $L_0^{t+1} = \{T \leq t+1\}$.

Intuitively, we may think of each $\omega \in \Omega$ as a trajectory of process that evolves over time (in the second part of the chapter, this will be a PP run). Then, $\mathbf{X}_t$ is a vector function depending on the first t time steps (each $\mathbf{X}_t[j]$ is $\mathcal{F}_t$-measurable), while T is the time at which the trajectory is stopped. Then in point 3 of the definition, the first two items encode the expected (conditional) lexicographic decrease of $\mathbf{X}_t$, the third item encodes non-negativity of components to the left (inclusive) of the one which "ranks" ω in step t, and the last item encodes the expected leftward non-negativity (sketched in Section 1). For each $1 \leq j \leq n$ and time step $t \geq 0$, the set L_j^t contains all $\omega \in \{T > t\}$ which are "ranked" by the component j at time t. An *instance* of an n-dimensional GLexRSM $\{\mathbf{X}_t\}_{t=0}^{\infty}$ is a tuple $(\mathbf{X}_{t=0}^{\infty}, \{L_1^t, \ldots, L_n^t\}_{t=0}^{\infty})$, where the second component is a sequence of partitions of Ω satisfying the condition in Definition 3.2.1. We say that $\omega \in \Omega$ has *level* j in step t of the instance $((\mathbf{X}_t)_{t=0}^{\infty}, (L_1^t, \ldots, L_n^t)_{t=0}^{\infty})$ if $T(\omega) > t$ and $\omega \in L_j^t$. If $T(\omega) \leq t$, we say that the level of ω at step t is 0.

We now state the main theorem of this section, which underlies the soundness of our new method for proving almost-sure termination.

Theorem 3.2.2. *Let* $(\Omega, \mathcal{F}, \mathbb{P})$ *be a probability space,* $(\mathcal{F}_t)_{t=0}^{\infty}$ *a filtration of* $\mathcal{F}$ *and* T *a stopping time w.r.t.* $\mathcal{F}$. *If there is an instance* $((\mathbf{X}_t)_{t=0}^{\infty}, (L_1^t, \ldots, L_n^t)_{t=0}^{\infty})$ *of a GLexRSM over* $(\Omega, \mathcal{F}, \mathbb{P})$ *for* T, *then* $\mathbb{P}[T < \infty] = 1$.

In [ACN18], a mathematical notion of LexRSMs is defined and a result for LexRSMs analogous to our Theorem 3.2.2 is established. Thus, the first part of our proof mostly resembles the proof of Theorem 3.3. in [ACN18], up to the point of defining the stochastic process $(Y_t)_{t=0}^{\infty}$ in eq. (3.1). After that, the proof of [ACN18] crucially relies on nonnegativity of each $\mathbf{X}_t[j]$ and Y_t at every $\omega \in \Omega$ that is guaranteed by LexRSMs, and it cannot be adapted to the case of GLexRSMs. Below we first show that, for GLexRSMs, $\mathbb{E}[Y_t] \geq 0$ for each $t \geq 0$, and then we present a very elegant argument via the Borel-Cantelli lemma [Wil91, Theorem 2.7] which shows that this boundedness of expectation is sufficient for the theorem claim to hold.

Proof of Theorem 3.2.2. We proceed by contradiction. Suppose that there exists an instance of a GLexRSM but that $\mathbb{P}[T = \infty] > 0$. First, we claim that there exists $1 \leq k \leq n$ and $s, M \in \mathbb{N}_0$ such that the set B of all $\omega \in \Omega$ for which the following

properties hold has positive measure, i.e. $\mathbb{P}[B] > 0$: (1) $T(\omega) = \infty$, (2) $\mathbf{X}_s[k](\omega) \leq M$, (3) for each $t \geq s$, the level of ω at step t is at least k, and (4) the level of ω equals k infinitely many times.

The claim is proved by several applications of the union bound. For each $\omega \in \Omega$, we define minlev(ω) to be the smallest $0 \leq j \leq n$ such that the level of ω is equal to j in infinitely many steps i. Let $B_k = \{\omega \in \Omega \mid T(\omega) = \infty \wedge \mathit{min\text{-}lev}(\omega) = k\}$ for each $1 \leq k \leq n$. Then

$$\{\omega \in \Omega \mid T(\omega) = \infty\} = \cup_{k=1}^{n} B_k,$$

and thus, by the union bound, there exists $1 \leq k \leq n$ for which $\mathbb{P}[B_k] > 0$. We may express B_k as a union of events over the time of the last visit to some L_j^i with $j < k$. If we write $B_k^s = \{\omega \in \Omega \mid T(\omega) = \infty \wedge (i \geq s \Rightarrow \omega \in \cup_{j=k}^{n} L_j^i\}$ for each $s \in \mathbb{N}_0$, we have that $B_k = \cup_{s \geq 0} B_k^s$. As by the union bound $\mathbb{P}[B_k] \leq \sum_{s=0}^{\infty} \mathbb{P}[B_k^s]$, there exists $s \in \mathbb{N}_0$ for which $\mathbb{P}[B_k^s] > 0$. Now, for each $M \in \mathbb{N}_0$, let $B_k^{s,M}$ be defined via

$$B_k^{s,M} = \{\omega \in B_k^s \mid \mathbf{X}_s[k] \leq M\}.$$

Then $B_k^s = \cup_{M=0}^{\infty} B_k^{s,M}$. By the union bound we have $\mathbb{P}[B_k^s] \leq \sum_{M=0}^{\infty} \mathbb{P}[B_k^{s,M}]$, and there exists $M \in \mathbb{N}_0$ such that $\mathbb{P}[B_k^{s,M}] > 0$. The set $B = B_k^{s,M}$ satisfies the conditions of the claim. Since B is defined in terms of tail properties of ω ("level is at least k *infinitely many times*") it is not necessarily $\mathcal{F}_t$-measurable for any t. Hence, we define a stochastic process $(Y_t)_{t=0}^{\infty}$ such that each Y_t is $\mathcal{F}_t$-measurable, and which satisfies the desirable properties of $(\mathbf{X}_t[k])_{t=0}^{\infty}$ on B.

Let $D = \{\omega \in \Omega \mid \mathbf{X}_s[k](\omega) \leq M \wedge \omega \in \cup_{j=k}^{n} L_j^s\}$. Note that D is $\mathcal{F}_t$-measurable for $t \geq s$. We define a stopping time F w.r.t. $(\mathcal{F}_t)_{t=0}^{\infty}$ via $F(\omega) = \inf\{t \geq s \mid \omega \notin \cup_{j'=k}^{n} L_{j'}^t\}$; then a stochastic process $(Y_t)_{t=0}^{\infty}$ via

$$Y_t(\omega) = \begin{cases} 0, & \text{if } \omega \notin D, \\ M, & \text{if } \omega \in D, \text{ and } t < s, \\ \mathbf{X}_t[k](\omega), & \text{if } \omega \in D,\ t \geq s \text{ and } F(\omega) > t, \\ \mathbf{X}_{F(\omega)}[k](\omega), & \text{else.} \end{cases} \tag{3.1}$$

A straightforward argument (presented in Section 3.6.1) shows that for each $t \geq s$ we have $\mathbb{E}[Y_{t+1}] \leq \mathbb{E}[Y_t] - \mathbb{P}[L_k^t \cap D \cap \{F > t\}]$. By a simple induction we obtain:

$$\mathbb{E}[Y_s] \geq \mathbb{E}[Y_t] + \sum_{r=s}^{t-1} \mathbb{P}[L_k^r \cap D \cap \{F > r\}]. \tag{3.2}$$

Now, we show that $\mathbb{E}[Y_t] \geq 0$ for each $t \in \mathbb{N}_0$. The claim is clearly true for $t < s$, so suppose that $t \geq s$. We can then expand $\mathbb{E}[Y_t]$ as follows

$$\begin{aligned}
\mathbb{E}[Y_t] &= \mathbb{E}[Y_t \cdot \mathbb{I}(F = s)] + \sum_{r=s+1}^{t} \mathbb{E}[Y_t \cdot \mathbb{I}(F = r)] + \mathbb{E}[Y_t \cdot \mathbb{I}(F > t)] \\
&\quad (Y_s \geq 0 \text{ as } D \subseteq \cup_{j=k}^{n} L_j^s \text{ and } Y_t(\omega) \geq 0 \text{ whenever } F(\omega) > t) \\
&\geq \sum_{r=s+1}^{t} \mathbb{E}[Y_t \cdot \mathbb{I}(F = r)] = \sum_{r=s+1}^{t} \mathbb{E}[Y_t \cdot \mathbb{I}(\{F = r\} \cap D)] \\
&\quad (Y_t(\omega) = \mathbf{X}_{F(\omega)}[k](\omega) \text{ whenever } \omega \in D,\ t \geq s \text{ and } F(\omega) \leq t) \\
&= \sum_{r=s+1}^{t} \mathbb{E}[\mathbf{X}_r[k] \cdot \mathbb{I}(\cup_{j=0}^{k-1} L_j^r) \cdot \mathbb{I}(\{F > r-1\} \cap D)] \\
&\quad (\text{properties of cond. exp. \& } \mathbb{I}(\{F > r-1\} \cap D) \text{ is } \mathcal{F}_{r-1}\text{-measurable}) \\
&= \sum_{r=s+1}^{t} \mathbb{E}\Big[\mathbb{E}[\mathbf{X}_r[k] \cdot \mathbb{I}(\cup_{j=0}^{k-1} L_j^r) \mid \mathcal{F}_{r-1}] \cdot \mathbb{I}(\{F > r-1\} \cap D)\Big] \geq 0 \\
&\quad (\mathbb{E}[\mathbf{X}_r[k] \cdot \mathbb{I}(\cup_{j=0}^{k-1} L_j^r) \mid \mathcal{F}_{r-1}](\omega) \geq 0 \text{ for } \omega \in \{F > r-1\} \subseteq \cup_{j=k}^{n} L_j^{r-1}).
\end{aligned}$$

Plugging into eq. (3.2) that $\mathbb{E}[Y_t] \geq 0$, we get $\mathbb{E}[Y_s] \geq \sum_{r=s}^{t-1} \mathbb{P}[L_k^r \cap D \cap \{F > r\}]$ for each $t \geq s$. By letting $t \to \infty$, we conclude $\mathbb{E}[Y_s] \geq \sum_{r=s}^{\infty} \mathbb{P}[L_k^r \cap D \cap \{F > r\}]$. As $Y_s \leq M$ and $Y_s = 0$ outside D, we know that $\mathbb{E}[Y_s] \leq M \cdot \mathbb{P}[D]$. We get

$$\sum_{r=s}^{\infty} \mathbb{P}[L_k^r \cap D \cap \{F = \infty\}] \leq \sum_{r=s}^{\infty} \mathbb{P}[L_k^r \cap D \cap \{F > r\}] \leq M \cdot \mathbb{P}[D] < \infty.$$

By the Borel-Cantelli lemma, $\mathbb{P}[L_k^r \cap D \cap \{F = \infty\} \text{ for infinitely many } r] = 0$. But the event $\{L_k^r \cap D \cap \{F = \infty\} \text{ for infinitely many } r\}$ is precisely the set of all runs $\omega \in \Omega$ for which (1) $T(\omega) = \infty$ (as ω never has level zero by $\omega \in L_k^r$ for inf. many k), (2) $\mathbf{X}_s[k](\omega) \leq M$, (3) for each $r \geq s$ the level of ω at step t is at least k, and (4) the level of ω is k infinitely many times. Hence, $B = \{L_k^r \cap D \cap \{F = \infty\} \text{ for infinitely many } r\}$ and $\mathbb{P}[B] = 0$, a contradiction. □

GLexRSMs would be unsound without the expected leftward nonnegativity.

Example 3.2.1. *Consider a one-dimensional stochastic process $(Y_t)_{t=0}^{\infty}$ s.t. $Y_0 = 1$ with probability 1 and then the process evolves as follows: in every step t, if $Y_t \geq 0$, then with probability $p_t = \frac{1}{4} \cdot \frac{1}{2^t}$ we put $Y_{t+1} = Y_t - \frac{2}{p_t}$ and with probability $1 - p_t$ we put $Y_{t+1} = Y_t + \frac{1}{1-p_t}$. If $Y_t < 0$, we put $Y_{t+1} = Y_t$. The underlying probability space can be constructed by standard techniques and we consider the filtration $(\mathcal{F}_t)_{t=0}^{\infty}$ s.t. $\mathcal{F}_t$ is the smallest sub-sigma-algebra making Y_t measurable. Finally, consider the stopping time T returning the first point in time when $Y_t < 0$. Then $T < \infty$ if and*

only if the process ever performs the update $Y_{t+1} = Y_t - \frac{2}{p_t}$, but the probability that this happens is bounded by $\frac{1}{4} + \frac{3}{4} \cdot \frac{1}{8} + \frac{3}{4} \cdot \frac{7}{8} \cdot \frac{1}{16} + \cdots < \frac{1}{4} \sum_{t=0}^{\infty} \frac{1}{2^t} = \frac{1}{2} < 1$. At the same time, putting $L_1^t = \{Y_t \geq 0\}$ we get that the tuple $((Y_t)_{t=0}^{\infty}, (L_1^t)_{t=0}^{\infty})$ satisfies all conditions of Definition 3.2.1 apart from the last bullet of point 3.

3.3 GLexRSMs for Probabilistic Programs

We now define a syntactic proof rule for a.s. termination of PPs based on GLexRSMs, showing its soundness via Theorem 3.2.2. To formally define our new proof rule, we first need to define the notions of measurable maps, generalized transitions and pre-expectation. In what follows, let $\mathcal{C} = (L, V, \ell_{init}, \mathbf{x}_{init}, \mapsto, G, Pr, Up)$ be a pCFG.

Defintion 3.3.1 (Measurable map)**.** *An n-dimensional measurable map (MM) is a vector $\boldsymbol{\eta} = (\eta_1, \ldots, \eta_n)$, where each η_i is a function mapping each location ℓ to a real-valued Borel-measurable function $\eta_i(\ell)$ over program variables. We say that $\boldsymbol{\eta}$ is a linear expression map (LEM) if $\eta_i(\ell)$ is representable by a linear expression over program variables for each $1 \leq i \leq n$ and for each location ℓ in $\mathcal{C}$.*

Next, we define generalized transitions in pCFGs. Intuitively, these are identical to transitions in pCFGs as defined in Section 2.2, except that all transitions outgoing from some probabilistic branching location are merged into one generalized transition. Thinking about generalized transitions will simplify reasoning about probability distributions of successor states upon executing one step in the pCFG and will allow simultaneous ranking of all transitions outgoing from a probabilistic branching location.

Defintion 3.3.2 (Generalized transition)**.** *Consider a pCFG $\mathcal{C} = (L, V, \ell_{init}, \mathbf{x}_{init}, \mapsto, G, Pr, Up)$ and the partition $L = L_C \cup L_N \cup L_A \cup L_P$ of its locations as in Section 2.2. A generalized transition is defined as either*

- *a transition $\tau = (\ell, \ell') \in \mapsto$ with $\ell \in L_C \cup L_N \cup L_A$, i.e. τ is induced by a transition whose source location is not a location of probabilistic branching, or*
- *a tuple $\tau = (\ell, \delta)$, where $\ell \in L_P$ and δ is a probability distribution over locations in $\mathcal{C}$ with $\delta(\ell') = Pr((\ell, \ell'))$ for each transition $(\ell, \ell') \in \mapsto$ outgoing from ℓ, i.e. τ is induced by a probabilistic branching in $\mathcal{C}$.*

We use Δ_{NPB} to denote the set of all generalized transitions that are not induced by a probabilistic branching (defined in the first item above), Δ_{PB} to denote the set of all generalized transitions that are induced by a probabilistic branching (defined in the second item above), and $\Delta = \Delta_{NPB} \cup \Delta_{PB}$ to denote the set of all generalized

transitions. With a slight abuse of notation, we identify each generalized transition in Δ_{NPB} *with the transition in* $\mathcal{C}$ *that induces it.*

In both cases above, we say that ℓ is the *source location* of τ. For a generalized transition τ, we define its *guard* $G(\tau)$ via $G(\tau) = G((\ell, \ell'))$ if $\tau = (\ell, \ell')$ with $\ell \in L_C$ being a location of conditional branching, and $G(\tau) =$ true otherwise.

The notion of pre-expectation was introduced in [Koz85], was made syntactic in the Dijkstra wp-style in [MM99], and was extended to programs with continuous distributions in [CS13]. It formalizes the "one-step" expectation operator $\mathbb{E}^\tau$ we used on an intuitive level in the introduction. In what follows, we generalize the definition of pre-expectation presented in [CS13] in order to allow taking expectation over subsets of successor states in $\mathcal{C}$ (a necessity for handling the *EXP-NNEG* constraint). We say that a set S of states in $\mathcal{C}$ is *measurable*, if for each location ℓ in $\mathcal{C}$ we have that $\{\mathbf{x} \in \mathbb{R}^{|V|} \mid (\ell, \mathbf{x}) \in S\} \in \mathcal{B}(\mathbb{R}^{|V|})$, i.e. it is in the Borel sigma-algebra of $\mathbb{R}^{|V|}$. Furthermore, we also differentiate between the *maximal* and *minimal pre-expectation*, which may differ in the case of non-determinism in programs and intuitively are equal to the maximal resp. minimal value of the next-step expectation over all non-deterministic choices. Let η be a 1-dimensional MM, τ a generalized transition and S be a measurable set of states in $\mathcal{C}$. We denote by max-pre$^\tau_{\eta,S}(s)$ the *maximal pre-expectation* of η in τ given S (i.e. the maximal expected value of η after making a step from s computed over successor states belonging to S), and similarly we denote by min-pre$^\tau_{\eta,S}$ the *minimal pre-expectation* of η in τ given S. In what follows, we use $\mathbb{I}(S)$ to denote a function which to each state $(\ell, \mathbf{x})$ assigns 1 if $(\ell, \mathbf{x}) \in S$ and 0 otherwise.

Defintion 3.3.3 (Pre-expectation)**.** *Let* η *be a 1-dimensional MM,* τ *be a generalized transition with source location* ℓ *and* S *be a measurable set of states in* $\mathcal{C}$*. A* maximal pre-expectation *of* η *in* τ *given* S *is the function max-pre*$^\tau_{\eta,S}$ *assigning to each state* $(\ell, \mathbf{x})$ *the following value:*

1. *If* $\tau = (\ell, \delta)$ *with* $\ell \in L_P$*, then*

$$\textit{max-pre}^\tau_{\eta,S}(\ell, \mathbf{x}) = \sum_{\ell' \in L} \delta(\ell') \cdot \eta(\ell', \mathbf{x}) \cdot \mathbb{I}(S)(\ell', \mathbf{x}).$$

2. *If* $\tau = (\ell, \ell')$ *with* $\ell \in L_C \cup L_N$ *then:*

$$\textit{max-pre}^\tau_{\eta,S}(\ell, \mathbf{x}) = \eta(\ell', \mathbf{x}) \cdot \mathbb{I}(S)(\ell', \mathbf{x}).$$

3. *If* $\tau = (\ell, \ell')$ *with* $\ell \in L_A$*, then:*
   - *If* $Up(\tau) = (j, \bot)$*, then max-pre*$^\tau_{\eta,S}(\ell, \mathbf{x}) = \eta(\ell', \mathbf{x}) \cdot \mathbb{I}(S)(\ell', \mathbf{x})$.

- *If* $Up(\tau) = (j, u)$ *with* $u : \mathbb{R}^{|V|} \to \mathbb{R}$ *an expression, then*

$$\textit{max-pre}^{\tau}_{\eta,S}(\ell, \mathbf{x}) = \eta(\ell', \mathbf{x}[x_j \leftarrow u(\mathbf{x}_i)]) \cdot \mathbb{I}(S)(\ell', \mathbf{x}[x_j \leftarrow u(\mathbf{x}_i)]).$$

- *If* $Up(\tau) = (j, u)$ *with* $u = d$ *a probability distribution, then*

$$\textit{max-pre}^{\tau}_{\eta,S}(\ell, \mathbf{x}) = \mathbb{E}_{X \sim d}\Big[\eta(\ell', \mathbf{x}[x_j \leftarrow X) \cdot \mathbb{I}(S)(\ell', \mathbf{x}[x_j \leftarrow X])\Big].$$

- *If* $Up(\tau) = (j, u)$ *with* $u = [a, b]$ *a non-deterministic interval, then*

$$\textit{max-pre}^{\tau}_{\eta,S}(\ell, \mathbf{x}) = \sup_{X \in [a,b]} \Big[\eta(\ell', \mathbf{x}[x_j \leftarrow X) \cdot \mathbb{I}(S)(\ell', \mathbf{x}[x_j \leftarrow X])\Big].$$

A minimal pre-expectation *is denoted by* $\textit{min-pre}^{\tau}_{\eta,S}$ *and is defined analogously, with the only difference being that in the last item we use* $\inf$ *instead of* $\sup$. *We omit the subscript* S *when* S *is the set of all states of* $\mathcal{C}$.

GLexRSM-Based Proof Rule for Almost-Sure Termination. We now present our GLexRSM-based proof rule for a.s. termination proving. Given $n \in \mathbb{N}$, we call a map $\mathsf{lev} : \Delta \to \{0, 1, \ldots, n\}$ a *level map*. For $\tau \in \Delta$ we say that $\mathsf{lev}(\tau)$ is its level. The level of a state is the largest level of any generalized transition enabled at that state. We denote by $S^{\leq j}_{\mathsf{lev}}$ the set of states with level $\leq j$.

As in the case of non-probabilistic programs, termination certificates are supported by program invariants over-approximating the set of reachable states. An *invariant* in $\mathcal{C}$ is a function I which to each location ℓ of $\mathcal{C}$ assigns a Borel-measurable set $I(\ell) \subseteq \mathbb{R}^{|V|}$ such that for any state $(\ell, \mathbf{x})$ reachable in $\mathcal{C}$ it holds that $\mathbf{x} \in I(\ell)$. If each $I(\ell)$ is given by a conjunction of linear inequalities over program variables, we say that I is a *linear invariant*.

Defintion 3.3.4 (GLexRSM Map)**.** *Let* $\boldsymbol{\eta}$ *be an* n*-dimensional MM and* I *an invariant in* $\mathcal{C}$. *We say that* $\boldsymbol{\eta}$ *is a* generalized lexicographic ranking supermartingale map (GLexRSM map) *supported by* I*, if there is a level map* $\mathsf{lev} : \Delta \to \{0, 1, \ldots, n\}$ *such that* $\mathsf{lev}(\tau) = 0$ *iff* τ *is a self-loop generalized transition at* ℓ_{out}*, and for any generalized transition* τ *with the source location* $\ell \neq \ell_{out}$ *the following conditions hold:*

1. *P-RANK*$(\boldsymbol{\eta}, \tau) \equiv \mathbf{x} \in I(\ell) \cap G(\tau) \Rightarrow \Big(\textit{max-pre}^{\tau}_{\eta_{\mathsf{lev}(\tau)}}(\ell, \mathbf{x}) \leq \eta_{\mathsf{lev}(\tau)}(\ell, \mathbf{x}) - 1 \wedge \textit{max-pre}^{\tau}_{\eta_j}(\ell, \mathbf{x}) \leq \eta_j(\ell, \mathbf{x})$ *for all* $1 \leq j < \mathsf{lev}(\tau)\Big)$;
2. *P-NNEG*$(\boldsymbol{\eta}, \tau) \equiv \mathbf{x} \in I(\ell) \cap G(\tau) \Rightarrow \Big(\eta_j(\ell, \mathbf{x}) \geq 0$ *for all* $1 \leq j \leq \mathsf{lev}(\tau)\Big)$;

3. *EXP-NNEG*$(\boldsymbol{\eta}, \tau) \equiv \mathbf{x} \in I(\ell) \cap G(\tau) \Rightarrow$ *min-pre*$^{\tau}_{\eta_j, S^{\leq j-1}_{\mathsf{lev}}}(\ell, \mathbf{x}) \geq 0$ *for all* $1 \leq j \leq \mathsf{lev}(\tau)$.

A GLexRSM map $\boldsymbol{\eta}$ *is* linear *(or LinGLexRSM map) if it is also an LEM.*

Theorem 3.3.5 (Soundness of GLexRSM-maps for a.s. termination)**.** *Let* $\mathcal{C}$ *be a pCFG and* I *an invariant in* $\mathcal{C}$*. Suppose that* $\mathcal{C}$ *admits an* n*-dimensional GLexRSM map* $\boldsymbol{\eta}$ *supported by* I*, for some* $n \in \mathbb{N}$*. Then* $\mathcal{C}$ *terminates a.s.*

Theorem 3.3.5 instantiates Theorem 3.2.2 to probability spaces of pCFGs and its proof can be found in Section 3.6.2. We stress that the instantiation is *not* straightforward. In particular, in proving it we establish two interesting results on PPs with non-determinism that were hitherto not studied in the PP analysis literature. In the rest of this section, we discuss these results in more detail. The proof of Theorem 3.3.5 uses these results to prove that a GLexRSM map induces a GLexRSM as in Definition 3.2.1 in the probability space over the set of runs defined by the pCFG.

First, to ensure that a scheduler cannot "escape" ranking by intricate probabilistic mixing of transitions, we show that in order to prove a.s. termination, it is sufficient to consider *deterministic* schedulers, which do not introduce randomization among transitions. This statement is formalized in the following proposition. In what follows, we use $\{TimeTerm = \infty\}$ to denote the set of all runs in $\mathcal{C}$ whose termination time is infinite, i.e. of all runs that do not terminate.

Proposition 3.3.6. *Let* $\mathcal{C}$ *be a pCFG and suppose that there exist a measurable scheduler* σ *and an initial state* $(\ell_{init}, \mathbf{x}_{init})$ *such that* $\mathbb{P}^{\sigma}_{\ell_{init}, \mathbf{x}_{init}}[TimeTerm = \infty] > 0$*. Then there exists a measurable scheduler* σ^**, which is deterministic in the sense that to each finite path it assigns a single transition with probability* 1*, such that* $\mathbb{P}^{\sigma^*}_{\ell_{init}, \mathbf{x}_{init}}[TimeTerm = \infty] > 0$.

Proof sketch, full proof in Section 3.6.2. The proof of Proposition 3.3.6 proceeds by constructing a sequence $\sigma = \sigma_0, \sigma_1, \sigma_2, \ldots$ of schedulers, where for each $i \in \mathbb{N}_0$ we have that σ_{i+1} and σ_i agree on histories of length at most $i - 1$, and σ_{i+1} "refines" σ_i on histories of length i in such a way that:

- σ_{i+1} is deterministic on histories of length at most i;
- σ_{i+1} is measurable;
- $\mathbb{P}^{\sigma_{i+1}}[TimeTerm = \infty] \geq \mathbb{P}^{\sigma_i}[TimeTerm = \infty]$.

Then we define the scheduler σ^* as $\sigma^*(\rho) = \sigma_i(\rho)$ whenever the length of a finite history ρ is i. The construction of each σ_{i+1} requires care as it needs to ensure that the newly constructed scheduler is measurable, that it satisfies $\mathbb{P}^{\sigma_{i+1}}[TimeTerm = \infty] \geq \mathbb{P}^{\sigma_i}[TimeTerm = \infty]$, and finally that the resulting scheduler σ^* is measurable. The fact that $\mathbb{P}^{\sigma^*}_{\ell_{init},\mathbf{x}_{init}}[TimeTerm = \infty] > 0$ then follows by the application of the Monotone Convergence Theorem [Wil91]. The formal construction is highly non-trivial and uses advanced results from probability theory. □

Second, previous martingale-based certificates of a.s. termination [FH15, CFNH18, FC19, ACN18] often impose either nonnegativity or integrability of random variables defined by measurable maps in programs to ensure that their conditional expectations exist. We show that these conditional expectations exist even without such assumptions and in the presence of nondeterminism, and that they can be explicitly expressed by extending the definition of pre-expectation in [CS13] to also depend on a fixed scheduler that is used to resolve non-determinism. This generalizes the result of [CS13] to PPs with nondeterminism.

Defintion 3.3.7 (Pre-expectation with respect to a scheduler)**.** *Let S be a set of measurable states in $\mathcal{C}$. The* pre-expectation with respect to a scheduler σ of a MM η *given S is a map $pre_{\sigma,\eta,S} : Fpath_{\mathcal{C}} \to \mathbb{R}$ defined as follows. Let $\rho \in Fpath_{\mathcal{C}}$ be a finite path ending in $(\ell, \mathbf{x})$. We define $pre_{\sigma,\eta,S}(\rho)$ as follows:*

1. *If $\ell \in L_N$ is a location of non-deterministic branching, denote by $\sigma(\rho)$ the probability distribution over (generalized) transitions enabled in $(\ell, \mathbf{x})$ defined by the scheduler σ. Then*

$$pre_{\sigma,\eta,S}(\rho) = \sum_{\tau \in supp(\sigma(\rho))} \sigma(\rho)(\tau) \cdot pre^{\tau}_{\sigma,\eta,S}(\ell, \mathbf{x}).$$

2. *Otherwise, let τ be the unique generalized transition with source location ℓ. Then*

$$pre_{\sigma,\eta,S}(\rho) = pre^{\tau}_{\sigma,\eta,S}(\ell, \mathbf{x}),$$

where $pre^{\tau}_{\sigma,\eta,S}$ is defined in the same way as the standard pre-expectation $pre^{\tau}_{\eta,S}$, except for the case *when τ is transition (ℓ, ℓ') whose update element is a non-deterministic assignment from an interval, i.e. $u(\tau) = (j, u)$ with u being an interval; in such a case, we put $pre^{\tau}_{\sigma,\eta,S}(\ell, \mathbf{x}) = \eta(\ell', \mathbf{x}(j \leftarrow \mathbb{E}[d_\sigma(\rho, \tau) \cdot \mathbb{I}(S)])$.*

Intuitively, $\text{pre}_{\sigma,\eta,S}$ takes a finite path $\rho \in Fpath_{\mathcal{C}}$ as an input, and returns the *expected value* of η in the next step when the integration is performed over program states in S, given the program run history and the choices of the scheduler σ. In the proof of Theorem 3.3.5 we show that, for any scheduler σ, all conditional expectations

whose existence is required in order to define a GLexRSM as in Definition 3.2.1 in the probability space over the set of runs defined by the pCFG $\mathcal{C}$ and the scheduler σ indeed exist and can be expressed via pre-expectations with respect to the scheduler σ.

Remark 3.3.1 (Comparison to [HFCG19])**.** *The work [HFCG19] considers a modular approach. Given a loop whose body has already been proved a.s. terminating, they show that the loop terminates a.s. if it admits a 1-dimensional MM satisfying P-RANK for each generalized transition in the loop, P-NNEG for the generalized transition entering the loop, and the "*bounded expected difference*" property for all transitions. Hence, their approach is suited mainly for PPs with incremental variable updates. Modularity is also a feature of the approaches based on the weakest pre-expectation calculus [MM04, MM05, MMKK18].*

3.4 Algorithms for Linear Probabilistic Programs

We now present two algorithms for proving a.s. termination in a linear probabilistic program (LinPP). The first algorithm considers LinPPs with sampling from bounded-support distributions, and we show that the problem of deciding the existence of LinGLexRSM maps for such LinPPs is decidable. Our second algorithm extends the first algorithm into a sound a.s. termination prover for general LinPPs. In what follows, let $\mathcal{C}$ be a LinPP and I a linear invariant in $\mathcal{C}$. For both algorithms, we assume that all non-deterministic variable updates in the LinPP use bounded-support intervals.

3.4.1 Linear Programs with Distributions of Bounded Support

Restricting to linear arithmetic is standard in automated a.s. termination proving, allowing to encode the existence of the termination certificate into systems of linear constraints [CS13, CFNH18, ACN18, CH20]. In the case of LinGLexRSM maps, the difficulty lies in encoding the *EXP-NNEG* condition, as it involves integrating distributions in variable updates which cannot always be done analytically. We show, however, that for LinPPs with bounded-support sampling, we can define another condition which is easier to encode and which can replace *EXP-NNEG*. Formally, we say that a distribution $d \in \mathcal{D}$ has a *bounded support*, if there exists $N(d) \geq 0$ such that $\mathbb{P}_{X\sim d}[|X| > N(d)] = 0$. Here, we use $\mathbb{P}_{X\sim d}$ to denote the probability measure induced by a random variable X with the probability distribution d. We say that a LinPP has the *bounded support property (BSP)* if all distributions in the program have bounded support. For instance, the program in Fig. 3.1b has the BSP, whereas the program in Fig. 3.1a does not. Using the same notation as in Definition 3.3.4, we put:

$$\textit{W-EXP-NNEG}(\boldsymbol{\eta}, \tau) \equiv \mathbf{x} \in I(\ell) \cap G(\tau) \Rightarrow \forall 1 \leq j \leq \mathsf{lev}(\tau)\ \mathsf{min\text{-}pre}^{\tau}_{\eta_j}(\ell, \mathbf{x}) \geq 0.$$

(The 'W' stands for "weak.") Intuitively, *EXP-NNEG* requires nonnegativity of the expected value of η_j when integrated over successor states of level smaller than j, whereas the condition *W-EXP-NNEG* requires nonnegativity of the expected value of η_j when integrated over all successor states. Since η_j is nonnegative at successor states of level at least j, this new condition is weaker than *EXP-NNEG*. Nevertheless, the following lemma shows that in order to decide existence of LinGLexRSM maps for programs with the BSP, we may w.l.o.g. replace *EXP-NNEG* by *W-EXP-NNEG* for all transitions but for those of probabilistic branching.

Lemma 3.4.1. *Let $\mathcal{C}$ be a LinPP with the BSP and I be a linear invariant in $\mathcal{C}$. If a LEM $\boldsymbol{\eta}$ satisfies conditions P-RANK and P-NNEG for all generalized transitions, EXP-NNEG for all generalized transitions in Δ_{PB} and W-EXP-NNEG for all other generalized transitions, then $\boldsymbol{\eta}$ may be increased pointwise by a constant value in order to obtain a LinGLexRSM map.*

Proof. To prove the lemma, we need to show that there exists $K > 0$ such that the LEM $\boldsymbol{\eta}'$ of the same dimension as $\boldsymbol{\eta}$, and defined via $\eta'_j(\ell, \mathbf{x}) = \eta_j(\ell, \mathbf{x}) + K$ for each component j and state $(\ell, \mathbf{x})$, is a LinGLexRSM map in $\mathcal{C}$ supported by I (with the level map being the same as for $\boldsymbol{\eta}$). Note that increasing $\boldsymbol{\eta}$ pointwise by a constant $K > 0$ preserves the *P-NNEG*, *P-RANK* conditions for each generalized transition in $\mathcal{C}$, as well as *EXP-NNEG* for each generalized transition of probabilistic branching. Hence, we are left to show that there exists $K > 0$ such that for any transition $\tau = (\ell, \ell')$ which is not a generalized transition of probabilistic branching we have

$$\textit{EXP-NNEG}(\boldsymbol{\eta}', \tau) \equiv \mathbf{x} \in I(\ell) \cap G(\tau) \Rightarrow \text{min-pre}^{\tau}_{\eta'_j, S^{\leq j-1}_{\text{lev}}}(\ell, \mathbf{x}) \geq 0 \text{ for all } 1 \leq j \leq \text{lev}(\tau).$$

Since the LinPP that induces $\mathcal{C}$ satisfies the BSP property and since all non-deterministic assignments are defined by closed intervals, there exists $N > 0$ such that $\mathbb{P}_{X \sim d}[|X| > N] = 0$ for each distribution $d \in \mathcal{D}$, and $[a, b] \subseteq [-N, N]$ for each interval $[a, b]$ appearing in non-deterministic assignments.

We claim that $K = 2 \cdot N \cdot \text{max-coeff}(\boldsymbol{\eta})$ satisfies the claim, where max-coeff$(\boldsymbol{\eta})$ is the maximal absolute value of a coefficient appearing in any expression $\boldsymbol{\eta}(\ell')$ for any location ℓ'. To prove this, let $\tau = (\ell, \ell_1)$ be a transition which is not a generalized transition of probabilistic branching. Let $\mathbf{x} \in I(\ell) \cap G(\tau)$ and let $1 \leq j \leq \text{lev}(\tau)$. In order to prove that $\text{min-pre}^{\tau}_{\eta'_j, S^{\leq j-1}_{\text{lev}}}(\ell, \mathbf{x}) \geq 0$ holds, we distinguish between three cases:

1. $Up(\tau) = \bot$ or $Up(\tau) = (i, u)$ where u is a linear expression. Then $(\ell, \mathbf{x})$ has a single successor state $(\ell_1, \mathbf{x}_1)$ upon executing τ.
   - If the level of $(\ell_1, \mathbf{x}_1)$ is at least j, then $S^{\leq j-1}_{\text{lev}}$ contains no successor states and we have that $\text{min-pre}^{\tau}_{\eta'_j, S^{\leq j-1}_{\text{lev}}}(\ell, \mathbf{x}) = 0$ as the integration is performed over the empty set.

- Otherwise, $S_{\mathsf{lev}}^{\leq j-1}$ contains $(\ell_1, \mathbf{x}_1)$ and

  $$\text{min-pre}^{\tau}_{\eta'_j, S_{\mathsf{lev}}^{\leq j-1}}(\ell, \mathbf{x}) = \eta'_j(\ell_1, \mathbf{x}_1) = \eta_j(\ell_1, \mathbf{x}_1) + K = \text{min-pre}^{\tau}_{\eta_j}(\ell, \mathbf{x}) + K \geq 0,$$

  where the inequality $\text{min-pre}^{\tau}_{\eta_j}(\ell, \mathbf{x}) \geq 0$ holds since *W-EXP-NNEG*$(\boldsymbol{\eta}, \tau)$.

2. If $Up(\tau) = (i, u)$ where $u = d \in \mathcal{D}$ is a probability distribution, let $X \sim d$ denote a random variable distributed according to d. Then, by linearity of η_j, we have

$$\begin{aligned}
&\text{min-pre}^{\tau}_{\eta'_j, S_{\mathsf{lev}}^{\leq j-1}}(\ell, \mathbf{x}) = \mathbb{E}_{X\sim d}\Big[\eta'_j(\ell_1, \mathbf{x}[i \leftarrow X]) \cdot \mathbb{I}(\text{next state has level } \leq j-1)\Big] \\
&= \mathbb{E}_{X\sim d}\Big[(\eta_j(\ell_1, \mathbf{x}[i \leftarrow 0]) + K + \text{coeff}[i] \cdot X) \cdot \mathbb{I}(\text{next state has level } \leq j-1)\Big] \\
&= \mathbb{E}_{X\sim d}\Big[(\eta_j(\ell_1, \mathbf{x}[i \leftarrow 0]) + \text{coeff}[i] \cdot \mathbb{E}[X]) \cdot \mathbb{I}(\text{next state has level } \leq j-1)\Big] \\
&\quad + \mathbb{E}_{X\sim d}\Big[(K - \text{coeff}[i] \cdot \mathbb{E}[X] + \text{coeff}[i] \cdot X) \cdot \mathbb{I}(\text{next state has level } \leq j-1)\Big] \\
&\geq \mathbb{E}_{X\sim d}\Big[\text{min-pre}^{\tau}_{\eta_j}(\ell, \mathbf{x}) \cdot \mathbb{I}(\text{next state has level } \leq j-1)\Big] \\
&\quad + \mathbb{E}_{X\sim d}\Big[(K - 2 \cdot N \cdot \text{max-coeff}(\boldsymbol{\eta})) \cdot \mathbb{I}(\text{next state has level } \leq j-1)\Big] \geq 0,
\end{aligned} \tag{3.3}$$

   where $\text{min-pre}^{\tau}_{\eta_j}(\ell, \mathbf{x}) \geq 0$ holds since *W-EXP-NNEG*$(\boldsymbol{\eta}, \tau)$, and $\mathbb{E}[X] \geq -N$ holds and $X \geq -N$ holds almost-surely since $\mathbb{P}_{X\sim d}[|X| > N] = 0$ by the definition of N. Here, coeff$[i]$ denotes the linear coefficient in η_j of the i-th variable in the variable valuation $\mathbf{x}$.

3. If $Up(\tau) = (i, u)$ where $u = [a, b]$ denotes a non-deterministic update, then

$$\begin{aligned}
\text{min-pre}^{\tau}_{\eta'_j, S_{\mathsf{lev}}^{\leq j-1}}(\ell, \mathbf{x}) &= \inf_{X \in [a,b] \wedge (\ell_1, \mathbf{x}(i \leftarrow X)) \in S_{\mathsf{lev}}^{\leq j-1}} \Big[\eta'_j(\ell_1, \mathbf{x}[i \leftarrow X])\Big] \\
&\geq \inf_{X \in [a,b]} \Big[\eta'_j(\ell_1, \mathbf{x}[i \leftarrow X])\Big] \\
&= \text{min-pre}^{\tau}_{\eta_j}(\ell, \mathbf{x}) + K \geq 0,
\end{aligned}$$

   where $\text{min-pre}^{\tau}_{\eta_j}(\ell, \mathbf{x}) \geq 0$ holds since *W-EXP-NNEG*$(\boldsymbol{\eta}, \tau)$ and $K \geq 0$.

□

Observe that the proof of Lemma 3.4.1 in item (2) essentially depends on the assumption that we work over linear arithmetic. Indeed, the second expected value in the first inequality in eq. (3.3) might not be possible to bounded from below by 0 for any $K > 0$ if we allowed non-linear arithmetic and program variable values that are not bounded.

Algorithm 3.1: Synthesis of LinGLexRSM maps in LinPPs with the BSP.

input : A LinPP $\mathcal{C}$ with the BSP property, linear invariant I.
output : LinGLexRSM map supported by I if it exists, otherwise "No LinGLexRSM map"

$\mathcal{T} \longleftarrow$ all generalized transitions in $\mathcal{C}$; $d \longleftarrow 0$
while $\mathcal{T}$ *is non-empty* **do**
  $d \longleftarrow d + 1$
  construct $\mathcal{LP}_{\mathcal{T}}$
  if $\mathcal{LP}_{\mathcal{T}}$ *is feasible* **then**
    $\eta_d \longleftarrow$ LEM defined by the optimal solution of $\mathcal{LP}_{\mathcal{T}}$
    $\mathcal{T} \longleftarrow \mathcal{T} \backslash \{\tau \in \mathcal{T} \mid \tau$ is 1-ranked by $\eta_d\}$
  end
  else return No LinGLexRSM map
end
$\max \longleftarrow$ max-coeff($\boldsymbol{\eta}$)
$N \longleftarrow$ constant such that all distributions and intervals supported in $[-N, N]$
for $1 \leq j \leq d$ **do**
  $\eta_j \longleftarrow \eta_j + 2 \cdot N \cdot \max$
end
return $(\eta_1, \ldots, \eta_d)$

Algorithmic results. Let $\textsc{LinGLexPP}^{\textsc{bounded}}$ be the set of pairs $(\mathcal{C}, I)$ of a pCFG $\mathcal{C}$ representing a LinPP with the BSP and all non-deterministic assignments having bounded support and a linear invariant I in $\mathcal{C}$, such that $\mathcal{C}$ admits a LinGLexRSM map supported by I.

Theorem 3.4.2. *There is a polynomial-time algorithm deciding if a tuple $(\mathcal{C}, I)$ belongs to the set $\textsc{LinGLexPP}^{\textsc{bounded}}$. Moreover, if the answer is yes, the algorithm outputs a witness in the form of a LinGLexRSM map of minimal dimension.*

The algorithm behind Theorem 3.4.2 is a generalization of algorithms in [ADFG10, ACN18] finding LinLexRFs in non-probabilistic programs and LinLexRSM maps in PPs, respectively. The pseudocode is shown in Algorithm 3.1. Suppose that we are given a LinPP $\mathcal{C} = (L, V, \ell_{init}, \mathbf{x}_{init}, \mapsto, G, Pr, Up)$ with the BSP and a linear invariant I. Our algorithm stores a set $\mathcal{T}$ initialized to all generalized transitions in $\mathcal{C}$. It then proceeds in iterations to compute new components of the witness. In each iteration it searches for a LEM η which is required to (ℓ denotes the source location of τ in each case)

1. be nonnegative on each $\tau \in \mathcal{T}$, i.e. $\forall \mathbf{x}.\, \mathbf{x} \in I(\ell) \cap G(\tau) \Rightarrow \eta(\ell, \mathbf{x}) \geq 0$;

2. be unaffecting on each $\tau \in \mathcal{T}$, i.e. $\forall \mathbf{x}.\, \mathbf{x} \in I(\ell) \cap G(\tau) \Rightarrow$ max-pre$^{\tau}_{\eta}(\ell, \mathbf{x}) \leq \eta(\ell, \mathbf{x})$;

3. have nonnegative minimal pre-expectation for each $\tau \in \mathcal{T} \setminus \Delta_{PB}$, i.e. $\forall \mathbf{x}.\, \mathbf{x} \in I(\ell) \cap G(\tau) \Rightarrow$ min-pre$^{\tau}_{\eta}(\ell, \mathbf{x}) \geq 0$;

4. if S is the set of states in $\mathcal{C}$ whose all enabled generalized transitions have been removed from $\mathcal{T}$ in the previous algorithm iterations, $\forall \tau \in \mathcal{T} \cap \Delta_{PB}$, $\forall \mathbf{x}.\, \mathbf{x} \in I(\ell) \cap G(\tau) \Rightarrow$ pre$^{\tau}_{\eta,S}(\ell, \mathbf{x}) \geq 0$; and

5. 1-rank the maximal number of generalized transitions in $\tau \in \mathcal{T}$, i.e. $\forall \mathbf{x}.\, \mathbf{x} \in I(\ell) \cap G(\tau) \Rightarrow$ max-pre$^{\tau}_{\eta}(\ell, \mathbf{x}) \leq \eta(\ell, \mathbf{x}) - 1$ for as many τ as possible.

This is done by fixing an LEM template for each location ℓ in $\mathcal{C}$, and converting the above constraints to an equivalent linear program $\mathcal{LP}_{\mathcal{T}}$ in template variables via Farkas' lemma (FL). The FL conversion (and its extension to strict inequalities [CFNH18]) is standard in termination proving and encoding conditions 1-3 and 5 above is analogous to [ADFG10, ACN18], hence we omit the details. To encode condition 4, let $\tau = (\ell, \delta) \in \mathcal{T}$ be a generalized transition of probabilistic branching. Then $supp(\delta) = (\ell_1, \ell_2)$ as each probabilistic branching in our PP syntax in Section 2.2 has two branches. Furthermore, there are no variable updates, so for any $\mathbf{x} \in I(\ell) \cap G(\tau)$ we have

$$\text{pre}^{\tau}_{\eta,S}(\ell, \mathbf{x}) = \delta(\ell_1) \cdot \eta(\ell_1, \mathbf{x}) \cdot \mathbb{I}(S)(\ell_1, \mathbf{x}) + \delta(\ell_2) \cdot \eta(\ell_2, \mathbf{x}) \cdot \mathbb{I}(S)(\ell_2, \mathbf{x}),$$

i.e. we include the term $\delta(\ell_i) \cdot \eta(\ell_i, \mathbf{x})$ for $i \in \{1, 2\}$ whenever $(\ell_i, \mathbf{x}) \in S$. Hence, to encode condition 4 for τ, define $G_1 = \neg(\vee_{\tau' = (\ell_1, \_) \in \mathcal{T}} G(\tau'))$ and $G_2 = \neg(\vee_{\tau' = (\ell_2, \_) \in \mathcal{T}} G(\tau'))$, and encode the following 3 conditions:

- $\forall \mathbf{x}.\, \mathbf{x} \in I(\ell) \cap G(\tau) \cap G_1 \cap G_2 \Rightarrow \delta(\ell_1) \cdot \eta(\ell_1, \mathbf{x}) + \delta(\ell_2) \cdot \eta(\ell_2, \mathbf{x}) \geq 0$,
- $\forall \mathbf{x}.\, \mathbf{x} \in I(\ell) \cap G(\tau) \cap G_1 \cap \neg G_2 \Rightarrow \delta(\ell_1) \cdot \eta(\ell_1, \mathbf{x}) \geq 0$, and
- $\forall \mathbf{x}.\, \mathbf{x} \in I(\ell) \cap G(\tau) \cap \neg G_1 \cap G_2 \Rightarrow \delta(\ell_2) \cdot \eta(\ell_2, \mathbf{x}) \geq 0$.

Each condition can be encoded via linear constraint as in [ADFG10, ACN18]. Clearly, the size of the encoding is polynomial. An important thing to note is that the negations in G_1 and G_2 might result in strict inequalities appearing in the above constraints. However, it was shown in [CFNH18] that this is not an issue for the Farkas' lemma (FL) conversion. Indeed, Lemma 4.3 in [CFNH18] shows that, whenever a system of linear inequalities on the LHS of a constraint is feasible, the strict inequalities may without loss of generality be replaced by non-strict inequalities. On the other hand, Lemma 4.7 in [CFNH18] shows that this check can be done in polynomial time.

In each algorithm iteration, all generalized transitions that have been 1-ranked are removed from $\mathcal{T}$ and the algorithm proceeds to the next iteration. If all generalized transitions are removed from $\mathcal{T}$, the algorithm concludes that the program admits a LinGLexRSM map (obtained by increasing the constructed LEM by a constant defined in the proof of Lemma 3.4.1). If in some iteration a new component which 1-ranks at least 1 generalized transition in $\mathcal{T}$ cannot be found, the program does not admit a LinGLexRSM map.

We show that Algorithm 3.1 satisfies the claim of Theorem 3.4.2.

Proof of Theorem 3.4.2. We first prove that the algorithm is sound, i.e. that $\boldsymbol{\eta} = (\eta_1, \ldots, \eta_d)$ computed by Algorithm 3.1 is a LinGLexRSM map supported by I, and thus that $\mathcal{C}$ is a.s. terminating. Define the level map lev : $\Delta \to \{0, 1 \ldots, d\}$ with the self loop at ℓ_{out} having level 0, and for any other generalized transition τ we define lev(τ) as the index of algorithm iteration in which it was removed from $\mathcal{T}$. The fact that $\boldsymbol{\eta}$ computed in lines 1-8 in Algorithm 3.1 satisfies *P-NNEG*, *P-RANK*, *EXP-NNEG* for generalized transitions of probabilistic branching and *W-EXP-NNEG* for all other generalized transitions then easily follows from conditions imposed by the algorithm in each iteration. From the proof of Lemma 3.4.1, it then follows that $\boldsymbol{\eta}$ obtained upon increasing each component by a constant term in lines 9-13 satisfies *EXP-NNEG* for every generalized transition. Hence $\boldsymbol{\eta}$ is a LinGLexRSM map supported by I and this concludes the soundness proof.

The proof of completeness as well as of the minimality of dimension is similar in spirit to the completeness proofs for the algorithm of [ADFG10] for computing LinLexRFs in non-probabilistic programs and for the algorithm of [ACN18] for computing LinLexRSMs in PPs. First, we observe that a pointwise sum of two LinGLexRSM maps supported by I is also a LinGLexRSM map supported by I. This follows by linearity of integration and therefore the pre-expectation operator. The argument is straightforward, thus we omit it. However, this simple observation will be central in the rest of the proof. Suppose first that the program admits a LinGLexRSM $\boldsymbol{\eta}' = (\eta'_1, \ldots, \eta'_m)$ supported by I. We show that Algorithm 3.1 then finds one such LinGLexRSM map (up to a constant term), hence the algorithm is complete. We prove this by contradiction. Suppose that the algorithm stops after the d-th iteration, after having computed $(\eta_1, \ldots, \eta_d)$ but with $\mathcal{T}$ still containing at least one generalized transition. Then $(\eta_1, \ldots, \eta_d)$ does not rank every generalized transition in the pCFG. Thus, $\boldsymbol{\eta}'$ ranks strictly more generalized transitions than $(\eta_1, \ldots, \eta_d)$. We distinguish two cases:

1. There exists the smallest $1 \leq j \leq \min\{d, m\}$ such that
   - for each $1 \leq j' < j$, $\eta_{j'}$ and $\eta'_{j'}$ would rank exactly the same generalized transitions if computed by the algorithm in the j'-th iteration, but

- η'_j ranks a generalized transition which is not ranked by η_j in the j-th iteration of the algorithm.

Then the algorithm could have ranked strictly more generalized transitions by computing $\eta_j + \eta'_j$ instead of η_j, which contradicts the maximality condition for computing new components that is imposed by the algorithm.

2. There is no such index. But then, since $\boldsymbol{\eta}' = (\eta'_1, \ldots, \eta'_m)$ is the LinGLexRSM supported by I, it must follow that $m > d$ and that η'_{d+1} would satisfy all the conditions imposed by the algorithm in the $(d+1)$-st iteration and it would rank at least 1 new generalized transition, thus the algorithm couldn't terminate after iteration d.

Thus, in both cases we reach contradiction, and completeness of Algorithm 3.1 follows.

Minimality of dimension is proved analogously, by contradiction. If there exists a LinGLexRSM map w.r.t. S supported by I of dimension strictly smaller by that found by the algorithm, we can use it analogously as above to show that at some iteration the algorithm could have ranked a strictly larger number of generalized transitions, contradicting the maximality condition for computing new components that is imposed by the algorithm. Thus the minimality of dimension claim follows. □

We conclude by showing that our motivating example in Fig. 3.1b admits a LinGLexRSM map supported by a very simple linear invariant. Thus, by completeness, our algorithm is able to prove its a.s. termination.

Example 3.4.1. *Consider the program in Figure 3.1b with a linear invariant $I(\ell_0) =$ true, $I(\ell_1) = x \geq -7$. Its a.s. termination is witnessed by a LEM $\boldsymbol{\eta}(\ell_0, (x, y)) = (1, x+7, y+7)$, $\boldsymbol{\eta}(\ell_1, (x, y)) = (1, x+8, y+7)$ and $\boldsymbol{\eta}(\ell_{out}, (x, y)) = (0, x+7, y+7)$. Since $\Delta_{PB} = \emptyset$ here, and since P-RANK, P-NNEG and W-EXP-NNEG are satisfied by $\boldsymbol{\eta}$, by Lemma 3.4.1, $\mathcal{C}$ admits a LinGLexRSM map supported by I.*

3.4.2 Algorithm for general LinPPs

While imposing *W-EXP-NNEG* lets us avoid integration in LinPPs with the BSP, this is no longer the case if we discard the BSP. Intuitively, the problem in imposing the condition *W-EXP-NNEG* instead of *EXP-NNEG* for LinPPs without the BSP, is that the set of states of smaller level over which *EXP-NNEG* performs integration might have a very small probability, however the value of the LinGLexRSM component on that set is negative and arbitrarily large in absolute value. Thus, a naive solution for general LinPPs would be to "cut off" the tail events where the LinGLexRSM component can become arbitrarily negative and over-approximate them by a constant value in order

to obtain a piecewise linear GLexRSM map. However, this might lead to the jump in maximal pre-expectation and could violate *P-RANK*.

In what follows, we impose a slight restriction on the syntax of LinPPs that we consider, and introduce a new condition on LEMs that allows the over-approximation trick mentioned above while ensuring that the *P-RANK* condition is not violated. We consider the subclass LinPP* of LinPPs in which no generalized transition of probabilistic branching and a generalized transition with a sampling instruction share a target location. This is a very mild restriction (satisfied, e.g. by our motivating example in Fig. 3.1b) which is enforced for technical reasons arising in the proof of Lemma 3.4.3. Each LinPP can be converted to satisfy this property by adding a **skip** instruction in the program's source code where necessary. Second, using the notation of Definition 3.3.4, we define the new condition *UNBOUND* for a generalized transition τ as follows:

$$\begin{aligned}\textit{UNBOUND}(\boldsymbol{\eta}, \tau) \equiv\ & \text{if } \tau = (\ell, \ell') \text{ with } \ell \in L_A \text{ and } Up(\tau) = (i, u) \text{ with } u \\ & \text{a probability distribution of unbounded support, then the coefficient of} \\ & \text{the variable with index } i \text{ in } \eta_j(\ell') \text{ is } 0 \text{ for all } 1 \leq j < \mathsf{lev}(\tau).\end{aligned}$$

The following technical lemma is an essential ingredient in the soundness proof of our algorithm for programs in LinPP*.

Lemma 3.4.3. *Let $\mathcal{C}$ be a LinPP* and I be a linear invariant in $\mathcal{C}$. If a LEM $\boldsymbol{\eta}$ satisfies P-RANK and P-NNEG for all generalized transitions, EXP-NNEG for all generalized transitions of probabilistic branching, W-EXP-NNEG for all other generalized transitions, as well as UNBOUND, then $\mathcal{C}$ admits a piecewise linear GLexRSM map supported by I.*

Proof sketch, full proof in Section 3.6.3. Analogously as in the proof of Lemma 3.4.1, we may increase $\boldsymbol{\eta}$ by a constant term in order to ensure that all generalized transitions satisfy *EXP-NNEG*, except for maybe those that in the variable update involve sampling from a distribution of unbounded support. So, without loss of generality, assume that $\boldsymbol{\eta}$ satisfies *EXP-NNEG* for all other generalized transitions. Denote the set of all generalized transitions in $\mathcal{C}$ that involve sampling from distributions of unbounded support by $\mapsto^{\mathsf{unb}}$.

As before, denote by max-coeff($\boldsymbol{\eta}$) the maximal absolute value of a coefficient appearing in $\boldsymbol{\eta}$. Also, define N analogously as in the proof of Lemma 3.4.1, i.e. for all distributions of bounded support that appear in sampling instructions and for all bounded intervals appearing in non-deterministic assignments, we have that they are supported in $[-N, N]$. Finally, since in our definition of pCFGs in Section 2.2 we assume that each distribution appearing in sampling instructions is integrable, we have $\mathbb{E}_{X\sim d}[|X|] < \infty$ for each $d \in \mathcal{D}$ where we use $\mathcal{D}$ to denote the set of all probability distributions appearing in sampling instructions in $\mathcal{C}$. Note that $\mathcal{D}$ is finite since the number of transitions in $\mathcal{C}$

is finite. Thus, by triangle inequality, we also have $\mathbb{E}_{X\sim d}[|X - \mathbb{E}[X]|] < \infty$. Hence, as $\mathbb{E}_{X\sim d}[|X - \mathbb{E}[X]| \cdot \mathbb{I}(|X - \mathbb{E}[X]| < k)] \to \mathbb{E}_{X\sim d}[|X - \mathbb{E}[X]|]$ as $k \to \infty$ by the Monotone Convergence Theorem [Wil91], for each $d \in \mathcal{D}$ there exists $k(d) \in \mathbb{N}$ such that

$$\mathbb{E}_{X\sim d}\Big[|X - \mathbb{E}[X]| \cdot \mathbb{I}\Big(|X - \mathbb{E}[X]| \geq k\Big)\Big] < \frac{1}{2 \cdot \text{max-coeff}(\boldsymbol{\eta})}.$$

for all $k \geq k(d)$. Define $K = \max_{d\in\mathcal{D}} k(d)$, which is finite as $\mathcal{D}$ is finite.

Next, define the set $U \subseteq \{1, 2, \ldots, \dim(\boldsymbol{\eta})\} \times L$ of pairs of indices of components of $\boldsymbol{\eta}$ and locations in $\mathcal{C}$ as follows:

$$U = \{(j, \ell') \mid \exists \tau \in \mapsto^{\text{unb}} \text{ s.t. } \text{lev}(\tau) = j \text{ and } \ell' \text{ is the target location of } \tau\}.$$

Thus, U is the set of pairs of indices of components of $\boldsymbol{\eta}$ and locations in $\mathcal{C}$ on which the condition *UNBOUND* imposes additional template restrictions.

We define an LEM $\boldsymbol{\eta}'$ which is of the same dimension as $\boldsymbol{\eta}$, and for each component η'_j we define

$$\eta'_j(\ell, \mathbf{x}) = \begin{cases} 0 & \text{if } (j, \ell) \in U \text{ and } \eta_j(\ell, \mathbf{x}) < -C, \\ 2 \cdot \eta_j(\ell, \mathbf{x}) + 2 \cdot C & \text{otherwise,} \end{cases}$$

where $C > 0$ is a constant to be determined. We claim that there exists $C > 0$ for which $\boldsymbol{\eta}'$ is a piecewise linear GLexRSM map supported by I (with the level map being the same as for $\boldsymbol{\eta}$). The fact that $\boldsymbol{\eta}'$ is piecewise linear for every $C > 0$ is clear from its definition, so we are left to verify that there exists $C > 0$ for which *P-NNEG*$(\boldsymbol{\eta}', \tau)$, *P-RANK*$(\boldsymbol{\eta}', \tau)$ and *EXP-NNEG*$(\boldsymbol{\eta}', \tau)$ hold for each generalized transition τ. A careful analysis, which can be found in Section 3.6.3, shows that $C = (2N + K) \cdot \text{max-coeff}(\boldsymbol{\eta}) + 1$ satisfies the claim. Hence, $\mathcal{C}$ admits a piecewise linear GLexRSM map supported by I. □

Algorithm. The new algorithm shares an overall structure with the algorithm from Section 3.4.1. Thus, we only give a high level overview and focus on novel aspects. The algorithm pseudocode is presented in Algorithm 3.2.

The condition *UNBOUND* is encoded by modifying the templates for the new LEM components. Let $\mapsto^{\text{unb}}$ be the set of generalized transitions in $\mathcal{C}$ induced by a transition in $\mathcal{C}$ containing sampling from unbounded support distributions, and for any such generalized transition τ let ℓ'_τ be the target location of the transition in $\mathcal{C}$ that induces it. Then for any set of generalized transitions $\mathcal{T}$, construct a linear program $\mathcal{LP}^{\text{unb}}_{\mathcal{T}}$ analogously to $\mathcal{LP}_{\mathcal{T}}$ in Section 3.4.1, additionally enforcing that for each $\tau \in \mapsto^{\text{unb}} \cap\, \mathcal{T}$, the coefficient of the variable updated by τ in the LEM template at ℓ'_τ is 0. Algorithm 3.2 first tries to prune as many generalized transitions as possible by repeatedly solving $\mathcal{LP}^{\text{unb}}_{\mathcal{T}}$ and removing ranked generalized transitions from $\mathcal{T}$, see lines 3-6. Once no

Algorithm 3.2: Synthesis of LinGLexRSM maps in PPs contained in LinPP*.

input : A LinPP* $\mathcal{C}$, linear invariant I.
output : An LEM satisfying the conditions of Lemma 3.4.3, if it exists

```
1  T ⟵ all generalized transitions in C; d ⟵ 0
2  while T is non-empty do
3      construct LP_T^unb
4      if LP_T^unb is feasible then
5          d ⟵ d + 1; η_d ⟵ LEM defined by the optimal solution of LP_T^unb
6          T ⟵ T \ {τ ∈ T | τ is 1-ranked by η_d}
7      end
8      else
9          found ⟵ false
10         for τ_0 ∈ ↦^unb ∩ T do
11             construct LP_T^{τ_0,unb}
12             if LP_T^{τ_0,unb} is feasible then
13                 d ⟵ d + 1; found ⟵ true
14                 η_d ⟵ LEM defined by the optimal solution of LP_T^{τ_0,unb}
15                 T ⟵ T \ {τ ∈ T | τ is 1-ranked by η_d}
16             end
17         end
18         if not found then return No LEM as in Lemma 3.4.3
19     end
20 end
21 return (η_1, ..., η_d)
```

more generalized transitions can be ranked, the algorithm tries to rank new generalized transitions by allowing non-zero template coefficients previously required to be 0, while still enforcing *UNBOUND*. For a set of generalized transitions $\mathcal{T}$ and for $\tau_0 \in \mapsto^{\mathsf{unb}} \cap \mathcal{T}$, we construct a linear program $\mathcal{LP}_{\mathcal{T}}^{\tau_0,\mathsf{unb}}$ analogously to $\mathcal{LP}_{\mathcal{T}}^{\mathsf{unb}}$ but allowing a non-zero coefficient of the variable updated by τ_0 at ℓ'_{τ_0}. However, we further impose that the new component 1-ranks any other generalized transition in $\mapsto^{\mathsf{unb}} \cap \mathcal{T}$ with the target location ℓ'_{τ_0}. This new linear program is solved for all $\tau_0 \in \mapsto^{\mathsf{unb}} \cap \mathcal{T}$ and all 1-ranked generalized transitions are removed from $\mathcal{T}$, as in Algorithm 3.2, lines 7-15. The process continues until all generalized transitions are pruned from $\mathcal{T}$ or until no remaining generalized transition can be 1-ranked, in which case no LEM as in Lemma 3.4.3 exists.

Theorem 3.4.4. *Algorithm 3.2 decides in polynomial time if a LinPP* $\mathcal{C}$ admits an LEM which satisfies all conditions of Lemma 3.4.3 and which is supported by I. Thus, if the algorithm outputs an LEM, then $\mathcal{C}$ is a.s. terminating and admits a piecewise*

linear GLexRSM map supported by I.

The proof of Theorem 3.4.4 uses Lemma 3.4.3 and the completeness argument is similar to that in the proof of Theorem 3.4.2, thus we omit the details. The full proof can be found in Section 3.6.4. We conclude by showing that Algorithm 3.2 can prove a.s. termination of our motivating example in Fig. 3.1a and the simple loop in Fig. 3.3.

Example 3.4.2. *Consider the program in Figure 3.1a with a linear invariant $I(\ell_0) = \text{true}$, $I(\ell_1) = y \geq 0$. The LEM defined via $\boldsymbol{\eta}(\ell_0, (x, y)) = (1, 2y + 2, x + 1)$, $\boldsymbol{\eta}(\ell_1, (x, y)) = (1, 2y+1, x+1)$ and $\boldsymbol{\eta}(\ell_{out}, (x, y)) = (0, 2y+2, x+1)$ satisfies P-RANK, P-NNEG and W-EXP-NNEG, which is easy to check. Furthermore, the only generalized transition containing a sampling instruction is the self-loop at ℓ_1 which is ranked by the third component of $\boldsymbol{\eta}$. As the coefficients of x of the first two components at ℓ_1 are equal to 0, $\boldsymbol{\eta}$ also satisfies UNBOUND. Hence, $\boldsymbol{\eta}$ satisfies all conditions of Lemma 3.4.3 and Algorithm 3.2 proves a.s. termination.*

Example 3.4.3. *Consider now the simple loop in Figure 3.3 with a linear invariant $I(\ell_0) = \text{true}$. The LEM defined via $\boldsymbol{\eta}(\ell_0, x) = (1, x + 1)$, and $\boldsymbol{\eta}(\ell_{out}, x) = (0, x + 1)$ satisfies P-RANK, P-NNEG and W-EXP-NNEG, which is easy to check. The only generalized transition containing a sampling instruction is the self-loop at ℓ_0 which is ranked by the second component of $\boldsymbol{\eta}$, and the coefficient of x of the first components at ℓ_0 is equal to 0. Hence, $\boldsymbol{\eta}$ also satisfies UNBOUND. Hence, $\boldsymbol{\eta}$ satisfies all conditions of Lemma 3.4.3 and Algorithm 3.2 proves a.s. termination.*

3.5 Related Work

Martingale-based termination literature mostly focused on 1-dimensional RSMs [CS13, CFNH18, CFG16, CNZ17, FH15, MMKK18, HFC18, FC19, MBKK21a, GGH19]. RSMs themselves can be seen as generalizations of Lyapunov ranking functions from control theory [BG05, Fos53]. Recently, the work [HFCG19] pointed out the unsoundess of the 1-dimensional RSM-based proof rule in [FH15] due to insufficient lower bound conditions and provided a corrected version. On the multi-dimensional front, it was shown in [FH15] that requiring components of (lexicographic) RSMs to be nonnegative only at points where they are used to rank some enabled transition (analogue of Bradley-Manna-Sipma LexRFs [BMS05a]) is unsound for proving a.s. termination. This illustrates the intricacies of dealing with lower bounds in the design of a.s. termination certificates. Lexicographic RSMs with strong non-negativity were introduced in [ACN18]. The work [CH20] produces an ω-regular decomposition of program's control-flow graph, with each program component ranked by a different RSM. This approach does not require a lexicographic ordering of RSMs, but each component in the decomposition must be ranked by a single-dimensional non-negative RSM. RSM approaches were also

used for cost analysis [NCH18, WFG+19, AMS20] and additional liveness and safety properties [CVS16, BEFFH16, CNZ17, CGMZ22b].

Logical calculi for reasoning about properties of PPs (including termination) were studied in [Koz81, FH82, Koz85, Fel84] and extended to programs with non-determinism in [MM04, MM05, KKMO18, OKKM16, GKM14]. In particular [MM04, MM05, MMKK18] formalize RSM-like proof certificates within the *weakest pre-expectation (WPE)* calculus [MMS96, MM99]. The power of this calculus allows for reasoning about complex programs [MMKK18, Section 5], but the proofs typically require a human input. Theoretical connections between martingales and the WPE calculus were recently explored in [HKGK20]. There is also a rich body of work on analysis of probabilistic functional programs, where the aim is typically to obtain a general type system [LG19, ADG19, KLG20, DLFR21] for reasoning about termination properties (automation for discrete probabilistic term rewrite systems was shown in [ALY20]).

As for other approaches to a.s. termination, for *finite-state programs* with nondeterminism a sound and complete method was given in [EGK12], while [Mon01] considers a.s. termination proving through abstract interpretation. The work [KKM18] shows that proving a.s. termination is harder (in terms of arithmetical hierarchy) than proving termination of non-probabilistic programs.

The computational complexity of the construction of lexicographic ranking functions in non-probabilistic programs was studied in [BAG13, BAG15].

3.6 Technical Proofs

3.6.1 Proof of Inequality in the Proof of Theorem 3.2.2

We prove the inequality that was used in the proof of Theorem 3.2.2, which claims that for each $t \geq s$ we have

$$\mathbb{E}[Y_{t+1}] \leq \mathbb{E}[Y_t] - \mathbb{P}[L_k^t \cap D \cap \{F > t\}].$$

To prove the inequality, note that

$$\mathbb{E}[Y_{t+1}] = \mathbb{E}[Y_{t+1} \cdot \mathbb{I}(\Omega \backslash D)] + \mathbb{E}[Y_{t+1} \cdot \mathbb{I}(D \cap \{F \leq t\})] + \mathbb{E}[Y_{t+1} \cdot \mathbb{I}(D \cap \{F > t\})], \quad (3.4)$$

where

- $\mathbb{E}[Y_{t+1} \cdot \mathbb{I}(\Omega \backslash D)] = 0$ since $Y_{t+1} = 0$ on $\Omega \backslash D$;
- $\mathbb{E}[Y_{t+1} \cdot \mathbb{I}(D \cap \{F \leq t\})] = \mathbb{E}[Y_t \cdot \mathbb{I}(D \cap \{F \leq t\})]$, as $Y_{t+1} = Y_t$ on $D \cap \{F \leq t\}$;

- $\mathbb{E}[Y_{t+1} \cdot \mathbb{I}(D \cap \{F > t\})] = \mathbb{E}[\mathbf{X}_{t+1}[k] \cdot \mathbb{I}(D \cap \{F > t\})]$, as $Y_{t+1} = \mathbf{X}_{t+1}[k]$ on $D \cap \{F > t\}$.

Now, by the theorem assumptions the conditional expectation $\mathbb{E}[\mathbf{X}_{t+1}[k] \mid \mathcal{F}_t]$ exists. Hence, as $D \cap \{F > t\}$ is $\mathcal{F}_t$-measurable for $t \geq s$, we have that

$$\mathbb{E}\Big[\mathbf{X}_{t+1}[k] \cdot \mathbb{I}(D \cap \{F > t\})\Big] = \mathbb{E}\Big[\mathbb{E}[\mathbf{X}_{t+1}[k] \mid \mathcal{F}_t] \cdot \mathbb{I}(D \cap \{F > t\})\Big].$$

By the theorem assumptions, we also know that $\mathbb{E}[\mathbf{X}_{t+1}[k] \mid \mathcal{F}_t] \leq \mathbf{X}_t[k] - \mathbb{I}(L_k^t)$ on $\cup_{j=k}^n L_j^t$. Thus, as $D \cap \{F > t\} \subseteq \cup_{j=k}^n L_j^t$ for $t \geq s$, by plugging this into the previous equation we conclude

$$\begin{aligned}\mathbb{E}\Big[\mathbf{X}_{t+1}[k] \cdot \mathbb{I}(D \cap \{F > t\})\Big] &\leq \mathbb{E}\Big[(\mathbf{X}_t[k] - \mathbb{I}(L_k^t)) \cdot \mathbb{I}(D \cap \{F > t\})\Big] \\ &= \mathbb{E}\Big[(Y_t - \mathbb{I}(L_k^t)) \cdot \mathbb{I}(D \cap \{F > t\}\Big],\end{aligned}$$

where in the last row we used that $\mathbf{X}_t[k] = Y_t$ on $D \cap \{F > t\}$. Summing up our upper bounds on each term on the RHS of eq. (3.4), we conclude that

$$\begin{aligned}\mathbb{E}[Y_{t+1}] &\leq 0 + \mathbb{E}[Y_t \cdot \mathbb{I}(D \cap \{F \leq t\})] + \mathbb{E}\Big[(Y_t - \mathbb{I}(L_k^t)) \cdot \mathbb{I}(D \cap \{F > t\})\Big] \\ &= \mathbb{E}[Y_t \cdot \mathbb{I}(D)] - \mathbb{P}[L_k^t \cap D \cap \{F > t\}], \\ &= \mathbb{E}[Y_t] - \mathbb{P}[L_k^t \cap D \cap \{F > t\}].\end{aligned}$$

where in the last row we used that $Y_t = 0$ on $\Omega \backslash D$. This proves the desired inequality.

3.6.2 Proof of Theorem 3.3.5

Theorem (Soundness of GLexRSM-maps for a.s. termination)**.** *Let $\mathcal{C}$ be a pCFG and I an invariant in $\mathcal{C}$. Suppose that $\mathcal{C}$ admits an n-dimensional GLexRSM map $\boldsymbol{\eta}$ supported by I, for some $n \in \mathbb{N}$. Then $\mathcal{C}$ terminates a.s.*

Proof. Let $\mathcal{C} = (L, V, \ell_{init}, \mathbf{x}_{init}, \mapsto, G, Pr, Up)$ be a pCFG. The proof proceeds by using the GlexRSM map $\boldsymbol{\eta}$ to define a GLexRSM w.r.t. the termination time $TimeTerm$ in the probability space $(\Omega_{\mathcal{C}}, \mathcal{F}_{\mathcal{C}}, \mathbb{P})$ associated to the pCFG. Here, $\mathbb{P}$ is a probability measure defined by an arbitrary (but throughout the proof fixed) scheduler and an initial configuration. The proof then uses Theorem 3.2.2 to conclude a.s. termination of the program. See Section 2.2 for necessary preliminaries on pCFG semantics and Section 2.4 for the formal definition of termination time.

First, however, we state again definition the auxiliary notion of *pre-expectation with respect to a scheduler*, also stated in Chapter 3.

Definition (Pre-expectation with respect to a scheduler)**.** *Let S be a set of measurable states in $\mathcal{C}$. The* pre-expectation with respect to a scheduler σ of a MM η *given S is a map $pre_{\sigma,\eta,S} : Fpath_{\mathcal{C}} \to \mathbb{R}$ defined as follows. Let $\rho \in Fpath_{\mathcal{C}}$ be a finite path ending in $(\ell, \mathbf{x})$. We define $pre_{\sigma,\eta,S}(\rho)$ as follows:*

1. *If $\ell \in L_N$ is a location of non-deterministic branching, denote by $\sigma(\rho)$ the probability distribution over (generalized) transitions enabled in $(\ell, \mathbf{x})$ defined by the scheduler σ. Then*

$$pre_{\sigma,\eta,S}(\rho) = \sum_{\tau \in supp(\sigma(\rho))} \sigma(\rho)(\tau) \cdot pre^{\tau}_{\sigma,\eta,S}(\ell, \mathbf{x}).$$

2. *Otherwise, let τ be the unique generalized transition with source location ℓ. Then*

$$pre_{\sigma,\eta,S}(\rho) = pre^{\tau}_{\sigma,\eta,S}(\ell, \mathbf{x}),$$

where $pre^{\tau}_{\sigma,\eta,S}$ is defined in the same way as the standard pre-expectation $pre^{\tau}_{\eta,S}$, except for the case *when τ is transition (ℓ, ℓ') whose update element is a non-deterministic assignment from an interval, i.e. $u(\tau) = (j, u)$ with u being an interval; in such a case, we put $pre^{\tau}_{\sigma,\eta,S}(\ell, \mathbf{x}) = \eta(\ell', \mathbf{x}(j \leftarrow \mathbb{E}[d_{\sigma}(\rho, \tau) \cdot \mathbb{I}(S)])$.*

Intuitively, $pre_{\sigma,\eta,S}$ takes a finite path $\rho \in Fpath_{\mathcal{C}}$ as an input, and returns the *expected value* of η in the next step when the integration is performed over program states in S, given the program run history and the choices of the scheduler σ.

For a run $\rho \in \Omega_{\mathcal{C}}$, let $(\ell^{\rho}_i, \mathbf{x}^{\rho}_i)$ denote the i-th configuration along ρ, τ^{ρ}_i the i-th generalized transition taken along ρ and ρ_i the prefix of ρ of length i. We define an n-dimensional stochastic process $(\mathbf{X}_i)_{i=0}^{\infty}$ over $(\Omega_{\mathcal{C}}, \mathcal{F}_{\mathcal{C}}, \mathbb{P})$ by setting

$$\mathbf{X}_i[j](\rho) = \begin{cases} \boldsymbol{\eta}_j(\ell^{\rho}_i, \mathbf{x}^{\rho}_i) & \text{if } \ell^{\rho}_i \neq \ell_{out} \\ -1 & \text{otherwise.} \end{cases}$$

for each $i \in \mathbb{N}_0$, $1 \leq j \leq n$ and $\rho \in \Omega_{\mathcal{C}}$. We consider the canonical filtration $(\mathcal{R}_i)_{i=0}^{\infty}$ where $\mathcal{R}_i$ is the smallest sub-sigma-algebra of $\mathcal{F}_{\mathcal{C}}$ such that all the functions $\ell_j(\rho) = \ell^{\rho}_j$, $\mathbf{x}_j(\rho) = \mathbf{x}^{\rho}_j$, $0 \leq j \leq i$, are $\mathcal{R}_i$-measurable. We also consider the stopping time $TimeTerm$ with respect to $(\mathcal{R}_i)_{i=0}^{\infty}$ defined by the first hitting time of ℓ_{out}. For each $i \in \mathbb{N}_0$, we define a partition of $\{TimeTerm > i\}$ into n sets $L^i_1, \ldots, L^i_n$ with $L^i_j = \{\rho \in \Omega_{\mathcal{C}} \mid \ell^{\rho}_i \neq \ell_{out},$ level of $(\ell^{\rho}_i, \mathbf{x}^{\rho}_i)$ is $j\}$. We show that $(\{\mathbf{X}^{\infty}_{i=0}, \{L^i_1, \ldots, L^i_n\}^{\infty}_{i=0})$ is an instance of a GLexRSM in the probability space $(\Omega_{\mathcal{C}}, \mathcal{F}_{\mathcal{C}}, \mathbb{P})$. We prove this by verifying each of the defining conditions in Definition 3.2.1:

1. Clearly, each $\mathbf{X}_i[j]$ is $\mathcal{R}_i$-measurable as $\mathbf{X}_i[j]$ is defined in terms of the i-th configuration of a program run.

2. We now show that all conditional expectations required in Definition 3.2.1 exist. Let $B \in \mathcal{R}_{i+1}$, we need to show that $\mathbb{E}[\mathbf{X}_{i+1}[j] \cdot \mathbb{I}(B) \mid \mathcal{R}_i]$ exists. Fix $\rho \in \Omega_{\mathcal{C}}$, and let ρ_i denote a finite prefix of ρ of length i. Then define $\mathsf{proj}(B)(\rho_i)$ as

$$\mathsf{proj}(B)(\rho_i) = \{(\ell_{i+1}^{\rho'}, \mathbf{x}_{i+1}^{\rho'}) \mid \rho' \in B \wedge \rho'_i = \rho_i\},$$

i.e. it is the set of all $(i+1)$-st states of infinite runs in B whose finite prefix of length i coincides with ρ_i. Then we show that

$$\mathbb{E}[\mathbf{X}_{i+1}[j] \cdot \mathbb{I}(B) \mid \mathcal{R}_i](\rho) = \mathsf{pre}_{\sigma,\boldsymbol{\eta}_j,\mathsf{proj}(B)(\rho_i)}(\rho_i),$$

i.e. this conditional expectation can be expressed in terms of the pre-expectation w.r.t. the scheduler that we define in Definition 3.3.7.

To see this, recall that the conditional expectation $\mathbb{E}[\mathbf{X}_{i+1}[j] \cdot \mathbb{I}(B) \mid \mathcal{R}_i]$ is defined as the unique $\mathcal{R}_i$-measurable random variable Y for which $\mathbb{E}[Y \cdot \mathbb{I}(A)] = \mathbb{E}[\mathbf{X}_{i+1}[j] \cdot \mathbb{I}(B \cap A)]$ for any $A \in \mathcal{R}_i$. For fixed $A \in \mathcal{R}_i$, $\mathbb{E}[\mathbf{X}_{i+1}[j] \cdot \mathbb{I}(B \cap A)]$ is equal to the integral of $\mathbf{X}_{i+1}[j]$ when integration is done over all runs in B whose finite prefix of length i ensures that the run belongs to A. But this is exactly the value obtained by integrating $\mathsf{pre}_{\sigma,\boldsymbol{\eta}_j,\mathsf{proj}(B)(\rho_i)}(\rho_i)$ over runs $\rho \in A$. This proves that $\mathsf{pre}_{\sigma,\boldsymbol{\eta}_j,\mathsf{proj}(B)(\rho_i)}(\rho_i)$ satisfies all the properties of the conditional expectation for any $\rho \in \Omega_{\mathcal{C}}$. As conditional expectation is a.s. unique whenever it exists [Wil91], the claim follows.

3. To see that lexicographic ranking, nonnegativity and boundedness of conditional expectation conditions in Definition 3.2.1 hold, we need to show that for each $i \in \mathbb{N}_0$, $1 \leq j \leq n$ and $\rho \in L_j^i$, we have

   - $\mathbb{E}[\mathbf{X}_{i+1}[j'] \mid \mathcal{F}_i](\rho) \leq \mathbf{X}_i[j'](\rho)$ for each $1 \leq j' < j$,
   - $\mathbb{E}[\mathbf{X}_{i+1}[j] \mid \mathcal{F}_i](\rho) \leq \mathbf{X}_i[j](\rho) - 1$,
   - $\mathbf{X}_i[j'](\rho) \geq 0$ for each $1 \leq j' \leq j$, and
   - $\mathbb{E}[\mathbf{X}_{i+1}[j'] \cdot \mathbb{I}(\cup_{t=0}^{j'-1} L_t^{i+1}) \mid \mathcal{F}_i](\rho) \geq 0$ for each $1 \leq j' \leq j$ (here, $L_0^{i+1} = \{TimeTerm \leq i+1\}$).

   Before checking these properties, we first prove in Proposition 3.6.1 below that, to show that a program terminates almost-surely, it suffices to consider those schedulers that to each finite program run assign a single transition to be taken. Thus, we without loss of generality assume that σ is such a scheduler, and let τ be the only transition in the support of $\sigma(\rho_i)$. Then since $\rho \in L_j^i$, we have $j = \mathsf{lev}(\tau)$. As τ is the only transition in the support of $\sigma(\rho_i)$, we have

$$\begin{aligned}\mathbb{E}[\mathbf{X}_{i+1}[j] \mid \mathcal{R}_i](\rho) = \mathsf{pre}_{\sigma,\boldsymbol{\eta_j}}(\rho_i) = \mathsf{pre}^{\tau}_{\sigma,\boldsymbol{\eta_j}}(\ell_i^\rho, \mathbf{x}_i^\rho) \leq \mathsf{max\text{-}pre}^{\tau}_{\boldsymbol{\eta_j}}(\ell_i^\rho, \mathbf{x}_i^\rho) \\ \leq \boldsymbol{\eta}_j(\ell_i^\rho, \mathbf{x}_i^\rho) - 1 = \mathbf{X}_i[j](\rho) - 1,\end{aligned}$$

The inequality $\text{pre}^{\tau}_{\sigma,\boldsymbol{\eta_j}}(\ell^{\rho}_i, \mathbf{x}^{\rho}_i) \leq \text{max-pre}^{\tau}_{\boldsymbol{\eta_j}}(\ell^{\rho}_i, \mathbf{x}^{\rho}_i)$ holds since $\text{pre}^{\tau}_{\sigma,\boldsymbol{\eta_j}}$ in Definition 3.3.7 and $\text{max-pre}^{\tau}_{\boldsymbol{\eta_j}}$ in Definition 3.3.3 only differ in the way in which we treat update elements given by nondeterministic assignments where in the first case we take expectation and in the second case the global maximum. The last inequality holds by the *P-RANK* condition of GLexRSM maps and since $j = \text{lev}(\tau)$. It follows analogously that $\mathbb{E}[\mathbf{X}_{i+1}[j'] \mid \mathcal{F}_i](\rho) \leq \mathbf{X}_i[j'](\rho)$ for each $1 \leq j' < j$. The fact that $\mathbf{X}_i[j'](\rho) \geq 0$ for each $1 \leq j' \leq j$ follows from the *P-NNEG* condition of GLexRSM maps. Finally, for each $1 \leq j' \leq j$ we have that $\mathbb{E}[\mathbf{X}_{i+1}[j'] \cdot \mathbb{I}(\cup_{t=0}^{j'-1} L_t^{i+1}) \mid \mathcal{F}_i](\rho) \geq \text{min-pre}^{\tau}_{\eta_{j'}, S^{\leq j'-1}_{\text{lev}}}(\ell^{\rho}_i, \mathbf{x}^{\rho}_i)$ as $\text{lev}(\tau) = j$, so the expected leftward nonnegativity follows from the *EXP-NNEG* condition of GLexRSM maps.

Therefore, $((\mathbf{X}_i)_{i=0}^{\infty}, (L_1^i, \ldots, L_n^i)_{i=0}^{\infty})$ is an instance of a GLexRSM w.r.t. the stopping time $TimeTerm$. Thus, and from Theorem 3.2.2 we have $\mathbb{P}[TimeTerm < \infty] = 1$. □

Proposition 3.6.1. *Let $\mathcal{C}$ be a pCFG and suppose that there exist a measurable scheduler σ and an initial configuration $(\ell_{init}, \mathbf{x}_{init})$ such that $\mathbb{P}^{\sigma}_{\ell_{init}, \mathbf{x}_{init}}[TimeTerm = \infty] > 0$. Then there exists a measurable scheduler σ^* which is deterministic in the sense that to each finite path it assigns a single transition with probability 1, such that $\mathbb{P}^{\sigma^*}_{\ell_{init}, \mathbf{x}_{init}}[TimeTerm = \infty] > 0$.*

Proof. In what follows we fix the initial configuration $(\ell_{init}, \mathbf{x}_{init})$ and omit it from the notation of the probability measure, which we denote by $\mathbb{P}^{\sigma}$. We construct σ^* by constructing a sequence $\sigma = \sigma_0, \sigma_1, \sigma_2, \ldots$ of schedulers, where for each $i \in \mathbb{N}_0$ we have that σ_{i+1} and σ_i agree on histories of length at most $i - 1$, and σ_{i+1} "refines" σ_i on histories of length i in such a way that:

- σ_{i+1} deterministically chooses transitions on histories of length at most i;
- σ_{i+1} is measurable;
- $\mathbb{P}^{\sigma_{i+1}}[TimeTerm = \infty] \geq \mathbb{P}^{\sigma_i}[TimeTerm = \infty]$.

Then we define the scheduler σ^* as $\sigma^*(\rho) = \sigma_i(\rho)$ whenever the length of a finite history ρ is i.

The construction of σ_{i+1} from σ_i proceeds as follows. For finite runs of length different than i, we let $\sigma_{i+1}(\rho) = \sigma_i(\rho)$. For finite runs of length i ending in a location $\ell \in L_A$ with the unique outgoing transition having non-deterministic update, we let $\sigma_{i+1}(\rho) = \sigma_i(\rho)$. We are left to define σ_{i+1} on histories of length exactly i ending in a state $(\ell, \mathbf{x})$ with a non-deterministic branching location $\ell \in L_N$. Fix such a finite

history ρ in $\mathcal{C}$ of length i. For each transition $\tau \in supp(\sigma_i(\rho))$, let $p_i(\rho, \tau)$ denote the probability of non-termination in the probability space of infinite runs, when starting in the last configuration of ρ, executing the transition τ and then following the scheduler σ_i (where the history ρ is taken into account to resolve nondeterminism). Then set

$$\tau_i(\rho) = \arg \max_{\tau \in supp(\sigma_i(\rho))} p_i(\rho, \tau),$$

i.e. the transition in the support of $\sigma(\rho)$ which maximizes this probability. Then we define $\sigma_{i+1}(\rho)$ to be a Dirac-delta distribution which assigns probability 1 to the transition $\tau_i(\rho)$.

We now check that the constructed scheduler σ_{i+1} satisfies each of the 3 properties above:

1. The fact that, σ_{i+1} is deterministic on histories of length at most i immediately follows by induction on i and by construction of σ_{i+1}.

2. To see that σ_{i+1} is measurable, we again proceed by induction on i and assume that σ_i is measurable (base case holds since $\sigma_0 = \sigma$). We need to show that, for each program transition τ, we have $\sigma_{i+1}^{-1}(\tau) \in \mathcal{F}_{\mathsf{fin}}$ where $\mathcal{F}_{\mathsf{fin}}$ is the σ-algebra of finite runs (for the definition of this σ-algebra, as well as for details on measurable schedulers, see [ACN18]). Write

   $$\sigma_{i+1}^{-1}(\tau) = A_{\tau,<i} \cup A_{\tau,=i} \cup A_{\tau,>i},$$

   where $A_{\tau,<i}$ is the set of all finite runs in $\sigma_{i+1}^{-1}(\tau)$ of length at most $i-1$ and analogously for the other two sets. Since σ_{i+1} and σ_i coincide on histories of length different than i, by measurability of σ_i it follows that $A_{\tau,<i}$ and $A_{\tau,>i}$ are in $\mathcal{F}_{\mathsf{fin}}$. Thus it suffices to show that $A_{\tau,=i} \in \mathcal{F}_{\mathsf{fin}}$.

   We partition $A_{\tau,=i}$ as $\cup_{\ell \in L} A_{\tau,=i,l}$, where $A_{\tau,=i,l}$ is the set of all finite runs in $A_{\tau,=i}$ with the last location being l. It suffices to prove that each $A_{\tau,=i,l} \in \mathcal{F}_{\mathsf{fin}}$.

   To show that $A_{\tau,=i,l} \in \mathcal{F}_{\mathsf{fin}}$, we again partition $A_{\tau,=i,l}$ in terms of transitions in the support of σ_i. Therefore, it suffices to prove for each finite set of transitions $T \subseteq \mapsto$ with $\tau \in T$ that $A_{\tau,=i,l,T} \in \mathcal{F}$, where $A_{\tau,=i,l,T}$ is the set of all finite paths of length i and ending in l such that $supp(\sigma_i(\rho)) = T$ and $p_i(\rho, \tau) \geq p_i(\rho, \tau')$ for any $\tau' \in T$.

   Since σ_i is measurable by induction hypothesis, all conditions except the last one define events in $\mathcal{F}_{\mathsf{fin}}$. To see that this is also the case for the condition that $p_i(\rho, \tau) \geq p_i(\rho, \tau')$ for any $\tau' \in T$, observe that this event can be rewritten as a projection onto the set of all finite paths of length i of the set of all infinite runs

$\tilde{\rho}$ satisfying the following property

$$\mathbb{E}^{\sigma_i}\Big[\mathbb{I}_{TimeTerm=\infty\wedge\tau_i^\rho=\tau}|\sigma(\mathcal{F}_i,\mathbb{I}_{\tau_i^\rho=\tau})\Big](\tilde{\rho}) \geq \max_{\tau'\in T}\mathbb{E}^{\sigma_i}\Big[\mathbb{I}_{TimeTerm=\infty\wedge\tau_i^\rho=\tau'}|\sigma(\mathcal{F}_i,\mathbb{I}_{\tau_i'^\rho=\tau'})\Big](\tilde{\rho}),$$

where $\sigma(\mathcal{F}_i,\mathbb{I}_{\tau_i'^\rho=\tau'})$ is the smallest σ-algebra containing $\mathcal{F}_i$ and the σ-algebra generated by the random variable $\mathbb{I}_{\tau_i'^\rho=\tau'}$. All random variables involved in the above inequality are $\mathcal{F}$-measurable by measurability of σ_i, and so the set of infinite runs satisfying this conditions is in $\mathcal{F}$. This shows that the last condition also defines a measurable event. Thus, $A_{\tau,=i,l}\in\mathcal{F}_{\text{fin}}$ and the claim follows.

3. We need to show that $\mathbb{P}^{\sigma_{i+1}}[TimeTerm=\infty]\geq\mathbb{P}^{\sigma_i}[TimeTerm=\infty]$ for each i. To do this, it suffices to show that $\mathbb{P}^{\sigma_{i+1}}[TimeTerm=\infty\wedge\ell_i^\rho=\ell]\geq\mathbb{P}^{\sigma_i}[TimeTerm=\infty\wedge\ell_i^\rho=\ell]$ for each location $\ell\in L$ where the event $\{\ell_i^\rho=\ell\}$ denotes that ℓ is the $(i+1)$-st location along the program run. If we show this, by taking the sum over all locations on both sides of the inequality, the claim follows.

   Fix a location ℓ. Then

$$\begin{aligned}\mathbb{P}^{\sigma_{i+1}}[TimeTerm=\infty\wedge\ell_i^\rho=\ell] &= \mathbb{E}^{\sigma_{i+1}}[\mathbb{I}_{TimeTerm=\infty}\cdot\mathbb{I}_{\ell_i^\rho=\ell}]\\ &= \mathbb{E}^{\sigma_{i+1}}[\mathbb{E}^{\sigma_{i+1}}[\mathbb{I}_{TimeTerm=\infty}\cdot\mathbb{I}_{\ell_i^\rho=\ell}|\mathcal{F}_i]]\\ &= \mathbb{E}^{\sigma_{i+1}}[\mathbb{I}_{\ell_i^\rho=\ell}\cdot\mathbb{E}^{\sigma_{i+1}}[\mathbb{I}_{TimeTerm=\infty}|\mathcal{F}_i]]\\ &= \mathbb{E}^{\sigma_i}[\mathbb{I}_{\ell_i^\rho=\ell}\cdot\mathbb{E}^{\sigma_{i+1}}[\mathbb{I}_{TimeTerm=\infty}|\mathcal{F}_i]].\end{aligned}\tag{3.5}$$

   The first equality holds since the probability of the event is equal to the expected value of its indicator function. The second equality holds by the definition of conditional expectation. The third equality holds since we may take out from the conditional expectation any variable that is measurable w.r.t. the σ-algebra that we condition on [Wil91]. The fourth inequality holds since the probability measures defined by σ_{i+1} and σ_i by construction agree on $\mathcal{F}_i$-measurable sets.

   But, by construction we have $\mathbb{E}^{\sigma_{i+1}}[\mathbb{I}_{TimeTerm=\infty}|\mathcal{F}_i](\rho)\geq\mathbb{E}^{\sigma_i}[\mathbb{I}_{TimeTerm=\infty}|\mathcal{F}_i](\rho)$ for each infinite run ρ, because in the $(i+1)$-st configuration σ_{i+1} picks the transition which maximizes the probability of non-termination among all transitions in $supp(\sigma_i(\rho_i))$. Hence, plugging this back into eq. (3.5) we conclude

$$\begin{aligned}\mathbb{P}^{\sigma_{i+1}}[TimeTerm=\infty\wedge\ell_i^\rho=\ell] &= \mathbb{E}^{\sigma_{i+1}}[\mathbb{I}_{TimeTerm=\infty}\cdot\mathbb{I}_{\ell_i^\rho=\ell}]\\ &= \mathbb{E}^{\sigma_i}[\mathbb{I}_{\ell_i^\rho=\ell}\cdot\mathbb{E}^{\sigma_{i+1}}[\mathbb{I}_{TimeTerm=\infty}|\mathcal{F}_i]]\\ &\geq \mathbb{E}^{\sigma_i}[\mathbb{I}_{\ell_i^\rho=\ell}\cdot\mathbb{E}^{\sigma_i}[\mathbb{I}_{TimeTerm=\infty}|\mathcal{F}_i]]\\ &= \mathbb{P}^{\sigma_i}[TimeTerm=\infty\wedge\ell_i^\rho=\ell],\end{aligned}$$

where the last equality follows by the same sequence of equalities as in eq. (3.5) with $i+1$ replaced by i. This proves the claim.

Hence, schedulers $\sigma = \sigma_0, \sigma_1, \sigma_2, \ldots$ satisfy all the desired properties. Finally, note that by construction σ^* is deterministic too. We are left to show that σ^* is measurable and that $\mathbb{P}^{\sigma^*}_{\ell_{init},\mathbf{x}_{init}}[TimeTerm = \infty] > 0$.

To see that σ^* is measurable, we need to show that $(\sigma^*)^{-1}(\tau)$ is in the σ-algebra of finite runs for each transition τ. This follows since, if we write $(\sigma^*)^{-1}(\tau) = \cup_{i\in\mathbb{N}_0} A_i$ where A_i is the set of all finite runs in $(\sigma^*)^{-1}(\tau)$ of length i, we have that each A_i is in the σ-algebra since σ^* coincides with σ_i on histories of length at most i.

To see that $\mathbb{P}^{\sigma^*}[TimeTerm = \infty] > 0$, suppose that on the contrary $\mathbb{P}^{\sigma^*}[TimeTerm = \infty] = 0$ so $\mathbb{P}^{\sigma^*}[TimeTerm < \infty] = 1$. Let $\delta = \mathbb{P}^{\sigma}[TimeTerm = \infty] > 0$, where σ is the potentially non-deterministic scheduler from the proposition statement. Since we have $\mathbb{P}^{\sigma^*}[TimeTerm \leq i] \to \mathbb{P}^{\sigma^*}[TimeTerm < \infty]$ as $i \to \infty$ by the Monotone Convergence Theorem [Wil91], there exists $i \in \mathbb{N}_0$ such that $\mathbb{P}^{\sigma^*}[TimeTerm \leq i] \geq 1 - \delta/2$. But

$$\mathbb{P}^{\sigma^*}[TimeTerm \leq i] \leq \mathbb{P}^{\sigma_i}[TimeTerm < \infty] \leq \mathbb{P}^{\sigma_0}[TimeTerm < \infty] = 1 - \delta$$

as $\mathbb{P}^{\sigma_i}[TimeTerm = \infty] \geq \mathbb{P}^{\sigma_0}[TimeTerm = \infty]$ by the above monotonicity property of $(\mathbb{P}^{\sigma_i}[TimeTerm = \infty])_{i=0}^{\infty}$. This gives contradiction, hence $\mathbb{P}^{\sigma^*}[TimeTerm = \infty] > 0$ as claimed. □

3.6.3 Proof of Lemma 3.4.3

Lemma. *Let $\mathcal{C}$ be a LinPP* and I be a linear invariant in $\mathcal{C}$. If a LEM $\boldsymbol{\eta}$ satisfies P-RANK and P-NNEG for all generalized transitions, EXP-NNEG for all generalized transitions of probabilistic branching, W-EXP-NNEG for all other generalized transitions, as well as UNBOUND, then $\mathcal{C}$ admits a piecewise linear GLexRSM map supported by I.*

Proof. Let $\boldsymbol{\eta}$ be an LEM whose existence is assumed in the lemma. Analogously as in the proof of Lemma 3.4.1, we may increase $\boldsymbol{\eta}$ by a constant term in order to ensure that all generalized transitions satisfy *EXP-NNEG*, except for maybe those induced by transitions that in the variable update involve sampling from a distribution of unbounded support. So without loss of generality assume that $\boldsymbol{\eta}$ satisfies *EXP-NNEG* for all other generalized transitions. Denote the set of all generalized transitions in $\mathcal{C}$ induced by transitions that involve sampling from distributions of unbounded support by $\mapsto^{\mathsf{unb}}$.

As before, denote by max-coeff($\boldsymbol{\eta}$) the maximal absolute value of a coefficient appearing in $\boldsymbol{\eta}$. Also, define N analogously as in the proof of Lemma 3.4.1, i.e. for all distributions of bounded support that appear in sampling instructions and for all bounded intervals

appearing in non-deterministic assignments, we have that they are supported in $[-N, N]$. Finally, since in our definition of pCFGs in Section 2.2 we assume that each distribution appearing in sampling instructions is integrable, we have $\mathbb{E}_{X\sim d}[|X|] < \infty$ for each $d \in \mathcal{D}$ where we use $\mathcal{D}$ to denote the set of all probability distributions appearing in sampling instructions in $\mathcal{C}$. Note that $\mathcal{D}$ is finite since the number of transitions in $\mathcal{C}$ is finite. Thus, by triangle inequality, we also have $\mathbb{E}_{X\sim d}[|X - \mathbb{E}[X]|] < \infty$. Hence, as $\mathbb{E}_{X\sim d}[|X - \mathbb{E}[X]| \cdot \mathbb{I}(|X - \mathbb{E}[X]| < k)] \to \mathbb{E}_{X\sim d}[|X - \mathbb{E}[X]|]$ as $k \to \infty$ by the Monotone Convergence Theorem [Wil91], for each $d \in \mathcal{D}$ there exists $k(d) \in \mathbb{N}$ such that

$$\mathbb{E}_{X\sim d}\Big[|X - \mathbb{E}[X]| \cdot \mathbb{I}\Big(|X - \mathbb{E}[X]| \geq k\Big)\Big] < \frac{1}{2 \cdot \text{max-coeff}(\boldsymbol{\eta})}.$$

for all $k \geq k(d)$. Define $K = \max_{d\in\mathcal{D}} k(d)$, which is finite as $\mathcal{D}$ is finite.

Next, define the set $U \subseteq \{1, 2, \ldots, \dim(\boldsymbol{\eta})\} \times L$ of pairs of indices of components of $\boldsymbol{\eta}$ and locations in $\mathcal{C}$ as follows:

$$U = \{(j, \ell') \mid \exists \tau \in \mapsto^{\text{unb}} \text{ s.t. } \text{lev}(\tau) = j \text{ and } \ell' \text{ is the target location of } \tau\}.$$

Thus, U is the set of pairs of indices of components of $\boldsymbol{\eta}$ and locations in $\mathcal{C}$ on which the condition *UNBOUND* imposes additional template restrictions.

We now define an LEM $\boldsymbol{\eta}'$ which is of the same dimension as $\boldsymbol{\eta}$, and for each component η'_j we define

$$\eta'_j(\ell, \mathbf{x}) = \begin{cases} 0 & \text{if } (j, \ell) \in U \text{ and } \eta_j(\ell, \mathbf{x}) < -C, \\ 2 \cdot \eta_j(\ell, \mathbf{x}) + 2 \cdot C & \text{otherwise,} \end{cases}$$

where $C > 0$ is a constant to be determined. We claim that there exists $C > 0$ for which $\boldsymbol{\eta}'$ is a piecewise linear GLexRSM map supported by I (with the level map being the same as for $\boldsymbol{\eta}$). The fact that $\boldsymbol{\eta}'$ is piecewise linear for every $C > 0$ is clear from its definition, so we are left to verify that there exists $C > 0$ for which *P-NNEG*$(\boldsymbol{\eta}', \tau)$, *P-RANK*$(\boldsymbol{\eta}', \tau)$ and *EXP-NNEG*$(\boldsymbol{\eta}', \tau)$ hold for each generalized transition τ.

Generalized transitions not in $\mapsto^{\text{unb}}$. We show that, for generalized transitions not in $\mapsto^{\text{unb}}$, the claim holds for every $C > 2 \cdot N \cdot \text{max-coeff}(\boldsymbol{\eta})$.

First, suppose that τ is a generalized transition induced by probabilistic branching. Let ℓ be its source location and ℓ_1, ℓ_2 its target locations (in pCFGs induced by PPs in our syntax, every probabilistic branching has two target locations, see Section 2.2). Since we assume that $\mathcal{C}$ is induced by a program in LinPP* meaning that neither ℓ_1 nor ℓ_2 are target locations of any generalized transition in $\mapsto^{\text{unb}}$, by definition of U and $\boldsymbol{\eta}'$ we must have $\eta'_j(\ell_1, \mathbf{x}) = 2 \cdot \eta_j(\ell_1, \mathbf{x}) + 2 \cdot C$ and $\eta'_j(\ell_2, \mathbf{x}) = 2 \cdot \eta_j(\ell_2, \mathbf{x}) + 2 \cdot C$ for each component η'_j and each $\mathbf{x}$. On the other hand, the piecewise linear transformation

defining $\boldsymbol{\eta}'$ ensures that $\eta'_j(\ell, \mathbf{x}) \geq 2 \cdot \eta_j(\ell, \mathbf{x}) + 2 \cdot C$ for each component η'_j and each $\mathbf{x}$. Hence, it is easy to see that *P-NNEG*$(\boldsymbol{\eta}', \tau)$, *P-RANK*$(\boldsymbol{\eta}', \tau)$ and *EXP-NNEG*$(\boldsymbol{\eta}', \tau)$ all hold as they hold for $\boldsymbol{\eta}$. Note that this proof allows any $C > 0$.

Next, let $\tau \notin \mapsto^{\mathsf{unb}}$ be a generalized transition which is also not a generalized transition induced by probabilistic branching. Denote by ℓ its source location, ℓ_1 its target location, $\mathbf{x} \in I(\ell) \cap G(\tau)$, and $(\ell_1, \mathbf{x}_1)$ some state that is reachable from $(\ell, \mathbf{x})$ by executing τ. We claim that, for each $1 \leq j \leq \mathsf{lev}(\tau)$,

$$\eta_j(\ell_1, \mathbf{x}_1) \geq -2 \cdot N \cdot \mathsf{max\text{-}coeff}(\boldsymbol{\eta}),$$

By definition of $\boldsymbol{\eta}'$, if $C > 2 \cdot N \cdot \mathsf{max\text{-}coeff}(\boldsymbol{\eta})$ this would imply that $\eta'_j(\ell_1, \mathbf{x}_1) = 2 \cdot \eta_j(\ell_1, \mathbf{x}_1) + 2 \cdot C$ for each successor state $(\ell_1, \mathbf{x}_1)$. Since $\eta'_j(\ell, \mathbf{x}) \geq 2 \cdot \eta_j(\ell, \mathbf{x}) + 2 \cdot C$, it is again easy to see that *P-NNEG*$(\boldsymbol{\eta}', \tau)$, *P-RANK*$(\boldsymbol{\eta}', \tau)$ and *EXP-NNEG*$(\boldsymbol{\eta}', \tau)$ all remain true as they are true for $\boldsymbol{\eta}$.

To prove the claim, fix $1 \leq j \leq \mathsf{lev}(\tau)$ and a successor state $(\ell_1, \mathbf{x}_1)$. By the condition *W-EXP-NNEG*$(\boldsymbol{\eta}, \tau)$, we have $\mathsf{min\text{-}pre}^{\tau}_{\eta_j}(\ell, \mathbf{x}) \geq 0$. If τ is not induced by a transition of probabilistic assignment, then we must have $\eta_j(\ell_1, \mathbf{x}_1) \geq \mathsf{min\text{-}pre}^{\tau}_{\eta_j}(\ell, \mathbf{x})$ by definition of min-pre, and the claim follows. Otherwise, suppose that $Up(\tau) = (i, u)$ with $u = d$ a probability distribution of bounded support and $X \sim d$ (recall, we assumed that $\tau \notin \mapsto^{\mathsf{unb}}$). Then, by linearity of η_j we see that

$$\begin{aligned}
\eta_j(\ell_1, \mathbf{x}_1) &= \mathsf{min\text{-}pre}^{\tau}_{\eta_j}(\ell, \mathbf{x}) + \mathsf{coeff}[i] \cdot X - \mathsf{coeff}[i] \cdot \mathbb{E}[X] \\
&\geq \mathsf{coeff}[i] \cdot X - \mathsf{coeff}[i] \cdot \mathbb{E}[X] \\
&\geq -2 \cdot |\mathsf{coeff}[i]| \cdot N \\
&\geq -2 \cdot \mathsf{max\text{-}coeff}(\boldsymbol{\eta}) \cdot N,
\end{aligned}$$

as claimed. The first inequality follows from *W-EXP-NNEG*$(\boldsymbol{\eta}, \tau)$, and the rest follows by definition of N and the assumption that d has bounded support. Here, we used $\mathsf{coeff}[i]$ to denote the coefficient in η_j of the variable with index i at location ℓ_1.

Generalized transitions in $\mapsto^{\mathsf{unb}}$. Let $\tau \in \mapsto^{\mathsf{unb}}$, let ℓ be its source location and ℓ_1 its target location. We claim that each of the conditions *P-NNEG*$(\boldsymbol{\eta}', \tau)$, *P-RANK*$(\boldsymbol{\eta}', \tau)$ and *EXP-NNEG*$(\boldsymbol{\eta}', \tau)$ holds if $C > K \cdot \mathsf{max\text{-}coeff}(\boldsymbol{\eta})$.

1. *P-NNEG*$(\boldsymbol{\eta}', \tau)$: By definition of $\boldsymbol{\eta}'$, for each component η'_j and $\mathbf{x}$ we have $\eta'_j(\ell, \mathbf{x}) \geq 2 \cdot \eta_j(\ell, \mathbf{x}) + C$. Hence, the claim follows since *P-NNEG*$(\boldsymbol{\eta}, \tau)$ holds.
2. *P-RANK*$(\boldsymbol{\eta}', \tau)$: If $\mathbf{x} \in I(\ell) \cap G(\tau)$, we need to show that for $1 \leq j < \mathsf{lev}(\tau)$ we have $\eta'_j(\ell, \mathbf{x}) \geq \mathsf{max\text{-}pre}^{\tau}_{\eta'_j}(\ell, \mathbf{x})$, and that $\eta'_{\mathsf{lev}(\tau)}(\ell, \mathbf{x}) \geq \mathsf{max\text{-}pre}^{\tau}_{\eta'_{\mathsf{lev}(\tau)}}(\ell, \mathbf{x}) + 1$.

First, fix $1 \leq j < \mathsf{lev}(\tau)$. From the *UNBOUND* condition, we know that the coefficient in η_j of the variable updated by τ at ℓ_1 is 0. Hence, η_j has the same value at each successor state of $(\ell, \mathbf{x})$ upon executing τ which is thus equal to $\mathsf{max\text{-}pre}^{\tau}_{\eta_j}(\ell, \mathbf{x})$. By the *W-EXP-NNEG*$(\boldsymbol{\eta}, \tau)$ this value has to be nonnegative. Hence, the value of η'_j is also the same at each successor state and equal to $2 \cdot \mathsf{max\text{-}pre}^{\tau}_{\eta_j}(\ell, \mathbf{x}) + C$. Thus, as $\eta'_j(\ell, \mathbf{x}) \geq 2 \cdot \eta_j(\ell, \mathbf{x}) + C$, the desired inequality holds as *P-RANK*$(\boldsymbol{\eta}, \tau)$ holds.

We now prove that $\eta'_{\mathsf{lev}(\tau)}(\ell, \mathbf{x}) \geq \mathsf{max\text{-}pre}^{\tau}_{\eta'_{\mathsf{lev}(\tau)}}(\ell, \mathbf{x}) + 1$. Let $Up(\tau) = (i, u)$ with $u = d$ a probability distribution of unbounded support and $X \sim d$. Since $(\mathsf{lev}(\tau), \ell_1) \in U$, we have that

$$
\begin{aligned}
&\mathsf{max\text{-}pre}^{\tau}_{\eta'_{\mathsf{lev}(\tau)}}(\ell, \mathbf{x}) = \mathbb{E}_{X \sim d}\Big[\eta'_{\mathsf{lev}(\tau)}(\ell_1, \mathbf{x}[i \leftarrow X])\Big] \\
&= \mathbb{E}_{X \sim d}\Big[(2 \cdot \eta_{\mathsf{lev}(\tau)}(\ell_1, \mathbf{x}[i \leftarrow X]) + 2 \cdot C) \cdot \mathbb{I}\Big(\eta_{\mathsf{lev}(\tau)}(\ell_1, \mathbf{x}[i \leftarrow X]) \geq -C\Big)\Big] \\
&= 2 \cdot C + 2 \cdot \mathsf{max\text{-}pre}^{\tau}_{\eta_{\mathsf{lev}(\tau)}}(\ell, \mathbf{x}) \\
&- \mathbb{E}_{X \sim d}\Big[(2 \cdot \eta_{\mathsf{lev}(\tau)}(\ell_1, \mathbf{x}[i \leftarrow X]) + 2 \cdot C) \cdot \mathbb{I}\Big(\eta_{\mathsf{lev}(\tau)}(\ell_1, \mathbf{x}[i \leftarrow X]) < -C\Big)\Big] \\
&\qquad \text{(use that } \mathsf{max\text{-}pre}^{\tau}_{\eta_{\mathsf{lev}(\tau)}}(\ell, \mathbf{x}) \leq \eta_{\mathsf{lef}(\tau)}(\ell, \mathbf{x}) - 1 \text{ by } \textit{P-RANK}(\boldsymbol{\eta}, \tau)) \\
&\leq 2 \cdot C + 2 \cdot \eta_{\mathsf{lef}(\tau)}(\ell, \mathbf{x}) - 2 \\
&- \mathbb{E}_{X \sim d}\Big[(2 \cdot \eta_{\mathsf{lev}(\tau)}(\ell_1, \mathbf{x}[i \leftarrow X]) + 2 \cdot C) \cdot \mathbb{I}\Big(\eta_{\mathsf{lev}(\tau)}(\ell_1, \mathbf{x}[i \leftarrow X]) < -C\Big)\Big] \\
&\qquad \text{(from definition of } \boldsymbol{\eta}', \text{ have } 2 \cdot C + 2 \cdot \eta_{\mathsf{lef}(\tau)}(\ell, \mathbf{x}) \leq \eta'_{\mathsf{lev}(\tau)}(\ell, \mathbf{x})) \\
&\leq \eta'_{\mathsf{lev}(\tau)}(\ell, \mathbf{x}) - 2 \\
&- \mathbb{E}_{X \sim d}\Big[(2 \cdot \eta_{\mathsf{lev}(\tau)}(\ell_1, \mathbf{x}[i \leftarrow X]) + 2 \cdot C) \cdot \mathbb{I}\Big(\eta_{\mathsf{lev}(\tau)}(\ell_1, \mathbf{x}[i \leftarrow X]) < -C\Big)\Big].
\end{aligned}
$$

Now, $\eta_{\mathsf{lev}(\tau)}(\ell_1, \mathbf{x}[i \leftarrow X]) = \eta_{\mathsf{lev}(\tau)}(\ell_1, \mathbf{x}[i \leftarrow \mathbb{E}[X]) + (X - \mathbb{E}[X]) \cdot \mathsf{coeff}[i] = \mathsf{min\text{-}pre}^{\tau}_{\eta_{\mathsf{lev}(\tau)}}(\ell, \mathbf{x}) + (X - \mathbb{E}[X]) \cdot \mathsf{coeff}[i] \geq (X - \mathbb{E}[X]) \cdot \mathsf{coeff}[i]$, which holds by linearity of $\boldsymbol{\eta}$ and the last inequality follows from *W-EXP-NNEG*$(\boldsymbol{\eta}, \tau)$. Here, we use $\mathsf{coeff}[i]$ to denote the coefficient at ℓ_1 of the variable with index i in $\eta_{\mathsf{lev}(\tau)}$. Hence, continuing the above sequence inequalities we have that (note that the integrand is negative on the set over which integration is performed hence, as we have the minus sign outside the integral, the whole expression increases if we further decrease the integrand but enlarge the event over which the integration is performed in a way which keeps the integrand negative)

$$
\begin{aligned}
&\leq \eta'_{\mathsf{lev}(\tau)}(\ell, \mathbf{x}) - 2 \\
&- \mathbb{E}_{X \sim d}\Big[(2 \cdot (X - \mathbb{E}[X]) \cdot \mathsf{coeff}[i] + 2 \cdot C) \cdot \mathbb{I}\Big((X - \mathbb{E}[X]) \cdot \mathsf{coeff}[i] < -C\Big)\Big].
\end{aligned}
$$

Thus, to conclude that $\eta'_{\mathsf{lev}(\tau)}(\ell, \mathbf{x}) \geq \text{max-pre}^{\tau}_{\eta'_{\mathsf{lev}(\tau)}}(\ell, \mathbf{x}) + 1$ it suffices to show

$$\mathbb{E}_{X\sim d}\Big[(2\cdot(X-\mathbb{E}[X])\cdot \text{coeff}[i] + 2\cdot C)\cdot \mathbb{I}\Big((X-\mathbb{E}[X])\cdot \text{coeff}[i] < -C\Big)\Big] \geq -1.$$

Now observe that, in order for $(X-\mathbb{E}[X])\cdot \text{coeff}[i] < -C$ to hold we must have $X-\mathbb{E}[X]$ and coeff$[i]$ be of opposite signs. Therefore, we have

$$\begin{aligned}
&\mathbb{E}_{X\sim d}\Big[(2\cdot(X-\mathbb{E}[X])\cdot \text{coeff}[i] + 2\cdot C)\cdot \mathbb{I}\Big((X-\mathbb{E}[X])\cdot \text{coeff}[i] < -C\Big)\Big] \\
&\geq \mathbb{E}_{X\sim d}\Big[(-2|X-\mathbb{E}[X]|\cdot \text{max-coeff}(\boldsymbol{\eta}) + 2\cdot C) \\
&\qquad \cdot \mathbb{I}\Big(|X-\mathbb{E}[X]| > C/\text{max-coeff}(\boldsymbol{\eta})\Big)\Big] \\
&= 2\text{max-coeff}(\boldsymbol{\eta})\cdot \mathbb{E}_{X\sim d}\Big[(-|X-\mathbb{E}[X]| + C/\text{max-coeff}(\boldsymbol{\eta})) \\
&\qquad \cdot \mathbb{I}\Big(|X-\mathbb{E}[X]| > C/\text{max-coeff}(\boldsymbol{\eta})\Big)\Big] \\
&\geq 2\text{max-coeff}(\boldsymbol{\eta})\cdot \mathbb{E}_{X\sim d}\Big[(-|X-\mathbb{E}[X]|) \\
&\qquad \cdot \mathbb{I}\Big(|X-\mathbb{E}[X]| > C/\text{max-coeff}(\boldsymbol{\eta})\Big)\Big] \\
&\geq -1,
\end{aligned}$$

where the last inequality holds as $C > K\cdot \text{max-coeff}(\boldsymbol{\eta})$ and by definition of K.

3. *EXP-NNEG*$(\boldsymbol{\eta'}, \tau)$: Let $\mathbf{x} \in I(\ell)\cap G(\tau)$, we show that $\text{min-pre}^{\tau}_{\eta'_j, S^{\leq j-1}_{\mathsf{lev}}}(\ell, \mathbf{x}) \geq 0$ for all $1 \leq j \leq \mathsf{lev}(\tau)$. For $1 \leq j < \mathsf{lev}(\tau)$, by the *UNBOUND* condition we know that, at ℓ_1, the coefficient in η_j of the variable which is updated by τ is 0. Hence, the value of η_j at all successor states of $(\ell, \mathbf{x})$ upon executing τ is the same, and is equal to $\text{min-pre}^{\tau}_{\eta_j}(\ell, \mathbf{x})$ which is nonnegative by *W-EXP-NNEG*$(\boldsymbol{\eta}, \tau)$. Therefore, we must have $\eta'_j(\ell_1, \mathbf{x}_1) = 2\cdot\eta_j(\ell_1, \mathbf{x}_1) + 2\cdot C$ at each state $(\ell_1, \mathbf{x}_1)$ reachable from $(\ell, \mathbf{x})$ by executing τ. Therefore, we also have $\text{min-pre}^{\tau}_{\eta'_j, S^{\leq j-1}_{\mathsf{lev}}}(\ell, \mathbf{x}) = 2\cdot \text{min-pre}^{\tau}_{\eta_j, S^{\leq j-1}_{\mathsf{lev}}}(\ell, \mathbf{x}) + 2\cdot C \geq 0$, where the last inequality holds since *EXP-NNEG*$(\boldsymbol{\eta}, \tau)$.
For the component $\mathsf{lev}(\tau)$, note that $(\mathsf{lev}(\tau), \ell_1) \in U$. Thus, from our definition of η' it follows that $\eta'_{\mathsf{lev}(\tau)}(\ell_1, \mathbf{x}_1) \geq 0$ for every variable valuation $\mathbf{x}_1$. Hence,

$$\text{min-pre}^{\tau}_{\eta'_{\mathsf{lev}(\tau)}, S^{\leq \mathsf{lev}(\tau)-1}_{\mathsf{lev}}}(\ell, \mathbf{x}) \geq 0$$

since it is just an integral of a non-negative function over the set $S^{\leq \mathsf{lev}(\tau)-1}_{\mathsf{lev}}$.

Choice of C. From the analysis of all cases above, we see that

$$C = (2N + K) \cdot \text{max-coeff}(\boldsymbol{\eta}) + 1$$

ensures that $\boldsymbol{\eta}'$ is a piecewise linear GLexRSM map, which proves the lemma claim. □

3.6.4 Proof of Theorem 3.4.4

Theorem. *Algorithm 3.2 decides in polynomial time if a LinPP* $\mathcal{C}$ admits an LEM which satisfies all conditions of Lemma 3.4.3 and which is supported by I. Thus, if the algorithm outputs an LEM, then $\mathcal{C}$ is a.s. terminating and admits a piecewise linear GLexRSM map supported by I.*

Proof. We first prove that the algorithm is sound, i.e. that the LEM $\boldsymbol{\eta}$ which algorithm outputs must satisfy all conditions of Lemma 3.4.3. Let k be the total number of algorithm iterations, so that $\boldsymbol{\eta} = (\eta_1, \ldots, \eta_k)$. Define the level map lev : $\Delta \to \{0, 1 \ldots, k\}$ with the self loop at ℓ_{out} having level 0, and for any other generalized transition τ we define lev(τ) as the index of algorithm iteration in which it was removed from $\mathcal{T}$. The fact that $\boldsymbol{\eta}$ satisfies *P-NNEG*, *P-RANK*, *EXP-NNEG* for generalized transitions induced by probabilistic branching and *W-EXP-NNEG* for all other generalized transitions then easily follows from conditions imposed by the algorithm in each iteration. Furthermore, the way we constructed linear programs $\mathcal{LP}_{\mathcal{T}}^{\text{unb}}$ and $\mathcal{LP}_{\mathcal{T}}^{\tau,\text{unb}}$ for each $\tau \in \mapsto^{\text{unb}} \cap \mathcal{T}$ ensures that $\boldsymbol{\eta}$ satisfies *UNBOUND*. Hence $\boldsymbol{\eta}$ is an LEM supported by I which satisfies all conditions of Lemma 3.4.3.

We now prove completeness, i.e. that for any $(\mathcal{C}, I)$ with $\mathcal{C}$ coming from a program in LinPP*, Algorithm 3.2 decides the existence of an LEM supported by I which satisfies conditions of Lemma 3.4.3. First, observe that for any two LEMs supported by I and which satisfy all the conditions in Lemma 3.4.3, their pointwise sum also satisfies all the conditions in Lemma 3.4.3. Hence, an argument analogous to that in the proof of Theorem 3.4.2 shows that the algorithm finds an LEM satisfying all the conditions of Lemma 3.4.3 whenever one such LEM exists by observing that whenever an LEM exists but $\mathcal{T}$ is non-empty, either $\mathcal{LP}_{\mathcal{T}}^{\text{unb}}$ or $\mathcal{LP}_{\mathcal{T}}^{\tau,\text{unb}}$ for at least one $\tau \in \mapsto^{\text{unb}} \cap \mathcal{T}$ has a solution which 1-ranks at least one new generalized transition.

Note that due to a fixed ordering of generalized transitions in $\mapsto^{\text{unb}} \cap \mathcal{T}$ through which the algorithm iterates, the dimension of the computed LEM which satisfies all the conditions of Lemma 3.4.3 need not be minimal. However, this was not the claim of our theorem. □

CHAPTER 4

Quantitative Termination and Safety Analysis in PPs

This section is based on the following publications:

- Krishnendu Chatterjee, Petr Novotný, Đorđe Žikelić.[†] *Stochastic Invariants for Probabilistic Termination*. In 44th ACM SIGPLAN Symposium on Principles of Programming Languages, **POPL 2017**[‡]
- Krishnendu Chatterjee, Amir Kafshdar Goharshady, Tobias Meggendorfer, Đorđe Žikelić.[†] *Sound and Complete Certificates for Quantitative Termination Analysis of Probabilistic Programs*. In Computer Aided Verification - 34th International Conference, **CAV 2022**

4.1 Introduction

Quantitative termination/reachability and safety analysis. We now turn our attention to quantitative termination/reachability and safety analysis in PPs. Recall, in Section 2.2.3 we showed that reachability analysis can be reduced to termination analysis. Compared to their qualitative counterparts, quantitative analyses in PPs are typically more challenging as they require more fine-grained reasoning about the

[†]Authors ordered alphabetically.

[‡]This work was completed prior to the beginning of the author's PhD studies, however it is a result of an internship project at ISTA that preceeded the PhD studies. Chapter 4 contains results of a part of this publication that introduces stochastic invariants and proves their soundness for quantitative termination analysis. The rest of Chapter 4 is based on [CGMZ22b].

probability of a property being satisfied. As we discuss in Section 4.9, prior to the work presented in this chapter, methods for quantitative reachability and safety analysis in PPs were either not fully automated or could only reason about terminating PP executions and as such are not suitable for PPs that model infinite-time horizon systems.

Our approach – stochastic invariants. We present a framework for quantitative termination and safety analysis in PPs that is *fully automated* and that is applicable to not necessarily a.s. terminating PPs that may model *infinite-time horizon* systems. At the core of our framework lies the notion of stochastic invariants, which we introduce. A *stochastic invariant* is a tuple (SI, p) consisting of a set SI of PP states and an upper-bound p on the probability of a random program run ever leaving SI. Stochastic invariants can be viewed as a stochastic extension of the classical notion of invariants in programs. However, in contrast to invariants which over-approximate the set of all states that no program run can ever leave, stochastic invariants are also annotated with a probability threshold p and over-approximate the set of all states that a random program run can leave with probability at most p. We show that stochastic invariants provide sound proof rules for quantitative termination and safety analyses:

1. *Quantitative termination.* In order to reason about quantitative termination and prove that a PP terminates with probability at least $1 - p$, it suffices to find a stochastic invariant (SI, p) such that, with probability 1, a random program run either terminates or leaves SI. Then, since SI is left with probability at most p, the PP must terminate with probability at least $1 - p$. Hence, the combination of stochastic invariants and probability 1 reachability certificates provides a sound proof rule for quantitative termination analysis in PPs.

2. *Quantitative safety.* In order to reason about quantitative safety and prove that a PP avoids some set of states S with probability at least $1 - p$, it suffices to find a stochastic invariant (SI, p) such that $SI \cap S = \emptyset$. Then, since SI is left with probability at most p, the PP stays within SI and avoids S with probability at least $1 - p$. Hence, stochastic invariants also provide a sound proof rule for quantitative safety analysis in PPs.

Challenges. The above outline provides sound proof rules for proving quantitative termination and safety that are based on stochastic invariants. However, in order to go from these proof rules to effective and fully automated methods for quantitative termination and safety analysis in PPs, we need to solve the following two challenges:

1. *Completeness.* The above outline shows that our proof rules are sound. Moreover, our proof rule for quantitative safety is easily seen to be complete as well. Indeed,

if S is a set of states that a random run in a pCFG $\mathcal{C}$ avoids with probability at least $1-p$, then (SI, p) with $SI = State_{\mathcal{C}} \backslash S$ defines a stochastic invariant with $SI \cap S = \emptyset$. However, it is not clear whether our proof rule for quantitative termination is complete. Completeness is important in order to ensure that our proof rule is general and not only applicable to a small subclass of PPs.

2. *Automation.* In order to go from theoretical proof rules to fully automated quantitative termination and safety analysis, we need to automate the computation of stochastic invariants that satisfy properties required by our proof rules. Note that this is a hard problem. Reasoning about quantitative safety with respect to a given set of states is already a very hard and unexplored problem in infinite state stochastic system analysis. Here, we are required to *compute* a set of states SI while *reasoning about its quantitative safety* probability p.

Contributions. Our contributions in this chapter can be summarized as follows:

1. We introduce *stochastic invariants* in PPs and use them to derive sound and complete proof rules for quantitative termination and safety analysis in PPs.
2. The definitions of stochastic invariants and our proof rules alone do not provide automated methods for quantitative termination and safety analysis in PPs. In order to enable automation of stochastic invariant computation, we introduce a martingale-based certificate function for stochastic invariants that we call *stochastic invariant indicators (SI-indicators)*. We prove that SI-indicators provide a sound and complete characterization of stochastic invariants.
3. We present a *constraint-solving based algorithm* for fully automatically computing SI-indicators in PPs with affine or polynomial arithmetic. Our algorithm is *relatively complete*, meaning that it is guaranteed to compute an affine/polynomial SI-indicator in a certain subclass of affine/polynomial PPs whenever a polynomial SI-indicator exist (we formally define this subclass of affine/polynomial PPs in later sections). Due to SI-indicators providing a sound and complete characterization of stochastic invariants, we also obtain *relatively complete algorithms* for quantitative termination and safety analysis in PPs.
4. We implement a prototype of our approach that uses SI-indicators to compute stochastic invariants and perform quantitative termination analysis in PPs. We demonstrate the applicability of our approach on various PP benchmarks.

The definition of stochastic invariants and their soundness for quantitative termination analysis (part of contribution point 1) are based on our results in [CNZ17]. The completeness of stochastic invariant based proof rule for quantitative termination

analysis (part of contribution point 1), definition of SI-indicators and their soundness and completeness for characterizing stochastic invariants (contribution point 2), the relatively complete algorithm for SI-indicator computation and quantitative termination analysis in PPs (contribution point 3) and experimental results (contribution point 4) are based on our results in [CGMZ22b]. While both of these publications focused on quantitative termination analysis, the applicability of stochastic invariants to quantitative safety analysis follows immediately from our results in [CNZ17, CGMZ22b] and we expand this thesis with their straightforward extension to quantitative safety analysis in order to highlight this. In order to follow the organization of [CNZ17, CGMZ22b], we first present our results on quantitative termination analysis in PPs. Then, towards the end of this chapter, we generalize our results to quantitative safety analysis in PPs.

Chapter organization. Section 4.2 provides an overview of our framework for quantitative termination analysis in PPs. In Section 4.3 we formally define stochastic invariants, present our proof rule for quantitative termination analysis in PPs and prove its soundness and completeness. In Section 4.4, we then present a martingale-based certificate called stochastic invariant indicators (SI-indicators) for characterizing stochastic invariants and show that they provide sound and complete characterization of stochastic invariants. Section 4.5 defines ranking supermartingales (RSMs), a formal certificate for probability 1 termination, and combines them with stochastic invariants towards instantiating our proof rule for quantitative termination analysis. Section 4.6 presents our fully automated algorithm for quantitative termination analysis. Section 4.7 presents our experimental results. Section 4.8 shows that our results easily extend to quantitative safety. Section 4.9 discusses related work. Finally, Section 4.10 contains full proofs of results presented in earlier sections that are deferred to this section in order to enhance readability.

Repulsing supermartingales. Prior to introducing SI-indicators in [CGMZ22b], we first introduced a *repulsing supermartingale (RepSM)* [CNZ17] as a sound but incomplete martingale-based characterization of stochastic invariants. While RepSMs provide an effective approach to automating the synthesis of stochastic invariants in PPs, their incompleteness was shown in [TOUH21, Example 6.6]. Our SI-indicators overcome this limitation of RepSMs and provide a sound and complete characterization of stochastic invariants. To that end, in this chapter we only present SI-indicators in order to keep our presentation focused on a single characterization and algorithm for computing stochastic invariants. However, we note that the introduction of repulsing supermartingales is another important contribution of our work [CNZ17].

```
        x = 0
ℓ_init: while x ≥ 0 do
ℓ_1:        r_1 := Uniform([−1, 0.5])
ℓ_2:        x := x + r_1
ℓ_3:        if x ≥ 100 then
ℓ_4:            if ⋆ then
ℓ_5:                r_2 := Uniform([−1, 2])
ℓ_6:                x := x + r_2
ℓ_out:
```

Figure 4.1: Our running example for this chapter. Whenever r_1 and r_2 are evaluated, their values are sampled from the uniform distributions $\mathit{Uniform}([-1, 0.5])$ and $\mathit{Uniform}([-1, 2])$, independently from previous samples. The command **if** $\star$ **then** denotes non-deterministic branching. The labels ℓ_{init}, $\ell_1, \ldots, \ell_6$ and ℓ_{out} denote locations in the program's pCFG. The pCFG for this PP was presented in Figure 2.3.

4.2 Overview of Our Approach

We now illustrate our approach to quantitative termination analysis on an example PP shown in Figure 4.1. The program models a 1-dimensional discrete-time random walk over the real line that starts at $x = 0$ and terminates once a point with $x < 0$ is reached. In every time step, x is incremented by a random value sampled according to the uniform distribution $\mathit{Uniform}([-1, 0.5])$. However, if the stochastic process is in a point with $x \geq 100$, then the value of x might also be incremented by a random value independently sampled from $\mathit{Uniform}([-1, 2])$. The choice on whether the second increment happens is non-deterministic. By a standard random walk argument, the program does not terminate almost-surely.

Outline of our approach Let $p = 0.01$. To prove this program terminates with probability at least $1 - p = 0.99$, our method computes the following two objects:

1. *Stochastic invariant.* A stochastic invariant is a tuple (SI, p) s.t. SI is a set of program states that a random program run leaves with probability at most p.
2. *Termination proof for the stochastic invariant.* A *probability* 1 *termination certificate* is computed in order to prove that the program will, with probability 1, either terminate or leave the set SI. Since SI is left with probability at most p, the program must terminate with probability at least $1 - p$.

In Section 4.3, we formally define this proof rule and prove that it is sound and complete for quantitative termination analysis in PPs.

Synthesizing a stochastic invariant The definition of stochastic invariants and the formulation of our proof rule do not provide insight into how to automate stochastic invariant computation. In order to allow for a fully automated quantitative termination analysis in PPs, in Section 4.4 we propose a martingale-based characterization of stochastic invaraints in the form of stochastic invariant indicators. A *stochastic invariant indicator (SI-indicator)* is a state function f which assigns a non-negative real value to each reachable program state and which serves the following two purposes: First, exactly those states which are assigned a value strictly less than 1 are considered a part of the stochastic invariant SI. Second, the value assigned to each state is an upper-bound on the probability of leaving SI if the program starts from that state. Finally, by requiring that the value of the SI-indicator at the initial state of the program is at most p, we ensure a random program run leaves the stochastic invariant with probability at most p. In Section 4.4, we will define SI-indicators in terms of conditions that ensure the properties above and facilitate automated computation. We also show that SI-indicators serve as a *sound and complete* characterization of stochastic invariants, which is one of the core contributions of this chapter. The significance of completeness of the characterization is that, in order to search for a stochastic invariant with a given probability threshold p, one may equivalently search for an SI-indicator with the same probability threshold whose computation can be automated. The previous approach in [CNZ17] to the synthesis of stochastic invariants was neither complete nor provided tight probability bounds. For Figure 4.1, we have the following set SI which will be left with probability at most $p = 0.01$:

$$SI(\ell) = \begin{cases} (x < 99) & \text{if } \ell \in \{\ell_{init}, \ell_1, \ell_2, \ell_3, \ell_{out}\} \\ \text{false} & \text{otherwise.} \end{cases} \tag{4.1}$$

An SI-indicator for this stochastic invariant is:

$$f(\ell, x, r_1, r_2) = \begin{cases} \frac{x+1}{100} & \text{if } \ell \in \{\ell_{init}, \ell_1, \ell_3, \ell_{out}\} \text{ and } x < 99 \\ \frac{x+1+r_1}{100} & \text{if } \ell = \ell_2 \text{ and } x < 99 \\ 1 & \text{otherwise.} \end{cases} \tag{4.2}$$

It is easy to check that $(SI, 0.01)$ is a stochastic invariant and that for every state $s = (\ell, x, r_1, r_2)$, the value $f(s)$ is an upper-bound on the probability of eventually leaving SI if program execution starts at s. Also, $s \in SI \Leftrightarrow f(s) < 1$.

Synthesizing a termination proof To prove that a PP terminates with probability at least $1-p$, our method searches for a stochastic invariant (SI, p) for which, additionally, a random program run with probability 1 either leaves SI or terminates. In order to impose this additional condition, our method simultaneously computes a ranking supermartingale for the set of states $\neg SI \cup State_{term}$, where $State_{term}$ is the set of all

terminal states. Recall, ranking supermartingales are a classical certificate for proving almost-sure termination or reachability in PPs. Formally, a state function η is said to be a *ranking supermartingale (RSM)* [CS13] for $\neg SI \cup State_{term}$ if it satisfies the following two conditions:

- *Non-negativity.* $\eta(\ell, x, r_1, r_2) \geq 0$ for any reachable state $(\ell, x, r_1, r_2) \in SI$;
- ε*-decrease in expectation.* There exists $\varepsilon > 0$ such that, for any reachable non-terminal state $(\ell, x, r_1, r_2) \in SI$, the value of η decreases in expectation by at least ε after a one-step execution of the program from (ℓ, x, r_1, r_2).

The existence of an RSM for $\neg SI \cup State_{term}$ implies that the program will, with probability 1, either terminate or leave SI. As (SI, p) is a stochastic invariant, we can readily conclude that the program terminates with probability at least $1 - p = 0.99$. An example RSM with $\varepsilon = 0.05$ for our example above is:

$$\eta(\ell, x, r_1, r_2) = \begin{cases} x + 1.1 & \text{if } \ell = \ell_{init} \\ x + 1.05 & \text{if } \ell = \ell_1 \\ x + 1.2 + r_1 & \text{if } \ell = \ell_2 \\ x + 1.15 & \text{if } \ell = \ell_3 \\ x + 1 & \text{if } \ell = \ell_{out} \\ 100 & \text{otherwise.} \end{cases} \tag{4.3}$$

We formally define RSMs in Section 4.5. We also show that, by using RSMs as a probability 1 reachability certificate in our sound and complete proof rule for quantitative termination analysis, we are able to derive tight lower bound on termination probability for a PP that was demonstrated in [TOUH21] not to admit a repulsing supermartingale [CNZ17] or a gamma-scaled supermartingale [TOUH21].

Simultaneous synthesis Our method employs a template-based approach in order to automate the synthesis of the SI-indicator and the RSM in affine/polynomial PPs. The SI-indicator and the RSM are synthesized *simultaneously*. We assume that our method is provided with an affine/polynomial invariant I which over-approximates the set of all reachable states in the program, which is necessary since the defining conditions of SI-indicators and RSMs are required to hold at all reachable program states. Note that invariant generation is an orthogonal and well-studied problem and can be automated using [CFGG20]. For both the SI-indicator and the RSM, our method first fixes a symbolic template affine/polynomial expression for each location in the program. Then, all the defining conditions of SI-indicators and RSMs are encoded as a system of constraints over the symbolic template variables, where reachability of

program states is encoded using the invariant I, and the synthesis proceeds by solving this system of constraints. We describe our algorithm in Section 4.6, and show that it is *relatively complete* with respect to the provided invariant I and the probability threshold $1 - p$. On the other hand, we note that our algorithm can also be adapted to *compute* lower bounds on the termination probability by combining it with a binary search on p.

Completeness vs relative completeness Our characterization of stochastic invariants using indicator functions is complete. So is our reduction from quantitative termination analysis to the problem of synthesizing an SI-indicator function and a certificate for almost-sure reachability. These are our core theoretical contributions in this work. Nevertheless, as shown in [FC19], RSMs are complete only for finite termination, not a.s. termination. Moreover, template-based approaches lead to completeness guarantees only for solutions that match the template, e.g. polynomial termination certificates of a bounded degree. Therefore, our end-to-end approach is only relatively complete. These losses of completeness are due to Rice's undecidability theorem and inevitable even in *qualitative* termination analysis. In this chapter, we successfully provide approaches for *quantitative* termination analysis that are as complete as the best known methods for the qualitative case.

4.3 Stochastic Invariants and a Proof Rule for Quantitative Termination

In this section, we introduce the notion of stochastic invariants and use them to present a sound and complete proof rule for quantitative termination analysis in PPs. In what follows, we fix a pCFG $\mathcal{C} = (L, V, \ell_{init}, \mathbf{x}_{init}, \mapsto, G, Pr, Up)$. Recall from Section 2.2 that a *predicate function* in $\mathcal{C}$ is a map F that to every location $\ell \in L$ assigns a logical formula $F(\ell)$ over program variables. It naturally induces a set of states, which we require to be Borel-measurable for the semantics to be well-defined and which we identify with the predicate function F by a slight abuse of notation. We use $\neg F$ to denote the negation of a predicate function, i.e. $(\neg F)(\ell) = \neg F(\ell)$.

An *invariant* in $\mathcal{C}$ is a predicate function I which additionally over-approximates the set of reachable states in $\mathcal{C}$, i.e. for every $(\ell, \mathbf{x}) \in \mathrm{Reach}_{\mathcal{C}}$ we have $\mathbf{x} \models I(\ell)$. *Stochastic invariants* can be viewed as a probabilistic extension of invariants, which a random program run leaves only with a certain probability. An example of a stochastic invariant for the PP in Fig. 4.1 is shown in eq. (4.1).

Defintion 4.3.1 (Stochastic invariant)**.** *Let SI a predicate function in $\mathcal{C}$ and $p \in [0, 1]$ a probability. The tuple (SI, p) is a* stochastic invariant (SI) *if the probability of a run*

in $\mathcal{C}$ leaving the set of states defined by SI is at most p under any scheduler. Formally, we require that

$$\sup_\sigma \mathbb{P}^\sigma\Big[\rho \in Run_{\mathcal{C}} \mid \rho \text{ reaches some } (\ell, \mathbf{x}) \text{ with } \mathbf{x} \not\models SI(\ell)\Big] \leq p.$$

We show that stochastic invariants in combination with an almost-sure termination certificate for PPs yield a sound and complete proof rule for quantitative termination analysis. Note that Theorem 4.3.2 below states a general result about termination probabilities that is agnostic to the termination certificate.

Theorem 4.3.2 (Soundness and Completeness of SIs for Quantitative Termination)**.** *Let $\mathcal{C} = (L, V, \ell_{init}, \mathbf{x}_{init}, \mapsto, G, Pr, Up)$ be a pCFG and (SI, p) a stochastic invariant in $\mathcal{C}$. Suppose that, with respect to every scheduler, a run in $\mathcal{C}$ almost-surely either terminates or reaches a state in $\neg SI$, i.e.*

$$\inf_\sigma \mathbb{P}^\sigma\Big[Term \cup \mathsf{Reach}(\neg SI)\Big] = 1. \tag{4.4}$$

Then $\mathcal{C}$ terminates with probability at least $1 - p$. Conversely, if $\mathcal{C}$ terminates with probability at least $1 - p$, then there exists a stochastic invariant (SI, p) in $\mathcal{C}$ such that, with respect to every scheduler, a run in $\mathcal{C}$ almost-surely either terminates or reaches a state in $\neg SI$.

Proof sketch, full proof in Section 4.10.1. The first part (soundness) follows directly from the definition of SI and (4.4). The completeness proof is conceptually and technically involved and presented in Section 4.10.1. In short, the central idea is to construct, for every n greater than a specific threshold n_0, a stochastic invariant $(SI_n, p + \frac{1}{n})$ such that a run almost-surely either terminates or exists SI_n. Then, we show that $\cap_{n=n_0}^{\infty} SI_n$ is our desired SI. To construct each SI_n, we consider the infimum termination probability at every state $(\ell, \mathbf{x})$ and call it $r(\ell, \mathbf{x})$. The infimum is taken over all schedulers. We then let SI_n be the set of states $(\ell, \mathbf{x})$ for whom $r(\ell, \mathbf{x})$ is greater than a specific threshold α. Intuitively, our stochastic invariant is the set of program states from which the probability of termination is at least α, no matter how the non-determinism is resolved. Let us call these states likely-terminating. The intuition is that a random run of the program will terminate or eventually leave the likely-terminating states with high probability. □

4.4 Stochastic Invariant Characterization via SI-indicators

In the previous section, we defined stochastic invariants and showed that they yield a sound and complete proof rule for quantitative termination analysis in PPs. However,

the definition of stochastic invariants and the formulation of our proof rule do not provide insight into how to automate stochastic invariant computation. In order to allow for a fully automated quantitative termination analysis in PPs, we now propose a sound and complete martingale-based characterization of stochastic invaraints through the novel notion of *stochastic invariant indicators (SI-indicators)*.

Intuitively, an SI-indicator is a function that to each state assigns an upper-bound on the probability of violating the stochastic invariant if we start the program in that state. Since the definition of an SI-indicator imposes conditions on its value at reachable states and since computing the exact set of reachable states is in general infeasible, we define SI-indicators with respect to a supporting invariant with the later automation in mind. In order to understand the ideas of this section, one may assume for simplicity that the invariant exactly equals the set of reachable states. A *state-function* in $\mathcal{C}$ is a function f that to each location $\ell \in L$ assigns a Borel-measurable real-valued function over program variables $f(\ell) : \mathbb{R}^{|V|} \to \mathbb{R}$. We use $f(\ell, \mathbf{x})$ and $f(\ell)(\mathbf{x})$ interchangeably.

Defintion 4.4.1 (Stochastic invariant indicator)**.** *A tuple (f_{SI}, p) comprising a state function f_{SI} and probability $p \in [0, 1]$ is a* stochastic invariant indicator (SI-indicator) *with respect to an invariant I, if it satisfies the following conditions:*

(C_1) Non-negativity. *For every location $\ell \in L$, we have $\mathbf{x} \models I(\ell) \Rightarrow f_{SI}(\ell, \mathbf{x}) \geq 0$.*

(C_2) Non-increasing expected value. *For every location $\ell \in L$, we have:*

(C_2^1) *If $\ell \in L_C$, then for any transition $\tau = (\ell, \ell')$ we have*

$$\mathbf{x} \models I(\ell) \wedge G(\tau) \Rightarrow f_{SI}(\ell, \mathbf{x}) \geq f_{SI}(\ell', \mathbf{x}).$$

(C_2^2) *If $\ell \in L_P$, then*

$$\mathbf{x} \models I(\ell) \Rightarrow f_{SI}(\ell, \mathbf{x}) \geq \sum_{\tau=(\ell,\ell') \in \mapsto} \Pr(\tau) \cdot f_{SI}(\ell', \mathbf{x}).$$

(C_2^3) *If $\ell \in L_N$, then*

$$\mathbf{x} \models I(\ell) \Rightarrow f_{SI}(\ell, \mathbf{x}) \geq \max_{\tau=(\ell,\ell') \in \mapsto} f_{SI}(\ell', \mathbf{x}).$$

(C_2^4) *If $\ell \in L_A$ with $\tau = (\ell, \ell')$ the unique outgoing transition from ℓ, then:*

- *If $Up(\tau) = (j, \bot)$, we have*

$$\mathbf{x} \models I(\ell) \Rightarrow f(\ell, \mathbf{x}) \geq f(\ell', \mathbf{x}).$$

- *If $Up(\tau) = (j, u)$ with $u : \mathbb{R}^{|V|} \to \mathbb{R}$ an expression, we have*

$$\mathbf{x} \models I(\ell) \Rightarrow f(\ell, \mathbf{x}) \geq f(\ell', \mathbf{x}[x_j \leftarrow u(\mathbf{x}_i)]).$$

- *If* $Up(\tau) = (j, u)$ *with* $u = d$ *a distribution, we have*

$$\mathbf{x} \models I(\ell) \Rightarrow f(\ell, \mathbf{x}) \geq \mathbb{E}_{X \sim d}[f(\ell', \mathbf{x}[x_j \leftarrow X])].$$

- *If* $Up(\tau) = (j, u)$ *with* $u = [a, b]$ *an interval, we have*

$$\mathbf{x} \models I(\ell) \Rightarrow f(\ell, \mathbf{x}) \geq \sup_{X \in [a,b]} \{f(\ell', \mathbf{x}[x_j \leftarrow X])\}.$$

(C_3) Initial condition. *We have* $f(\ell_{init}, \mathbf{x}_{init}) \leq p$.

Intuition behind SI-indicators. (C_1) imposes that f is nonnegative at any state contained in the invariant I. Next, for any state in I, (C_2) imposes that the value of f does not increase in expectation upon a one-step execution of the pCFG under any scheduler. Finally, the condition (C_3) imposes that the initial value of f in $\mathcal{C}$ is at most p. Together, the indicator thus intuitively over-approximates the probability of violating SI. An example of an SI-indicator for our running example in Figure 3.2 is given in (4.2). The following theorem formalizes the above intuition and is our main result of this section. In essence, we prove that (SI, p) is a stochastic invariant in $\mathcal{C}$ iff there exists an SI-indicator (f_{SI}, p) such that SI contains all states at which f_{SI} is strictly smaller than 1. This implies that, for every stochastic invariant (SI, p), there exists an SI-indicator such that (SI', p) defined via $SI'(\ell) = (\mathbf{x} \models I(\ell) \wedge f_{SI}(\ell, \mathbf{x}) < 1)$ is a stochastic invariant that is at least as tight as (SI, p). Eq. (4.2) shows an example SI-indicator for the PP in Fig. 4.1 and the stochastic invariant in eq. (4.1).

The following theorem is the main result of this section and it formally shows that SI-indicators provide a sound and complete characterization of stochastic invariants.

Theorem 4.4.2 (Soundness and Completeness of SI-indicators)**.** *Let* $\mathcal{C}$ *be a pCFG,* I *an invariant in* $\mathcal{C}$ *and* $p \in [0, 1]$. *For any SI-indicator* (f_{SI}, p) *with respect to* I, *the predicate map* SI *defined as*

$$SI(\ell) = (\mathbf{x} \models I(\ell) \wedge f_{SI}(\ell, \mathbf{x}) < 1)$$

yields a stochastic invariant (SI, p) *in* $\mathcal{C}$. *Conversely, for every stochastic invariant* (SI, p) *in* $\mathcal{C}$, *there exist an invariant* I_{SI} *and a state function* f_{SI} *such that* (f_{SI}, p) *is an SI-indicator with respect to* I_{SI} *and for each* $\ell \in L$ *we have*

$$SI(\ell) \supseteq (\mathbf{x} \models I_{SI}(\ell) \wedge f_{SI}(\ell, \mathbf{x}) < 1).$$

Proof sketch, full proof in Section 4.10.2. Since the proof is technically involved, we present the main ideas here and defer the details to Section 4.10.2.

First, suppose that I is an invariant in $\mathcal{C}$ and that (f_{SI}, p) is an SI-indicator with respect to I, and let $SI(\ell) = (\mathbf{x} \models I(\ell) \wedge f_{SI}(\ell, \mathbf{x}) < 1)$ for each $\ell \in L$. We

need to show that (SI, p) is a stochastic invariant in $\mathcal{C}$. Let $\sup_\sigma \mathbb{P}^\sigma_{(\ell,\mathbf{x})}[\text{Reach}(\neg SI)]$ be a state function that maps each state $(\ell, \mathbf{x})$ to the probability of reaching $\neg SI$ from $(\ell, \mathbf{x})$. We consider a lattice of non-negative semi-analytic state-functions $(\mathcal{L}, \sqsubseteq)$ with the partial order defined via $f \sqsubseteq f'$ if $f(\ell, \mathbf{x}) \leq f'(\ell, \mathbf{x})$ holds for each state $(\ell, \mathbf{x})$ in I. See Section 2.5 for a review of lattice theory. It follows from a result in [TOUH21] that the probability of reaching $\neg SI$ can be characterized as the least fixed point of the *next-time operator* $\mathbb{X}_{\neg SI} : \mathcal{L} \to \mathcal{L}$. Away from $\neg SI$, the operator $\mathbb{X}_{\neg SI}$ simulates a one-step execution of $\mathcal{C}$ and maps $f \in \mathcal{L}$ to its maximal expected value upon one-step execution of $\mathcal{C}$ where the maximum is taken over all schedulers, and at states contained in $\neg SI$ the operator $\mathbb{X}_{\neg SI}$ is equal to 1. It was also shown in [TOUH21] that, if a state function $f \in \mathcal{L}$ is a pre-fixed point of $\mathbb{X}_{\neg SI}$, then it satisfies $\sup_\sigma \mathbb{P}^\sigma_{(\ell,\mathbf{x})}[\text{Reach}(\neg SI)] \leq f(\ell, \mathbf{x})$ for each $(\ell, \mathbf{x})$ in I. Now, by checking the defining properties of pre-fixed points and recalling that f_{SI} satisfies Non-negativity condition (C_1) and Non-increasing expected value condition (C_2) in Definition 4.4.1, we can show that f_{SI} is contained in the lattice $\mathcal{L}$ and is a pre-fixed point of $\mathbb{X}_{\neg SI}$. It follows that $\sup_\sigma \mathbb{P}^\sigma_{(\ell_{init},\mathbf{x}_{init})}[\text{Reach}(\neg SI)] \leq f_{SI}(\ell_{init}, \mathbf{x}_{init})$. On the other hand, by initial condition (C_3) in Definition 4.4.1 we know that $f_{SI}(\ell_{init}, \mathbf{x}_{init}) \leq p$. Hence, we have $\sup_\sigma \mathbb{P}^\sigma_{(\ell_{init},\mathbf{x}_{init})}[\text{Reach}(\neg SI)] \leq p$ so (SI, p) is a stochastic invariant.

Conversely, suppose that (SI, p) is a stochastic invariant in $\mathcal{C}$. We show in Section 4.10.2 that, if we define I_{SI} to be the trivial true invariant and define $f_{SI}(\ell, \mathbf{x}) = \sup_\sigma \mathbb{P}^\sigma_{(\ell,\mathbf{x})}[\text{Reach}(\neg SI)]$, then (f_{SI}, p) forms an SI-indicator with respect to I_{SI}. The claim follows by again using the fact that f_{SI} is the least fixed point of the operator $\mathbb{X}_{\neg SI}$, from which we can conclude that (f_{SI}, p) satisfies conditions (C_1) and (C_2) in Definition 4.4.1. On the other hand, the fact that (SI, p) is a stochastic invariant and our choice of f_{SI} imply that (f_{SI}, p) satisfies the initial condition (C_3) in Definition 4.4.1. Hence, (f_{SI}, p) forms an SI-indicator with respect to I_{SI}. Furthermore, $SI(\ell) \supseteq (\mathbf{x} \models I_{SI}(\ell) \wedge f_{SI}(\ell, \mathbf{x}) < 1)$ follows since $1 > f_{SI}(\ell, \mathbf{x}) = \sup_\sigma \mathbb{P}^\sigma_{(\ell,\mathbf{x})}[\text{Reach}(\neg SI)]$ implies that $(\ell, \mathbf{x})$ cannot be contained in $\neg SI$ so $\mathbf{x} \models SI(\ell)$. This concludes the proof. □

Based on the theorem above, in order to compute a stochastic invariant in $\mathcal{C}$ for a given probability threshold p, it suffices to synthesize a state function f_{SI} that together with p satisfies all the defining conditions in Definition 4.4.1 with respect to some supporting invariant I, and then consider a predicate function SI defined via $SI(\ell) = (\mathbf{x} \models I(\ell) \wedge f_{SI}(\ell, \mathbf{x}) < 1)$ for each $\ell \in L$. This will be the guiding principle of our algorithmic approach in Section 4.6.

Intuition on characterization. Stochastic invariants can essentially be thought of as quantitative safety specifications in probabilistic programs – (SI, p) is a stochastic invariant if and only if a random PP run leaves SI with probability at most p. However, what makes their computation hard is that they do not consider probabilities of staying

within a specified safe set. Rather, the computation of stochastic invariants requires computing *both* the safe set *and* the certificate that it is left with at most the given probability. Nevertheless, in order to reason about them, we may consider SI as an implicitly defined safe set. Hence, if we impose conditions on a state function f_{SI} to be an upper bound on the reachability probability for the target set of states $(\mathbf{x} \models I(\ell) \wedge f_{SI}(\ell, \mathbf{x}) < 1)$, and in addition impose that $f_{SI}(\ell_{init}, \mathbf{x}_{init}) \leq p$, then these together will entail that p is an upper bound on the probability of ever leaving SI when starting in the initial state. This is the intuitive idea behind our construction of SI-indicators, as well as our soundness and completeness proof. In the proof, we show that conditions (C_1) and (C_2) in Definition 4.4.1 indeed entail the necessary conditions to be an upper bound on the reachability probability of the set $(\mathbf{x} \models I(\ell) \wedge f_{SI}(\ell, \mathbf{x}) < 1)$.

4.5 Stochastic Invariants and RSMs for Quantitative Termination

Theorem 4.3.2 provides us with a recipe for computing lower bounds on the probability of termination once we are able to compute stochastic invariants: if (SI, p) is a stochastic invariant in a pCFG $\mathcal{C}$, it suffices to prove that the set of states $State_{term} \cup \neg SI$ is reached almost-surely with respect to any scheduler in $\mathcal{C}$, i.e. the program terminates or violates SI. Note that this is simply a qualitative a.s. termination problem, except that the set of terminal states is now augmented with $\neg SI$. Then, since (SI, p) is a stochastic invariant, it would follow that a terminal state is reached with probability at least $1 - p$. Moreover, the theorem shows that this approach is both sound and complete. In other words, proving quantitative termination, i.e. that we reach $State_{term}$ with probability at least $1 - p$ is now reduced to (i) finding a stochastic invariant (SI, p) and (ii) proving that the program $\mathcal{C}'$ obtained by adding $\neg SI$ to the set of terminal states of $\mathcal{C}$ is a.s. terminating. Note that, to preserve completeness, (i) and (ii) should be achieved in tandem, i.e. an approach that first synthesizes and fixes SI and then tries to prove a.s. termination for $\neg SI$ is not complete.

Ranking supermartingales. While our reduction above is agnostic to the type of proof/certificate that is used to establish a.s. termination, in this work we use Ranking Supermartingales (RSMs) [CS13], which are a standard and classical certificate for proving a.s. termination and reachability. Let $\mathcal{C} = (L, V, \ell_{init}, \mathbf{x}_{init}, \mapsto, G, Pr, Up)$ be a pCFG and I an invariant in $\mathcal{C}$. Note that as in Definition 4.4.1, the main purpose of the invariant is to allow for automated synthesis and one can again simply assume it to equal the set of reachable states. An ε-RSM for a subset T of states is a state function that is non-negative in each state in I, and whose expected value decreases by at least $\varepsilon > 0$ upon a one-step execution of $\mathcal{C}$ in any state that is not contained in the

target set T. Thus, intuitively, a program run has an expected tendency to approach the target set T where the distance to T is given by the value of the RSM which is required to be non-negative in all states in I. The ε-ranked expected value condition is formally captured via the next-time operator $\mathbb{X}$ (see Section 4.10.2). An example of an RSM for our running example in Figure 4.1 and the target set of states $\neg SI \cup State_{term}$ with SI the stochastic invariant in Equation (4.1) is given in Equation (4.3).

Defintion 4.5.1 (Ranking supermartingales)**.** *Let T be a predicate function defining a set of target states in $\mathcal{C}$, and let $\varepsilon > 0$. A state function η is said to be an* ε-ranking supermartingale (ε-RSM) *for T with respect to the invariant I if it satisfies the following conditions:*

- Non-negativity. *For each location $\ell \in L$ and $\mathbf{x} \in I(\ell)$, we have $\eta(\ell, \mathbf{x}) \geq 0$.*
- ε-ranked expected value. *For each location $\ell \in L$ and $\mathbf{x} \models I(\ell) \cap \neg T(\ell)$, we have $\eta(\ell, \mathbf{x}) \geq \mathbb{X}(\eta)(\ell, \mathbf{x}) + \varepsilon$.*

Note that the second condition can be expanded according to location types in the exact same manner as in condition C_2 of Definition 4.4.1. The only difference is that in Definition 4.4.1, the expected value had to be non-increasing, whereas here it has to decrease by ε. It is well-known that the two conditions above entail that T is reached with probability 1 with respect to any scheduler [CS13, CFNH18].

Theorem 4.5.2 (Proof in Section 4.10.4)**.** *Let $\mathcal{C}$ be a pCFG, I an invariant in $\mathcal{C}$ and T a predicate function defining a target set of states. If there exist $\varepsilon > 0$ and an ε-RSM for T with respect to I, then T is a.s. reached under any scheduler, i.e.*

$$\inf_\sigma \mathbb{P}^\sigma_{(\ell_{init}, \mathbf{x}_{init})}\Big[\mathsf{Reach}(T)\Big] = 1.$$

The following theorem is an immediate corollary of Theorems 4.3.2 and 4.5.2.

Theorem 4.5.3. *Let $\mathcal{C}$ be a pCFG and I be an invariant in $\mathcal{C}$. Suppose that there exist a stochastic invariant (SI, p), an $\varepsilon > 0$ and an ε-RSM η for $State_{term} \cup \neg SI$ with respect to I. Then $\mathcal{C}$ terminates with probability at least $1 - p$.*

Therefore, in order to prove that $\mathcal{C}$ terminates with probability at least $1 - p$, it suffices to find (i) a stochastic invariant (SI, p) in $\mathcal{C}$, and (ii) an ε-RSM η for $State_{term} \cup \neg SI$ with respect to I and some $\varepsilon > 0$. Note that these two tasks are interdependent. We cannot simply choose any stochastic invariant. For instance, the trivial predicate function defined via $SI = \text{true}$ always yields a valid stochastic invariant for any $p \in [0, 1]$, but it does not help termination analysis. Instead, we need to compute a stochastic invariant and an RSM for it *simultaneously*.

$$
\begin{array}{lll}
 & x = \textbf{ndet}((0,1)) & \\
\ell_{init}: & \textbf{while } x < 1 \textbf{ do} & \{0 < x < 2\} \\
\ell_1: & \quad x := 2 \cdot x & \{0 < x < 1\} \\
\ell_2: & \quad \textbf{if prob}(0.5) \textbf{ then} & \{1 \leq x < 2\} \\
\ell_3: & \quad\quad \textbf{while true do skip od} & \{1 \leq x < 2\} \\
\ell_{out}: & & \{1 \leq x < 2\}
\end{array}
$$

Figure 4.2: A program that was shown in [TOUH21] not to admit a repulsing supermartingale [CNZ17] or a gamma-scaled supermartingale [TOUH21], but for which our method can certify the tight lower-bound of 0.5 on the probability of termination.

Power of completeness. We end this section by showing that our approach certifies a tight lower-bound on termination probability for a program that was proven in [TOUH21] not to admit any of the previously-existing certificates for lower bounds on termination probability. This shows that our completeness pays off in practice and our approach is able to handle programs that were beyond the reach of previous methods. Consider the program in Figure 4.2 annotated by an invariant I. We show that our approach certifies that this program terminates with probability at least 0.5. Indeed, consider a stochastic invariant $(SI, 0.5)$ with

$$
SI(\ell) = \begin{cases} \text{true} & \text{if } \ell \in \{\ell_{init}, \ell_1, \ell_2, \ell_{out}\} \\ \text{false} & \text{if } \ell = \ell_3 \end{cases}
$$

and a state function defined via

$$
\eta(\ell_{init}, x) = \begin{cases} -\log(x) + \log(2) + 3 & \text{if } \ell = \ell_{init} \\ -\log(x) + \log(2) + 2 & \text{if } \ell = \ell_1 \\ 1 & \text{if } \ell = \ell_2 \\ 0 & \text{if } \ell \in \{\ell_3, \ell_{out}\} \end{cases}
$$

for each x. Then one can check by inspection that $(SI, 0.5)$ is a stochastic invariant and η is a $(\log(2) - 1)$-RSM for $State_{term} \cup \neg SI$ with respect to I. Therefore, it follows by Theorem 4.5.3 that the PP in Figure 4.2 terminates with probability at least 0.5.

4.6 Algorithm for Quantitative Termination

We now provide template-based relatively complete algorithms for simultaneous and automated synthesis of SI-indicators and RSMs, in order to solve the quantitative termination problem over pCFGs with affine/polynomial arithmetic. Our approach builds upon the ideas of [ACF+21, CFG16] for qualitative and non-probabilistic cases.

Input and assumptions. The input to our algorithms consists of a pCFG $\mathcal{C}$ together with a probability $p \in [0, 1]$, an invariant I,* and technical variables δ and M, which specify polynomial template sizes used by the algorithm and which will be discussed later. In this section, we limit our focus to affine/polynomial pCFGs, i.e. we assume that all guards $G(\tau)$ in $\mathcal{C}$ and all invariants $I(\ell)$ are conjunctions of affine/polynomial inequalities over program variables. Similarly, we assume that every update function $u : \mathbb{R}^{|V|} \to \mathbb{R}$ used in deterministic variable assignments is an affine/polynomial expression in $\mathbb{R}[V]$.

Output. The goal of our algorithms is to synthesize a tuple (f, η, ε) where f is an SI-indicator function, η is a corresponding RSM, and $\varepsilon > 0$, such that:

- At every location ℓ of $\mathcal{C}$, both $f(\ell)$ and $\eta(\ell)$ are affine/polynomial expressions of fixed degree δ over the program variables V.
- Having $SI(\ell) := \{\mathbf{x} \mid f(\ell, \mathbf{x}) < 1\}$, the pair (SI, p) is a valid stochastic invariant and η is an ε-RSM for $State_{term} \cup \neg SI$ with respect to I.

As shown in Sections 4.4 and 4.5, such a tuple $w = (f, \eta, \varepsilon)$ serves as a certificate that the probabilistic program modeled by $\mathcal{C}$ terminates with probability at least $1 - p$. We call w a quantitative termination certificate.

Overview. Our algorithm is a standard template-based approach similar to [ACF+21, CFG16]. We encode the requirements of Definitions 4.4.1 and 4.5.1 as entailments between affine/polynomial inequalities with unknown coefficients and then apply the classical Farkas' Lemma [Far02] or Putinar's Positivstellensatz [Put93] to reduce the synthesis problem to Quadratic Programming (QP). Finally, we solve the resulting QP using a numerical optimizer or an SMT-solver. Our approach consists of the four steps below. Step 3 follows [ACF+21] exactly. Hence, we refer to [ACF+21] for more details on this step.

Step 1. Setting up templates. The algorithm sets up symbolic templates with unknown coefficients for f, η and ε.

- First, for each location ℓ of $\mathcal{C}$, the algorithm sets up a template for $f(\ell)$ which is a polynomial consisting of all possible monomials of degree at most δ over program variables, each appearing with an unknown coefficient. For example, consider the program in Figure 4.1 of Section 4.2. This program has three variables: x, r_1 and

*We assume an invariant is given as part of the input. Invariant generation is an orthogonal and well-studied problem and can be automated using [CSS03, CFGG20].

r_2. If $\delta = 1$, i.e. if the goal is to find an affine SI-indicator, at every location ℓ_i of the program, the algorithm sets $f(\ell_i, x, r_1, r_2) := \widehat{c_{i,0}} + \widehat{c_{i,1}} \cdot x + \widehat{c_{i,2}} \cdot r_1 + \widehat{c_{i,3}} \cdot r_2$. Similarly, if the desired degree is $\delta = 2$, the algorithm symbolically computes: $f(\ell_i, x, r_1, r_2) := \widehat{c_{i,0}} + \widehat{c_{i,1}} \cdot x + \widehat{c_{i,2}} \cdot r_1 + \widehat{c_{i,3}} \cdot r_2 + \widehat{c_{i,4}} \cdot x^2 + \widehat{c_{i,5}} \cdot x \cdot r_1 + \widehat{c_{i,6}} \cdot x \cdot r_2 + \widehat{c_{i,7}} \cdot r_1^2 + \widehat{c_{i,8}} \cdot r_1 \cdot r_2 + \widehat{c_{i,9}} \cdot r_2^2$. Note that every monomial of degree at most 2 appears in this expression. The goal is to synthesize suitable real values for each unknown coefficient $\widehat{c_{i,j}}$ such that f becomes an SI-indicator. Throughout this section, we use the $\widehat{\cdot}$ notation to denote an unknown coefficient whose value will be synthesized by our algorithm.

- The algorithm creates an unknown variable $\widehat{\varepsilon}$ whose final value will serve as ε.
- Finally, at each location ℓ of $\mathcal{C}$, the algorithm sets up a template for $\eta(\ell)$ in the exact same manner as the template for $f(\ell)$. The goal is to synthesize values for $\widehat{\varepsilon}$ and the $\widehat{c}$ variables in this template such that η becomes a valid $\widehat{\varepsilon}$-RSM for $State_{term} \cup \neg SI$ with respect to I.

Step 2. Generating entailment constraints. In this step, the algorithm symbolically computes the requirements of Definition 4.4.1, i.e. C_1–C_3, and their analogues in Definition 4.5.1 using the templates generated in the previous step. Note that all of these requirements are entailments between affine/polynomial inequalities over program variables whose coefficients are unknown. In other words, they are of the form $\forall \mathbf{x} \;\; A(\mathbf{x}) \Rightarrow b(\mathbf{x})$ where A is a set of affine/polynomial inequalities over program variables whose coefficients contain the unknown variables $\widehat{c}$ and $\widehat{\varepsilon}$ generated in the previous step and b is a single such inequality. For example, for the program of Figure 3.2, the algorithm symbolically computes condition C_1 at line ℓ_1 as follows: $\forall \mathbf{x} \;\; I(\ell_1, \mathbf{x}) \Rightarrow f(\ell_1, \mathbf{x}) \geq 0$. Assuming that the given invariant is $I(\ell_1, \mathbf{x}) := (x \leq 1)$ and an affine (degree 1) template was generated in the previous step, the algorithm expands this to:

$$\forall \mathbf{x} \;\; 1 - \mathbf{x} \geq 0 \Rightarrow \widehat{c_{1,0}} + \widehat{c_{1,1}} \cdot x + \widehat{c_{1,2}} \cdot r_1 + \widehat{c_{1,3}} \cdot r_2 \geq 0. \tag{4.5}$$

The algorithm generates similar entailment constraints for every location and every requirement in Definitions 4.4.1 and 4.5.1.

Step 3. Quantifier elimination. At the end of the previous step, we have a system of constraints of the form $\bigwedge_i (\forall \mathbf{x} \;\; A_i(\mathbf{x}) \Rightarrow b_i(\mathbf{x}))$. In this step, the algorithm sets off to eliminate the universal quantification over $\mathbf{x}$ in every constraint. First, consider the affine case. If A_i is a set of linear inequalities over program variables and b_i is one such linear inequality, then the algorithm attempts to write b_i as a linear combination with non-negative coefficients of the inequalities in A_i and the trivial inequality $1 \geq 0$. For example, it rewrites (4.5) as $\widehat{\lambda_1} \cdot (1 - x) + \widehat{\lambda_2} = \widehat{c_{1,0}} + \widehat{c_{1,1}} \cdot x + \widehat{c_{1,2}} \cdot r_1 + \widehat{c_{1,3}} \cdot r_2$

where $\widehat{\lambda_i}$'s are new *non-negative* unknown variables for which we need to synthesize non-negative real values. This inequality should hold for all valuations of program variables. Thus, we can equate the corresponding coefficients on both sides and obtain this equivalent system:

$$\begin{array}{cc} \widehat{\lambda_1} + \widehat{\lambda_2} = \widehat{c_{1,0}} & \text{(the constant factor)} \\ -\widehat{\lambda_1} = \widehat{c_{1,1}} & \text{(coefficient of } x\text{)} \\ 0 = \widehat{c_{1,2}} = \widehat{c_{1,3}} & \text{(coefficients of } r_1 \text{ and } r_2\text{)} \end{array} \tag{4.6}$$

This transformation is clearly sound, but it is also complete due to the well-known Farkas' lemma [Far02]. Now consider the polynomial case. Again, we write b_i as a combination of the polynomials in A_i. The only difference is that instead of having non-negative real coefficients, we use sum-of-square polynomials as our multiplicands. For example, suppose our constraint is

$$\forall \mathbf{x} \;\; g_1(\mathbf{x}) \geq 0 \wedge g_2(\mathbf{x}) \geq 0 \Rightarrow g_3(\mathbf{x}) > 0,$$

where the g_i's are polynomials with unknown coefficients. The algorithm writes

$$g_3(\mathbf{x}) = h_0(\mathbf{x}) + h_1(\mathbf{x}) \cdot g_1(\mathbf{x}) + h_2(\mathbf{x}) \cdot g_2(\mathbf{x}), \tag{4.7}$$

where each h_i is a sum-of-square polynomial of degree at most M. The algorithm sets up a template of degree M for each h_i and adds well-known quadratic constraints that enforce it to be a sum of squares. See [ACF+21, Page 22] for details. It then expands (4.7) and equates the corresponding coefficients of the LHS and RHS as in the linear case. The soundness of this transformation is trivial since each h_i is a sum-of-squares and hence always non-negative. Completeness follows from Putinar's Positivstellensatz [Put93]. Since the arguments for completeness of this method are exactly the same as the method in [ACF+21], we refer the reader to [ACF+21] for more details and an extension to entailments between strict polynomial inequalities.

Step 4. Quadratic programming. All of our constraints are converted to Quadratic Programming (QP) over template variables, e.g. see (4.6). Our algorithm passes this QP instance to an SMT solver or a numerical optimizer. If a solution is found, it plugs in the values obtained for the $\widehat{c}$ and $\widehat{\varepsilon}$ variables back into the template of Step 1 and outputs the resulting termination witness (f, η, ε).

We end this section by noting that our algorithm is sound and relatively complete for synthesizing affine/polynomial quantitative termination certificates.

Theorem 4.6.1 (Soundness and Completeness in the Affine Case)**.** *Given an affine pCFG $\mathcal{C}$, an affine invariant I, and a non-termination upper-bound $p \in [0, 1]$, if $\mathcal{C}$ admits a quantitative termination certificate $w = (f, \eta, \varepsilon)$ in which both f and η are*

affine expressions at every location, then w corresponds to a solution of the QP instance solved in Step 4 of the algorithm above. Conversely, every such solution, when plugged back into the template of Step 1, leads to an affine quantitative termination certificate showing that $\mathcal{C}$ terminates with probability at least $1-p$ over every scheduler.

Theorem 4.6.2 (Soundness and Relative Completeness in the Polynomial Case)**.** *Given a polynomial pCFG $\mathcal{C}$, a polynomial invariant I which is a compact subset of $\mathbb{R}^{|V|}$ at every location ℓ, and a non-termination upper-bound $p \in [0,1]$, if $\mathcal{C}$ admits a quantitative termination certificate $w = (f, \eta, \varepsilon)$ in which both f and η are polynomial expressions of degree at most δ at every location, then there exists an $M \in \mathbb{N}$, for which w corresponds to a solution of the QP instance solved in Step 4 of the algorithm above. Conversely, every such solution, when plugged back into the template of Step 1, leads to a polynomial quantitative termination certificate of degree at most δ showing that $\mathcal{C}$ terminates with probability at least $1-p$ over every scheduler.*

Proof. Step 2 encodes the conditions of an SI-indicator (Definition 4.4.1) and RSM (Definition 4.5.1). Theorem 4.5.3 shows that an SI-indicator together with an RSM is a valid quantitative termination certificate. The transformation in Step 3 is sound and complete as argued in [ACF+21, Theorems 4 and 10][†]. The affine version relies on Farkas' lemma [Far02] and is complete with no additional constraints. The polynomial version is based on Putinar's Positivstellensatz [Put93] and is only complete for large enough M, i.e. a high-enough degree for sum-of-square multiplicands. This is why we call our algorithm *relatively* complete. In practice, small values of M are enough to synthesize w and we use $M = 2$ in all of our experiments. □

4.7 Experiments

Implementation. We implemented a prototype of our approach in Python and used SymPy [M+17] for symbolic computations and the MathSAT5 SMT Solver [CGSS13] for solving the final QP instances. We also applied basic optimizations, e.g. checking the validity of each entailment and thus removing tautological constraints.

Machine and parameters. All results were obtained on an Intel Core i9-10885H machine (8 cores, 2.4 GHz, 16 MB Cache) with 32 GB of RAM running Ubuntu 20.04. We always synthesized quadratic termination certificates and set $\delta = M = 2$.

Benchmarks. We generated a variety of random walks with complicated behavior, including nested combinations of probabilistic and non-deterministic branching and

[†]We need a more involved transformation for *strict* inequalities. See [ACF+21, Theorem 8].

Table 4.1: Summary of our experimental results on a subset of our benchmark set. See [CGMZ22a][Appendix J] for benchmark details and for the results on all benchmarks. For each benchmark, the column with the value $1-p$ denotes the lower bound on termination probability that our method proved.

Benchmark	Short Explanation	p	$1-p$	Runtime (s)
Figure 4.1	Our running example	0.01	0.99	2.38
Figure 7	Nested probabilistic and non-deterministic branches leading to infinite loop with maximum probability	0.25	0.75	1.40
Figure 9	An a.s. terminating biased random walk with uniformly distributed steps	0	1	0.73
Figure 10	A random walk that starts at $x = 10$ and takes a step of $Uniform(-2, 1)$ each time. Terminates if $x < 0$ and loops forever as soon as $x \geq 100$.	0.12	0.88	1.10
Figure 11	A 2-D random walk starting at $(50, 50)$. In each iteration, x is incremented, while y is increased by $Uniform(-1, 1)$. Terminates when $x > 100$. Loops when $y \leq 0$.	0.07	0.93	3.52
Figure 14	A 3-D random walk. In each iteration, each of x, y, z are incremented with a higher probability than decremented. Terminates when $x + y + z < 0$.	0.999	0.001	3.22
Figure 15	An example with both probabilistic and non-deterministic in an assignment.	0.51	0.49	2.73
Figure 16	A variant of Figure 15 with unbounded non-determinism in an assignment.	0.51	0.49	2.70
Figure 17	A probabilistic branch between an a.s. terminating loop and a loop with small termination probability.	0.4	0.6	5.17
Figure 18	A skewed random walk with two barriers, only one of which leads to program termination.	0.51	0.49	5.26
Figure 19	Taken from [CNZ17] and conceptually similar to Figure 5.	0.24	0.76	0.94
Figure 22	A more complicated and non-a.s.-terminating random walk taken from [CNZ17].	0.1	0.9	1.15
Figure 23	A 2-D variant of Figure 22, also from [CNZ17].	0.08	0.92	4.01

loops. We also took a number of benchmarks from [CNZ17]. Table 4.1 presents experimental results on a subset of our benchmark set, together with short descriptions of these benchmarks. Complete evaluation as well as details on all benchmarks are provided in the extended version of the paper [CGMZ22a][Appendix J].

Results and discussion. Our experimental results are summarized in Table 4.1, with complete results provided in [CGMZ22a][Appendix J]. In every case, our approach was able to synthesize a certificate that the program terminates with probability at least $1-p$ under any scheduler. Moreover, our runtimes are consistently small and less than 6 seconds per benchmark. Our approach was able to handle programs that are beyond the reach of previous methods, including those with unbounded differences and unbounded non-deterministic assignments to which approaches such as [CNZ17] and [TOUH21] are not applicable, as was demonstrated in [TOUH21]. This adds experimental confirmation to our theoretical power-of-completeness result at the end of Section 4.5, which showed the wider applicability of our method. Finally, it is noteworthy that the termination probability lower-bounds reported in Table 4.1 are not tight. There are two reasons for

this. First, while our theoretical approach is sound and complete, our algorithm can only synthesize affine/polynomial certificates for quantitative termination, and the best polynomial certificate of a certain degree might not be tight. Second, we rely on an SMT-solver to solve our QP instances. The QP instances often become harder as we decrease p, leading to the solver's failure even though the constraints are satisfiable.

4.8 Extension to Quantitative Safety

We conclude this chapter by showing that all our results naturally extend to quantitative safety analysis in PPs. In particular, we first show that stochastic invariants trivially yield a sound and complete proof rule for quantitative safety analysis and demonstrate it on an example. We then, we show that our relatively complete algorithm in Section 4.6 straightforwardly extends to a relatively complete algorithm for quantitative safety analysis in affine/polynomial PPs.

Recall, given a pCFG $\mathcal{C} = (L, V, \ell_{init}, \mathbf{x}_{init}, \mapsto, G, Pr, Up)$ and a predicate function S in $\mathcal{C}$ that defines a set of states in $\mathcal{C}$, the safety probability of S in $\mathcal{C}$ is defined as the infimum probability of a random run not reaching S, i.e. $\inf_\sigma \mathbb{P}^\sigma[\mathsf{Safe}(S)]$. The goal of quantitative safety analysis is then to prove that the safety probability of S in $\mathcal{C}$ is above some probability threshold. See Section 2.2 for formal definitions of reachability and safety probabilities.

4.8.1 Stochastic Invariants for Quantitative Safety

The following theorem shows that stochastic invariants yield a sound and complete proof rule for quantitative safety analysis in PPs. The proof rule for quantitative safety is based on the fact that stochastic invariants encode an upper bound on the safety probability with respect to the complement of a set of states. Intuitively, given a predicate function S, the safety probability of S in $\mathcal{C}$ is at least $1 - p$ if and only if there exists a stochastic invariant (SI, p) in $\mathcal{C}$ such that $S \cap SI = \emptyset$, when S and SI are regarded as sets of states in $\mathcal{C}$.

Theorem 4.8.1 (Soundness and Completeness of SIs for Quantitative Safety)**.** *Let $\mathcal{C} = (L, V, \ell_{init}, \mathbf{x}_{init}, \mapsto, G, Pr, Up)$ be a pCFG, S a predicate function in $\mathcal{C}$ and (SI, p) a stochastic invariant in $\mathcal{C}$. Suppose that there exists no state $(\ell, \mathbf{x})$ in $\mathcal{C}$ such that $\mathbf{x} \models S(\ell)$ and $\mathbf{x} \models SI(\ell)$. Then, the safety probability of S in $\mathcal{C}$ is greater than or equal to $1 - p$, i.e.*

$$\inf_\sigma \mathbb{P}^\sigma[\mathsf{Safe}(S)] \geq 1 - p.$$

Conversely, if the safety probability of S in $\mathcal{C}$ is greater than or equal to $1 - p$, then there exists a stochastic invariant (SI, p) in $\mathcal{C}$ for which there exists no state $(\ell, \mathbf{x})$ in $\mathcal{C}$ such that $\mathbf{x} \models S(\ell)$ and $\mathbf{x} \models SI(\ell)$.

Proof. To prove the first part (soundness), observe that

$$\inf_{\sigma} \mathbb{P}^{\sigma}[\mathsf{Safe}(S)] \geq \inf_{\sigma} \mathbb{P}^{\sigma}[\mathsf{Safe}(\neg SI)] = 1 - \sup_{\sigma} \mathbb{P}^{\sigma}[\mathsf{Reach}(\neg SI)] \geq 1 - p.$$

Here, the first inequality follows since $S \cap SI = \emptyset$ as sets of states so $S \subseteq \neg SI$, whereas the last inequality follows by the definition of stochastic invariants.

To prove the second part (completeness), suppose that the safety probability of S in $\mathcal{C}$ is greater than or equal to $1 - p$. Consider a predicate function $SI = \neg S$. Clearly, we have that there exists no state $(\ell, \mathbf{x})$ in $\mathcal{C}$ such that $\mathbf{x} \models S(\ell)$ and $\mathbf{x} \models SI(\ell)$. Hence, to prove the theorem claim, it suffices to prove that (SI, p) is a stochastic invariant. This follows since

$$\sup_{\sigma} \mathbb{P}^{\sigma}[\mathsf{Reach}(\neg SI)] = 1 - \inf_{\sigma} \mathbb{P}^{\sigma}[\mathsf{Safe}(\neg SI)] \leq 1 - \inf_{\sigma} \mathbb{P}^{\sigma}[\mathsf{Safe}(S)] \leq p.$$

Here, the first inequality follows since $S \cap SI = \emptyset$ so $S \subseteq \neg SI$, whereas the last inequality follows since the safety probability of S is greater than or equal to $1 - p$. □

Example 4.8.1. *Consider again our running example for this chapter in Fig. 4.1 and a predicate function defined via*

$$S(\ell) = \begin{cases} (x \geq 100) & \text{if } \ell \in \{\ell_{init}, \ell_1, \ell_2, \ell_3, \ell_{out}\} \\ true & \text{otherwise.} \end{cases}$$

*Intuitively, S captures the set of states that could be reached upon executing the **if**-branch in line ℓ_3. Consider also the predicate function SI defined in eq. (4.1). In Section 4.2, we showed that $(SI, 0.01)$ is a stochastic invariant. On the other hand, comparing S to SI, we can see that $S \cap SI = \emptyset$ as sets. Therefore, our proof rule in Theorem 4.8.1 implies that the safety probability of S is greater than or equal to $1 - 0.01 = 0.99$.*

4.8.2 Relatively Complete Algorithm for Quantitative Safety

We now show that our algorithm in Section 4.6 can be straightforwardly adapted to a relatively complete algorithm for quantitative safety analysis in PPs. The algorithm is very similar to the algorithm in Section 4.6, thus in what follows we only present a high level overview and highlight the differences compared to our algorithm in Section 4.6.

Input. As before, the algorithm considers affine/polynomial PPs and takes as input a pCFG $\mathcal{C}$, a probability $p \in [0, 1]$, an affine/polynomial invariant I and technical variables δ and M that have the same purpose as in Section 4.6. In addition, the algorithm also takes as input a predicate function S in $\mathcal{C}$ whose quantitative safety we wish to analyze, which we require to be specified via a conjunction of affine/polynomial inequalities over program variables.

Output. The algorithm synthesizes an SI-indicator f for which we require that

- if we define a predicate function SI via $SI(\ell) := \{\mathbf{x} \mid f(\ell, \mathbf{x}) < 1\}$ for each location ℓ in $\mathcal{C}$, the pair (SI, p) is a valid stochastic invariant, and
- there exists no state $(\ell, \mathbf{x})$ such that $\mathbf{x} \models S(\ell)$ and $\mathbf{x} \models SI(\ell)$.

The first condition will be enforced analogously as in the algorithm in Section 4.6. To enforce the latter condition, it suffices to enforce that $f(\ell, \mathbf{x}) \geq 1$ for each $\mathbf{x} \models S(\ell)$. We call $w = (f)$ a quantitative safety certificate for S.

Algorithm. Our algorithm for quantitative safety analysis proceeds in the analogous four steps as the algorithm in Section 4.6. In what follows, we summarize those steps and highlight the differences compared to the algorithm in Section 4.6.

Step 1. Setting up templates. Analogously as in Section 4.6, for each location ℓ of $\mathcal{C}$, our algorithm sets up a template for $f(\ell)$ which is a polynomial consisting of all possible monomials of degree at most δ over program variables, each appearing with an unknown coefficient.

Step 2. Generating entailment constraints. The algorithm symbolically computes the requirements for f to be an SI-indicator, i.e. conditions $C_1 - C_3$ in Definition 4.4.1. In addition, it symbolically computes the following entailment for each location ℓ in $\mathcal{C}$:

$$\forall \mathbf{x} \;\; \mathbf{x} \models S(\ell) \Rightarrow f(\ell, \mathbf{x}) \geq 1.$$

These entailments encode that, if a state is contained in S, then the value of f at the state is greater than or equal to 1. Thus, we have that $S \cap SI = \emptyset$ for the predicate function SI defined via $SI(\ell) := \{\mathbf{x} \mid f(\ell, \mathbf{x}) < 1\}$ for each location ℓ in $\mathcal{C}$. The generation of these entailments is done analogously as in Step 2 in Section 4.6.

Step 3. Quantifier elimination. This step proceeds analogously as the Step 3 in Section 4.6. In the affine case, quantifier elimination uses Farkas' lemma [Far02] and in the polynomial case it uses Putinar's Positivstellensatz [Put93], which provide soundness and completeness guarantees analogously as in Section 4.6.

Step 4. Quadratic programming. The resulting constraints are converted to Quadratic Programming (QP) over template variables and the resulting QP instance is passed to an SMT solver or a numerical optimizer. If a solution is found, the algorithm plugs in the obtained values back into the template of Step 1 and outputs the resulting safety witness $w = (f)$.

Our algorithm for quantitative safety analysis is sound and relatively complete for synthesizing affine/polynomial quantitative termination certificates, analogously to our algorithm for quantitative termination analysis in Section 4.6.

Theorem 4.8.2 (Soundness and Completeness in the Affine Case)**.** *Given an affine pCFG $\mathcal{C}$, an affine invariant I, an affine predicate function S and an upper bound $p \in [0,1]$ on the non-safety probability of S, if $\mathcal{C}$ admits a quantitative safety certificate $w = (f)$ in which f is specified via an affine expression at every location, then w corresponds to a solution of the QP instance solved in Step 4 of the algorithm above. Conversely, every such solution, when plugged back into the template of Step 1, leads to an affine quantitative safety certificate showing that the safety probability of S in $\mathcal{C}$ is at least $1 - p$ over every scheduler.*

Theorem 4.8.3 (Soundness and Relative Completeness in the Polynomial Case)**.** *Given a polynomial pCFG $\mathcal{C}$, a polynomial invariant I which is a compact subset of $\mathbb{R}^{|V|}$ at every location ℓ, a polynomial predicate function S and an upper bound $p \in [0,1]$ on the non-safety probability of S, if $\mathcal{C}$ admits a quantitative safety certificate $w = (f)$ in which f is specified via a polynomial expression of degree at most δ at every location, then there exists an $M \in \mathbb{N}$, for which w corresponds to a solution of the QP instance solved in Step 4 of the algorithm above. Conversely, every such solution, when plugged back into the template of Step 1, leads to a polynomial quantitative safety certificate of degree at most δ showing that the safety probability of S in $\mathcal{C}$ is at least $1 - p$ over every scheduler.*

Proof. Step 2 encodes the conditions of an SI-indicator (Definition 4.4.1) and and in addition encodes that f is at least 1 at every state contained in S. Theorem 4.8.1 shows that an SI-indicator whose indicated stochastic invariant does not intersect S is a valid quantitative safety certificate. The transformation in Step 3 is sound and complete as argued in [ACF+21, Theorems 4 and 10][‡]. The affine version relies on Farkas' lemma [Far02] and is complete with no additional constraints. The polynomial version is based on Putinar's Positivstellensatz [Put93] and is only complete for large enough M, i.e. a high-enough degree for sum-of-square multiplicands. This is why we call our algorithm *relatively* complete. □

4.9 Related Work

Quantitative termination and safety. For this chapter to be self contained, we repeat our overview of related work in Chapter 1. Prior to the work in this thesis, automated methods for quantitative termination and safety analysis in PPs derived

[‡]We need a more involved transformation for *strict* inequalities. See [ACF+21, Theorem 8].

bounds on termination or safety probability by considering terminating executions. As such, they are not well suited for the formal analysis of PPs that may model infinite-time horizon systems. For PPs with bounded loops, exact inference methods perform weighted model counting [HdBM20] or symbolic execution and integration [GMV16, GSV20] in order to compute exact expected values of functions or probabilities of events upon PP execution. Approximate inference methods include [CMMV16, CDM17, HDM22]. The work [SCG13] considers linear arithmetic PPs and uses symbolic execution and integration to compute bounds on reachability and safety probability. To the best of our knowledge, existing exact and approximate inference methods do not support PPs with non-determinism. Abstract interpretation for quantitative safety analysis in PPs was proposed in [Mon00], however it remains unclear how to perform the widening operation in the presence of loops and automation is only briefly discussed. Automated methods for the analysis of prob-solvable loops, i.e. loops whose body contains a sequence of probabilistic assignments but no conditional branching or nested loops, that are based on recurrence solving have been proposed in [BKS19, MBKK21a, MBKK21b].

Following the publication of our work [CNZ17], several other approaches to quantitative termination and safety analysis have been proposed, of which some are applicable to non almost-surely terminating PPs [TOUH21, WSF$^+$21, BO21]. In what follows, we compare the work in this chapter to these works in more detail. In particular, none of these works provide a sound and complete proof rule for quantitative termination analysis in PPs, and to the best of our knowledge our work [CGMZ22b] provides the only such proof rule whose computation can be automated.

Comparison to [TOUH21]. The work [TOUH21] proposed novel martingale-based proof rules for quantitative termination and safety analysis in PPs and algorithms for their automated computation. For quantitative safety analysis, it introduces nonnegative repulsing supermartingales (NNRepSMs), that relax some of the restrictive conditions of RepSMs of [CNZ17] and provide a sound and complete proof rule for quantitative safety analysis. For quantitative termination analysis, it introduces γ-scaled submartingales (γ-SclSubMs) and proves their effectiveness for computing lower bounds on termination probability. Intuitively, for $\gamma \in (0, 1)$, a state function f is a γ-SclSubM if it is a bounded nonnegative function whose value in each non-terminal state decreases in expected value at least by a factor of γ upon a one-step execution of the pCFG. One may think of the second condition as a multiplicative decrease in expected value. However, this condition is too strict and γ-SclSubMs are not complete for lower bounds on termination probability [TOUH21, Example 6.6]. Unlike our approaches in [CNZ17, CGMZ22b] which combine martingale-based certificates and stochastic invariants to reason about quantitative termination, γ-SclSubMs aim to reason about quantitative termination directly and are not complete.

Comparison to [WSF$^+$21]. The work [WSF$^+$21] uses martingale-based methods and presents two approaches to quantitative safety analysis in PPs. First, it is shown that RepSMs provide a tighter upper bound on the probability of reaching a set of states compared to the bound proved in [CNZ17]. Second, they present a sound and complete proof rule for quantitative safety analysis in PPs. This proof rule is equivalent to NNRepSMs of [TOUH21], however the algorithm for synthesizing them is fundamentally different and allows template-based synthesis of bounds with exponential templates. This work also provides a martingale-based method for quantitative reachability analysis, however the method is only applicable to *almost-surely terminating* PPs.

Comparison to [BO21]. The work of [BO21] proposes a type system for functional PPs that allows incrementally searching for type derivations and accumulating a lower bound on termination probability. In the limit, it finds arbitrarily tight lower bounds on termination probability, however it does not provide any completeness or precision guarantees in finite time and it only reasons about terminating executions.

Supermartingale-based methods for other PP analyses. Martingale-based methods for proving probability 1 termination/reachability were studied extensively. Since we discussed these works in detail in Chapter 3, we omit the repetition. Martingale-based methods were also used to for the formal analysis of probability 1 recurrence and persistance [CVS16], cost [NCH18, WFG$^+$19] and sensitivity [WFC$^+$20] analysis.

Other approaches. Logical calculi for reasoning about various properties of PPs (including termination and safety) were studied in [Koz85, FH82, Fel84] and extended to programs with non-determinism in [MM05, KKMO18, OKKM16, KMMM10]. These works consider proof systems for PPs based on the weakest pre-expectation calculus. The expressiveness of this calculus allows reasoning about very complex programs, but the proofs typically require human input. In contrast, we aim for a fully automated approach for probabilistic programs with polynomial arithmetic. Connections between martingales and the weakest pre-expectation calculus were studied in [HKGK20].

Cores in MDPs. *Cores* are a conceptually equivalent notion to stochastic invariants introduced in [KM20] for finite state MDPs. [KM20] presents a sampling-based algorithm for their computation. The questions related to the computational complexity of computing cores in MDPs were further studied in [ACG$^+$22].

4.10 Technical Proofs

4.10.1 Proof of Theorem 4.3.2

Theorem (Soundness and Completeness of SIs for Quantitative Termination)**.** *Let $\mathcal{C} = (L, V, \ell_{init}, \mathbf{x}_{init}, \mapsto, G, Pr, Up)$ be a pCFG and (SI, p) a stochastic invariant in $\mathcal{C}$. Suppose that, with respect to every scheduler, a run in $\mathcal{C}$ almost-surely either terminates or reaches a state in $\neg SI$, i.e.*

$$\inf_\sigma \mathbb{P}^\sigma\Big[\mathit{Term} \cup \mathsf{Reach}(\neg SI)\Big] = 1. \tag{4.8}$$

Then $\mathcal{C}$ terminates with probability at least $1-p$. Conversely, if $\mathcal{C}$ terminates with probability at least $1-p$, then there exists a stochastic invariant (SI, p) in $\mathcal{C}$ such that, with respect to every scheduler, a run in $\mathcal{C}$ almost-surely either terminates or reaches a state in $\neg SI$.

Proof. The first part (soundness) follows directly from the definition of SI and (4.8). In what follows, we present the completeness proof. Suppose that $p \in [0,1]$ and that $\mathcal{C}$ is a pCFG that terminates with probability at least $1-p$. We need to prove that there exists a stochastic invariant (SI, p) in $\mathcal{C}$, such that a run in $\mathcal{C}$ with respect to every scheduler almost-surely reaches either some terminal state or a state in $\neg SI$.

If $p = 1$, then letting $SI(\ell) = \mathbb{R}^{|V|}$ for each location ℓ in $\mathcal{C}$ and V the set of variables in $\mathcal{C}$ trivially satisfies the theorem claim. Otherwise, let $n_0 \in \mathbb{N}$ be the smallest natural number such that $p + \frac{1}{n_0} < 1$. To show that there exists a stochastic invariant (SI, p) with the desired property, we construct for each $n \geq n_0$ a stochastic invariant $(SI_n, p + \frac{1}{n})$ such that a run in $\mathcal{C}$ with respect to every scheduler almost-surely reaches either some terminal state or a state in $\neg SI_n$. We then show that, by taking all of the constructed stochastic invariants and defining $SI(\ell) := \cap_{n=n_0}^{\infty} SI_n(\ell)$ for each location ℓ in $\mathcal{C}$, the tuple (SI, p) defines a stochastic invariant such that a run in $\mathcal{C}$ with respect to every scheduler almost-surely reaches either some terminal state or a state in $\neg SI$, as desired. We will explain in the construction and proof of desired properties for $(SI_n, p + \frac{1}{n})$ why we need to impose that $n \geq n_0$.

Construction of $(SI_n, p + \frac{1}{n})$. Let $\mathit{State}_\mathcal{C}$ denote the set of all states in $\mathcal{C}$. For every state $(\ell, \mathbf{x}) \in \mathit{State}_\mathcal{C}$, we define

$$r(\ell, \mathbf{x}) = \inf_\sigma \mathbb{P}^\sigma_{(\ell,\mathbf{x})}\Big[\mathit{Term}\Big]$$

to denote the infimum termination probability over all schedulers in $\mathcal{C}$ when the initial state is $(\ell, \mathbf{x})$. The state function r is Borel-measurable, and the proof proceeds

analogously as in Appendix 4.10.3. The only difference in the proof is that we consider ε-optimal schedulers for the *infimum* reachability probability over all measurable schedulers for a given Borel-measurable target set, and their existence was also shown in [TOUH18, Appendix C]. Now, fix $n \geq n_0$ and define $\alpha_n \in (0,1)$ via the equality $p + \frac{1}{n} = \frac{p}{1-\alpha_n}$. Define $SI_n = \{(\ell, \mathbf{x}) \in State_{\mathcal{C}} \mid r(s) > \alpha_n\}$. We show that $(SI_n, p + \frac{1}{n})$ is a stochastic invariant such that a run in $\mathcal{C}$ with respect to every scheduler almost-surely reaches either some terminal state or a state in $\neg SI_n$. Our proof follows from Claim 1, which shows that $(SI_n, p + \frac{1}{n})$ is a stochastic invariant, and Claim 2, which shows that a run in $\mathcal{C}$ with respect to every scheduler almost-surely reaches either some terminal state or a state in $\neg SI_n$.

Claim 1. $(SI_n, p + \frac{1}{n})$ is a stochastic invariant in $\mathcal{C}$.

Proof of Claim 1. By theorem assumption, the program terminates with probability at least $1 - p$, thus we have $r(\ell_{init}, \mathbf{x}_{init}) \geq 1 - p$. On the other hand, by our choice of n_0 and the assumption that $n \geq n_0$, we have $\frac{p}{1-\alpha_n} = p + \frac{1}{n} \leq p + \frac{1}{n_0} < 1$ and so $1 - p > \alpha_n$. Combining the two inequalities, we conclude that $r(\ell_{init}, \mathbf{x}_{init}) > \alpha_n$ and the initial state is contained SI_n. Note that this is the part of the proof (ensuring that SI_n contains the initial state) in which it is essential to have $n \geq n_0$.

We are left to show that the set SI is left with probability at most p. Let $t = \sup_\sigma \mathbb{P}^\sigma[\mathsf{Reach}(\neg SI_n)]$ be the supremum probability of reaching $\neg SI_n$ over all schedulers in $\mathcal{C}$. In order to show that $(SI_n, p + \frac{1}{n})$ is a stochastic invariant, we need to prove that $t \leq p + \frac{1}{n}$. We prove this by contradiction. Suppose, on the contrary, that $t > p + \frac{1}{n}$. Then there exits a scheduler $\sigma_{\neg SI_n}$ for which $\mathbb{P}^{\sigma_{\neg SI_n}}[\mathsf{Reach}(\neg SI)] > p + \frac{1}{n}$. Consider a scheduler $\sigma'_{\neg SI_n}$ that follows $\sigma_{\neg SI_n}$ until a state in $\neg SI_n$ is reached, upon which it starts following a scheduler that minimizes the probability of termination. By the definition of SI_n and the choice of the scheduler $\sigma'_{\neg SI_n}$, it then follows that, with respect to the scheduler $\sigma'_{\neg SI_n}$, $\mathcal{C}$ does not terminate with probability at least $\mathbb{P}^{\sigma_{\neg SI_n}}[\mathsf{Reach}(\neg SI_n)] \cdot (1 - \alpha_n)$. On the other hand, it is a theorem assumption that $\mathcal{C}$ terminates with probability at least $1-p$ with respect to every scheduler, and hence does not terminate with probability at most p with respect to every scheduler. Hence, we have that $\mathbb{P}^{\sigma_{\neg SI_n}}[\mathsf{Reach}(\neg SI_n)] \cdot (1 - \alpha_n) \leq p$, and so $\mathbb{P}^{\sigma_{\neg SI}}[\mathsf{Reach}(\neg SI)] \leq \frac{p}{1-\alpha_n} = p + \frac{1}{n}$ by our choice of α_n. This leads to contradiction, and Claim 1 follows.

Claim 2. $\inf_\sigma \mathbb{P}^\sigma[Term \cup \mathsf{Reach}(\neg SI_n)] = 1$.

Proof of Claim 2. Our proof of Claim 2 assumes familiarity with the notions of conditional expectation, filtration and stopping time from probability theory, as well as the notion of canonical filtration in the probability space induced by a probabilistic program. Recall, an overview of all the required notions was presented in Section 2.4. Fix a scheduler σ.

In order to prove Claim 2, we need to show that $\mathbb{P}^{\sigma}[\mathit{Term} \cup \mathsf{Reach}(\neg SI_n)] = 1$. Our proof proceeds in several steps.

Step 1: Definition of k^*_σ. Define a state function k^*_σ via

$$k^*_\sigma(\ell, \mathbf{x}) = \min_{k \in \mathbb{N}_0} \left\{ \mathbb{P}^{\sigma}_{(\ell,\mathbf{x})}[\text{termination in at most } k \text{ steps}] > \alpha_n \right\}$$

for every $(\ell, \mathbf{x}) \in SI_n$, and $k^*_\sigma(\ell, \mathbf{x}) = 0$ otherwise. In order for this to be a state function, we need to show that $k^*_\sigma(\ell, \mathbf{x})$ is indeed finite in each state in SI_n, and that the resulting function is measurable. To prove finiteness, let $(\ell, \mathbf{x}) \in SI_n$. By definition of SI_n, we know that $r(\ell, \mathbf{x}) > \alpha_n$. Hence,

$$\begin{aligned}
\alpha_n < r(\ell, \mathbf{x}) &= \inf_{\sigma'} \mathbb{P}^{\sigma'}_{(\ell,\mathbf{x})}\Big[\mathit{Term}\Big] \leq \mathbb{P}^{\sigma}_{(\ell,\mathbf{x})}\Big[\mathit{Term}\Big] \\
&= \sum_{k=0}^{\infty} \mathbb{P}^{\sigma}_{(\ell,\mathbf{x})}\Big[\text{termination in exactly } k \text{ steps}\Big] \\
&= \sup_{k \in \mathbb{N}_0} \mathbb{P}^{\sigma}_{(\ell,\mathbf{x})}\Big[\text{termination in at most } k \text{ steps}\Big].
\end{aligned}$$

Hence, $\mathbb{P}^{\sigma}_{(\ell,\mathbf{x})}[\text{termination in at most } k \text{ steps}] > \alpha_n$, holds for a sufficiently large k and so $k^*_\sigma(\ell, \mathbf{x})$ is finite. To prove that k^*_σ is measurable observe that, for each $(\ell, \mathbf{x})$, we have

$$k^*_\sigma(\ell, \mathbf{x}) = \min_{k \in \mathbb{N}_0} \left\{ \mathbb{P}^{\leq k, \sigma}[\text{terminate } \neg SI_n] > \alpha_n \right\} \cdot \mathbb{I}_{(\ell,\mathbf{x}) \in SI_n}$$

with $\mathbb{P}^{\leq k,\sigma}[\,\cdot\,]$ being the operator defined as the probability of reaching some target set of states in at most k steps. As the measurability of this operator was proved in [TOUH21], the minimum is taken over a countable set and the indicator function is measurable, the measurability of k^*_σ follows.

Step 2: A sequence of stopping times $(T_i)_{i=0}^{\infty}$. We now inductively define a sequence of stopping times $(T_i)_{i=0}^{\infty}$ with respect to the canonical filtration $(\mathcal{R}_i)_{i=0}^{\infty}$ in the probability space $(\mathit{Run}_{\mathcal{C}}, \mathcal{F}_{\mathcal{C}}, \mathbb{P}^{\sigma})$ as follows:

- Set $T_0(\rho) = 0$ for each $\rho \in \mathit{Run}_{\mathcal{C}}$.
- For each $i \geq 1$, define T_i for each $\rho \in \mathit{Run}_{\mathcal{C}}$ via

$$T_i(\rho) = \begin{cases} T_{i-1}(\rho) + k^*(\rho_{T_{i-1}(\rho)}, \sigma), & \text{if } \rho \text{ does not leave } SI \\ & \text{or terminate in the first} \\ & T_{i-1}(\rho) + k^*(\rho_{T_{i-1}(\rho)}, \sigma) \text{ steps,} \\ T_{i-1}(\rho), & \text{otherwise} \end{cases}$$

where we use $\rho_{T_{i-1}(\rho)}$ to denote the $T_{i-1}(\rho)$-th state along ρ. Intuitively, $T_i(\rho)$ denotes the sum of the lengths of the first i finite paths of length $k^*((\ell, \mathbf{x}), \sigma)$, unless the program run ρ leaves SI_n or terminates.

The measurability of each T_i follows by induction and by the measurability of k^*_σ. To show that each T_i is a stopping time with respect to the canonical filtration $(\mathcal{R}_i)_{i=0}^{\infty}$, we need to show that for every $t \in \mathbb{N}_0$ we have $\{\rho \in Run_{\mathcal{C}} \mid T_i(\rho) \leq t\} \in \mathcal{R}_t$. This follows since the fact whether $T_i(\rho) \leq t$ for a run $\rho \in Run_{\mathcal{C}}$ is determined by the first t states along ρ.

Step 3: Stopping time T^*. Next, consider the filtration $(\mathcal{F}_{T_i})_{i=0}^{\infty}$ defined by the sequence $(T_i)_{i=0}^{\infty}$ of stopping times. That is, for each $i \in \mathbb{N}_0$, we define $\mathcal{F}_{T_i}$ via

$$\mathcal{F}_{T_i} := \cup_{t=0}^{\infty}\{A \cap \{T_i \leq t\} \mid A \in \mathcal{R}_t\}.$$

This set is non-empty since each stopping time T_i is a.s. finite (which follows by induction on i and the fact that k^*_σ is finite in every state). Furthermore, each $\mathcal{F}_{T_i}$ can be proved to be a σ-algebra by checking that all the defining conditions are satisfied. Hence, $(\mathcal{F}_{T_i})_{i=0}^{\infty}$ is an increasing sequence of σ-algebras and defines a filtration. Thus, we may define a stopping time T^* with respect to the filtration $(\mathcal{F}_{T_i})_{i=0}^{\infty}$ via

$$T^*(\rho) = \inf_{k \in \mathbb{N}_0} \Big\{\rho \text{ terminates or leaves } SI_n \text{ in the first } T_k(\rho) \text{ steps}\Big\}.$$

The fact that T^* is measurable and a stopping time follows since it is the first hitting time of the set $\neg SI_n \cup State_{term}$ with respect to the filtration $(\mathcal{F}_{T_i})_{i=0}^{\infty}$, and it is a standard result on stopping times that the first hitting time of a set is a stopping time [Wil91, Section 10.8].

Step 4: Proof that $\mathbb{P}^\sigma[Term \cup \text{Reach}(\neg SI)] = 1$. We are finally ready to prove the desired claim. By the definitions of k^*_σ, $(T_i)_{i=0}^{\infty}$ and T^*, it follows that

$$\mathbb{P}^\sigma[T^* \leq k+1 \mid \mathcal{F}_{T_k}] > \alpha_n$$

holds for each $k \in \mathbb{N}_0$. Since $\alpha_n > 0$ does not depend on the index k, it follows from a known result on stopping times [Wil91, Lemma 10.11] that $\mathbb{E}^\sigma[T^*] < \infty$. But $\mathbb{E}^\sigma[T^*] < \infty$ implies $\mathbb{P}^\sigma[T^* < \infty] = 1$ and we have $\{\rho \in Run_{\mathcal{C}} \mid T^*(\rho) < \infty\} = Term \cup \text{Reach}(\neg SI_n)$, so Claim 2 follows.

Proof that (SI, p) is a stochastic invariant with $\neg SI \cup State_{term}$ reached a.s. Indeed, due to our definitions of stochastic invariants and predicate functions, $SI_n(\ell) \subseteq \mathbb{R}^{|V|}$ is Borel-measurable for each $n \geq n_0$ and a location ℓ in $\mathcal{C}$. Hence, $SI(\ell) = \cap_{n=n_0}^{\infty} SI_n \subseteq \mathbb{R}^{|V|}$ is also Borel-measurable as a countable intersection of

Borel-measurable sets. Next, we need to show that (SI, p) is a stochastic invariant, i.e. that SI contains the initial state and that a random run leaves SI with probability at most p. The fact that SI contains the initial program state follows since all SI_n's contain the initial state due to $(SI_n, p + \frac{1}{n})$ being stochastic invariants. Now, let $t = \sup_\sigma \mathbb{P}^\sigma[\text{Reach}(\neg SI)]$. Since $SI \subseteq SI_n$ and $(SI_n, p + \frac{1}{n})$ is a stochastic invariant, we have that $t = \sup_\sigma \mathbb{P}^\sigma[\text{Reach}(\neg SI_n)] \geq \sup_\sigma \mathbb{P}^\sigma[\text{Reach}(\neg SI_n)] \geq p + \frac{1}{n}$ for each $n \geq n_0$. Thus, by letting $n \to \infty$, we conclude that $t \geq p$ so (SI, p) is a stochastic invariant. Finally, to show that a run in $\mathcal{C}$ with respect to every scheduler almost-surely reaches either some terminal state or a state in $\neg SI$, set $n = n_0$ and observe that by assumption a run almost-surely reaches either some terminal state or a state in $\neg SI_{n_0} \subseteq \neg SI$. This concludes the proof that (SI, p) yields a desired stochastic invariant. □

4.10.2 Proof of Theorem 4.4.2

Theorem (Soundness and Completeness of SI-indicators)**.** *Let $\mathcal{C}$ be a pCFG, I an invariant in $\mathcal{C}$ and $p \in [0, 1]$. For any SI-indicator (f_{SI}, p) with respect to I, the predicate map SI defined as*

$$SI(\ell) = (\mathbf{x} \models I(\ell) \wedge f_{SI}(\ell, \mathbf{x}) < 1)$$

yields a stochastic invariant (SI, p) in $\mathcal{C}$. Conversely, for every stochastic invariant (SI, p) in $\mathcal{C}$, there exist an invariant I_{SI} and a state function f_{SI} such that (f_{SI}, p) is an SI-indicator with respect to I_{SI} and for each $\ell \in L$ we have

$$SI(\ell) \supseteq (\mathbf{x} \models I_{SI}(\ell) \wedge f_{SI}(\ell, \mathbf{x}) < 1).$$

Our proof of the theorem builds on the existing results on reachability analysis in PPs from [TOUH21]. To that end, we first recall the result of [TOUH21] which shows that, if we are provided with a target set of states, then the reachability probabilities for that target set can be characterized as the least fixed point of a suitably constructed operator that simulates one-step execution of the program's pCFG. In the sequel, we assume basic familiarity with fixed point theory. An overview of the required notions is provided in Section 2.5.

Lattice of state functions. We consider the lattice of nonnegative *upper semianalytic* state functions in $\mathcal{C}$, that map states in the invariant I to nonnegative (possibly infinite) values:

$$\mathcal{L} = \{f \text{ upper semianalytic } \mid f : State_{\mathcal{C}}^{I} \to [0, \infty]\}.$$

The class of upper semianalytic state functions extends Borel-measurable state functions (that we considered so far), and this is a technical condition needed for the next-time

operator defined below to be closed in this lattice [TOUH21]. This technical condition does not affect any of our results and hence we do not define this notion formally but refer the reader to [TOUH21, BS04]. We define the partial order $\sqsubseteq$ on $\mathcal{L}$ in an intuitive manner. For a pair of state functions f, f' in $\mathcal{L}$, we write $f \sqsubseteq f'$ if $f(\ell, \mathbf{x}) \leq f'(\ell, \mathbf{x})$ for each state $(\ell, \mathbf{x})$ in I. With all operations defined state-wise, one easily sees that $(\mathcal{L}, \sqsubseteq)$ is a lattice with $f \wedge f' = \min\{f, f'\}$ and $f \vee f' = \max\{f, f'\}$. Furthermore, it is ω-complete, meaning that each ascending chain $f_1 \sqsubseteq f_2 \sqsubseteq \dots$ has a supremum given by $f = \sup\{f_1, f_2, \dots\}$. The bottom and the top elements are defined via $\bot(\ell, \mathbf{x}) = 0$ and $\top(\ell, \mathbf{x}) = \infty$, respectively, for each state $(\ell, \mathbf{x})$ in I.

Next-time operator. Intuitively, the next-time operator $\mathbb{X} : \mathcal{L} \to \mathcal{L}$ simulates a one-step execution of $\mathcal{C}$ and maps f to a state function equal to its maximal expected value with respect to all schedulers upon this one-step execution. To formally define it, let $f \in \mathcal{L}$. Then, for any state $(\ell, \mathbf{x})$ in I, depending on the type of the location ℓ in $\mathcal{C}$ we define $\mathbb{X}(f)(\ell, \mathbf{x})$ as follows:

- If $\ell \in L_C$, let $\tau = (\ell, \ell')$ be the transition with $\mathbf{x} \models G(\tau)$. Then $\mathbb{X}(f)(\ell, \mathbf{x}) = f(\ell', \mathbf{x})$.
- If $\ell \in L_P$, then $\mathbb{X}(f)(\ell, \mathbf{x}) = \sum_{\tau=(\ell,\ell') \in \mapsto} \Pr(\tau) \cdot f(\ell', \mathbf{x})$.
- If $\ell \in L_N$, then $\mathbb{X}(f)(\ell, \mathbf{x}) = \max_{\tau=(\ell,\ell') \in \mapsto} f(\ell', \mathbf{x})$.
- If $\ell \in L_A$ with $\tau = (\ell, \ell')$ the unique outgoing transition from ℓ and $Up(\tau) = (j, u)$, then:
  - If $u = \bot$, then $\mathbb{X}(f)(\ell, \mathbf{x}) = f(\ell', \mathbf{x})$.
  - If $u : \mathbb{R}^{|V|} \to \mathbb{R}$, then $\mathbb{X}(f)(\ell, \mathbf{x}) = f(\ell', \mathbf{x}[x_j \leftarrow u(\mathbf{x})])$.
  - If $u = d$, then $\mathbb{X}(f)(\ell, \mathbf{x}) = \mathbb{E}_{X \sim d}[f(\ell', \mathbf{x}[x_j \leftarrow X])]$.
  - If $u = [a, b]$, then $\mathbb{X}(f)(\ell, \mathbf{x}) = \sup_{X \in [a,b]}\{f(\ell', \mathbf{x}[x_j \leftarrow X])\}$.

The fact that for an upper semianalytic state function $f \in \mathcal{L}$ we have $\mathbb{X}(f) \in \mathcal{L}$ was proved in [TOUH21].

Characterization of reachability probabilities. Let T be a predicate function in $\mathcal{C}$. Then define the operator $\mathbb{X}_T : \mathcal{L} \to \mathcal{L}$ in the lattice $(\mathcal{L}, \sqsubseteq)$ as follows:

$$\mathbb{X}_T(f)(\ell, \mathbf{x}) = \begin{cases} \mathbb{X}(f)(\ell, \mathbf{x}), & \text{if } \mathbf{x} \not\models T(\ell) \\ 1, & \text{otherwise,} \end{cases} \tag{4.9}$$

for each $f \in \mathcal{L}$ and $\mathbf{x} \models I(\ell)$. Thus, $\mathbb{X}_T$ behaves analogously as $\mathbb{X}$ and simulates a one-step execution of $\mathcal{C}$ in states not contained in T, but evaluates to 1 for states in T. The following proposition states that reachability probabilities for the target set T can be characterized in terms of the least fixed point of the operator $\mathbb{X}_T$, and that pre-fixed points of $\mathbb{X}_T$ can be used to bound the reachability probabilities from above (Proposition 4.2 and Corollary 4.7 in [TOUH21], respectively). Recall, a pre-fixed point of $\mathbb{X}_T$ is a state function $f \in \mathcal{L}$ that satisfies $\mathbb{X}_T(f) \sqsubseteq f$.

Proposition 4.10.1 ([TOUH21]). *The operator $\mathbb{X}_T : \mathcal{L} \to \mathcal{L}$ is ω-continuous, and we have $\sup_\sigma \mathbb{P}^\sigma_{(\ell,\mathbf{x})}[\mathsf{Reach}(T)] = \mathrm{lfp}\, \mathbb{X}_T(\ell, \mathbf{x})$ for each state $(\ell, \mathbf{x})$ in I. For any state function $f \in \mathcal{L}$ which is a pre-fixed point of $\mathbb{X}_T$ and for each state $(\ell, \mathbf{x})$ in I, we have $\sup_\sigma \mathbb{P}^\sigma_{(\ell,\mathbf{x})}[\mathsf{Reach}(T)] \leq f(\ell, \mathbf{x})$.*

We are now ready to prove the claim of Theorem 4.4.2.

Proof of Theorem 4.4.2. Suppose first that we are given an invariant I in $\mathcal{C}$ and an SI-indicator (f_{SI}, p) with respect to I. We want to show that, if we define a predicate function SI via $SI(\ell) = (\mathbf{x} \models I(\ell) \wedge f_{SI}(\ell, \mathbf{x}) < 1)$ for each $\ell \in L$, then (SI, p) is a stochastic invariant in $\mathcal{C}$. Consider the operator $\mathbb{X}_{\neg SI} : \mathcal{L} \to \mathcal{L}$ in the lattice $(\mathcal{L}, \sqsubseteq)$, so that the target set of states is the complement of SI. Observe that $f_{SI} \in \mathcal{L}$. Indeed, $f_{SI}(\ell)$ is Borel-measurable for each $\ell \in L$ hence also upper semianalytic, and the fact that it is nonnegative at each state in I follows from (C_1) in Definition 4.4.1. We now claim that f_{SI} is a pre-fixed point of $\mathbb{X}_{\neg SI}$ in $\mathcal{L}$:

- If $(\ell, \mathbf{x})$ is a state with $\mathbf{x} \in I(\ell) \cap SI(\ell)$, then we have $\mathbb{X}_{\neg SI}(f_{SI})(\ell, \mathbf{x}) = \mathbb{X}(f_{SI})(\ell, \mathbf{x}) \leq f_{SI}(\ell, \mathbf{x})$, where the first equation follows from eq. (4.9), and the second from the condition (C_2) on non-increasing expected value in Definition 4.4.1.
- If $(\ell, \mathbf{x})$ is a state with $\mathbf{x} \in I(\ell) \cap (\neg SI(\ell))$, then by definition of SI we have $f_{SI}(\ell, \mathbf{x}) \geq 1$. But $\mathbb{X}_{\neg SI}(f_{SI})(\ell, \mathbf{x}) = 1$ by eq. (4.9), and so $\mathbb{X}_{\neg SI}(f_{SI})(\ell, \mathbf{x}) \leq f_{SI}(\ell, \mathbf{x})$.

Hence f_{SI} is a pre-fixed point of $\mathbb{X}_{\neg SI}$ and by Proposition 4.10.1 it follows that $\sup_\sigma \mathbb{P}^\sigma_{(\ell,\mathbf{x})}[\mathsf{Reach}(\neg SI)] \leq f_{SI}(\ell_{init}, \mathbf{x}_{init})$. But from (C_3) in Definition 4.4.1 of SI-indicators we know that $f_{SI}(\ell_{init}, \mathbf{x}_{init}) \leq p$, so $\sup_\sigma \mathbb{P}^\sigma_{(\ell,\mathbf{x})}[\mathsf{Reach}(\neg SI)] \leq p$. This concludes the proof that (SI, p) is a stochastic invariant in $\mathcal{C}$.

Now we prove the second part of the theorem. Suppose that (SI, p) is a stochastic invariant in $\mathcal{C}$. We need to show that there exist an invariant I_{SI} and a state function f_{SI} such that (f_{SI}, p) is an SI-indicator with respect to I_{SI} and for each $\ell \in L$ we

have $SI(\ell) \equiv (\mathbf{x} \models I_{SI}(\ell) \wedge f_{SI}(\ell, \mathbf{x}) < 1)$. We prove this by giving an explicit construction for I_{SI} and f_{SI}. Define the invariant I_{SI} to be the trivial true invariant, i.e. $I_{SI}(\ell) = true$ for each $\ell \in L$. As for f_{SI}, for each state $(\ell, \mathbf{x})$ in I define

$$f_{SI}(\ell, \mathbf{x}) = \sup_{\sigma} \mathbb{P}^{\sigma}_{(\ell,\mathbf{x})}[\mathsf{Reach}(\neg SI)].$$

First, we need to show that $f_{SI}(\ell)$ is Borel-measurable for each $\ell \in L$ so that f_{SI} is a state function. We defer this technical proof to Appendix 4.10.3 below. Next, by Proposition 4.10.1, we know that f_{SI} is the least fixed point of the operator $\mathbb{X}_{\neg SI}$. This implies that (f_{SI}, p) satisfies both conditions (C_1) (Nonnegativity) and (C_2) (Non-increasing expected value) in Definition 4.4.1 with respect to the trivial invariant I_{SI} of all states in $\mathcal{C}$. Finally, since (SI, p) is a stochastic invariant in $\mathcal{C}$, we have $f_{SI}(\ell_{init}, \mathbf{x}_{init}) = \sup_{\sigma} \mathbb{P}^{\sigma}_{(\ell,\mathbf{x}_{init})}[\mathsf{Reach}(\neg SI)] \leq p$, so (f_{SI}, p) satisfies the Initial condition (C_3) in Definition 4.4.1. Hence, (f_{SI}, p) is an SI-indicator with respect to I_{SI}. The fact that $SI(\ell) \supseteq (\mathbf{x} \models I_{SI}(\ell) \wedge f_{SI}(\ell, \mathbf{x}) < 1)$ follows since $1 > f_{SI}(\ell, \mathbf{x}) = \sup_{\sigma} \mathbb{P}^{\sigma}_{(\ell,\mathbf{x})}[\mathsf{Reach}(\neg SI)]$ implies that $(\ell, \mathbf{x})$ cannot be contained in $\neg SI$ so $\mathbf{x} \models SI(\ell)$. □

4.10.3 Measurability Argument in the Proof of Theorem 4.4.2

Let $\mathcal{C} = (L, V, \ell_{init}, \mathbf{x}_{init}, \mapsto, G, Pr, Up)$ be a pCFG. Let T be a predicate function defining a target set of states in $\mathcal{C}$, and $\varepsilon > 0$. We say that a scheduler σ is *ε-optimal* for the reachability objective T, if

$$\mathbb{P}^{\sigma}_{(\ell,\mathbf{x})}\Big[Reach(T)\Big] \geq \sup_{\sigma'} \mathbb{P}^{\sigma'}_{(\ell,\mathbf{x})}\Big[Reach(T)\Big] - \varepsilon$$

for any state $(\ell, \mathbf{x}) \in State_{\mathcal{C}}$.

It was shown in [TOUH18, Appendix C] that a pCFG can be translated to an equivalent *infinite horizon stochastic optimal control model* [BS04]. In infinite horizon stochastic optimal control models, a cost is incurred in each state, and it is known that an ε-optimal scheduler for the objective to maximize the discounted cost exists [BS04, Proposition 9.20]. The work [TOUH18, Appendix C] then shows that, once a pCFG is translated into an infinite horizon stochastic optimal control model, costs can be chosen in such a way that the total discounted cost with the discount factor $\alpha = 1$ is equal to the supremum reachability probability over all measurable schedulers for any given Borel-measurable target set. Hence, for every $\varepsilon > 0$ and a predicate function T defining target set of states, there exits an ε-optimal scheduler for T (recall, in Section 2.2 we assume that predicate functions are defined in terms of Borel-measurable expressions).

To prove that $\sup_{\sigma} \mathbb{P}^{\sigma}_{(\ell,\mathbf{x})}[Reach(\neg SI)]$ is a Borel-measurable state function, for each $n \in \mathbb{N}$ let σ_n be a $\frac{1}{n}$-optimal scheduler for the target set of states defined by $\neg SI$. We

then have

$$\sup_{\sigma} \mathbb{P}^{\sigma}_{(\ell,\mathbf{x})}\Big[Reach(\neg SI)\Big] = \sup_{n\in\mathbb{N}} \mathbb{P}^{\sigma_n}_{(\ell,\mathbf{x})}\Big[Reach(\neg SI)\Big].$$

Thus, it suffices to prove that each $\mathbb{P}^{\sigma_n}_{(\ell,\mathbf{x})}[Reach(\neg SI)]$ is a Borel-measurable state function, since a supremum of *countably many* Borel-measurable functions is Borel-measurable.

Fix $n \in \mathbb{N}$. To prove that $\mathbb{P}^{\sigma_n}_{(\ell,\mathbf{x})}[Reach(\neg SI)]$ is a Borel-measurable state function, observe that

$$\mathbb{P}^{\sigma_n}_{(\ell,\mathbf{x})}\Big[Reach(\neg SI)\Big] = \sup_{m\in\mathbb{N}} \mathbb{P}^{\sigma_n}_{(\ell,\mathbf{x})}\Big[Reach^{\leq m}(\neg SI)\Big],$$

where $Reach^{\leq m}(\neg SI)$ denotes the set of all runs in $\mathcal{C}$ whose finite prefix of length at most m reaches a state in $\neg SI$. The last equality follows by first observing that the sequence of indicator function $\mathbb{I}(Reach^{\leq m}(\neg SI)) \to \mathbb{I}(Reach(\neg SI))$ converges pointwise as $m \to \infty$ and is a pointwise increasing sequence, and then applying the Monotone Convergence Theorem [Wil91]. Again, since a countable supremum of Borel-measurable functions is Borel-measurable, it suffices to show that $\mathbb{P}^{\sigma_n}_{(\ell,\mathbf{x})}[Reach^{\leq m}(\neg SI)]$ defines a Borel-measurable state function for each $n, m \in \mathbb{N}$.

But for fixed $n, m \in \mathbb{N}$, this set can be defined inductively in terms of finitely many expected value operators, due to finiteness of m (note, scheduler σ_n is fixed, so we need not take a supremum). Hence a simple induction on m shows that $\mathbb{P}^{\sigma_n}_{(\ell,\mathbf{x})}[Reach^{\leq m}(\neg SI)]$ defines a Borel-measurable state function for each $n, m \in \mathbb{N}$, which concludes the proof.

4.10.4 Proof of Theorem 4.5.2

Theorem. *Let $\mathcal{C}$ be a pCFG, I an invariant in $\mathcal{C}$ and T a predicate function defining a target set of states. If there exist $\varepsilon > 0$ and an ε-RSM for T with respect to I, then T is a.s. reached under any scheduler, i.e.*

$$\inf_{\sigma} \mathbb{P}^{\sigma}_{(\ell_{init},\mathbf{x}_{init})}\Big[\mathsf{Reach}(T)\Big] = 1.$$

In order to prove the theorem, we first need to recall the mathematical notion of ranking supermartingales. Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space, $(\mathcal{F}_i)_{i=0}^{\infty}$ a filtration in it and $\varepsilon > 0$. Let T be a stopping time in $(\Omega, \mathcal{F}, \mathbb{P})$ with respect to the filtration $(\mathcal{F}_i)_{i=0}^{\infty}$. A stochastic process $(X_i)_{i=0}^{\infty}$ is said to be an *ε-ranking supermartingale (ε-RSM)* with respect to the stopping time T if it satisfies the following conditions:

- Each X_i is $\mathcal{F}_i$-measurable.
- We have $X_i(\omega) \geq 0$ for each $i \geq 0$ and $\omega \in \Omega$.

- Each X_i is integrable, i.e. $\mathbb{E}[|X_i|] = \mathbb{E}[X_i] < \infty$.
- For each $i \geq 0$ and $\omega \in \Omega$, we have $\mathbb{E}[X_{i+1} \mid \mathcal{F}_i](\omega) \leq X_i(\omega) - \varepsilon \cdot \mathbb{I}(T(\omega) < i)$.

The following theorem is a classical result on ranking supermartingales for their use in the probabilistic program analysis.

Theorem 4.10.2 ([FC19]). *Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space, $(\mathcal{F}_i)_{i=0}^{\infty}$ a filtration and $\varepsilon > 0$. Let T be a stopping time in $(\Omega, \mathcal{F}, \mathbb{P})$ with respect to the filtration $(\mathcal{F}_i)_{i=0}^{\infty}$. Suppose that there exists an ε-RSM $(X_i)_{i=0}^{\infty}$ with respect to T. Then $\mathbb{P}[T < \infty] = 1$.*

We are now ready to prove Theorem 4.5.2.

Proof of Theorem 4.5.2. By the theorem assumption, there exist $\varepsilon > 0$ and a state function η in $\mathcal{C}$ which is an ε-RSM for the target set of states T with respect to the invariant I. We need to show that, with respect to any scheduler in $\mathcal{C}$, a state in T is reached with probability 1.

Fix a scheduler σ. Recall, $\mathcal{C}$ and σ together give rise to a probability space $(\Omega_{\mathcal{C}}, \mathcal{F}_{\mathcal{C}}, \mathbb{P}^{\sigma})$ over the set of runs in $\mathcal{C}$. In order to prove the theorem claim, we define the stopping time $TimeReach_T$ with respect to the canonical filtration $(\mathcal{R}_i)_{i=0}^{\infty}$ to be the first time of reaching the target set of states T. We then use η to construct an ε-RSM with respect to $TimeReach_T$, which by Theorem 4.10.2 implies that $\mathbb{P}^{\sigma}[TimeReach_T] < \infty$ and therefore that T is reached with probability 1 with respect to the scheduler σ. Since σ was arbitrary, the theorem claim follows.

For each run $\rho \in Run_{\mathcal{C}}$, let $(\ell_i^{\rho}, \mathbf{x}_i^{\rho})$ be the i-th state along ρ. Define a stochastic process $(X_i)_{i=0}^{\infty}$ in $(\Omega_{\mathcal{C}}, \mathcal{F}_{\mathcal{C}}, \mathbb{P}^{\sigma})$ as follows

$$X_i(\rho) = \begin{cases} \eta(\ell_i^{\rho}, \mathbf{x}_i^{\rho}), & \text{if } TimeReach_T(\rho) < i, \\ \eta(\ell^{\rho}_{TimeReach_T(\rho)}, \mathbf{x}^{\rho}_{TimeReach_T(\rho)}), & \text{otherwise.} \end{cases} \tag{4.10}$$

We show that $(X_i)_{i=0}^{\infty}$ is indeed an ε-RSM with respect to $TimeReach_T$ by verifying that each of the four defining conditions of the mathematical notion of ε-RSMs is satisfied:

- Each X_i is defined in terms of the i-th state along a program run, hence is $\mathcal{R}_i$-measurable.
- We have $\mathbb{E}[X_0] = \eta(\ell_{init}, \mathbf{x}_{init}) < \infty$ since the codomain of η are real numbers. Once we show in the fourth item below that $\mathbb{E}[X_{i+1} \mid \mathcal{F}_i](\rho) \leq X_i(\rho) - \varepsilon \cdot \mathbb{I}(TimeReach_T(\rho) < i)$ for each i and $\rho \in Run_{\mathcal{C}}$, by taking the expected value

on both sides and by recalling the definition of conditional expectation, it will follow that $\mathbb{E}[X_{i+1}] \leq \mathbb{E}[X_i]$ for each i. Hence, a simple induction shows that $\mathbb{E}[X_i] \leq \mathbb{E}[X_0] < \infty$ for each i.

- By the Nonnegativity condition in Definition 4.5.1, we know that $\eta(\ell, \mathbf{x}) \geq 0$ for any location ℓ and a variable valuation $\mathbf{x} \models I(\ell)$. For any $\rho \in Run_{\mathcal{C}}$ and any $i \geq 0$, the state $(\ell_i^\rho, \mathbf{x}_i^\rho)$ is reachable and hence by the definition of invariants we have $\mathbf{x}_i^\rho \models I(\ell_i^\rho)$. Thus, $\eta(\ell_i^\rho, \mathbf{x}_i^\rho) \geq 0$ for any run ρ and $i \geq 0$. It follows from eq. (4.10) that $X_i(\rho) \geq 0$ for any run ρ and $i \geq 0$.

- We need to show that $\mathbb{E}[X_{i+1} \mid \mathcal{F}_i](\rho) \leq X_i(\rho) - \varepsilon \cdot \mathbb{I}(\mathit{TimeReach}_T(\rho) < i)$ for each i and $\rho \in Run_{\mathcal{C}}$. Fix $i \geq 0$ and $\rho \in Run_{\mathcal{C}}$.

  If $\mathit{TimeReach}_T(\rho) \geq i$, then it follows by the definition of $(X_i)_{i=0}^\infty$ both sides of the formula are equal to $\eta(\ell^\rho_{\mathit{TimeReach}_T(\rho)}, \mathbf{x}^\rho_{\mathit{TimeReach}_T(\rho)})$ and the claim follows.

  If $\mathit{TimeReach}_T(\rho) < i$, we have

$$\begin{aligned}&\mathbb{E}[X_{i+1} \mid \mathcal{F}_i](\rho) \leq \mathbb{X}(\eta)(\ell_i^\rho, \mathbf{x}_i^\rho) \leq \eta(\ell_i^\rho, \mathbf{x}_i^\rho) - \varepsilon \\ &= X_i(\rho) - \varepsilon \cdot \mathbb{I}(\mathit{TimeReach}_T(\rho) < i),\end{aligned}$$

  as wanted, where the second inequality holds by the ε-ranked expected value condition in Definition 4.5.1.

Hence $(X_i)_{i=0}^\infty$ is an ε-RSM with respect to $\mathit{TimeReach}_T$, and the theorem claim follows. From Theorem 4.10.2 we may now conclude that $\mathbb{P}^\sigma[\mathit{TimeReach}_T] < \infty$ and therefore that T is reached with probability 1 with respect to the scheduler σ. Since σ was arbitrary, the theorem claim follows. □

CHAPTER 5

Non-termination Analysis in Programs

This section is based on the following publication:

- Krishnendu Chatterjee, Ehsan Kafshdar Goharshady, Petr Novotný, Đorđe Žikelić.[†] *Proving Non-termination by Program Reversal*. In 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation, **PLDI 2021**

5.1 Introduction

Disproving properties in PPs with non-determinism. Previous chapters have focused on *proving* properties in PPs with non-determinism. Recall, the goal of qualitative (resp. quantitative) termination analysis is to show that a PP terminates with probability 1 (resp. at least $p \in [0, 1]$), with respect to *every* scheduler. Similarly, the goal of the qualitative (resp. quantitative) safety analysis is to show that the PP avoids some set of states S with probability 1 (resp. at least $p \in [0, 1]$), again with respect to *every* scheduler. Hence, in PPs with non-determinism, it would be too conservative to disprove a termination property by performing safety analysis and vice-versa and it would lead to incomplete approaches, since both analyses prove lower bounds on probability with respect to *every* scheduler. In order to *disprove* a property in PPs with non-determinism, it suffices to show that the logical negation of the property is satisfied with respect to *some* scheduler.

[†]Authors ordered alphabetically.

Non-termination proving. In this chapter, we study the problem of disproving termination, or in other words proving non-termination. In contrast to previous chapters, we study non-termination proving in *non-probabilistic programs*. This is because our results have important implications already in non-probabilistic program analysis and significantly advance the state of the art. We will then show that our method for proving non-termination in non-probabilistic programs naturally extends to disproving qualitative termination in *PPs with discrete probability distributions*, i.e. PPs in which all probability distributions appearing in sampling instructions have countable support. In what follows, we start by surveying existing approaches on non-termination proving in non-probabilistic programs and identifying their limitations, before presenting the contributions of this chapter.

Program analysis of non-probabilistic programs. There are two relevant directions in program analysis: to prove program correctness and to find bugs. While a correctness proof is obtained once, the procedure of bug finding is more relevant during software development and is repeatedly applied, even for incomplete or partial programs. In terms of specifications, the most basic properties in program analysis are safety and liveness. The analysis of non-probabilistic programs with respect to safety properties has received a lot of attention [BR02, HJMS02, GKS05, GHK$^+$06], and for safety properties to report errors the witnesses are finite traces violating the safety property. The most basic liveness property is termination. There is a huge body of work for proving correctness with respect to the termination property [FGKP85, CS02, BMS05b, Cou05], e.g. sound and complete methods based on ranking functions have been developed [CS01, PR04a, PR04b], and efficient computational approaches based on lexicographic ranking functions have also been considered [BMS05b, CSZ13, BCF13].

Non-termination proving in non-probabilistic programs. The bug finding problem for the termination property, or proving non-termination, is a challenging problem. Conceptually, while for a safety property the violating witness is a finite trace, for a termination property the violating witnesses are infinite traces. There are several approaches for proving non-termination; here we discuss some key ones, which are most related in spirit to our new method (for a detailed discussion of related work, see Section 5.7). For the purpose of this overview, we (rather broadly and with a certain grain of salt) classify the approaches into two categories: *trace-based* approaches, which look for a non-terminating trace (e.g. [GHM$^+$08, LH18, FG19]), and *set-based approaches,* which look for a set of non-terminal program states in which the program can stay indefinitely (e.g. [CCF$^+$14, LNO$^+$14, GAB$^+$17]). For instance, the work of [GHM$^+$08] considers computing "lassos" (where a lasso is a finite prefix followed by a finite cycle infinitely repeated) as counter-examples for termination and presents a trace-based approach based on lassos to prove non-termination of deterministic

programs. In general, finite lassos are not sufficient to witness non-termination. While lassos are periodic, proving non-termination for programs with aperiodic infinite traces via set-based methods has been considered in [CCF+14, LNO+14] for programs with non-determinism. In [CCF+14], a method is proposed where "closed recurrence sets" of states are used to prove non-termination. Intuitively, a closed recurrence set must contain some initial state, must contain no terminal states, and cannot be escaped once entered. In [CCF+14], closed recurrence sets are defined with respect to under-approximations of the transition relation, and an under-approximation search guided by several calls to a safety prover is used to compute a closed recurrence set. In [LNO+14], a constraint solving-based method is proposed to search for "quasi-invariants" (sets of states which cannot be left once entered) exhaustively in all strongly-connected subgraphs. A safety prover is used to check reachability for every obtained quasi-invariant. For constraint solving, Max-SMT is used in [LNO+14].

Limitations of previous approaches. While the previous works represent significant advancement for proving non-termination, each of them has, to the best of our knowledge, at least one of the following limitations:

1. They do not support non-determinism, e.g. [VR08, GSV08].
2. They only work for lassos (i.e. periodic non-terminating traces), e.g. [GHM+08].
3. *Theoretical limitation* of not providing any *(relative) completeness guarantees.* Clearly, a non-termination proving algorithm cannot be both sound and complete, since non-termination is well-known to be undecidable. However, as in the case of termination proving, it can be beneficial to provide relative completeness guarantees, i.e. conditions on the input program under which the algorithm is guaranteed to prove non-termination. To our best knowledge, the only approaches with such guarantees are [LH18, GSV08]; however, both of them only provide guarantees for a certain class of *deterministic* programs.
4. Most of the previous approaches do not support programs with polynomial arithmetic (with an exception of [CFNO14, FG19]).

Contributions. In this work we propose a new set-based approach to non-termination proving in integer programs with polynomial arithmetic. Intuitively, it searches for a diverging program state, i.e. a state that is reachable but from which no program run is terminating (after resolving non-determinism using symbolic polynomial assignments). Our approach is based on a simple technique of *program reversal*, which reverses each transition in the program's transition system to produce the reversed transition system. The key property of this construction is that, given a program state, there

is a terminating run starting in it if and only if it is reachable from the terminal location in the reversed transition system. This allows over-approximating the set of all program states from which termination can be reached by computing an invariant in the program's reversed transition system. We refer to the invariants in reversed transition systems as *backward invariants*. To generate the backward invariant, we may employ state-of-the-art polynomial invariant generation techniques to the reversed transition system as a single-shot procedure which is the main practical benefit of the program reversal. Our method proves non-termination by generating a backward invariant whose complement is reachable. Hence, our new method adapts the classical and well-studied techniques for inductive invariant generation in order to find non-termination proofs by combining forward and backward analysis of a program. While such a combined analysis is common in safety analysis where the goal is to show that no program run reaches some annotated set of states [Bou93], to our best knowledge it has never been considered for proving non-termination in programs with non-determinism, where we need to find a single program run that does not terminate. The key features of our method are as follows:

1. Our approach supports programs with non-determinism.
2. Our approach is applicable to programs where all non-terminating traces are aperiodic.
3. *Relative completeness guarantee:* The work of [CCF+14] establishes that closed recurrence sets with respect to under-approximations are a sound and complete *certificate* of non-termination, yet the algorithm based on these certificates does not in itself provide any relative completeness guarantee (in the above sense). For our approach we show the following: If there is an under-approximation of the transition relation where non-determinism can be resolved by polynomial assignments such that the resolved program contains a closed recurrence set representable as a polynomial predicate function, then our approach is guaranteed to prove non-termination. We obtain such guarantee by employing relatively complete methods for inductive invariant synthesis, which is another key advantage of adapting invariant generation techniques to non-termination proving. Moreover, we provide even stronger relative completeness guarantees for programs in which non-determinism appears only in branching (but not in variable assignments).
4. Our approach supports programs with polynomial arithmetic.

We developed a prototype tool RevTerm which implements our approach. We experimentally compared our tool with state-of-the-art non-termination provers on standard benchmarks from the Termination and Complexity Competition (TermComp'19 [GRS+19]).

Our tool demonstrates performance on par with the most efficient of the competing provers, while providing additional guarantees. In particular, with a proper configuration, our tool achieved the largest number of benchmarks proved non-terminating.

Extension to disproving qualitative termination in PPs. Finally, while the above results consider programs with non-determinism, we note that our approach naturally yields a method for disproving qualitative termination analysis in PPs with non-determinism in which *all probability distributions are discrete*. To see this, note that the existence of any diverging state in such a PP implies that there exists a scheduler under which the PP terminates with probability strictly less than 1. Indeed, consider the scheduler under which a diverging state s is reached. Then, due to all probability distributions being discrete, it follows that s is reached with strictly positive probability. Hence, as s is diverging so any path that reaches s is non-terminating, it follows that under this scheduler the PP terminates with probability strictly less than 1. Thus, in order to disprove qualitative termination in a PP with non-determinism and with discrete probability distributions, it suffices to (1) replace all probabilistic branching and assignment instructions with non-deterministic assignment and branching instructions, and (2) use our method for proving non-termination in non-probabilistic programs in order to disprove qualitative termination of the original PP. We note that, while it is a straightforward extension of our method, this extension to PPs was not discussed in [CGNZ21] and we highlight it in this thesis for the first time.

Chapter organization. The rest of this chapter is organized as follows. Since this chapter is the only chapter that studies non-probabilistic programs, in Section 5.2 we present necessary preliminaries on non-probabilistic programs. We then present our approach and its novel aspects in the following order: first we introduce the technique of program reversal in Section 5.3; then we present a new certificate for non-termination based on so-called *backward invariants* in Section 5.4, as well as an invariant generation-based algorithm for automating the computation of this certificate in Section 5.5. We present our experiments in Section 5.6. Section 5.7 discusses related work.

5.2 Preliminaries for Non-probabilistic Programs

Since we only study non-probabilistic programs in this chapter, we start by specifying the class of programs that we consider and define a formal model for their analysis.

Syntax. We consider simple imperative arithmetic programs with polynomial integer arithmetic and with non-determinism. They consist of standard programming constructs such as conditional branching, while-loops and (deterministic) variable assignments. In addition, we allow constructs for non-deterministic assignments of the form $x := \textbf{ndet}()$,

which assign any integral value to x. The adjective *polynomial* refers to the fact that all arithmetic expressions are polynomials in program variables.

Example 5.2.1 (Running example)**.** *Fig. 5.1 shows a program which will serve as our running example in this chapter. The second line contains a non-deterministic assignment, in which any integral value can be assigned to the variable x.*

Removing non-deterministic branching. We may *without loss of generality* assume that non-determinism does not appear in branching: for the purpose of termination analysis, one can replace each non-deterministic branching with a non-deterministic assignment. Indeed, non-deterministic branching in programs is given by a command **if** $*$ **then**, meaning that the control-flow can follow any of the two subsequent branches. By introducing an auxiliary program variable $x_{\textbf{ndet}}$ and replacing each command **if** $*$ **then** with two commands

$$x_{\textbf{ndet}} := \textbf{ndet}()$$
$$\textbf{if } x_{\textbf{ndet}} \geq 0 \textbf{ then}$$

we obtain a program which terminates on every input if and only if the original program does. This removal is done for the sake of easier presentation and neater definition of the *resolution of non-determinism*, see Section 5.5.1.

Valuation, predicate, assertion. Similarly as in Section 2.2, given a finite set V of program variables, we say that a *variable valuation* of V is $\mathbf{x} \in \mathbb{Z}^{|V|}$ since in this section we consider integer programs. A *predicate* over a set of variables V is again a Boolean combination of atomic predicates of the form $E \leq E'$, where E, E' are arithmetic expressions over V. We need not differentiate between non-strict and strict inequalities since we work over integer arithmetic. An *assertion* is a conjunction of atomic predicates of the form $E \leq E'$ (so without disjunctions). We write $\mathbf{x} \models \phi$ to denote that the predicate ϕ given by a formula over program variables is satisfied by substituting values in $\mathbf{x}$ for corresponding variables in ϕ. For a predicate ϕ, we define $\neg\phi$ for the predicate obtained by the logical negation of ϕ.

Transition system. We model programs using transition systems [CSS03]. Transition systems allow for a more streamlined reasoning about program transitions compared to pCFGs, in which we differentiated between different types of locations in order to simplify reasoning about probabilistic instructions.

Defintion 5.2.1 (Transition system)**.** *A* transition system is an ordered tuple $\mathcal{T} = (L, V, \ell_{init}, \Theta_{init}, \mapsto)$, where L is a finite set of *locations*; V is a finite set of *program variables*; ℓ_{init} is the *initial location*; Θ_{init} is the set of *initial variable valuations*; and $\mapsto \subseteq L \times L \times \mathcal{P}(\mathbb{Z}^{|V|} \times \mathbb{Z}^{|V|})$ is a finite set of *transitions*. Each transition is defined

as an ordered triple $\tau = (l, l', \rho_\tau)$, with l its *source* and l' the *target location*, and the *transition relation* $\rho_\tau \subseteq \mathbb{Z}^{|V|} \times \mathbb{Z}^{|V|}$. The transition relation is usually given by an assertion over V and V', where V represents the source-state variables and V' the target-state variables.

Each program P naturally defines a transition system $\mathcal{T}$, with each transition relation given by an assertion over program variables. Its construction is standard and we omit it. The only difference is that here ℓ_{init} will correspond to the first non-assignment command in the program code, whereas the sequence of assignments preceding ℓ_{init} specifies Θ_{init} (unspecified variables may take any value). Hence Θ_{init} will also be an assertion. For transition systems derived from programs, we assume the existence of a special *terminal location* ℓ_{out}, which represents a "final" line of the program code. It has a single outgoing transition which is a self-loop with a transition relation $\rho = \{(\mathbf{x}, \mathbf{x}) \mid \mathbf{x} \in \mathbb{Z}^{|V|}\}$.

Similarly to pCFGs in Section 2.2, a *state* of a transition system $\mathcal{T}$ is an ordered pair $(l, \mathbf{x})$ where l is a location and $\mathbf{x}$ is a vector of variable valuations. A state $(l', \mathbf{x}')$ is a *successor* of a state $(l, \mathbf{x})$ if there is a transition $\tau = (l, l', \rho_\tau)$ with $(\mathbf{x}, \mathbf{x}') \in \rho_\tau$. The self-loop at ℓ_{out} allows us to without loss of generality assume that each state has at least one successor in a transition system $\mathcal{T}$ derived from a program. Given a state $\mathbf{c}$, a *finite path from* $\mathbf{c}$ in $\mathcal{T}$ is a finite sequence of states $\mathbf{c} = (l_0, \mathbf{x}_0), \ldots, (l_k, \mathbf{x}_k)$ where for each $0 \leq i < k$ we have that $(l_{i+1}, \mathbf{x}_{i+1})$ is a successor of $(l_i, \mathbf{x}_i)$. A *run* (or *execution*) *from* $\mathbf{c}$ in $\mathcal{T}$ is an infinite sequence of states whose every finite prefix is a finite path from $\mathbf{c}$. A state is said to be *initial* if it belongs to the set $\{(\ell_{init}, \mathbf{x}) \mid \mathbf{x} \models \Theta_{init}\}$. A state $(l, \mathbf{x})$ is *reachable from* $\mathbf{c}$ if there is a finite path from $\mathbf{c}$ with the last state $(l, \mathbf{x})$. When we omit specifying the state $\mathbf{c}$, we refer to a finite path, execution and reachability from some initial state. A state $(l, \mathbf{x})$ is said to be *terminal* if $l = \ell_{out}$.

Example 5.2.2. *The transition system for our running example is presented in Fig. 5.2 left. It contains 6 locations $L = \{l_0, l_1, l_2, l_3, l_4, \ell_{out}\}$ with $\ell_{init} = l_0$, and two program variables $V = \{x, y\}$. Since there are no assignments preceding the initial program location, we have $\Theta_{init} = \mathbb{Z}^2$. Locations are depicted by labeled circles, transitions by directed arrows between program locations and their transition relations are given in the associated rectangular boxes.*

Invariants and inductive predicate functions. Given a transition system $\mathcal{T}$, a *predicate function* is a map I assigning to each location in $\mathcal{T}$ a predicate over the program variables. A predicate function naturally defines a set of states in $\mathcal{T}$ and we will freely interchange between the two notions. A predicate function is *of type-*(c, d) if it assigns to each program location a predicate which is a disjunction of d assertions, each being a conjunction of c inequalities. For a predicate function I, we define the

l_0: **while** $x \geq 9$ **do**
l_1: $x := \textbf{ndet}()$
l_2: $y := 10 \cdot x$
l_3: **while** $x \leq y$ **do**
l_4: $x := x + 1$
od
od

Figure 5.1: Our running example in this chapter.

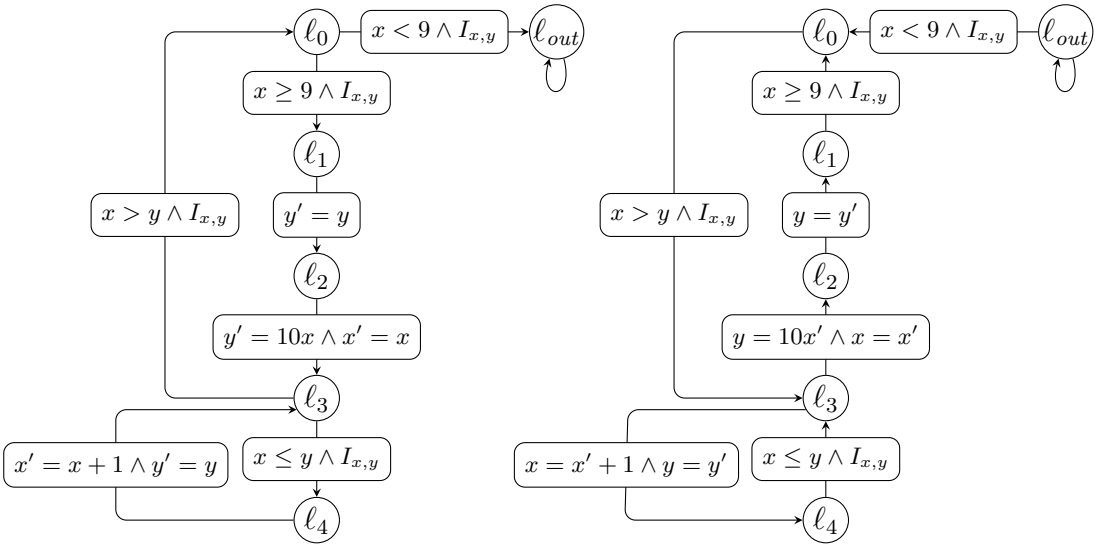

Figure 5.2: Transition system (left) and its reversed transition system (right) associated to our running example in Fig. 5.1. $I_{x,y}$ denotes $x' = x \wedge y' = y$ and is used for readability.

complement predicate function $\neg I$ as $(\neg I)(l) = \neg I(l)$ for each location l.
A predicate function I is said to be an *invariant* if for every reachable state $(l, \mathbf{x})$ in $\mathcal{T}$, we have $\mathbf{x} \models I(l)$. Intuitively, invariants are over-approximations of the set of reachable states in the transition system. A predicate function is *inductive* if it is inductive with respect to every transition $\tau = (l, l', \rho_\tau)$, i.e. if for any pair of states $(l, \mathbf{x})$ and $(l', \mathbf{x}')$ with $\mathbf{x} \models I(l)$ and $(\mathbf{x}, \mathbf{x}') \in \rho_\tau$, we also have $\mathbf{x}' \models I(l')$.

Termination problem. Given a program and its transition system $\mathcal{T}$, we say that a run reaching ℓ_{out} is *terminating*. The program is said to be *terminating* if every run in $\mathcal{T}$ is terminating. Otherwise it is said to be *non-terminating*. One witness to non-termination can be a state that is reachable but from which there are no terminating executions. We call such state *diverging*.

Example 5.2.3. *Consider again the running example in Fig. 5.1 and its transition system in Fig. 5.2. For any initial state with $x \geq 9$, executions that always assign $x := 9$ when passing the non-deterministic assignment are non-terminating. On the other hand, the execution that assigns $x := 0$ in the non-deterministic assignment enters the outer loop only once and then terminates. Thus, no initial state is diverging. One can similarly check that other states are also not diverging.*

5.3 Transition System Reversal

We now show that it is possible to "reverse" a transition system by reversing its transitions. This construction is the core concept of our approach to proving non-termination, since states in the program from which ℓ_{out} is reachable will be precisely those states which can be reached from ℓ_{out} in the reversed transition system. We then present a sound and complete certificate for non-termination based on this construction.

Defintion 5.3.1 (Reversed transition system)**.** *Given a transition system $\mathcal{T} = (L, V, \ell_{init}, \Theta_{init}, \mapsto)$ and a transition $\tau = (l, l', \rho_\tau) \in \mapsto$, let*

$$\rho'_\tau = \{(\mathbf{x}', \mathbf{x}) \mid (\mathbf{x}, \mathbf{x}') \in \rho_\tau\}.$$

If ρ_τ is given by an assertion over $V \cup V'$, ρ'_τ is obtained from ρ_τ by replacing each unprimed variable in the defining assertion for ρ_τ with its primed counterpart, and vice-versa. Then for an assertion Θ, we define the reversed transition system *of $\mathcal{T}$ with initial variable valuations Θ as a tuple $\mathcal{T}^{r,\Theta} = (L, V, \ell_{out}, \Theta, \mapsto^r)$, where $\mapsto^r = \{(l', l, \rho'_\tau) \mid (l, l', \rho_\tau) \in \mapsto\}$.*

Note that this construction satisfies Definition 5.2.1 and thus yields another transition system. All notions that were defined before (e.g. state, finite path, etc.) are defined analogously for the reversed transition systems.

Example 5.3.1. *Fig. 5.2 right shows the reversed transition system $\mathcal{T}^{r,\Theta}$ of the program in Fig. 5.1. Note that for every transition τ in $\mathcal{T}$ for which ρ_τ is given by a conjunction of an assertion over unprimed program variables and $x' = x \wedge y' = y$, after reversing we obtain the conjunction of the same assertion just now over primed variables and $x' = x \wedge y' = y$. Hence, for such τ the transition relation is invariant under reversing. For example, a transition from l_0 to l_1 in $\mathcal{T}$ has transition relation $x \geq 9 \wedge x' = x \wedge y' = y$ so the reversed transition has transition relation $x' \geq 9 \wedge x = x' \wedge y = y'$. As $x' = x$, this is the same relation as prior to reversal.*

The following lemma is the key property of this construction.

Lemma 5.3.2 (Key property of reversed transition systems)**.** *Let $\mathcal{T}$ be a transition system, Θ an assertion and $\mathcal{T}^{r,\Theta}$ the reversed transition system of $\mathcal{T}$ with initial variable valuations Θ. Let $\mathbf{c}$ and $\mathbf{c}'$ be two states. Then $\mathbf{c}'$ is reachable from $\mathbf{c}$ in $\mathcal{T}$ if and only if $\mathbf{c}$ is reachable from $\mathbf{c}'$ in $\mathcal{T}^{r,\Theta}$.*

Proof. We prove that if $\mathbf{c}'$ is reachable from $\mathbf{c}$ in $\mathcal{T}$ then $\mathbf{c}$ is reachable from $\mathbf{c}'$ in $\mathcal{T}^{r,\Theta}$, the other direction follows analogously. Suppose that $\mathbf{c} = (l_0, \mathbf{x}_0), (l_1, \mathbf{x}_1), \ldots, (l_k, \mathbf{x}_k) = \mathbf{c}'$ is a path from $\mathbf{c}$ to $\mathbf{c}'$ in $\mathcal{T}$. Then for each $0 \leq i < k$ there is a transition $\tau_i = (l_i, l_{i+1}, \rho_{\tau_i})$ in $\mathcal{T}$ for which $(\mathbf{x}_i, \mathbf{x}_{i+1}) \in \rho_{\tau_i}$. But then $(\mathbf{x}_{i+1}, \mathbf{x}_i) \in \rho'_{\tau_i}$ and $\tau_i^r = (l', l, \rho'_{\tau_i})$, hence $(l_i, \mathbf{x}_i)$ is a successor of $(l_{i+1}, \mathbf{x}_{i+1})$ in $\mathcal{T}^{r,\Theta}$. Thus $\mathbf{c}' = (l_k, \mathbf{x}_k), (l_{k-1}, \mathbf{x}_{k-1}), \ldots, (l_0, \mathbf{x}_0) = \mathbf{c}$ is a finite path in $\mathcal{T}^{r,\Theta}$, proving the claim. □

Backward Invariants. Lemma 5.3.2 implies that generating invariants for the reversed transition system $\mathcal{T}^{r,\Theta}$ provides a way to over-approximate the set of states in $\mathcal{T}$ from which some state in the set $\{(\ell_{out}, \mathbf{x}) \mid \mathbf{x} \models \Theta\}$ is reachable. This motivates the notion of a *backward invariant*, which will be important in what follows.

Defintion 5.3.3 (Backward invariant)**.** *For a transition system $\mathcal{T}$ and an assertion Θ, we say that the predicate function BI is a* backward invariant *in $\mathcal{T}^{r,\Theta}$ if it is an invariant in $\mathcal{T}^{r,\Theta}$. The word backward is used to emphasize that we are working in the reversed transition system.*

We conclude this section with a theorem illustrating the behavior of inductive predicate functions under program reversal.

Theorem 5.3.4. *Let $\mathcal{T}$ be a transition system, Θ an assertion, I a predicate function and $\mathcal{T}^{r,\Theta}$ the reversed transition system. Then I is inductive in $\mathcal{T}$ if and only if $\neg I$ is inductive in $\mathcal{T}^{r,\Theta}$.*

Proof. We show that I being inductive in $\mathcal{T}$ implies that $\neg I$ is inductive in $\mathcal{T}^{r,\Theta}$. The other direction of the lemma follows analogously.
Let $\tau^r = (l', l, \rho'_\tau)$ be a transition in $\mathcal{T}^{r,\Theta}$ obtained by reversing $\tau = (l, l', \rho_\tau)$ in $\mathcal{T}$. Assume that $\mathbf{x}' \in \neg I(l')$. To show inductiveness of $\neg I$ in the reversed transition system, we take a successor $(l, \mathbf{x})$ of $(l', \mathbf{x}')$ in the reversed transition system with $(\mathbf{x}', \mathbf{x}) \in \rho'_\tau$, and we need to show that $\mathbf{x} \in \neg I(l)$. By definition of the reversed transition we have $(\mathbf{x}, \mathbf{x}') \in \rho_\tau$. So, if on the contrary we had $\mathbf{x} \in I(l)$, inductiveness of I in $\mathcal{T}$ would imply that $\mathbf{x}' \in I(l')$. This would contradict the assumption that $\mathbf{x}' \in \neg I(l')$. Thus, we must have $\mathbf{x} \in \neg I(l)$, and $\neg I$ is inductive in $\mathcal{T}^{r,\Theta}$. □

5.4 Sound and Complete Certificate for Non-termination

Lemma 5.3.2 indicates that reversed transition systems are relevant for the termination problem, as they provide means to describe states from which the terminal location can be reached. We now introduce the BI-*certificate for non-termination*, based on the reversed transition systems and backward invariants. We show that it is both *sound* and *complete* for proving non-termination and hence characterizes it (i.e. a program is non-terminating if and only if it admits the certificate). This is done by establishing a connection to recurrence sets [GHM+08, CCF+14], a notion which provides a necessary and sufficient condition for a program to be non-terminating.

Recurrence set. A *recurrence set* [GHM+08] in a transition system $\mathcal{T}$ is a non-empty set of states $\mathcal{G}$ which (1) contains some state reachable in $\mathcal{T}$, (2) every state in $\mathcal{G}$ has at least one successor in $\mathcal{G}$, and (3) contains no terminal states. The last condition was not present in [GHM+08] and we add it to account for the terminal location and the self-loop at it, but the definitions are easily seen to be equivalent. In [GHM+08], it is shown that a program is non-terminating if and only if its transition system contains a recurrence set. The work in [CCF+14] notes that one may without loss of generality restrict attention to recurrence sets which contain some initial state (which they call *open recurrence sets*). Indeed, to every recurrence set one can add states from some finite path reaching it to obtain an open recurrence set, and there is at least one such path since each recurrence set contains a reachable state.

Closed recurrence set. A *closed recurrence set* [CCF+14] is an open recurrence set $\mathcal{C}$ with the additional property of being inductive, i.e. for every state in $\mathcal{C}$ *each* of its successors is also contained in $\mathcal{C}$. The work [CCF+14, Theorems 1 and 2] shows that closed recurrence sets can be used to define a sound and complete certificate for non-termination, which we describe next. Call $U = (L, V, \ell_{init}, \Theta_{init}, \mapsto_U)$ an *under-approximation* of $\mathcal{T} = (L, V, \ell_{init}, \Theta_{init}, \mapsto)$ if for every $(l, l', \rho^u_\tau) \in \mapsto_U$ there exists $(l, l', \rho_\tau) \in \mapsto$ with $\rho^u_\tau \subseteq \rho_\tau$. Then $\mathcal{T}$ contains an open recurrence set if and only if there is an under-approximation U of $\mathcal{T}$ and a closed recurrence set in U.

Proper under-approximations. We introduce a notion of proper under-approximation. An under-approximation U of $\mathcal{T}$ is *proper* if every state which has a successor in $\mathcal{T}$ also has at least one successor in U. This is a new concept and restricts general under-approximations, but it will be relevant in defining the BI-certificate for non-termination and establishing its soundness and completeness. The next lemma is technical and shows that closed recurrence sets in proper under-approximations are sound and complete for proving non-termination.

Lemma 5.4.1. *Let P be a non-terminating program and $\mathcal{T}$ its transition system. Then there exist a proper under-approximation U of $\mathcal{T}$ and a closed recurrence set $\mathcal{C}$ in U.*

Proof. Since P is non-terminating, we have that $\mathcal{T} = (L, V, \ell_{init}, \Theta_{init}, \mapsto)$ admits an under-approximation $U' = (L, V, \ell_{init}, \Theta_{init}, \mapsto^{U'})$ and a closed recurrence set $\mathcal{C}$ in U'. Define the under-approximation $U = (L, V, \ell_{init}, \Theta_{init}, \mapsto^{U})$ of $\mathcal{T}$ by defining ρ_τ^U for each $\tau = (l, l', \rho_\tau) \in \mapsto$ as follows:

$$\rho_\tau^U = \rho_\tau^{U'} \cup \{(\mathbf{x}, \mathbf{x}') \mid (\mathbf{x}, \mathbf{x}') \in \rho_\tau, (l, \mathbf{x}) \notin \mathcal{C}\}.$$

Thus we extend $\rho_\tau^{U'}$ in order to make each state outside $\mathcal{C}$ have the same set of successors as in $\mapsto$. We claim that the under-approximation U of $\mathcal{T}$ is proper and that $\mathcal{C}$ is a closed recurrence set in U, proving the lemma. Indeed, by the construction of U every state outside $\mathcal{C}$ which has a successor in $\mathcal{T}$ also has at least one successor in U. For a state in $\mathcal{C}$ this is immediate since its set of successors in U is the same in U' and $\mathcal{C}$ is a closed recurrence set in U'. Hence U is proper. To see that $\mathcal{C}$ is a closed recurrence set in U, note again that every state in $\mathcal{C}$ has the same successor set in U' and in U, thus it has at least one successor in U which is in $\mathcal{C}$, and also all of its successors in U are in $\mathcal{C}$. Therefore, as $\mathcal{C}$ contains an initial state in $\mathcal{T}$ and thus in U, the claim follows. □

BI-certificate for non-termination. We introduce and explain how backward invariants in combination with proper under-approximations can be used to characterize non-termination. Suppose P is a program we want to show is non-terminating, and $\mathcal{T}$ is its transition system. Let $Reach_{\mathcal{T}}(\ell_{out})$ be the set of variable valuations of all reachable terminal states in $\mathcal{T}$. A *BI-certificate for non-termination* will consist of an ordered triple (U, BI, Θ) of a proper under-approximation U of $\mathcal{T}$, a predicate function BI and an assertion Θ such that

- $\Theta \supseteq Reach_{\mathcal{T}}(\ell_{out})$;
- BI is an inductive backward invariant in $U^{r,\Theta}$;
- BI is not an invariant in $\mathcal{T}$.

Theorem 5.4.2 (Soundness of our certificate)**.** *Let P be a program and $\mathcal{T}$ its transition system. If there exists a BI-certificate (U, BI, Θ) in $\mathcal{T}$, then P is non-terminating.*

Proof. Consider a predicate function $\neg BI$. Note that in a transition system defined by a program, every state has at least one successor. So as U is proper, every state in $\neg BI$ has at least one successor in U. On the other hand, by Theorem 5.3.4 we have that $\neg BI$ is inductive in U since BI is inductive in $U^{r,\Theta}$. Hence for every state

in $\neg BI$, all of its successors in U are also in $\neg BI$. We also know that BI is not an invariant in $\mathcal{T}$, so there exists a reachable state $\mathbf{c}$ in $\mathcal{T}$ which is contained in $\neg BI$. We use it to construct a non-terminating execution. Pick any finite path $\mathbf{c}_0, \mathbf{c}_1, \ldots, \mathbf{c}_k = \mathbf{c}$ in $\mathcal{T}$, which exists by reachability. Since terminal states do not have non-terminal successors, none of the states along this finite path is terminal. Then, starting from $\mathbf{c}$ we may inductively pick successors in U which are in $\neg BI$. This is possible from what we showed above. Moreover, none of the picked successors can be terminal all reachable terminal states are contained in BI. Thus we obtain a non-terminating run, as wanted. □

Example 5.4.1. *Consider again the running example and its transition system $\mathcal{T}$ presented in Fig. 5.1 and Fig. 5.2, respectively. Let U be the under-approximation of $\mathcal{T}$ defined by restricting the transition relation of the non-deterministic assignment $x := \textbf{\textit{ndet}}()$ as $\rho_\tau^U = \{(x, y, x', y') \mid x' = 9, y' = y\}$. Intuitively, U is a transition system of the program obtained by replacing the non-deterministic assignment in P with $x := 9$. Define a predicate function BI as*

$$BI(l) = \begin{cases} (1 \geq 0) & \text{if } l = \ell_{out} \\ (x \leq 8) & \text{if } l \in \{l_0, l_2, l_3, l_4\} \\ (-1 \geq 0) & \text{if } l = l_1, \end{cases}$$

i.e. $BI(l_1)$ is empty, and let $\Theta = \mathbb{Z}^2$. $U^{r,\Theta}$ can be obtained from $\mathcal{T}^{r,\Theta}$ by replacing the transition relation from l_2 to l_1 with $x = 9 \wedge y = y'$ in Fig. 5.2 right. Then U is proper, and BI is an inductive backward invariant for $U^{r,\Theta}$ since no transition can increase x. On the other hand, $(l_0, 9, 0)$ is reachable in $\mathcal{T}$ but not contained in BI, thus BI is not an invariant in $\mathcal{T}$. Hence (U, BI, Θ) is a BI-certificate for non-termination and the program is non-terminating.

By making a connection to closed recurrence sets, the following theorem shows that backward invariants in combination with proper under-approximations of $\mathcal{T}$ also provide a *complete characterization* of non-termination.

Theorem 5.4.3 (Complete characterization of non-termination)**.** *Let P be a non-terminating program with transition system $\mathcal{T}$. Then $\mathcal{T}$ admits a proper under-approximation U and a predicate function BI such that BI is an inductive backward invariant in the reversed transition system $U^{r,\mathbb{Z}^{|V|}}$, but not an invariant in $\mathcal{T}$.*

Proof. Since P is non-terminating, from Lemma 5.4.1 we know that $\mathcal{T}$ admits a proper under-approximation U and a closed recurrence set $\mathcal{C}$ in U. For each location l in $\mathcal{T}$, let $\mathcal{C}(l) = \{\mathbf{x} \mid (l, \mathbf{x}) \in \mathcal{C}\}$. Define the predicate function BI as $BI(l) = \neg\mathcal{C}(l)$ for each l. We claim that BI is an inductive backward invariant in $U^{r,\mathbb{Z}^{|V|}}$, but not an

invariant in $\mathcal{T}$. By definition of closed recurrence sets, BI contains all terminal states in $\mathcal{T}$, i.e. initial states of $U^{r,\mathbb{Z}^{|V|}}$. On the other hand, for every state in $\mathcal{C}$ all of its successors in U are also contained in $\mathcal{C}$. Thus, as a predicate function $\mathcal{C}$ is inductive, hence by Theorem 5.3.4 $BI = \neg\mathcal{C}$ is inductive in $U^{r,\mathbb{Z}^{|V|}}$. This shows that BI is an inductive backward invariant in $U^{r,\mathbb{Z}^{|V|}}$. On the other hand, as a closed recurrence set $\mathcal{C}$ contains an initial state in $\mathcal{T}$, BI does not contain all reachable states and is thus not an invariant in $\mathcal{T}$. The claim of the theorem follows. □

Remark 5.4.1 (Connection to the *pre*-operator)**.** *There is a certain similarity between reversal of an individual transition and application of the* pre*-operator, the latter being a well known concept in program analysis. However, in our approach we introduce reversed transition systems which are obtained by reversing* all transitions *(hence the name "program reversal"). This allows us using black-box invariant generation techniques as a* one-shot *method of computing sets from which a terminal location can be reached, as presented in the next section. This is in contrast to approaches which rely on an iterative application of the* pre*-operator.*

5.5 Algorithm for Proving Non-termination

We now present our algorithm for proving non-termination based on program reversing and BI-certificates introduced in Section 5.4. It uses a black box constraint solving-based method for generating (possibly disjunctive) inductive invariants, as in [CSS03, GSV08, KBBR17, KCBR18, HOPW18, RK04, RK07, CFGG20]. This is a classical approach to invariant generation and it fixes a template for the invariant (i.e. a type-(c, d) predicate function with polynomial expressions as well as an upper bound D on the degree of polynomials, where c, d and D are provided by the user), introduces a fresh variable for each template coefficient, and encodes invariance and inductiveness conditions as existentially quantified constraints on template coefficient variables. The obtained system is then solved and any solution yields an inductive invariant. Moreover, the method is relatively complete [RK04, RK07, CFGG20] in the sense that every inductive invariant of the fixed template and maximal polynomial degree is a solution to the system of constraints. Efficient practical approaches to polynomial inductive invariant generation have been presented in [KBBR17, KCBR18].

We first introduce *resolution of non-determinism* which induces a type of proper under-approximations of the program's transition system of the form that allows searching for them via constraint solving. We then proceed to our main algorithm. In what follows, P will denote a program with polynomial arithmetic and $\mathcal{T} = (L, V, \ell_{init}, \Theta_{init}, \mapsto)$ will be its transition system. Furthermore, in what follows we assume that all predicates and assertions are defined in terms of polynomial expressions over program variables.

5.5.1 Resolution of Non-determinism

As we saw in Example 5.2.3, there may exist non-diverging program states which become diverging when supports of non-deterministic assignments are restricted to suitably chosen subsets. Here we define one such class of restrictions which "resolves" each non-deterministic assignment by replacing it with a polynomial expression over program variables. Such resolution ensures that the resulting under-approximation of the program's transition relation is proper. Let $T_{NA} \subseteq \mapsto$ be the set of transitions corresponding to non-deterministic assignments in P.

Defintion 5.5.1 (Resolution of non-determinism)**.** *A* resolution of non-determinism *for $\mathcal{T}$ is a map R^{NA} which to each $\tau \in T_{NA}$ assigns a polynomial expression $R^{NA}(\tau)$ over program variables. It naturally defines a* restricted transition system *$\mathcal{T}_{R^{NA}}$ which is obtained from $\mathcal{T}$ by letting the transition relation of $\tau \in T_{NA}$ corresponding to an assignment $x :=$ **ndet**$()$ be*

$$\rho_\tau^{R^{NA}}(\mathbf{x}, \mathbf{x}') := (x' = R^{NA}(\tau)(\mathbf{x})) \wedge \bigwedge_{y \in V \setminus \{x\}} y' = y.$$

Note that $\mathcal{T}_{R^{NA}}$ is a proper under-approximation of $\mathcal{T}$. If there exists a resolution of non-determinism R^{NA} and a state $\mathbf{c}$ which is reachable in $\mathcal{T}$ but from which no execution in $\mathcal{T}_{R^{NA}}$ terminates, then any such execution is non-terminating in $\mathcal{T}$ as well. We say that any such state $\mathbf{c}$ is *diverging with respect to (w.r.t.) R^{NA}*.

Example 5.5.1. *Looking back at the program in Figure 3.2, define a resolution of non-determinism R^{NA} to assign constant expression 9 to the non-deterministic assignment $x :=$ **ndet**$()$. Then every initial state with $x \geq 9$ becomes diverging w.r.t. R^{NA}.*

5.5.2 Algorithm

Main idea. To prove non-termination, our algorithm uses a constraint solving approach to find a BI-certificate. It searches for a resolution of non-determinism R^{NA}, a predicate function BI and an assertion Θ such that:

1. $\Theta \supseteq Reach_{\mathcal{T}}(\ell_{out})$ (recall that $Reach_{\mathcal{T}}(\ell_{out})$ is the set of variable valuations of all reachable terminal states in $\mathcal{T}$);
2. BI is an inductive backward invariant for the reversed transition system $\mathcal{T}_{R^{NA}}^{r,\Theta}$;
3. BI is not an invariant for $\mathcal{T}$.

Need for inductive invariants and safety checking. Using the aforementioned black box invariant generation, our algorithm encodes the conditions on R^{NA}, BI, and Θ as polynomial constraints and then solves them. However, the method is only able to generate *inductive* invariants, which is to say that encoding "BI is not an invariant for $\mathcal{T}$" is not possible. Instead, we modify the third requirement on BI above to get:

1. $\Theta \supseteq Reach_{\mathcal{T}}(\ell_{out})$;
2. BI is an inductive backward invariant for $\mathcal{T}^{r,\Theta}_{R^{NA}}$;
3. BI is not an inductive invariant for $\mathcal{T}$.

The third requirement does not guarantee that we get a proper BI-certificate. However it guides invariant generation to search for BI which is less likely to be an invariant for $\mathcal{T}$. It follows that the algorithm needs to do additional work to ensure that the triple (R^{NA}, BI, Θ) is a BI-certificate.

Splitting the algorithm into two checks. The predicate function BI is not an inductive invariant for $\mathcal{T}$ if and only if it has one of the following properties: either it does not contain some initial state or is not inductive with respect to some transition in $\mathcal{T}$. For each of these two properties, we can separately compute BI satisfying it and the properties (1) and (2) above, followed by a check whether the computed BI indeed proves non-termination. We refer to these two independent computations as two checks of our algorithm:

- *Check 1* - the algorithm checks if there exist R^{NA}, BI and Θ as above so that BI does not contain some initial state and conditions (1) and (2) are satisfied. By Theorem 5.3.4, BI is inductive for $\mathcal{T}^{r,\Theta}_{R^{NA}}$ if and only if the complement $\neg BI$ is an inductive predicate function for $\mathcal{T}_{R^{NA}}$. Moreover, since $\neg BI$ contains an initial state there is no need for an additional reachability check to conclude that BI is not an invariant for $\mathcal{T}$. Hence by fixing $\Theta = \mathbb{Z}^{|V|}$, to prove non-termination it suffices to check if there exist a resolution of non-determinism R^{NA}, a predicate function I and an initial state $\mathbf{c}$ in $\mathcal{T}$ such that I contains $\mathbf{c}$, I is inductive for $\mathcal{T}_{R^{NA}}$ and $I(\ell_{out}) = \emptyset$.
- *Check 2* - the algorithm checks if there exist R^{NA}, Θ and BI as above so that BI is not inductive in $\mathcal{T}$ and conditions (1) and (2) are satisfied. If a solution is found, the algorithm still needs to find a state in $\neg BI$ which is reachable in $\mathcal{T}$, via a call to a safety prover.

Algorithm 5.1: Algorithm for proving non-termination

input : A program P, its transition system $\mathcal{T}$, predicate function template size (c, d), maximal polynomial degree D.
output : Proof of non-termination if found, otherwise "Unknown"

```
1  set a template for each polynomial defined by resolution of non-determinism R^NA
2  construct restricted transition system T_{R^NA}
3  set templates for state c and for an invariant I of type-(c, d)
4  encode Φ_1 = φ_c ∧ φ_{I,R^NA}
5  if Φ_1 feasible then return Non-termination
6  else
7      set templates for invariant Ĩ of type-(c, 1) and for a backward invariant BI
         of type-(c, d)
8      construct reversed transition system T^{r,Ĩ(ℓ_out)}_{R^NA}
9      foreach τ ∈ ↦ do  set templates for x_τ, x'_τ
10     encode Φ_2 = φ_Ĩ ∧ φ_{BI,R^NA} ∧ ⋁_{τ∈↦} φ_τ
11     if Φ_2 feasible then
12         if ∃ (l, x) Reachable in T with x ⊨ ¬BI(l) then return
             Non-termination
13         else return Unknown
14     end
15     else return Unknown
16 end
```

Algorithm summary. As noted at the beginning of Section 5.5, the invariant generation method first needs to fix a template for the predicate function and the maximal polynomial degree. Thus our algorithm is parametrized by c and d which are bounds on the template size of predicate functions (d being the maximal number of disjunctive clauses and c being the maximal number of conjunctions in each clause), and by an upper bound D on polynomial degrees. The algorithm consists of two checks, which can be executed either sequentially or in parallel:

Check 1 - the algorithm checks if there exist a resolution of non-determinism R^{NA}, a predicate function I and an initial state $\mathbf{c}$ such that (1) I is an inductive invariant in $\mathcal{T}_{R^{NA}}$ for the single initial state $\mathbf{c}$, and (2) $I(\ell_{out}) = \emptyset$. To do this, we fix a template for each of R^{NA}, I and $\mathbf{c}$, and encode these properties as polynomial constraints:

- For each transition τ in T_{NA}, fix a template for a polynomial $R^{NA}(\tau)$ over program variables of degree at most D. That is, introduce a fresh template variable for each coefficient of such a polynomial.

- Introduce fresh variables $c_1, c_2, \ldots, c_{|V|}$ defining the variable valuation of $\mathbf{c}$. Then substitute these variables into the assertion Θ_{init} specifying initial states in $\mathcal{T}$ to obtain the constraint $\phi_{\mathbf{c}}$ for $\mathbf{c}$ being an initial state.
- Fix a template for the predicate function I of type-(c, d) and maximal polynomial degree D. The fact that I is an inductive invariant for $\mathcal{T}_{R^{NA}}$ with the single initial state $\mathbf{c}$ and $I(\ell_{out}) = \emptyset$ is encoded by the invariant generation method (e.g. [CSS03, RK04]) into a constraint $\phi_{I,R^{NA}}$.

The algorithm then tries to solve $\Phi_1 = \phi_{\mathbf{c}} \wedge \phi_{I,R^{NA}}$ using an off-the-shelf SMT solver. If a solution is found, $\mathbf{c}$ is an initial diverging state w.r.t. $\mathcal{T}_{R^{NA}}$, so the algorithm reports non-termination.

Check 2 - the algorithm checks if there exist a resolution of non-determinism R^{NA}, an assertion Θ, a predicate function BI and a transition $\tau \in T_{NA}$ such that (1) $\Theta \supseteq Reach_{\mathcal{T}}(\ell_{out})$, (2) BI is an inductive backward invariant for $\mathcal{T}^{r,\Theta}_{R^{NA}}$, and (3) BI is not inductive w.r.t. τ in $\mathcal{T}$. To encode $\Theta \supseteq Reach_{\mathcal{T}}(\ell_{out})$, we introduce another predicate function $\tilde{I}$ (purely conjunctive for the sake of efficiency), and impose a requirement on it to be an inductive invariant for $\mathcal{T}$. We may then define the initial variable valuations for $\mathcal{T}^{r,\Theta}_{R^{NA}}$ as $\Theta = \tilde{I}(\ell_{out})$. The algorithm introduces fresh template variables for R^{NA}, $\tilde{I}$ and BI, as well as for a pair of variable valuations $\mathbf{x}_\tau$ and $\mathbf{x}'_\tau$ for each transition $\tau = (l, l', \rho_\tau)$ in $\mathcal{T}$ and imposes the following constraints:

- For each transition τ in T_{NA}, fix a template for a polynomial expression $R^{NA}(\tau)$ of degree at most D over program variables.
- Fix a template for the predicate function $\tilde{I}$ of type-$(c, 1)$ (as explained above, for efficiency reasons we make $\tilde{I}$ conjunctive) and impose a constraint $\phi_{\tilde{I}}$ that $\tilde{I}$ is an inductive invariant for $\mathcal{T}$.
- Fix a template for the predicate function BI of type-(c, d) and impose a constraint $\phi_{BI,R^{NA}}$ that BI is an inductive backward invariant for $\mathcal{T}^{r,\tilde{I}(\ell_{out})}_{R^{NA}}$.
- For each transition τ in $\mathcal{T}$, the constraint ϕ_τ encodes non-inductiveness of BI with respect to τ in $\mathcal{T}$:

$$\mathbf{x}, \mathbf{x}' \models BI(l) \wedge \rho_\tau \wedge \neg BI(l').$$

The algorithm then solves $\Phi_2 = \phi_{\tilde{I}} \wedge \phi_{BI,R^{NA}} \wedge \bigvee_{\tau \in \mapsto} \phi_\tau$ by using an SMT-solver. If a solution is found, the algorithm uses an off-the-shelf safety prover to check if there exists a state in $\neg BI$ reachable in $\mathcal{T}$. Such state is then diverging w.r.t. $\mathcal{T}_{R^{NA}}$, so we report non-termination.

The pseudocode for our algorithm is shown in Algorithm 5.1. The following theorem proves soundness of our algorithm.

Theorem 5.5.2 (Soundness)**.** *If Algorithm 5.1 outputs "Non-termination" for some program P, then P is non-terminating.*

Proof. If the algorithm outputs "Non-termination" for an input program P, then it was either able to show that Φ_1 is feasible for P, or that Φ_2 is feasible for P with the subsequent safety check being successful.

Suppose that the algorithm showed that Φ_1 is feasible and found a resolution of non-determinism R^{NA}, an initial state $\mathbf{c}$ in $\mathcal{T}$, and a type-(c, d) predicate function I satisfying properties in Check 1 of the algorithm. We claim that $(\mathcal{T}_{R^{NA}}, \neg I, \mathbb{Z}^{|V|})$ is a BI-certificate for P and thus, by Theorem 5.4.2, P is non-terminating. By Definition 5.5.1, R^{NA} defines a proper under-approximation $\mathcal{T}_{R^{NA}}$ of $\mathcal{T}$. On the other hand, I is inductive for $\mathcal{T}_{R^{NA}}$, so by Theorem 5.3.4, $\neg I$ is inductive for $\mathcal{T}_{R^{NA}}^{r,\mathbb{Z}^{|V|}}$. Moreover, $I(\ell_{out}) = \emptyset$, so $\neg I(\ell_{out}) = \mathbb{Z}^{|V|}$ contains all terminal states and $\neg I$ is an inductive backward invariant for $\mathcal{T}_{R^{NA}}^{r,\mathbb{Z}^{|V|}}$. Finally, $\neg I$ does not contain the initial state $\mathbf{c}$ in $\mathcal{T}$ and is thus not an invariant for $\mathcal{T}$.

Suppose now that Φ_2 was shown to be feasible and that the subsequent safety check was successful. Then the algorithm had to find a resolution of non-determinism R^{NA}, a type-$(c, 1)$ predicate function $\tilde{I}$, a type-(c, d) predicate function BI, and a transition τ in $\mathcal{T}$ satisfying constraints in Check 2 of the algorithm. We claim that $(\mathcal{T}_{R^{NA}}, BI, \tilde{I}(\ell_{out}))$ is a BI-certificate for P and thus, by Theorem 5.4.2, P is non-terminating. Again, by Definition 5.5.1, R^{NA} defines a proper under-approximation $\mathcal{T}_{R^{NA}}$ of $\mathcal{T}$. BI is an inductive backward invariant for $\mathcal{T}_{R^{NA}}^{r,\tilde{I}(\ell_{out})}$ and $\tilde{I}(\ell_{out})$ contains all terminal states in $\mathcal{T}$, as $\tilde{I}$ is an invariant for $\mathcal{T}$. Finally, since the safety check was successful there exists a reachable state in $\mathcal{T}$ contained in $\neg BI$, showing that BI is not an invariant for $\mathcal{T}$. □

Remark 5.5.1 (Algorithm termination)**.** *Our algorithm might not always terminate because either the employed SMT-solver or the safety prover might diverge. Thus, in practice one needs to impose a timeout in order to ensure algorithm termination.*

5.5.3 Demonstration on Examples

We demonstrate our algorithm on two examples illustrating the key aspects. We then present an example demonstrating an application of our method on program whose all non-terminating traces are aperiodic.

Example 5.5.2. *Consider again our running example in Fig. 5.1. We demonstrate that Check 1 of our algorithm can prove that it is non-terminating. Define the resolution*

```
      n := 0, b := 0, u := 0
l0 : while b == 0 and n ≤ 99 do
l1 :     u := ndet()
l2 :     if u ≤ −1 then
l3 :         b := −1
     else if u == 0 then
l4 :         b := 0
l5 :     else b := 1 fi
l6 :     n := n + 1
l7 :     if n ≥ 100 and b ≥ 1 then
l8 :         while true do
l9 :             skip
     od fi od
```

Figure 5.3: An example of a program without an initial diverging state with respect to any resolution of non-determinism that uses polynomials of degree less than 100, but for which Check 2 proves non-termination.

```
l0 : while x ≥ 1 do
l1 :   y := 10 · x
l2 :   while x ≤ y do
l3 :     x := x + 1
       od od
```

Figure 5.4: Example program illustrating aperiodic non-termination.

of non-determinism R^{NA} to assign a constant expression 9 to the non-deterministic assignment, an initial state $\mathbf{c} = (\ell_{init}, 9, 0)$, and a predicate function I as $I(\ell) = (x \geq 9)$ for $\ell \neq \ell_{out}$ and $I(\ell_{out}) = \emptyset$. Then I is an inductive invariant for $\mathcal{T}_{R^{NA}}$ with the initial state $\mathbf{c}$. Thus the system of polynomial constraints constructed by Check 1 is feasible, proving that this program is non-terminating.

Example 5.5.3. *Consider the program in Fig. 5.3. Its initial variable valuation is given by the assertion $(n = 0 \wedge b = 0 \wedge u = 0)$, and a program execution is terminating so long as it does not assign 0 to u in the first 99 iterations of the outer loop, and then at least 1 in the 100-th iteration. Thus, if the initial state was diverging with respect to a resolution of non-determinism which resolves the non-deterministic assignment of u by a polynomial $p(n, b, u)$, this polynomial would need to satisfy $p(n, 0, 0) = 0$ for $n = 0, 1, \ldots, 98$ and $p(99, 0, 0) \geq 1$. Hence, the degree of p would have to be at least 100, and this program has no initial diverging state with respect to any resolution of non-determinism that is feasible to compute by using the Check 1 of our algorithm.*

We now show that Check 2 can prove non-termination of this program using only polynomials of degree 0*, i.e. constant polynomials. Define* R^{NA}*,* $\tilde{I}$*,* BI *and* τ *as follows:*

- R^{NA} *assigns constant expression* 1 *to the assignment of* u *at* ℓ_1*;*
- $\tilde{I}(\ell) = (0 \leq n \leq 100)$ *for each location* ℓ*;*
- BI *is a predicate function defined via*

$$BI(\ell) = \begin{cases} (0 \leq n \leq 100) & \text{if } \ell = \ell_{out} \\ (n \leq 100) & \text{if } \ell = \ell_0 \\ (n \leq 99) \vee (n = 100 \wedge b \leq 0) & \text{if } \ell = \ell_7 \\ (n \leq 98) \vee (n = 99 \wedge b \leq 0) & \text{if } \ell = \ell_6 \\ (n \leq 98) & \text{if } \ell \in \{\ell_1, \ell_5\} \\ (n \leq 99) & \text{if } \ell \in \{\ell_3, \ell_4\} \\ (n \leq 98) \vee (n = 99 \wedge u \leq 0) & \text{if } \ell = \ell_2 \\ (1 \leq 0) & \text{if } \ell \in \{l_8, l_9\}; \end{cases}$$

- τ *is the transition from* ℓ_0 *to* ℓ_1*.*

To show that these R^{NA}*,* $\tilde{I}$*,* BI *and* τ *satisfy each condition in Check 2, we note that:*

1. *The set of variable valuations reachable in the program upon termination is* $(n, b) \in \{(n, b) \mid 1 \leq n \leq 99 \wedge b! = 0\} \cup \{(100, b) \mid b \leq 0\}$*, thus* $\Theta = \tilde{I}(\ell_{out})$ *contains it;*
2. BI *is an inductive backward invariant for* $\mathcal{T}_{R^{NA}}^{r,\tilde{I}(\ell_{out})}$ *(which can be checked by inspection of the reversed transition system in Fig. 5.5;*
3. BI *is not inductive w.r.t.* τ *in* $\mathcal{T}$*, since* $(99, 0, 0) \in BI(\ell_0)$ *but the variable valuation* $(99, 0, 0)$ *obtained by executing* τ *in* $\mathcal{T}$ *is not contained in* $BI(\ell_1)$*.*

Thus, these R^{NA}*,* $\tilde{I}$*,* BI *and* τ *present a solution to the system of constraints defined by Check 2. Since the state* $(\ell_1, 99, 0, 0)$ *is reachable in this program by assigning* $u := 0$ *in the first* 99 *iterations of the outer loop, but* $(99, 0, 0) \notin BI(\ell_1)$*, the safety prover will be able to show that a state in* $\neg BI$ *is reachable. Hence our algorithm is able to prove non-termination.*

Example 5.5.4. *Consider the program in Fig. 5.4. Note that non-determinism in this program is implicit and appears in assigning the initial variable valuation. Moreover, every initial state defines a unique execution which is terminating if and only if the initial value of x is non-positive. We claim that every non-terminating execution in this program is aperiodic. Consider an execution starting in an initial state (ℓ_0, x_1, y_1) with $x_1 \geq 1$. First, note that the inner program loop is terminating thus the execution will execute the outer loop infinitely many times. The i-th iteration of the outer loop in this execution is of the form $\ell_0, \ell_1, (\ell_2, \ell_3)^{(9x_i)}$ where (ℓ_0, x_i, y_i) the state in this execution upon starting the i-th iteration of the outer loop. A simple inductive argument shows that $x_i = 10^{i-1}$ for each $i \in \mathbb{N}$. This proves the claim that each non-terminating execution is indeed aperiodic. We now show that Check 1 of our algorithm can prove that this program is non-terminating. Since there are no non-deterministic assignments, the resolution of non-determinism R^{NA} is trivial. Consider the initial state $\mathbf{c} = (\ell_0, 1, 1)$ and a predicate function I defined via $I(\ell) = (x \geq 1)$ if $\ell \in \{\ell_0, \ell_1, \ell_2, \ell_3\}$, and $I(\ell_{out}) = \emptyset$. Then R^{NA}, I and $\mathbf{c}$ satisfy all the properties in Check 1 of our algorithm, and our algorithm can prove non-termination of the program in Fig. 5.4.*

5.5.4 Relative Completeness

At the beginning of Section 5.5 we noted that constraint solving-based inductive invariant generation is relatively complete [CSS03, GSV08, RK04, RK07, CFGG20], in the sense that whenever there is an inductive invariant representable using the given template, the algorithm will find such an invariant. This means that our algorithm is also relatively complete in checking whether the program satisfies properties encoded as polynomial constraints in Check 1 and Check 2. Since successful Check 1 does not require a subsequent call to a safety prover, it provides to the best of our knowledge the first *relatively complete algorithm* for proving non-termination of programs with polynomial integer arithmetic and non-determinism.

Theorem 5.5.3 (Relative completeness)**.** *Let P be a program with polynomial integer arithmetic and $\mathcal{T}$ its transition system. Suppose that $\mathcal{T}$ admits a proper under-approximation U which restricts each non-deterministic assignment to a polynomial assignment, and a predicate function C which is a closed recurrence set in U. Then for sufficiently high values of parameters c, d and D bounding the template size for invariants and the maximal polynomial degree, our algorithm proves non-termination of the program P.*

While relative completeness guarantees in Theorem 5.5.3 are the first such guarantees for programs with non-determinism, they only apply to non-terminating programs that contain an initial diverging state w.r.t. some resolution of non-determinism. However, Example 5.5.3 shows that finding such a state might require using very high degree

polynomials to resolve non-determinism, and in general such a state need not exist at all in non-terminating programs. In order to ensure catching non-termination bugs in such examples, an algorithm with stronger guarantees is needed. To that end, we propose a modification of our algorithm for programs in which non-determinism appears only in branching. The new algorithm provides *stronger relative completeness guarantees* that can detect non-terminating behavior in programs with no initial diverging states or for which Check 1 is not practical, including the program in Example 5.5.3 (that is, its equivalent version in which non-determinism appears only in branching as we demonstrate in Example 5.5.5).

To motivate this modification, let us look back at the conditions imposed on the predicate function BI by our algorithm. BI is required not to be an invariant, so that $\neg BI$ contains a reachable state. However, this reachability condition cannot be encoded using polynomial constraints, so instead we require that $\neg BI$ is not an inductive invariant, and then employ a safety prover which does not provide any guarantees. Our modification is based on the work of [ACF+21], which presents a relatively complete method for reachability analysis in polynomial programs with non-determinism appearing only in branching.

Relatively complete reachability analysis. We give a high level description of the method in [ACF+21]. Let P be a program with non-determinism appearing only in branching, $\mathcal{T}$ its transition system, and C a set of states defined by a predicate function. The goal of the analysis is to check whether some state in C is reachable in $\mathcal{T}$. The witness for the reachability of C in [ACF+21] consists of (1) an initial state $\mathbf{c}$, (2) a predicate function $C^\diamond$ that contains $\mathbf{c}$, and (3) a polynomial ranking function f^C for $C^\diamond$ with respect to C. A *polynomial ranking function* for $C^\diamond$ with respect to C is a map f^C that to each location $\ell \in L$ assigns a polynomial expression $f^C(l)$ over program variables, such that each state $(l, \mathbf{x}) \in C^\diamond \backslash C$ has a successor $(l', \mathbf{x}') \in C^\diamond$ with

$$f^C(l)(\mathbf{x}) \geq f^C(l')(\mathbf{x}') + 1 \ \wedge \ f^C(l)(\mathbf{x}) \geq 0,$$

where $C^\diamond$ and C are treated as sets of states. Intuitively, this means that for each state $(l, \mathbf{x}) \in C^\diamond \backslash C$, the value of f^C at this state is non-negative and there is a successor of this state in $C^\diamond$ at which the value of f^C decreases by at least 1. If the program admits such a witness, then we may exhibit a path from $\mathbf{c}$ to a state in C by inductively picking either a successor in C (and thus proving reachability), or a successor in $C^\diamond \backslash C$ along which f^C decreases by 1. As the value of f^C in $\mathbf{c}$ is finite and f^C is non-negative on $C^\diamond \backslash C$, decrease can happen only finitely many times and eventually we will have to pick a state in C. It is further shown in [ACF+21] that any reachable C admits a witness in the form of an initial state, a predicate function and a (not necessarily polynomial) ranking function.

For programs with non-determinism appearing only in branching, it is shown in [ACF+21] that all the defining properties of $\mathbf{c}$, $C^\diamond$ and f^C can be encoded using polynomial constraints. Thus [ACF+21] searches for a reachability witness by introducing template variables for $\mathbf{c}$, $C^\diamond$ and f^C, encoding the defining properties using polynomial constraints and then reducing to constraint solving. The obtained constraints are at most quadratic in the template variables, as was the case in our algorithm for proving non-termination. Moreover, their analysis is relatively complete - if a witness of reachability in the form of an initial state $\mathbf{c}$, a predicate function $C^\diamond$ and a polynomial ranking function f^C exists, the method of [ACF+21] will find it.

Modification of our algorithm. The modified algorithm is similar to Check 2, with only difference being that we encode reachability of $\neg BI$ using polynomial constraints instead of requiring it not to be inductive in $\mathcal{T}$. The algorithm introduces a template of fresh variables determining R^{NA}, $\tilde{I}$ and BI. In addition, it introduces a template of fresh variables determining an initial state $\mathbf{c}$, a predicate function $C^\diamond$ and a polynomial ranking function $f^{\neg BI}$. The algorithm then imposes the following polynomial constraints:

- Encode the same conditions on R^{NA}, $\tilde{I}$ and BI as in Check 2 to obtain Φ_{backward}.
- Introduce fresh variables $c_1, c_2, \ldots, c_{|V|}$ defining the variable valuation of $\mathbf{c}$. Then substitute these variables into the assertion Θ_{init} specifying initial states in $\mathcal{T}$ to obtain the constraint $\phi_{\mathbf{c}}$ for $\mathbf{c}$ being an initial state.
- Fix a template for the predicate function $C^\diamond$ of type-(c, d) and maximal polynomial degree D. Encode that $C^\diamond$ contains $\mathbf{c}$ into the constraint $\phi_{\mathbf{c},C^\diamond}$.
- For each location ℓ in $\mathcal{T}$, fix a template for a polynomial $f^{\neg BI}(\ell)$ over program variables of degree at most D. That is, introduce a fresh template variable for each coefficient of such a polynomial.
- Using the method of [ACF+21], for each location ℓ encode the following condition

$$\forall \mathbf{x}.\mathbf{x} \models C^\diamond(\ell) \Rightarrow \mathbf{x} \in \neg BI(\ell) \vee \Big(\Big(\bigvee_{\tau=(\ell,\ell',\rho_\tau)} \mathbf{x}' \models C^\diamond(\ell') \wedge$$
$$\rho_\tau(\mathbf{x},\mathbf{x}') \wedge f^{\neg BI}(\ell)(\mathbf{x}) \geq f^{\neg BI}(\ell')(\mathbf{x}') + 1\Big) \wedge\ f^{\neg BI}(\ell)(\mathbf{x}) \geq 0\Big),$$

  as a polynomial constraint $\phi_{\ell,\mathit{reach}}$. Note that, since we assume that non-determinism appears only in branching and not in variable assignments, for any $\mathbf{x}$ there is at most one variable valuation $\mathbf{x}'$ such that $\rho_\tau(\mathbf{x},\mathbf{x}')$ is satisfied. Thus, the above condition indeed encodes the condition that, if $(\ell,\mathbf{x}) \notin \neg BI$, then at least one successor state satisfies the ranking function property. It is

shown in [ACF+21] that this condition can be encoded into existentially quantified polynomial constraints over template variables, by using analogous semi-algebraic techniques that are used for inductive invariant generation in [CSS03, CFGG20] and which we use for invariant synthesis. We then take $\Phi_{\mathit{reach}} = \wedge_{\ell} \phi_{\ell,\mathit{reach}}$.

The algorithm then tries to solve $\Phi_{\mathit{modified}} = \Phi_{\mathit{backward}} \wedge \phi_{\mathbf{c}} \wedge \phi_{\mathbf{c},C^{\diamond}} \wedge \Phi_{\mathit{reach}}$.

Soundness of the modified algorithm follows the same argument as the proof of Theorem 5.5.2. The following theorem presents the stronger relative completeness guarantees provided by the modified algorithm.

Theorem 5.5.4 (Stronger relative completeness)**.** *Let P be a program with polynomial integer arithmetic, in which non-determinism appears only in branching. Let $\mathcal{T}$ be its transition system. Suppose that $\mathcal{T}$ admits*

1. *a proper under-approximation U restricting each non-deterministic assignment to a polynomial assignment,*
2. *a predicate function $\tilde{I}$ which is an inductive invariant in $\mathcal{T}$,*
3. *a predicate function BI which is an inductive backward invariant in $\mathcal{T}_U^{r,\tilde{I}(\ell_{out})}$, and*
4. *a witness of reachability of $\neg BI$ as in [ACF+21].*

Then for high enough values of c, d and D bounding the template size for invariants and the polynomial degree, our algorithm proves non-termination of the program P.

Remark 5.5.2. *The method of [ACF+21] encodes constraints for programs in which non-determinism appears only in branching, whereas in this work we talked about constraint encoding for programs in which non-determinism appears only in assignments. This is not an issue in the modified algorithm - we can always start with a program in which non-determinism appears only in branching to encode the reachability witness constraints, and then apply the trick from Section 5.2 to replace each non-deterministic branching by an assignment.*

Example 5.5.5. *We show that the relative completeness guarantees of the modified algorithm apply to the program obtained from Fig. 5.3 by replacing the non-deterministic assignment of u and the subsequent conditional branching with the non-deterministic branching given by **if** $*$ **then**. Specifically, the new program is obtained by removing the non-deterministic assignment of u from the program, merging ℓ_1 and ℓ_2 in Fig. 5.5.3 into the new location $\ell_{1,2}$ and replacing the conditional by the non-deterministic branching. The reachability constraints for the modified algorithm are then encoded with respect*

to this new program. On the other hand, to encode the constraints as in Check 2*, we consider the original program in Fig. 5.3.*

To see that this program satisfies the conditions of Theorem 5.5.4, we define R^{NA}*,* $\tilde{I}$ *and* BI *as in Example 5.5.3. Then, one witness of reachability of* $\neg BI$ *(where we identify* $\ell_{1,2}$ *with* ℓ_1*) is defined by* $\mathbf{c} = (\ell_{init}, 0, 0, 0)$*,*

$$C^{\diamond}(\ell) = \begin{cases} (0 \leq n \leq 99 \wedge b = 0 \wedge u = 0) & \text{if } \ell \in \{\ell_0, \ell_{1,2}\} \\ (0 \leq n \leq 98 \wedge b = 0 \wedge u = 0) & \text{if } \ell \in \{\ell_4, \ell_6\} \\ (1 \leq n \leq 99 \wedge b = 0 \wedge u = 0) & \text{if } \ell = \ell_7 \\ (1 \leq 0) & \text{otherwise;} \end{cases}$$

and

$$f^{\neg BI}(\ell, n, b, u) = \begin{cases} 5 \cdot (100 - n) + 3 & \text{if } \ell = \ell_0 \\ 5 \cdot (100 - n) + 2 & \text{if } \ell = \ell_{1,2} \\ 5 \cdot (100 - n) + 1 & \text{if } \ell = \ell_4 \\ 5 \cdot (100 - n) + 0 & \text{if } \ell = \ell_6 \\ 5 \cdot (100 - n) + 4 & \text{if } \ell = \ell_7 \\ 0 & \text{otherwise;} \end{cases}$$

To see that this is indeed the witness of reachability of $\neg BI$*, observe* $C^{\diamond}$ *contains precisely the set of all states along the path from* $\mathbf{c} = (\ell_{init}, 0, 0, 0)$ *to the state* $(\ell_1, 99, 0, 0)$ *in* $\neg BI$ *that we described in Example 5.5.3 (recall, for reachability analysis we identify* ℓ_1 *with* $\ell_{1,2}$ *in the modified program in which non-determinism appears only in branching), and that* $f^{\neg BI}$ *is non-negative along this path and decreases by exactly* 1 *in each step along the path.*

5.6 Experiments

We present a prototype implementation of our algorithm in our tool RevTerm. We follow a standard approach to invariant generation [CSS03, GSV08, CFGG20] which only fixes predicate function templates at cutpoint locations. For safety prover we use CPAchecker [BK11] and for constraint solving we use three SMT-solvers: Barcelogic 1.2 [BNO+08], MathSAT5 [CGSS13] and Z3 [dMB08]. Since non-determinism in all our benchmarks appears in variable assignments only, we implemented only our main algorithm and not the modified algorithm with stronger guarantees for programs with branching-only non-determinism.

Benchmarks. We evaluated our approach on benchmarks from the category *Termination of C-Integer Programs* of the Termination and Complexity Competition

Table 5.1: Experimental results with evaluation performed on the first platform. The NO/YES/MAYBE rows contain the total number of benchmarks which were proved non-terminating, terminating, or for which the tool proved neither, respectively. The next row contains the number of benchmarks proved to be non-terminating only by the respective tool. We also report the average and standard deviation (std. dev.) of runtimes. The last two rows show the runtime statistics limited to successful non-termination proofs.

	RevTerm	Ultimate	VeryMax
NO	107	97	103
YES	0	209	213
MAYBE	228	29	19
Unique NO	3	1	0
Avg. time	1.2s	5.0s	3.7s
Std. dev.	3.0s	3.7s	7.3s
Avg. time for NO	1.2s	4.4s	10.6s
Std. dev. for NO	3.0s	3.8s	9.4s

(TermComp'19 [GRS+19]). The benchmark suite consists of 335 programs with non-determinism: 111 non-terminating, 223 terminating, and the Collatz conjecture for which termination is unknown. We compared RevTerm against the best state-of-the-art tools that participated in this category, namely AProVE [GAB+17], Ultimate [CHL+18], VeryMax [BBL+17], and also LoAT [FG19].

Configurations of our tool. Recall that our algorithm is parameterized by the template size for predicate functions and the maximal polynomial degree. Also, it performs two checks which can be run sequentially or in parallel. Thus a *configuration* of RevTerm is defined by (a) the choice of whether we are running Check 1 or Check 2, (b) the template size (c, d) for predicate functions and the maximal polynomial degree D, and (c) the choice of an SMT-solver. Our aim is to compare our algorithm to other existing approaches to non-termination proving and demonstrate generality of its relative completeness guarantees, rather than develop an optimized tool. Hence we test each configuration separately and count the total number of benchmarks that were proved to be non-terminating by at least one of the configurations. We consider configurations for both checks, each of the three SMT-solvers, and all template sizes in the set $\{(c, d, D) \mid 1 \leq c \leq 5, 1 \leq d \leq 5, 1 \leq D \leq 2\}$.

Experimental results. Our experiments were run on two platforms, and we include the results for each of them in separate tables. The first platform is Debian, 128 GB RAM, Intel(R) Xeon(R) CPU E5-1650 v3 @ 3.50GHz, 12 Threads. The experimental

Table 5.2: Experimental results with evaluation performed on StarExec [SST14]. The meaning of data is the same as in Table 5.1.

	RevTerm	LoAT	AProVE	Ultimate	VeryMax
NO	103	96	99	97	102
YES	0	0	216	209	212
MAYBE	232	239	20	29	21
Unique NO	2	1	0	0	0
Avg. time	1.8s	2.6s	4.2s	7.4s	3.8s
Std. dev.	6.6s	0.9s	4.1s	4.9s	7.4s
Avg. time NO	1.8s	2.6s	5.0s	7.0s	10.8s
Std. dev. NO	6.6s	0.9s	3.9s	7.1s	9.5s

results are presented in Table 5.1 and the timeout for each experiment was 60s. We could not install the dependencies for AProVE on the first platform and LoAT does not support the input format of benchmarks, so we also evaluate all tools except for LoAT on StarExec [SST14] which is a platform on which TermComp'19 was run. We take the results of the evaluation of LoAT on StarExec from [FG19] which coupled it with AProVE for conversion of benchmarks to the right input format. Note however that the solver Barcelogic 1.2 is not compatible with StarExec so the number of non-terminations RevTerm proves is smaller compared to Table 5.1. The experimental results are presented in Table 5.2, and the timeout for each experiment was 60s. The timeout in both cases is on wallclock time and was chosen to match that in [FG19]. We note that in TermComp'19 the timeout was 300s and Ultimate proved 100 non-terminations, whereas AProVE and VeryMax proved the same number of non-terminations as in Table 5.2.

From Tables 5.1 and 5.2 we can see that RevTerm outperforms other tools in terms of the number of proved non-terminations. The average time for RevTerm is computed by taking the fastest successful configuration on each benchmark, so the times indicate that running multiple configurations in parallel would outperform the state-of-the-art. Since AProVE, Ultimate and VeryMax attempt to prove either termination or non-termination of programs, we include both their average times for all solved benchmarks and for non-termination proofs only.

Performance by configuration. We now discuss the performance of each configuration based on whether it runs Check 1 or Check 2 and based on which SMT-solver it uses. For the purpose of this comparison we only consider evaluation on the first platform which supports Barcelogic 1.2. Comparison of configurations in terms of the total number of solved benchmarks is presented in Table 5.3. We make two observations:

Table 5.3: Comparison of configurations based on which check they run and the SMT-solver used.

	Barcelogic 1.2	MathSAT5	Z3	Total
Check 1	84	98	80	103
Check 2	69	54	63	74
Total	96	98	82	107

Table 5.4: Comparison of configurations based on the template size for predicate functions. A cell in the table corresponding to $(C = i, D = j)$ contains the number of benchmarks that were proved to be non-terminating by a configuration using template size (c, d) with $c \leq i$ and $d \leq j$.

	$D = 1$	$D = 2$	$D = 3$	$D = 4$	$D = 5$
$C = 1$	58	76	77	78	78
$C = 2$	90	97	97	97	97
$C = 3$	102	107	107	107	107
$C = 4$	103	107	107	107	107
$C = 5$	103	107	107	107	107

- Configurations using Check 1 prove 103 out of 112 non-terminations, which matches the performance of all other tools. This means that the relative completeness guarantees provided by our approach are quite general.

- Even though some SMT-solvers perform well and solve many benchmarks, none of them reaches the number 107. This means that our performance is dependent on the solver choice and designing a successful tool would possibly require multiple solvers. For example, from our results we observed that MathSAT5 performs particularly well for Check 1 with templates of small size ($c, d \in \{1, 2\}$), while Barcelogic 1.2 is best suited for templates of larger size (with $c \geq 3$) and for Check 2. While this could be seen as a limitation of our approach, it also implies that our algorithm would become even more effective with the improvement of SMT-solvers.

Finally, in Table 5.4 we present a comparison of configurations based on the template sizes for predicate functions. A key observation here is that for any benchmark that RevTerm proved to be non-terminating, it was sufficient to use a template for predicate functions with $c \leq 3$, $d \leq 2$ and $D \leq 2$. This implies that with a smart choice of configurations, it suffices to run a relatively small number of configurations which if run in parallel would result in a tool highly competitive with the state-of-the-art.

5.7 Related Work

In what follows, we present a more detailed comparison to existing methods for non-termination proving in non-probabilistic programs.

A large number of techniques for proving non-termination consider *lasso-shaped* programs, which consist of a finite prefix (or stem) followed by a single loop without branching [GHM+08, LH18]. Such techniques are suitable for being combined with termination provers [HLNR10]. Many modern termination provers repeatedly generate traces which are then used to refine the termination argument in the form of a ranking function, either by employing safety provers [CPR06] or by checking emptiness of automata [HHP14]. When refinement is not possible, a trace is treated like a lasso program and the prover would try to prove non-termination. However, lassos are not sufficient to detect *aperiodic* non-termination, whereas our approach handles it. Moreover, programs with nested loops typically contain infinitely many lassos which may lead to divergence, and such methods do not provide relative completeness guarantees.

TNT [GHM+08] proves non-termination by exhaustively searching for candidate lassos. For each lasso, it searches for a *recurrence set* (see Section 5.4) and this search is done via constraint solving. The method does not support non-determinism.

Closed recurrence sets (see Section 5.4) are a stronger notion than the recurrence sets, suited for proving non-termination of non-deterministic programs. The method for computing closed recurrent sets in [CCF+14] was implemented in T2 and it uses a safety prover to eliminate terminating paths iteratively until it finds a program under-approximation and a closed recurrence set in it. The method can detect aperiodic non-termination. However it is likely to diverge in the presence of many loops, as noted in [LNO+14].

The method in [LNO+14] was implemented in VeryMax [BBL+17] and it searches for witnesses to non-termination in the form of quasi-invariants, which are sets of states that cannot be left once they are entered. Their method searches for a quasi-invariant in each strongly-connected subgraph of the program by using Max-SMT solving. Whenever a quasi-invariant is found, safety prover is used to check its reachability. The method relies on multiple calls to a safety prover and does not provide relative completeness guarantees.

AProVE [GAB+17] proves non-termination of Java programs [BSOG11] with non-determinism. It uses constraint solving to find a recurrence set in a given loop, upon which it checks reachability of the loop. The key limitation of this approach is that for programs with nested loops for which the loop condition is not a loop invariant, it can only detect recurrence sets with a single variable valuation at the loop head.

An orthogonal approach to recurrence sets was presented in [LH18]. It considers lasso-

shaped programs with linear arithmetic and represents infinite runs as geometric series. Their method provides relative completeness guarantees for the case of deterministic lasso-shaped programs. It also supports non-determinism, but does not provide relative completeness guarantees. The method has been implemented as a non-termination prover for lasso traces in Ultimate [CHL+18].

The method in [UGK16] tries to prove either termination or non-termination of programs with non-determinism by making multiple calls to a safety prover. For each loop, a termination argument is incrementally refined by using a safety prover to sample a terminating trace that violates the argument. Once such terminating traces cannot be found, a safety prover is again used to check the existence of non-terminating traces in the loop.

The work of [GSV08] considers deterministic programs with linear integer arithmetic. They present a constraint solving-based method for finding the *weakest liberal precondition (w.l.p.)* of a fixed predicate function template. They then propose a method for proving non-termination which computes the w.l.p. for the postcondition "false", and then checks if it contains some initial state. While this approach is somewhat similar to Check 1, encoding and solving the *weakest* precondition constraints of a given template is computationally expensive and unnecessary for the purpose of proving non-termination. In Check 1, we do not impose such a strict condition. Moreover, initial diverging states are not sufficient to prove non-termination of non-deterministic programs. It is not immediately clear how one could use w.l.p. calculus to find a diverging state within a loop, like in Example 5.5.3.

The tool Invel [VR08] proves non-termination of Java programs using constraint solving and heuristics to search for recurrence sets. It only supports deterministic programs. In [LQC15] a Hoare-style approach is developed to infer sufficient preconditions for terminating and non-terminating behavior of programs. As the paper itself mentions, the approach is not suitable for programs with non-determinism.

While all of the methods discussed above are restricted to programs with linear arithmetic, the following two methods also consider non-linear programs.

The tool Anant [CFNO14] proves non-termination of programs with non-linear arithmetic and heap-based operations. They define *live abstractions*, which over-approximate a program's transition relation while keeping it sound for proving non-termination. Their method then over-approximates non-linear assignments and heap-based commands with non-deterministic linear assignments using heuristics to obtain a live abstraction with only linear arithmetic. An approach similar to [GHM+08] but supporting non-determinism is then used, to exhaustively search for lasso traces and check if they are non-terminating. The over-approximation heuristic they present is compatible with our approach and could be used to extend our method to support operations on the heap.

LoAT [FG19] proves non-termination of integer programs by using loop acceleration. If a loop cannot be proved to be non-terminating, the method tries to accelerate it in order to find paths to other potentially non-terminating loops.

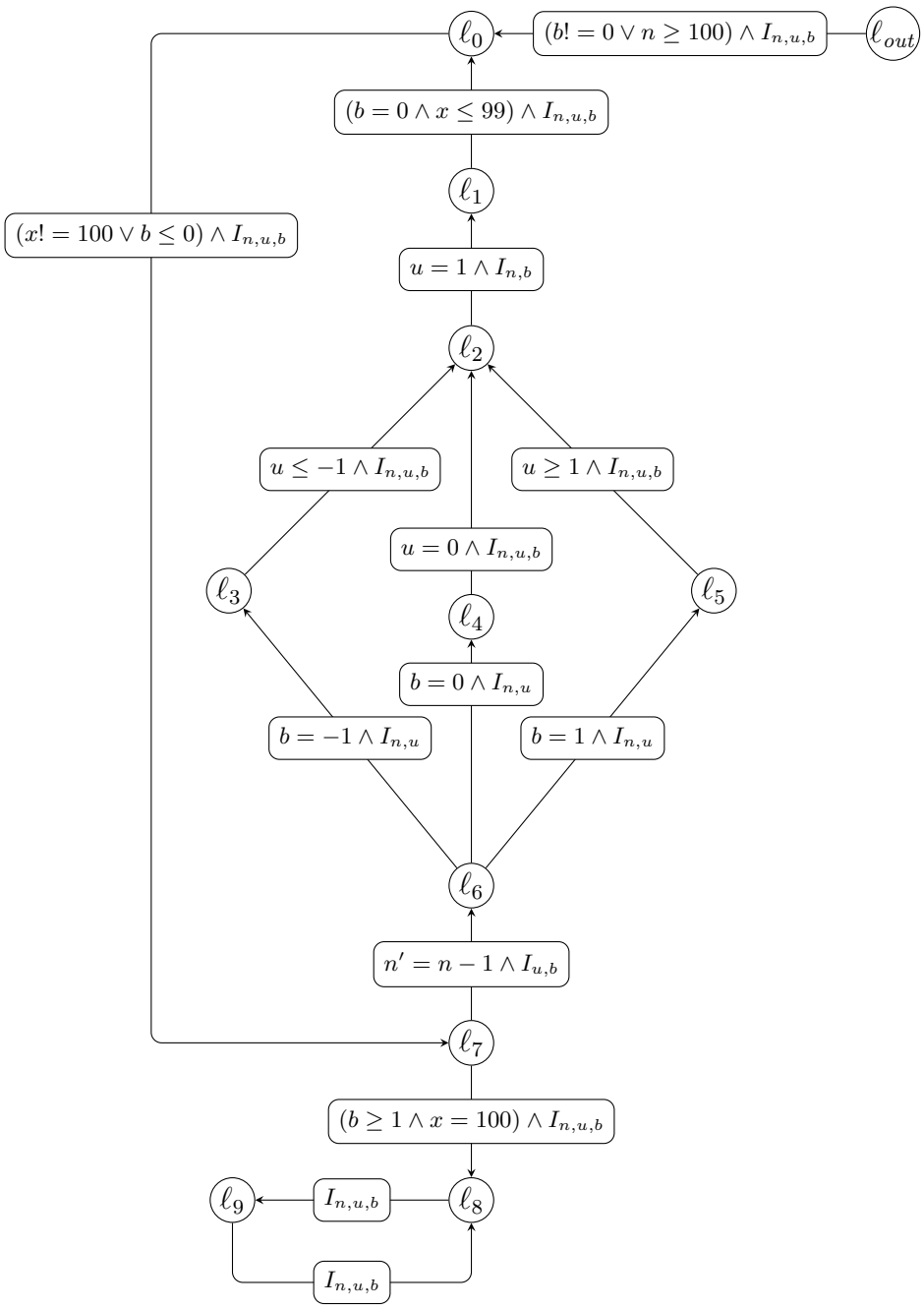

Figure 5.5: Reversed transition system of the program in Fig. 5.3 with the resolution of non-determinism that assigns the constant expression 1 to the non-deterministic assignment of the variable u. For readability, we use $I_{n,u,b}$ to denote $n' = n \wedge u' = u \wedge b' = b$, $I_{u,b}$ to denote $u' = u \wedge b' = b$, etc.

CHAPTER 6

Learning-based Stochastic Control with Almost-sure Reachability

This section is based on the following publications:

- Mathias Lechner*, Đorđe Žikelić*, Krishnendu Chatterjee, Thomas A. Henzinger. *Stability Verification in Stochastic Control Systems via Neural Network Supermartingales*. In Thirty-Sixth AAAI Conference on Artificial Intelligence, **AAAI 2022**
- Đorđe Žikelić*, Mathias Lechner*, Krishnendu Chatterjee, Thomas A. Henzinger. *Learning Stabilizing Policies in Stochastic Control Systems*. In ICLR 2022 Workshop on Socially Responsible Machine Learning, **SRML 2022**

6.1 Introduction

Stochastic control with qualitative reachability guarantees. We now proceed to studying formal controller synthesis in discrete-time stochastic dynamical systems. Recall, given a stochastic dynamical system and a specification, the goal of formal controller synthesis is to compute a controller (or a control policy) under which the system is guaranteed to satisfy the specification. In this chapter, we study formal controller synthesis in stochastic dynamical systems with *probability* 1 *(a.k.a. almost-sure) reachability* guarantees. To the best of our knowledge, we present the first framework for learning neural control policies in stochastic dynamical systems with

*Equal contribution.

formal almost-sure reachability guarantees. Our framework also yields a method for formally verifying almost-sure reachability under neural network control policies in stochastic dynamical systems. In order to motivate the necessity of learning-based control methods that provide formal guarantees and thus explain the significance of our contributions in this chapter, we start by overviewing existing approaches to formal controller synthesis in dynamical systems and explaining their limitations. A detailed survey of related work is presented in Section 6.6.

Prior approaches to formal controller synthesis. Discrete-time dynamical systems, be it deterministic or stochastic, are defined by a dynamics function over a continuous state space which defines how the system evolves over time under a control policy. Formal controller synthesis in dynamical systems is typically achieved by synthesizing a control policy together with a certificate function which formally proves that the desired specification is satisfied. For instance, Lyapunov functions [HC11] are standard certificate functions for reachability and stability and control barrier functions [ACE+19] are standard certificate functions for safety in deterministic dynamical systems. Classical methods for formal controller synthesis either use hand-crafted certificate functions and are thus only semi-automated, or consider *polynomial systems* and utilize semidefinite programming (SDP) to synthesize polynomial policies and certificate functions [HG05, Par00, GK01, JWFT+03]. However, dynamical systems appearing in practice are often not polynomial. To alleviate this problem, classical methods consider polynomial approximations of system dynamics, but this introduces approximation error that accumulates over time and allows only finite and typically short time horizon guarantees. Another approach is to first construct a *finite-state abstraction* of the dynamical system, solve the control task for this finite-state abstraction and project the resulting controller back to the original continuous system. Notable examples of such approaches for stochastic dynamical systems include [SGA15, LKSZ20, CA19, VGO19]. However, due to the approximation error being accumulated over time, these methods provide guarantees only over a finite and a priori fixed time horizon. Recently, a few abstraction-based controller synthesis methods for infinite-time horizon stochastic dynamical systems with affine dynamics [HS21], control affine dynamics [DHC22] or in which control input space is finite [MMSS24, DHC22] have been proposed. However, to the best of our knowledge, no existing method can handle formal controller synthesis in infin ite-time horizon non-polynomial stochastic dynamical systems with continuous control inputs.

Learning-based control. Learning-based methods and in particular reinforcement learning (RL) provide a promising approach to solving highly nonlinear control problems without imposing restrictions on the time horizon, thus overcoming the limitations discussed above. However, most RL algorithms only focus on learning a policy that maximizes the expectation of some reward function [SB18] and do not take other

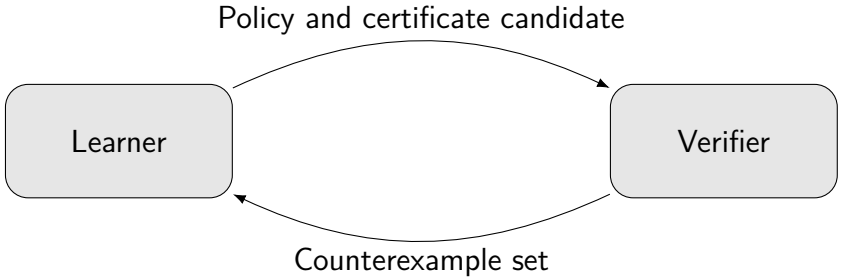

Figure 6.1: The learner-verifier loop, figure taken from [CHLZ23, Figure 1].

constraints into account. The safe RL paradigm considers constrained Markov decision processes (cMDPs) [Alt99, Gei06, AHTA17, CNDG18] which are equipped with both a reward and a cost function. Solving cMDPs consists of optimizing the expected reward while satisfying some expected cost constraint. However, the existing safe RL methods only aim to satisfy the cost constraint in expectation and furthermore they do not provide formal guarantees. This raises concerns about the suitability of RL methods for safety-critical applications such as autonomous vehicles or healthcare. A fundamental challenge for the deployment of policies learned via RL algorithms in safety-critical applications is certifying their safety and correctness [GF15, AOS+16]. A recent trend in formal controller synthesis for deterministic dynamical systems is to learn and verify a neural network control policy, *together with a neural network certificate function* that formally proves the specification of interest. In particular, methods that jointly learn and verify a neural control policy and a neural certificate function have been proposed in [RBK18, CRG19, AAGP21] for reachability and stability and in [PAA21] for safety in discrete- and continuous-time deterministic dynamical systems. However, while these works make significant contributions towards setting the foundations of learning-based formal controller synthesis in deterministic dynamical systems, their methods do not extended to the setting of stochastic dynamical systems (reasons impeding their extension to stochastic dynamical systems are discussed below). Prior to the work in this thesis, no learning-based formal controller synthesis framework for stochastic dynamical systems has been proposed.

Our approach – learning and verification of neural supermartingales. In this chapter, we present the first formal controller synthesis method that learns neural controllers for stochastic dynamical systems with almost-sure reachability guarantees. Our method jointly learns a control policy and a ranking supermartingale that formally certifies almost-sure reachability of some target set of states, both parametrized as neural network functions mapping system states to real numbers. Recall, *ranking supermartingales (RSMs)* [CS13] were introduced for almost-sure termination and reachability analysis in PPs. An RSM for some target region is a nonnegative function

that decreases in expectation by at least $\varepsilon > 0$ after every one-step evolution of the system and in each state that is not in the target region. We prove that RSMs can also be used to define almost-sure reachability certificates for stochastic control problems.

At the core of our approach is the *learner-verifier framework*, which proceeds in the counterexample guided invariant synthesis (CEGIS) fashion [STB+06] and draws insights from the learner-verifier framework for deterministic dynamical systems first proposed in [CRG19]. The framework consists of two modules: the *learner* which learns a control policy and an RSM candidate in the form of neural networks, and the *verifier* which then formally verifies correctness of the learned RSM candidate. If the verification step succeeds, the method outputs the control policy and the RSM which formally proves almost-sure reachability. Otherwise, if the verification step fails, a set of counterexamples showing that the candidate is not a valid RSM is computed, which are then used by the learner to fine-tune the candidate. This learner-verifier loop is repeated until the learned control policy and the RSM candidate are successfully verified or until a timeout is reached. See Fig. 6.1 for an illustration of the learner-verifier loop. Our framework is applicable to stochastic dynamical systems defined over compact (i.e. closed and bounded) state spaces with general continuous dynamics and with continuous control input spaces (note that every continuous function defined over a compact domain is also Lipschitz continuous). Thus, it handles infinite-time horizon non-polynomial stochastic dynamical systems with continuous control inputs, as desired.

Key challenge - verification of the expected decrease condition. One of the key algorithmic challenges in designing the verifier module, compared to the case of deterministic dynamical systems, is that we need to verify the expected decrease condition of RSMs which requires being able to compute the *expected value* of a neural network function over a probability distribution. Note, sampling cannot be used for this task since it only allows computing statistical bounds. On the other hand, the verification step of existing learner-verifier methods for deterministic dynamical systems [CRG19, AAGP21, PAA21] proceeds by reduction to a decision procedure and invoking an off-the-shelf solver. In our case, that would require being able to compute a closed-form expression of the expected value of a neural network function over a probability distribution. However, it is not clear how this closed-form expression can be computed or if it exists at all in the general case. To solve this challenge, we propose a method for efficiently computing formal and tight bounds on the expected value of an arbitrary neural network function over a probability distribution. The verifier then formally verifies the expected decrease condition by *discretizing* the state space and formally verifying a slightly stricter condition at the discretization vertices. By carefully choosing the mesh of the discretization and computing an additional error term, we obtain a sound verification procedure applicable to general Lipschitz continuous systems.

The choice of RSMs and reachability time. RSMs can intuitively be viewed as a stochastic extension of Lyapunov functions for reachability and stability analysis in deterministic dynamical systems [Kha02]. However, prior theoretical work on reachability and stability analysis in stochastic dynamical systems has proposed several different extensions of Lyapunov functions (see [Kus14] for a survey). There are two key advantages of using RSMs instead of other stochastic extensions of Lyapunov functions. First, we show that the defining properties of RSMs are much easier to encode within a learning framework. Second, we show that RSMs also provide bounds on the *reachability time*, which stochastic Lyapunov functions do not. Ensuring that reachability happens within some tolerable time limit can also be important. For instance, consider the scenario of a self-driving car where we wish to train a policy which decreases its speed whenever the speed exceeds the allowed speed limit. If the self-driving car drives at a very high speed, it is not sufficient to *only* ensure that the speed eventually reaches the allowed speed limit. A good speed stabilizing policy *additionally* needs to provide plausible guarantees on the speed stabilization time. One of the key benefits of using RSMs is that we show that they provide such guarantees.

Verification procedure for Lipschitz continuous policies. We show that our framework also yields a method for formally verifying almost-sure reachability under a given control policy, by simply fixing the control policy in the learner-verifier loop and only learning and verifying the RSM. Our method only assumes that the given control policy is Lipschitz continuous. This allows for neural network control policies with all standard activation functions since they are known to be Lipschitz continuous [SZS+14].

Contributions. Our contributions in this chapter can be summarized as follows:

1. We show that ranking supermartingales (RSMs) provide a formal certificate for almost-sure reachability in stochastic dynamical systems, as well as guarantees on the reachability time.

2. We present a learner-verifier framework for jointly learning a control policy and an RSM which formally proves almost-sure reachability, both parametrized as neural networks. Our method is applicable to stochastic dynamical systems with compact state space and (Lipschitz) continuous dynamics function.

3. By fixing a control policy and only learning and verifying the RSM, our framework also yields a method for formal verification of almost-sure reachability under a given Liphscitz continuous control policy.

4. As a part of our verification procedure, we present a method for efficiently computing formal bounds on the expected value of a neural network function

over a probability distribution. We are not aware of any existing works that tackle this problem and believe that this contribution may be of independent interest for neural network analysis.

5. We empirically validate our approach and show that it successfully learns and verifies neural network control policies for ensuring almost-sure reachability in stochastic dynamical systems

This chapter is based on [LZCH22], which introduced our learner-verifier framework for formal controller synthesis and verification in stochastic dynamical systems. The presentation in [LZCH22] focused more on formal controller verification whereas here we adapt the presentation to focus on formal controller synthesis. Additional experiments on formal controller synthesis and the practical study of neural network initialization were presented in [ZLCH22], which we also include in this chapter.

Almost-sure reachability and stability. We note that our method is also applicable to formal controller synthesis and verification with *almost-sure asymptotic stability* guarantees, assuming that the stabilizing set is *closed under system dynamics*. Stability is one of the most important specifications appearing in control tasks [Lya92]. Given a stochastic dynamical system and a stabilizing set $\mathcal{X}_s$, we say that $\mathcal{X}_s$ is almost-sure asymptotically stable under a control policy if the system reaches and eventually stays within $\mathcal{X}_s$ with probability 1. The closedness under system dynamics assumption imposes that the stabilizing set $\mathcal{X}_s$ cannot be left once entered. Hence, under this assumption, the almost-sure asymptotic stability specification reduces to the almost-sure reachability specification. Assuming the closedness under system dynamics of the stabilization set is a reasonable and a realistic choice, due to dynamical systems typically expressing weak dynamics around the systems' stable points. Formal controller synthesis with stability guarantees under the closedness under system dynamics assumption has been the focus of the existing learning-based formal control methods for deterministic dynamical systems [BTSK17, RBK18, CRG19], and the work [LZCH22] on which this chapter is based has also focused on this problem for stochastic dynamical systems, with the reduction to reachability discussed in [LZCH22, Preliminaries]. However, since this thesis focuses on reachability and safety specifications, we adapt the presentation of [LZCH22] to consider almost-sure reachability.

Chapter organization. The rest of this chapter is organized as follows. We formally define the problem considered in this chapter in Section 6.2. In Section 6.3, we present our theoretical results and show that RSMs can be used as a formal certificate function for almost-sure reachability in stochastic dynamical systems. In Section 6.4, we present our learner-verifier framework for stochastic dynamical systems. In Section 6.5 we present our experimental results. We survey related work in Section 6.6. Finally,

Section 6.7 contains full proofs of results presented in earlier sections that are deferred to this section in order to enhance readability.

6.2 Problem Statement

We consider a discrete-time stochastic dynamical system

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t, \omega_t),\ t \in \mathbb{N}_0,$$

defined by a dynamics function $f : \mathcal{X} \times \mathcal{U} \times \mathcal{N} \to \mathcal{X}$ as in Section 2.3. We use d to denote the probability distribution over the stochastic disturbance space $\mathcal{N}$ from which the stochastic disturbance vector ω_t is sampled at each time step.

Problem statement. We consider the *qualitative reachability analysis* problem in stochastic dynamical systems. Let $\mathcal{X}_t \subseteq \mathcal{X}$ be a Borel-measurable *target set* and

$$\mathsf{Reach}(\mathcal{X}_t) = \left\{ (\mathbf{x}_t, \mathbf{u}_t, \omega_t)_{t \in \mathbb{N}_0} \mid \exists t \in \mathbb{N}_0 .\, \mathbf{x}_t \in \mathcal{X}_t \right\}$$

be the set of all trajectories of the system that reach the target set $\mathcal{X}_t$. Our goal is to synthesize a control policy π such that, for every initial state $\mathbf{x}_0 \in \mathcal{X}$, we have

$$\mathbb{P}^{\pi}_{\mathbf{x}_0}\Big[\mathsf{Reach}(\mathcal{X}_t)\Big] = 1.$$

Assumptions. Reachability analysis in stochastic dynamical systems would be impossible without additional assumptions on the system, so that the model is sufficiently well-behaved. To that end, as in Section 2.3, we assume that $\mathcal{X}$, $\mathcal{U}$ and $\mathcal{N}$ are all Borel-measurable for the system semantics to be well-defined. We also assume that $\mathcal{X} \subseteq \mathbb{R}^m$ is compact (i.e. closed and bounded) in the Euclidean topology of $\mathbb{R}^m$. The dynamics function f is assumed to be continuous, which is a common assumption in control theory, including the existing works on learning-based formal controller synthesis with stability guarantees in deterministic dynamical systems [RBK18, CRG19]. Note that every continuous function defined over a compact state space is also Lipschitz continuous. Finally, we assume that d has bounded support or that it is a product of independent univariate probability distributions, which is needed for efficient sampling and expected value computation.

6.3 Theoretical Results

We now present a theoretical framework for formally verifying almost-sure reachability in discrete-time stochastic dynamical system. Our framework is based on ranking

supermartingales, which we recall and formally define in the setting of stochastic dynamical systems below. In what follows, we assume that a control policy π is fixed. In Section 6.4 we will then present our algorithm for learning and verifying a control policy together with a ranking supermartingale.

Ranking supermartingales Consider a discrete-time stochastic dynamical system defined by a dynamics function f, a policy π and a probability distribution d with model assumptions as in the previous section, and let $\mathcal{X}_t \subseteq \mathcal{X}$. Intuitively, a ranking supermartingale (RSM) for $\mathcal{X}_t$ is a nonnegative continuous function whose value at each state in $\mathcal{X} \backslash \mathcal{X}_t$ decreases in expectation by at least $\varepsilon > 0$ (is ε-*ranked*) after a one-step evolution of the system under the policy π, where the expected value is taken with respect to the probability distribution d over disturbance vectors. RSMs were first introduced in [CS13] for the termination analysis of PPs, and we adapt them to the setting of stochastic dynamical systems below.

Defintion 6.3.1. *A continuous function $V : \mathcal{X} \to \mathbb{R}$ is said to be a* ranking supermartingale (RSM) for $\mathcal{X}_t$, *if $V(\mathbf{x}) \geq 0$ holds for any $\mathbf{x} \in \mathcal{X}$ and if there exists $\varepsilon > 0$ such that*

$$\mathbb{E}_{\omega \sim d}\Big[V\Big(f(\mathbf{x}, \pi(\mathbf{x}), \omega)\Big)\Big] \leq V(\mathbf{x}) - \varepsilon \tag{6.1}$$

holds for every $\mathbf{x} \in \mathcal{X} \backslash \mathcal{X}_t$.

We note that RSMs differ from the commonly considered stochastic Lyapunov functions for discrete-time stochastic systems [Kus14], which require V to be continuous and to satisfy the following conditions:

- $\mathbb{E}_{\omega \sim d}[V(f(\mathbf{x}, \pi(\mathbf{x}), \omega))] < V(\mathbf{x})$ for $\mathbf{x} \in \mathcal{X} \backslash \mathcal{X}_t$,
- $V(\mathbf{x}) > 0$ for $\mathbf{x} \in \mathcal{X} \backslash \mathcal{X}_t$, and
- $V(\mathbf{x}) = 0$ for $\mathbf{x} \in \mathcal{X}_t$.

The third condition would be quite restrictive if we tried to learn V in the form of a neural network, which is our goal. Thus, one of the key benefits of considering RSMs instead of stochastic Lyapunov functions is that we may replace the $V(\mathbf{x}) = 0$ for $\mathbf{x} \in \mathcal{X}_t$ condition by a slightly stricter expected decrease condition that requires the decrease by at least some $\varepsilon > 0$. Theorem 6.3.2 establishes that RSMs are indeed sufficient to prove a.s. reachability.

Theorem 6.3.2. *Let $f : \mathcal{X} \times \mathcal{U} \times \mathcal{N} \to \mathcal{X}$ be a continuous dynamics function, $\pi : \mathcal{X} \to \mathcal{U}$ a continuous control policy and d a probability distribution over $\mathcal{N}$. Let $\mathcal{X}_t \subseteq \mathcal{X}$. Suppose that there exists an RSM $V : \mathcal{X} \to \mathbb{R}$ for $\mathcal{X}_t$. Then, for every initial state $\mathbf{x}_0 \in \mathcal{X}$, we have $\mathbb{P}^{\pi}_{\mathbf{x}_0}[\mathsf{Reach}(\mathcal{X}_t)] = 1$.*

The main idea behind the proof of Theorem 6.3.2 is as follows. For each state $\mathbf{x}_0 \in \mathcal{X}$, we consider the probability space of all trajectories that start in $\mathbf{x}_0$. We then show that the RSM V for $\mathcal{X}_t$ gives rise to an instance of the mathematical notion of RSMs in this probability space, and use results from probability theory on the convergence of RSMs to conclude that $\mathcal{X}_t$ is reached almost-surely. The proof is presented in Section 6.7.2.

Bounds on the reachability time While formally verifying that the system reaches the target set with probability 1 is very important for safety critical applications, a practical concern is to ensure that reachability happens within some tolerable time limit. Another important caveat of using RSMs for almost-sure reachability analysis in stochastic dynamical systems is that they provide formal guarantees on the reachability time. For a system trajectory $(\mathbf{x}_t, \mathbf{u}_t, \omega_t)_{t \in \mathbb{N}_0}$, we define its reachability time $T_{\mathcal{X}_t} = \inf\{t \in \mathbb{N}_0 \mid \mathbf{x}_t \in \mathcal{X}_t\}$ to be the first hitting time of the target set $\mathcal{X}_t$ (with $T_{\mathcal{X}_s} = \infty$ if trajectory never reaches $\mathcal{X}_s$). Given $c > 0$, the system has *c-bounded differences* if the distance between any two consecutive system states with respect to the l_1-norm does not exceed c, i.e. for any $\mathbf{x} \in \mathcal{X}$ and $\omega \in supp(d)$ we have $||f(\mathbf{x}, \pi(\mathbf{x}), \omega) - \mathbf{x}||_1 \leq c$.

Theorem 6.3.3. *Let $f : \mathcal{X} \times \mathcal{U} \times \mathcal{N} \to \mathcal{X}$ be a continuous dynamics function, $\pi : \mathcal{X} \to \mathcal{U}$ a continuous control policy and d a probability distribution over $\mathcal{N}$. Let $\mathcal{X}_t \subseteq \mathcal{X}$. Suppose that there exists an ε-RSM $V : \mathcal{X} \to \mathbb{R}$ for $\mathcal{X}_t$. Then, for any initial state $\mathbf{x}_0 \in \mathcal{X}$,*

1. $\mathbb{E}_{\mathbf{x}_0}[T_{\mathcal{X}_t}] \leq \frac{V(\mathbf{x}_0)}{\varepsilon}$.
2. $\mathbb{P}_{\mathbf{x}_0}[T_{\mathcal{X}_t} \geq t] \leq \frac{V(\mathbf{x}_0)}{\varepsilon \cdot t}$, *for any time* $t \in \mathbb{N}$.
3. *If the system has* c-bounded differences *for* $c > 0$, *then* $\mathbb{P}_{\mathbf{x}_0}[T_{\mathcal{X}_t} \geq t] \leq A \cdot e^{-t \cdot \varepsilon^2/(2 \cdot (c+\varepsilon)^2)}$ *for any time* $t \in \mathbb{N}$ *and* $A = e^{\varepsilon \cdot V(\mathbf{x}_0)/(c+\varepsilon)^2}$.

The proof of Theorem 6.3.3 can be found in Section 6.7.3 and here we present the key ideas. The first part shows that the expected reachability time is bounded from above by the initial value of V divided by ε. To prove it, we show that the reachability time gives rise to a stopping time in the probability space of all trajectories that start in $\mathbf{x}_0$. We then observe that the RSM V satisfies the expected decrease condition until $T_{\mathcal{X}_s}$ is exceeded and use the results from probability theory on the convergence of RSMs to conclude the bound on the expected value of this stopping time. The second part shows a bound on the probability that the reachability time exceeds a threshold t, and it follows immediately from the first part by an application of Markov's inequality. Note that this bound decays linearly in t, as $t \to \infty$. Finally, the third part shows an asymptotically tighter bound with the decay in t being exponential, for systems that have c-bounded differences. The proof follows by an application of Azuma's inequality [Azu67].

6.4 Learner-verifier Framework

We now present our learner-verifier framework for formal controller synthesis with almost-sure reachability guarantees by jointly learning and verifying a control policy and an RSM. The framework consists of two modules which alternate within a loop: the *learner* and the *verifier*. In each loop iteration, the learner first learns a control policy π_ν and an RSM candidate V_θ in the form of neural networks, where ν and θ are vectors of neural network parameters. The control policy and the RSM candidate are then passed to the verifier, which checks whether V_θ is a valid RSM. If the answer is positive, the verifier terminates the loop and concludes the target set of states is reached almost-surely under the learned policy π_ν. Otherwise, the verifier computes a set of counterexamples which show that the candidate is not an RSM and passes it to the learner, which then proceeds with the next learning iteration. This process is repeated until either a learned candidate is verified or a given timeout is reached.

In what follows, we consider a discrete-time stochastic dynamical system defined by a dynamics function f and a probability distribution d with model assumptions as in the previous sections, and a target set $\mathcal{X}_t \subseteq \mathcal{X}$. The rest of this section describes the details behind our framework. The underlying algorithm is shown in Algorithm 6.1.

6.4.1 Initialization and Discretization

Policy initialization. Our algorithm initializes the neural network policy π_ν by using proximal policy optimization (PPO) [SWD+17]. In order to be able to run PPO, the algorithm induces an MDP from the system and defines a reward function $r : \mathcal{X} \to [0, 1]$ via $r(\mathbf{x}) := \mathbb{I}[\mathcal{X}_t](\mathbf{x})$. Our experimental results will show that a naive initialization often does not allow learning a policy that ensures almost-sure reachability, thus proper initialization is important. The importance of initialization was also observed in [CRG19]. Specifically, [CRG19] initialized their linear policy with the LQR solution of the system linearized at the origin. However, it is not clear how one could linearize stochastic dynamics with non-additive stochastic disturbance or systems that are highly non-linear, whereas our initialization is applicable to general stochastic dynamical systems.

State space discretization. Recall, an RSM V_θ needs to satisfy the expected decrease condition in eq. (6.1) at each state in $\mathcal{X}\backslash\mathcal{X}_t$. However, one of the main difficulties in verifying this condition when V_θ has a neural network form is that it is not clear how to compute a closed form for the expected value of V_θ at a successor system state. In order to be able to verify neural network RSM candidates, our method discretizes the state space and then verifies the expected decrease condition only at the states in the discretization (which we will show to be possible due to f, π_ν and V_θ all being Lipschitz continuous and $\mathcal{X}$ being compact). The *discretization* $\tilde{\mathcal{X}}$ of $\mathcal{X}\backslash\mathcal{X}_t$

Algorithm 6.1: Algorithm for learning almost-sure reachability policies

input : Dynamics function f, disturbance distribution d, target set $\mathcal{X}_t \subseteq \mathcal{X}$, Lipschitz constant L_f, parameters $\tau > 0$, $N \in \mathbb{N}$, $\lambda > 0$

output : "Almost-sure reachability" or "Unknown"

$\tilde{\mathcal{X}} \leftarrow$ discretization of $\mathcal{X} \backslash \mathcal{X}_t$ with mesh τ
for $\mathbf{x}$ *in* $\tilde{\mathcal{X}}$ **do**
  $\mathcal{D}_{\mathbf{x}} \leftarrow N$ sampled successor states of $\mathbf{x}$
end
while *timeout not reached* **do**
  $\pi_\nu, V_\theta \leftarrow$ trained policy and RSM candidate by minimizing the loss in eq. (6.4)
  $L_\pi, L_V \leftarrow$ Lipschitz constants of π_ν and V_θ
  $K \leftarrow L_V \cdot (L_f \cdot (L_\pi + 1) + 1)$
  if $\exists \mathbf{x} \in \tilde{\mathcal{X}}$ *s.t.* $\mathbb{E}_{\omega \sim d}[V_\theta(f(\mathbf{x}, \pi_\nu(\mathbf{x}), \omega))] \geq V_\theta(\mathbf{x}) - \tau \cdot K$ **then**
    $\mathcal{D}_{\mathbf{x}} \leftarrow$ add N sampled successor states of $\mathbf{x}$
  end
  else
    return *"Almost-sure reachability"*
  end
end
return *Unknown*

satisfies the property that, for each $\mathbf{x} \in \mathcal{X} \backslash \mathcal{X}_t$, there is $\tilde{\mathbf{x}} \in \tilde{\mathcal{X}}$ with $||\mathbf{x} - \tilde{\mathbf{x}}||_1 < \tau$, with τ an algorithm parameter that we call the *mesh* of $\tilde{X}$. Since $\mathcal{X}$ is compact and so $\mathcal{X} \backslash \mathcal{X}_t$ is bounded, the discretization consists of finitely many states.

The method also initializes the collection of pairs $\mathcal{D} = \{(\mathbf{x}, \mathcal{D}_{\mathbf{x}}) \mid \mathbf{x} \in \tilde{\mathcal{X}}\}$, where each $\mathcal{D}_{\mathbf{x}}$ consists of N successor states of $\mathbf{x}$ obtained by independent sampling of N successor states for each $\mathbf{x} \in \tilde{\mathcal{X}}$. Here, $N \in \mathbb{N}$ is an algorithm parameter. The collection $\mathcal{D}$ will be used to approximate expected values at successor states for each $\mathbf{x}$ in $\tilde{\mathcal{X}}$ in the loss function used by the learner.

6.4.2 Verifier

In order to motivate the form of the loss function used by the learner, we first describe the verifier module of our algorithm. For a neural network V_θ to be an RSM as in Definition 6.3.1, it needs to be (1) continuous, (2) nonnegative at each state, and (3) to satisfy the expected decrease condition in eq. (6.1) for each state in $\mathcal{X} \backslash \mathcal{X}_t$. Since V_θ is a neural network we already know that it is a continuous function. Moreover, since $\mathcal{X}$ is compact and V is continuous, the function V admits a finite global lower

bound $-m \in \mathbb{R}$. Hence, if we verify that V satisfies the expected decrease condition, we may consider the function $V'(\mathbf{x}) = V(\mathbf{x}) + m$ which is in addition nonnegative and thus an RSM to conclude almost-sure reachability of $\mathcal{X}_t$. Therefore, the verifier only needs to check that V_θ satisfies the expected decrease condition in eq. (6.1) for each state in $\mathcal{X} \backslash \mathcal{X}_t$, from which it immediately follows that V'_θ is an RSM.

As explained above, checking this for each state in $\mathcal{X} \backslash \mathcal{X}_t$ is not feasible since we cannot compute a closed form for the expected value of V_θ at a successor system state. Instead, we show that it is sufficient to check a slightly stricter condition on states in the discretization $\tilde{\mathcal{X}}$. Let L_f, L_π and L_V be the Lipschitz constants of f, the learned control policy π_ν and the candidate RSM V_θ, respectively. We assume that the Lipschitz constant for the dynamics function f is provided, and use the method of [SZS+14] to compute the Lipschitz constant of neural networks π_ν and V_θ. Then define

$$K = L_V \cdot (L_f \cdot (L_\pi + 1) + 1). \tag{6.2}$$

In order to verify that V_θ satisfies the expected decrease condition in eq. (6.1) for each state in $\mathcal{X} \backslash \mathcal{X}_t$, the verifier checks for each $\mathbf{x}$ in the discretization $\tilde{\mathcal{X}}$ that

$$\mathbb{E}_{\omega \sim d}\Big[V_\theta\Big(f(\mathbf{x}, \pi_\nu(\mathbf{x}), \omega)\Big)\Big] < V_\theta(\mathbf{x}) - \tau \cdot K. \tag{6.3}$$

If eq. (6.3) holds for each $\mathbf{x} \in \tilde{\mathcal{X}}$, the verifier concludes almost-sure reachability of $\mathcal{X}_t$. Otherwise, if $\mathbf{x} \in \tilde{\mathcal{X}}$ for which eq. (6.3) does not hold is found, it is passed to the learner by independently sampling N successor states of $\mathbf{x}$ which are added to $\mathcal{D}_\mathbf{x}$.

Theorem 6.4.1 establishes the correctness of Algorithm 6.1 by showing that it indeed suffices to check eq. (6.3) for states in the discretization. The proof uses the fact that f is Lipschitz continuous and that $\mathcal{X}$ is compact, and is provided in Section 6.7.4.

Theorem 6.4.1. *Suppose that the verifier in Algorithm 6.1 verifies that V_θ satisfies eq. (6.3) for each $\mathbf{x} \in \tilde{\mathcal{X}}$. Let $-m \in \mathbb{R}$ be such that $V_\theta(\mathbf{x}) \geq -m$ for each $\mathbf{x} \in \mathcal{X}$. Then, the function $V'_\theta(\mathbf{x}) = V_\theta(\mathbf{x}) + m$ is an RSM for $\mathcal{X}_t$. Hence, $\mathcal{X}_t$ is reached almost-surely.*

We remark that the cardinality of the discretization $\tilde{\mathcal{X}}$ grows exponentially in the dimension of the state space, which in turn implies an exponential complexity for each verification step in our algorithm. This limitation is also present in related works on stability analysis in deterministic dynamical systems [BTSK17]. A potential approach to overcome the complexity bottleneck would be to discretize different dimensions and regions of the state space with a heterogeneous instead of a uniform granularity. In our implementation, we use one such optimization which locally refines the discretization grid at counterexamples computed by the verifier, see [LZCH21b, Supplementary Material].

Expected value computation What is left to be described is how our algorithm computes the expected value in eq. (6.3) for a given state $\mathbf{x} \in \tilde{\mathcal{X}}$. This is *not* trivial, since V_θ is a neural network and so we do not have a closed form for the expected value. However, we can bound the expected value via interval arithmetic abstract interpretation (IAAI). In particular, let $\mathbf{x} \in \tilde{\mathcal{X}}$ be a throughout fixed state for which we want to bound the expected value $\mathbb{E}_{\omega\sim d}[V_\theta(f(\mathbf{x}, \pi_\nu(\mathbf{x}), \omega))]$. Our algorithm partitions the disturbance space $\mathcal{N} \subseteq \mathbb{R}^p$ into finitely many hyperrectangular cells $\mathsf{cell}(\mathcal{N}) = \{\mathcal{N}_1, \ldots, \mathcal{N}_k\}$, with k being the number of cells. We use $\mathrm{maxvol} = \max_{\mathcal{N}_i \in \mathsf{cell}(\mathcal{N})} \mathsf{vol}(\mathcal{N}_i)$ to denote the maximal volume with respect to the Lebesgue measure over $\mathbb{R}^p$ of any cell in the partition. The algorithm then bounds the expected value via

$$\mathbb{E}_{\omega\sim d}\Big[V_\theta\Big(f(\mathbf{x}, \pi_\nu(\mathbf{x}), \omega)\Big)\Big] \leq \sum_{\mathcal{N}_i \in \mathsf{cell}(\mathcal{N})} \mathrm{maxvol} \cdot \sup_{\omega \in \mathcal{N}_i} F(\omega)$$

where $F(\omega) = V_\theta(f(\mathbf{x}, \pi_\nu(\mathbf{x}), \omega))$. Each supremum is then bounded from above via interval arithmetic by using the method of [GDS+18]. In particular, if the dependence of $f(\mathbf{x}, \pi_\nu(\mathbf{x}), \omega)$ on the stochastic disturbance ω is affine, then for each cell $\mathcal{N}_i \in \mathsf{cell}(\mathcal{N})$ we directly obtain a tight hyperrectangle $\{f(\mathbf{x}, \pi_\nu(\mathbf{x}), \omega) \mid \omega \in \mathcal{N}_i\}$ of states and then use the method of [GDS+18] to bound the supremum of V_θ over this hyperrectangle. This is the case in all our benchmarks in experimental evaluation and we observed that this method computes very tight bounds when the number of cells is sufficiently large. Otherwise, if the dependence of $f(\mathbf{x}, \pi_\nu(\mathbf{x}), \omega)$ on ω is not affine, our method relies on an off-the-shelf tool to compute a hyperrectangular over-approximation of the set $\{f(\mathbf{x}, \pi_\nu(\mathbf{x}), \omega) \mid \omega \in \mathcal{N}_i\}$. Note that maxvol is not finite in cases when $\mathcal{N}$ is unbounded. In order to allow expected value computation for an unbounded $\mathcal{N}$, we first apply the probability integral transform [Mur12] to each univariate probability distribution in d. Recall, in our model assumptions we assumed that d is a product of univariate distributions and our dynamics function f takes the most general form.

6.4.3 Learner

We now describe the learner module of our algorithm. A control policy and a candidate RSM are learned as neural networks by minimizing the following loss function

$$\mathcal{L}(\nu, \theta) = \mathcal{L}_{\mathsf{RSM}}(\nu, \theta) + \lambda \cdot \mathcal{L}_{\mathsf{Lipschitz}}(\nu, \theta). \tag{6.4}$$

The first loss term $\mathcal{L}_{\mathsf{RSM}}(\nu, \theta)$ is defined via

$$\mathcal{L}_{\mathsf{RSM}}(\theta) = \frac{1}{|\tilde{\mathcal{X}}|} \sum_{\mathbf{x} \in \tilde{\mathcal{X}}} \left(\max \left\{ \sum_{\mathbf{x}' \in \mathcal{D}_\mathbf{x}} \frac{V_\theta(\mathbf{x}')}{|\mathcal{D}_\mathbf{x}|} - V_\theta(\mathbf{x}) + \tau \cdot K, 0 \right\} \right).$$

Intuitively, for $\mathbf{x} \in \tilde{\mathcal{X}}$, the corresponding term in the sum incurs a loss whenever the condition in eq. (6.3) is violated. Since the closed form for the expected value in

eq. (6.3) in terms of parameters θ cannot be computed, for each $\mathbf{x} \in \tilde{\mathcal{X}}$ we approximate it as the mean of the values of V_θ at sampled successor states of $\mathbf{x}$ that the algorithm stores in the set $\mathcal{D}_\mathbf{x}$.

The second loss term $\lambda \cdot \mathcal{L}_{\mathsf{Lipschitz}}(\nu, \theta)$ is the regularization term used to incentivize that the Lipschitz constants L_π and L_V of π_ν and V_θ do not exceed some tolerable threshold, and hence to enforce that $\tau \cdot K$ in eq. (6.3) is sufficiently small. The constant λ is an algorithm parameter balancing the two loss terms, and we define

$$\mathcal{L}_{\mathsf{Lipschitz}}(\theta) = \max\left\{L_{V_\theta} - \frac{\delta}{\tau \cdot (L_f \cdot (L_\pi + 1) + 1)}, 0\right\}.$$

Here, δ is the threshold parameter, and L_π and L_θ are computed as in [SZS+14].

To conclude this section, we note that the loss function $\mathcal{L}(\nu, \theta)$ is nonnegative but is not necessarily equal to 0 even if $V_{\nu,\theta}$ satisfies eq. (6.3) for each $\mathbf{x} \in \tilde{\mathcal{X}}$ and its Lipschitz constant is below the allowed threshold. This is because $\mathcal{L}(\nu, \theta)$ depends on samples in $\mathcal{D}$ which are used to *approximate* the expected values in eq. (6.3). However, in Theorem 6.4.2 we show that the loss $\mathcal{L}(\nu, \theta) \to 0$ almost-surely as we add samples to the set $\mathcal{D}_\mathbf{x}$ for each $\mathbf{x} \in \tilde{\mathcal{X}}$, whenever V_θ satisfies eq. (6.3) for each $\mathbf{x} \in \tilde{\mathcal{X}}$ and the Lipschitz constant L_π and L_θ are below the allowed threshold. The claim follows from the Strong Law of Large Numbers [Wil91, Section 12.10] and the proof can be found in Section 6.7.5.

Theorem 6.4.2. *Let $M = \min_{\mathbf{x} \in \tilde{\mathcal{X}}} |\mathcal{D}_\mathbf{x}|$. If V_θ satisfies eq. (6.3) for each $\mathbf{x} \in \tilde{\mathcal{X}}$ and if $L_\pi, L_{V_\theta} \leq \delta/(\tau \cdot (L_f \cdot (L_\pi + 1) + 1))$, then $\lim_{M \to \infty} \mathcal{L}(\nu, \theta) = 0$ holds almost-surely.*

6.4.4 Formal Verification of Almost-sure Reachability

While the presentation in this chapter has focused on the formal controller synthesis problem under almost-sure reachability specifications, our approach can also be adapted to the setting in which we want to *verify* that the target set $\mathcal{X}_t$ is reached almost-surely. This can be done by simply freezing the parameters ν of the control policy and only learning and verifying an RSM. To achieve this, we simply need to replace the loss function in eq. (6.4) with

$$\mathcal{L}(\theta) = \mathcal{L}_{\mathsf{RSM}}(\theta) + \lambda \cdot \mathcal{L}_{\mathsf{Lipschitz}}(\theta).$$

The correctness of our algorithm proved in Theorem 6.4.1 then ensures that any learned and verified RSM is valid. The method for formal verification only requires that the control policy is Lipschitz continuous, which allows neural network policies with all classical activation functions [SZS+14].

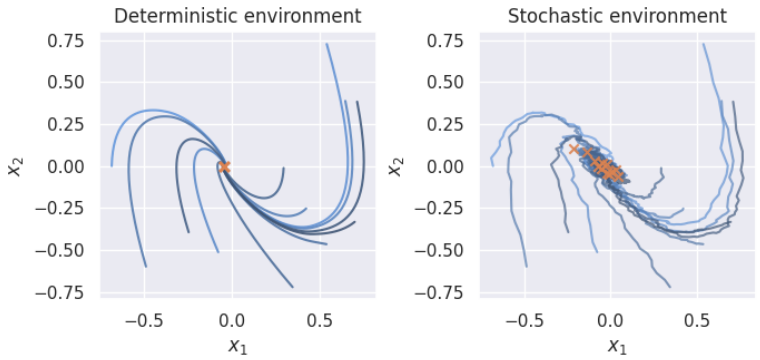

Figure 6.2: Example of a deterministic and a stochastic dynamical system with the dynamics function differing only in an additive stochastic noise term, illustrating the difficulties of proving almost-sure reachability in stochastic dynamical systems. The orange markers indicate the system state after 200 time steps.

Environment	Iters.	Mesh (τ)
2D system	4	0.002
Inverted pendulum	2	0.01

Table 6.1: Number of learner-verifier loop iterations and mesh of the discretization used by the verifier.

6.5 Experiments

We validate our algorithm empirically on two RL benchmark environments. Our first benchmark is a two-dimensional dynamical system of the form $\mathbf{x}_{t+1} = A\mathbf{x}_t + Bg(\mathbf{u}_t) + \omega$, where ω is a disturbance vector sampled from a zero-mean triangular distribution. The function g clips the control action to stay within the interval $[-1, 1]$. Our second benchmark is the inverted pendulum problem [BCP+16]. Contrarily to the standard inverted pendulum task, which has deterministic dynamics, we consider a more difficult stochastic variant. The system has two state variables x_1 and x_2 which represent the angle and the angular velocity of the pendulum. The objective of this task is to balance the pendulum in an upright position through control actions in the form of a torque that is applied to the pendulum. Our stochastic variant of the task applies a zero-mean triangular noise to both state variables. Details on both benchmarks and on our experimental setup can be found in [LZCH21b, Supplementary Material].

For each RL task, we consider the state space $\mathcal{X} = \{\mathbf{x} \mid ||\mathbf{x}||_1 \leq 0.5\}$ and initialize a control policy comprised of two hidden layers with 128 ReLU units each by using proximal policy optimization [SWD+17], while applying our Lipschitz regularization to

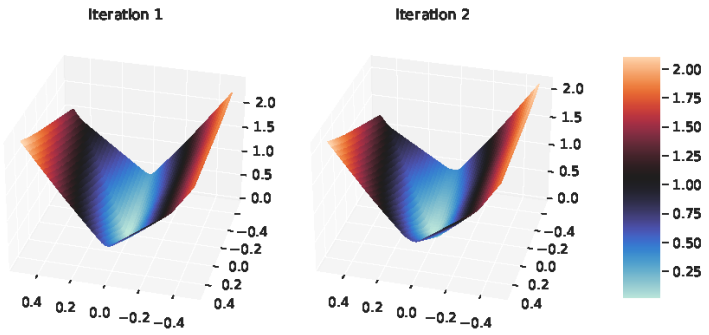

Figure 6.3: Learned RSM candidates after 1 and 2 iterations of our algorithm for the stochastic inverted pendulum task. The candidate on the left violates the expected decrease condition while the function of the right is a verified RSM.

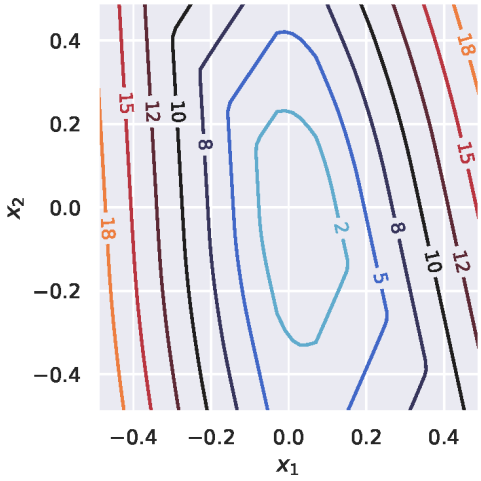

Figure 6.4: Contour lines of the expected reachability time bounds obtained from the RSM on the inverted pendulum task.

keep the Lipschitz constant of the policy within a reasonable bound. We then run our learner-verifier framework to learn and verify a control policy for which the target set $\mathcal{X}_t = \{\mathbf{x} \mid ||\mathbf{x}||_1 \leq 0.2\}$ is almost-surely reached. Our RSM neural networks consist of one hidden layer with 128 ReLU units.

Example trajectories under the initialized policy for the first benchmark with the deterministic ($\omega = 0$) and stochastic dynamics are shown in Figure 6.2. The trajectories of the deterministic system reach the origin, however this is not the case for the stochastic system. This illustrates the intricacies of verifying almost-sure reachability in stochastic dynamical systems.

Our method could successfully learn and verify a control policy and an RSM for both

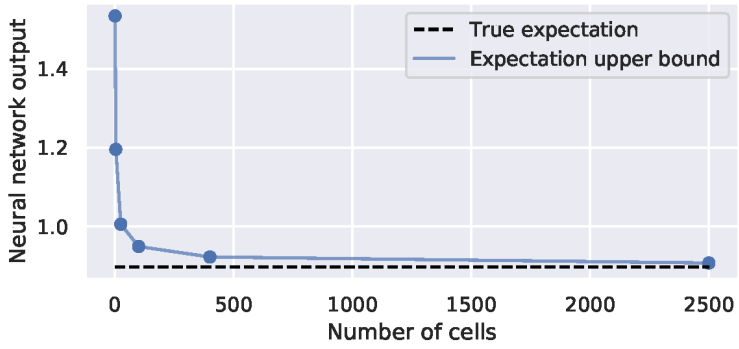

Figure 6.5: Comparison of our method for bounding the expected value of an RSM neural network with the ground-truth expected value on 100 randomly sampled states of the inverted pendulum environment.

systems, as shown in Table 6.1. For illustration, the final RSM neural network for the inverted pendulum task is shown in Figure 6.3. We further computed the ε of the RSM network according to Definition 6.3.1 for the inverted pendulum task to obtain the reachability time bounds as outlined in Theorem 6.3.3. The resulting reachability time bounds are shown in Figure 6.4.

We perform an additional experiment to study the effectiveness of our method for computing bounds on the expected value a neural network. In particular, we sample 100 random states of the inverted pendulum environment. For each sampled state, we use our method to compute the bound on the expected value of the final RSM neural network (shown in Figure 6.3) in a successor system state, with different sizes of the cell partition. We then compute the ground-truth of the expected value by averaging the RSM value at 1000 independently sampled successor states (Strong Law of Large Numbers). The results shown in Figure 6.5 indicate that, even with a modest size of the cell partition, a tight bound can be obtained. As the partition is further refined, the expected value bound converges to the ground-truth.

6.6 Related Work

Reachability and stability via Lyapunov functions. Formal controller synthesis and verification under reachability and stability specifications in *deterministic* dynamical systems has received a lot of attention in recent works. Classical methods compute a control policy together with a Lyapunov function that formally certifies reachability and stability. For systems with polynomial dynamics and Lyapunov functions restricted to the sum-of-squares (SOS) form, polynomial control policy and Lyapunov function can be computed via semi-definite programming [HG05, Par00, GK01, JWFT$^+$03]. A learner-

verifier framework similar to ours but for computing polynomial Lyapunov functions has been proposed in [RS19]. However, these methods require polynomial approximations and may not be efficient for systems with general nonlinearities. Moreover, it is known that even some simple dynamical systems that are asymptotically stable do not admit polynomial Lyapunov functions [AKP11]. Learning control policies and Lyapunov functions in the form of neural networks has been considered in [RBK18, CRG19, AAGP21], and it is an approach that is better suited to dynamical systems with general nonlinearities. In particular, [RBK18] formally verify stability by learning a Lyapunov function together with a region which the system should reach and stabilize within by first discretizing the state space of the system, then learning a Lyapunov function candidate which tries to maximize the number of the discrete states at which the Lyapunov condition holds, and finally verifying that the candidate is indeed a Lyapunov function. The works [CRG19, AAGP21] propose a learner-verifier framework which uses counterexamples found by the verifier to improve the loss function and thus learn a new candidate. This loop is repeated until the verifier certifies that the Lyapunov function is correct. Our method combines and extends ideas from these works to the setting of stochastic dynamical systems. Note, while these methods consider stability, they all assume that the stabilizing region (typically the origin) is closed under system dynamics thus they essentially study reachability.

Computation of reachable sets under neural controllers. There are several works and tools that consider *deterministic* continuous-time dynamical systems with neural network controllers and compute an over-approximation of the set of all reachable states over a given finite-time horizon. Some notable examples are Sherlock [DCS19], ReachNN [HFL+19] and ReachNN* [FHC+20] which use polynomial approximations to over-approximate the reachable set over a given time horizon, NNV [TYL+20] which is based on abstract interpretation, LRT-NG [GCL+20] which overapproximates the reachable set as sequence of hyperspheres, or Verisig [IWA+19] which reduces the problem to reachability analysis in hybrid systems. Furthermore, GoTube [GLH+21] constructs the reachable set of a deterministic continuous-time system with statistical guarantees about the constructed set overapproximating the true reachable states.

Abstraction-based methods. Abstraction-based methods provide an approach to formal controller synthesis in deterministic [Tab09] and stochastic [ADB11] dynamical systems with respect to different specifications, including almost-sure reachability. For stochastic dynamical systems, these methods construct a finite-state MDP or stochastic game approximation of the problem and use probabilistic model checking for controller synthesis. Most existing methods are applicable to finite-time horizon systems due to accumulation of the approximation error [SGA15, LKSZ20, CA19, VGO19]. Recently, a few abstraction-based controller synthesis methods for infinite-time horizon stochastic

dynamical systems with affine dynamics [HS21], control affine dynamics [DHC22] or stochastic dynamical systems with finite control input spaces [MMSS24, DHC22] have been proposed. In this chapter, we consider *infinite-time horizon and non-polynomial* stochastic dynamical systems with *continuous* control input spaces.

Other methods for stochastic dynamical systems. There are several other classes of approaches to formal controller synthesis for *finite-time horizon* stochastic dynamical systems with respect to reachability specifications. These include methods based on dynamic programming [APLS08] or Hamilton-Jacobi (HJ) reachability analysis [BCHT17].

Stability for stochastic dynamical systems. While there are several theoretical results on the stability of stochastic dynamical systems (see [Kus14] for a comprehensive survey), to our best knowledge there are very few works that consider their automated stability verification without the reduction to reachability analysis [Vai15, CS03]. Both of these are numerical approaches that first partition the system's state space into finitely many regions and then over-approximate the system's continuous dynamics via a discrete finite-state abstraction. Thus, the computed stability certificates are piecewise-constant. Furthermore, [Vai15] verifies a weaker notion of stability called "coarse stochastic stability" that depends on the partition of the state space, and [CS03] imposes stability by requiring the system to reach the stabilizing region within some pre-specified finite time and deterministically (i.e. for each sample path).

Safe exploration. RL algorithms need to explore the environment via randomized actions to learn which actions lead to a high future reward. However, in safety-critical environments, random actions may lead to catastrophic results. Safe exploration RL aims to restrict the exploratory actions to those that ensure safety of the environment. The most dominant approach to addressing this problem is learning the system dynamics' uncertainty bounds and limiting the exploratory actions within a high probability safety region. In the literature, Gaussian Processes [KBTK18, TBK19, Ber19], linearized models [DDV$^+$18] , deep robust regression [LSC$^+$20], and Bayesian neural networks [LZCH21a] are used to learn uncertainty bounds.

Learning stable dynamics. Learning dynamics from observation data is the first step in many control methods as well as model-based RL. Recent works considered learning deterministic system dynamics with guarantees on stability of some specified region [KM19]. Learning stochastic dynamics from observation data has been studied in [UH17, LLF$^+$20].

RSMs for probabilistic programs. Finally, as has been extensively discussed in this thesis, ranking supermartingales (RSMs) were first introduced in the programming

languages community in order to reason about termination of PPs [CS13]. Our theoretical results instantiate RSMs to the setting of stochastic dynamical systems and show that they provide a formal certificate for almost-sure reachability. While our theoretical results are motivated by the works on PPs, our approach to their computation differs significantly from the existing methods for RSM computation in PPs [CS13, CFNH16, CFG16]. In particular, these methods compute linear/polynomial RSMs via linear/semi-definite programming, and are more similar to the early methods for the computation of polynomial Lyapunov functions that we discussed above. On the contrary, our method learns an RSM in the form of a neural network. The only method for learning RSMs in PPs has been presented in the recent work of [AGR21] which computes neural network RSMs with a single hidden layer. In contrast, one of the main algorithmic novelties of our work is that we propose a general framework for computing the expected value of a neural network function over some probability distribution, which allows us to learn multi-layer neural network RSMs for general nonlinear systems.

6.7 Technical Proofs

6.7.1 Further Background on Martingale Theory

Some of the proofs of results in this chapter assume the background on probability and martingale theory presented in Sections 2.1 and 2.4. In addition, we need to recall the mathematical notion of ranking supermartingales in probability spaces and two standard result on them before proving some of the theorems.

Ranking supermartingale. We define the mathematical notion of ranking supermartingales. Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space and let $\varepsilon \geq 0$. Suppose that T is a stopping time with respect to a filtration $\{\mathcal{F}_i\}_{i=0}^{\infty}$. An *$\varepsilon$-ranking supermartingale (ε-RSM) with respect to T* is a stochastic process $(X_i)_{i=0}^{\infty}$ such that

- X_i is $\mathcal{F}_i$-measurable, for each $i \geq 0$,
- $X_i(\omega) \geq 0$, for each $i \geq 0$ and $\omega \in \Omega$, and
- $\mathbb{E}[X_{i+1} \mid \mathcal{F}_i](\omega) \leq X_i(\omega) - \varepsilon \cdot \mathbb{I}_{T>i}(\omega)$, for each $i \geq 0$ and $\omega \in \Omega$.

We now state two results on RSMs that we will use in our proofs. A variant of the first result was presented in works on termination analysis of probabilistic programs [FH15, CFNH16]. Since our variant slightly differs from the ones presented in those works, we provide the proof for the sake of completeness (while previous variants consider the special case of stopping times defined by the first time after which the value of an RSM falls below some given threshold, our variant considers general stopping times).

Proposition 6.7.1. *Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space, let $(\mathcal{F}_i)_{i=0}^{\infty}$ be a filtration and let T be a stopping time with respect to $(\mathcal{F}_i)_{i=0}^{\infty}$. Suppose that $(X_i)_{i=0}^{\infty}$ is an ε-RSM with respect to T, for some $\varepsilon > 0$. Then*

1. $\mathbb{P}[T < \infty] = 1$,
2. $\mathbb{E}[T] \leq \frac{\mathbb{E}[X_0]}{\varepsilon}$, *and*
3. $\mathbb{P}[T \geq t] \leq \frac{\mathbb{E}[X_0]}{\varepsilon \cdot t}$, *for each $t \in \mathbb{N}$.*

Proof. We first prove by induction on i that, for each $i \in \mathbb{N}$,

$$\mathbb{E}[X_i] \leq \mathbb{E}[X_0] - \varepsilon \cdot \sum_{j=0}^{i-1} \mathbb{P}[T > j]. \tag{6.5}$$

The base case $i = 1$ follows immediately from the definition of an ε-RSM. We now suppose that the claim is true for i, and we prove it for $i + 1$:

$$\begin{aligned}\mathbb{E}[X_{i+1}] = \mathbb{E}\Big[\mathbb{E}[X_{i+1} \mid \mathcal{F}_i]\Big] &\leq \mathbb{E}[X_i] - \varepsilon \cdot \mathbb{P}[T > i] \\ &\leq \mathbb{E}[X_0] - \varepsilon \cdot \sum_{j=0}^{i} \mathbb{P}[T > i],\end{aligned}$$

where the first inequality holds since $(X_i)_{i=0}^{\infty}$ is an ε-RSM with respect to T, and the second inequality holds by the induction hypothesis. Hence, the claim follows.

But we know that $X_i(\omega) \geq 0$ for each i, ω by the definition of an ε-RSM, hence by plugging $\mathbb{E}[X_i] \geq 0$ into eq. (6.5) we conclude that $0 \leq \mathbb{E}[X_0] - \varepsilon \cdot \sum_{j=0}^{i-1} \mathbb{P}[T > j]$ for each $i \in \mathbb{N}$, and thus

$$\sum_{j=0}^{\infty} \mathbb{P}[T > j] \leq \frac{\mathbb{E}[X_0]}{\varepsilon} < \infty.$$

It then follows that:

1. $\mathbb{P}[T = \infty] = \lim_{t \to \infty} \mathbb{P}[T > t] = 0$, as $\sum_{j=0}^{\infty} \mathbb{P}[T > j]$ converges,
2. $\mathbb{E}[T] = \sum_{t=0}^{\infty} \mathbb{P}[T > t] \leq \frac{\mathbb{E}[X_0]}{\varepsilon}$, and
3. $\mathbb{P}[T \geq t] \leq \frac{\mathbb{E}[T]}{t} \leq \frac{\mathbb{E}[X_0]}{\varepsilon \cdot t}$, where the first inequality follows by Markov's inequality.

Hence, the proposition claim follows. □

The second is a classical result on supermartingales (and therefore RSMs) called Azuma's inequality [Azu67], that we will later use in proving the concentration bounds on the reachability time.

Proposition 6.7.2 (Azuma's inequality)**.** *Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space and let $(\mathcal{F}_i)_{i=0}^{\infty}$ be a filtration. Suppose that $(X_i)_{i=0}^{\infty}$ is a supermartingale with respect to $(\mathcal{F}_i)_{i=0}^{\infty}$, and let $(c_i)_{i=0}^{\infty}$ be a sequence of positive real numbers such that $|X_{i+1}(\omega) - X_i(\omega)| \leq c_i$ for each $i \geq 0$ and $\omega \in \Omega$. Then, for each $n \in \mathbb{N}$ and $t > 0$, we have that*

$$\mathbb{P}\Big[X_n - X_0 \geq t\Big] \leq e^{\frac{-t^2}{2 \cdot \sum_{i=0}^{n-1} c_i^2}}. \tag{6.6}$$

6.7.2 Proof of Theorem 6.3.2

Theorem. *Let $f : \mathcal{X} \times \mathcal{U} \times \mathcal{N} \to \mathcal{X}$ be a continuous dynamics function, $\pi : \mathcal{X} \to \mathcal{U}$ a continuous control policy and d a probability distribution over $\mathcal{N}$. Let $\mathcal{X}_t \subseteq \mathcal{X}$. Suppose that there exists an RSM $V : \mathcal{X} \to \mathbb{R}$ for $\mathcal{X}_t$. Then, for every initial state $\mathbf{x}_0 \in \mathcal{X}$, we have $\mathbb{P}^{\pi}_{\mathbf{x}_0}[\mathsf{Reach}(\mathcal{X}_t)] = 1$.*

Proof. Recall from Section 2.3 that, for each initial state $\mathbf{x}_0 \in \mathcal{X}$, the system dynamics induces a Markov process which gives rise to the probability space over the set of all system trajectories that start in the initial state $\mathbf{x}_0$. Denote this probability space by $(\Omega_{\mathbf{x}_0}, \mathcal{F}_{\mathbf{x}_0}, \mathbb{P}_{\mathbf{x}_0})$. The idea behind the proof is to show that any RSM for $\mathcal{X}_t$ gives rise to a mathematical RSM in the probability space $(\Omega_{\mathbf{x}_0}, \mathcal{F}_{\mathbf{x}_0}, \mathbb{P}_{\mathbf{x}_0})$. We then use Proposition 6.7.1 to deduce a.s. reachability.

In order to formally show that an RSM can be instantiated as a mathematical object in this probability space, we first need to define the canonical filtration in this probability space. This will allow defining the reachability time as a stopping time with respect to the canonical filtration, and finally instantiating the RSM in our stochastic dynamical system as a mathematical RSM with respect to this stopping time. The rest of this section formalizes the intuition behind the proofs that was outlined above.

Canonical filtration Fix an initial state $\mathbf{x}_0 \in \mathcal{X}$ and consider the probability space $(\Omega_{\mathbf{x}_0}, \mathcal{F}_{\mathbf{x}_0}, \mathbb{P}_{\mathbf{x}_0})$. For each $i \in \mathbb{N}_0$, define $\mathcal{F}_i \subseteq \mathcal{F}$ to be the σ-algebra containing the subsets of $\Omega_{\mathbf{x}_0}$ that, intuitively, contain all trajectories in $\Omega_{\mathbf{x}_0}$ whose first i states satisfy some specified property. Formally, we define $\mathcal{F}_i$ as follows. For each $j \in \mathbb{N}_0$, let $C_j : \Omega_{\mathbf{x}_0} \to \mathcal{X}$ be a map which to each trajectory $\rho = (\mathbf{x}_t, \mathbf{u}_t, \omega_t)_{t \in \mathbb{N}_0} \in \Omega_{\mathbf{x}_0}$ assigns the j-th state $\mathbf{x}_j$ along the trajectory. Then $\mathcal{F}_i$ is the smallest σ-algebra over $\Omega_{\mathbf{x}_0}$ with respect to which $C_0, C_1, \ldots, C_i$ are all measurable, where $\mathcal{X} \subseteq \mathbb{R}^m$ is equipped with the induced Borel-σ-algebra [Wil91, Section 1]. Clearly $\mathcal{F}_0 \subseteq \mathcal{F}_1 \subseteq \ldots$. We say that the sequence of σ-algebras $(\mathcal{F}_i)_{i=0}^{\infty}$ is the *canonical filtration* in the probability space $(\Omega_{\mathbf{x}_0}, \mathcal{F}_{\mathbf{x}_0}, \mathbb{P}_{\mathbf{x}_0})$.

Reachability stopping time In order to formally reason about a.s. reachability and the reachability time of the target set $\mathcal{X}_t$, we formalize the notion of the reachability stopping time. Define $T_{\mathcal{X}_t} : \Omega_{\mathbf{x}_0} \to \mathbb{N}_0 \cup \{\infty\}$ to be the first hitting time of the set $\mathcal{X}_t$. Since whether $T_{\mathcal{X}_t}(\rho) \leq i$ depends solely on the first i states along ρ, we clearly have $\{\rho \in \Omega_{\mathbf{x}_0} \mid T(\rho) \leq i\} \in \mathcal{F}_i$ for each i and so $T_{\mathcal{X}_t}$ is a stopping time with respect to $(\mathcal{F}_i)_{i=0}^{\infty}$. We call $T_{\mathcal{X}_t}$ the *reachability stopping time*.

We now prove the theorem claim that $\mathcal{X}_t$ is reached almost-surely, i.e. that for every initial state $\mathbf{x}_0 \in \mathcal{X}$, we have $\mathbb{P}^{\pi}_{\mathbf{x}_0}[\text{Reach}(\mathcal{X}_t)] = 1$. Let $\mathbf{x}_0 \in \mathcal{X}$. If $\mathbf{x}_0 \in \mathcal{X}_t$, then the claim trivially holds. Thus suppose without loss of generality that $\mathbf{x}_0 \notin \mathcal{X}_t$, and consider the probability space $(\Omega_{\mathbf{x}_0}, \mathcal{F}_{\mathbf{x}_0}, \mathbb{P}_{\mathbf{x}_0})$, the canonical filtration $(\mathcal{F}_i)_{i=0}^{\infty}$ and the reachability stopping time $T_{\mathcal{X}_t}$ in it.

Now, we define a stochastic process $(X_i)_{i=0}^{\infty}$ in $(\Omega_{\mathbf{x}_0}, \mathcal{F}_{\mathbf{x}_0}, \mathbb{P}_{\mathbf{x}_0})$ via

$$X_i(\rho) = \begin{cases} V(\mathbf{x}_i), & \text{if } i < T_{\mathcal{X}_t}(\rho) \\ V(\mathbf{x}_{T_{\mathcal{X}_t}(\rho)}), & \text{otherwise} \end{cases}$$

for each $i \geq 0$ and $\rho = (\mathbf{x}_t, \mathbf{u}_t, \omega_t)_{t \in \mathbb{N}_0} \in \Omega_{\mathbf{x}_0}$. Hence, if the stopping time $T_{\mathcal{X}_t}$ is not exceeded by time i we define X_i to be equal to the value of V at the i-th state along the trajectory, and after $T_{\mathcal{X}_t}$ is exceeded we define X_i to be equal to the value of V at the time step $T_{\mathcal{X}_t}$ at which the process is stopped.

We claim that $(X_i)_{i=0}^{\infty}$ is an ε-RSM with respect to the reachability stopping time $T_{\mathcal{X}_t}$. To prove this claim, we check each of the three defining properties of ε-RSMs:

- *Each X_i is $\mathcal{F}_i$-measurable.* The value of X_i is defined in terms of the first i states along a trajectory if $T_{\mathcal{X}_t} > i$, and in terms of the first $T_{\mathcal{X}_t}$ states if $i \geq T_{\mathcal{X}_t}$. By the definition of the canonical filtration, we have that X_i is $\mathcal{F}_i$-measurable for each $i \geq 0$.

- *Each $X_i(\rho) \geq 0$.* Since each X_i is defined in terms of V and since we know that $V(\mathbf{x}) \geq 0$ for each state $\mathbf{x} \in \mathcal{X}$, it follows that $X_i(\rho) \geq 0$ for each $i \geq 0$ and $\rho \in \Omega_{\mathbf{x}_0}$.

- *Each $\mathbb{E}[X_{i+1} \mid \mathcal{F}_i](\rho) \leq X_i(\rho) - \varepsilon \cdot \mathbb{I}_{T_{\mathcal{X}_t} > i}(\rho)$.* First, note that the conditional expectation exists since X_{i+1} is nonnegative for each $i \geq 0$. In order to prove the inequality, we distinguish two cases.

  First, if $T_{\mathcal{X}_t}(\rho) > i$, we need to show that $\mathbb{E}[X_{i+1} \mid \mathcal{F}_i](\rho) \leq X_i(\rho) - \varepsilon$. Let $\rho = (\mathbf{x}_t, \mathbf{u}_t, \omega_t)_{t \in \mathbb{N}_0}$. We have that $X_i(\rho) = V(\mathbf{x}_i)$. On the other hand, we have $\mathbb{E}[X_{i+1} \mid \mathcal{F}_i](\rho) = \mathbb{E}_{\omega \sim d}[V(f(\mathbf{x}_i, \pi(\mathbf{x}_i), \omega)]$. To see this, observe that $\mathbb{E}_{\omega \sim d}[V(f(\mathbf{x}_i, \pi(\mathbf{x}_i), \omega)]$ satisfies all the defining properties of conditional

expectation since it is the expected value of V at a subsequent state of $\mathbf{x}_i$, and recall that conditional expectation is a.s. unique whenever it exists. We thus have

$$\begin{aligned}\mathbb{E}[X_{i+1} \mid \mathcal{F}_i](\rho) &= \mathbb{E}_{\omega\sim d}[V(f(\mathbf{x}_i, \pi(\mathbf{x}_i), \omega)] \\ &\leq V(\mathbf{x}_i) - \varepsilon = X_i(\rho) - \varepsilon,\end{aligned}$$

where the inequality holds since V is an RSM for $\mathcal{X}_t$ and $\mathbf{x}_i \notin \mathcal{X}_t$ (as $T_{\mathcal{X}_t}(\rho) > i$). This proves the claim.

Second, if $T_{\mathcal{X}_t}(\rho) \leq i$, we need to show that $\mathbb{E}[X_{i+1} \mid \mathcal{F}_i](\rho) \leq X_i(\rho)$. Let $\rho = (\mathbf{x}_t, \mathbf{u}_t, \omega_t)_{t\in\mathbb{N}_0}$. We have $X_i(\rho) = V(\mathbf{x}_{T_{\mathcal{X}_t}(\rho)})$ and $\mathbb{E}[X_{i+1} \mid \mathcal{F}_i](\rho)] = V(\mathbf{x}_{T_{\mathcal{X}_t}(\rho)})$, so the equality follows.

This concludes the proof that $(X_i)_{i=0}^{\infty}$ is an ε-RSM with respect to $T_{\mathcal{X}_t}$. Therefore, by the first part of Proposition 1, we have that $\mathbb{P}_{\mathbf{x}_0}[T_{\mathcal{X}_t} < \infty] = 1$ which by the definition of the reachability stopping time implies $\mathbb{P}^{\pi}_{\mathbf{x}_0}[\mathsf{Reach}(\mathcal{X}_t)] = 1$. Since the initial state $\mathbf{x}_0$ was arbitrary, the claim follows. □

6.7.3 Proof of Theorem 6.3.3

Theorem. *Let $f : \mathcal{X} \times \mathcal{U} \times \mathcal{N} \to \mathcal{X}$ be a continuous dynamics function, $\pi : \mathcal{X} \to \mathcal{U}$ a continuous control policy and d a probability distribution over $\mathcal{N}$. Let $\mathcal{X}_t \subseteq \mathcal{X}$. Suppose that there exists an ε-RSM $V : \mathcal{X} \to \mathbb{R}$ for $\mathcal{X}_t$. Then, for any initial state $\mathbf{x}_0 \in \mathcal{X}$,*

1. $\mathbb{E}_{\mathbf{x}_0}[T_{\mathcal{X}_t}] \leq \frac{V(\mathbf{x}_0)}{\varepsilon}$.
2. $\mathbb{P}_{\mathbf{x}_0}[T_{\mathcal{X}_t} \geq t] \leq \frac{V(\mathbf{x}_0)}{\varepsilon \cdot t}$, *for any time* $t \in \mathbb{N}$.
3. *If the system has* c-bounded differences *for* $c > 0$, *then* $\mathbb{P}_{\mathbf{x}_0}[T_{\mathcal{X}_t} \geq t] \leq A \cdot e^{-t\cdot\varepsilon^2/(2\cdot(c+\varepsilon)^2)}$ *for any time* $t \in \mathbb{N}$ *and* $A = e^{\varepsilon\cdot V(\mathbf{x}_0)/(c+\varepsilon)^2}$.

Proof. In the proof of Theorem 6.3.2 in Section 6.7.2, we showed that $(X_i)_{i=0}^{\infty}$ is an ε-RSM with respect to $T_{\mathcal{X}_t}$. The first two parts of the theorem then follow from the second and the third item of Proposition 6.7.1.

The proof of the third part of the theorem is similar to the argument in [CFNH16, Section 5.1.2] which derives concentration bounds on the termination time in PPs. We define another stochastic process $(Y_i)_{i=0}^{\infty}$ as follows:

$$Y_i(\rho) = X_i(\rho) + \varepsilon \cdot \min\{i, T_{\mathcal{X}_t}(\rho)\}.$$

We claim that $(Y_i)_{i=0}^{\infty}$ is a supermartingale with respect to the canonical filtration $(\mathcal{F}_i)_{i=0}^{\infty}$. By applying Azuma's inequality to this newly constructed supermartingale, we will then deduce the statement of the third item in the theorem.

To prove this claim, note that each Y_i is $\mathcal{F}_i$-measurable and nonnegativity clearly holds, so we just need to check that $\mathbb{E}[Y_{i+1} \mid \mathcal{F}_i](\rho) \leq Y_i(\rho)$ for each $i \geq 0$ and $\rho \in \Omega_{\mathbf{x}_0}$. To prove this, observe that

$$\begin{aligned}
\mathbb{E}[Y_{i+1} \mid \mathcal{F}_i](\rho) &= \mathbb{E}\Big[X_{i+1} + \varepsilon \cdot \min\{i+1, T_{\mathcal{X}_t}\} \mid \mathcal{F}_i\Big](\rho) \\
&= \mathbb{E}\Big[X_{i+1} \mid \mathcal{F}_i\Big](\rho) + \varepsilon \cdot \mathbb{E}\Big[\min\{i+1, T_{\mathcal{X}_t}\} \mid \mathcal{F}_i\Big](\rho) \\
&\leq X_i(\rho) - \varepsilon \cdot \mathbb{I}_{T_{\mathcal{X}_t} > i}(\rho) + \varepsilon \cdot \mathbb{E}[\min\{i+1, T_{\mathcal{X}_t}\} \mid \mathcal{F}_i](\rho)
\end{aligned} \tag{6.7}$$

Now, we distinguish between two cases.

- If $T_{\mathcal{X}_t}(\rho) > i$, then $\min\{i+1, T_{\mathcal{X}_t}\} = i+1$ and so $\mathbb{E}[\min\{i+1, T_{\mathcal{X}_t}\} \mid \mathcal{F}_i](\rho) = i+1$. Then, continuing on the right-hand-side of eq. (6.7), we have

$$\begin{aligned}
\mathbb{E}[Y_{i+1} \mid \mathcal{F}_i](\rho) &= \mathbb{E}\Big[X_{i+1} + \varepsilon \cdot \min\{i+1, T_{\mathcal{X}_t}\} \mid \mathcal{F}_i\Big](\rho) \\
&\leq X_i(\rho) - \varepsilon \cdot \mathbb{I}_{T_{\mathcal{X}_t} > i}(\rho) + \varepsilon \cdot \mathbb{E}[\min\{i+1, T_{\mathcal{X}_t}\} \mid \mathcal{F}_i](\rho) \\
&= X_i(\rho) - \varepsilon + \varepsilon \cdot (i+1) \\
&= X_i(\rho) - \varepsilon \cdot i = Y_i(\rho)
\end{aligned}$$

- If $T_{\mathcal{X}_t}(\rho) \leq i$, then $\min\{i+1, T_{\mathcal{X}_t}\}(\rho) = T_{\mathcal{X}_t}(\rho) = T_{\mathcal{X}_t}(\rho) \cdot \mathbb{I}_{T_{\mathcal{X}_t} \leq i}(\rho)$. But the random variable $T_{\mathcal{X}_t} \cdot \mathbb{I}_{T_{\mathcal{X}_t} \leq i}$ is $\mathcal{F}_i$-measurable, so by the properties of conditional expectation we have that $\mathbb{E}[T_{\mathcal{X}_t} \cdot \mathbb{I}_{T_{\mathcal{X}_t} \leq i} \mid \mathcal{F}_i] = T_{\mathcal{X}_t} \cdot \mathbb{I}_{T_{\mathcal{X}_t} \leq i}$. Plugging this back into the right-hand-side of eq. (6.7), we have

$$\begin{aligned}
\mathbb{E}[Y_{i+1} \mid \mathcal{F}_i](\rho) &= \mathbb{E}\Big[X_{i+1} + \varepsilon \cdot \min\{i+1, T_{\mathcal{X}_t}\} \mid \mathcal{F}_i\Big](\rho) \\
&\leq X_i(\rho) - \varepsilon \cdot \mathbb{I}_{T_{\mathcal{X}_t} > i}(\rho) + \varepsilon \cdot \mathbb{E}[\min\{i+1, T_{\mathcal{X}_t}\} \mid \mathcal{F}_i](\rho) \\
&= X_i(\rho) - 0 + \varepsilon \cdot T_{\mathcal{X}_t}(\rho) = Y_i(\rho).
\end{aligned}$$

Hence, $(Y_i)_{i=0}^{\infty}$ is a supermartingale with respect to the canonical filtration $(\mathcal{F}_i)_{i=0}^{\infty}$. Moreover, note that $(Y_i)_{i=0}^{\infty}$ has $(c+\varepsilon)$-bounded differences, as $(X_i)_{i=0}^{\infty}$ has c-bounded differneces.

Finally, the statement the third item of the theorem follows from the following sequence of inequalities

$$\begin{aligned}
&\mathbb{P}_{\mathbf{x}_0}\Big[T_{\mathcal{X}_t} \geq t\Big] = \mathbb{P}_{\mathbf{x}_0}\Big[X_t \geq 0 \wedge T_{\mathcal{X}_t} \geq t\Big] \\
&= \mathbb{P}_{\mathbf{x}_0}\Big[X_t + \varepsilon \cdot \min\{t, T_{\mathcal{X}_t}\} - X_0 \geq \varepsilon \cdot \min\{t, T_{\mathcal{X}_t}\} - X_0 \wedge T_{\mathcal{X}_t} \geq t\Big] \\
&= \mathbb{P}_{\mathbf{x}_0}\Big[X_t + \varepsilon \cdot \min\{t, T_{\mathcal{X}_t}\} - X_0 \geq \varepsilon \cdot t - X_0 \wedge T_{\mathcal{X}_t} \geq t\Big] \\
&\leq \mathbb{P}_{\mathbf{x}_0}\Big[X_t + \varepsilon \cdot \min\{t, T_{\mathcal{X}_t}\} - X_0 \geq \varepsilon \cdot t - X_0\Big] \\
&= \mathbb{P}_{\mathbf{x}_0}\Big[Y_t - Y_0 \geq \varepsilon \cdot t - X_0\Big] \\
&\leq e^{\frac{-(\varepsilon \cdot t - X_0)^2}{2 \cdot \sum_{i=0}^{t-1} (c+\varepsilon)^2}} = e^{\frac{-(\varepsilon \cdot t - X_0)^2}{2 \cdot t \cdot (c+\varepsilon)^2}} = e^{\frac{-\varepsilon^2 \cdot t}{2 \cdot (c+\varepsilon)^2}} \cdot e^{\frac{\varepsilon \cdot X_0}{(c+\varepsilon)^2}} \cdot e^{\frac{-X_0^2}{2 \cdot t \cdot (c+\varepsilon)^2}} \\
&= e^{\frac{-\varepsilon^2 \cdot t}{2 \cdot (c+\varepsilon)^2}} \cdot e^{\frac{\varepsilon \cdot V(\mathbf{x}_0)}{(c+\varepsilon)^2}} \cdot e^{\frac{-V(\mathbf{x}_0)^2}{2 \cdot t \cdot (c+\varepsilon)^2}} \leq e^{\frac{-\varepsilon^2 \cdot t}{2 \cdot (c+\varepsilon)^2}} \cdot e^{\frac{\varepsilon \cdot V(\mathbf{x}_0)}{(c+\varepsilon)^2}} \cdot 1 \\
&= A \cdot e^{-t \cdot \frac{\varepsilon^2}{2 \cdot (c+\varepsilon)^2}}
\end{aligned}$$

with $A = e^{\varepsilon \cdot V(\mathbf{x}_0)/(c+\varepsilon)^2}$, where in sixth row we applied Azuma's inequality to the supermartingale $(Y_i)_{i=0}^{\infty}$ and in the ninth row we use $e^{-V(\mathbf{x}_0)^2/(2 \cdot t \cdot (c+\varepsilon)^2)} \leq 1$. □

6.7.4 Proof of Theorem 6.4.1

Theorem. *Suppose that the verifier in Algorithm 6.1 verifies that V_θ satisfies eq. (6.3) for each $\mathbf{x} \in \tilde{\mathcal{X}}$. Let $-m \in \mathbb{R}$ be such that $V_\theta(\mathbf{x}) \geq -m$ for each $\mathbf{x} \in \mathcal{X}$. Then, the function $V'_\theta(\mathbf{x}) = V_\theta(\mathbf{x}) + m$ is an RSM for $\mathcal{X}_t$. Hence, $\mathcal{X}_t$ is reached almost-surely.*

Proof. The function V' is continuous since V is continuous. Moreover, V' is nonnegative since $V'(\mathbf{x}) = V(\mathbf{x}) + m \geq -m + m = 0$ for each $\mathbf{x} \in \mathcal{X}$. Thus, we only need to show that V' satisfies the expected decrease condition for each $\mathbf{x} \in \mathcal{X} \backslash \mathcal{X}_t$, i.e. that there exists $\varepsilon > 0$ such that

$$\mathbb{E}_{\omega \sim d}\Big[V\Big(f(\mathbf{x}, \pi(\mathbf{x}), \omega)\Big)\Big] \leq V(\mathbf{x}) - \varepsilon$$

for each $\mathbf{x} \in \mathcal{X} \backslash \mathcal{X}_t$. We prove that $\varepsilon > 0$ defined via

$$\varepsilon = \min_{\mathbf{x} \in \tilde{\mathcal{X}}} \Big(V(\mathbf{x}) - \tau \cdot K - \mathbb{E}_{\omega \sim d}\Big[V\Big(f(\mathbf{x}, \pi(\mathbf{x}), \omega)\Big)\Big]\Big)$$

satisfies the claim, where by theorem assumptions we know that ε is indeed strictly positive. To show this, fix $\mathbf{x} \in \mathcal{X} \backslash \mathcal{X}_t$ and let $\tilde{\mathbf{x}} \in \tilde{\mathcal{X}}$ be such that $||\mathbf{x} - \tilde{\mathbf{x}}||_1 \leq \tau$. Such

$\tilde{\mathbf{x}}$ exists by the definition of a discretization. Then, by Lipschitz continuity of f, π and V, we have

$$\begin{aligned}
&\mathbb{E}_{\omega\sim d}\Big[V\Big(f(\mathbf{x},\pi(\mathbf{x}),\omega)\Big)\Big] \\
&\leq \mathbb{E}_{\omega\sim d}\Big[V\Big(f(\tilde{\mathbf{x}},\pi(\tilde{\mathbf{x}}),\omega)\Big)\Big] + ||f(\tilde{\mathbf{x}},\pi(\tilde{\mathbf{x}}),\omega) - f(\mathbf{x},\pi(\mathbf{x}),\omega)||_1 \cdot L_V \\
&\leq \mathbb{E}_{\omega\sim d}\Big[V\Big(f(\tilde{\mathbf{x}},\pi(\tilde{\mathbf{x}}),\omega)\Big)\Big] + ||(\tilde{\mathbf{x}},\pi(\tilde{\mathbf{x}}),\omega) - (\mathbf{x},\pi(\mathbf{x}),\omega)||_1 \cdot L_V \cdot L_f \\
&\leq \mathbb{E}_{\omega\sim d}\Big[V\Big(f(\tilde{\mathbf{x}},\pi(\tilde{\mathbf{x}}),\omega)\Big)\Big] + ||\tilde{\mathbf{x}} - \mathbf{x}||_1 \cdot L_V \cdot L_f \cdot (1 + L_\pi) \\
&\leq \mathbb{E}_{\omega\sim d}\Big[V\Big(f(\tilde{\mathbf{x}},\pi(\tilde{\mathbf{x}}),\omega)\Big)\Big] + \tau \cdot L_V \cdot L_f \cdot (1 + L_\pi),
\end{aligned} \tag{6.8}$$

where in the last row we used $||\mathbf{x} - \tilde{\mathbf{x}}||_1 < \tau$. On the other hand, by Lipschitz continuity of V we have

$$V(\mathbf{x}) \geq V(\tilde{\mathbf{x}}) + ||\tilde{\mathbf{x}} - \mathbf{x}||_1 \cdot L_V \geq V(\tilde{\mathbf{x}}) - \tau \cdot L_V. \tag{6.9}$$

Thus combining eq.(6.8) and (6.9) we get

$$\begin{aligned}
&V(\mathbf{x}) - \mathbb{E}_{\omega\sim d}\Big[V\Big(f(\mathbf{x},\pi(\mathbf{x}),\omega)\Big)\Big] \\
&\geq V(\tilde{\mathbf{x}}) - \tau \cdot L_V - \mathbb{E}_{\omega\sim d}\Big[V\Big(f(\tilde{\mathbf{x}},\pi(\tilde{\mathbf{x}}),\omega)\Big)\Big] - \tau \cdot L_V \cdot L_f \cdot (1 + L_\pi) \\
&= V(\tilde{\mathbf{x}}) - \tau \cdot K - \mathbb{E}_{\omega\sim d}\Big[V\Big(f(\tilde{\mathbf{x}},\pi(\tilde{\mathbf{x}}),\omega)\Big)\Big] \\
&\geq \varepsilon
\end{aligned} \tag{6.10}$$

where the equality in the second last row follows by the definition of K, and the inequality in the last row follows by our choice of ε. This concludes the proof that V' is an RSM for $\mathcal{X}_t$. The claim that reached almost-surely $\mathcal{X}_t$ then follows from Theorem 6.3.2. □

6.7.5 Proof of Theorem 6.4.2

Theorem. *Let $M = \min_{\mathbf{x}\in\tilde{\mathcal{X}}} |\mathcal{D}_{\mathbf{x}}|$. If V_θ satisfies eq. (6.3) for each $\mathbf{x} \in \tilde{\mathcal{X}}$ and if $L_\pi, L_{V_\theta} \leq \delta/(\tau \cdot (L_f \cdot (L_\pi + 1) + 1))$, then $\lim_{M\to\infty} \mathcal{L}(\nu,\theta) = 0$ holds almost-surely.*

Proof. Since $L_{V_\theta} \leq \delta/(\tau \cdot (L_f \cdot (L_\pi + 1) + 1))$, we have that $\mathcal{L}_{\text{Lipschitz}}(\theta) = 0$. Thus,

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{RSM}}(\theta) = \frac{1}{|\tilde{\mathcal{X}}|} \sum_{\mathbf{x}\in\tilde{\mathcal{X}}} \Big(\max\Big\{ \sum_{\mathbf{x}'\in\mathcal{D}_{\mathbf{x}}} \frac{V_\theta(\mathbf{x}')}{|\mathcal{D}_{\mathbf{x}}|} - V_\theta(\mathbf{x}) + \tau \cdot K, 0 \Big\}\Big).$$

Hence, it suffices to prove that for each $\mathbf{x} \in \tilde{\mathcal{X}}$ we almost-surely have

$$\lim_{M\to\infty} \max\left\{ \sum_{\mathbf{x}'\in\mathcal{D}_\mathbf{x}} \frac{V_\theta(\mathbf{x}')}{|\mathcal{D}_\mathbf{x}|} - V_\theta(\mathbf{x}) + \tau \cdot K, 0 \right\} = 0.$$

To prove this, observe that $\{V_\theta(\mathbf{x}') \mid \mathbf{x}' \in \mathcal{D}_\mathbf{x}\}$ is a set of values of V in at least M independently sampled successor states of $\mathbf{x}$ according to the distribution over the successor states of $\mathbf{x}$ defined by the system dynamics and the probability distribution d over disturbance vectors. Since $\mathcal{X}$ is compact and V_θ is continuous, the random value defined by the value of V at a randomly sampled successor state of $\mathbf{x}$ is bounded, thus has a well-defined and finite first moment.

The Strong Law of Large Numbers [Wil91, Section 12.10] states that, given a distribution μ with a finite first moment and a sequence $X_1, X_2, \ldots$ of independent identically distributed random variables distributed according to μ, we have that

$$\lim_{n\to\infty} \frac{X_1 + \cdots + X_n}{n} = \mathbb{E}_{X\sim\mu}[X]$$

holds almost-surely. Applying the Strong Law of Large Numbers to $\{V_\theta(\mathbf{x}') \mid \mathbf{x}' \in \mathcal{D}_\mathbf{x}\}$ we conclude that, almost-surely,

$$\begin{aligned}
&\lim_{M\to\infty} \max\left\{ \sum_{\mathbf{x}'\in\mathcal{D}_\mathbf{x}} \frac{V_\theta(\mathbf{x}')}{|\mathcal{D}_\mathbf{x}|} - V_\theta(\mathbf{x}) + \tau \cdot K, 0 \right\} \\
&= \max\left\{ \lim_{M\to\infty} \sum_{\mathbf{x}'\in\mathcal{D}_\mathbf{x}} \frac{V_\theta(\mathbf{x}')}{|\mathcal{D}_\mathbf{x}|} - V_\theta(\mathbf{x}) + \tau \cdot K, 0 \right\} \\
&= \max\left\{ \mathbb{E}_{\omega\sim d}\Big[V\Big(f(\mathbf{x}, \pi(\mathbf{x}), \omega) \Big) \Big] - V_\theta(\mathbf{x}) + \tau \cdot K, 0 \right\} \\
&= 0,
\end{aligned}$$

where the first equality holds since limits may be interchanged with the maximum function over finitely many arguments, the second equality holds almost-surely by the Strong Law of Large Numbers, and the third inequality holds since V_θ satisfies eq. (3) for each $\mathbf{x} \in \tilde{\mathcal{X}}$. Hence, $\mathcal{L}_{\mathsf{RSM}}(\theta) = 0$, as claimed. □

CHAPTER 7

Learning-based Stochastic Control with Quantitative Reach-avoidance

This section is based on the following publication:

- Đorđe Žikelić*, Mathias Lechner*, Thomas A. Henzinger, Krishnendu Chatterjee. *Learning Control Policies for Stochastic Systems with Reach-avoid Guaranteess*. In Thirty-Seventh AAAI Conference on Artificial Intelligence, **AAAI 2023**

7.1 Introduction

Stochastic control with quantitative reach-avoid guarantees. This chapter considers formal controller synthesis in discrete-time stochastic dynamical systems with quantitative reach-avoid guarantees. In particular, we revisit our learner-verifier framework for stochastic dynamical systems introduced in Chapter 6 and significantly extend and generalize it to enable learning and verifying a neural network control policy and a certificate function which formally verifies that the reach-avoid specification is satisfied with at least the desired probability. This yields the first framework for learning neural controllers in stochastic dynamical systems with quantitative reach-avoid guarantees. Our framework also yields a method for formally verifying quantitative reach-avoidance under a neural network control policy in stochastic dynamical systems.

Reach-avoid specifications. Reach-avoid specifications are one of the most common and practically relevant specifications appearing in safety-critical applications that

*Equal contribution.

generalize both reachability and safety specifications [SL10]. Given a target region and an unsafe region, the reach-avoid specification requires that a system controlled by a policy reaches the target region while avoiding the unsafe region. For instance, a lane-keeping constraint requires a self-driving car to reach its destination without leaving the allowed car lanes [VE03]. In the case of stochastic systems, reach-avoid specifications are also specified by a probability threshold with which the system controlled by a policy needs to satisfy the reach-avoid specification.

Prior approaches and learning-based control. As discussed in Chapter 6, classical automated methods for formal controller synthesis are mostly restricted to dynamical systems with polynomial dynamics or to finite-time horizon systems. Learning-based methods provide a promising approach to overcoming these limitations. However, the key challenge is to enable formal verification of neural network control policies in stochastic dynamical systems. Prior to the work in this thesis, several works have contributed to setting the foundations of learning-based control in *deterministic* dynamical system with formal reachability and stability [RBK18, CRG19, AAGP21] and safety [PAA21] guarantees. In Chapter 6, we presented to the best of our knowledge the first learning-based control method with formal almost-sure reachability guarantees for infinite-time horizon systems that learns and verifies neural controllers and is applicable to general continuous stochastic dynamical systems over compact state spaces (recall, every continuous function defined over a compact state space is also Lipschitz continuous).

Our approach – learning and verification of neural reach-avoid supermartingales. We extend and generalize the method in Chapter 6 to enable formal controller synthesis with quantitative reach-avoid guarantees. Our method jointly learns a control policy and a reach-avoid supermartingale, both parametrized as neural networks. A *reach-avoid supermartingale (RASM)* is a martingale-based certificate function for formally certifying quantitative reach-avoidance that we introduce in this work. Informally, an RASM is a function assigning a nonnegative real value to each system state that is required to strictly decrease in expected value until the target region is reached, but needs to strictly increase for the system to reach the unsafe region. By carefully choosing the ratio of the initial level set of the RASM and the least level set that the RASM needs to attain for the system to reach the unsafe region (here we use the standard level set terminology of Lyapunov functions [HC11]), we obtain a formal reach-avoid certificate. Our RASMs generalize and unify the stochastic control barrier functions which are a standard certificate for safe control of stochastic dynamical systems [PJP04, PJP07] and ranking supermartingales that certify probability 1 reachability [CS13]. Our method in this chapter has two key novel aspects compared to the learner-verifier framework for almost-sure reachability in Chapter 6. First, we use introduce and use RASMs as certificate functions. Second, we show how the learner and the verifier modules can be

generalized to allow effective learning and formal verification of RASMs.

Verification procedure for Lipschitz continuous policies. Analogously as in Chapter 6, our framework also yields a method for formally verifying quantitative reach-avoidance under a given control policy, by simply fixing the control policy in the learner-verifier loop and only learning and verifying the RASM. Our method only assumes that the given control policy is Lipschitz continuous. This allows for neural network control policies with all standard activation functions since they are known to be Lipschitz continuous [SZS+14].

Contributions. Our contributions in this chapter can be summarized as follows:

1. We introduce *reach-avoid supermartingales (RASMs)*, a novel martingale-based certificate function for formally certifying quantitative reach-avoidance. Our RASMs unify and generalize stochastic barrier functions [PJP04, PJP07] and ranking supermartingales [CS13].

2. We present a learner-verifier framework for jointly learning a control policy and an RASM which formally proves quantitative reach-avoidance, both parametrized as neural networks. Our method is applicable to stochastic dynamical systems with compact state space and (Lipschitz) continuous dynamics function. As a special cases, our method can be used for formal controller synthesis with quantitative reachability or with quantitative safety guarantees as well.

3. By fixing a control policy and only learning and verifying the RASM, our framework also yields a method for formal verification of quantitative reach-avoidance under a given Liphscitz continuous control policy.

4. We empirically validate our approach and demonstrate its effectiveness on 3 reinforcement learning benchmarks.

Chapter organization. The rest of this chapter is organized as follows. We formally define the problem considered in this chapter in Section 7.2. In Section 7.3, we introduce RASMs and prove their soundness for formally certifying quantitative reach-avoidance in stochastic dynamical systems. In Section 7.4, we present a learner-verifier framework for stochastic dynamical systems that jointly learns a control policy and an RASM as neural networks. In Section 7.5 we present our experimental results. We discuss related work in Section 7.6. Finally, Section 7.7 contains full proofs of results presented in earlier sections that are deferred to this section in order to enhance readability.

7.2 Problem Statement

Analogously as in Chapter 6, we consider a discrete-time stochastic dynamical system

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t, \omega_t),\ t \in \mathbb{N}_0,$$

defined by a dynamics function $f : \mathcal{X} \times \mathcal{U} \times \mathcal{N} \to \mathcal{X}$ as in Section 2.3. We use d to denote the probability distribution over the stochastic disturbance space $\mathcal{N}$ from which the stochastic disturbance vector ω_t is sampled at each time step.

Problem statement. We consider the *quantitative reach-avoidance analysis* problem in stochastic dynamical systems. Let $\mathcal{X}_0 \subseteq \mathcal{X}$ denote a Borel-measurable initial set of states. Let $\mathcal{X}_t \subseteq \mathcal{X}$ and $\mathcal{X}_u \subseteq \mathcal{X}$ be Borel-measurable *target set* and *unsafe set*, respectively, and let $p \in [0, 1]$. We define

$$\mathsf{ReachAvoid}(\mathcal{X}_t, \mathcal{X}_u) = \left\{ (\mathbf{x}_t, \mathbf{u}_t, \omega_t)_{t \in \mathbb{N}_0} \mid \exists t \in \mathbb{N}_0.\, \mathbf{x}_t \in \mathcal{X}_t \wedge (\forall t' \leq t.\, \mathbf{x}_{t'} \notin \mathcal{X}_u) \right\}$$

to be the set of all trajectories that reach $\mathcal{X}_t$ while avoiding $\mathcal{X}_u$. Our goal is to synthesize a control policy π such that, for every initial state $\mathbf{x}_0 \in \mathcal{X}_0$, we have

$$\mathbb{P}^{\pi}_{\mathbf{x}_0}\Big[\mathsf{ReachAvoid}(\mathcal{X}_t, \mathcal{X}_u)\Big] \geq p.$$

We restrict to the cases when either $p < 1$, or $p = 1$ and $\mathcal{X}_u = \emptyset$. Our approach is not applicable to the case $p = 1$ and $\mathcal{X}_u \neq \emptyset$ due to technical issues that arise in defining our formal certificate, which we discuss in the following section. We remark that quantitative reachability is a special instance of our problem obtained by setting $\mathcal{X}_u = \emptyset$. On the other hand, we cannot directly obtain quantitative safety by assuming any specific form of the target set $\mathcal{X}_t$, however we will show in the following section that our method implies quantitative safety with respect to $\mathcal{X}_u$ if we set $\mathcal{X}_t = \emptyset$.

Assumptions. As in Section 2.3, we assume that $\mathcal{X}$, $\mathcal{U}$ and $\mathcal{N}$ as well as $\mathcal{X}_0, \mathcal{X}_t, \mathcal{X}_u \subseteq \mathcal{X}$ are all Borel-measurable for the system semantics to be well-defined. We also assume that $\mathcal{X} \subseteq \mathbb{R}^m$ is compact (i.e. closed and bounded) in the Euclidean topology of $\mathbb{R}^m$. The dynamics function f is assumed to be continuous, which is a common assumption in control theory. Note that every continuous function defined over a compact state space is also Lipschitz continuous. Finally, we assume that d has bounded support or that it is a product of independent univariate probability distributions, which is needed for efficient sampling and expected value computation.

7.3 Theoretical Results

We now introduce *reach-avoid supermartingales (RASMs)*, our novel martingale-based certificate function for formally certifying quantitative reach-avoidance in stochastic dynamical systems. Note that, in this section only, we assume that the policy is fixed. In the next section, we will present our algorithm for learning policies that provide formal reach-avoid guarantees in which RASMs will be an integral ingredient. In what follows, we consider a discrete-time stochastic dynamical system defined as in the previous section. For now, we assume that the probability threshold is strictly smaller than 1, i.e. $p < 1$. We will later show that our approach straightforwardly extends to the case $p = 1$ and $\mathcal{X}_u = \emptyset$.

Reach-avoid supermartingales. We define a *reach-avoid supermartingale (RASM)* to be a continuous function $V : \mathcal{X} \to \mathbb{R}$ that assigns real values to system states. The name is chosen to emphasize the connection to supermartingale processes from probability theory [Wil91], which we will explore later in order to prove the effectiveness of RASMs for verifying reach-avoid properties. The value of V is required to be nonnegative over the state space $\mathcal{X}$ (Nonnegativity condition), to be bounded from above by 1 over the set of initial states $\mathcal{X}_0$ (Initial condition) and to be bounded from below by $\frac{1}{1-p}$ over the set of unsafe states $\mathcal{X}_u$ (Safety condition). Hence, in order for a system trajectory to reach an unsafe state and violate the safety specification, the value of the RASM V needs to increase at least $\frac{1}{1-p}$ times along the trajectory. Finally, we require the existence of $\varepsilon > 0$ such that the value of V decreases in expected value by at least ε after every one-step evolution of the system from every system state $\mathbf{x} \in \mathcal{X} \backslash \mathcal{X}_t$ for which $V(\mathbf{x}) \leq \frac{1}{1-p}$ (Expected decrease condition). Intuitively, this last condition imposes that the system has a tendency to strictly *decrease* the value of V until either the target set $\mathcal{X}_t$ is reached or a state with $V(\mathbf{x}) \geq \frac{1}{1-p}$ is reached. However, as the value of V needs to *increase* at least $\frac{1}{1-p}$ times in order for the system to reach an unsafe state, these four conditions will allow us to use RASMs to certify that the reach-avoid constraint is satisfied with probability at least p.

Defintion 7.3.1 (Reach-avoid supermartingales)**.** *Let $\mathcal{X}_t \subseteq \mathcal{X}$ and $\mathcal{X}_u \subseteq \mathcal{X}$ be the target set and the unsafe set, and let $p \in [0, 1)$ be the probability threshold. A continuous function $V : \mathcal{X} \to \mathbb{R}$ is said to be a* reach-avoid supermartingale (RASM) *with respect to $\mathcal{X}_t$, $\mathcal{X}_u$ and p if it satisfies:*

1. Nonnegativity condition. $V(\mathbf{x}) \geq 0$ *for each* $\mathbf{x} \in \mathcal{X}$.
2. Initial condition. $V(\mathbf{x}) \leq 1$ *for each* $\mathbf{x} \in \mathcal{X}_0$.
3. Safety condition. $V(\mathbf{x}) \geq \frac{1}{1-p}$ *for each* $\mathbf{x} \in \mathcal{X}_u$.

4. Expected decrease condition. *There exists* $\varepsilon > 0$ *such that, for each* $\mathbf{x} \in \mathcal{X} \setminus \mathcal{X}_t$ *at which* $V(\mathbf{x}) \leq \frac{1}{1-p}$, *we have* $V(\mathbf{x}) \geq \mathbb{E}_{\omega \sim d}[V(f(\mathbf{x}, \pi(\mathbf{x}), \omega))] + \varepsilon$.

Comparison to Lyapunov functions. The defining properties of RASMs hint a connection to Lyapunov functions for deterministic control systems. However, the key difference between Lyapunov functions and our RASMs is that Lyapunov functions deterministically decrease in value whereas RASMs decrease in expectation. Deterministic decrease ensures that each level set of a Lyapunov function, i.e. a set of states at which the value of Lyapunov functions is at most l for some $l \geq 0$, is an invariant of the system. However, it is in general not possible to impose such a condition on stochastic systems. In contrast, our RASMs only require expected decrease in the level, and the Initial and the Unsafe conditions can be viewed as conditions on the *maximal initial level set* and the *minimal unsafe level set*. The choice of a ratio of these two level values allows us to use existing results from martingale theory in order to obtain probabilistic avoidance guarantees, while the Expected decrease condition by $\varepsilon > 0$ furthermore provides us with probabilistic reachability guarantees.

Certifying reach-avoid constraints via RASMs. We now show that the existence of an ε-RASM for some $\varepsilon > 0$ implies that the reach-avoid constraint is satisfied with probability at least p.

Theorem 7.3.2. *Let* $\mathcal{X}_t \subseteq \mathcal{X}$ *and* $\mathcal{X}_u \subseteq \mathcal{X}$ *be the target set and the unsafe set, respectively, and let* $p \in [0, 1)$ *be the probability threshold. Suppose that there exists an RASM* V *with respect to* $\mathcal{X}_t$, $\mathcal{X}_u$ *and* p. *Then, for every* $\mathbf{x}_0 \in \mathcal{X}_0$, $\mathbb{P}_{\mathbf{x}_0}[\mathsf{ReachAvoid}(\mathcal{X}_t, \mathcal{X}_u)] \geq p$.

Proof sketch. The complete proof of Theorem 7.3.2 is provided in Section 7.7.1. In what follows, we sketch the key ideas behind our proof, in order to illustrate how quantitative reachability and safety analysis can be combined by using martingale-based certificate functions. To prove the theorem, we first show that an ε-RASM V induces a *supermartingale* [Wil91] in the probability space over the set of all trajectories that start in an initial state $\mathbf{x}_0 \in \mathcal{X}_0$. Recall, a supermartingale in a probability space $(\Omega, \mathcal{F}, \mathbb{P})$ is a stochastic process $(X_t)_{t=0}^{\infty}$ such that, for each $t \in \mathbb{N}_0$, the expected value of X_{t+1} conditioned on the value of X_t is less than or equal to X_t. We refer the reader to Section 2.4 for a necessary background on probability and martingale theory that is needed to understand the proof of Theorem 7.3.2.

Now, let $(\Omega_{\mathbf{x}_0}, \mathcal{F}_{\mathbf{x}_0}, \mathbb{P}_{\mathbf{x}_0})$ be the probability space of trajectories that start in $\mathbf{x}_0$. Then,

for each time step $t \in \mathbb{N}_0$, we define a random variable

$$X_t(\rho) = \begin{cases} V(\mathbf{x}_t), & \text{if } \mathbf{x}_i \notin \mathcal{X}_t \text{ and } V(\mathbf{x}_i) < \frac{1}{1-p} \\ & \text{for each } 0 \leq i \leq t \\ 0, & \text{if } \mathbf{x}_i \in \mathcal{X}_t \text{ for some } 0 \leq i \leq t \\ & \text{and } V(\mathbf{x}_j) < \frac{1}{1-p} \text{ for each } 0 \leq j \leq i \\ \frac{1}{1-p}, & \text{otherwise} \end{cases}$$

for each trajectory $\rho = (\mathbf{x}_t, \mathbf{u}_t, \omega_t)_{t \in \mathbb{N}_0} \in \Omega_{\mathbf{x}_0}$. In other words, the value of X_t is equal to the value of V at $\mathbf{x}_t$, unless either the target set $\mathcal{X}_t$ has been reached first in which case we set all future values of $\mathcal{X}_t$ to 0, or a state in which V exceeds $\frac{1}{1-p}$ has been reached first in which case we set all future values of $\mathcal{X}_t$ to $\frac{1}{1-p}$. Then, since V satisfies the Nonnegativity and the Expected decrease condition of RASMs, we may show that $(X_t)_{t=0}^{\infty}$ is a supermartingale. in the probability space $(\Omega_{\mathbf{x}_0}, \mathcal{F}_{\mathbf{x}_0}, \mathbb{P}_{\mathbf{x}_0})$.

Next, we show that the nonnegative supermartingale $(X_t)_{t=0}^{\infty}$ with probability 1 converges to and reaches 0 or a value that is greater than or equal to $\frac{1}{1-p}$. To do this, we first employ the Supermartingale Convergence Theorem (see Section 7.7.1 for formal details) which states that every nonnegative supermartingale converges to some value with probability 1. We then use the fact that, in the Expected decrease condition of RASMs, the decrease in expected value is strict and by at least $\varepsilon > 0$, in order to conclude that this value is reached and has to be either 0 or greater than or equal to $\frac{1}{1-p}$.

Finally, we use another classical result from martingale theory (see Section 7.7.1 for formal details) which states that, given a nonnegative supermartingale $(X_t)_{t=0}^{\infty}$ and $\lambda > 0$,

$$\mathbb{P}\Big[\sup_{i \geq 0} X_i \geq \lambda\Big] \leq \frac{\mathbb{E}[X_0]}{\lambda}.$$

Plugging $\lambda = \frac{1}{1-p}$ into the above inequality, it follows that $\mathbb{P}_{\mathbf{x}_0}[\sup_{i \geq 0} X_i \geq \frac{1}{1-p}] \leq (1-p) \cdot \mathbb{E}_{\mathbf{x}_0}[X_0] \leq 1-p$. The second inequality follows since $X_0(\rho) = V(\mathbf{x}_0) \leq 1$ for every $\rho \in \Omega_{\mathbf{x}_0}$ by the Initial condition of RASMs. Hence, as $(X_t)_{t=0}^{\infty}$ with probability 1 either reaches 0 or a value that is greater than or equal to $\frac{1}{1-p}$, we conclude that $(X_t)_{t=0}^{\infty}$ reaches 0 without reaching a value that is greater than or equal to $\frac{1}{1-p}$ with probability at least p. By the definition of each X_t and by the Safety condition of RASMs, this implies that with probability at least p the system will reach the target set $\mathcal{X}_t$ without reaching the unsafe set $\mathcal{X}_u$, i.e. that $\mathbb{P}_{\mathbf{x}_0}[\text{ReachAvoid}(\mathcal{X}_t, \mathcal{X}_u)] \geq p$. □

Quantitative safety. In order to solve the quantitative safety analysis problem and verify that a control policy guarantees that the unsafe set $\mathcal{X}_u$ is not reached with probability at least p, we may modify the Expected decrease condition of RASMs by

setting $\mathcal{X}_t = \emptyset$. Thus, RASMs are also effective for the quantitative safety analysis. This claim follows immediately from our proof of Theorem 7.3.2. In this case and if we set $\varepsilon = 0$, then our RASMs coincide with stochastic barrier functions of [PJP04, PJP07]. However, if $\mathcal{X}_t$ is not empty, then we must have $\varepsilon > 0$ in order to enforce convergence and reachability of $\mathcal{X}_t$.

Extension to $p = 1$ and $\mathcal{X}_u = \emptyset$ and comparison to RSMs. So far, we have only considered $p \in [0, 1)$. The difficulty in the case $p = 1$ arises since the value $\frac{1}{1-p}$ in the Safety and the Expected decrease conditions in Definition 7.3.1 would not be well-defined. However, if $\mathcal{X}_u = \emptyset$, then the Safety condition need not be imposed at any state. Moreover, it follows directly from our proof that imposing the expected decrease condition at all states in $\mathcal{X} \setminus \mathcal{X}_t$ makes RASMs sound for certifying probability 1 reachability. In fact, in this special case our RASMs reduce to the RSMs of [CS13] and used in [LZCH22]. The key novelty of our RASMs over RSMs is that we also employ *level set reasoning* in order to obtain probabilistic reach-avoid guarantees, thus presenting a true *stochastic extension of Lyapunov functions* that allow reasoning both about reach-avoid specifications as well as quantitative reasoning about the probability with which they are satisfied. In contrast, RSMs do not reason about level sets and can only certify probability 1 reachability.

7.4 Learner-verifier Framework with RASMs

We now present our learner-verifier framework for learning control policies with reach-avoid guarantees, by jointly learning a neural network policy and an RASM certificate. The algorithm consists of two modules called *learner* and *verifier*, which are composed into a loop. In each loop iteration, the learner learns a policy together with an RASM candidate as two neural networks π_θ and V_ν, with θ and ν being vectors of neural network parameters. The verifier then formally verifies whether the learned RASM candidate is indeed an RASM for the system and the learned policy. If the answer is positive, then the algorithm concludes that the learned policy provides formal reach-avoid guarantees. Otherwise, the verifier computes a counterexample which shows that the learned RASM candidate is not an RASM. The counterexample is passed to the learner and used to modify the loss function towards learning a new policy and an RASM candidate. The loop is repeated until either a candidate is successfully verified or the algorithm reaches a specified timeout. The algorithm is presented in Algorithm 7.1. We note that our algorithm can also *verify* whether a given Lipschitz continuous policy guarantees quantitative reach-avoidance, by fixing the policy and learning the RASM.

Algorithm 7.1: Algorithm for learning quantitative reach-avoidance policies

input : Dynamics function f, disturbance distribution d, state space $\mathcal{X}$, initial, target and unsafe sets $\mathcal{X}_0, \mathcal{X}_t, \mathcal{X}_u \subseteq \mathcal{X}$, probability $p \in [0, 1)$, Lipschitz constant L_f, parameters $\tau > 0$, $N \in \mathbb{N}$, $\lambda > 0$

output : "Reach-avoid guarantee with probability p" or "Unknown"

$\pi_\theta \leftarrow$ trained by PPO
$\tilde{\mathcal{X}} \leftarrow$ discretization of $\mathcal{X}$ with mesh τ
$C_{\text{init}}, C_{\text{unsafe}}, C_{\text{decrease}} \leftarrow \tilde{\mathcal{X}} \cap \mathcal{X}_0, \tilde{\mathcal{X}} \cap \mathcal{X}_u, \tilde{\mathcal{X}} \cap (\mathcal{X} \backslash \mathcal{X}_t)$
$V_\nu \leftarrow$ trained by minimizing the loss function
while *timeout not reached* **do**
  $L_\pi, L_V \leftarrow$ Lipschitz constants of π_θ, V_ν
  $K \leftarrow L_V \cdot (L_f \cdot (L_\pi + 1) + 1)$
  $\tilde{\mathcal{X}}_e \leftarrow$ vertices of discr. $\tilde{\mathcal{X}}$ whose adjacent cells intersect $\mathcal{X} \backslash \mathcal{X}_t$ and contain $\mathbf{x}$ s.t. $V_\nu(\mathbf{x}) < \frac{1}{1-p}$
  $\text{Cells}_{\mathcal{X}_0}, \text{Cells}_{\mathcal{X}_u} \leftarrow$ discr. cells that intersect $\mathcal{X}_0$, $\mathcal{X}_u$
  if $\exists \tilde{\mathbf{x}} \in \tilde{\mathcal{X}}_e \cap (\mathcal{X} \backslash \mathcal{X}_t)$ *s.t.* $\mathbb{E}_{\omega \sim d}[V_\nu(f(\tilde{\mathbf{x}}, \pi(\tilde{\mathbf{x}}), \omega))] \geq V_\nu(\tilde{\mathbf{x}}) - \tau \cdot K$ *and* $V_\nu(\tilde{\mathbf{x}}) < \frac{1}{1-p}$ **then**
    $C_{\text{decrease}} \leftarrow C_{\text{decrease}} \cup \{\mathbf{x}\}$
  end
  else if $\exists$*cell* $\in$ *Cells*$_{\mathcal{X}_0}$ *s.t.* $\sup_{\mathbf{x} \in \textit{cell}} V_\nu(\mathbf{x}) > 1$ **then**
    $C_{\text{init}} \leftarrow C_{\text{init}} \cup (\{\text{vertices of cell}\} \cap \mathcal{X}_0)$
  end
  else if $\exists$*cell* $\in$ *Cells*$_{\mathcal{X}_u}$ *s.t.* $\inf_{\mathbf{x} \in \textit{cell}} V_\nu(\mathbf{x}) < \frac{1}{1-p}$ **then**
    $C_{\text{unsafe}} \leftarrow C_{\text{unsafe}} \cup (\{\text{vertices of cell}\} \cap \mathcal{X}_u)$
  end
  else
    return *"Reach-avoid guarantee with probability p"*
  end
  $V_\nu, \pi_\theta, \leftarrow$ trained by minimizing the loss function
  $\tilde{\mathcal{X}} \leftarrow$ refined discretization
end
return *"Unknown"*

7.4.1 Initialization and Discretization

Policy Initialization. Learning two networks concurrently with multiple objectives can be unstable due to dependencies between the two networks and differences in the scale of the objective loss terms. To mitigate these instabilities, we propose pre-training of the policy network so that our algorithm starts from a proper initialization. In

particular, from the given dynamical system and the safety specification, we induce a Markov decision process (MDP) intending to reach the target set while avoiding the unsafe set. The reward term r_t is given by $r_t := \mathbb{I}[\mathcal{X}_t](\mathbf{x}_t) - \mathbb{I}[\mathcal{X}_u](\mathbf{x}_t)$ and we use proximal policy optimization (PPO) [SWD+17] to train the policy.

State Space Discretization. When it comes to verifying learned candidates, the key difficulty lies in checking the Expected decrease condition. This is because, in general, it is not possible to compute a closed form expression for the expected value of an RASM over successor system states, as both the policy and the RASM are neural networks. In order to overcome this difficulty, our algorithm discretizes the state space of the system. Given a *mesh* parameter $\tau > 0$, a *discretization* $\tilde{\mathcal{X}}$ of $\mathcal{X}$ with mesh τ is a set of states such that, for every $\mathbf{x} \in \mathcal{X}$, there exists a state $\tilde{\mathbf{x}} \in \tilde{\mathcal{X}}$ such that $||\mathbf{x} - \tilde{\mathbf{x}}||_1 < \tau$. Due to $\mathcal{X}$ being compact and therefore bounded, for any $\tau > 0$ it is possible to compute its finite discretization with mesh τ by simply considering vertices of a grid with sufficiently small cells. Note that f, π_θ and V_ν are all continuous, hence due to $\mathcal{X}$ being compact f, π_θ and V_ν are also Lipschitz continuous. This will allow us to verify that the Expected decrease condition is satisfied by checking a slightly stricter condition only at the vertices of the discretization grid. The initial discretization $\tilde{\mathcal{X}}$ is also used to initialize counterexample sets used by the learner. In particular, the learner initializes three sets $C_{\mathsf{init}} = \tilde{\mathcal{X}} \cap \mathcal{X}_0$, $C_{\mathsf{unsafe}} = \tilde{\mathcal{X}} \cap \mathcal{X}_u$ and $C_{\mathsf{decrease}} = \tilde{\mathcal{X}} \cap (\mathcal{X} \backslash \mathcal{X}_t)$. These sets will later be extended by counterexamples computed by the verifier. Conversely, the discretization used by the verifier for checking the defining properties of RASMs will at each iteration of the loop be refined by a discretization with a smaller mesh, in order to relax the conditions that are checked by the verifier.

7.4.2 Verifier

We now describe the verifier module of our algorithm. Suppose that the learner has learned a policy π_θ and an RASM candidate V_ν. Since V_ν is a neural network, we know that it is a continuous function. Furthermore, we design the learner to apply a softplus activation function to the output layer of V_ν, which ensures that the Nonnegativity condition of RASMs is satisfied by default. Thus, the verifier only needs to check the Initial, Safety and Expected decrease conditions in Definition 7.3.1.

Let L_f, L_π and L_V be the Lipschitz constants of f, π_θ and V_ν, respectively. We assume that a Lipschitz constant for the dynamics function f is provided, and use the method of [SZS+14] to compute Lipschitz constants of neural networks π_θ and V_ν. To verify the Expected decrease condition, the verifier collects the superset $\tilde{\mathcal{X}}_e$ of discretization points whose adjacent grid cells contain a non-target state and over which V_ν attains a value that is smaller than $\frac{1}{1-p}$. This set is computed by first collecting all cells that intersect $\mathcal{X} \backslash \mathcal{X}_t$, then using interval arithmetic abstract interpretation (IA-AI) [CC77, GDS+18]

which propagates interval bounds across neural network layers in order to bound from below the minimal value that V_ν attains over each collected cell, and finally collecting vertices of all cells at which this lower bound is less than $\frac{1}{1-p}$. The verifier then checks a stricter condition for each state $\tilde{\mathbf{x}} \in \tilde{\mathcal{X}}_e$:

$$\mathbb{E}_{\omega \sim d}\Big[V_\nu\Big(f(\tilde{\mathbf{x}}, \pi_\theta(\tilde{\mathbf{x}}), \omega)\Big)\Big] < V_\nu(\tilde{\mathbf{x}}) - \tau \cdot K, \tag{7.1}$$

where $K = L_V \cdot (L_f \cdot (L_\pi + 1) + 1)$. The expected value in eq. (7.1) is also bounded from above via IA-AI by using the method that we presented in Chapter 6, where one partitions the support of d into intervals, propagates intervals and multiplies each interval bound by its probability weight in order to bound the expected value of a neural network function over a probability distribution.

In order to verify the Initial condition, the verifier collects the set $\text{Cells}_{\mathcal{X}_0}$ of all cells of the discretization grid that intersect the initial set $\mathcal{X}_0$. Then, for each cell $\in \text{Cells}_{\mathcal{X}_0}$, it checks whether

$$\sup_{\mathbf{x} \in \text{cell}} V_\nu(\mathbf{x}) > 1, \tag{7.2}$$

where the supremum of V_ν over the cell is bounded from above by using IA-AI. Similarly, to verify the Unsafe condition, the verifier collects the set $\text{Cells}_{\mathcal{X}_u}$ of all cells of the discretization grid that intersect the unsafe set $\mathcal{X}_u$. Then, for each cell $\in \text{Cells}_{\mathcal{X}_u}$, it uses IA-AI to check whether

$$\inf_{\mathbf{x} \in \text{cell}} V_\nu(\mathbf{x}) < \frac{1}{1-p}. \tag{7.3}$$

If the verifier shows that V_ν satisfies eq. (7.1) for each $\tilde{\mathbf{x}} \in \tilde{\mathcal{X}}_e$, eq. (7.2) for each cell $\in \text{Cells}_{\mathcal{X}_0}$ and eq. (7.3) for each cell $\in \text{Cells}_{\mathcal{X}_u}$, it concludes that V_ν is an RASM. Otherwise, if a counterexample $\tilde{\mathbf{x}}$ to eq. (7.1) is found and we have $\tilde{\mathbf{x}} \in \mathcal{X} \backslash \mathcal{X}_t$ and $V_\nu(\mathbf{x}) < \frac{1}{1-p}$, it is added to C_{decrease}. Similarly, if counterexample cells to eq. (7.2) and eq. (7.3) are found, all their vertices that are contained in $\mathcal{X}_0$ and $\mathcal{X}_u$ are added to C_{init} and C_{unsafe}, respectively.

The following theorem shows that checking the above conditions is sufficient to formally verify whether an RASM candidate is indeed an RASM. The proof follows by exploiting the fact that f, π_θ and V_ν are all Lipschitz continuous and that $\mathcal{X}$ is compact, and we include it in Section 7.7.2.

Theorem 7.4.1. *Suppose that the verifier verifies that V_ν satisfies eq. (7.1) for each $\tilde{\mathbf{x}} \in \tilde{\mathcal{X}}_e$, eq. (7.2) for each cell $\in \text{Cells}_{\mathcal{X}_0}$ and eq. (7.3) for each cell $\in \text{Cells}_{\mathcal{X}_u}$. Then the function V_ν is an RASM for the system with respect to $\mathcal{X}_t$, $\mathcal{X}_u$ and p.*

7.4.3 Learner

A policy and an RASM candidate are learned by minimizing the loss function

$$\begin{aligned}\mathcal{L}(\theta,\nu) =& \mathcal{L}_{\mathsf{Init}}(\nu) + \mathcal{L}_{\mathsf{Unsafe}}(\nu) + \mathcal{L}_{\mathsf{Decrease}}(\theta,\nu) \\ &+ \lambda \cdot \Big(\mathcal{L}_{\mathrm{Lipschitz}}(\theta) + \mathcal{L}_{\mathrm{Lipschitz}}(\nu)\Big).\end{aligned}$$

The first three loss terms are used to guide the learner towards learning a true RASM by forcing the learned candidate towards satisfying the Initial, Safety and Expected decrease conditions in Definition 7.3.1. They are defined as follows:

$$\begin{aligned}&\mathcal{L}_{\mathsf{Init}}(\nu) = \max_{\mathbf{x}\in C_{\mathsf{init}}} \{V_\nu(\mathbf{x}) - 1, 0\} \\ &\mathcal{L}_{\mathsf{Unsafe}}(\nu) = \max_{\mathbf{x}\in C_{\mathsf{unsafe}}} \{\frac{1}{1-p} - V_\nu(\mathbf{x}), 0\} \\ &\mathcal{L}_{\mathsf{Decrease}}(\theta,\nu) = \frac{1}{|C_{\mathsf{decrease}}|}\cdot \\ &\sum_{\mathbf{x}\in C_{\mathsf{decrease}}} \left(\max\left\{\sum_{\omega_1,\ldots,\omega_N\sim\mathcal{N}} \frac{V_\nu\Big(f(\mathbf{x},\pi_\theta(\mathbf{x}),\omega_i)\Big)}{N} - V_\theta(\mathbf{x}) + \tau\cdot K, 0\right\}\right)\end{aligned}$$

Each loss term is designed to incur a loss at a state whenever that state violates the corresponding condition in Definition 7.3.1 that needs to be checked by the verifier. In the expression for $\mathcal{L}_{\mathsf{Decrease}}(\theta,\nu)$, we approximate the expected value of V_ν by taking the mean value of V_ν at N sampled successor states, where $N \in \mathbb{N}$ is an algorithm parameter. This is necessary as it is not possible to compute a closed form expression for the expected value of a neural network V_ν.

The last loss term $\lambda \cdot (\mathcal{L}_{\mathrm{Lipschitz}}(\theta) + \mathcal{L}_{\mathrm{Lipschitz}}(\nu))$ is the regularization term used to guide the learner towards a policy and an RASM candidate with Lipschitz constants below a tolerable threshold ρ, with $\lambda > 0$ being a regularization constant. By preferring networks with small Lipschitz constants, we allow the verifier to use a wider mesh, which significantly speeds up the verification process. The regularization term for π_θ (and analogously for V_ν) is defined via

$$\mathcal{L}_{\mathsf{Lipschitz}}(\theta) = \max\Big\{\prod_{W,b\in\theta} \max_j \sum_i |W_{i,j}| - \rho, 0\Big\},$$

where W and b weight matrices and bias vectors for each layer in π_θ so $L_{V_\theta} = \prod_{W,b\in\theta} \max_j \sum_i |W_{i,j}|$ is a Lipschitz constant for V_θ.

Finally, in our implementation we also add an auxiliary loss term $\mathcal{L}_{\mathsf{Aux}}(\nu)$ that does not enforce any of the defining conditions of RASMs, however it is used to guide the learner towards a candidate that attains the global minimum in a state that is contained within

Environment	RSM (reach-avoid extension)	RASM (ours)
2D system	83.4%	93.3%
Inverted pendulum	47.9%	92.1%
Collision avoidance	Fail	90.4%

Table 7.1: Reach-avoid probability obtained by our method and by the naive extension of RSMs. In each case, we report the largest probability successfully verified by the method.

the target set $\mathcal{X}_t$. We empirically observed that this term sometimes helps the updated policy from diverging from its objective to stabilize the system. It is defined via

$$\begin{aligned}\mathcal{L}_{\mathsf{Aux}}(\nu) = &\max\{V_\nu(\hat{\mathbf{x}}_{\mathsf{Target}}) - \varepsilon, 0\} \\ &+ \max\{\min_{\mathbf{x}\in\tilde{X}\cap\mathcal{X}_t} V_\nu(\mathbf{x}) - \min_{\mathbf{x}\in C_{\mathsf{init}}} V_\nu(\mathbf{x}), 0\} \\ &+ \max\{\min_{\mathbf{x}\in\tilde{X}\cap\mathcal{X}_t} V_\nu(\mathbf{x}) - \min_{\mathbf{x}\in C_{\mathsf{unsafe}}} V_\nu(\mathbf{x}), 0\}\end{aligned}$$

with $\tilde{\mathbf{x}}_{\mathsf{Target}}$ being some state contained in $\mathcal{X}_t$ and $\varepsilon \geq 0$ an algorithm parameter.

We remark that the loss function is always nonnegative but is not necessarily equal to 0 even if V_ν satisfies all conditions checked by the verifier and if Lipschitz constants are below the specified thresholds. This is because the expected values in $\mathcal{L}_{\mathsf{Decrease}}(\theta, \nu)$ are approximated via sample means. However, in the following theorem we show that in this case $\mathcal{L}(\theta, \nu) \to 0$ with probability 1 as we add independent samples. The claim follows from the Strong Law of Large Numbers and the proof can be found in Section 7.7.3.

Theorem 7.4.2. *Let N be the number of samples used to approximate expected values in $\mathcal{L}_{Decrease}(\theta, \nu)$. Suppose that V_ν satisfies eq. (7.1) for each $\tilde{\mathbf{x}} \in \tilde{\mathcal{X}}_e$, eq. (7.2) for each cell $\in$ Cells$_{\mathcal{X}_0}$ and eq. (7.3) for each cell $\in$ Cells$_{\mathcal{X}_u}$. Suppose that Lipschitz constants of π_θ and V_ν are below the thresholds specified by $\mathcal{L}_{Lipschitz}(\theta)$ and $\mathcal{L}_{Lipschitz}(\nu)$ and that the samples in $\mathcal{L}_{Decrease}(\theta, \nu)$ are independent. Then $\lim_{N\to\infty} \mathcal{L}(\theta, \nu) = 0$ with probability 1.*

7.5 Experiments

We experimentally validate our method on 3 non-linear RL environments. Our first two environments are a linear 2D system with non-linear control bounds and the stochastic inverted pendulum control problem. The linear 2D system is of the form $\mathbf{x}_{t+1} = A\mathbf{x}_t + Bg(\mathbf{u}_t) + \omega_t$, where $g : u \mapsto \min(\max(u, -1), 1))$ limits the admissible

action of the policy and ω_t is sampled from a triangular noise distribution. The inverted pendulum environment is taken from the OpenAI Gym [BCP+16] and made more difficult by adding noise perturbations to its state. Our third environment concerns a collision avoidance task. The objective of this environment is to navigate an agent to the target region while avoiding crashing into one of two obstacles. Further details on all environments can be found in [ZLHC22, Appendix].

The policy and RASM networks consist of two hidden layers (128 units each, ReLU). The RASM network has a single output unit with a softplus activation. We run our algorithm with a timeout of 3 hours.

The goal of our first experiment is to empirically evaluate the ability of our approach to learn policies for quantitative reach-avoidance and to understand the importance of combining reachability with level set reasoning towards safety in stochastic systems. For all tasks, we pre-train the policy networks using 100 iterations of PPO. To evaluate our approach, we run our algorithm with several probability thresholds and report the highest threshold for which a policy together with an RASM is successfully learned. In order to understand the importance of simultaneous reasoning about reachability and level sets, we then compare our approach with a much simpler extension of the method of [LZCH22] presented in Chapter 6, which learns RSMs to certify probability 1 reachability but does not consider any form of safety specifications. In particular, we run the method in Chapter 6 without the safety constraint and, in case a valid RSM is found, we normalize the function such that the Nonnegativity and the Initial conditions of RASMs are satisfied. We then bound from below the smallest value that the RSM attains over the unsafe region, and extract the corresponding reach-avoid probability bound according to the Safety condition of RASMs. Note that, even though this extension also exploits the ideas behind the level set reasoning in our RASMs, it *first* performs reachability analysis and only *afterwards* considers safety. We remark that there is no existing method that provides reach-avoid guarantees in non-polynomial stochastic systems over the infinite time horizon, i.e. there is no existing baseline to compare against, thus we compare our level set reasoning with the extension of the method in Chapter 6.

Table 7.1 shows results of our first experiment. In particular, in the third column we see that our method successfully learns policies that provide high probability reach-avoid guarantees for all benchmarks. On the other hand, comparison to the second column shows that *simultaneous* reasoning about reachability and safety that is allowed by our RASMs provides significantly better quantitative reach-avoid guarantees than when such reasoning is decoupled. Figure 7.1 visualizes the RSM computed by the baseline and our RASM.

In our second experiment, we study how well our algorithm can *repair* (or *fine-tune*) an unsafe policy. In particular, we pre-train the policy network using only 20 PPO

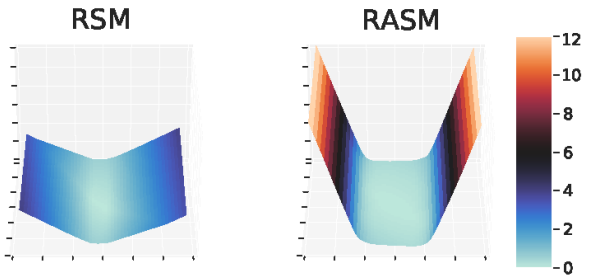

Figure 7.1: Visualization of a neural network RSM and RASM on the inverted pendulum task. The RASM provides better probability bounds of reaching the unsafe states.

	V_ν	V_ν and π_θ
2D system	Fail (10 iters.)	96.7% (4 iters.)
Collision avoidance	Fail (9 iters.)	80.9% (3 iters.)
Inverted pendulum	Fail (7 iters.)	Fail (7 iters.)

Table 7.2: Reach-avoid probabilities obtained by repairing unsafe policies. Verifying a policy by only learning the RASM V_ν times out, while jointly optimizing V_ν and π_θ yields a valid RASM. In each case, we report the largest reach-avoid probability successfully verified by the respective method.

iterations. We then run our algorithm with fixed policy parameters θ, i.e. we only learn an RASM in order to verify quantitative reach-avoid guarantee provided by a pre-trained policy. Next, we run our Algorithm 7.1 with both ν and θ as trainable parameters. Table 7.2 shows that, compared to a standalone verification method, our algorithm is able to repair unsafe policies in practice. However, the inability to repair the inverted pendulum policy illustrates that a decent starting policy is necessary for our algorithm, emphasizing the importance of policy initialization. Since the Policy Initialization step in Algorithm 7.1 initializes the policy by using PPO with a reward function that encodes the reach-avoid specification, our second experiment also demonstrates that a policy initialised by using RL on a tailored reward function is not sufficient to learn a reach-avoid policy with guarantees and that the learned policy requires "correction" in order to provide reach-avoid guarantees. The "correction" is achieved precisely by keeping the policy parameters trainable in the learner-verifier framework and fine-tuning them.

7.6 Related Work

To conclude this chapter, we overview related work on reachability and safety analysis in dynamical systems. For an overview of classical approaches to formal controller

synthesis with reachability and safety guarantees in deterministic dynamical systems, we refer the reader to Section 6.6 where these works were already discussed. We also refer the reader to Section 6.6 for discussions on safe exploration RL and martingale-based methods for PP analysis in order to avoid repetition. In what follows, we discuss quantitative reachability and safety analysis in stochastic dynamical systems.

Formal controller synthesis for stochastic dynamical systems. Formal controller synthesis for stochastic dynamical systems has received less attention compared to their deterministic counterparts. Most existing approaches are abstraction based – they construct a finite-state MDP or stochastic game approximation of the problem and then use probabilistic model checking for controller synthesis. Due to accumulation of the approximation error in each time step, many abstraction-based methods are applicable to systems that evolve over finite and a priori known time horizons. Notable examples include [SGA15, LKSZ20, CA19, VGO19]. Recently, a few abstraction-based controller synthesis methods for infinite-time horizon stochastic dynamical systems with affine dynamics [HS21], control affine dynamics [DHC22] or with finite control input spaces [MMSS24, DHC22] have been proposed. An abstraction based method for obtaining infinite-time horizon PAC-style guarantees on reach-avoidance in linear stochastic systems was proposed in [BRAJ22]. This method is applicable to systems with both aleatoric and epistemic uncertainty. Another approach that was discussed above is to consider stochastic dynamical systems with polynomial dynamics and utilize stochastic control barrier functions and convex optimization tools to compute polynomial control policies with quantitative safety guarantees [PJP04, PJP07, ST12, SDC21, MMB+22]. The method of [XLZF21] considers polynomial stochastic dynamical systems and uses convex optimization to synthesize polynomial control policies with quantitative reach-avoid guarantees. Concurrently to our work, [MCL23] proposed a learning-based method for formal safety verification in continuous stochastic control systems over a fixed *finite-time horizon*, by learning a neural network stochastic control barrier function. Other methods use dynamic programming [APLS08] or Hamilton-Jacobi (HJ) reachability analysis [BCHT17] for finite time horizon stochastic systems.

Constrained MDPs. Safe RL has also been studied in the context of constrained MDPs (cMDPs) [Alt99, Gei06]. An agent in a cMDP must satisfy hard constraints on expected cost for one or more auxiliary notions of cost aggregated over an episode. Several works study RL algorithms for cMDPs [UD07], notably the Constrained Policy Optimization (CPO) [AHTA17] or the method [CNDG18] which proposed a Lyapunov method for solving cMDPs. While these algorithms perform well, their constraints are satisfied in expectation which makes them less suitable for safety-critical systems. Furthermore, these methods try to satisfy constraints empirically and do not guarantee constraint satisfaction. On the other hand, these methods operate in the model-free

setting and do not assume knowledge of system dynamics.

Safe RL via shielding. Some safe RL approaches ensure safety by computing two control policies – the main policy that optimizes the expected reward, and the backup policy that the system falls back to whenever a safety constraint may be violated [MM93, PB02, ABE+18, EBA+21, GHKW21]. The backup policy can thus be of simpler form. Shielding for stochastic linear systems with additive disturbances has been considered in [WZ18]. [LB20, BL21] are applicable to stochastic non-linear systems, however their safety guarantees are *statistical* – their algorithms are randomized with parameters $\delta, \varepsilon \in (0,1)$ and they with probability $1-\delta$ compute an action that is safe in the current state with probability at least $1-\varepsilon$. The statistical error is accumulated at each state, hence these approaches are not suitable for infinite or long time horizons. In contrast, our approach targets *formal* guarantees for *infinite* time horizon problems.

7.7 Technical Proofs

7.7.1 Proof of Theorem 7.3.2

Theorem. *Let $\mathcal{X}_t \subseteq \mathcal{X}$ and $\mathcal{X}_u \subseteq \mathcal{X}$ be the target set and the unsafe set, respectively, and let $p \in [0,1)$ be the probability threshold. Suppose that there exists an RASM V with respect to $\mathcal{X}_t$, $\mathcal{X}_u$ and p. Then, for every $\mathbf{x}_0 \in \mathcal{X}_0$, $\mathbb{P}_{\mathbf{x}_0}[\mathsf{ReachAvoid}(\mathcal{X}_t, \mathcal{X}_u)] \geq p$.*

The proof assumes the background on probability and martingale theory presented in Sections 2.1 and 2.4. Before we prove the theorem, we recall two additional results from martingale theory that will be key ingredients in the proof. The first is Doob's Supermartingale Convergence Theorem (see [Wil91], Section 11) which shows that every nonnegative supermartingale converges almost-surely to some finite value. The second theorem (see [Kus14], Theorem 7.1) provides a bound on the probability that the value of the supemartingale ever exceeds some threshold, and it will allow us to reason about both probabilistic reachability and safety. This is a less standard result from martingale theory, so we prove it below. In what follows, let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space and $(\mathcal{F}_i)_{i=0}^{\infty}$ be a filtration in it.

Theorem 7.7.1 (Supermartingale convergence theorem)**.** *Let $(X_i)_{i=0}^{\infty}$ be a nonnegative supermartingale with respect to $(\mathcal{F}_i)_{i=0}^{\infty}$. Then, there exists a random variable X_∞ in $(\Omega, \mathcal{F}, \mathbb{P})$ to which the supermartingale converges to with probability 1, i.e. $\mathbb{P}[\lim_{i\to\infty} X_i = X_\infty] = 1$.*

Theorem 7.7.2. *Let $(X_i)_{i=0}^{\infty}$ be a nonnegative supermartingale with respect to $(\mathcal{F}_i)_{i=0}^{\infty}$. Then, for every $\lambda > 0$, we have*

$$\mathbb{P}\Big[\sup_{i\geq 0} X_i \geq \lambda\Big] \leq \frac{\mathbb{E}[X_0]}{\lambda}.$$

Proof. Fix $\lambda > 0$. Define a stopping time $T : \Omega \to \mathbb{N}_0 \cup \{\infty\}$ via $T = \inf_{i\in\mathbb{N}_0}\{X_i \geq \lambda\}$. Then, for each $n \in \mathbb{N}_0$, define a random variable

$$X_{T\wedge n} = X_T \cdot \mathbb{I}(T \leq n) + X_n \cdot \mathbb{I}(T > n)$$

where X_T is a random variable defined via $X_T(\omega) = X_{T(\omega)}(\omega)$ for each $\omega \in \Omega$, and $\mathbb{I}$ is the indicator function. It is a classical result from martingale theory that, for any $n \in \mathbb{N}_0$, we have $\mathbb{E}[X_{T\wedge n}] \leq \mathbb{E}[X_0]$ (see [Wil91], Section 10.9). Hence, in order to prove the desired inequality, it suffices to prove $\lambda \cdot \mathbb{P}[\sup_{i\geq 0} X_i \geq \lambda] \leq \sup_{n\in\mathbb{N}_0} \mathbb{E}[X_{T\wedge n}]$.

To prove the desired inequality we observe that, for each $n \in \mathbb{N}_0$, we have that

$$\begin{aligned}
\mathbb{E}[X_{T\wedge n}] &= \mathbb{E}[X_T \cdot \mathbb{I}(T \leq n)] + \mathbb{E}[X_n \cdot \mathbb{I}(T > n)] \\
&\geq \mathbb{E}[\lambda \cdot \mathbb{I}(T \leq n)] + \mathbb{E}[X_n \cdot \mathbb{I}(T > n)] \\
&= \lambda \cdot \mathbb{P}[T \leq n] + \mathbb{E}[X_n \cdot \mathbb{I}(T > n)] \\
&\geq \lambda \cdot \mathbb{P}[T \leq n] = \lambda \cdot \mathbb{P}[\sup_{0\leq i\leq n} X_i \geq \lambda].
\end{aligned} \tag{7.4}$$

where in the first inequality we use the fact that $X_T \geq \lambda$, in the second inequality we use the fact that each X_n is nonnegative and in the last equality we use the fact that $T = \inf_{i\in\mathbb{N}_0}\{X_i \geq \lambda\}$. Finally, $(\mathbb{P}[\sup_{0\leq i\leq n} X_i \geq \lambda])_{n=0}^{\infty}$ is a sequence of probabilities of events that are increasing with respect to set inclusion, so by the Monotone Convergence Theorem (see [Wil91], Section 5.3) it follows that

$$\lim_{n\to\infty} \mathbb{P}[\sup_{1\leq i\leq n} X_i \geq \lambda] = \mathbb{P}[\sup_{i\in\mathbb{N}_0} X_i \geq \lambda].$$

Hence, by taking the supremum over $n \in \mathbb{N}_0$ of both sides of eq. (7.4), we conclude that $\lambda \cdot \mathbb{P}[\sup_{i\geq 0} X_i \geq \lambda] \leq \sup_{n\in\mathbb{N}_0} \mathbb{E}[X_{T\wedge n}]$, as desired. This concludes the proof of Theorem 7.7.2 since

$$\mathbb{P}\Big[\sup_{i\geq 0} X_i \geq \lambda\Big] \leq \sup_{n\in\mathbb{N}_0} \frac{\mathbb{E}[X_{T\wedge n}]}{\lambda} \leq \frac{\mathbb{E}[X_0]}{\lambda}.$$

□

Proof of Theorem 7.3.2. Fix an initial state $\mathbf{x}_0 \in \mathcal{X}_0$ so that we need to show that $\mathbb{P}_{\mathbf{x}_0}[\mathsf{ReachAvoid}(\mathcal{X}_t, \mathcal{X}_u)] \geq p$. Let V be an RASM with respect to $\mathcal{X}_t$, $\mathcal{X}_u$ and $p \in [0, 1)$

whose existence is assumed in the theorem. First, we show that V gives rise to a supermartingale in the probability space $(\Omega_{\mathbf{x}_0}, \mathcal{F}_{\mathbf{x}_0}, \mathbb{P}_{\mathbf{x}_0})$ of all trajectories of the system that start in $\mathbf{x}_0$. Then, we use Theorem 7.7.1 and Theorem 7.7.2 to prove probabilistic reachability and safety.

For each time step $t \in \mathbb{N}_0$, define $\mathcal{F}_{\mathbf{x}_0,t} \subseteq \mathcal{F}_{\mathbf{x}_0}$ to be a sub-σ-algebra that, intuitively, contains events that are defined in terms of the first t states of the system. Formally, for each $j \in \mathbb{N}_0$, let $C_j : \Omega_{\mathbf{x}_0} \to \mathcal{X}$ assign to each trajectory $\rho = (\mathbf{x}_t, \mathbf{u}_t, \omega_t)_{t\in\mathbb{N}_0} \in \Omega_{\mathbf{x}_0}$ the j-th state $\mathbf{x}_j$ along the trajectory. We define $\mathcal{F}_i$ to be the smallest σ-algebra over $\Omega_{\mathbf{x}_0}$ with respect to which $C_0, C_1, \ldots, C_i$ are all measurable, where $\mathcal{X} \subseteq \mathbb{R}^m$ is equipped with the induced subset Borel-σ-algebra. The sequence $(\mathcal{F}_{\mathbf{x}_0,t})_{t=0}^{\infty}$ is increasing with respect to set inclusion.

Now, define a stochastic process $(X_t)_{t=0}^{\infty}$ in the probability space $(\Omega_{\mathbf{x}_0}, \mathcal{F}_{\mathbf{x}_0}, \mathbb{P}_{\mathbf{x}_0})$ via

$$X_t(\rho) = \begin{cases} V(\mathbf{x}_t), & \text{if } \mathbf{x}_i \notin \mathcal{X}_t \text{ and } V(\mathbf{x}_i) < \frac{1}{1-p} \\ & \text{for each } 0 \leq i \leq t \\ 0, & \text{if } \mathbf{x}_i \in \mathcal{X}_t \text{ for some } 0 \leq i \leq t \text{ and} \\ & V(\mathbf{x}_j) < \frac{1}{1-p} \text{ for each } 0 \leq j \leq i \\ \frac{1}{1-p}, & \text{otherwise} \end{cases}$$

for each $t \in \mathbb{N}_0$ and a trajectory $\rho = (\mathbf{x}_t, \mathbf{u}_t, \omega_t)_{t\in\mathbb{N}_0}$. In other words, the value of X_t is equal to the value of V at $\mathbf{x}_t$, unless either the target set $\mathcal{X}_t$ has been reached first in which case we set all future values of $\mathcal{X}_t$ to 0, or a state in which V exceeds $\frac{1}{1-p}$ has been reached first in which case we set all future values of $\mathcal{X}_t$ to $\frac{1}{1-p}$. We claim that $(X_t)_{t=0}^{\infty}$ is a nonnegative supermartingale with respect to $(\mathcal{F}_{\mathbf{x}_0,t})_{t=0}^{\infty}$. Indeed, each X_t is $\mathcal{F}_{\mathbf{x}_0,t}$-measurable as it is defined in terms of the first t states along a trajectory. It is also nonnegative as V is nonnegative by the Nonnegativity condition of RASMs. Finally, to see that $\mathbb{E}_{\mathbf{x}_0}[X_{t+1} \mid \mathcal{F}_{\mathbf{x}_0,t}](\rho) \leq X_t(\rho)$ holds for each $t \in \mathbb{N}_0$ and $\rho = (\mathbf{x}_t, \mathbf{u}_t, \omega_t)_{t\in\mathbb{N}_0}$, we consider 3 cases:

1. If $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_t \notin \mathcal{X}_t$ and $V(\mathbf{x}_i) < \frac{1}{1-p}$ for each $0 \leq i \leq t$, then

$$\begin{aligned}
&\mathbb{E}_{\mathbf{x}_0}[X_{t+1} \mid \mathcal{F}_{\mathbf{x}_0,t}](\rho) \\
&= \mathbb{E}_{\mathbf{x}_0}\Big[X_{t+1} \cdot \Big(\mathbb{I}(\mathbf{x}_{t+1} \notin \mathcal{X}_t \wedge V(\mathbf{x}_{t+1}) < \frac{1}{1-p}) \\
&\quad + \mathbb{I}(\mathbf{x}_{t+1} \in \mathcal{X}_t) + \mathbb{I}(V(\mathbf{x}_{t+1}) \geq \frac{1}{1-p})\Big) \mid \mathcal{F}_{\mathbf{x}_0,t}\Big](\rho) \\
&= \mathbb{E}_{\mathbf{x}_0}[X_{t+1} \cdot \mathbb{I}(\mathbf{x}_{t+1} \notin \mathcal{X}_t) \mid \mathcal{F}_{\mathbf{x}_0,t}](\rho) \\
&\quad + 0 + \frac{1}{1-p} \cdot \mathbb{E}[\mathbb{I}(V(\mathbf{x}_{t+1}) \geq \frac{1}{1-p}) \mid \mathcal{F}_{\mathbf{x}_0,t}](\rho) \\
&\leq \mathbb{E}_{\omega \sim d}[V(f(\mathbf{x}_t, \mathbf{u}_t, \omega_t)) \cdot \mathbb{I}(\mathbf{x}_{t+1} \notin \mathcal{X}_t \wedge V(\mathbf{x}_{t+1}) < \frac{1}{1-p})] \\
&\quad + \mathbb{E}_{\omega \sim d}[V(f(\mathbf{x}_t, \mathbf{u}_t, \omega_t)) \cdot \mathbb{I}(\mathbf{x}_{t+1} \in \mathcal{X}_t)] \\
&\quad + \mathbb{E}_{\omega \sim d}[V(f(\mathbf{x}_t, \mathbf{u}_t, \omega_t)) \cdot \mathbb{I}(V(\mathbf{x}_{t+1}) \geq \frac{1}{1-p})] \\
&= \mathbb{E}_{\omega \sim d}[V(f(\mathbf{x}_t, \mathbf{u}_t, \omega_t))] \leq V(\mathbf{x}_t) - \varepsilon.
\end{aligned}$$

   Here, the first equality follows by the law of total probability, the second equality follows by our definition of each X_t, the third inequality follows by observing that $V(\mathbf{x}_{t+1}) \geq X_{t+1}(\rho)$ in this case, the fourth equality is just the sum of expectations over disjoint sets, and finally the fifth inequality follows by the Expected decrease condition in Definition 1 since $\mathbf{x}_t \notin \mathcal{X}_t$ and $V(\mathbf{x}_t) < \frac{1}{1-p}$, by the assumption of this case.

2. If $\mathbf{x}_i \in \mathcal{X}_t$ for some $0 \leq i \leq t$ and $V(\mathbf{x}_j) < \frac{1}{1-p}$ for all $0 \leq j \leq i$, then we have $\mathbb{E}_{\mathbf{x}_0}[X_{t+1} \mid \mathcal{F}_{\mathbf{x}_0,t}](\rho) = X_{t+1}(\rho) = 0$.

3. Otherwise, we must have $V(\mathbf{x}_i) \geq \frac{1}{1-p}$ and $\mathbf{x}_0, \ldots, \mathbf{x}_i \notin \mathcal{X}_t$ for some $- \leq i \leq t$, thus $\mathbb{E}_{\mathbf{x}_0}[X_{t+1} \mid \mathcal{F}_{\mathbf{x}_0,t}](\rho) = X_{t+1}(\rho) = \frac{1}{1-p}$.

Hence, we have proved that $(X_t)_{t=0}^{\infty}$ is a nonnegative supermartingale.

Now, by Theorem 7.7.1 it follows that the value of the nonnegative supermartingale $(X_t)_{t=0}^{\infty}$ with probability 1 converges. In what follows, we show that $(X_t)_{t=0}^{\infty}$ with probability 1 converges to and reaches either 0 or a value that is greater than or equal to $\frac{1}{1-p}$. To do this, we use the fact that the Expected decrease condition of RASMs enforces the value of V to decrease in expected value by at least $\varepsilon > 0$ after every one-step evolution of the system in any non-target state at which $V(\mathbf{x}) < \frac{1}{1-p}$. Define the stopping time $T : \Omega_{\mathbf{x}_0} \to \mathbb{N}_0 \cup \{\infty\}$ via

$$T(\rho) = \inf_{t \in \mathbb{N}_0} \Big\{ X_t(\rho) = 0 \vee X_t(\rho) \geq \frac{1}{1-p} \Big\}.$$

Our goal is then to prove that $\mathbb{P}_{\mathbf{x}_0}[T < \infty] = 1$. Using the argument in the proof that $(X_t)_{t=0}^{\infty}$ is a nonnegative supermartingale (in particular, the proof of supermartingale property in Case 1), we can in fact deduce a stronger inequality

$$\mathbb{E}_{\mathbf{x}_0}[X_{t+1} \mid \mathcal{F}_{\mathbf{x}_0,t}](\rho) \leq X_t(\rho) - \varepsilon \cdot \mathbb{I}(T(\rho) > t)$$

for each $\rho \in \Omega_{\mathbf{x}_0}$. But now, we may use Proposition 7.7.3 stated below to deduce that $\mathbb{E}_{\mathbf{x}_0}[T] \leq \mathbf{E}_{\mathbf{x}_0}[X_0] = V(\mathbf{x}_0) < \infty$, which in turn implies that $\mathbb{P}_{\mathbf{x}_0}[T < \infty] = 1$, as desired. This concludes the proof.

The following proposition states a results on probability 1 convergence of ranking supermartingales (RSMs). We note that RASMs generalize RSMs in the sense that RSMs coincide with RASMs in the special case when the unsafe set is empty and we only consider a probability 1 reachability specification, i.e. $\mathcal{X}_u = \emptyset$.

Proposition 7.7.3 ([CS13]). *Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space, let $(\mathcal{F}_i)_{i=0}^{\infty}$ be an increasing sequence of sub-σ-algebras in $\mathcal{F}$ and let T be a stopping time with respect to $(\mathcal{F}_i)_{i=0}^{\infty}$. Suppose that $(X_i)_{i=0}^{\infty}$ is a stochastic process such that each X_i is nonnegative and we have that*

$$\mathbb{E}[X_{i+1} \mid \mathcal{F}_i](\omega) \leq X_i(\omega) - \varepsilon \cdot \mathbb{I}(T(\omega) > i)$$

holds for each $i \in \mathbb{N}_0$ and $\omega \in \Omega$. Then $\mathbb{P}[T < \infty] = 1$.

Finally, by using Theorem 7.7.2 for the nonnegative supermartingale $(X_t)_{t=0}^{\infty}$ and $\lambda = \frac{1}{1-p} > 0$, it follows that $\mathbb{P}_{\mathbf{x}_0}[\sup_{i \geq 0} X_i \geq \frac{1}{1-p}] \leq (1-p) \cdot \mathbb{E}_{\mathbf{x}_0}[X_0] \leq 1 - p$. The second inequality follows since $X_0(\rho) = V(\mathbf{x}_0) \leq 1$ for every $\rho \in \Omega_{\mathbf{x}_0}$ by the Initial condition of RASMs. Hence, as $(X_t)_{t=0}^{\infty}$ with probability 1 either reaches 0 or a value that is greater than or equal to $\frac{1}{1-p}$, we conclude that $(X_t)_{t=0}^{\infty}$ reaches 0 without reaching a value that is greater than or equal to $\frac{1}{1-p}$ with probability at least p. By the definition of each X_t and by the Safety condition of RASMs, this implies that with probability at least p the system will reach the target set $\mathcal{X}_t$without reaching the unsafe set $\mathcal{X}_u$, i.e. that $\mathbb{P}_{\mathbf{x}_0}[\text{ReachAvoid}(\mathcal{X}_t, \mathcal{X}_u)] \geq p$. □

7.7.2 Proof of Theorem 7.4.1

Theorem. *Suppose that the verifier verifies that V_ν satisfies eq. (7.1) for each $\tilde{\mathbf{x}} \in \tilde{\mathcal{X}}_e$, eq. (7.2) for each cell $\in$ Cells$_{\mathcal{X}_0}$ and eq. (7.3) for each cell $\in$ Cells$_{\mathcal{X}_u}$. Then the function V_ν is an RASM for the system with respect to $\mathcal{X}_t$, $\mathcal{X}_u$ and p.*

Proof. Suppose that the verifier verifies that V satisfies eq. (7.1) for each $\tilde{\mathbf{x}} \in \tilde{\mathcal{X}}_e$, eq. (7.2) for each cell $\in$ Cells$_{\mathcal{X}_0}$ and eq. (7.3) for each cell $\in$ Cells$_{\mathcal{X}_u}$. The fact that the Initial and the Unsafe conditions in Definition 1 of RASMs are satisfied

by V then follows from the correctness of interval arithmetic abstract interpretation (IA-AI) of [GDS+18]. Thus, we only need to show that V satisfies the Expected decrease condition. To show this, we need to show that there exists $\varepsilon > 0$ such that $V(\mathbf{x}) \geq \mathbb{E}_{\omega\sim d}[V(f(\mathbf{x}, \pi(\mathbf{x}), \omega))] + \varepsilon$ holds for all $\mathbf{x} \in \mathcal{X}\backslash\mathcal{X}_t$ at which $V(\mathbf{x}) \leq \frac{1}{1-p}$. We prove that $\varepsilon > 0$ defined via

$$\varepsilon = \min_{\tilde{\mathbf{x}}\in\tilde{\mathcal{X}}_\varepsilon} \Big(V(\tilde{\mathbf{x}}) - \tau \cdot K - \mathbb{E}_{\omega\sim d}\Big[V\Big(f(\tilde{\mathbf{x}}, \pi(\tilde{\mathbf{x}}), \omega)\Big)\Big]\Big)$$

satisfies this property. Note that $\varepsilon > 0$, as each $\tilde{\mathbf{x}} \in \tilde{\mathcal{X}}_e$ satisfies eq. (7.1).

To show this, fix $\mathbf{x} \in \mathcal{X}\backslash\mathcal{X}_t$ with $V(\mathbf{x}) \leq \frac{1}{1-p}$ and let $\tilde{\mathbf{x}} \in \tilde{\mathcal{X}}_e$ be such that $||\mathbf{x}-\tilde{\mathbf{x}}||_1 \leq \tau$. By construction, the set $\tilde{\mathcal{X}}_e$ contains vertices of each discretization cell that intersects $\mathcal{X}\backslash\mathcal{X}_t$ and that contains at least one state at which V is less than or equal to $\frac{1}{1-p}$, hence such $\tilde{\mathbf{x}}$ exists. We then have

$$\begin{aligned}
&\mathbb{E}_{\omega\sim d}\Big[V\Big(f(\mathbf{x}, \pi(\mathbf{x}), \omega)\Big)\Big] \\
&\leq \mathbb{E}_{\omega\sim d}\Big[V\Big(f(\tilde{\mathbf{x}}, \pi(\tilde{\mathbf{x}}), \omega)\Big)\Big] + ||f(\tilde{\mathbf{x}}, \pi(\tilde{\mathbf{x}}), \omega) - f(\mathbf{x}, \pi(\mathbf{x}), \omega)||_1 \cdot L_V \\
&\leq \mathbb{E}_{\omega\sim d}\Big[V\Big(f(\tilde{\mathbf{x}}, \pi(\tilde{\mathbf{x}}), \omega)\Big)\Big] + ||(\tilde{\mathbf{x}}, \pi(\tilde{\mathbf{x}}), \omega) - (\mathbf{x}, \pi(\mathbf{x}), \omega)||_1 \cdot L_V \cdot L_f \\
&\leq \mathbb{E}_{\omega\sim d}\Big[V\Big(f(\tilde{\mathbf{x}}, \pi(\tilde{\mathbf{x}}), \omega)\Big)\Big] + ||\tilde{\mathbf{x}} - \mathbf{x}||_1 \cdot L_V \cdot L_f \cdot (1 + L_\pi) \\
&\leq \mathbb{E}_{\omega\sim d}\Big[V\Big(f(\tilde{\mathbf{x}}, \pi(\tilde{\mathbf{x}}), \omega)\Big)\Big] + \tau \cdot L_V \cdot L_f \cdot (1 + L_\pi).
\end{aligned} \tag{7.5}$$

On the other hand, we also have

$$V(\mathbf{x}) \geq V(\tilde{\mathbf{x}}) - ||\tilde{\mathbf{x}} - \mathbf{x}||_1 \cdot L_V \geq V(\tilde{\mathbf{x}}) - \tau \cdot L_V. \tag{7.6}$$

Combining eq.(7.5) and (7.6), we conclude that

$$\begin{aligned}
&V(\mathbf{x}) - \mathbb{E}_{\omega\sim d}\Big[V\Big(f(\mathbf{x}, \pi(\mathbf{x}), \omega)\Big)\Big] \\
&\geq V(\tilde{\mathbf{x}}) - \tau \cdot L_V - \mathbb{E}_{\omega\sim d}\Big[V\Big(f(\tilde{\mathbf{x}}, \pi(\tilde{\mathbf{x}}), \omega)\Big)\Big] - \tau \cdot L_V \cdot L_f \cdot (1 + L_\pi) \\
&= V(\tilde{\mathbf{x}}) - \tau \cdot K - \mathbb{E}_{\omega\sim d}\Big[V\Big(f(\tilde{\mathbf{x}}, \pi(\tilde{\mathbf{x}}), \omega)\Big)\Big] \\
&\geq \varepsilon
\end{aligned} \tag{7.7}$$

where the equality in the second last row follows by the definition of K, and the inequality in the last row follows by our choice of ε. Hence, V satisfies the Expected decrease condition and is indeed an RASM. □

7.7.3 Proof of Theorem 7.4.2

Theorem. *Let N be the number of samples used to approximate expected values in $\mathcal{L}_{Decrease}(\theta, \nu)$. Suppose that V_ν satisfies eq. (7.1) for each $\tilde{\mathbf{x}} \in \tilde{\mathcal{X}}_e$, eq. (7.2) for each cell $\in$ Cells$_{\mathcal{X}_0}$ and eq. (7.3) for each cell $\in$ Cells$_{\mathcal{X}_u}$. Suppose that Lipschitz constants of π_θ and V_ν are below the thresholds specified by $\mathcal{L}_{Lipschitz}(\theta)$ and $\mathcal{L}_{Lipschitz}(\nu)$ and that the samples in $\mathcal{L}_{Decrease}(\theta, \nu)$ are independent. Then $\lim_{N\to\infty} \mathcal{L}(\theta, \nu) = 0$ with probability 1.*

Proof. Since Lipschitz constants of π_θ and V_ν are below the thresholds specified by $\mathcal{L}_{\text{Lipschitz}}(\theta)$ and $\mathcal{L}_{\text{Lipschitz}}(\nu)$, we have that $\lambda \cdot (\mathcal{L}_{\text{Lipschitz}}(\theta) + \mathcal{L}_{\text{Lipschitz}}(\nu)) = 0$. Moreover, our initialization of C_{init} and our design of the verifier module ensure that C_{init} contains only states in $\mathcal{X}_0$, hence $\mathcal{L}_{\text{Init}}(\nu) = 0$ as V_ν satisfies the Initial condition of RASMs. Note, V_ν satisfies all conditions checked by the verifier, hence by Theorem 2 we know that it is an RASM. Similarly, C_{unsafe} contains only states in $\mathcal{X}_u$, hence $\mathcal{L}_{\text{Unsafe}}(\nu) = 0$ as V_ν satisfies the Safety condition of RASMs. Thus, under theorem assumptions we have that

$$\begin{aligned}
&\mathcal{L}(\theta, \nu) = \mathcal{L}_{\text{Decrease}}(\theta, \nu) \\
&= \frac{1}{|C_{\text{decrease}}|} \sum_{\mathbf{x} \in C_{\text{decrease}}} \left(\max \left\{ \sum_{\omega_1, \ldots, \omega_N \sim \mathcal{N}} \frac{V_\nu\Big(f(\mathbf{x}, \pi_\theta(\mathbf{x}), \omega_i)\Big)}{N} - V_\theta(\mathbf{x}) + \tau \cdot K, 0 \right\} \right)
\end{aligned}$$

Hence, in order to prove that $\lim_{N\to\infty} \mathcal{L}(\theta, \nu) = 0$ with probability 1, it suffices to prove that for each $\mathbf{x} \in C_{\text{decrease}}$ with probability 1 we have

$$\lim_{N\to\infty} \max \left\{ \sum_{\omega_1, \ldots, \omega_N \sim \mathcal{N}} \frac{V_\nu\Big(f(\mathbf{x}, \pi_\theta(\mathbf{x}), \omega_i)\Big)}{N} - V_\theta(\mathbf{x}) + \tau \cdot K, 0 \right\} = 0.$$

The above sum is the mean of N independently sampled successor states of $\mathbf{x}$, which are sampled according to the probability distribution defined by the system dynamics and the probability distribution d over disturbance vectors. Since the state space of the system is assumed to be compact and V_θ is continuous as it is a neural network, the random value defined by the value of V at a sampled successor state is bounded and therefore admits a well-defined and finite first moment. The Strong Law of Large Numbers [Wil91] then implies that the above sum converges to the expected value of

this distribution as $N \to \infty$. Thus, with probability 1, we have that

$$\begin{aligned}
&\lim_{N\to\infty} \max\Big\{ \sum_{\omega_1,\ldots,\omega_N\sim\mathcal{N}} \frac{V_\nu\Big(f(\mathbf{x},\pi_\theta(\mathbf{x}),\omega_i)\Big)}{N} - V_\theta(\mathbf{x}) + \tau\cdot K, 0\Big\} \\
&= \max\Big\{ \lim_{M\to\infty} \sum_{\omega_1,\ldots,\omega_N\sim\mathcal{N}} \frac{V_\nu\Big(f(\mathbf{x},\pi_\theta(\mathbf{x}),\omega_i)\Big)}{N} - V_\theta(\mathbf{x}) + \tau\cdot K, 0\Big\} \\
&= \max\Big\{ \lim_{M\to\infty} \mathbb{E}_{\omega\sim d}\Big[f(\mathbf{x},\pi_\theta(\mathbf{x}),\omega)\Big] - V_\theta(\mathbf{x}) + \tau\cdot K, 0\Big\} \\
&= 0.
\end{aligned}$$

The first equality holds since a limit may be interchanged with the maximum function over a finite number of arguments, the second equality holds with probability 1 by the Strong Law of Large Numbers, and the third equality holds since V_ν satisfies eq. (7.1) for each $\tilde{\mathbf{x}} \in \tilde{\mathcal{X}}_e$ and we have $C_{\text{decrease}} \subseteq \tilde{\mathcal{X}}_e$. This concludes our proof that $\lim_{N\to\infty} \mathcal{L}(\theta,\nu) = 0$ with probability 1. □

CHAPTER 8

Discussion and Conclusion

8.1 Discussion

"All things can be conjoined."
*Miriel, Elden Ring**

This thesis has predominantly focused on the formal verification and controller synthesis problems for infinite state stochastic systems. However, it also introduces or advances several theoretical concepts and algorithmic methods – novel martingale-based certificates for proving properties of stochastic systems, constraint solving-based methods for formal synthesis of polynomial certificates in probabilistic and non-probabilistic programs, as well as the learner-verifier framework for formal synthesis of non-polynomial controllers and certificates in dynamical systems. As it turns out, the applicability of these techniques stretches beyond probabilistic program analysis and controller synthesis in stochastic dynamical systems. In this section, we overview some of the other research projects that the author of this thesis has worked on during the PhD period, in which these techniques have found useful and sometimes unexpected applications. These works *do not constitute* a part of this thesis, as they consider problems not directly related to infinite state stochastic system analysis.

8.1.1 Differential Cost Analysis

This section is based on the following publication:

*A wonderful game published by FromSoftware Inc. and Bandai Namco Entertainment.

- Đorđe Žikelić*, Bor-Yuh Evan Chang, Pauline Bolignano, Franco Raimondi. *Differential Cost Analysis with Simultaneous Potentials and Anti-potentials*. In 43rd ACM SIGPLAN International Conference on Programming Language Design and Implementation, **PLDI 2022**

Differential cost analysis [ÇBG+17] considers the problem of statically bounding the *difference* in resource usage (i.e. *cost*) between two program versions. In particular, for two program versions and a set of inputs, the goal of differential cost analysis is to compute a *threshold* value that bounds the maximal difference in cost usage between the two programs.The notion of cost may be quite generic, and it encompasses metrics such as runtime, memory usage, the number of object allocations or the number of thread allocations, etc. This static analysis problem has many practical applications. For instance, in software development, programs are often modified and extended with new features. A program revision might lead to unacceptable jumps in cost usage. For performance critical software, it is crucial to detect such undesired performance regressions prior to releasing the software to production.

In this work, we propose a new method for differential cost analysis in numerical imperative programs with polynomial arithmetic and with non-determinism. Our method uses potential functions from amortized analysis [Tar85] to reason about costs incurred in individual programs. Potential functions are a well-known certificate function for computing upper bounds on the cost incurred in a single program [HAH11, HH10, HDW17, CHS15]. In this work we also use their lower-bound analogue—that we call *anti-potential* functions—to reason about the relative cost between two program versions. While we have drawn inspiration from lower-bound analogues in other domains [FNBG20, WFG+19, NDFH17], the key contribution here is computing a differential threshold value on the maximal difference in cost between two programs *simultaneously* with potential and anti-potential functions—one that provides an upper bound on the cost incurred in the new version and the other that provides a lower bound on the cost incurred in the old for the *same inputs*.

The simultaneous computation is done by employing a *constraint solving-based approach* similar to automation methods used in Chapters 3, 4 and 5, which collects the necessary constraints on potential and anti-potential functions to serve as upper and lower bounds on incurred cost in the program versions, as well as the differential cost constraint. The constraint solving-based approach allows our method to provide several key properties: (1) our method can be fully automated, (2) it reduces synthesis to linear programming, hence it allows efficient *optimization* of the threshold value by introducing a minimization objective in the linear program, (3) it does not depend on

*This work was performed in part while the author was an Applied Scientist Intern at Amazon.

syntactic alignment of programs and is suitable for programs that are not syntactically similar, and (4) it supports non-determinism in the programming language.

Contributions. Our contributions can be summarized as follows:

1. We present a new method for differential cost analysis that uses potential and anti-potential functions to reason about relative incurred cost.
2. We give an algorithm for deriving potential and anti-potential functions simultaneously with a threshold value on the difference in cost in imperative numerical programs with polynomial arithmetic and non-determinism.
3. Our experimental evaluation demonstrates the ability of our method to compute *tight* threshold values for differential cost analysis.

8.1.2 Verification of Bayesian Neural Networks

This section is based on the following publication:

- Mathias Lechner*, Đorđe Žikelić,* Krishnendu Chatterjee, Thomas A. Henzinger. *Infinite Time Horizon Safety of Bayesian Neural Networks*. In Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems, **NeurIPS 2021**

Bayesian neural networks (BNNs) are a family of neural networks that place distributions over their weights [Nea12]. This allows learning uncertainty in the data and the network's prediction, while preserving the strong modelling capabilities of neural networks [Mac92] and makes BNNs very appealing for robotic and medical applications [MGK+17] where uncertainty is a central component of data. However, despite the large body of literature on verifying safety of neural networks, the formal safety verification of BNNs has received less attention. Notably, [CKL+19, WLPK20, MWL+20] have proposed sampling-based techniques for obtaining statistical guarantees about BNNs. Although these approaches provide insight into BNN safety, they suffer from two key limitations – they do not provide formal guarantees and can only simulate the system for a finite time.

In this work, we study the safety verification problem for BNN policies in safety-critical systems over the infinite time horizon. Formally, we consider a discrete-time deterministic dynamical system with a BNN policy. Given a set of initial states and a set of unsafe (or bad) states, the goal of the safety verification problem is to verify that no system execution starting in an initial state can reach an unsafe state. Our

*Equal contribution.

goal is to verify *sure safety*, i.e. safety of every system execution of the system. We present a method for computing *safe weight sets* for which every system execution is safe as long as the BNN samples its weights from this set. Verifying that a weight set is safe allows re-calibrating the BNN policy by rejecting unsafe weight samples in order to guarantee safety.

To verify safety of a weight set, we search for a safety certificate in the form of a *safe positive invariant* (also known as *safe inductive invariant*). A safe positive invariant is a set of system states that contains all initial states, is closed under the system dynamics and does not contain any unsafe state. We parametrize safe positive invariant candidates by (deterministic) neural networks that classify system states for determining set inclusion and present a learner-verifier framework for learning and formally verifying safe positive invariants building on the ideas presented in Chapters 6 and 7.

Contributions. Our contributions can be summarized as follows:

1. We define a safety verification problem for BNN policies by computing and verifying safe weight sets. The problem generalizes the sure safety verification of BNNs and solving it allows re-calibrating BNN policies via rejection sampling to guarantee safety.
2. We introduce a method for computing safe weight sets in BNN policies in the form of products of intervals around the BNN weights' means. To verify safety of a weight set, our novel algorithm learns a safe positive invariant in the form of a deterministic neural network.
3. We evaluate our methodology on a series of benchmark applications, including non-linear systems with safety specifications.

8.1.3 Verification of Distributional Safety in MDPs

This section is based on the following publication:

- S. Akshay, Krishnendu Chatterjee, Tobias Meggendorfer, Đorđe Žikelić[†]. *MDPs as Distribution Transformers: Affine Invariant Synthesis for Safety Objectives*. In Computer Aided Verification - 35th International Conference, **CAV 2023**

Markov decision processes (MDPs) are a classical model for probabilistic decision making systems. In the verification community, MDPs are often viewed through an automata-theoretic lens, as state transformers, with runs being sequences of states

[†]Authors ordered alphabetically.

with certain probability for taking each run. However, in several contexts such as modeling biochemical networks or queuing theory, it is also convenient to view MDPs as transformers of probability distributions over the states, and define objectives over these distributions [CKV+11, KVAK10, AGV18]. In this framework, we can, for instance, reason about properties such as the probability in a set of states always being above a given threshold or comparing the probability in two states at some future time point. As shown in [BRS06] such distribution-based properties cannot be expressed in pCTL*.

Unfortunately, and perhaps surprisingly, when we view them as distribution transformers even the simplest reachability and safety problems are known to be computationally intractable. In [AAOW15], it is shown that even with just Markov chains, reachability is as hard as the so-called Skolem problem, and safety is as hard as the positivity problem [OW14, OW15], and the decidability of both are long-standing open problems in linear recurrence sequences. In light of this difficulties, what can one do to tackle these problems *in theory and in practice*? In this work, we take an over-approximation route to tackling these questions, not only to check existence of strategies for safety but also synthesize them. Our goal is to develop a novel invariant-synthesis based approach towards strategy synthesis in MDPs. We restrict our attention to a class of safety objectives on MDPs, which are already general enough to capture several interesting and natural problems on MDPs.

Contributions. Our contributions can be summarized as follows:

1. We define the notion of *inductive distributional invariants* for safety in MDPs and show that they provide *sound and complete certificates* for proving safety objectives in MDPs. In doing so, we formalize the link between strategies and distributional invariants in MDPs.

2. We develop two algorithms for automated synthesis of *affine* inductive distributional invariants that prove safety in MDPs, and *at the same time*, synthesize strategies. The first algorithm is restricted to synthesizing memoryless strategies but is *relatively complete*, whereas the second algorithm can synthesize general strategies but is incomplete. In both cases, we draw insight from invariant generation in program analysis and employ a constraint solving-based approach similar to automation methods in Chapters 3, 4 and 5 to reduce synthesis to the existential first-order theory of reals, giving a PSPACE complexity upper bound.

3. We implement our approaches and show that for several practical and non-trivial examples, affine invariants suffice.

8.1.4 Bidding Games on Graphs

This section is based on the following publications:

- Guy Avni, Thomas A. Henzinger, Đorđe Žikelić[†]. *Bidding Mechanisms in Graph Games*. In 44th International Symposium on Mathematical Foundations of Computer Science, **MFCS 2019**
- Guy Avni, Ismaël Jecker, Đorđe Žikelić[†]. *Infinite-duration All-pay Bidding Games*. In 2021 ACM-SIAM Symposium on Discrete Algorithms, **SODA 2021**
- Guy Avni, Thomas A. Henzinger, Đorđe Žikelić[†]. *Bidding Mechanisms in Graph Games*. In Journal of Computer and System Sciences, **JCSS 2021**
- Guy Avni, Ismaël Jecker, Đorđe Žikelić[†]. *Bidding Graph Games with Partially-observable Budgets*. In Thirty-Seventh AAAI Conference on Artificial Intelligence, **AAAI 2023**

Two player infinite-duration games on graphs are a central class of games studied in formal methods with applications in reactive synthesis [PR89]. A graph game places a token at some initial vertex, and the game proceeds by players moving the token along the graph edges in order to produce an infinite path in the graph. There are several mechanisms to determine which player gets to move the token. In *bidding games* [LLPU96, LLP$^+$99], players have budgets and in each turn hold an *auction (i.e. bidding)* to determine which player gets to move the token. Both players simultaneously submit bids and the player with the higher bid wins the auction and gets to move the token. Bidding games on graphs provide a natural model for stateful and ongoing auctions in which budgets do not contribute to players' utilities, with applications in settings such as online advertising.

A bidding game is specified by the bidding mechanism which determines what happens with the invested bids and by the players' objectives. We classify bidding mechanisms according to two orthogonal properties: *who pays* and *who is the recipient*. In terms of who pays the bids, we distinguish between *first-price* bidding, in which only the higher bidder pays, and *all-pay* in which both players pay their bids. In terms of who receives the bids, we distinguish between *Richman bidding* in which payments are paid to the other player [LLPU96], and *poorman bidding* in which payments are paid to the "bank" thus the money is lost [LLP$^+$99]. A payment scheme called *taxman* spans the spectrum between Richman and poorman [LLP$^+$99]. In terms of players' objectives, we consider zero-sum games with qualitative (e.g. reachability or more generally parity) and with

[†]Authors ordered alphabetically.

mean-payoff objectives where vertices of the graph are assigned weights and the goal of players is to maximize and minimize the long-run average sum of weights.

The central notion in the study of bidding games is the *initial ratio* of budgets – if Player 1 has initial budget B_1 and Player 2 has initial budget B_2, then the initial ratio of budgets is $B_1/(B_1 + B_2)$. In bidding games with qualitative objectives (such as reachability or parity), we are interested in computing a *threshold ratio* which is the smallest necessary and sufficient initial ratio with which Player 1 is guaranteed to have a winning strategy. In bidding games with mean-payoff objectives, given an initial ratio of budgets we are interested in the *optimal payoff* that Player 1 can guarantee.

Prior work has focused on *first-price bidding games*. In particular, for games with qualitative objectives, algorithms have been proposed for computing threshold ratios for first-price Richman reachability [LLPU96] and more generally parity [AHC19] games, as well as first-price poorman reachability [LLP$^+$99] and more generally parity games [AHI18]. For games with mean-payoff objectives, algorithms for computing optimal payoffs have been proposed first-price Richman mean-payoff games [AHC19] and first-price poorman mean-payoff games [AHI18].

Contributions. Our contributions in this line of work can be summarized as follows:

- In [AHZ19, AHZ21], we study first-price taxman bidding games and propose algorithms for computing threshold ratios in reachability and parity games and optimal payoffs in mean-payoff games.

- In [AJZ21], we study all-pay bidding games under Richman, poorman and taxman bidding mechanisms on strongly-connected graphs. We propose algorithms for computing threshold ratios in parity games and optimal payoffs in mean-payoff games. Unlike first-price bidding games in which players have pure optimal strategies, we show that in all-pay mean-payoff games players may need mixed (i.e. randomized) strategies in order to achieve optimal payoff. Quite surprisingly, for solving mean-payoff games we use *martingale-based techniques* that we used in Chapters 3 and 4 to prove the "expected trend" achieved by a player's mixed strategy and deduce optimality of the strategy.

- In [AJZ23], we initiate the study of bidding games with *partially-observable budgets*, in which players only have prior belief about the probability distribution according to which the opponent's initial budget is distributed.

8.2 Conclusion and Future Perspective

In this thesis, we considered formal verification and controller synthesis in infinite state stochastic systems. We studied formal verification in the setting of static analysis of probabilistic programs and formal controller synthesis in the setting of control of stochastic dynamical systems. In both settings, our goal was to enable and advance

fully automated reachability and safety analysis
in infinite-time horizon stochastic systems,

which have been mostly out of the reach of the existing formal methods approaches.

To solve this challenge, we followed a *martingale-based approach*. The main theoretical contribution of this thesis is the design of novel martingale-based formal certificates for reasoning about qualitative reachability as well as quantitative reachability, safety and reach-avoidance in infinite state stochastic systems. These certificates are theoretical in nature and can be instantiated both in the setting of PPs and of stochastic dynamical systems and do not impose any restrictions on the time horizon over which the stochastic system is executed. The main algorithmic contribution of this thesis is full automation of the computation of these martingale-based certificates, both in PPs and in stochastic dynamical systems. More concretely, we studied the following problems:

- In Chapter 3, we the *probability* 1 *(a.k.a. almost-sure) termination/reachability* analysis in PPs. We introduced generalized lexicographic ranking supermartingales (GLexRSMs), a lexicographic extension of RSMs, allowing for sound and compositional reasoning about almost-sure termination analysis in PPs with more complex control-flow structure. We presented two constraint-solving based polynomial time algorithms for automated synthesis of linear GLexRSMs in linear arithmetic PPs. We demonstrated their applicability on PPs for which no existing linear arithmetic method could prove almost-sure termination, including PPs with double-sided unbounded support probability distributions such as normal distribution that commonly arise in probabilistic modelling.

- In Chapter 4, we studied the *quantitative termination/reachability and safety* analyses in PPs. To the best of our knowledge, we presented the first fully automated methods for quantitative reachability and safety analysis in PPs that may not be almost-surely terminating. We achieved this by introducing *stochastic invariants (SIs)* and using them to formulate sound and complete martingale-based certificates for quantitative reachability and safety. We then presented a constraint-solving based algorithm for automated synthesis of these SI-based certificates in polynomial arithmetic PPs. Our algorithm is sound, relatively

complete and applicable to polynomial arithmetic PPs that need not be almost-surely terminating. We experimentally evaluated our prototype implementation and demonstrate its effectiveness on a number of PPs that previous methods could not handle.

- In Chapter 5, we studied *non-termination* analysis, both in non-probabilistic programs and in PPs. We presented a new formal certificate for non-termination proving in programs with non-determinism. The certificate is based on a purely syntactic reversal of the program's transition system and a of combination forward and backward reasoning. We then presented an algorithm for automated constraint-solving based synthesis of our non-termination certificate in polynomial arithmetic programs. Experimental evaluation of our prototype tool RevTerm showed that, despite its simplicity and stronger theoretical guarantees, RevTerm outperforms all non-termination proving tools that competed in the TermComp'19 competition, both in the number of proved non-terminations and in runtime.
- In Chapter 6, we studied *controller synthesis with probability* 1 *reachability guarantees* in stochastic dynamical systems. We proposed the first method for learning and formally verifying neural controllers for almost-sure reachability in stochastic dynamical systems. Our learner-verifier method jointly learns and verifies a control policy and an RSM, both parametrized as neural networks. The method is applicable to infinite-time horizon systems with non-polynomial dynamics, thus overcoming the limitations of prior work, and can also be used to verify probability 1 reachability under a given neural network control policy. We experimentally evaluated our method on several non-linear RL environments.
- In Chapter 7, we considered *controller synthesis with quantitative reachability and safety guarantees* in stochastic dynamical systems. Building on and extending our results in Chapter 6, we proposed the first method for learning and formally verifying neural controllers for quantitative reachability, safety and reach-avoidance. The method jointly learns a neural control certificate and a neural *reach-avoid supermartingale (RASM)*, a novel martingale-based certificate for quantitative reach-avoidance that we introduce. As above, the method is applicable to infinite-time horizon systems with non-polynomial dynamics, thus overcoming the limitations of prior work, and can also be used to verify quantitative reach-avoidance under a given neural network control policy. We experimentally evaluated our method on several non-linear RL environments.

While this thesis advances the state of the art of automated formal analysis of infinite state stochastic systems, this problem is far from being solved. There are several important challenges that researchers in formal methods, programming languages, artificial intelligence and control theory communities need to address for formal analysis

of infinite state stochastic systems to scale to real-world applications. In what follows, we identify some of these challenges and discuss possible future research directions.

Scaling PP analysis to richer languages. Recent years have seen much work on PP analysis with several very exciting developments. However, automated methods for formal analysis of PPs are still mostly focused on academic examples towards demonstrating the ability to handle complex control-flow structures such as nested loops, nested branching or complicated probability distributions. In order for the PP analysis to move forward and ultimately be adopted by users of probabilistic programming and inference tools, there are two fundamental challenges that need to be addressed: *scalability* of automated analyses and *expressivity* of analyzed PP languages.

The key barrier to scaling PP analysis is that existing automated methods are *not compositional*. For instance, automation of methods presented in this thesis proceeds via constraint-solving based synthesis of martingale-based certificates, which reduces the analysis to solving a large system of constraints encoding the whole PP. Such an approach will not scale to large systems. A promising approach to overcoming this challenge is to derive *compositional variants* of martingale-based certificates. For instance, lexicographic extensions of RSMs provide compositional proof rules for probability 1 termination analysis [ACN18], so our GLexRSMs in Chapter 3 are a step in the right direction. An interesting direction of future work is to revisit martingale-based certificates for quantitative termination and safety and to think about whether we could use them to formulate compositional proof rules for PP analysis.

When it comes to the expressivity of PP languages, most formal analyses of PPs focus on programs with numerical datatypes. However, all modern PP languages contain datatypes for arrays which current automated methods do not support. A step in the right direction towards enabling formal analysis of PPs with arrays has been made in [BKK$^+$19], which proposes quantitative separation logic for PPs. Studying automated methods for formal analysis of PPs with arrays is an exciting venue for future work.

Stochastic control under general specifications. This thesis considers two important classes of specifications – reachability and safety, as well as reach-avoidance which is defined via their conjunction. However, these are by no means all specifications that one may consider in practice. An interesting direction of future work would be to extend the learner-verifier framework in Chapters 6 and 7 to a richer class of probabilistic specifications, e.g. pLTL specifications. Another venue for future work is to consider control problems where the goal is to optimize some quantitative specification (e.g. maximize expected total reward or mean-payoff) while guaranteeing satisfaction of some pLTL specification. For instance, we may want to learn a control policy that minimizes travel time to some target destination while ensuring avoidance of obstacles.

Bibliography

[AAGP21] Alessandro Abate, Daniele Ahmed, Mirco Giacobbe, and Andrea Peruffo. Formal synthesis of lyapunov neural networks. *IEEE Control. Syst. Lett.*, 5(3):773–778, 2021.

[AAOW15] S. Akshay, Timos Antonopoulos, Joël Ouaknine, and James Worrell. Reachability problems for markov chains. *Inf. Process. Lett.*, 115(2):155–158, 2015.

[ABE+18] Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. Safe reinforcement learning via shielding. In Sheila A. McIlraith and Kilian Q. Weinberger, editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 2669–2678. AAAI Press, 2018.

[ABH+21] Alejandro Aguirre, Gilles Barthe, Justin Hsu, Benjamin Lucien Kaminski, Joost-Pieter Katoen, and Christoph Matheja. A pre-expectation calculus for probabilistic sensitivity. *Proc. ACM Program. Lang.*, 5(POPL):1–28, 2021.

[ACE+19] Aaron D. Ames, Samuel Coogan, Magnus Egerstedt, Gennaro Notomista, Koushil Sreenath, and Paulo Tabuada. Control barrier functions: Theory and applications. In *17th European Control Conference, ECC 2019, Naples, Italy, June 25-28, 2019*, pages 3420–3431. IEEE, 2019.

[ACF+21] Ali Asadi, Krishnendu Chatterjee, Hongfei Fu, Amir Kafshdar Goharshady, and Mohammad Mahdavi. Polynomial reachability witnesses via stellensätze. In Stephen N. Freund and Eran Yahav, editors, *PLDI '21: 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation, Virtual Event, Canada, June 20-25, 2021*, pages 772–787. ACM, 2021.

[ACG+22] Ali Ahmadi, Krishnendu Chatterjee, Amir Kafshdar Goharshady, Tobias Meggendorfer, Roodabeh Safavi, and Dorde Zikelic. Algorithms and hardness results for computing cores of markov chains. In *42nd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS*, 2022.

[ACMZ23] S. Akshay, Krishnendu Chatterjee, Tobias Meggendorfer, and Dorde Zikelic. Mdps as distribution transformers: Affine invariant synthesis for safety objectives. In Constantin Enea and Akash Lal, editors, *Computer Aided Verification - 35th International Conference, CAV 2023, Paris, France, July 17-22, 2023, Proceedings, Part III*, volume 13966 of *Lecture Notes in Computer Science*, pages 86–112. Springer, 2023.

[ACN18] Sheshansh Agrawal, Krishnendu Chatterjee, and Petr Novotný. Lexicographic ranking supermartingales: an efficient approach to termination of probabilistic programs. *Proc. ACM Program. Lang.*, 2(POPL):34:1–34:32, 2018.

[ADB11] Alessandro Abate, Alessandro D'Innocenzo, and Maria Domenica Di Benedetto. Approximate abstractions of stochastic hybrid systems. *IEEE Trans. Autom. Control.*, 56(11):2688–2694, 2011.

[ADD00] R.B. Ash and C. Doléans-Dade. *Probability and Measure Theory*. Academic Press, 2000.

[ADFG10] Christophe Alias, Alain Darte, Paul Feautrier, and Laure Gonnord. Multi-dimensional rankings, program termination, and complexity bounds of flowchart programs. In *Proceedings of the 17th International Conference on Static Analysis*, SAS'10, pages 117–133, Berlin, Heidelberg, 2010. Springer-Verlag.

[ADG19] M. Avanzini, U. Dal Lago, and A. Ghyselen. Type-based complexity analysis of probabilistic functional programs. In *2019 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–13, 2019.

[AGR21] Alessandro Abate, Mirco Giacobbe, and Diptarko Roy. Learning probabilistic termination proofs. In Alexandra Silva and K. Rustan M. Leino, editors, *Computer Aided Verification - 33rd International Conference, CAV 2021, Virtual Event, July 20-23, 2021, Proceedings, Part II*, volume 12760 of *Lecture Notes in Computer Science*, pages 3–26. Springer, 2021.

[AGV18] S. Akshay, Blaise Genest, and Nikhil Vyas. Distribution-based objectives for markov decision processes. In Anuj Dawar and Erich Grädel, editors, *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018*, pages 36–45. ACM, 2018.

[AHC19] Guy Avni, Thomas A. Henzinger, and Ventsislav Chonev. Infinite-duration bidding games. *J. ACM*, 66(4):31:1–31:29, 2019.

[AHI18] Guy Avni, Thomas A. Henzinger, and Rasmus Ibsen-Jensen. Infinite-duration poorman-bidding games. In George Christodoulou and Tobias Harks, editors, *Web and Internet Economics - 14th International Conference, WINE 2018, Oxford, UK, December 15-17, 2018, Proceedings*, volume 11316 of *Lecture Notes in Computer Science*, pages 21–36. Springer, 2018.

[AHTA17] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *International Conference on Machine Learning*, pages 22–31. PMLR, 2017.

[AHZ19] Guy Avni, Thomas A. Henzinger, and Dorde Zikelic. Bidding mechanisms in graph games. In *44th International Symposium on Mathematical Foundations of Computer Science, MFCS*, 2019.

[AHZ21] Guy Avni, Thomas A. Henzinger, and Dorde Zikelic. Bidding mechanisms in graph games. *J. Comput. Syst. Sci.*, 2021.

[AJZ21] Guy Avni, Ismaël Jecker, and Dorde Zikelic. Infinite-duration all-pay bidding games. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA*, 2021.

[AJZ23] Guy Avni, Ismaël Jecker, and Dorde Zikelic. Bidding graph games with partially-observable budgets. In *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI*, 2023.

[AKP11] Amir Ali Ahmadi, Miroslav Krstic, and Pablo A. Parrilo. A globally asymptotically stable polynomial vector field with no polynomial lyapunov function. In *50th IEEE Conference on Decision and Control and European Control Conference, 11th European Control Conference, CDC/ECC 2011, Orlando, FL, USA, December 12-15, 2011*, pages 7579–7580. IEEE, 2011.

[Alt99] Eitan Altman. *Constrained Markov decision processes*, volume 7. CRC Press, 1999.

[ALY20] Martin Avanzini, Ugo Dal Lago, and Akihisa Yamada. On probabilistic term rewriting. *Sci. Comput. Program.*, 185, 2020.

[AMS20] Martin Avanzini, Georg Moser, and Michael Schaper. A modular cost analysis for probabilistic programs. *Proceedings of the ACM on Programming Languages*, 4((Proceedings of OOPSLA 2020).):1–30, 2020.

[AOS+16] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul F. Christiano, John Schulman, and Dan Mané. Concrete problems in AI safety. *CoRR*, abs/1606.06565, 2016.

[APLS08] Alessandro Abate, Maria Prandini, John Lygeros, and Shankar Sastry. Probabilistic reachability and safety for controlled discrete time stochastic hybrid systems. *Autom.*, 44(11):2724–2734, 2008.

[Azu67] Kazuoki Azuma. Weighted sums of certain dependent random variables. *Tohoku Mathematical Journal, Second Series*, 19(3):357–367, 1967.

[BAG13] Amir M. Ben-Amram and Samir Genaim. On the linear ranking problem for integer linear-constraint loops. In *Proceedings of the 40th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '13, pages 51–62, New York, NY, USA, 2013. ACM.

[BAG15] Amir M. Ben-Amram and Samir Genaim. Complexity of bradley-manna-sipma lexicographic ranking functions. In Daniel Kroening and Corina S. Păsăreanu, editors, *Computer Aided Verification: 27th International Conference, CAV 2015, San Francisco, CA, USA, July 18-24, 2015, Proceedings, Part II*, pages 304–321. Springer International Publishing, 2015.

[BBL+17] Cristina Borralleras, Marc Brockschmidt, Daniel Larraz, Albert Oliveras, Enric Rodríguez-Carbonell, and Albert Rubio. Proving termination through conditional termination. In *Tools and Algorithms for the Construction and Analysis of Systems - 23rd International Conference, TACAS 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22-29, 2017, Proceedings, Part I*, pages 99–117, 2017.

[BCF13] Marc Brockschmidt, Byron Cook, and Carsten Fuhs. Better termination proving through cooperation. In *Computer Aided Verification - 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings*, pages 413–429, 2013.

[BCHT17] Somil Bansal, Mo Chen, Sylvia L. Herbert, and Claire J. Tomlin. Hamilton-jacobi reachability: A brief overview and recent advances. In *56th IEEE Annual Conference on Decision and Control, CDC 2017, Melbourne, Australia, December 12-15, 2017*, pages 2242–2253. IEEE, 2017.

[BCI+16] Marc Brockschmidt, Byron Cook, Samin Ishtiaq, Heidy Khlaaf, and Nir Piterman. T2: Temporal property verification. In Marsha Chechik and Jean-François Raskin, editors, *Tools and Algorithms for the Construction and Analysis of Systems: 22nd International Conference, TACAS 2016, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2016, Eindhoven, The Netherlands, April 2-8, 2016, Proceedings*, pages 387–393, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[BCJ+19] Eli Bingham, Jonathan P. Chen, Martin Jankowiak, Fritz Obermeyer, Neeraj Pradhan, Theofanis Karaletsos, Rohit Singh, Paul A. Szerlip, Paul Horsfall, and Noah D. Goodman. Pyro: Deep universal probabilistic programming. *J. Mach. Learn. Res.*, 20:28:1–28:6, 2019.

[BCLR04] Thomas Ball, Byron Cook, Vladimir Levin, and Sriram K. Rajamani. SLAM and static driver verifier: Technology transfer of formal methods inside microsoft. In Eerke A. Boiten, John Derrick, and Graeme Smith, editors, *Integrated Formal Methods, 4th International Conference, IFM 2004, Canterbury, UK, April 4-7, 2004, Proceedings*, volume 2999 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2004.

[BCP+16] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[BEFFH16] Gilles Barthe, Thomas Espitau, Luis María Ferrer Fioriti, and Justin Hsu. Synthesizing probabilistic invariants via doob's decomposition. In Swarat Chaudhuri and Azadeh Farzan, editors, *Computer Aided Verification: 28th International Conference, CAV 2016, Toronto, ON, Canada, July 17-23, 2016, Proceedings, Part I*, pages 43–61. Springer International Publishing, 2016.

[Ber12] Dimitri Bertsekas. *Dynamic programming and optimal control: Volume I*, volume 1. Athena scientific, 2012.

[Ber19] Felix Berkenkamp. Safe exploration in reinforcement learning: Theory and applications in robotics. *Ph.D. thesis, ETH Zurich.*, 2019.

[BG05] Olivier Bournez and Florent Garnier. Proving Positive Almost-Sure Termination. In *RTA*, pages 323–337, 2005.

[BGB09] Gilles Barthe, Benjamin Grégoire, and Santiago Zanella Béguelin. Formal certification of code-based cryptographic proofs. In Zhong Shao and Benjamin C. Pierce, editors, *Proceedings of the 36th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2009, Savannah, GA, USA, January 21-23, 2009*, pages 90–101. ACM, 2009.

[BGG+13] Johannes Borgström, Andrew D. Gordon, Michael Greenberg, James Margetson, and Jurgen Van Gael. Measure transformer semantics for bayesian machine learning. *Log. Methods Comput. Sci.*, 9(3), 2013.

[BGG+16] Gilles Barthe, Marco Gaboardi, Benjamin Grégoire, Justin Hsu, and Pierre-Yves Strub. Proving differential privacy via probabilistic couplings. In Martin Grohe, Eric Koskinen, and Natarajan Shankar, editors, *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '16, New York, NY, USA, July 5-8, 2016*, pages 749–758. ACM, 2016.

[BGHP16] Gilles Barthe, Marco Gaboardi, Justin Hsu, and Benjamin C. Pierce. Programming language techniques for differential privacy. *ACM SIGLOG News*, 3(1):34–53, 2016.

[BHHK03] Christel Baier, Boudewijn R. Haverkort, Holger Hermanns, and Joost-Pieter Katoen. Model-checking algorithms for continuous-time markov chains. *IEEE Trans. Software Eng.*, 29(6):524–541, 2003.

[Bil95] P. Billingsley. *Probability and Measure*. Wiley, 3rd edition, 1995.

[BK08] Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT Press, 2008.

[BK11] Dirk Beyer and M. Erkan Keremoglu. Cpachecker: A tool for configurable software verification. In *Computer Aided Verification - 23rd International Conference, CAV 2011, Snowbird, UT, USA, July 14-20, 2011. Proceedings*, pages 184–190, 2011.

[BKK+19] Kevin Batz, Benjamin Lucien Kaminski, Joost-Pieter Katoen, Christoph Matheja, and Thomas Noll. Quantitative separation logic: a logic for reasoning about probabilistic pointer programs. *Proc. ACM Program. Lang.*, 3(POPL):34:1–34:29, 2019.

[BKKM21] Kevin Batz, Benjamin Lucien Kaminski, Joost-Pieter Katoen, and Christoph Matheja. Relatively complete verification of probabilistic programs: an expressive language for expectation-based reasoning. *Proc. ACM Program. Lang.*, 5(POPL):1–30, 2021.

[BKS19] Ezio Bartocci, Laura Kovács, and Miroslav Stankovic. Automatic generation of moment-based invariants for prob-solvable loops. In Yu-Fang Chen, Chih-Hong Cheng, and Javier Esparza, editors, *Automated Technology for Verification and Analysis - 17th International Symposium, ATVA 2019, Taipei, Taiwan, October 28-31, 2019, Proceedings*, volume 11781 of *Lecture Notes in Computer Science*, pages 255–276. Springer, 2019.

[BL21] Osbert Bastani and Shuo Li. Safe reinforcement learning via statistical model predictive shielding. In Dylan A. Shell, Marc Toussaint, and M. Ani Hsieh, editors, *Robotics: Science and Systems XVII, Virtual Event, July 12-16, 2021*, 2021.

[BMS05a] Aaron R. Bradley, Zohar Manna, and Henny B. Sipma. Linear Ranking with Reachability. In *Computer Aided Verification, 17th International Conference, CAV 2005, Edinburgh, Scotland, UK, July 6-10, 2005, Proceedings*, pages 491–504, 2005.

[BMS05b] Aaron R. Bradley, Zohar Manna, and Henny B. Sipma. Termination of polynomial programs. In *Verification, Model Checking, and Abstract Interpretation, 6th International Conference, VMCAI 2005, Paris, France, January 17-19, 2005, Proceedings*, pages 113–129, 2005.

[BNO+08] Miquel Bofill, Robert Nieuwenhuis, Albert Oliveras, Enric Rodríguez-Carbonell, and Albert Rubio. The barcelogic SMT solver. In *Computer Aided Verification, 20th International Conference, CAV 2008, Princeton, NJ, USA, July 7-14, 2008, Proceedings*, pages 294–298, 2008.

[BO21] Raven Beutner and Luke Ong. On probabilistic termination of functional programs with continuous distributions. In Stephen N. Freund and Eran Yahav, editors, *PLDI '21: 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation, Virtual Event, Canada, June 20-25, 2021*, pages 1312–1326. ACM, 2021.

[Bou93] François Bourdoncle. Abstract debugging of higher-order imperative languages. In Robert Cartwright, editor, *Proceedings of the ACM SIGPLAN'93 Conference on Programming Language Design and Implementation (PLDI), Albuquerque, New Mexico, USA, June 23-25, 1993*, pages 46–55. ACM, 1993.

[BOZ22] Raven Beutner, C.-H. Luke Ong, and Fabian Zaiser. Guaranteed bounds for posterior inference in universal probabilistic programming. In Ranjit Jhala and Isil Dillig, editors, *PLDI '22: 43rd ACM SIGPLAN International Conference on Programming Language Design and Implementation, San Diego, CA, USA, June 13 - 17, 2022*, pages 536–551. ACM, 2022.

[BR02] Thomas Ball and Sriram K. Rajamani. The SLAM project: debugging system software via static analysis. In *Conference Record of POPL 2002: The 29th SIGPLAN-SIGACT Symposium on Principles of Programming Languages, Portland, OR, USA, January 16-18, 2002*, pages 1–3, 2002.

[BRAJ22] Thom S. Badings, Licio Romao, Alessandro Abate, and Nils Jansen. Probabilities are not enough: Formal controller synthesis for stochastic dynamical models with epistemic uncertainty. *CoRR*, abs/2210.05989, 2022.

[BRS06] Danièle Beauquier, Alexander Moshe Rabinovich, and Anatol Slissenko. A logic of probability with decidable model checking. *J. Log. Comput.*, 16(4):461–487, 2006.

[BS96] Dimitri Bertsekas and Steven E Shreve. *Stochastic optimal control: the discrete-time case*, volume 5. Athena Scientific, 1996.

[BS04] Dimitir P Bertsekas and Steven Shreve. *Stochastic optimal control: the discrete-time case*. 2004.

[BSOG11] Marc Brockschmidt, Thomas Ströder, Carsten Otto, and Jürgen Giesl. Automated detection of non-termination and nullpointerexceptions for java bytecode. In *Formal Verification of Object-Oriented Software - International Conference, FoVeOOS 2011, Turin, Italy, October 5-7, 2011, Revised Selected Papers*, pages 123–141, 2011.

[BTSK17] Felix Berkenkamp, Matteo Turchetta, Angela P. Schoellig, and Andreas Krause. Safe model-based reinforcement learning with stability guarantees. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 908–918, 2017.

[CA19] Nathalie Cauchi and Alessandro Abate. Stochy-automated verification and synthesis of stochastic processes. In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*, pages 258–259, 2019.

[ÇBG+17] Ezgi Çiçek, Gilles Barthe, Marco Gaboardi, Deepak Garg, and Jan Hoffmann. Relational cost analysis. In Giuseppe Castagna and Andrew D. Gordon, editors, *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages, POPL 2017, Paris, France, January 18-20, 2017*, pages 316–329. ACM, 2017.

[CC77] Patrick Cousot and Radhia Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In Robert M. Graham, Michael A. Harrison, and Ravi Sethi, editors, *Conference Record of the Fourth ACM Symposium on Principles of Programming Languages, Los Angeles, California, USA, January 1977*, pages 238–252. ACM, 1977.

[CCF+14] Hong Yi Chen, Byron Cook, Carsten Fuhs, Kaustubh Nimkar, and Peter W. O'Hearn. Proving nontermination via safety. In *Tools and Algorithms for the Construction and Analysis of Systems - 20th International Conference, TACAS 2014, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2014, Grenoble, France, April 5-13, 2014. Proceedings*, pages 156–171, 2014.

[CDD+15] Cristiano Calcagno, Dino Distefano, Jérémy Dubreil, Dominik Gabi, Pieter Hooimeijer, Martino Luca, Peter W. O'Hearn, Irene Papakonstantinou, Jim Purbrick, and Dulma Rodriguez. Moving fast with software verification. In Klaus Havelund, Gerard J. Holzmann, and Rajeev Joshi, editors, *NASA Formal Methods - 7th International Symposium, NFM 2015, Pasadena, CA, USA, April 27-29, 2015, Proceedings*, volume 9058 of *Lecture Notes in Computer Science*, pages 3–11. Springer, 2015.

[CDM17] Dmitry Chistikov, Rayna Dimitrova, and Rupak Majumdar. Approximate counting in SMT and value estimation for probabilistic programs. *Acta Informatica*, 54(8):729–764, 2017.

[CFG16] Krishnendu Chatterjee, Hongfei Fu, and Amir Kafshdar Goharshady. Termination analysis of probabilistic programs through positivstellensatz's. In Swarat Chaudhuri and Azadeh Farzan, editors, *Computer Aided Verification - 28th International Conference, CAV 2016, Toronto, ON, Canada, July 17-23, 2016, Proceedings, Part I*, volume 9779 of *Lecture Notes in Computer Science*, pages 3–22. Springer, 2016.

[CFGG20] Krishnendu Chatterjee, Hongfei Fu, Amir Kafshdar Goharshady, and Ehsan Kafshdar Goharshady. Polynomial invariant generation for non-deterministic recursive programs. In Alastair F. Donaldson and Emina Torlak, editors, *Proceedings of the 41st ACM SIGPLAN International*

Conference on Programming Language Design and Implementation, PLDI 2020, London, UK, June 15-20, 2020, pages 672–687. ACM, 2020.

[CFNH16] Krishnendu Chatterjee, Hongfei Fu, Petr Novotný, and Rouzbeh Hasheminezhad. Algorithmic analysis of qualitative and quantitative termination problems for affine probabilistic programs. In Rastislav Bodík and Rupak Majumdar, editors, *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2016, St. Petersburg, FL, USA, January 20 - 22, 2016*, pages 327–342. ACM, 2016.

[CFNH18] Krishnendu Chatterjee, Hongfei Fu, Petr Novotný, and Rouzbeh Hasheminezhad. Algorithmic analysis of qualitative and quantitative termination problems for affine probabilistic programs. *ACM Trans. Program. Lang. Syst.*, 40(2):7:1–7:45, 2018.

[CFNO14] Byron Cook, Carsten Fuhs, Kaustubh Nimkar, and Peter W. O'Hearn. Disproving termination with overapproximation. In *Formal Methods in Computer-Aided Design, FMCAD 2014, Lausanne, Switzerland, October 21-24, 2014*, pages 67–74, 2014.

[CGMZ22a] Krishnendu Chatterjee, Amir Goharshady, Tobias Meggendorfer, and Dorde Zikelic. Sound and Complete Certificates for Quantitative Termination Analysis of Probabilistic Programs. 2022.

[CGMZ22b] Krishnendu Chatterjee, Amir Kafshdar Goharshady, Tobias Meggendorfer, and Dorde Zikelic. Sound and complete certificates for quantitative termination analysis of probabilistic programs. In *Computer Aided Verification - 34th International Conference, CAV*, 2022.

[CGN+21] Krishnendu Chatterjee, Ehsan Kafshdar Goharshady, Petr Novotný, Jiri Zárevúcky, and Dorde Zikelic. On lexicographic proof rules for probabilistic termination. In *Formal Methods - 24th International Symposium, FM*, 2021.

[CGNZ21] Krishnendu Chatterjee, Ehsan Kafshdar Goharshady, Petr Novotný, and Dorde Zikelic. Proving non-termination by program reversal. In *Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation, PLDI*, 2021.

[CGSS13] Alessandro Cimatti, Alberto Griggio, Bastiaan Joost Schaafsma, and Roberto Sebastiani. The MathSAT5 SMT solver. In *TACAS*, 2013.

[CH20] Jianhui Chen and Fei He. Proving almost-sure termination by omega-regular decomposition. In Alastair F. Donaldson and Emina Torlak, editors, *Proceedings of the 41st ACM SIGPLAN International Conference on Programming Language Design and Implementation, PLDI 2020, London, UK, June 15-20, 2020*, pages 869–882. ACM, 2020.

[CHL+18] Yu-Fang Chen, Matthias Heizmann, Ondrej Lengál, Yong Li, Ming-Hsien Tsai, Andrea Turrini, and Lijun Zhang. Advanced automata-based algorithms for program termination checking. In *Proceedings of the 39th ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2018, Philadelphia, PA, USA, June 18-22, 2018*, pages 135–150, 2018.

[CHLZ23] Krishnendu Chatterjee, Thomas A. Henzinger, Mathias Lechner, and Dorde Zikelic. A learner-verifier framework for neural network controllers and certificates of stochastic systems. In Sriram Sankaranarayanan and Natasha Sharygina, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 29th International Conference, TACAS 2023, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2022, Paris, France, April 22-27, 2023, Proceedings, Part I*, volume 13993 of *Lecture Notes in Computer Science*, pages 3–25. Springer, 2023.

[CHS15] Quentin Carbonneaux, Jan Hoffmann, and Zhong Shao. Compositional certified resource bounds. In David Grove and Stephen M. Blackburn, editors, *Proceedings of the 36th ACM SIGPLAN Conference on Programming Language Design and Implementation, Portland, OR, USA, June 15-17, 2015*, pages 467–478. ACM, 2015.

[CKGN+23] Krishnendu Chatterjee, Ehsan Kafshdar Goharshady, Petr Novotný, Jiří Zárevúcky, and undefinedorundefinede Žikelić. On lexicographic proof rules for probabilistic termination. *Form. Asp. Comput.*, 35(2), jun 2023.

[CKL+19] Luca Cardelli, Marta Kwiatkowska, Luca Laurenti, Nicola Paoletti, Andrea Patane, and Matthew Wicker. Statistical guarantees for the robustness of bayesian neural networks. In Sarit Kraus, editor, *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 5693–5700. ijcai.org, 2019.

[CKV+11] Rohit Chadha, Vijay Anand Korthikanti, Mahesh Viswanathan, Gul Agha, and YoungMin Kwon. Model checking mdps with a unique compact invariant set of distributions. In *Eighth International Conference on*

Quantitative Evaluation of Systems, QEST 2011, Aachen, Germany, 5-8 September, 2011, pages 121–130. IEEE Computer Society, 2011.

[CMMV16] Supratik Chakraborty, Kuldeep S. Meel, Rakesh Mistry, and Moshe Y. Vardi. Approximate probabilistic inference via word-level counting. In Dale Schuurmans and Michael P. Wellman, editors, *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, pages 3218–3224. AAAI Press, 2016.

[CNDG18] Yinlam Chow, Ofir Nachum, Edgar A. Duéñez-Guzmán, and Mohammad Ghavamzadeh. A lyapunov-based approach to safe reinforcement learning. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 8103–8112, 2018.

[CNZ17] Krishnendu Chatterjee, Petr Novotný, and Dorde Zikelic. Stochastic invariants for probabilistic termination. In *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages, POPL*, 2017.

[Cou05] Patrick Cousot. Proving program invariance and termination by parametric abstraction, lagrangian relaxation and semidefinite programming. In *Verification, Model Checking, and Abstract Interpretation, 6th International Conference, VMCAI 2005, Paris, France, January 17-19, 2005, Proceedings*, pages 1–24, 2005.

[CPR06] Byron Cook, Andreas Podelski, and Andrey Rybalchenko. Termination proofs for systems code. In *Proceedings of the ACM SIGPLAN 2006 Conference on Programming Language Design and Implementation, Ottawa, Ontario, Canada, June 11-14, 2006*, pages 415–426, 2006.

[CPR11] Byron Cook, Andreas Podelski, and Andrey Rybalchenko. Proving program termination. *Commun. ACM*, 54(5):88–98, 2011.

[CRG19] Ya-Chien Chang, Nima Roohi, and Sicun Gao. Neural lyapunov control. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 3240–3249, 2019.

[CS01] Michael Colón and Henny Sipma. Synthesis of linear ranking functions. In *Tools and Algorithms for the Construction and Analysis of Systems, 7th International Conference, TACAS 2001 Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2001 Genova, Italy, April 2-6, 2001, Proceedings*, pages 67–81, 2001.

[CS02] Michael Colón and Henny Sipma. Practical methods for proving program termination. In *Computer Aided Verification, 14th International Conference, CAV 2002,Copenhagen, Denmark, July 27-31, 2002, Proceedings*, pages 442–454, 2002.

[CS03] Luis G. Crespo and Jian-Qiao Sun. Stochastic optimal control via bellman's principle. *Autom.*, 39(12):2109–2114, 2003.

[CS13] Aleksandar Chakarov and Sriram Sankaranarayanan. Probabilistic program analysis with martingales. In Natasha Sharygina and Helmut Veith, editors, *Computer Aided Verification - 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings*, volume 8044 of *Lecture Notes in Computer Science*, pages 511–526. Springer, 2013.

[CSS03] Michael Colón, Sriram Sankaranarayanan, and Henny Sipma. Linear invariant generation using non-linear constraint solving. In Warren A. Hunt Jr. and Fabio Somenzi, editors, *Computer Aided Verification, 15th International Conference, CAV 2003, Boulder, CO, USA, July 8-12, 2003, Proceedings*, volume 2725 of *Lecture Notes in Computer Science*, pages 420–432. Springer, 2003.

[CSZ13] Byron Cook, Abigail See, and Florian Zuleger. Ramsey vs. lexicographic termination proving. In *Tools and Algorithms for the Construction and Analysis of Systems - 19th International Conference, TACAS 2013, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2013, Rome, Italy, March 16-24, 2013. Proceedings*, pages 47–61, 2013.

[CSZ+22] Krishnendu Chatterjee, Jakub Svoboda, Dorde Zikelic, Andreas Pavlogiannis, and Josef Tkadlec. Social balance on networks: Local minima and best-edge dynamics. *Phys. Rev. E*, 106, 2022.

[CVS16] Aleksandar Chakarov, Yuen-Lam Voronin, and Sriram Sankaranarayanan. Deductive Proofs of Almost Sure Persistence and Recurrence Properties. In Marsha Chechik and Jean-François Raskin, editors, *Tools and Algorithms for the Construction and Analysis of Systems: 22nd International*

Conference, TACAS 2016, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2016, Eindhoven, The Netherlands, April 2-8, 2016, Proceedings, pages 260–279, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[DCS19] Souradeep Dutta, Xin Chen, and Sriram Sankaranarayanan. Reachability analysis for neural feedback systems using regressive polynomial rule inference. In Necmiye Ozay and Pavithra Prabhakar, editors, *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control, HSCC 2019, Montreal, QC, Canada, April 16-18, 2019*, pages 157–168. ACM, 2019.

[DDV+18] Gal Dalal, Krishnamurthy Dvijotham, Matej Vecerík, Todd Hester, Cosmin Paduraru, and Yuval Tassa. Safe exploration in continuous action spaces. *ArXiv*, abs/1801.08757, 2018.

[DGF23] Charles Dawson, Sicun Gao, and Chuchu Fan. Safe control with learned certificates: A survey of neural lyapunov, barrier, and contraction methods for robotics and control. *IEEE Transactions on Robotics*, 2023.

[DHC22] Maxence Dutreix, Jeongmin Huh, and Samuel Coogan. Abstraction-based synthesis for stochastic systems with omega-regular objectives. *Nonlinear Analysis: Hybrid Systems*, 45:101204, 2022.

[Dij76] Edsger W. Dijkstra. *A Discipline of Programming*. Prentice-Hall, 1976.

[DLFR21] Ugo Dal Lago, Claudia Faggian, and Simona Ronchi Della Rocca. Intersection types and (positive) almost-sure termination. *Proc. ACM Program. Lang.*, 5(POPL), January 2021.

[dMB08] Leonardo Mendonça de Moura and Nikolaj Bjørner. Z3: an efficient SMT solver. In *Tools and Algorithms for the Construction and Analysis of Systems, 14th International Conference, TACAS 2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, March 29-April 6, 2008. Proceedings*, pages 337–340, 2008.

[EBA+21] Ingy Elsayed-Aly, Suda Bharadwaj, Christopher Amato, Rüdiger Ehlers, Ufuk Topcu, and Lu Feng. Safe multi-agent reinforcement learning via shielding. In Frank Dignum, Alessio Lomuscio, Ulle Endriss, and Ann Nowé, editors, *AAMAS '21: 20th International Conference on Autonomous Agents and Multiagent Systems, Virtual Event, United Kingdom, May 3-7, 2021*, pages 483–491. ACM, 2021.

[EGK12] Javier Esparza, Andreas Gaiser, and Stefan Kiefer. Proving termination of probabilistic programs using patterns. In *CAV 2012*, pages 123–138, 2012.

[Far02] Julius Farkas. Theorie der einfachen ungleichungen. *Journal für die reine und angewandte Mathematik*, 1902(124):1–27, 1902.

[FC19] Hongfei Fu and Krishnendu Chatterjee. Termination of nondeterministic probabilistic programs. In Constantin Enea and Ruzica Piskac, editors, *Verification, Model Checking, and Abstract Interpretation - 20th International Conference, VMCAI 2019, Cascais, Portugal, January 13-15, 2019, Proceedings*, volume 11388 of *Lecture Notes in Computer Science*, pages 468–490. Springer, 2019.

[Fel84] Yishai A. Feldman. A decidable propositional dynamic logic with explicit probabilities. *Information and Control*, 63(1):11–38, 1984.

[FG19] Florian Frohn and Jürgen Giesl. Proving non-termination via loop acceleration. In *2019 Formal Methods in Computer Aided Design, FMCAD 2019, San Jose, CA, USA, October 22-25, 2019*, pages 221–230, 2019.

[FGKP85] Nissim Francez, Orna Grumberg, Shmuel Katz, and Amir Pnueli. Proving termination of prolog programs. In *Logics of Programs, Conference, Brooklyn College, New York, NY, USA, June 17-19, 1985, Proceedings*, pages 89–105, 1985.

[FH82] Yishai A Feldman and David Harel. A probabilistic dynamic logic. In *Proceedings of the fourteenth annual ACM Symposium on Theory of computing*, pages 181–195. ACM, 1982.

[FH15] Luis María Ferrer Fioriti and Holger Hermanns. Probabilistic Termination: Soundness, Completeness, and Compositionality. In *Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2015, Mumbai, India, January 15-17, 2015*, pages 489–501, 2015.

[FHC+20] Jiameng Fan, Chao Huang, Xin Chen, Wenchao Li, and Qi Zhu. Reachnn*: A tool for reachability analysis of neural-network controlled systems. In Dang Van Hung and Oleg Sokolsky, editors, *Automated Technology for Verification and Analysis - 18th International Symposium, ATVA 2020, Hanoi, Vietnam, October 19-23, 2020, Proceedings*, volume 12302 of *Lecture Notes in Computer Science*, pages 537–542. Springer, 2020.

[FKM+16] Nate Foster, Dexter Kozen, Konstantinos Mamouras, Mark Reitblatt, and Alexandra Silva. Probabilistic netkat. In Peter Thiemann, editor, *Programming Languages and Systems - 25th European Symposium on Programming, ESOP 2016, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2016, Eindhoven, The Netherlands, April 2-8, 2016, Proceedings*, volume 9632 of *Lecture Notes in Computer Science*, pages 282–309. Springer, 2016.

[Flo67] Robert W. Floyd. Assigning meanings to programs. *Mathematical Aspects of Computer Science*, 19:19–33, 1967.

[FNBG20] Florian Frohn, Matthias Naaf, Marc Brockschmidt, and Jürgen Giesl. Inferring lower runtime bounds for integer programs. *ACM Trans. Program. Lang. Syst.*, 42(3), October 2020.

[Fos53] F. G. Foster. On the Stochastic Matrices Associated with Certain Queuing Processes. *The Annals of Mathematical Statistics*, 24(3):pp. 355–360, 1953.

[GAB+13] Andrew D. Gordon, Mihhail Aizatulin, Johannes Borgström, Guillaume Claret, Thore Graepel, Aditya V. Nori, Sriram K. Rajamani, and Claudio V. Russo. A model-learner pattern for bayesian reasoning. In Roberto Giacobazzi and Radhia Cousot, editors, *The 40th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL '13, Rome, Italy - January 23 - 25, 2013*, pages 403–416. ACM, 2013.

[GAB+17] Jürgen Giesl, Cornelius Aschermann, Marc Brockschmidt, Fabian Emmes, Florian Frohn, Carsten Fuhs, Jera Hensel, Carsten Otto, Martin Plücker, Peter Schneider-Kamp, Thomas Ströder, Stephanie Swiderski, and René Thiemann. Analyzing program termination and complexity automatically with aprove. *J. Autom. Reasoning*, 58(1):3–31, 2017.

[GCL+20] Sophie Gruenbacher, Jacek Cyranka, Mathias Lechner, Md. Ariful Islam, Scott A. Smolka, and Radu Grosu. Lagrangian reachtubes: The next generation. In *CDC*, pages 1556–1563. IEEE, 2020.

[GDS+18] Sven Gowal, Krishnamurthy Dvijotham, Robert Stanforth, Rudy Bunel, Chongli Qin, Jonathan Uesato, Relja Arandjelovic, Timothy A. Mann, and Pushmeet Kohli. On the effectiveness of interval bound propagation for training verifiably robust models. *CoRR*, abs/1810.12715, 2018.

[Gei06] Peter Geibel. Reinforcement learning for mdps with constraints. In Johannes Fürnkranz, Tobias Scheffer, and Myra Spiliopoulou, editors,

Machine Learning: ECML 2006, 17th European Conference on Machine Learning, Berlin, Germany, September 18-22, 2006, Proceedings, volume 4212 of *Lecture Notes in Computer Science*, pages 646–653. Springer, 2006.

[GF15] Javier García and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *J. Mach. Learn. Res.*, 16:1437–1480, 2015.

[GGH19] Jürgen Giesl, Peter Giesl, and Marcel Hark. Computing expected runtimes for constant probability programs. In Pascal Fontaine, editor, *Automated Deduction – CADE 27*, pages 269–286, Cham, 2019. Springer International Publishing.

[Gha15] Zoubin Ghahramani. Probabilistic machine learning and artificial intelligence. *Nature*, 521(7553):452–459, 2015.

[GHK+06] Bhargav S. Gulavani, Thomas A. Henzinger, Yamini Kannan, Aditya V. Nori, and Sriram K. Rajamani. SYNERGY: a new algorithm for property checking. In *Proceedings of the 14th ACM SIGSOFT International Symposium on Foundations of Software Engineering, FSE 2006, Portland, Oregon, USA, November 5-11, 2006*, pages 117–127, 2006.

[GHKW21] Mirco Giacobbe, Mohammadhosein Hasanbeig, Daniel Kroening, and Hjalmar Wijk. Shielding atari games with bounded prescience. In Frank Dignum, Alessio Lomuscio, Ulle Endriss, and Ann Nowé, editors, *AAMAS '21: 20th International Conference on Autonomous Agents and Multiagent Systems, Virtual Event, United Kingdom, May 3-7, 2021*, pages 1507–1509. ACM, 2021.

[GHM+08] Ashutosh Gupta, Thomas A. Henzinger, Rupak Majumdar, Andrey Rybalchenko, and Ru-Gang Xu. Proving non-termination. In *Proceedings of the 35th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2008, San Francisco, California, USA, January 7-12, 2008*, pages 147–158, 2008.

[GHNR14] Andrew D. Gordon, Thomas A. Henzinger, Aditya V. Nori, and Sriram K. Rajamani. Probabilistic programming. In James D. Herbsleb and Matthew B. Dwyer, editors, *Proceedings of the on Future of Software Engineering, FOSE 2014, Hyderabad, India, May 31 - June 7, 2014*, pages 167–181. ACM, 2014.

[GK01] J. W. Grizzle and Jun-Mo Kang. Discrete-time control design with positive semi-definite lyapunov functions. *Syst. Control. Lett.*, 43(4):287–292, 2001.

[GKM14] Friedrich Gretz, Joost-Pieter Katoen, and Annabelle McIver. Operational versus weakest pre-expectation semantics for the probabilistic guarded command language. *Performance Evaluation*, 73:110 – 132, 2014. Special Issue on the 9th International Conference on Quantitative Evaluation of Systems.

[GKS05] Patrice Godefroid, Nils Klarlund, and Koushik Sen. DART: directed automated random testing. In *Proceedings of the ACM SIGPLAN 2005 Conference on Programming Language Design and Implementation, Chicago, IL, USA, June 12-15, 2005*, pages 213–223, 2005.

[GLH+21] Sophie Gruenbacher, Mathias Lechner, Ramin Hasani, Daniela Rus, Thomas A Henzinger, Scott Smolka, and Radu Grosu. Gotube: Scalable stochastic verification of continuous-depth models. *arXiv preprint arXiv:2107.08467*, 2021.

[GMR+08] Noah D. Goodman, Vikash K. Mansinghka, Daniel M. Roy, Kallista A. Bonawitz, and Joshua B. Tenenbaum. Church: a language for generative models. In David A. McAllester and Petri Myllymäki, editors, *UAI 2008, Proceedings of the 24th Conference in Uncertainty in Artificial Intelligence, Helsinki, Finland, July 9-12, 2008*, pages 220–229. AUAI Press, 2008.

[GMR15] Laure Gonnord, David Monniaux, and Gabriel Radanne. Synthesis of ranking functions using extremal counterexamples. In *Proceedings of the 36th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '15, pages 608–618, New York, NY, USA, 2015. ACM.

[GMV16] Timon Gehr, Sasa Misailovic, and Martin T. Vechev. PSI: exact symbolic inference for probabilistic programs. In Swarat Chaudhuri and Azadeh Farzan, editors, *Computer Aided Verification - 28th International Conference, CAV 2016, Toronto, ON, Canada, July 17-23, 2016, Proceedings, Part I*, volume 9779 of *Lecture Notes in Computer Science*, pages 62–83. Springer, 2016.

[GRS+19] Jürgen Giesl, Albert Rubio, Christian Sternagel, Johannes Waldmann, and Akihisa Yamada. The termination and complexity competition. In *Tools and Algorithms for the Construction and Analysis of Systems - 25 Years of TACAS: TOOLympics, Held as Part of ETAPS 2019, Prague, Czech Republic, April 6-11, 2019, Proceedings, Part III*, pages 156–166, 2019.

[GS14] Noah D Goodman and Andreas Stuhlmüller. The Design and Implementation of Probabilistic Programming Languages. `http://dippl.org`, 2014.

[GSV08] Sumit Gulwani, Saurabh Srivastava, and Ramarathnam Venkatesan. Program analysis as constraint solving. In Rajiv Gupta and Saman P. Amarasinghe, editors, *Proceedings of the ACM SIGPLAN 2008 Conference on Programming Language Design and Implementation, Tucson, AZ, USA, June 7-13, 2008*, pages 281–292. ACM, 2008.

[GSV20] Timon Gehr, Samuel Steffen, and Martin T. Vechev. λpsi: exact inference for higher-order probabilistic programs. In Alastair F. Donaldson and Emina Torlak, editors, *Proceedings of the 41st ACM SIGPLAN International Conference on Programming Language Design and Implementation, PLDI 2020, London, UK, June 15-20, 2020*, pages 883–897. ACM, 2020.

[GvRB00] Indranil Gupta, Robbert van Renesse, and Kenneth P. Birman. A probabilistically correct leader election protocol for large groups. In Maurice Herlihy, editor, *Distributed Computing, 14th International Conference, DISC 2000, Toledo, Spain, October 4-6, 2000, Proceedings*, volume 1914 of *Lecture Notes in Computer Science*, pages 89–103. Springer, 2000.

[HAH11] Jan Hoffmann, Klaus Aehlig, and Martin Hofmann. Multivariate amortized resource analysis. In Thomas Ball and Mooly Sagiv, editors, *Proceedings of the 38th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2011, Austin, TX, USA, January 26-28, 2011*, pages 357–370. ACM, 2011.

[HC11] Wassim M Haddad and VijaySekhar Chellaboina. *Nonlinear dynamical systems and control*. Princeton university press, 2011.

[HdBM20] Steven Holtzen, Guy Van den Broeck, and Todd D. Millstein. Scaling exact inference for discrete probabilistic programs. *Proc. ACM Program. Lang.*, 4(OOPSLA):140:1–140:31, 2020.

[HDM22] Zixin Huang, Saikat Dutta, and Sasa Misailovic. Automated quantized inference for probabilistic programs with AQUA. *Innov. Syst. Softw. Eng.*, 18(3):369–384, 2022.

[HDW17] Jan Hoffmann, Ankush Das, and Shu-Chun Weng. Towards automatic resource bound analysis for ocaml. In Giuseppe Castagna and Andrew D.

Gordon, editors, *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages, POPL 2017, Paris, France, January 18-20, 2017*, pages 359–373. ACM, 2017.

[HFC18] Mingzhang Huang, Hongfei Fu, and Krishnendu Chatterjee. New approaches for almost-sure termination of probabilistic programs. In Sukyoung Ryu, editor, *Programming Languages and Systems*, pages 181–201, Cham, 2018. Springer International Publishing.

[HFCG19] Mingzhang Huang, Hongfei Fu, Krishnendu Chatterjee, and Amir Kafshdar Goharshady. Modular verification for almost-sure termination of probabilistic programs. *Proc. ACM Program. Lang.*, 3(OOPSLA):129:1–129:29, 2019.

[HFL+19] Chao Huang, Jiameng Fan, Wenchao Li, Xin Chen, and Qi Zhu. Reachnn: Reachability analysis of neural-network controlled systems. *ACM Trans. Embed. Comput. Syst.*, 18(5s):106:1–106:22, 2019.

[HG05] Didier Henrion and Andrea Garulli. *Positive polynomials in control*, volume 312. Springer Science & Business Media, 2005.

[HH10] Jan Hoffmann and Martin Hofmann. Amortized resource analysis with polynomial potential. In Andrew D. Gordon, editor, *Programming Languages and Systems, 19th European Symposium on Programming, ESOP 2010, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2010, Paphos, Cyprus, March 20-28, 2010. Proceedings*, volume 6012 of *Lecture Notes in Computer Science*, pages 287–306. Springer, 2010.

[HHP14] Matthias Heizmann, Jochen Hoenicke, and Andreas Podelski. Termination analysis by learning terminating programs. In *Computer Aided Verification - 26th International Conference, CAV 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 18-22, 2014. Proceedings*, pages 797–813, 2014.

[HJK+22] Christian Hensel, Sebastian Junges, Joost-Pieter Katoen, Tim Quatmann, and Matthias Volk. The probabilistic model checker storm. *Int. J. Softw. Tools Technol. Transf.*, 24(4):589–610, 2022.

[HJMS02] Thomas A. Henzinger, Ranjit Jhala, Rupak Majumdar, and Grégoire Sutre. Lazy abstraction. In *Conference Record of POPL 2002: The 29th SIGPLAN-SIGACT Symposium on Principles of Programming Languages, Portland, OR, USA, January 16-18, 2002*, pages 58–70, 2002.

[HKGK20] Marcel Hark, Benjamin Lucien Kaminski, Jürgen Giesl, and Joost-Pieter Katoen. Aiming low is harder: induction for lower bounds in probabilistic program verification. *Proc. ACM Program. Lang.*, 4(POPL):37:1–37:28, 2020.

[HLNR10] William R. Harris, Akash Lal, Aditya V. Nori, and Sriram K. Rajamani. Alternation for termination. In *Static Analysis - 17th International Symposium, SAS 2010, Perpignan, France, September 14-16, 2010. Proceedings*, pages 304–319, 2010.

[HLZ21] Thomas A. Henzinger, Mathias Lechner, and Dorde Zikelic. Scalable verification of quantized neural networks. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI*, 2021.

[Hoa62] Charles AR Hoare. Quicksort. *The computer journal*, 5(1):10–16, 1962.

[Hoa69] Charles Antony Richard Hoare. An axiomatic basis for computer programming. *Communications of the ACM*, 12(10):576–580, 1969.

[HOPW18] Ehud Hrushovski, Joël Ouaknine, Amaury Pouly, and James Worrell. Polynomial invariants for affine programs. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018*, pages 530–539, 2018.

[HS21] Sofie Haesaert and Sadegh Soudjani. Robust dynamic programming for temporal logic control of stochastic systems. *IEEE Trans. Autom. Control.*, 66(6):2496–2511, 2021.

[Ica17] Thomas Icard. Beyond almost-sure termination. In Glenn Gunzelmann, Andrew Howes, Thora Tenbrink, and Eddy J. Davelaar, editors, *Proceedings of the 39th Annual Meeting of the Cognitive Science Society, CogSci 2017, London, UK, 16-29 July 2017*. cognitivesciencesociety.org, 2017.

[IWA+19] Radoslav Ivanov, James Weimer, Rajeev Alur, George J. Pappas, and Insup Lee. Verisig: verifying safety properties of hybrid systems with neural network controllers. In Necmiye Ozay and Pavithra Prabhakar, editors, *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control, HSCC 2019, Montreal, QC, Canada, April 16-18, 2019*, pages 169–178. ACM, 2019.

[JM09] Ranjit Jhala and Rupak Majumdar. Software model checking. *ACM Comput. Surv.*, 41(4):21:1–21:54, 2009.

[JWFT+03] Zachary Jarvis-Wloszek, Ryan Feeley, Weehong Tan, Kunpeng Sun, and Andrew Packard. Some controls applications of sum of squares programming. In *42nd IEEE international conference on decision and control (IEEE Cat. No. 03CH37475)*, volume 5, pages 4676–4681. IEEE, 2003.

[Kam19] Benjamin Lucien Kaminski. *Advanced weakest precondition calculi for probabilistic programs*. PhD thesis, RWTH Aachen University, Germany, 2019.

[Kat16] Joost-Pieter Katoen. The probabilistic model checking landscape. In Martin Grohe, Eric Koskinen, and Natarajan Shankar, editors, *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '16, New York, NY, USA, July 5-8, 2016*, pages 31–45. ACM, 2016.

[KBBR17] Zachary Kincaid, Jason Breck, Ashkan Forouhi Boroujeni, and Thomas W. Reps. Compositional recurrence analysis revisited. In *Proceedings of the 38th ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2017, Barcelona, Spain, June 18-23, 2017*, pages 248–262, 2017.

[KBTK18] T. Koller, Felix Berkenkamp, Matteo Turchetta, and A. Krause. Learning-based model predictive control for safe exploration. *2018 IEEE Conference on Decision and Control (CDC)*, pages 6059–6066, 2018.

[KCBR18] Zachary Kincaid, John Cyphert, Jason Breck, and Thomas W. Reps. Non-linear reasoning for invariant synthesis. *PACMPL*, 2(POPL):54:1–54:33, 2018.

[KF09] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models - Principles and Techniques*. MIT Press, 2009.

[Kha02] H.K. Khalil. *Nonlinear Systems*. Pearson Education. Prentice Hall, 2002.

[KK17] Benjamin Lucien Kaminski and Joost-Pieter Katoen. A weakest pre-expectation semantics for mixed-sign expectations. In *32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, June 20-23, 2017*, pages 1–12. IEEE Computer Society, 2017.

[KKDD01] Harold Joseph Kushner Kushner, Harold J Kushner, Paul G Dupuis, and Paul Dupuis. *Numerical methods for stochastic control problems in continuous time*, volume 24. Springer Science & Business Media, 2001.

[KKM18] Benjamin Lucien Kaminski, Joost-Pieter Katoen, and Christoph Matheja. On the hardness of analyzing probabilistic programs. *Acta Informatica*, pages 1–31, 2018.

[KKMO18] Benjamin Lucien Kaminski, Joost-Pieter Katoen, Christoph Matheja, and Federico Olmedo. Weakest precondition reasoning for expected runtimes of randomized algorithms. *J. ACM*, 65(5):30:1–30:68, 2018.

[KLG20] Naoki Kobayashi, Ugo Dal Lago, and Charles Grellois. On the termination problem for probabilistic higher-order recursive programs. *Log. Methods Comput. Sci.*, 16(4), 2020.

[KM19] J. Zico Kolter and Gaurav Manek. Learning stable deep dynamics models. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 11126–11134, 2019.

[KM20] Jan Kretínský and Tobias Meggendorfer. Of cores: A partial-exploration framework for markov decision processes. *Log. Methods Comput. Sci.*, 16(4), 2020.

[KMMM10] Joost-Pieter Katoen, Annabelle McIver, Larissa Meinicke, and Carroll C. Morgan. Linear-Invariant Generation for Probabilistic Programs: - Automated Support for Proof-Based Methods. In *SAS*, volume LNCS 6337, Springer, pages 390–406, 2010.

[KNP11] M. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: Verification of probabilistic real-time systems. In G. Gopalakrishnan and S. Qadeer, editors, *Proc. 23rd International Conference on Computer Aided Verification (CAV'11)*, volume 6806 of *LNCS*, pages 585–591. Springer, 2011.

[KO21] Andrew Kenyon-Roberts and C.-H. Luke Ong. Supermartingales, ranking functions and probabilistic lambda calculus. In *36th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2021, Rome, Italy, June 29 - July 2, 2021*, pages 1–13. IEEE, 2021.

[Koz81] Dexter Kozen. Semantics of probabilistic programs. *J. Comput. Syst. Sci.*, 22(3):328–350, 1981.

[Koz85] Dexter Kozen. A probabilistic PDL. *J. Comput. Syst. Sci.*, 30(2):162–178, 1985.

[Kus14] Harold J. Kushner. A partial history of the early development of continuous-time nonlinear stochastic systems theory. *Autom.*, 50(2):303–334, 2014.

[KVAK10] Vijay Anand Korthikanti, Mahesh Viswanathan, Gul Agha, and YoungMin Kwon. Reasoning about mdps as transformers of probability distributions. In *QEST 2010, Seventh International Conference on the Quantitative Evaluation of Systems, Williamsburg, Virginia, USA, 15-18 September 2010*, pages 199–208. IEEE Computer Society, 2010.

[KY14] Frank Kelly and Elena Yudovina. *Stochastic networks*, volume 2. Cambridge University Press, 2014.

[LB20] Shuo Li and Osbert Bastani. Robust model predictive shielding for safe reinforcement learning with stochastic dynamics. In *2020 IEEE International Conference on Robotics and Automation, ICRA 2020, Paris, France, May 31 - August 31, 2020*, pages 7166–7172. IEEE, 2020.

[LG19] Ugo Dal Lago and Charles Grellois. Probabilistic termination by monadic affine sized typing. *ACM Trans. Program. Lang. Syst.*, 41(2):10:1–10:65, 2019.

[LH18] Jan Leike and Matthias Heizmann. Geometric nontermination arguments. In *Tools and Algorithms for the Construction and Analysis of Systems - 24th International Conference, TACAS 2018, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2018, Thessaloniki, Greece, April 14-20, 2018, Proceedings, Part II*, pages 266–283, 2018.

[LKSZ20] Abolfazl Lavaei, Mahmoud Khaled, Sadegh Soudjani, and Majid Zamani. AMYTISS: parallelized automated controller synthesis for large-scale stochastic systems. In Shuvendu K. Lahiri and Chao Wang, editors, *Computer Aided Verification - 32nd International Conference, CAV 2020, Los Angeles, CA, USA, July 21-24, 2020, Proceedings, Part II*, volume 12225 of *Lecture Notes in Computer Science*, pages 461–474. Springer, 2020.

[LLF+20] Nathan P. Lawrence, Philip D. Loewen, Michael G. Forbes, Johan U. Backström, and R. Bhushan Gopaluni. Almost surely stable deep dynamics. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[LLP+99] Andrew J. Lazarus, Daniel E. Loeb, James Gary Propp, Walter R. Stromquist, and Daniel H. Ullman. Combinatorial games under auction play. *Games and Economic Behavior*, 27:229–264, 1999.

[LLPU96] A. J. Lazarus, D. E. Loeb, J. G. Propp, and D. Ullman. Richman games. *Games of No Chance*, 29:439–449, 1996.

[LNO+14] Daniel Larraz, Kaustubh Nimkar, Albert Oliveras, Enric Rodríguez-Carbonell, and Albert Rubio. Proving non-termination using max-smt. In *Computer Aided Verification - 26th International Conference, CAV 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 18-22, 2014. Proceedings*, pages 779–796, 2014.

[LQC15] Ton Chanh Le, Shengchao Qin, and Wei-Ngan Chin. Termination and non-termination specification inference. In *Proceedings of the 36th ACM SIGPLAN Conference on Programming Language Design and Implementation, Portland, OR, USA, June 15-17, 2015*, pages 489–498, 2015.

[LSC+20] Anqi Liu, Guanya Shi, Soon-Jo Chung, Anima Anandkumar, and Yisong Yue. Robust regression for safe exploration in control. In *L4DC*, 2020.

[Lya92] Aleksandr Mikhailovich Lyapunov. The general problem of the stability of motion. *International journal of control*, 55(3):531–534, 1992.

[LZC+23] Mathias Lechner, Dorde Zikelic, Krishnendu Chatterjee, Thomas A. Henzinger, and Daniela Rus. Quantization-aware interval bound propagation for training certifiably robust quantized neural networks. In *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI*, 2023.

[LZCH21a] Mathias Lechner, Dorde Zikelic, Krishnendu Chatterjee, and Thomas A. Henzinger. Infinite time horizon safety of bayesian neural networks. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems, NeurIPS*, 2021.

[LZCH21b] Mathias Lechner, Dorde Zikelic, Krishnendu Chatterjee, and Thomas A. Henzinger. Stability verification in stochastic control systems via neural network supermartingales. *CoRR*, abs/2112.09495, 2021.

[LZCH22] Mathias Lechner, Dorde Zikelic, Krishnendu Chatterjee, and Thomas A. Henzinger. Stability verification in stochastic control systems via neural network supermartingales. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI*, 2022.

[M+17] Aaron Meurer et al. SymPy: symbolic computing in python. *PeerJ Comput. Sci.*, 2017.

[Mac92] David J. C. MacKay. A practical bayesian framework for backpropagation networks. *Neural Comput.*, 4(3):448–472, 1992.

[MBKK21a] Marcel Moosbrugger, Ezio Bartocci, Joost-Pieter Katoen, and Laura Kovács. Automated termination analysis of polynomial probabilistic programs. In Nobuko Yoshida, editor, *Programming Languages and Systems - 30th European Symposium on Programming, ESOP 2021, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2021, Luxembourg City, Luxembourg, March 27 - April 1, 2021, Proceedings*, volume 12648 of *Lecture Notes in Computer Science*, pages 491–518. Springer, 2021.

[MBKK21b] Marcel Moosbrugger, Ezio Bartocci, Joost-Pieter Katoen, and Laura Kovács. The probabilistic termination tool amber. In Marieke Huisman, Corina S. Pasareanu, and Naijun Zhan, editors, *Formal Methods - 24th International Symposium, FM 2021, Virtual Event, November 20-26, 2021, Proceedings*, volume 13047 of *Lecture Notes in Computer Science*, pages 667–675. Springer, 2021.

[MCL23] Frederik Baymler Mathiesen, Simeon Craig Calvert, and Luca Laurenti. Safety certification for stochastic systems via neural barrier functions. *IEEE Control. Syst. Lett.*, 7:973–978, 2023.

[MGK+17] Rowan McAllister, Yarin Gal, Alex Kendall, Mark Van Der Wilk, Amar Shah, Roberto Cipolla, and Adrian Weller. Concrete problems for autonomous vehicle safety: Advantages of bayesian deep learning. International Joint Conferences on Artificial Intelligence, Inc., 2017.

[MM93] Hannah Michalska and David Q. Mayne. Robust receding horizon control of constrained nonlinear systems. *IEEE Trans. Autom. Control.*, 38(11):1623–1633, 1993.

[MM99] Carroll Morgan and A McIver. pgcl: Formal reasoning for random algorithms. 1999.

[MM04] Annabelle McIver and Carroll Morgan. Developing and Reasoning About Probabilistic Programs in *pGCL*. In *PSSE*, pages 123–155, 2004.

[MM05] Annabelle McIver and Carroll Morgan. *Abstraction, Refinement and Proof for Probabilistic Systems*. Monographs in Computer Science. Springer, 2005.

[MMB+22] Rayan Mazouz, Karan Muvvala, Akash Ratheesh Babu, Luca Laurenti, and Morteza Lahijanian. Safety guarantees for neural network dynamic systems via stochastic barrier functions. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[MMKK18] Annabelle McIver, Carroll Morgan, Benjamin Lucien Kaminski, and Joost-Pieter Katoen. A new proof rule for almost-sure termination. *Proc. ACM Program. Lang.*, 2(POPL):33:1–33:28, 2018.

[MMS96] Carroll Morgan, Annabelle McIver, and Karen Seidel. Probabilistic predicate transformers. *ACM Trans. Program. Lang. Syst.*, 18(3):325–353, 1996.

[MMSS24] Rupak Majumdar, Kaushik Mallik, Anne-Kathrin Schmuck, and Sadegh Soudjani. Symbolic control for stochastic systems via finite parity games. *Nonlinear Analysis: Hybrid Systems*, 51:101430, 2024.

[Mon00] David Monniaux. Abstract interpretation of probabilistic semantics. In Jens Palsberg, editor, *Static Analysis, 7th International Symposium, SAS 2000, Santa Barbara, CA, USA, June 29 - July 1, 2000, Proceedings*, volume 1824 of *Lecture Notes in Computer Science*, pages 322–339. Springer, 2000.

[Mon01] David Monniaux. An abstract analysis of the probabilistic termination of programs. In Patrick Cousot, editor, *Static Analysis, 8th International Symposium, SAS 2001, Paris, France, July 16-18, 2001, Proceedings*, volume 2126 of *Lecture Notes in Computer Science*, pages 111–126. Springer, 2001.

[MR95] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[Mur12] Kevin P. Murphy. *Machine learning - a probabilistic perspective*. Adaptive computation and machine learning series. MIT Press, 2012.

[MWL+20] Rhiannon Michelmore, Matthew Wicker, Luca Laurenti, Luca Cardelli, Yarin Gal, and Marta Kwiatkowska. Uncertainty quantification with statistical guarantees in end-to-end autonomous driving control. In *2020 IEEE International Conference on Robotics and Automation, ICRA 2020, Paris, France, May 31 - August 31, 2020*, pages 7344–7350. IEEE, 2020.

[NCH18] Van Chan Ngo, Quentin Carbonneaux, and Jan Hoffmann. Bounded expectations: resource analysis for probabilistic programs. In *PLDI 2018*, pages 496–512, 2018.

[NDFH17] Van Chan Ngo, Mario Dehesa-Azuara, Matthew Fredrikson, and Jan Hoffmann. Verifying and synthesizing constant-resource implementations with types. In *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017*, pages 710–728. IEEE Computer Society, 2017.

[Nea12] Radford M Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

[NK07] Martin R. Neuhäußer and Joost-Pieter Katoen. Bisimulation and logical preservation for continuous-time markov decision processes. In *CONCUR*, 2007.

[NRZ+15] Chris Newcombe, Tim Rath, Fan Zhang, Bogdan Munteanu, Marc Brooker, and Michael Deardeuff. How amazon web services uses formal methods. *Commun. ACM*, 58(4):66–73, 2015.

[NSK09] Martin R. Neuhäußer, Mariëlle Stoelinga, and Joost-Pieter Katoen. Delayed nondeterminism in continuous-time markov decision processes. In *FOSSACS*, 2009.

[OGJ+18] Federico Olmedo, Friedrich Gretz, Nils Jansen, Benjamin Lucien Kaminski, Joost-Pieter Katoen, and Annabelle McIver. Conditioning in probabilistic programming. *ACM Trans. Program. Lang. Syst.*, 40(1):4:1–4:50, 2018.

[OKKM16] Federico Olmedo, Benjamin Lucien Kaminski, Joost-Pieter Katoen, and Christoph Matheja. Reasoning about recursive probabilistic programs. In Martin Grohe, Eric Koskinen, and Natarajan Shankar, editors, *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '16, New York, NY, USA, July 5-8, 2016*, pages 672–681. ACM, 2016.

[OW14] Joël Ouaknine and James Worrell. Positivity problems for low-order linear recurrence sequences. In Chandra Chekuri, editor, *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 366–379. SIAM, 2014.

[OW15] Joël Ouaknine and James Worrell. On linear recurrence sequences and loop termination. *ACM SIGLOG News*, 2(2):4–13, 2015.

[PAA21] Andrea Peruffo, Daniele Ahmed, and Alessandro Abate. Automated and formal synthesis of neural barrier certificates for dynamical models. In Jan Friso Groote and Kim Guldstrand Larsen, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 27th International Conference, TACAS 2021, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2021, Luxembourg City, Luxembourg, March 27 - April 1, 2021, Proceedings, Part I*, volume 12651 of *Lecture Notes in Computer Science*, pages 370–388. Springer, 2021.

[Par00] Pablo A Parrilo. *Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization*. California Institute of Technology, 2000.

[PB02] Theodore J. Perkins and Andrew G. Barto. Lyapunov design for safe reinforcement learning. *J. Mach. Learn. Res.*, 3:803–832, 2002.

[PJP04] Stephen Prajna, Ali Jadbabaie, and George J. Pappas. Stochastic safety verification using barrier certificates. In *43rd IEEE Conference on Decision and Control, CDC 2004, Nassau, Bahamas, December 14-17, 2004*, pages 929–934. IEEE, 2004.

[PJP07] Stephen Prajna, Ali Jadbabaie, and George J. Pappas. A framework for worst-case and stochastic safety verification using barrier certificates. *IEEE Trans. Autom. Control.*, 52(8):1415–1428, 2007.

[PR89] Amir Pnueli and Roni Rosner. On the synthesis of a reactive module. In *Conference Record of the Sixteenth Annual ACM Symposium on Principles of Programming Languages, Austin, Texas, USA, January 11-13, 1989*, pages 179–190. ACM Press, 1989.

[PR04a] Andreas Podelski and Andrey Rybalchenko. A complete method for the synthesis of linear ranking functions. In *Verification, Model Checking, and Abstract Interpretation, 5th International Conference, VMCAI 2004, Venice, Italy, January 11-13, 2004, Proceedings*, pages 239–251, 2004.

[PR04b] Andreas Podelski and Andrey Rybalchenko. Transition invariants. In *19th IEEE Symposium on Logic in Computer Science (LICS 2004), 14-17 July 2004, Turku, Finland, Proceedings*, pages 32–41, 2004.

[Put93] Mihai Putinar. Positive polynomials on compact semi-algebraic sets. *Indiana University Mathematics Journal*, 42(3):969–984, 1993.

[Put94] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley Series in Probability and Statistics. Wiley, 1994.

[RBK18] Spencer M. Richards, Felix Berkenkamp, and Andreas Krause. The lyapunov neural network: Adaptive stability certification for safe learning of dynamical systems. In *2nd Annual Conference on Robot Learning, CoRL 2018, Zürich, Switzerland, 29-31 October 2018, Proceedings*, volume 87 of *Proceedings of Machine Learning Research*, pages 466–476. PMLR, 2018.

[Ric53] Henry Gordon Rice. Classes of recursively enumerable sets and their decision problems. *Transactions of the American Mathematical society*, 74(2):358–366, 1953.

[RK04] Enric Rodríguez-Carbonell and Deepak Kapur. Automatic generation of polynomial loop. In *Symbolic and Algebraic Computation, International Symposium ISSAC 2004, Santander, Spain, July 4-7, 2004, Proceedings*, pages 266–273, 2004.

[RK07] Enric Rodríguez-Carbonell and Deepak Kapur. Automatic generation of polynomial invariants of bounded degree using abstract interpretation. *Sci. Comput. Program.*, 64(1):54–75, 2007.

[RS19] Hadi Ravanbakhsh and Sriram Sankaranarayanan. Learning control lyapunov functions from counterexamples and demonstrations. *Auton. Robots*, 43(2):275–307, 2019.

[SB18] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[SCG13] Sriram Sankaranarayanan, Aleksandar Chakarov, and Sumit Gulwani. Static analysis for probabilistic programs: inferring whole program properties from finitely many paths. In Hans-Juergen Boehm and Cormac Flanagan, editors, *ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI '13, Seattle, WA, USA, June 16-19, 2013*, pages 447–458. ACM, 2013.

[SDC21] Cesar Santoyo, Maxence Dutreix, and Samuel Coogan. A barrier function approach to finite-time stochastic system verification and control. *Autom.*, 125:109439, 2021.

[SG91] Kirack Sohn and Allen Van Gelder. Termination Detection in Logic Programs using Argument Sizes. In *Proceedings of the Tenth ACM*

SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, May 29-31, 1991, Denver, Colorado, USA, pages 216–226, 1991.

[SGA15] Sadegh Esmaeil Zadeh Soudjani, Caspar Gevaerts, and Alessandro Abate. FAUST ^{2} : Formal abstractions of uncountable-state stochastic processes. In *Tools and Algorithms for the Construction and Analysis of Systems - 21st International Conference, TACAS 2015, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2015, London, UK, April 11-18, 2015. Proceedings*, volume 9035 of *Lecture Notes in Computer Science*, pages 272–286. Springer, 2015.

[SL10] Sean Summers and John Lygeros. Verification of discrete time stochastic hybrid systems: A stochastic reach-avoid decision problem. *Autom.*, 46(12):1951–1961, 2010.

[SST14] Aaron Stump, Geoff Sutcliffe, and Cesare Tinelli. Starexec: A cross-community infrastructure for logic solving. In *Automated Reasoning - 7th International Joint Conference, IJCAR 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 19-22, 2014. Proceedings*, pages 367–373, 2014.

[ST12] Jacob Steinhardt and Russ Tedrake. Finite-time regional verification of stochastic non-linear systems. *Int. J. Robotics Res.*, 31(7):901–923, 2012.

[STB+06] Armando Solar-Lezama, Liviu Tancau, Rastislav Bodík, Sanjit A. Seshia, and Vijay A. Saraswat. Combinatorial sketching for finite programs. In John Paul Shen and Margaret Martonosi, editors, *Proceedings of the 12th International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS 2006, San Jose, CA, USA, October 21-25, 2006*, pages 404–415. ACM, 2006.

[SWD+17] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[SZS+14] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[Tab09] Paulo Tabuada. *Verification and Control of Hybrid Systems - A Symbolic Approach*. Springer, 2009.

[Tar85] Robert Endre Tarjan. Amortized computational complexity. *SIAM Journal on Algebraic Discrete Methods*, 6(2):306–318, 1985.

[TBK19] Matteo Turchetta, Felix Berkenkamp, and A. Krause. Safe exploration for interactive machine learning. In *NeurIPS*, 2019.

[TKD+16] Dustin Tran, Alp Kucukelbir, Adji B. Dieng, Maja Rudolph, Dawen Liang, and David M. Blei. Edward: A library for probabilistic modeling, inference, and criticism. *arXiv preprint arXiv:1610.09787*, 2016.

[TMKA17] Ilya Tkachev, Alexandru Mereacre, Joost-Pieter Katoen, and Alessandro Abate. Quantitative model-checking of controlled discrete-time markov processes. *Inf. Comput.*, 253:1–35, 2017.

[TOUH18] Toru Takisaka, Yuichiro Oyabu, Natsuki Urabe, and Ichiro Hasuo. Ranking and repulsing supermartingales for approximating reachability. *CoRR*, abs/1805.10749, 2018.

[TOUH21] Toru Takisaka, Yuichiro Oyabu, Natsuki Urabe, and Ichiro Hasuo. Ranking and repulsing supermartingales for reachability in randomized programs. *ACM Trans. Program. Lang. Syst.*, 43(2):5:1–5:46, 2021.

[TvdMYW16] David Tolpin, Jan-Willem van de Meent, Hongseok Yang, and Frank D. Wood. Design and implementation of probabilistic programming language anglican. In Tom Schrijvers, editor, *Proceedings of the 28th Symposium on the Implementation and Application of Functional Programming Languages, IFL 2016, Leuven, Belgium, August 31 - September 2, 2016*, pages 6:1–6:12. ACM, 2016.

[TYL+20] Hoang-Dung Tran, Xiaodong Yang, Diego Manzanas Lopez, Patrick Musau, Luan Viet Nguyen, Weiming Xiang, Stanley Bak, and Taylor T. Johnson. NNV: the neural network verification tool for deep neural networks and learning-enabled cyber-physical systems. In Shuvendu K. Lahiri and Chao Wang, editors, *Computer Aided Verification - 32nd International Conference, CAV 2020, Los Angeles, CA, USA, July 21-24, 2020, Proceedings, Part I*, volume 12224 of *Lecture Notes in Computer Science*, pages 3–17. Springer, 2020.

[UD07] Eiji Uchibe and Kenji Doya. Constrained reinforcement learning from intrinsic and extrinsic rewards. In *2007 IEEE 6th International Conference on Development and Learning*, pages 163–168. IEEE, 2007.

[UGK16] Caterina Urban, Arie Gurfinkel, and Temesghen Kahsai. Synthesizing ranking functions from bits and pieces. In *Tools and Algorithms for the*

Construction and Analysis of Systems - 22nd International Conference, TACAS 2016, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2016, Eindhoven, The Netherlands, April 2-8, 2016, Proceedings, pages 54–70, 2016.

[UH17] Jonas Umlauft and Sandra Hirche. Learning stable stochastic nonlinear dynamical systems. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 3502–3510. PMLR, 2017.

[Vai15] Umesh Vaidya. Stochastic stability analysis of discrete-time system using lyapunov measure. In *American Control Conference, ACC 2015, Chicago, IL, USA, July 1-3, 2015*, pages 4646–4651. IEEE, 2015.

[vdMPYW18] Jan-Willem van de Meent, Brooks Paige, Hongseok Yang, and Frank Wood. An introduction to probabilistic programming. *CoRR*, abs/1809.10756, 2018.

[VE03] Ardalan Vahidi and Azim Eskandarian. Research advances in intelligent collision avoidance and adaptive cruise control. *IEEE Trans. Intell. Transp. Syst.*, 4(3):143–153, 2003.

[VGO19] Abraham P. Vinod, Joseph D. Gleason, and Meeko M. K. Oishi. Sreach-tools: a MATLAB stochastic reachability toolbox. In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control, HSCC 2019, Montreal, QC, Canada, April 16-18, 2019*, pages 33–38. ACM, 2019.

[VR08] Helga Velroyen and Philipp Rümmer. Non-termination checking for imperative programs. In *Tests and Proofs, Second International Conference, TAP 2008, Prato, Italy, April 9-11, 2008. Proceedings*, pages 154–170, 2008.

[WFC+20] Peixin Wang, Hongfei Fu, Krishnendu Chatterjee, Yuxin Deng, and Ming Xu. Proving expected sensitivity of probabilistic programs with randomized variable-dependent termination time. *Proc. ACM Program. Lang.*, 4(POPL):25:1–25:30, 2020.

[WFG+19] Peixin Wang, Hongfei Fu, Amir Kafshdar Goharshady, Krishnendu Chatterjee, Xudong Qin, and Wenjun Shi. Cost analysis of nondeterministic probabilistic programs. In *PLDI 2019*, pages 204–220, 2019.

[Wil91] David Williams. *Probability with Martingales*. Cambridge mathematical textbooks. Cambridge University Press, 1991.

[WLPK20] Matthew Wicker, Luca Laurenti, Andrea Patane, and Marta Kwiatkowska. Probabilistic safety for bayesian neural networks. In Ryan P. Adams and Vibhav Gogate, editors, *Proceedings of the Thirty-Sixth Conference on Uncertainty in Artificial Intelligence, UAI 2020, virtual online, August 3-6, 2020*, volume 124 of *Proceedings of Machine Learning Research*, pages 1198–1207. AUAI Press, 2020.

[WSF+21] Jinyi Wang, Yican Sun, Hongfei Fu, Krishnendu Chatterjee, and Amir Kafshdar Goharshady. Quantitative analysis of assertion violations in probabilistic programs. In Stephen N. Freund and Eran Yahav, editors, *PLDI '21: 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation, Virtual Event, Canada, June 20-25, 2021*, pages 1171–1186. ACM, 2021.

[WZ18] Kim Peter Wabersich and Melanie N. Zeilinger. Linear model predictive safety certification for learning-based control. In *57th IEEE Conference on Decision and Control, CDC 2018, Miami, FL, USA, December 17-19, 2018*, pages 7130–7135. IEEE, 2018.

[XLZF21] Bai Xue, Renjue Li, Naijun Zhan, and Martin Fränzle. Reach-avoid analysis for stochastic discrete-time systems. In *2021 American Control Conference, ACC 2021, New Orleans, LA, USA, May 25-28, 2021*, pages 4879–4885. IEEE, 2021.

[ZCBR22] Dorde Zikelic, Bor-Yuh Evan Chang, Pauline Bolignano, and Franco Raimondi. Differential cost analysis with simultaneous potentials and anti-potentials. In *43rd ACM SIGPLAN International Conference on Programming Language Design and Implementation, PLDI*, 2022.

[ZLCH22] Dorde Zikelic, Mathias Lechner, Krishnendu Chatterjee, and Thomas A. Henzinger. Learning stabilizing policies in stochastic control systems. In *ICLR 2022 Workshop on Socially Responsible Machine Learning SRML*, 2022.

[ZLHC22] Dorde Zikelic, Mathias Lechner, Thomas A. Henzinger, and Krishnendu Chatterjee. Learning control policies for stochastic systems with reach-avoid guarantees. *CoRR*, abs/2210.05308, 2022.

[ZLHC23] Dorde Zikelic, Mathias Lechner, Thomas A. Henzinger, and Krishnendu Chatterjee. Learning control policies for stochastic systems with reach-

avoid guarantees. In *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI*, 2023.