Project no.: PIRSES-GA-2011-295261

Project full title: Mobility between Europe and Argentina applying Logics to Systems

Project Acronym: MEALS

Deliverable no.: 1.5 / 2

Title of Deliverable: Probabilistic Termination: Soundness, Completeness, and Compositionality

Contractual Date of Delivery to the CEC:	**30-Sep-2015**
Actual Date of Delivery to the CEC:	**30-Sep-2015**
Organisation name of lead contractor for this deliverable:	**UNC**
Author(s):	**Luis María Ferrer Fioriti, Holger Hermanns**
Participants(s):	**SAU**
Work package contributing to the deliverable:	**WP1**
Nature:	**R**
Dissemination Level:	**Public**
Total number of pages:	**33**
Start date of project:	1 Oct. 2011 **Duration:** 48 month

Abstract:

We propose a framework to prove almost sure termination for probabilistic programs with real valued variables. It is based on ranking supermartingales, a notion analogous to ranking functions on non-probabilistic programs. The framework is proven sound and complete for a meaningful class of programs involving randomization and bounded nondeterminism. We complement this foundational insight by a practical proof methodology, based on sound conditions that enable compositional reasoning and are amenable to a direct implementation using modern theorem provers. This is integrated in a small dependent type system, to overcome the problem that lexicographic ranking functions fail when combined with randomization. Among others, this compositional methodology enables the verification of probabilistic programs *outside* the complete class that admits ranking supermartingales.

Note:

This deliverable is has been published in *POPL'15, Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, Pages 489-501, ACM, 2015.

This project has received funding from the European Union Seventh Framework Programme (FP7 2007-2013) under Grant Agreement Nr. 295261.

Contents

1 Introduction

Termination is one of the simplest and yet most important liveness properties. If the program involves randomization, or is operating in a randomized environment, the natural analogon of termination is *almost sure termination*, saying that the probability of eventually terminating is one, so divergence has zero probability.

Proving termination of non-probabilistic programs is equivalent to constructing a *ranking function* [14]. That is a mapping $v : S \to D$ such that $v(s) \succ v(s')$ for all transitions $s \to s'$ in the program, where $(D, \succ)$ is well-founded (a partial order without infinite descending chains). In case of programs over countable state spaces and bounded branching, one can restrict to $(\mathbb{N}, >)$ as well-founded relation. Having a numeric domain allows one to synthesize ranking functions via the solution of convex optimization problems [34, 9, 33, 5]. In addition, the value $v(s)$ provides an upper bound on the maximum number of steps before termination if assuming the program to start at state s.

For probabilistic programs, ranking functions provide a sound method to prove termination, but are far from being complete. To illustrate this, consider the following probabilistic program:

while $n > 0$ **do**
  $c := \mathit{uniform}(0, 1)$
  if $c < 0.5$ **then**
    $n := n - 1$
  fi
od

Clearly, the program terminates with probability one, since ignoring the decrementing command forever has probability zero. Yet, there cannot exist a ranking function proving termination, because each time the conditional branch is skipped the program state remains unchanged, but well-founded relations are anti-reflexive. However, with probability $\frac{1}{2}$ the next value of n will be decremented to $n - 1$, and with the same probability it will be n. This means that the expected value of variable n after each iteration is decreased by 0.5.

Bournez and Garnier [3] were the first to connect probabilistic program termination to *Lyapunov ranking functions* [15] (or Foster-Lyapunov functions). These constitute the analogon of ranking functions for probabilistic programs, by considering expected values instead of values: They allow the value of some states to increase when executing some transitions, as long as the expected value (i.e. its average) decreases. Formally a mapping $v : S \to \mathbb{R}^+$ is a Lyapunov ranking functions if there exists a constant $\varepsilon > 0$, such that:

$$v(s) \geq \sum_{s' \in S} \mathbb{P}(s, \alpha, s') v(s') + \varepsilon \tag{1}$$

whenever α is a program instruction enabled at state s, and $\mathbb{P}(s, \alpha, s')$ is the probability of going from s to s' using the command a. Notably, $(\mathbb{R}^+, \succ)$ is a well-founded relation on the positive reals, where $a \succ b \equiv a > b + \varepsilon$. Lyapunov ranking functions provide a sound method to prove almost sure (a.s.) termination, and in analogy to ranking functions, $v(s)/\varepsilon$ is an upper bound on

$n, i, c := 0, 0, 0$
while $c < 0.5$ **do**
  if $*$ **then**
    $c := uniform(0, 1)$
    if $c < 0.5$ **then** $n := n + 1$ **else** $i := g(n)$ **fi**
  else
    $c, i := 1, f(n)$
  fi
od

while $0 < i$ **do** $i := i - 1$ **od**

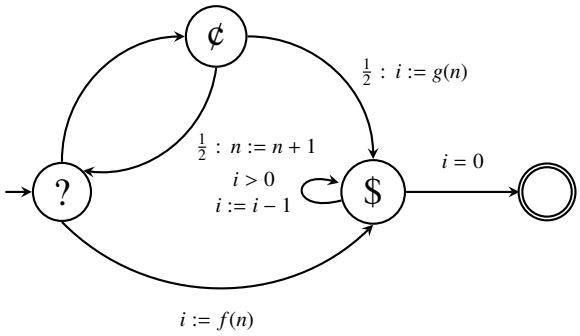

Figure 1: Quiz show example.

the expected number of steps before termination [15]. Thus, in fact, Lyapunov ranking functions guarantee a stronger property, namely, that the expected time to termination is finite. This is sometimes referred to as *positive almost sure termination* [3]. For probabilistic programs with countable state spaces, Lyapunov ranking functions not only are a sound method to prove positive almost sure termination *but* a complete one – but only as long as non-determinism is absent. We overcome this foundational restriction with this paper and embed the resulting theory into a practical, compositional proof method. When dealing with nondeterminism and probabilism, families of schedulers play an important role. As we will see, one of the distinct technical aspects of our approach is that the *semantic space* of a probabilistic program is set up in such a way that all possible schedulers share a common probability space. This enables correct reasoning about measurability of the quantities involved. In our setting the base quantities are ranking supermartingales, which in turn are a generalization of Lyapunov ranking functions. The main contributions of the paper are:

- We present a new semantics for non-deterministic programs where schedulers share a common probability space. (Section 4)
- We demonstrate that the Lyapunov ranking functions proposed by Bournez and Garnier [3] are *not* complete for positive almost sure termination in the non-deterministic case, and the same holds for ranking supermartingales. (Section 2)
- We extend the ranking supermartingale approach of Chakarov and Sankaranarayanan [7] to the bounded non-deterministic case, arriving at a sound and complete characterization of terminating programs. (Section 5)
- We provide a compositional and sound proof method for almost sure termination. The framework only relies on the synthesis of probabilistic linear loop invariants [7, 19]. (Sections 6, and 7)
- We provide a method to identify program variables that must have finite expectation. (Theorem 7.6)

2 Motivating Counter-Examples

Lyapunov ranking functions provide the apparatus to bound the expected number of steps a probabilistic program will take. As such, they are sound and complete for positive almost sure termination provided the probabilistic program is deterministic. In the presence of (bounded) non-determinism the notion appears to have a natural extension, obtained by requiring that the expected number of steps needed is finite for all possible resolutions of the non-determinism. However, as we illustrate below, and in contrast to results that were undisputed for a decade [3, Theorem 3], the completeness results for (positive) almost sure termination do not extend to the case of non-deterministic programs. Moreover, we show below that Lyapunov ranking functions are still incomplete even if besides all expectations also the supremum of all expectations is known to be finite.

A simple quiz show. Consider the program illustrated in Figure 1. It represents a quiz show, where after the quiz and prior to termination (state \$) the gained money is spent one unit at a time. A participant can initially (state ?) either answer a yes/no question or decide to stop the game immediately. If the participant decides to finish, he obtains a prize $f(n)$, where n is the number of questions answered thus far. If the participant answers correctly he can continue playing. In case of failing a question, he loses the game immediately and receives a consolation prize $g(n)$. For the participant in question, the quiz is so difficult that he realizes that the best strategy is to answer at random (state ¢). Analyzing this game, we first observe that the participant will receive (with probability one) a finite amount of money, because the event of answering questions correctly forever has probability zero. We now analyze the expected amount of money that the participant will receive. This depends on the number n of questions the participant is willing to answer before stopping, and of course on the choice of f and g:

$$E_n = \frac{f(n)}{2^n} + \sum_{1 \le k \le n} \frac{g(k-1)}{2^k} \qquad E_\infty = \sum_{1 \le k} \frac{g(k-1)}{2^k}$$

The quantity E_∞ corresponds to the expected gain if the participant never stops answering questions. Thus, he always gets the consolation prize.

$\forall_n E_n < \infty$ does not work. In a scenario where $f(n) = 3^n$, and $g(n) = n$, we see that clearly $E_n < E_{n+1} < \infty$ and $E_n \to \infty$ as $n \to \infty$. So it seems that the best strategy is to never give up. However, $E_\infty = 1$. This is because the limit of the expectation is *not* the expectation of the limit. In other words, there is *no* optimal strategy. Despite being able to find a Lyapunov ranking function for each particular resolution of the non-determinism (as all E_n are finite), we are unable to build a single, universal Lyapunov ranking function. It would require assigning ∞ to some of the states.

$\sup_n E_n < \infty$ neither. Consider now the case where $f(n) = 2^n$ and $g(n) = n$. Now we have $E_n \to 2$ as $n \to \infty$, so now the prize earned is bounded, and our participant will not get rich for

sure. In this case the aspect of interest is the expected money that the participant will receive under the assumption that he already answered m questions correctly:

$$E_{n,m} = \frac{2^n}{2^{n-m}} + \sum_{m+1 \le k \le n} \frac{k-1}{2^{k-m}} \quad \text{with } m \le n$$

Here, the maximum expectation after answering m questions turns out to be $2^m + m + 1$, obtained by $n \to \infty$. Yet, the maximum expectation after answering the $m+1$-th question (right or wrong) is $\frac{1}{2}\left(2^{m+1} + (m+1) + 1\right) + \frac{1}{2}m$, and that value is again $2^m + m + 1$. Therefore, the expectation remains invariant, i.e. a function assigning to each state the maximum expected termination time does not "rank" in expectation. And this function cannot be twisted to a ranking one without assigning ∞ to at least one of the states.

Cheating works! Now suppose that we, but not the participant, are able to record all results of the random choices and their effects prior to the show. Suppose also, that the results of the random choices are independent of the strategy of the participant about when to stop answering. We might for instance know that the fourth random choice will make the participant miss the fourth question, thus we are sure that he will get the consolation prize if he decides to answer at least four questions. We can then also infer that he will win the prize if he is prudent enough to withdraw before the fourth question. In other words, we know the maximum amount of money the participant is capable of winning *regardless* of his strategy. As this quantity is random we can analyze its expected value E^*. If $f(k+1) > f(k) > g(k)$, then:

$$E^* = \sum_{1 \le k} \frac{f(k)}{2^k}$$

In the above two examples, E^* is infinite because $f(k) \ge 2^k$. In Section 5 we show that a Lyapunov ranking function can only be constructed if and only if E^* is finite. The proper definition of E^* which implicitly ranges over all possible strategies, relies on a novel semantics for nondeterministic and probabilistic programs, introduced in Section 4.

3 Preliminaries

In this section we present some basic concepts of probability and measure theory, as needed for this paper. More detailed descriptions can be found in reference books such as Ash [1] and Williams [35].

Measure Theory. An arbitrary set Ω that contains all possible outcomes of an experiment is called a *sample space*. A *σ-algebra* $\mathcal{F}$ is a collection of subsets of Ω that is closed under complement and countable union, and $\Omega \in \mathcal{F}$. The elements of $\mathcal{F}$ are called *events*. We say that A is *$\mathcal{F}$-measurable* if $A \in \mathcal{F}$. In case $\Omega = \mathbb{R}$, the *Borel σ-algebra* denoted by $\mathcal{B}(\mathbb{R})$ is the σ-algebra generated by the class of open sets. A function $f : \Omega \to \Omega'$ is *measurable* with respect

to $\mathcal{F}$ and $\mathcal{F}'$ if $f^{-1}(B) \in \mathcal{F}$ for all $B \in \mathcal{F}'$. A *probability measure* μ is a function of signature $\mathcal{F} \to [0,1]$ such that $\mu(\Omega) = 1$, and it is countably additive: the probability of a countable disjoint union of events is given by the sum of the individual probabilities. An event A occurs *almost surely* (a.s.) if $\mu(A) = 1$. Similarly A is a *null event* if $\mu(A) = 0$. Two events A and B are *independent* when $\mu(A \cap B) = \mu(A)\mu(B)$. Two σ algebras $\mathcal{F}$ and $\mathcal{G}$ are independent if all events $A \in \mathcal{F}$ and $B \in \mathcal{G}$ are independent. Given an element $x \in \Omega$, δ_x is the *Dirac distribution* of x and satisfies $\delta_x(\{x\}) = 1$. The *conditional measure* of μ under a non-null event B is $\mu|_B(A) = \frac{\mu(A \cap B)}{\mu(B)}$ A *probability space* is a triple $(\Omega, \mathcal{F}, \mu)$, where $\mathcal{F}$ is a σ-algebra of Ω, and μ is a probability measure for $\mathcal{F}$.

Random Variables. A function $X : \Omega \to \mathbb{R}$ is a *random variable* (r.v.) on $(\Omega, \mathcal{F}, \mu)$, if it is $(\mathcal{F}, \mathcal{B}(\mathbb{R}))$-measurable. A r.v. X is *discrete* provided the range of X is countable. A particular discrete r.v. is the *indicator function* of a measurable set A. It takes values in $\{0, 1\}$, and satisfies $(\mathbf{1}_A(\omega) = 1) \equiv (\omega \in A)$. The *expected value* of a random variable X is the quantity:

$$\mathbb{E}(X) \triangleq \int_\Omega X d\mu$$

where $\int$ is the Lebesgue integral, see Ash [1] for a detailed explanation. In case of a discrete r.v., the expected value can be expressed as a weighted average. In particular, $\mathbb{E}(\mathbf{1}_A) = \mu(A)$. If $\mathbb{E}(|X|)$ is a finite value, we say X is *integrable*. It ensures that $\mathbb{E}(X\mathbf{1}_A)$ is finite for any measurable event A.

Filtrations. A finite or infinite sequence $\{\mathcal{F}_n\}$ of σ-algebras is a *filtration* if $\mathcal{F}_k \subseteq \mathcal{F}_{k+1} \subseteq \mathcal{F}$ for all $k \in \mathbb{N}$. For example consider the experiment of tossing two coins, i.e. the sample space Ω is $\{HH, TT, HT, TH\}$. A possible filtration is $\mathcal{F}_0 = \{\emptyset, \Omega\}$, $\mathcal{F}_1 = \{\emptyset, \{HH, HT\}, \{TT, TH\}, \Omega\}$, and $\mathcal{F}_2 = \mathcal{P}(\Omega)$. A sequence of random variables $\{X_n\}$ is *adapted* to a filtration $\{\mathcal{F}_n\}$ if X_n is $\mathcal{F}_n$-measurable for all n, and it is *predictable* if X_n is $\mathcal{F}_{n-1}$-measurable for all n.

Stopping Times. A discrete random variable T that takes values in $\mathbb{N} \cup \{\infty\}$ is a *stopping time* with respect to a filtration $\{\mathcal{F}_n\}$ if the event $\{T \leq n\}$ is $\mathcal{F}_n$-measurable. Since each element $\mathcal{F}_n$ of a filtration is a σ-algebra, then the events $\{T > n\}$ and $\{T = n\}$ are also $\mathcal{F}_n$-measurable. From the previous example a stopping time is the r.v. T_H that satisfy $T_H(H\_) = 1$, $T_H(TH) = 2$, and is infinity otherwise. We have $T_H^{-1}(0) = \emptyset \in \mathcal{F}_0$, $T_H^{-1}(1) = \{HH, HT\} \in \mathcal{F}_1$, and $T_H^{-1}(2) = \{TH\} \in \mathcal{F}_2$. Stopping times posses several properties that we will exploit in this paper. For example if T_1 and T_2 are stopping times so are $T_1 \wedge T_2$ (minimum), $T_1 \vee T_2$ (maximum), and $T_1 + T_2$. Also, any random variable that takes a constant value in $\mathbb{N}$ is trivially a stopping time. The expected value of a stopping time can be expressed in a very convenient form:

$$\mathbb{E}(T) = \sum_{1 \leq k} \mathbb{P}(T \geq k) \tag{2}$$

We can also define the σ-algebra $\mathcal{F}_T$ of events that occur up to time T as follows:

$$A \in \mathcal{F}_T \;\equiv\; \forall\, n :\; A \cap \{T \leq n\} \in \mathcal{F}_n. \tag{3}$$

Notice that T is measurable in $\mathcal{F}_T$. If S and T are stopping times with $S \leq T$, then $\mathcal{F}_S \subseteq \mathcal{F}_T$. Thus, sequences of increasing stopping times define a filtration.

Conditional Expectation. Given an integrable r.v. X on a probability space $(\Omega, \mathcal{F}, \mu)$, and $\mathcal{G}$ a sub-σ-algebra of $\mathcal{F}$, the *conditional expectation* $\mathbb{E}(X \mid \mathcal{G})$ is a r.v. measurable in $\mathcal{G}$ that satisfies:

$$\int_A X \, d\mu = \int_A \mathbb{E}(X \mid \mathcal{G}) \, d\mu \quad \text{for all } A \in \mathcal{G}$$

Intuitively $\mathbb{E}(X \mid \mathcal{G})$ is the expected outcome of a random experiment (represented by X), if some partial information of the result is available (represented by $\mathcal{G}$). Consider again the experiment of tossing two coins. The expected number of observed heads is one. However, if the result of one of the coins is already known, then the expected value changes. It would be 1.5 in case the revealed coin landed heads, and 0.5 otherwise. Some important properties about conditional expectations are the following:

$$\mathbb{E}(X \mid \{\emptyset, \Omega\}) = \mathbb{E}(X) \tag{4}$$

$$\mathbb{E}(X + Y \mid \mathcal{G}) = \mathbb{E}(X \mid \mathcal{G}) + \mathbb{E}(Y \mid \mathcal{G}) \tag{5}$$

$$\mathbb{E}(\mathbb{E}(X \mid \mathcal{H}) \mid \mathcal{G}) = \mathbb{E}(X \mid \mathcal{G}) \quad \text{if } \mathcal{G} \subset \mathcal{H} \tag{6}$$

$$\mathbb{E}(XY \mid \mathcal{G}) = X\mathbb{E}(Y \mid \mathcal{G}) \quad \text{if } X \text{ is } \mathcal{G}\text{-measurable} \tag{7}$$

Their intuitive meaning is as follows: (4) if no information is available it is the same as taking the expected value; (6) the conditional expectation depends on the maximum amount of available information; (7) the expectation depends only on unknown information.

Uniform Integrability. The concept of uniform integrability is an extension of integrability of a r.v. to a family of random variables, similar to the notion of uniform convergence in real analysis. A collection $\{X_n\}$ is *uniformly integrable* (UI), if the following holds:

$$\lim_{K \to \infty} \sup_n \mathbb{E}(|X_n| \mathbf{1}_{[|X_n| > K]}) = 0 \tag{8}$$

Informally, the expected values of a UI family are bounded and the probability mass is globally concentrated within this boundary, i.e. for any constant $\varepsilon > 0$, we can find a constant $\delta > 0$ such that $\mathbb{E}(|X_n| \mathbf{1}_A) \leq \varepsilon$ for all n, and all events A such that $\mu(A) \leq \delta$. Uniform integrability is a central concept in probability theory. It is a necessary and sufficient condition to guarantee that a sequence of r.v. converging in probabilities also converges in expectation, and it ensures that $\mathbb{E}(|X_n|)$ is finite, thus integrability. It will also play an important role in our findings.

4 Probabilistic Programs

In this section we present the syntax of our idealized probabilistic programs, and we provide a semantics in terms of random vectors over a probability space.

4.1 Syntax

Our programs will be composed by the following grammar:

⟨*statement*⟩ ::= ⟨*ident*⟩ ':=' ⟨*expr*⟩
| ⟨*ident*⟩ ':=' ⟨*distribution*⟩
| ' **if** ' ⟨*ndbexpr*⟩ ' **then** ' ⟨*statement*⟩ ' **else** ' ⟨*statement*⟩ ' **fi** '
| '**while** ' ⟨*bexpr*⟩ ' **do** ' ⟨*statement*⟩ ' **od** '
| ⟨*statement*⟩ ';' ⟨*statement*⟩

⟨*expr*⟩ ::= ⟨*constant*⟩ | ⟨*ident*⟩ | ⟨*expr*⟩ '+' ⟨*expr*⟩
| ⟨*expr*⟩ '*' ⟨*expr*⟩ | ⟨*expr*⟩ '-' ⟨*expr*⟩

⟨*bexpr*⟩ ::= ⟨*expr*⟩ '<' ⟨*expr*⟩ | ⟨*expr*⟩ '>' ⟨*expr*⟩
| ⟨*bexpr*⟩ '**and**' ⟨*bexpr*⟩ **|** ⟨*bexpr*⟩ **'or'** ⟨*bexpr*⟩

⟨*ndbexpr*⟩ **::= '*' |** ⟨*bexpr*⟩

Here, ⟨*distribution*⟩ stands for any probability distribution that is integrable, and has a known expected value, like uniform, exponential, normal, etc. The program **if** $*$ **then** P_1 **else** P_2 **fi** represents a non-deterministic choice between the statement P_1, and P_2. If P' is a statement contained inside a program P, we say P is the parent of P' and we denote it by $P' \triangleleft P$. $P' \trianglelefteq P$ is the reflexive version of $\triangleleft$. It is worth mentioning that probabilistic choices $P_1 \oplus_p P_2$ in the style of McIver and Morgan [30] can be encoded in our framework as $c := uniform(0, 1)$; **if** $c < p$ **then** P_1 **else** P_2 **fi**.

Given an arithmetic or boolean expression e, the set of variables that occur syntactically in e is denoted by $vars(e)$. The set of variables that are modified by a program P, i.e. all the program variables that appear on the left of assignment instructions, is denoted by $lvars(P)$.

4.2 Semantics

There are several ways to define semantics of probabilistic programs. Kozen [27] provides a denotational semantics in terms of continuous linear transformations of probability distributions. Given an initial distribution of the program variables, the semantics is a sub-probability distribution indicating the distribution of the program variables at termination time. The sub-distribution is a probability distribution provided the program a.s. terminates. Non-determinism is not considered in this semantics, but can be introduced using probabilistic power-domains [24]. On another route, probabilistic non-deterministic programs can be represented operationally as Markov Decision Processes [20]. The usual approach then is to define the sample space as the set of possible path executions, after non-determinism is resolved by means of a scheduler. The latter is basically a mapping from finite paths (determining the program state) to enabled commands. Each scheduler induces a dedicated probability measure on measurable paths. Therefore, the same program under two different schedulers yields two different probability spaces.

The approach we follow is closer to the latter but is equipped with a more refined semantics. The core innovation is that all schedulers are defined over the *same* probability space, still they

induce distinct random processes. This will allow us to break conceptual limitations of the earlier approaches. An event may then have distinct interpretations, according to the scheduler considered. Other events have a global meaning. These "global" events in turn will enable us to identify the set of programs where Lyapunov ranking functions provide a complete proof methodology.

Another feature of our semantics is that it does not have a built-in distinction between probabilistic and non-probabilistic commands. At each step a value is sampled from a distribution, but the executed command is free to use or to entirely ignore the sampled outcome.

Definition 4.1 (Guarded Command)**.** Given a finite set of real valued variables $Vars = \{X_1, \ldots, X_N\}$ a *guarded command* is a tuple $(l^i, G, \mathbf{f}, l^f)$ that consists of an initial location l^i, a final location l^f, a guard G that is a $\mathcal{B}(\mathbb{R})^N$-measurable set, an update vector $\mathbf{f} = (f_1, \ldots, f_N)$ of $(\mathcal{B}(\mathbb{R})^{N+1}, \mathcal{B}(\mathbb{R}))$-measurable functions.

A command is *enabled* if the program counter is l^i, and the variables satisfy G. After firing an enabled command, the program counter is set to l^f. The values of the program variables are modified according to the update vector. The value of X_k is determined by $f_k : \mathbb{R}^{N+1} \to \mathbb{R}$. The new valuation depends on the old values and a single sampled value at each step. The underlying probability space is assumed to ensure that the sampled value adheres to the right probability distribution.

Definition 4.2 (Probabilistic Program)**.** A *probabilistic program* is a tuple $(L, Vars, Cmds, l_0, l_\perp)$, where L is the program counter, $Vars = \{X_k\}_{k=1}^N$ is a finite set of real valued program variables, $Cmds = \{cmd_k\}_{k=1}^M$ is a finite set of guarded commands, l_0 is the initial program location, and $l_\perp$ is the location reached upon program termination. We require that for each location and variable valuation at least one command is enabled, except for the final location.

It is possible to have multiple commands enabled simultaneously, namely when several commands have the same initial location and their guards overlap. The resulting non-determinism in such programs is resolved by means of a *scheduler*. A realistic scheduler can at each step base its decision on the information currently available, including information accumulated about the past. In particular, a scheduler cannot decide to perform an action based on information about events that will occur in the future. Also, it must satisfy certain measurability conditions, so as to obtain a fully stochastic process when removing non-determinism. Before we dive into the formal definition of schedulers we first need to formalize the notion of "available information". This formalization will be provided by a filtration of the probability space of our program, and will be the same for *all* schedulers. This in turn will allow us to observe the behavior of all scheduler simultaneously.

Definition 4.3 (Semantic Space of Programs)**.** Let P be a probabilistic program with N variables. A *semantic space of* P is a tuple $(\Omega, \mathcal{F}, \mu, \{\mathcal{F}_n\}_{n=0}^\infty, \mathbf{X}_0, \{C_n\}_{n=1}^\infty)$, such that:

- $(\Omega, \mathcal{F}, \mu)$ is a probability space.

$$
\begin{aligned}
[\![X_k := E]\!] &= \{(l_0, \mathit{true}, (\pi_1, \ldots, \pi_{k-1}, E, \pi_{k+1}, \ldots, \pi_N), l_\perp)\} \\
[\![X_k := \langle \mathit{distribution} \rangle]\!] &= \{(l_0, \mathit{true}, (\pi_1, \ldots, \pi_{k-1}, \pi_{N+1}, \pi_{k+1}, \ldots, \pi_N), l_\perp)\} \\
[\![P_1; P_2]\!] &= [\![P_1]\!] \cup [\![P_2]\!] \\
[\![\textbf{if } G \textbf{ then } P_t \textbf{ else } P_f \textbf{ od }]\!] &= \{(l_0, G, (\pi_1, \ldots, \pi_N), l_0^{P_t}), (l_0, \overline{G}, (\pi_1, \ldots, \pi_N), l_0^{P_f})\} \cup [\![P_t]\!] \cup [\![P_f]\!] \\
&\quad \cup \{(l_\perp^{P_t}, \mathit{true}, (\pi_1, \ldots, \pi_N), l_\perp), (l_\perp^{P_f}, \mathit{true}, (\pi_1, \ldots, \pi_N), l_\perp)\} \\
[\![\textbf{if } * \textbf{ then } P_t \textbf{ else } P_f \textbf{ od }]\!] &= \{(l_0, \mathit{true}, (\pi_1, \ldots, \pi_N), l_0^{P_t}), (l_0, \mathit{true}, (\pi_1, \ldots, \pi_N), l_0^{P_f})\} \cup [\![P_t]\!] \cup [\![P_f]\!] \\
&\quad \cup \{(l_\perp^{P_t}, \mathit{true}, (\pi_1, \ldots, \pi_N), l_\perp), (l_\perp^{P_f}, \mathit{true}, (\pi_1, \ldots, \pi_N), l_\perp)\} \\
[\![\textbf{while } G \textbf{ do } P_W \textbf{ od }]\!] &= \{(l_0, G, (\pi_1, \ldots, \pi_N), l_0^{P_W}), (l_0, \overline{G}, (\pi_1, \ldots, \pi_N), l_\perp)\} \\
&\quad \cup [\![P_W]\!] \cup \{(l_\perp^{P_W}, \mathit{true}, (\pi_1, \ldots, \pi_N), l_0)\}
\end{aligned}
$$

Table 1: Semantics of probabilistic programs (all the program locations are assumed to be unique)

- $\{\mathcal{F}_n\}_{n=0}^{\infty}$ is a filtration of $\mathcal{F}$.
- $\mathbf{X}_0$ is a random vector of size N that is measurable in $\mathcal{F}_0$.
- $\{C_n\}_{n=1}^{\infty}$ is a sequence of identically distributed random variables, with C_n integrable, $\mathcal{F}_n$-measurable, and independent of $\mathcal{F}_{n-1}$.

$\mathbf{X}_0$ is the *initial valuation* of the program variables. C_n plays the role of a (possibly continuous-valued, finite-dimensional) coin, and represents a value sampled at step n.

This definition does not define a unique semantic space for a program, but instead specifies the minimum requirements to obtain a meaningful semantics. This generality is essential for the decomposition presented in Section 6. However, the most basic filtration with the desired properties is $\{\sigma(C_1, \ldots, C_n)\}_{n=1}^{\infty}$ where $\sigma(C_1, \ldots, C_n)$ is the smallest σ-algebra that makes $C_1, \ldots, C_n$ measurable.

Definition 4.3 implicitly assumes all sampling instructions of P, modeled by C_n, to have the same distribution. We impose this restriction for the sake of clarity of the exposition. It can be relinquished by fixing the uniform unit distribution for coins and reconstructing all other integrable distributions from there by inversion. But this forces us to explicitly deal with Polish spaces, which obfuscates our arguments with technicalities. All the proofs contained in this paper can be adapted to that general setting.

The filtration $\{\mathcal{F}_n\}$ faithfully characterizes the information available after executing the first n commands: the result of the initial distribution of the program variables (measurable in $\mathcal{F}_0 \subseteq \mathcal{F}_n$), and the sampled values of $C_1, \ldots, C_n$ (all measurable in $\mathcal{F}_n$). The independence requirement does not only guarantee C_n being independent of C_m, it also ensures that $\mathcal{F}_n$ does not leak any information about the future. That is C_n is sampled *precisely* at time n. To see this, take C_m with $m > n$. Independence implies $\mu(C_m \in [a, b) \cap A) = \mu(C_m \in [a, b))\mu(A)$ for all events $A \in \mathcal{F}_n$ and

all real intervals $[a, b)$. If the random variable C_m were $\mathcal{F}_n$-measurable we would be able to take $C_m \in [a, b)$ as A with $0 < \mu(C_m \in [a, b)) < 1$, rendering a contradiction in the above equality.

Definition 4.4 (Scheduler)**.** Given a semantic space $(\Omega, \mathcal{F}, \mu, \{\mathcal{F}_n\}, \mathbf{X}_0, \{C_n\})$, a *scheduler* ϕ is a collection $\{\phi_n\}_{n=1}^{\infty}$ of functions with signatures $\Omega \to \{1, \ldots, M\}$ such that ϕ_n is $\mathcal{F}_{n-1}$-measurable.

The measurability condition imposed on schedulers is crucial. The function ϕ_n selects the command to be executed at step n. The decision depends only on the initial state and the past observed samples, $\mathcal{F}_{n-1}$-measurable events. The value of the current and future samples, measurable in $\mathcal{F} \setminus \mathcal{F}_{n-1}$, have no influence on how the scheduler resolves the non-determinism present. Notice that the probability measure μ in the semantic space does *not* depend on the scheduler, thus different schedulers can be defined under the same probability space. This is the key feature of our semantics.

Schedulers that do incorporate randomization in their decisions can be modeled in our framework by augmenting the filtration in our semantic space. For example we can use $\{\sigma(U_0, C_1, U_1, \ldots C_n, U_n)\}$ with U_k uniform in the interval $(0, 1)$. The scheduler ϕ_n can use the result of U_{n-1} as it is $\mathcal{F}_{n-1}$ measurable.

Definition 4.5 (Operational Semantics)**.** Let $P = (L, Vars, Cmds, l_0, l_\perp)$ be a probabilistic program with $(\Omega, \mathcal{F}, \mu, \{\mathcal{F}_n\}, \mathbf{X}_0, \{C_n\})$ being its semantic space. The semantics of P under a scheduler $\phi = \{\phi_n\}$, is given by the following sequence of random vectors on the probabilistic space $(\Omega, \mathcal{F}, \mu)$ that is adapted to $\{\mathcal{F}_n\}$:

$$(L, \mathbf{X})_0^{\phi}(\omega) \triangleq (l_0, \mathbf{X}_0(\omega))$$
$$(L, \mathbf{X})_{n+1}^{\phi}(\omega) \triangleq (l^f_{k_{n+1}}, \mathbf{f}_{k_{n+1}}(\mathbf{X}_n^{\phi}(\omega), C_{n+1}(\omega)))$$

where $k_{n+1} = \phi_{n+1}(\omega)$, the command selected by the scheduler to be executed at time $n + 1$.

A scheduler is *valid* if every selected command is enabled in the respective state i.e., the following holds.

$$\phi_{n+1}(\omega) = k \quad \Rightarrow \quad L_n^{\phi} = l_k^i \wedge \mathbf{X}_n^{\phi} \in G_k \tag{9}$$

A valid scheduler ϕ is *memoryless* if it picks the next command to execute based solely on the current value of the program variables:

$$(L, \mathbf{X})_n^{\phi}(\omega) = (L, \mathbf{X})_m^{\phi}(\omega) \Rightarrow \phi_{n+1}(\omega) = \phi_{m+1}(\omega) \tag{10}$$

In the rest of the paper we only consider valid non-randomized memoryless schedulers. This restriction does not affect any of our results since this class is provably enough to prove almost sure termination [13]. However, it simplifies our proofs as ϕ can be viewed as a function $\phi : L \times \mathbb{R}^N \to \{1, \ldots M\}$.

The connection between our program syntax and guarded commands, i.e. the intuitive semantics of probabilistic programs is given in Table 1, where π_k is the projection of a tuple of size $N + 1$ on the k-th component and $^-$ denotes set complementation. (Note that the structure displayed on the right of figure 1 is a compressed version of what is obtained when applying this semantics to the program on its left.)

5 Almost Sure Termination

In this section we revise the notion of almost sure termination and its variants. For this, we introduce the notion of ranking supermartingales, which are a generalization of Lyapunov ranking functions. We prove that the synthesis of ranking supermartingales is a sound and complete proof method for positive almost sure termination for a restricted, yet useful, class of programs.

A program terminates if the time (i.e. number of steps) until the location $l_\perp$ is reached is finite. For this to hold, one has to prove that the program eventually reaches that location. We introduce two random variable T^ϕ and T^* that measure these times. The former indicates the time step when the program under scheduler ϕ reaches the final location $l_\perp$. The latter is the earliest time step for which it is guaranteed that *any* valid scheduler has already terminated. We stress that T^* can be defined in our setting just because all schedulers share a common probability space. The formal definition of these r.v. is as follows:

$$T^\phi(\omega) \triangleq \min\{n \,:\, L_n^\phi(\omega) = l_\perp\} \tag{11}$$

$$T^* \triangleq \sup_\phi T^\phi \tag{12}$$

It is not difficult to prove that T^ϕ is indeed a stopping time. More elaborated is the argument for T^*. The definition of T^* involves a supremum over an uncountable set of stopping times, and an uncountable number of operations over measurable sets does not necessarily yield a measurable object.

Lemma 5.1. *T^ϕ and T^* are stopping times.*

Proof. T^ϕ is a first arrival time, and is as such the typical example of stopping times [1, 35]. To see that T^* is also a stopping time it is enough to see that $\{T^* > n\} \in \mathcal{F}_n$. Take any sequence of length $c_1, \ldots, c_n$, such as $c_k \in \{1, \ldots, M\}$, and consider the scheduler $\phi^{c_1,\ldots,c_n} \triangleq \{\phi^{c_1,\ldots,c_n}{}_k\}_{k=1}^\infty$ where:

$$\phi^{c_1,\ldots,c_n}{}_k(\omega) \triangleq \begin{cases} c_k & k \leq n \text{ and } c_k \text{ enabled,} \\ i & \text{otherwise, with } i \text{ the minimum} \\ & \text{enabled command after } k \text{ steps.} \end{cases}$$

The definition for the case c_k not being enabled is to ensure that $\phi^{c_1,\ldots,c_n}{}_k$ is measurable. The command c_i is always well defined as each non terminal state has at least one enabled command. Any path that terminates in more than n steps, will be included in $\{T^{\phi^{c_1,\ldots,c_n}} > n\}$ for a proper sequence $c_1, \ldots, c_n$. The number of such sequences is exactly M^n, a finite quantity, and thus $\{T^* > n\} \equiv \bigcup\{T^{\phi^{c_1,\ldots,c_n}} > n\}$. □

Remark 5.2. The change in the program counter from step $n-1$ to step n depends only on the selected command, and not on the result of the probabilistic experiment performed at step n. Therefore, we can infer whether or not the program terminates at step n by inspecting the resolution of the scheduler at step n. That depends only on $\mathcal{F}_{n-1}$-measurable events. However, $\{T^\phi \leq n\}$ is still $\mathcal{F}_n$-measurable since $\mathcal{F}_{n-1} \subseteq \mathcal{F}_n$.

We are now in the position to define some important notions of termination in the presence of probability:

Definition 5.3 (Almost Sure Termination [3])**.** Given a program and its semantics $\{(L, \mathbf{X})_n^\phi\}$ for any valid schedulers ϕ the program is said to be *almost surely terminating* if for all ϕ, $\mathbb{P}(T^\phi < \infty) = 1$. If in addition $\mathbb{E}(T^\phi) < \infty$ for all valid schedulers, then it is said to be *positive almost surely terminating*.

Lyapunov ranking functions have been introduced to reason about programs involving discrete probability measures and countable state spaces. A Lyapunov ranking function v can be interpreted as a random variable V_n^ϕ as follows:

$$V_n^\phi(\omega) = v((L, \mathbf{X})_n^\phi(\omega)). \tag{13}$$

From this perspective, condition (1) implies $\{V_n^\phi\}$ is a discrete supermartingale [35]. Thus, the natural generalization of Lyapunov ranking functions on continuous distributions are supermartingales:

Definition 5.4 (Ranking Supermartingale)**.** A sequence of r.v. $\{Y_n\}$ adapted to $\{\mathcal{F}_n\}$ is a *supermartingale* if $\mathbb{E}(|Y_n|) < \infty$, and $\mathbb{E}(Y_{n+1} \,|\, \mathcal{F}_n) \leq Y_n$. In addition, it is a *ranking supermartingale* if $Y_n \geq 0$, and $\mathbb{E}(Y_{n+1} \,|\, \mathcal{F}_n) \leq Y_n - \varepsilon \mathbf{1}_{\{Y_n > 0\}}$ for some constant $\varepsilon > 0$.

This definition is a slight variation of ranking martingales from Chakarov and Sankaranarayanan [7], the difference being that we give an explicit lower bound. Thus, if $\mathbb{E}(Y_n^\phi) = 0$, then $Y_n^\phi = 0$ a.s. and therefore termination can be ensured. Another difference is a technicality, namely we do not require that the expectation always decreases. By $\mathbf{1}_{\{Y_n > 0\}}$ we require that only as long as the program is still running.

The connection of ranking supermartingales and positive almost sure termination is given by the following lemma. The result is similar to [3, 7] but we provide an alternative proof.

Lemma 5.5. *Let $\{Y_n\}$ be a ranking supermartingale with ε as required in the definition above, and $T(\omega) = \min_n \{n \,:\, Y_n(\omega) = 0\}$. Then, $\mathbb{P}(T < \infty) = 1$, and $\mathbb{E}(T) \leq \frac{\mathbb{E}(Y_0)}{\varepsilon}$.*

Proof. First, we prove by induction that $\mathbb{E}(Y_n) \leq \mathbb{E}(Y_0) - \varepsilon \sum_{0 \leq k < n} \mathbb{P}(Y_k > 0))$. The base case $n = 0$ is immediate since the sum ranges over an empty set. For the inductive case:

$$\begin{aligned}
& \mathbb{E}(Y_{n+1} \,|\, \mathcal{F}_n) \leq Y_n - \varepsilon \mathbf{1}_{\{Y_n > 0\}} \\
\Rightarrow \quad & \text{⦇ } X \leq Y \Rightarrow \mathbb{E}(X) \leq \mathbb{E}(Y) \text{ ⦈} \\
& \mathbb{E}(\mathbb{E}(Y_{n+1} \,|\, \mathcal{F}_n)) \leq \mathbb{E}(Y_n - \varepsilon \mathbf{1}_{\{Y_n > 0\}}) \\
\equiv \quad & \text{⦇ (4), (6), and linearity ⦈} \\
& \mathbb{E}(Y_{n+1}) \leq \mathbb{E}(Y_n) - \varepsilon \mathbb{E}(\mathbf{1}_{\{Y_n > 0\}}) \\
\equiv \quad & \text{⦇ Inductive hypothesis, } \mathbb{E}(\mathbf{1}_A) = \mathbb{P}(A) \text{ ⦈} \\
& \mathbb{E}(Y_{n+1}) \leq \mathbb{E}(Y_0) - \varepsilon \textstyle\sum_{0 \leq k < n} \mathbb{P}(Y_k > 0) - \varepsilon \mathbb{P}(Y_n > 0) \\
\equiv \quad & \text{⦇ Arithmetic ⦈} \\
& \mathbb{E}(Y_{n+1}) \leq \mathbb{E}(Y_0) - \varepsilon \textstyle\sum_{0 \leq k < n+1} \mathbb{P}(Y_k > 0)
\end{aligned}$$

Since $Y_n \geq 0$, and ε then $\sum_{0\leq k<n} \mathbb{P}(Y_k > 0) \leq \frac{\mathbb{E}(Y_0)}{\varepsilon}$ for all n. Therefore the series $\sum_{0\leq k} \mathbb{P}(Y_k > 0)$ converges. This implies that $\mathbb{P}(Y_n > 0) \to 0$ as $n \to \infty$, and hence in particular $\mathbb{P}(T < \infty) = 1$. It remains to prove that the expectation of T is bounded by $\mathbb{E}(Y_0)/\varepsilon$.

$$
\begin{array}{ll}
 & \mathbb{E}(T) \\
= & \quad \wr\ (2)\ \int \\
 & \sum_{1\leq k} \mathbb{P}(T \geq k) \\
= & \quad \wr\ \text{definition of } T\ \int \\
 & \sum_{1\leq k} \mathbb{P}(Y_k > 0) \\
\leq & \quad \wr\ \text{previous derivation, } \mathbb{P}(Y_0 \geq 0) \geq 0\ \int \\
 & \frac{\mathbb{E}(Y_0)}{\varepsilon}
\end{array}
$$

□

Theorem 5.6 (Soundness)**.** *Let P be a probabilistic program with semantics $\{(L, \mathbf{X})_n^\phi\}$, and a ranking supermartingale $\{Y_n\}$ such that $Y_n = 0$ implies $T^\phi \leq n$ for all valid schedulers ϕ. Then, P almost sure terminates, and T^* is integrable.*

Proof. This is a direct consequence of Lemma 5.5. □

The counterexample in Section 2 motivates the following incompleteness result.

Theorem 5.7 (Incompleteness)**.** *There exists a program P with T^ϕ integrable for all valid schedulers ϕ, and $\sup_\phi \mathbb{E}(T^\phi)$ finite such that there is no ranking supermartingale that guarantees P terminates.*

However, we can characterize the programs on which supermartingales can be used to prove termination:

Theorem 5.8 (Completeness)**.** *If T^* is integrable, then there exists a ranking supermartingale.*

Proof. We propose the random process $\{Y_n\}$ as a ranking supermartingale, where

$$Y_n \triangleq \mathbb{E}(T^* \,|\, \mathcal{F}_n) - T^* \wedge n \tag{14}$$

Notably, the conditional expectation is well defined since $0 \leq \mathbb{E}(T^*) < \infty$. To see that $\{Y_n\}$ is adapted to $\{\mathcal{F}_n\}$, we note that the first term is $\mathcal{F}_n$-measurable, by the definition of conditional expectation. The second term is $\mathcal{F}_n$-measurable since T^* is a stopping time and thus $\{T^* > n\} \in \mathcal{F}_n$. In fact for $n > 0$ the previous event is in $\mathcal{F}_{n-1}$. Lets prove that $\{Y_n\}$ is a ranking supermartingale:

$$
\begin{array}{ll}
 & \mathbb{E}(Y_{n+1} \,|\, \mathcal{F}_n) \\
= & \quad \wr\ \text{Definition of } Y_{n+1}\ \int \\
 & \mathbb{E}(\mathbb{E}(T^* \,|\, \mathcal{F}_{n+1}) - T^* \wedge (n+1) \,|\, \mathcal{F}_n) \\
= & \quad \wr\ \text{Linearity}\ \int
\end{array}
$$

$$
\begin{aligned}
& \mathbb{E}(\mathbb{E}(T^* \,|\, \mathcal{F}_{n+1}) \,|\, \mathcal{F}_n) - \mathbb{E}(T^* \wedge (n+1) \,|\, \mathcal{F}_n) \\
= \quad & \wr \text{ (6), (7), and } T^* \wedge (n+1) \text{ is } \mathcal{F}_n\text{-measurable} \int \\
& \mathbb{E}(T^* \,|\, \mathcal{F}_n) - T^* \wedge (n+1) \\
= \quad & \wr\ a \wedge (b+1) = a \wedge b + \mathbf{1}_{[a>b]} \int \\
& \mathbb{E}(T^* \,|\, \mathcal{F}_n) - T^* \wedge n - \mathbf{1}_{\{T^*>n\}} \\
= \quad & \wr \text{ (14) twice, and } a > b \equiv a > (a \wedge b) \int \\
& Y_n - \mathbf{1}_{\{Y_n>0\}}
\end{aligned}
$$

□

Theorem 5.8 is concise, elegant and seems constructive, but one needs to know about the terminating behavior beforehand to apply it, and still one might face that the synthesis of distributions is in general non-computable [18]. In the remainder of this paper we attack this problem by building a practical framework to synthesize ranking supermartingales compositionally.

6 Program Manipulation

We now turn to the question how to analyze properties of complex probabilistic programs. The general idea is to decompose them into ones that are smaller and easier to reason about. For example if our programs is a sequential composition of several subprograms, we would like to infer properties for each subprogram and then transfer the results to the original program. Kozen [27] showed that sequential programs correspond to function composition, conditional branching to conditional probabilities, and iterations to fixed points or to infinite summations. The results presented in this section echo the Kozen decomposition in the context of filter-based semantics. For a program P we have a semantic space $(\Omega, \mathcal{F}, \mu, \{\mathcal{F}_n\}, \mathbf{X}_0, \{C_n\})$ for the program semantics, and we would like to distill an adequate semantics for a subprogram P' of P. That is, we need to build a semantic space $(\Omega, \mathcal{F}, \mu, \{\mathcal{F}'_n\}, \mathbf{X}'_0, \{C'_n\})$ for P' in such a way that we can relate the semantics $\mathbf{X}$ of P with the semantics $\mathbf{X}'$ of P', in the probability space $(\Omega, \mathcal{F}, \mu)$.

It does not come as a surprise that the $\{\mathcal{F}'_n\}$ should correspond to the events that occur once l'_0 is reached in P, i.e. $\mathcal{F}'_n = \mathcal{F}_{T+n}$, where T is a stopping time such that $L^{\phi}_T = l'_0$. Similarly, the candidates for C'_n are also C_{T+n}, yet we must prove that they satisfy the conditions of Definition 4.3. This is only the case if l'_0 is a.s. reached.

Lemma 6.1. *Let T be an a.s. finite stopping time with respect to the filtration $\{\mathcal{F}_n\}$, and $\{C_n\}$ be as in Definition 4.3. Then, for all $n \geq 1$ C_{T+n} has the same distribution as $\{C_n\}$. Moreover, C_{T+n} is $\mathcal{F}_{T+n}$-measurable and independent of $\mathcal{F}_{T+n-1}$.*

Proof. The result follows from the fact that the event $\{T = \infty\}$ can be ignored. We first prove that C_{T+n} has the same distribution as $\{C_n\}$.

$$
\begin{aligned}
& \mathbb{P}(C_{T+n} \in A) \\
= \quad & \wr\ T \text{ a.s. finite} \int
\end{aligned}
$$

$\sum_{0\le k} \mathbb{P}(C_{T+n} \in A \wedge T = k)$
$=$ ⟅ substitution ⟆
$\sum_{0\le k} \mathbb{P}(C_{k+n} \in A \wedge T = k)$
$=$ ⟅ C_{k+n} independent of $\mathcal{F}_{k+n-1} \supseteq \mathcal{F}_k$ ⟆
$\sum_{0\le k} \mathbb{P}(C_{k+n} \in A)\mathbb{P}(T = k)$
$=$ ⟅ $\{C_n\}$ identically distributed ⟆
$c_A \sum_{0\le k} \mathbb{P}(T = k)$
$=$ ⟅ T a.s. finite ⟆
c_A

The proof of C_{T+n} being independent of $\mathcal{F}_{T+n-1}$ is similar and thus omitted. We now prove that C_{T+n} is $\mathcal{F}_{T+n}$-measurable.

$\{C_{T+n} \in A\} \cap \{T + n = m\} \in \mathcal{F}_m$
$\equiv$ ⟅ $A \cap B \equiv (A \cap B) \cap B$ ⟆
$\{C_m \in A\} \cup \{T = m - n\} \in \mathcal{F}_m$
$\equiv$ ⟅ C_m is $\mathcal{F}_m$-measurable, and $\mathcal{F}_{m-n} \subseteq \mathcal{F}_m$ ⟆
true

□

6.1 Sequential programs

Consider a sequential program P of the form $P_1; \ldots; P_M$. We can define the random variable $T^{k,\phi}$ as the time when the subprogram P_k is reached, and $\Delta T^{k,\phi}$ as the amount of time spent on it.

$$T^{k,\phi} \triangleq \min_n \{n \,:\, L_n^\phi = l_k^i\} \tag{15}$$

$$\Delta T^{k,\phi} \triangleq T^{k+1,\phi} - T^{k,\phi} \tag{16}$$

Being a first hitting time indicator, $T^{k,\phi}$ *is* a stopping time. The other random variable, $\Delta T^{k,\phi}$, is however not necessarily a stopping time, as it measures a relative amount of time. However, we can transform it into a stopping time by adjusting the time frame.

Lemma 6.2. *$\Delta T^{k,\phi}$ is a stopping time with respect to the filtration $\{\mathcal{F}_{T^{k,\phi}+n}\}$.*

Proof. We need to prove $\{\Delta T^{W,\phi} \le n\}$ is $\mathcal{F}_{T^{k,\phi}+n}$-measurable:

$\{\Delta T^{k+1,\phi} \le n\} \in \mathcal{F}_{T^{k,\phi}+n}$
$\equiv$ ⟅ definition of $\mathcal{F}_T$ ⟆
$\forall\, m \quad \{\Delta T^{k+1,\phi} \le n\} \cap \{T^{k,\phi} + n \le m\} \in \mathcal{F}_m$
$\equiv$ ⟅ (16) ⟆
$\forall\, m \quad \{T^{k+1,\phi} \le m\} \in \mathcal{F}_m$

$$\equiv \quad \wr\ T^{k+1,\phi} \text{ is a stopping time } \int$$
$$\mathit{true}$$

□

Another interesting property is that if an event ω is not terminating in P_k, i.e. $T^{k,\phi} < \infty$ and $T^{k+1,\phi} = \infty$, then $\Delta T^{k,\phi} = \infty$ and $\Delta T^{l,\phi} = 0$ for all $l > k$. Also $T^{k,\phi}$ can be reconstructed as $\sum_{l<k} \Delta T^{l,\phi}$.

Lemma 6.3. *Let $P_1; \ldots; P_M$ be a probabilistic program with semantic space $(\Omega, \mathcal{F}, \mu, \{\mathcal{F}_n\}, \mathbf{X}_0, \{C_n\})$, a memoryless scheduler ϕ and its associated semantics $\{(L, \mathbf{X})_n^\phi\}$, such that $T^{k,\phi}$ a.s. finite. Then, $(\Omega, \mathcal{F}, \mu', \{\mathcal{F}'_n\}, \mathbf{X}'_0, \{C'_n\})$ is a valid semantic space for program P_k, where:*

$$\mu' \triangleq \mu \qquad \mathbf{X}_0 \triangleq \mathbf{X}_{T^{k,\phi}}$$
$$\mathcal{F}'_n \triangleq \mathcal{F}_{T^{k,\phi}+n} \qquad C'_n \triangleq C_{T^{k,\phi}+n}$$

Moreover, both semantics are related as follows:

$$(L', \mathbf{X}')_n^\phi = (L, \mathbf{X})_{T^{k,\phi}+n}^\phi$$
$$T'^\phi = \Delta T^{k,\phi}$$

The identities in the previous Lemma are well defined because all random variables belong to the *same* probability space $(\Omega, \mathcal{F}, \mu)$.

6.2 Conditional Branching and Non-Determinism

For branching structures like **if** G **then** P_t **else** P_f **od** we define the following stopping times:

$$T^{t,\phi} \triangleq \min_n\{n \ :\ L^\phi = l_\perp^t\} \tag{17}$$
$$T^{f,\phi} \triangleq \min_n\{n \ :\ L^\phi = l_\perp^f\} \tag{18}$$

Since both branches cannot be taken simultaneously, the statement $T^{t,\phi} = \infty \vee T^{f,\phi} = \infty$ holds with *certainty*. In turn, we can reconstruct the time of termination of the original program as follows:

$$T^\phi = (T^{t,\phi} \wedge T^{f,\phi}) + 1$$

The second summand corresponds to the execution of the join transition from Table 1. One may be tempted to analyze P_t and P_f in $(\Omega, \mathcal{F}, \mu, \{\mathcal{F}_{n+1}\}, \mathbf{X}_1, \{C_{n+1}\})$. In fact, if both P_t and P_f terminate then the full program does, but the converse does not hold. For example a non-terminating P_t does not prohibit termination for initial valuations that do not satisfy the condition. Thus, we should "eliminate" such events from the semantics of P_t and P_f. To do so, we define the semantics of both branches in different probability spaces, but we maintain the same measurable space $(\Omega, \mathcal{F})$. Moreover, null events in one probability space are a.s. valid in the other one.

Lemma 6.4. *Let* **if** G **then** P_t **else** P_f **od** *be a probabilistic program with a semantic space* $(\Omega, \mathcal{F}, \mu, \{\mathcal{F}_n\}, \mathbf{X}_0, \{C_n\})$ *and* ϕ *a memoryless scheduler defining a semantics* $\{(L, \mathbf{X})_n^\phi\}$*. Then,* $(\Omega, \mathcal{F}, \mu^t, \{\mathcal{F}_n^t\}, \mathbf{X}_0^t, \{C_n^t\})$ *is a valid semantic space for* P_t*, with*

$$\mu^t \triangleq \mu|_{G_0} \qquad \mathbf{X}_0^t \triangleq \mathbf{X}_0$$
$$\mathcal{F}_n^t \triangleq \mathcal{F}_{n+1} \qquad C_n^t \triangleq C_{n+1}$$

and similarly for $(\Omega, \mathcal{F}, \mu^f, \{\mathcal{F}_n^f\}, \mathbf{X}_0^f, \{C_n^f\})$ *and* P_f*. where* $\mu^f \triangleq \mu|_{\overline{G}_0}$*. The semantics are related as follows:*

$$(L^t, \mathbf{X}^t)_n^\phi = (L, \mathbf{X})_{n+1}^\phi \qquad \text{in } (\Omega, \mathcal{F}, \mu^t)$$
$$(L^f, \mathbf{X}^f)_n^\phi = (L, \mathbf{X})_{n+1}^\phi \qquad \text{in } (\Omega, \mathcal{F}, \mu^f)$$
$$(L, \mathbf{X})_{n+1}^\phi = \mathbf{1}_{G_0}(L^t, \mathbf{X}^t)_n^\phi + \mathbf{1}_{\overline{G}_0}(L^f, \mathbf{X}^f)_n^\phi \qquad \text{in } (\Omega, \mathcal{F}, \mu)$$

Notice that $\mathbf{X}_n$, $\mathbf{X}_n^t$, and $\mathbf{X}_n^f$, are all random vectors in the probability spaces $(\Omega, \mathcal{F}, \mu)$, $(\Omega, \mathcal{F}, \mu^t)$, and $(\Omega, \mathcal{F}, \mu^f)$, since they are $(\mathcal{F}, \mathcal{B}(\mathbb{R}))$-measurable. The different probability measures just induce different stochastic behaviors, as characterized in the above lemma. Notice that since ϕ is memoryless we can use the same analysis for the non-deterministic case, because the non-deterministic condition '$*$' can be replaced by a measurable set.

6.3 While Loops

The last program construction that we have to decompose is the while loop $W \triangleq$ **while** G **do** P_W **od** . There are some important random variables that characterize W.

$$T_k^{W,\phi} \triangleq \min_n\{n \,:\, L_n^\phi = l_G \wedge \#_n\, l_G = k\} \wedge T^\phi \tag{19}$$
$$\Delta T_k^{W,\phi} \triangleq T_{k+1}^{W,\phi} - T_k^{W,\phi} \tag{20}$$
$$S^{W,\phi} \triangleq \#_\infty\, l_G \tag{21}$$

where $\#_n\, l(\omega) = \sum_{0\le k\le n} \mathbf{1}_{[L_k^\phi = l]}(\omega)$ is the number of times location l was reached in the first n steps. The random variable $T_k^{W,\phi}$ indicates the time when W starts the k-th iteration. In case W performs the loop body less than k times, we could consider $T_k^{W,\phi}$ to be infinite, but instead we consider the k-th iteration to take place immediately at the time W terminates. This technicality ensures that $T_k^{W,\phi} \le T_{k+1}^{W,\phi} \le T^\phi$, i.e. we have a partial order on the stopping times, with T^ϕ as a supremum. Similar to our handling of the sequential operator, $\Delta T_k^{W,\phi}$ indicates the time spent at the k-th iteration, with $\Delta T_k^{W,\phi} = 0$ in case the loop finishes or diverges prior to the kth iterations. The last random variable introduced above, $S^{W,\phi}$, counts the numbers of times the guard of the loop is evaluated before termination. Some important identities can be derived:

$$T_k^{W,\phi} = \sum_{l<k} \Delta T_l^{W,\phi} \tag{22}$$
$$T^\phi = \sum_{l\le S^{W,\phi}} \Delta T_l^{W,\phi} \tag{23}$$

Similar random variables are to be defined in case of nested while loops. For the present work we only need the analogon of $S^{W,\phi}$. Considering the case where $W \triangleleft W_m \triangleleft \ldots \triangleleft W_1$, and k_j indicates the current iteration of loop W_j, we define

$$S^{W,\phi}_{k_1,\ldots,k_m} \triangleq \sum_{0\leq n} \mathbf{1}_{[L^\phi_n = l_G \wedge \forall_{j\leq m} \#_n l_{G_k} = k_j]} \tag{24}$$

Notice that $S^{W,\phi}_{k_1,\ldots,k_m} = 0$ whenever one of the parent loops W_j terminated before k_j iterations.

Lemma 6.5. *Let W be a program of the form* **while** G **do** P_W **od** *equipped with a semantic space $(\Omega, \mathcal{F}, \mu, \{\mathcal{F}_n\}, \mathbf{X}_0, \{C_n\})$, and a memoryless scheduler ϕ that defines a semantics $\{(L, \mathbf{X})^\phi_n\}$. Then $(\Omega, \mathcal{F}, \mu^k, \{\mathcal{F}^k_n\}, \mathbf{X}^k_0, \{C^k_n\})$ is a valid semantic space for P_W where*

$$\mu^k \triangleq \mu|_{G^\phi_k} \qquad \mathbf{X}^k_0 \triangleq \mathbf{X}^\phi_{T^{W,\phi}_k}$$
$$\mathcal{F}^k_n \triangleq \mathcal{F}_{T^{W,\phi}_k+n+1} \qquad C^k_n \triangleq C_{T^{W,\phi}_k+n+1}$$

Then the following equalities a.s. hold, provided G^ϕ_k is not empty, and $T^{W,\phi}_k$ is a.s. finite.

$$\begin{aligned}
(L^k, \mathbf{X}^k)^\phi_n &= (L, \mathbf{X})^\phi_{T^{W,\phi}_k+n} && \text{in } (\Omega, \mathcal{F}, \mu^k)\\
\mathbf{X}^\phi_{T^{W,\phi}_{k+1}} &= \mathbf{1}_{G^\phi_k} \mathbf{X}^k_{T^k} + \mathbf{1}_{\overline{G}^\phi_k} \mathbf{X}^k_0 && \text{in } (\Omega, \mathcal{F}, \mu)\\
(L, \mathbf{X})^\phi_{T^{W,\phi}_k+n+1} &= \mathbf{1}_{G^\phi_k} (L^k, \mathbf{X}^k)^\phi_n \\
&\quad + \mathbf{1}_{\overline{G}^\phi_k} (L^k, \mathbf{X}^k)^\phi_{T^{W,\phi}_k} && \text{in } (\Omega, \mathcal{F}, \mu)
\end{aligned}$$

A (maybe) surprising result is that $S^{W,\phi}$ *is* a stopping time for a specific filtration.

Lemma 6.6. *The random variable $S^{W,\phi}$ is a stopping time with respect to the filtration $\{\mathcal{F}_{T^{W,\phi}_n}\}$.*

Proof. We need to prove $\{S^{W,\phi} \leq n\}$ is $\mathcal{F}_{T^{W,\phi}_n}$-measurable:

$$\begin{aligned}
& \{S^{W,\phi} \leq n\} \in \mathcal{F}_{T^{W,\phi}_n} \\
\equiv & \quad \langle\!| \text{ definition of } \mathcal{F}_T \text{ } |\!\rangle \\
& \forall\, m \quad \{S^{W,\phi} \leq n\} \cap \{T^{W,\phi}_n \leq m\} \in \mathcal{F}_m \\
\equiv & \quad \langle\!| \text{ } S^{W,\phi} \leq n \equiv T^{W,\phi}_n = T^{W,\phi} \text{ } |\!\rangle \\
& \forall\, m \quad \{T^{W,\phi} \leq m\} \in \mathcal{F}_m \\
\equiv & \quad \langle\!| \text{ } T^{W,\phi} \text{ is an stopping time } |\!\rangle \\
& \textit{true}
\end{aligned}$$

□

The above lemma turns out to be very important to prove termination. It suggests that we only have to look at the random process $\{(L, \mathbf{X})^\phi_{T^{W,\phi}_n}\}$ in order to prove termination of W.

7 Compositional Reasoning about Termination

In this section we present a compositional framework for almost sure termination that fully exploits the program structure. In general we are facing a loop $W = \textbf{while}\, G\, \textbf{do}\, P_1 \ldots P_N\, \textbf{od}$, where each P_k can be another loop. We aim to prove W is a.s. terminating by showing that each nested loop of W is a.s. terminating, together with an a.s. termination proof for a sliced, unnested version of W. We take great care to make the resulting framework apt for implementation.

7.1 A probabilistic lexicographic argument

If W does not involve any randomization, them we can apply a very simple compositional variant rule [14, 26] using Floyd-Hoare logic:

$$\text{V-RULE}\ \frac{\begin{array}{l}\forall\, k\ :\ P_k \text{ terminates and } \{\, R = z \,\}\, P_k\, \{\, R \leq z \,\} \\ \exists\, k\, \{\, R = z \,\}\, P_k\, \{\, R < z \,\} \qquad < \text{ is well-founded}\end{array}}{\textbf{while}\, G\, \textbf{do}\, P_1 \ldots P_N\, \textbf{od} \quad \text{terminates}}$$

This proof rule induces a lexicographic ranking function on well-structured programs. To apply this rule we need to prove the termination of each inner loop separately, and ensure that the variant argument R does not increase in any P_k. Additionally, we need to find at least one P_k where R indeed decreases. The inference rule V-RULE is sound but not complete as R can be increased in an inner loop as long as that gets compensated later. Despite its simplicity, it is a powerful inference rule, since it can be applied recursively until a simple while loop without nesting remains. Thus, one only needs a procedure to synthesize ranking functions for simple loops, combined with effective decision procedures to verify the non-increasing and strictly decreasing conditions.

Condition $\{\, R = z \,\}\, P_k\, \{\, R \leq z \,\}$ is basically another way of expressing $R' \leq R$ where R' is the value of the expression R after executing P_k. Such a predicate, involving primed and unprimed expressions, can be regarded as a *transition invariant* [32]. Also, predicates $R' \leq R$ and $R' < R$ correspond to conditions *Unaffecting* and *Ranking* in the work of Bradley, Manna and Sipma [5].

We aim for an extension of the V-RULE to the probabilistic case, implying the need to synthesize ranking supermartingales instead of ranking functions. We thus need to construct the probabilistic counterpart of transition invariants, or more precisely of conditions *Unaffecting* and *Ranking*. Since we are dealing with decreasing expectations, the natural way forward considers *probabilistic transition invariants* as relations between unprimed expressions and the conditional expectation of primed ones. This means requiring $\mathbb{E}(R' \,|\, R) \leq R$ for *Unaffecting*. An effective way of ensuring this is given by the following inductive definition:

$$\begin{aligned} \mathit{Dec}_{\leq}(E, X := e) &\triangleq \mathbb{E}(E[X/e] \,|\, \mathit{Vars}) \leq E \\ \mathit{Dec}_{\leq}(E, P_1; P_2) &\triangleq \mathit{Dec}_{\leq}(E, P_1) \wedge \mathit{Dec}_{\leq}(E, P_2) \\ \mathit{Dec}_{\leq}(E, \textbf{if}\ \ldots) &\triangleq \mathit{Dec}_{\leq}(E, P_t) \wedge \mathit{Dec}_{\leq}(E, P_f) \\ \mathit{Dec}_{\leq}(E, \textbf{while}\ \ldots) &\triangleq \mathit{Dec}_{\leq}(E, P_W) \end{aligned}$$

Notice that the base case corresponds to the weakest precondition of that simple assignment. In case that E is *propositionally linear*, i.e. an expression of the form $\mathbf{1}_{G_1}E_1 + \ldots + \mathbf{1}_{G_n}E_n$ where G_k is a conjunctive linear predicate, and E_k is a linear expression, the Motzkin transposition theorem can be applied to check $Dec_{\leq}(E, P)$ as pointed out by Katoen et al. [25]. If $Dec_{\leq}(E, P)$ is satisfied, it ensures that $\{E_n^{\phi}\}$ is a supermartingale irrespective of the scheduler ϕ, provided all E_k are integrable.

The probabilistic counterpart of the *Ranking* condition is similar:

$$\begin{aligned} Dec_{<}(E, X := e) &\triangleq \mathbb{E}(E[X/e] \,|\, Vars) \leq E - \varepsilon \\ Dec_{<}(E, P_1; P_2) &\triangleq (Dec_{<}(E, P_1) \wedge Dec_{\leq}(E, P_2)) \vee \\ &\qquad (Dec_{\leq}(E, P_1) \wedge Dec_{<}(E, P_2)) \\ Dec_{<}(E, \textbf{if } \ldots) &\triangleq Dec_{<}(E, P_t) \wedge Dec_{<}(E, P_f) \\ Dec_{<}(E, \textbf{while } \ldots) &\triangleq false \end{aligned}$$

where $\varepsilon > 0$, is existentially quantified. $Dec_{<}(E, P)$ guarantees that the expectation of E strictly decreases by the execution of P. Notably, E is not necessarily decreased after the execution of a loop, because the loop body might be executed zero times. The conditions $Dec_{\leq}$ and $Dec_{<}$ constitute an over approximation and can be weakened so as to provide a more precise but costly analysis.

Definition 7.1 (Compositional Ranking Supermartingale)**.** Given a loop **while** G **do** P_W **od** , a *compositional ranking supermartingale* R is a propositionally linear expression over the program variables such that $Dec_{\leq}(R, P_W)$ and $Dec_{<}(R, P_W)$.

7.2 A counter-example

Although it does appear plausible that with this probabilistic interpretation the V-RULE is sound in the context of probabilistic programs, this is *not* the case in general. Consider the following non-terminating program P with a nested loop P':

$X := 1;$
while $X > 0$ **do**
  $C := 0$
  while $C < 0.5$ **do**
    $C := uniform(0, 1);$
    if $C < 0.5$ **then** $X := 2(X - 1)$ **else** $X := 2$ **fi**
  od
  $X := X - 1$
od

The inner loop P' finishes once the sampled value is greater than 0.5, in that case the variable X is set to 2. The next instruction decrements the value of X to 1, and thus the guard of the outer loop remains satisfied. Suppose that we overlook that the program never terminates, and we instead want to find a compositional ranking supermartingale for P so as to prove a.s. termination. The

obvious choice would be X: It is decremented in every outer iteration, and the program finishes as soon as X is 0 in the guard of P. We only need to prove that, the value of X decreases in expectation after P' terminates, i.e. we aim to show

$$\mathbb{E}(X_T \,|\, \mathcal{F}_0) \leq X_0 \tag{25}$$

where X_0, and X_T are the values of X before and after executing P'. We want to prove the above by showing that the value of X does not increase in expectation after a single iteration:

$$\mathbb{E}(X_{n+1} \,|\, \mathcal{F}_n) \leq X_n \quad \text{for all } n \tag{26}$$

The latter is true because $\mathbb{E}(X_{n+1} \,|\, \mathcal{F}_n) = \frac{1}{2}\mathbb{E}(2(X_n - 1) \,|\, \mathcal{F}_n) + \frac{1}{2}\mathbb{E}(2 \,|\, \mathcal{F}_n) = X_n$. Effectively X is a compositional ranking supermartingale according to Definition 7.1. But, $\mathbb{E}(X_T \,|\, \mathcal{F}_0) = 2 > 1 = X_0$ since $X_T = 2$, i.e. condition (26) does not imply (25), while it should be implied if the probabilistic version of the V-RULE were sound. The root of this problem is that $Dec_{\leq}$ cannot be used to prove (25). In the following section we show under which circumstances $Dec_{\leq}$ can indeed be used safely.

7.3 Preserving probabilistic transition invariants

In this subsection we present sufficient conditions to guarantee that supermartingale properties are kept after almost sure termination. We also give stronger conditions that are relatively straightforward to automate. As a basis we use that the supermartingale properties are ensured at a stopping time whenever the underlying supermartingale is uniformly integrable:

Theorem 7.2 (Optional Sampling [1])**.** *Let $\{X_n\}$ be a supermartingale and $\{T_n\}$ a sequence of increasing stopping times, then $\{X_{T_n}\}$ is a supermartingale in $\{\mathcal{F}_{T_n}\}$ if either one of the following conditions holds:*

- *$T_n \leq c_n$ for all n where c_n are constants.*
- *$\{X_n\}$ is uniformly integrable.*

The first condition allows us to infer the supermartingale property on bounded terminating programs, for example without loops. For the general case however, we are left with the second condition, which in turn forces us to prove condition (8). Unfortunately, the latter is not readily mechanizable in contemporary theorem provers, and thus is an impediment for our efforts to arrive at a directly implementable framework. We therefore embark on making it more manageable.

Lemma 7.3 (UI of Dominated Sequences [35])**.** *Let $\{X_n\}$ be a random process, and Y be an integrable r.v. such that $|X_n| \leq Y$ for all n. Then the random process $\{X_n\}$ is uniformly integrable.*

Lemma 7.4. *Let $\{X_n\}$ a random process adapted to $\{\mathcal{F}_n\}$ such that X_n integrable and $\mathbb{E}(|X_{n+1} - X_n| \,|\, \mathcal{F}_n) \leq c$ for some fixed constant c, and let T an integrable stopping time. Then $\{X_{n\wedge T}\}$ is uniformly integrable.*

Proof. Let $D_n \triangleq |X_n - X_{n-1}|\mathbf{1}_{\{T \geq n\}}$, for all $0 < n$, then $|X_{n \wedge T}| \leq |X_0| + \sum_{1 \leq k} D_k$ for all n. Using Lemma 7.3, we only need to prove that the expectation of the summation is finite.

$$
\begin{array}{ll}
& \mathbb{E}(D_{n+1} \,|\, \mathcal{F}_n) \\
= & \quad \wr \text{ definition of } D_n \int \\
& \mathbb{E}(|X_{n+1} - X_n|\mathbf{1}_{\{T \geq n+1\}} \,|\, \mathcal{F}_n) \\
= & \quad \wr \{T > n\} \text{ is } \mathcal{F}_n\text{-measurable, and (7)} \int \\
& \mathbb{E}(|X_{n+1} - X_n| \,|\, \mathcal{F}_n)\mathbf{1}_{\{T \geq n+1\}} \\
\leq & \quad \wr \text{ hypothesis} \int \\
& c\mathbf{1}_{\{T \geq n+1\}}
\end{array}
$$

Applying (4) we obtain $\mathbb{E}(D_n) \leq c\mathbb{P}(T \geq n)$, then:

$$
\begin{array}{ll}
& \mathbb{E}(\sum_{1 \leq k} D_k) \\
= & \quad \wr \text{ linearity} \int \\
& \sum_{1 \leq k} \mathbb{E}(D_k) \\
\leq & \quad \wr \text{ previous derivation} \int \\
& c \sum_{1 \leq k} \mathbb{P}(T \geq k) \\
= & \quad \wr \text{ characterization of } \mathbb{E}(T) \int \\
& c\mathbb{E}(T) \\
< & \quad \wr \ T \text{ integrable} \int \\
& \infty
\end{array}
$$

□

As we use supermartingales to guarantee a.s. termination, requiring T integrable is not a impediment in our framework. The main implementation benefit of this lemma is that the condition $\mathbb{E}(|X_{n+1} - X_n| \,|\, \mathcal{F}_n) \leq c$ can be directly inferred from a program statement using normal static analysis or the following function that take into account the expected behavior:

$$
BM(x := e) \triangleq \begin{cases} Vars & \text{if } \mathbb{E}(|e - X| \,|\, Vars) \leq c \\ Vars \setminus \{x\} & \text{otherwise} \end{cases}
$$

$$
\begin{aligned}
BM(P_1; P_2) &\triangleq BM(P_1) \cap BM(P_2) \\
BM(\textbf{ if } G \textbf{ then } P_t \textbf{ else } P_f \textbf{ od }) &\triangleq BM(P_t) \cap BM(P_f) \\
BM(\textbf{while } G \textbf{ do } P_W \textbf{ od }) &\triangleq Vars \setminus lvars(S)
\end{aligned}
$$

As uniform integrable families preserve martingale properties in general, one can use Lemma 7.4 also as a means to prove that a program is *not* positive a.s. terminating: This boils down to finding a positive *submartingale* that is 0 only when the program terminates.

Corollary 7.5. *Let $\{Y_n\}$ a positive submartingale with respect to the filtration $\{\mathcal{F}_n\}$, i.e. $\mathbb{E}(Y_{n+1}\,|\,\mathcal{F}_n) \geq Y_n$ and $Y_n \geq 0$ for all n, such that $\mathbb{E}(|Y_{n+1} - Y_n|\,|\,\mathcal{F}_n) \leq c$. Then, the stopping time $T = \min_n\{n : Y_n = 0\}$ is not integrable unless $Y_0 = 0$.*

$$\text{BASE}\frac{}{\{\,\emptyset\,\}\,P\,\{\,\emptyset\,\}} \qquad \text{SEQ}\frac{\{\,A_1\,\}\,P_1\,\{\,B_1\,\} \quad \{\,A_2\,\}\,P_2\,\{\,B_2\,\} \quad A_2 \subseteq B_1}{\{\,A_1\,\}\,P_1; P_2\,\{\,B_2\,\}}$$

$$\text{MOD}\frac{\{\,A_1\,\}\,P\,\{\,B_1\,\} \quad A_1 \subseteq A_2 \quad B_1 \supseteq B_2}{\{\,A_2\,\}\,P\,\{\,B_2\,\}}$$

$$\text{ASSIG-L}\frac{e \text{ linear} \quad vars(e) \subseteq A}{\{\,A\,\}\,x := e\,\{\,A \cup \{x\}\,\}} \qquad \text{ASSIG-NL}\frac{e \text{ non-linear} \quad vars(e) \subseteq A}{\{\,A\,\}\,x := e\,\{\,A \setminus \{x\}\,\}}$$

$$\text{WHILE}\frac{\{\,A\,\}\,P\,\{\,B\,\} \quad A \subseteq B}{\{\,A\,\}\,\textbf{while}\,G\,\textbf{do}\,P\,\textbf{od}\,\{\,A \cap BM(P)\,\}} \qquad \text{IF}\frac{\{\,A_1\,\}\,P_1\,\{\,B_1\,\} \quad \{\,A_2\,\}\,P_2\,\{\,B_2\,\}}{\{\,A_1 \cup A_2\,\}\,\textbf{if}\,G\,\textbf{then}\,P_1\,\textbf{else}\,P_2\,\textbf{fi}\,\{\,B_1 \cap B_2\,\}}$$

Table 2: Inference Rules

7.4 Probabilistic V-RULE + typechecking ⇒ a.s. termination

The theory developed thus far revolves around probabilistic assertions on the program variables, and we discussed the basis of obtaining these assertions in a symbolic manner. However, the results are correct only if we guarantee that the program variables appearing in the assertions are integrable. This task is in itself not trivial. A program could induce an arbitrary distribution on the program variables, including some with for example infinite expectation. We know that all variables are of type $\mathbb{R}$, but we need means to infer which ones are integrable. We will attack this remaining problem now, by means of static typechecking with dependent types. To guarantee that the relevant expressions are integrable we again rely on uniform integrability as this implies bounded expectation.

We shall label each program instruction with two sets of program variables yielding the form $\{\,Y_1, \ldots, Y_n\,\}\,P\,\{\,Z_1, \ldots, Z_m\,\}$. Informally, it means that if variables $Y_1, \ldots, Y_n$ are integrable before executing P, then $Z_1, \ldots, Z_m$, will be integrable after P terminates. A program will be *well-labeled* if the labeling is obtained using the inference rules in Table 2. The rule for loops is based on the requirements of Lemma 7.4. The linear combination of integrable random variables is also integrable. A r.v. of the form $X * Y$ is not necessarily integrable even if both X and Y are. As an example, consider X that is $(\sqrt{2})^n$ with probability $\frac{1}{2^n}$, then $\mathbb{E}(X) = \frac{\sqrt{2}}{\sqrt{2}-1}$, but $\mathbb{E}(X^2) = \infty$. This restriction can be relaxed if a previous static analysis of the program shows that at least one the two variables is bounded. The rest of the rules are straightforward.

Theorem 7.6. *Given a well-labeled program $\{\,A\,\}\,P\,\{\,B\,\}$ with semantic space $(\Omega, \mathcal{F}, \mu, \{\mathcal{F}_n\}, \mathbf{X}_0, \{C_n\})$, and a valid scheduler ϕ such that:*

(i) $\mathbb{P}(T^\phi < \infty) = 1$,

(ii) Y_0^ϕ *integrable for all variables* $Y \in A$*, and*

(iii) $S^{W,\phi}_{k1,\ldots,k_m}$ *integrable for all while-loops* $W \trianglelefteq P$*.*

Then, $Z^\phi_{T^\phi}$ *is integrable for all* $Z \in B$*, and for all propositionally linear expressions* E *such that* $vars(E) \subseteq A_S \cap B_S$ *for all* $\{ A_S \}\, S\, \{ B_S \}$ *with* $S \trianglelefteq P$*:*

(a) $Dec_{\leq}(E, P) \Rightarrow \mathbb{E}(E^\phi_{T^\phi} \,|\, \mathcal{F}_0) \leq E^\phi_0$

(b) $Dec_{<}(E, P) \Rightarrow \mathbb{E}(E^\phi_{T^\phi} \,|\, \mathcal{F}_0) \leq E^\phi_0 - \varepsilon$*, for some fixed* $\varepsilon > 0$

(c) $E \subseteq BM(P) \Rightarrow \mathbb{E}(|E^\phi_{T^\phi} - E^\phi_0| \,|\, \mathcal{F}_0) \leq c$*, for some fixed* $c > 0$*.*

Proof. Structural induction on P for *any* semantic space. For the base case $\{ A \}\, X := e\, \{ B \}$ we only need to analyze the case when $X \in B$. The only rule from Table 2 that can be applied is ASSIGN-L. Thus, e is linear; $vars(e) \subseteq A$; Y^ϕ_0 integrable for all $Y \in vars(e)$; $X^\phi_{T^\phi} = e^\phi_0$ is integrable. Properties (a)-(c) follow immediately from the definition of $Dec_{\leq}$, $Dec_{<}$, and BM.

For the sequential composition $\{ A \}\, P_1\, \{ B \}\, P_2\, \{ C \}$ and using (i), we have $\Delta T^{1,\phi}$, and $\Delta T^{2,\phi}$ a.s. finite. Applying the induction hypothesis (IH) on $\{ A \}\, P_1\, \{ B \}$ using the same semantic space, we obtain $Y^\phi_{\Delta T^{1,\phi}}$ integrable for all $Y \in B$, and (a)-(c) for all propositionally linear expressions. We can apply IH on $\{ B \}\, P_2\, \{ C \}$ and the semantic space $(\Omega, \mathcal{F}, \mu, \{\mathcal{F}_{\Delta T^{1,\phi}+n}\}, \mathbf{X}_0, \{C_{\Delta T^{1,\phi}+n}\})$. Using Lemma 6.2 we obtain $Z^\phi_{T^\phi} = Z^\phi_{\Delta T^{1,\phi}+\Delta T^{2,\phi}}$ is integrable for all $Z \in C$. Properties (a)-(c) are again straightforward from the function definitions. The conditional branching case proceeds similar, but uses Lemma 6.4.

Let's prove the looping case $\{ A \}$ **while** G **do** P_W **od** $\{ B \}$. From (i) we can deduce $\Delta T^{W,\phi}_k$ a.s. finite. We also have that $\{ A \}\, P_W\, \{ A \}$ is well-labeled, thus we can apply IH with respect to the semantic space $(\Omega, \mathcal{F}, \mu^k, \{\mathcal{F}^k_n\}, \mathbf{X}^k_0, \{C^k_n\})$ as defined in Lemma 6.5. By induction on k, we obtain Y integrable at time $T^{W,\phi}_k$ for all $Y \in A$, and conditions (a)-(c) are satisfied at time $T^{W,\phi}_k$. If the program variable $Z \in B$ then $Z \in BM(P_W)$. In turn, we can apply Lemma 7.4 as $S^{W,\phi}$ is integrable, and obtain Z uniform integrable when sampled at times $\{T^{W,\phi}_k\}$. Thus, Z is integrable when the program terminates. Condition (a) follows from Lemma 7.4 and Theorem 7.2. Conditions (b) and (c) are trivial since $Dec_{<}(E, W) = false$, and $BM(W) \cap lvars(W) = \emptyset$. □

The previous theorem ensures uniform integrability of program variables at specific points in the execution. However, it requires to prove that loops are expected to be executed a finite number of times. This is weaker than requiring positive a.s. termination, but is clearly stronger than a.s. termination in case of nested loops. The next theorem shows that the probabilistic lexicographic argument is sound if a well-labeling is provided. Yet, such a labeling must ensure that compositional ranking supermartingales employed are integrable.

Theorem 7.7 (Compositional Termination)**.** *Given a well-labeled program* $\{ A \}\, P\, \{ B \}$ *and a semantic space* $(\Omega, \mathcal{F}, \mu, \{\mathcal{F}_n\}, \mathbf{X}_0, \{C_n\})$*, such that* Y_0 *is integrable for all* $Y \in A$ *and for each while-loop* $\{ A_L \}\, W\, \{ B_L \}$ *in* P *we have a compositional ranking supermartingale* R *satisfying*

(i) $vars(R) \in A_L$*.*

(ii) $P' \triangleleft W$ *and* $\{ A' \}\, P' \,\{ B' \}$ *imply* $vars(R) \in A' \cap B'$

Then, $\mathbb{P}(T^{\phi} < \infty) = 1$*, and* $\mathbb{E}(S^{W,\phi}) < \infty$ *for all loops, and all schedulers* ϕ*.*

Proof. By induction on the structure of P for *any* semantic space. The base case is trivially terminating. For the sequence and conditional branching we have to apply similar arguments as in Theorem 7.6, thus we omit them.

We turn to proving that **while** G **do** P_W **od** is a.s. terminating. We can assume that $\{ A \}\, P_W \,\{ A \}$ is well-labeled, i.e. that the MOD rule was not applied to P and therefore $A = A_W \subseteq B_W$. Nesting an induction on k we can prove using IH that P_W a.s. terminates with the semantic space $(\Omega, \mathcal{F}, \mu^k, \{\mathcal{F}_n^k\}, \mathbf{X}_0^k, \{C_n^k\})$. Using Lemma 6.5 and Theorem 7.6 we can then infer that $\Delta T_k^{W,\phi}$ is a.s. finite, and that:

$$\mathbb{E}(R_{T_{k+1}^{W,\phi}} \,|\, \mathcal{F}_{T_k^{W,\phi}}) \leq R_{T_k^{W,\phi}} - \varepsilon \mathbf{1}_{G_0 \cap \ldots G_{T_k^{W,\phi}}}$$

in the original semantic space of P because $Dec_{<}(R, P_W)$ and $vars(R) \in A$. Thus $\{R_{T_n^{W,\phi}} \mathbf{1}_{R_{T_n^{W,\phi}} > 0}\}$ is a ranking supermartingale with respect to the filtration $\{\mathcal{F}_{T_n^{W,\phi}}\}$, and from Lemma 5.5 $S^{W,\phi}$ is integrable. We are only left with the need to prove that T^{ϕ} is a.s. finite. Here, the identity (22) comes in handy to ensure that, since all the involved r.v. are a.s. finite. □

Theorem 7.7 only ensures a.s. termination, i.e. the random variable T^* is not necessarily integrable. The terminating argument built into the proof is not a global supermartingale that decreases in expectation at each step, instead it is a ranking supermartingale that decreases after a.s. finite random steps.

As a concrete example, the following labelled program will be used to demonstrate that our proof system is indeed capable of proving a.s. termination of programs where $\mathbb{E}(T^*)$ is not finite. In other words it exemplifies that we can prove a.s. termination compositionally in settings where global supermartingales arguments must fail.

$\{X\}$
$C, U := 0, 1;$
$\{C, U, X\}$
while $C < 0.5$ **do**
  $\{C, U, X\}$
  $X := U$
  while $X > 0$ **do**
    $\{C, U, X\}$
    $X := X - uniform(0, 1);$
    $\{C, U, X\}$
  od
  $\{C, U, X\}$
  $C, U := uniform(0, 1), 2U;$
  $\{C, U, X\}$
od
$\{C\}$

It is easy to see that the program is well-labeled according to the rules in Table 2. Furthermore, $\mathbf{1}_{[C<0.5]}$ is an obvious ranking supermartingale for the outer loop, and so is X for the inner loop. Thus we can apply Theorem 7.7 to decide that the program a.s. terminates. Furthermore, we can infer that S^W (the number of times the outer loop is executed), and ΔT_k^W (the number of steps in the k-th iteration) are all integrable. In particular, $\mathbb{E}(S^W) = 2$ since S^W is a simple geometric distribution, and $\mathbb{E}(\Delta T_k^W) \leq 4^k$ as the value of X is decreased by 0.5 on average, starting off from 2^k at the beginning of the k-th iteration. Also $\Delta T_k^W > 2^k$, as X is decreased by at most 1. This plausibilises why the program a.s. terminates. However $\mathbb{E}(\sum_{k\leq S^W} T_k^W) \geq \sum_{0<k} \mathbb{P}(S = k)2^k = \infty$, i.e. the program does not terminate in finite mean time.

7.5 An extensible framework

Independent of the above type system, one would intuitively expect that if a program almost surely terminates for all possible initial states (i.e. initializing via Dirac distributions), then the program terminates under any arbitrary distribution, including those violating the initial precondition in our labeling.

Theorem 7.8. *A program P with variables $\{X_n\}$ terminates for all initial valuations such that $\{X_{n,0} = \infty\} = \emptyset$ if and only if it terminates for all initial valuations such that $\mathbb{E}(X_{n,0}) < \infty$.*

Proof. Notice that one implication is trivially true. Suppose P terminates for all initial valuations that make the variables integrable. Let us fix an arbitrary probability space $(\Omega, \mathcal{F}, \mu)$, and an initial valuation $\mathbf{X}_0$, such that all $X_{n,0}(\omega)$ are finite. Consider the semantics of this program under a memoryless scheduler ϕ, and define $A \triangleq \{T^\phi < \infty\}$. We want to prove $\mu(A) = 1$. Let's consider the semantics of the same program under the same probability space, but with a different initial valuation. Namely, $X_{n,0}^m \triangleq X_{n,0}\mathbf{1}_{[|X_{n,0}|\leq m]}$. Thus, we can consider the following measurable sets $A_m \triangleq \{T^{\phi,m} < \infty\}$, and $I_m \triangleq \{\exists_n |X_{n,0}| > m\}$. Notice that $X_{n,0}^m$ is integrable, and therefore $\mu(A_m) = 1$. We also have $A_m \setminus I_m \subseteq A$, $A_m \uparrow A$ and $I_m \downarrow \emptyset$. Thus, $\mu(A_m \setminus I_m) \uparrow 1$, and in turn $\mu(A) = 1$ as needed. □

This theorem is weaker than expected, namely we guarantee termination for those initial distributions that make integrable the program variables. But inside our framework we get this for free owed to the MOD rule. On the other hand, Theorem 7.8 allows us to incorporate a powerful inference rule into our framework

$$\text{ABS}\frac{P \text{ a.s. terminating} \qquad A \subseteq Vars \setminus lvars(P)}{\{\,A\,\}\,P\,\{\,A\,\}}$$

thereby relaxing the requirements on compositional ranking supermartingales so that they only need to ensure integrability of unmodified variables. This rule makes it possible to feed into our framework background knowledge about program fragments, especially obtainable by standard non-probabilistic termination arguments.

8 Related work

Termination of concurrent probabilistic programs was first studied by Hart, Sharir and Pnueli [22]. They focused on finite state programs under fairness assumptions. In that setting termination is a pure topological property, actual probabilities do not play a role. Later, they gave a sound and complete characterization for almost sure termination in case of countable state spaces [21]. Esparza, Gaiser and Kiefer [12], proposed a sound and complete method for proving termination of finite state programs. Their approach is based on [22] and proceeds by finding a ω-regular expression on the possible outcomes of the random experiments that guarantees termination regardless of the actual state. Monniaux [31] used abstract interpretation techniques to approximate the probability distribution of un-nested loops. He proved termination in cases where the probability of executing a loop more than n times decreases exponentially with n. Brázdil et al. [6] proved a.s. termination in the context of probabilistic push-down automata using Azuma's Lemma. McIver and Morgan [30] propose a sound variant rule for probabilistic programs. Their notion is less general than ranking supermartingales as they require variants to be upper bounded. Bournez and Garnier [3] studied positive a.s. termination in the context of term rewriting systems, Gnaedig [17] developed a proof system for positive a.s. termination for rewrite systems under innermost strategies using induction. It is not clear how the author dealt with the integrability problems discussed throughout this paper. Chakarov and Sankaranarayanan [7] applied aspects of martingale theory in the context of verification of deterministic probabilistic programs, presenting sound conditions to prove termination using global ranking supermartingales.

Termination analysis for non-probabilistic programs has grown immensely in the last decade. There are several approaches to prove termination of functional programs and rewriting systems including the size change principle [29], and dependency pairs [16]. In case of imperative programs modern approaches rely on synthesizing several small ranking functions and combining them. Lexicographic approaches [4, 5, 10] try to find an order in the ranking functions in such that the largest modification always decreases. Ramsey methods [11, 8, 32, 2, 28] exploit the fact that a relation is well-founded if its transitive closure is contained in a disjunction of well-founded relation. For a summary on both methods we refer to [10] that proposes a hybrid approach.

9 Discussion

Based on a novel filtration-based treatment of probabilistic program semantics, this paper has successfully addressed several intriguing open questions in probabilistic program termination research. On the foundational side, we have extended the theory of ranking supermartingales to provably work with bounded nondeterministic programs. This is complemented with a sound and complete proof method for almost sure termination, together with a safe method to identify the set of program variables having finite expectation. The framework is strictly decoupled from the distributions appearing in a program, as long as they are integrable and their expectations are known. Care has been taken to make the entire approach readily implementable. Our framework is thus capable of proving automatically a.s. termination on programs that state of the art methods

like [3, 12, 7] are unable to verify. Moreover, it goes strictly beyond positive a.s. termination that is the largest setting where Lyapunov ranking functions are complete.

Among others, our contributions can be used to improve the weakest expectation calculus of McIver and Morgan [30] in two dimensions. First, we can drastically weaken their requirement to provide a proof of termination so as to ensure total correctness. The second aspect to improve upon is related to the fact that soundness of their method is in practice restricted to weakly finite programs (i.e. where the number of reachable states from any initial states is finite). This is rooted in a requirement that all variables are bounded, which in turn directly ensures that all relevant expressions are integrable. Our Theorem 7.6 in conjunction with Theorem 7.7 can be used to drop boundedness restrictions, and thereby extend the scope of analyzable programs considerably.

Probabilistic programming [18] is a relatively new area of research that merges concepts form programming languages, machine learning, static analysis, and probabilistic model checking. Probabilistic models are described as probabilistic programs, and inference methods are used to compute exact or approximate values of probabilities and expectations. Our work can be used in these contexts to perform static analysis of probabilistic programs prior to using Monte Carlo methods, as simulation results on non-terminating programs or programs where first moments do not exist are misleading and usually meaningless. Furthermore, termination analysis is also an important ingredient to guarantee the correctness of probabilistic program slicing [23].

Acknowledgments: This work is supported by the EU 7th Framework Programme under grant agreements 295261 (MEALS) and 318490 (SENSATION), the DFG Transregional Collaborative Research Centre SFB/TR 14 AVACS, and by the CAS/SAFEA International Partnership Program for Creative Research Teams. We would would like to thank Pedro R. D'Argenio and Jan Krčál for their comments in a preliminary version of this paper, Hassan Hatefi for many insightful discussions, and the many anonymous reviewers for their invaluable comments.

Bibliography

[1] R. B. Ash and C. Doléans-Dade. *Probability and Measure Theory*. Harcourt, 2000.

[2] J. Berdine, A. Chawdhary, B. Cook, D. Distefano, and P. W. O'Hearn. Variance analyses from invariance analyses. In *POPL*, pages 211–224. ACM, 2007.

[3] O. Bournez and F. Garnier. Proving positive almost-sure termination. In *RTA*, *LNCS* 3467:323–337. Springer, 2005.

[4] A. R. Bradley, Z. Manna, and H. B. Sipma. The polyranking principle. In *ICALP*, *LNCS* 3580:1349–1361. Springer, 2005a.

[5] A. R. Bradley, Z. Manna, and H. B. Sipma. Linear ranking with reachability. In *CAV*, *LNCS* 3576:491–504. Springer, 2005b.

[6] T. Brázdil, J. Esparza, S. Kiefer, and A. Kucera. Analyzing probabilistic pushdown automata. *Formal Methods in System Design*, 43(2):124–163, 2013.

[7] A. Chakarov and S. Sankaranarayanan. Probabilistic program analysis with martingales. In *CAV*, *LNCS* 8044:511–526. Springer, 2013.

[8] M. Codish and S. Genaim. Proving termination one loop at a time. In *WLPE*, *CW371 Report*, pages 48–59. K. U. Leuven, 2003.

[9] M. Colón and H. Sipma. Synthesis of linear ranking functions. In *TACAS*, *LNCS* 2031:67–81. Springer, 2001.

[10] B. Cook, A. See, and F. Zuleger. Ramsey vs. lexicographic termination proving. In *TACAS*, *LNCS* 7795:47–61. Springer, 2013.

[11] N. Dershowitz, N. Lindenstrauss, Y. Sagiv, and A. Serebrenik. A general framework for automatic termination analysis of logic programs. *Appl. Algebra Eng. Commun. Comput.*, 12(1/2):117–156, 2001.

[12] J. Esparza, A. Gaiser, and S. Kiefer. Proving termination of probabilistic programs using patterns. In *CAV*, *LNCS* 7358:123–138. Springer, 2012.

[13] J. Filar and K. Vrieze. *Competitive Markov Decision Processes*. Springer-Verlag New York, Inc., New York, NY, USA, 1996.

[14] R. W. Floyd. Assigning meanings to programs. *Mathematical aspects of computer science*, 19:19–32, 1967.

[15] F. G. Foster. On the stochastic matrices associated with certain queuing processes. *The Annals of Mathematical Statistics*, 24(3):355–360, 09 1953.

[16] J. Giesl, R. Thiemann, and P. Schneider-Kamp. The dependency pair framework: Combining techniques for automated termination proofs. In *LPAR*, *LNCS* 3452:301–331. Springer, 2004.

[17] I. Gnaedig. Induction for positive almost sure termination. In *PPDP*, pages 167–178. ACM, 2007.

[18] A. D. Gordon, T. A. Henzinger, A. V. Nori, and S. K. Rajamani. Probabilistic programming. In *FOSE*, pages 167–181. ACM, 2014.

[19] F. Gretz, J.-P. Katoen, and A. McIver. Prinsys - on a quest for probabilistic loop invariants. In *QEST*, *LNCS* 8054:193–208. Springer, 2013.

[20] F. Gretz, J.-P. Katoen, and A. McIver. Operational versus weakest pre-expectation semantics for the probabilistic guarded command language. *Perform. Eval.*, 73:110–132, 2014.

[21] S. Hart and M. Sharir. Concurrent probabilistic programs, or: How to schedule if you must. *SIAM J. Comput.*, 14(4):991–1012, 1985.

[22] S. Hart, M. Sharir, and A. Pnueli. Termination of probabilistic concurrent program. *ACM Trans. Program. Lang. Syst.*, 5(3):356–380, 1983.

[23] C.-K. Hur, A. V. Nori, S. K. Rajamani, and S. Samuel. Slicing probabilistic programs. In *PLDI*, pages 133–144. ACM, 2014.

[24] C. Jones. *Probabilistic non-determinism*. PhD thesis, University of Edinburgh, 1989.

[25] J.-P. Katoen, A. McIver, L. Meinicke, and C. C. Morgan. Linear-invariant generation for probabilistic programs: - automated support for proof-based methods. In *SAS*, *LNCS* 6337:390–406. Springer, 2010.

[26] S. Katz and Z. Manna. A closer look at termination. *Acta Inf.*, 5:333–352, 1975.

[27] D. Kozen. Semantics of probabilistic programs. *J. Comput. Syst. Sci.*, 22(3):328–350, 1981.

[28] D. Kroening, N. Sharygina, A. Tsitovich, and C. M. Wintersteiger. Termination analysis with compositional transition invariants. In *CAV*, *LNCS* 6174:89–103. Springer, 2010.

[29] C. S. Lee, N. D. Jones, and A. M. Ben-Amram. The size-change principle for program termination. In *POPL*, pages 81–92. ACM, 2001.

[30] A. McIver and C. Morgan. *Abstraction, Refinement And Proof For Probabilistic Systems*. Springer, 2004.

[31] D. Monniaux. An abstract analysis of the probabilistic termination of programs. In *SAS*, *LNCS* 2126:111–126. Springer, 2001.

[32] A. Podelski and A. Rybalchenko. Transition invariants. In *LICS*, pages 32–41. IEEE Computer Society, 2004a.

[33] A. Podelski and A. Rybalchenko. A complete method for the synthesis of linear ranking functions. In *VMCAI*, *LNCS* 2937:239–251. Springer, 2004b.

[34] K. Sohn and A. V. Gelder. Termination detection in logic programs using argument sizes. In *PODS*, pages 216–226. ACM Press, 1991.

[35] D. Williams. *Probability with Martingales*. Cambridge University Press, 1991.

MEALS Partner Abbreviations

SAU: Saarland University, D

RWT: RWTH Aachen University, D

TUD: Technische Universität Dresden, D

INR: Institut National de Recherche en Informatique et en Automatique, FR

IMP: Imperial College of Science, Technology and Medicine, UK

ULEIC: University of Leicester, UK

TUE: Technische Universiteit Eindhoven, NL

UNC: Universidad Nacional de Córdoba, AR

UBA: Universidad de Buenos Aires, AR

UNR: Universidad Nacional de Río Cuarto, AR

ITBA: Instituto Técnológico Buenos Aires, AR