Probabilistic Program Analysis with Martingales (Seminar Probabilistic Programs)

Frederik Zwilling 304314

January 26, 2015

Abstract

Verifying probabilistic programs is an important challenge because randomized algorithms are used in many safety-critical domains, such as robotics or networking. This report presents an approach to analyze probabilistic programs with the use of martingale theory. Loosely speaking, martingales are expressions whose current value equals the expected value after the next execution step. The approach includes an automate synthesis of martingales in linear probabilistic programs, verification of probabilistic assertions and proving almost sure termination with super martingales, an extension of the notation of martingales. The approach can handle continuous distributions but the automated synthesis is limited to linear programs.

1 Introduction

Probabilistic programs are becoming increasingly important because randomized algorithms are used in many emerging domains, such as robotics, artificial intelligence and network protocols [1]. For instance in the domain of robotics, a widely used example of probabilistic programs is the monte-carlo localization of a mobile robot [2]. Here, randomization is used to create position estimates and keep estimates with higher likelihood if the current sensor data matches the expected data.
Probabilistic programs can be defined as ordinary programs that sample values form probability distributions. As for ordinary programs, it is desirable to analyze probabilistic programs by investigating program termination and verifying probabilistic assertions about the outcome of the program. In the

aforementioned example, verification is important to ensure the safety of the robot, especially if the robot is deployed in industry or space, where the safety requirements are high. Unfortunately, verification of probabilistic programs can be hard [3] and many approaches, such as *quantitative invariants* proposed by McIver and Morgan [4], are limited to discrete probabilistic choices or specific probability distributions.
In this report, we present the approach of *probabilistic program analysis with martingales* by Chakarov and Sankaranarayanan [5]. The approach is an extension of quantitative invariants [4, 6] and allows continuous probability distributions, such as the Gaussian or Poisson distribution. Simplified, a *martingale* can be understood as an expression whose current value equals its expected value after the next execution step of the probabilistic program. Thus, the expected value of the expression does not change during the program execution. We also use *super martingales* which are similar to martingales, except that their value decreases during execution of the program. We use these martingales and combine them with martingale theory, in particular with the *Azuma-Hoeffding theorem*, to derive probabilistic bounds of the martingale value, which can be used to prove probabilistic assertions. Furthermore, we use the notation of *super martingale ranking functions* to give a sound, but incomplete, approach to prove almost sure termination. To be able to automate the verification, we also provide a method to find (super) martingales and super martingale ranking functions in linear probabilistic programs.

1.1 Motivating Examples

The following two probabilistic programs are simple examples which accompany us through the report and show what kind of probabilistic assertions and termination property we want to prove.

Example 1.1. The probabilistic program in Figure 1 sums up 500 samples from a uniform distribution in the interval $[0, 1]$. Because the sum is computed in a bounded loop and the expected value of each random variable that is added in each loop iteration is 0.5, the expected value of the sum after the loop terminates is 250. The sample paths shown on the right side of Figure 1 show that the distribution of the resulting sum is clustered around the expected value 250. Static analysis can use the invariant $0 \leq x \leq i+1$ in the loop to derive that $0 \leq x \leq 501$ is valid at the loop exit. In Section 4 we use the approach provided in this report to prove the probabilistic assertion $Pr(x \in [200, 300]) \geq 1 - 9.1 * 10^{-5}$.

```
real x = 0;
real N = 500;
for ( i=0; i < N; ++i )
  x = x + unifRand(0,1);
// Pr(x∈[200,300]) ?
```

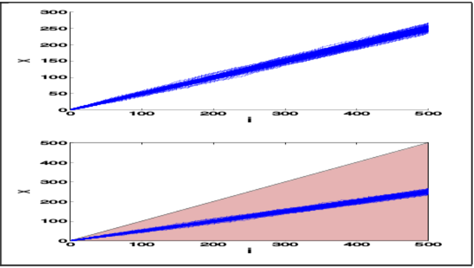

Figure 1: (Left) Probabilistic program of Example 1.1 which sums up random variables. We want to analyze with which likelihood the sum is in a certain interval around the expected sum. (Right) Each blue line represents a sample path through the program where the variables i and x are plotted for each step along the path. The read area represents the space of all theoretically possible states [5].

```
real h, t;
// h is hare and t is tortoise
h = 0; t = 30;
while ( h <= t ){
  if (flip (0.5) )
       h = h + unifRand(0,10);
  t = t +1;
} // almost sure terminate?
```

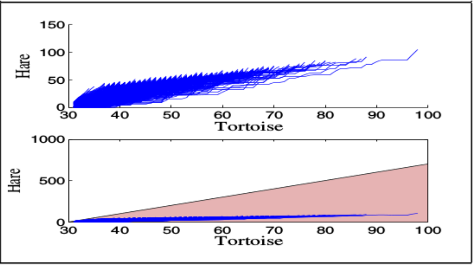

Figure 2: (Left) Probabilistic program of Example 1.2 with a loop. The termination depends on the probabilistic increment of the variable h and we want to prove if the program terminates almost surely. (Right) Sample paths of program executions as in Figure 1. The red area of possible states shows that there can be executions which take arbitrary long [5].

Example 1.2. The probabilistic program shown in Figure 2 contains a loop and two variables h and t that represent the positions of a hare and tortoise playing a race . t starts with a lead of 30 and is incremented by 1 in every loop iteration. h is increased by a random variable uniformly distributed in the interval $[0, 10]$ with probability $\frac{1}{2}$. The race ends if the hare surpasses the tortoise, or equivalently if $h > t$. Ordinary static analysis can not show that the program terminates because there are possible executions that take arbitrary long. Because the expected increase of h per loop iteration is 2.5, it is intuitive that the program terminates after a finite number of iterations, which is also illustrated by the sample paths. This motivates the definition of *almost sure termination* in the next section. To prove the almost sure termination of this program, we use the super martingale expression $t - h$ which represents the distance between h and t and decreases in every iteration in expected value.

2 Preliminary Definitions

Before describing the martingale approach to analyze probabilistic programs, we need to introduce some preliminary definitions [5].

To describe the possible execution paths, actions and conditional branching of a probabilistic program we need a formalism to describe the program. We model the program as a *probabilistic transition system* that comprises among others a set of *program variables*, a set of program *locations* indicated by the program line number and *transitions* between the locations. The transitions going out from a location consist of a *guard assertion* which determine the enabled transition at a given location and a list of probabilistic *forks*, which represent a probabilistic branching. Each fork contains a probability value (with which the fork is taken), an update function for the program variables which represent the assignments of the program and a target location.

Figure 3 shows the probabilistic transition system of the program in Example 1.2. It contains two transitions outgoing from line 4 to model the loop guard. The lower transition τ_1 models the loop iteration and contains two forks for the coin-flip in line 5. The effects of the two update functions, with the increment of h and without, are also written next to the two forks. On the loop exit, the upper transition τ_2 is used. Here, there is only one fork because there is no probabilistic branching. Formally, the notion of probabilistic transition systems is captured by the following definition:

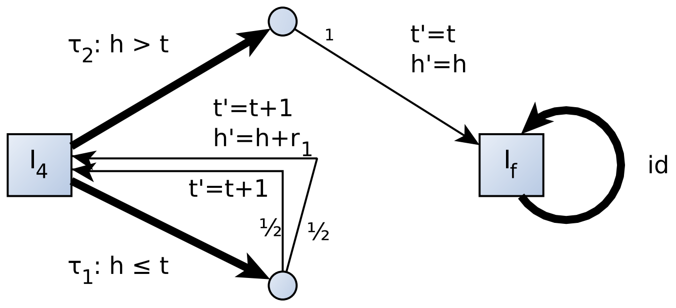

Figure 3: Probabilistic transition system associated to the program in Example 1.2.

Definition 2.1 (Probabilistic Transition System)**.**
A *Probabilistic Transition System (PTS)* Π is a tuple $\langle X, R, L, \mathcal{T}, l_0, x_0, l_f \rangle$, where

- X is a vector containing all program variables.
- R is a vector containing all random variables with joint distribution $\mathcal{D}$.
- L is a finite set of locations in the program (indicated by the line number).
- l_0 and x_0 are the initial location and values of the program variables.
- l_f is the final location where the program is considered as terminated.
- $\mathcal{T}$ is a finite set of transitions. Each transition $\tau \in \mathcal{T}$ is a tuple $\langle l, \phi, f_1, ..., f_k \rangle$ with
  - Source location $l \in L$ and guard assertion ϕ over X,
  - Forks $f_1, ..., f_k$ where each fork f_i is a tuple (p_i, F_i, m_i) with fork probability $p_i \in (0, 1]$, $\sum_{i=1}^{k} f_i = 1$, an update function $F_i(X, R)$ and a destination location $m_i \in L$.

A *state* of a PTS is a tuple (l, x) where $l \in L$ is a location and x is a variable valuation of the program variables X. The transition $\tau = \langle l, \phi, f_1, ..., f_k \rangle \in \mathcal{T}$ is said to be *enabled* in state (l, x) if $x \models \phi$.

To make the transition choice in a PTS deterministic and simplify our development, we consider only PTS where in every location with any variable valuation x of X exactly one transition is enabled. This restriction is called

```
int x := 0;
while (flip (0.5))
   x ++;
// end
```

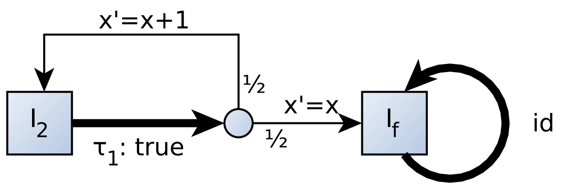

Figure 4: An almost sure terminating program and its PTS [5].

No Demonic Restriction and formally means that for each location $l \in L$ and all outgoing transitions $\tau_1, ..., \tau_n$ from l with guards $\phi_1, ..., \phi_n$, $\phi_1, ..., \phi_n$ are mutually exclusive and their conjunction is always valid. Therefore, every state s is mapped to only one enabled transition denoted $\tau(s)$.

As shown in Figure 3, we add a transition labeled id of the form $\langle l_f, true, (1, id, l_f)\rangle$ at the end of each program to satisfy the no demonic restriction. The program is considered terminated when it reaches l_f.

An execution step of a PTS in state (l_i, x) uses the enabled transition $\tau = \langle l_i, \phi, f_1, ..., f_k\rangle$ and chooses a fork $f_j = (p_j, F_j, m_j)$ with probability p_j. The state is updated by τ to $(m_j, F_j(x, r))$ where r is sampled from $\mathcal{D}$. The distribution of states after execution of a step from state s is denoted `Post-Distrib`(s). Now, we are able to define *sample executions*, which models specific sample paths as shown in Figure 1 and 2.

Definition 2.2 (Sample Executions). A *sample execution* σ of a PTS $\langle X, R, L, \mathcal{T}, l_0, x_0\rangle$ is a countably infinite sequence of states $(l_0, x_0) \xrightarrow{\tau_1} (l_1, x_1) \xrightarrow{\tau_2} (l_2, x_2) \xrightarrow{\tau_3} ...$ with τ_i being the enabled transition in state (l_{i-1}, x_{i-1}) and $(l_i, x_i) \in$ `Post-Distrib`(l_{i-1}, x_{i-1}) for all $i \in \mathbb{N}$.
A sample execution is *terminating* if it reaches a state (l_f, x) for some x.
The probability $\mu(\sigma) \in [0, 1]$ of a sample execution σ is the product of all fork probabilities used along the sequence of states.

With the PTS and sample executions we can now define the concept of almost sure termination informally presented in Example 1.2. For the program in Example 1.1, every sample execution terminates after 500 loop iterations and therefore the program terminates surely. For the program in Figure 4, where the loop condition depends on a coin flip, the only non-terminating sample execution σ_∞ has the probability $\mu(\sigma_\infty) = lim_{n\to\infty} \prod_1^n \frac{1}{2} = 0$. Therefore, the sum of all terminating sample executions is 1 and the program is almost sure terminating. We formalize this in the following definition:

Definition 2.3 (Almost-Sure Termination). A PTS *terminates almost surely* iff the probabilities of all terminating sample executions sums up to 1.

To compute the expected value of an expression e after the next execution step we compute the expected values in all possible following states (with respect to the random variables with joint distribution $\mathcal{D}$) and weight them with the probability of the used fork. In other words, we evaluate the expression over the `Post-Distrib` of the current state. We use $e[x/F(x,r)]$ to denote the substitution in e of every program variable with the corresponding value given by the update function F.

Example 2.1. Continuing with Example 1.2, we are in state $s = (l_4, (h,t))$ with $h \leq t$ and want to investigate the expected value of the expression $e = 5t - 2h$ after the next step. At the end of the example we identify that e is a martingale. Figure 3 shows that the enabled transition in s is $\tau_1 = \langle l_4, (h \leq t), f_1, f_2 \rangle$ with the two forks $f_1 = (\frac{1}{2}, F_1 : (h,t,r_1) \mapsto (h, t+1), l_4)$ for a failed coin-flip and $f_2 = (\frac{1}{2}, F_2 : (h,t,r_1) \mapsto (h + r_1, t+1), l_4)$ for a successful one. That means that the a posteriori value of e when taking fork F_1 is $5 * (t+1) - 2h$. If F_2 is taken instead, we get the a posteriori value $5*(t+1) - 2*(h+r_1)$. The expected value of e after applying τ_1 is the sum of the expected values resulting from taking forks F_1 and F_2 weighted by the fork probabilities:

$$\begin{aligned}
\mathbb{E}_{\tau_1}(e \mid s) &= \frac{1}{2}\mathbb{E}(e[x/F_1(h,t,r_1)]) + \frac{1}{2}\mathbb{E}(e[x/F_2(h,t,r_1)]) \\
&= \frac{1}{2}\mathbb{E}(5(t+1) - 2h) + \frac{1}{2}\mathbb{E}(5(t+1) - 2(h + r_1)) \\
&= 5t - 2h + 5 - \mathbb{E}(r_1) \overset{(r_1 \text{from unifRand}(0,10))}{=} 5t - 2h = e
\end{aligned}$$

This idea of computing the expected value of an expression after the next execution step is captured by the following definition:

Definition 2.4 (Post-Expectation). Given a state $s = (l, x)$ with an enabled transition $\tau = \langle l, \phi, f_1, ..., f_k \rangle, k \geq 1, f_i = (p_i, F_i, m_i)$, the *Post-Expectation* $\mathbb{E}_\tau(e \mid s)$ of an expression e over program variables is the expected value of e over `Post-Distrib`(s) under the distribution $\mathcal{D}$ for random variables r:

$$\mathbb{E}_\tau(e \mid s) := \sum_{i=1}^{k} p_i * \mathbb{E}(e[x/F_i(x,r)])$$

3 Martingales and Martingale Expressions

In this section we introduce the notation of martingales and some extensions. Thereof we need to analyze a probabilistic program. As calculated in Example 2.1, the value of the expression $e = 5t - 2h$ in the state $s = (l_4, (h,t))$

equals $\mathbb{E}_\tau(e \mid s)$, the expected value after the next execution step. This is the martingale property we use later to analyze the PTS. It is easy to see, that e has this property also for the other two transitions τ_2 and *id*. This is because neither transition modifies the program variables. Now we continue with the definition of martingales [5].

Definition 3.1 (Martingale and Super Martingale Expressions)**.** An expression e over program variables X is called *martingale* for a PTS iff for every state $s = (l, x)$ with enabled transition $\tau(s)$, the Post-Expectation of e equals the current value of e:

$$\forall s = (l, x) : \mathbb{E}_{\tau(s)}(e \mid s) = e \ .$$

Similarly, an expression is called a *super-martingale* iff

$$\forall s = (l, x) : \mathbb{E}_{\tau(s)}(e \mid s) \leq e \ .$$

Example 3.1. We now show that $e = t - h$ is a super martingale of Example 1.2. This super martingale is interesting because it decreases across the loop iterations making the guard eventually false and proving thus termination. Again, for the states (l_f, x) and $(l_4, (h, t))$ with $h > t$, the transitions do not change the program variables and therefore the Post-Expectation of e equals the current value of e. For the remaining states $s = (l_4, (h, t))$ with $h \leq t$, we have

$$\begin{aligned} \mathbb{E}_{\tau_1}(e \mid s) &= \frac{1}{2}\mathbb{E}(t + 1 - h) + \frac{1}{2}\mathbb{E}(t + 1 - (h + r_1)) \\ &= t - h + 1 - \frac{1}{2}\mathbb{E}(r_1) \overset{(r_1 \text{from unifRand}(0,10))}{=} t - h - 1.5 \leq e \ , \end{aligned}$$

which shows that e is a super martingale.

In literature of probability theory, martingales are usually defined in the context of *stochastic processes* $\{M_n\}$ which are sequences of random variables $M_0, M_1, M_2, ...$ with samples $m_0, m_1, ...$ [7]. We introduce the formal definition of such martingales to link the theorems in the probability theory area with the martingale expressions we use to verify probabilistic programs:

Definition 3.2 (Martingales and Super Martingales)**.** A discrete stochastic process $\{M_n\}$ is a *martingale* iff for each $n > 0$, the expected value of the next step equals the current value:

$$\mathbb{E}(M_n \mid m_{n-1}, ..., m_0) = m_{n-1}$$

Similarly, $\{M_n\}$ is a *super martingale* iff for each $n > 0$

$$\mathbb{E}(M_n \mid m_{n-1}, ..., m_0) \leq m_{n-1} \ .$$

This implies that $\{M_n\}$ and $\{-M_n\}$ are super martingales whenever $\{M_n\}$ is a martingale. To convert a (super) martingale expression e into a (super) martingale, we set M_n as the random variable capturing the value of e in the n-th execution step, i.e. $M_n = (e)_n$.

The previous examples contain very simple martingales. For more complex probabilistic programs it is usually difficult to find simple martingale expressions. In such cases it is useful to consider a different martingale expression for each location to reason about termination.

Example 3.2. The program shown in Figure 4 is simple but already shows the difficulty to find a single linear martingale expression as in Example 2.1. Every linear martingale of the form $e = cx$ would be 0 because

$$\mathbb{E}_{\tau_1}(cx \mid (l_2, x)) = \frac{1}{2}(c(x+1) + cx) = cx + \frac{c}{2} \overset{!}{=} cx \ .$$

If we consider instead the function

$$\eta(l, x) = \begin{cases} x & , l = l_2 \\ x - 1 & , l = l_f \end{cases} \quad ,$$

we get a map that satisfies a lifted version of the martingale property. For an expression map η, the Post-Expectation w.r.t. τ is given by $\mathbb{E}_\tau(\eta \mid s) = \sum_{i=1}^{k} p_i * \mathbb{E}(\eta(l_i)[x/F_i(x, r)])$. The notation of Martingale Expression Map is captured by the following definition:

Definition 3.3 (Martingale and Super Martingale Expression Maps)**.**
A *flow-sensitive expression map* η of a PTS is a map from the locations L to expressions over X.
A flow-sensitive expression map η is a *martingale* of a PTS iff $\forall s = (l, x)$: $\mathbb{E}_{\tau(s)}(\eta \mid s) = \eta(l)$. Similarly, η is a *super martingale* of a PTS iff $\forall s = (l, x)$: $\mathbb{E}_{\tau(s)}(\eta \mid s) \leq \eta(l)$.

Now, we show that the map proposed in Example 3.2 is actually such a martingale expression map:

$$\mathbb{E}_{\tau_1}(\eta(l,x) \mid (l_2,x)) = \frac{1}{2}(\underbrace{x+1}_{\eta(l_2)[x/x+1]}) + \frac{1}{2}(\underbrace{x-1}_{\eta(l_f)[x/x]}) = x = \eta(l_2,x) \ .$$

Showing the same for l_f and id is trivial again.

4 Probabilistic Assertions

With the preliminary definitions of the previous sections we can now use martingale theory to derive probabilistic assertions. The main theorem we use from martingale theory is the *Azuma-Hoeffding Theorem.* It uses a super martingale process to provide a probabilistic statement that bounds the deviation of the process from its initial value. The theorem consists in a concentration of measure inequality and is proven e.g. in [8].

Theorem 4.1 (Azuma-Hoeffding Theorem)**.** If $\{M_n\}$ is a super martingale with $|m_n - m_{n-1}| < c$ for some constant c, then for all $n \in \mathbb{N}$ and $t \in \mathbb{R}_0^+$, it follows that

$$Pr(M_n - M_0 \geq t) \leq exp\left(\frac{-t^2}{2nc^2}\right).$$

If $\{M_n\}$ is a martingale with the same properties we can combine the upper bound for $\{M_n\}$ and $\{-M_n\}$ to obtain

$$Pr(|M_n - M_0| \geq t) \leq 2 * exp\left(\frac{-t^2}{2nc^2}\right).$$

Example 4.1. We apply the Azuma-Hoeffding theorem to the program in Example 1.1 to bound the probability that $x \in [200, 300]$ after 500 iterations. An easy to find martingale is $2x - i$ because in every iteration, we add an expected value of 0.5 to x, therefore $2x$ coincides with the increment of i, which is 1. The initial value of $2x - i$ is 0. To get the right interval we observe that

$$\begin{aligned} &Pr(|M_n - M_0| \geq t) = Pr(|(2x-i)_{500} - (2x-i)_0| \geq t) \\ &= Pr(|2x - 500 - 0| \geq t) = Pr(|x - 250| \geq \frac{t}{2}) = 1 - Pr(|x - 250| < \frac{t}{2}) \end{aligned}$$

and thus we choose $t = 100$. Because we add in each iteration a maximum of 1 to x, we can choose $c = 1$. Now, we can conclude from the Azuma-Hoeffding theorem that

$$Pr(x \in [200, 300]) = 1 - Pr(|M_{500} - M_0| \geq 100)$$
$$\geq 1 - 2exp\left(\frac{-100^2}{2 * 500 * 1}\right) \geq 1 - 9.1 * 10^{-5} \; .$$

Thus, it is very likely that x is in the interval $[200, 300]$ after 500 iterations.

5 Almost Sure Termination

In this section, we present an approach to show almost sure termination by using super martingales. The super martingales we use indicate, like the expression $t - h$ in Example 3.1, the termination of the program by getting smaller than 0. To prove that an expression or expression map becomes negative, the monotonic decrease is not enough because the martingale could converge to a value greater than 0. Therefore, we can only use those martingales with the property $\mathbb{E}(M_{n+1} \mid m_n, ..., m_0) \leq m_n - \epsilon$ for some $\epsilon > 0$ because those martingales become eventually negative. Because we also want to verify complex probabilistic programs, we generalize our approach to expression maps which only become negative in the final location. This results in the following definition of super martingale ranking functions:

Definition 5.1 (Super Martingale Ranking Function)**.** A *super martingale ranking function (SMRF)* η is a super martingale expression map of a PTS $\Pi = \langle X, R, L, \mathcal{T}, l_0, x_0, l_f \rangle$ with the properties:

- $\eta(l) \geq 0$ for all $l \in L \backslash \{l_f\}$
- $\eta(l_f) \in [-K, 0)$ for some $K > 0$
- For some constant $\epsilon > 0$ and all transitions $\tau \in \mathcal{T} \backslash \{id\}$, the following holds: $\forall s = (l, x) : \mathbb{E}_{\tau(s)}(\eta \mid s) \leq \eta(l) - \epsilon$

Now, we can introduce the main theorem of the almost sure termination analysis:

Theorem 5.1. Every PTS Π that has a super martingale ranking function η terminates almost surely.

Intuitively, we already explained this theorem by using the unbounded monotonic decrease of η until l_f is reached. A proof can be found with a slightly modified notation in [5].

```
int x := 10; while (x >= 0) { if (flip(0.5)) x++; else x --; }
```

Figure 5: A probabilistic program which terminates almost surely and has no SMRF [5].

Example 5.1. We now find a SMRF for the PTS of the program in Example 1.2. We already showed in Example 3.1 that $t - h$ is a super martingale. Therefore, also $t - h + 9$ is a super martingale. From that, we can construct a SMRF candidate η with $\eta(l_4) = t - h + 9$ and $\eta(l_f) = t - h$. The '+9' ensures that $\eta(l_4)$ is non-negative, because the maximum difference of $\eta(l_4)$ in each step is 9 and the program leaves l_4 if $t - h + 9 < 9$. $\eta(l_f)$ is negative, because otherwise the program would not have left the loop. To also show that the last restriction of Definition 5.1 holds, we can choose $\epsilon = 1$ and have for the transition τ_2:

$$\mathbb{E}_{\tau_2}(\eta \mid s) = \eta(l_f)[t/t, h/h] = t - h \leq t - h + 9 - 1 = \eta(l_4) - \epsilon$$

and for the transition τ_1:

$$\mathbb{E}_{\tau_1}(\eta \mid s) = \mathbb{E}(\eta(l_4)[t/t+1, h/h+r_1]) \overset{\text{Ex.3.1}}{=} t - h - 1.5 \leq \eta(l_4) - \epsilon\ .$$

Thus the PTS has a SMRF and is almost sure terminating.

Example 5.2. To show almost sure termination in Example 3.2, we can use the expression map η with $\eta(l_2) = 1$ and $\eta(l_f) = -1$ and show that η is a SMRF. The first two properties of Definition 5.1 are satisfied because $\eta(l_2) = 1 > 0$ and $\eta(l_f) = -1 < 0$. For the third property, we only need to check the transition τ_1:

$$\mathbb{E}_{\tau_1}(\eta \mid s) = \frac{1}{2}\mathbb{E}(\eta(l_2)[x/x+1]) + \frac{1}{2}\mathbb{E}(\eta(l_f)[x/x]) = \frac{1}{2} - \frac{1}{2} = 0 < 1 - \epsilon = \eta(l_f) - \epsilon$$

Therefore, we can choose $\epsilon = \frac{1}{2}$ and have shown that η is a SMRF and thus the program is almost sure terminating.

Our approach of using Theorem 5.1 to prove almost sure termination is sound, but not complete as the program in Figure 5 shows. The program is almost sure terminating due to the recurrence properties of symmetric random walks [5]. Because the expected value of x after every iteration equals the current value of x and the same is valid for an expression over x, there is no SMRF.

6 Discovering Martingales

So far we know what martingales are, how to check if an expression (map) is a martingale and how to derive probabilistic assertions and prove almost sure termination. The remaining problem is to find (super) martingales for a given program. This section shows how to discover (super) martingales for *affine PTS*. An affine PTS only uses conjunctions of linear inequalities as transition guards and affine update functions $F_i(x,r) = A_i x + B_i r + a_i$ with the vector of program variables $x = (x_1, ..., x_n)^T$ and random samples $r = (r_1, ..., r_n)^T$ matrices A_i and B_i and vector a_i. For example the fork F_1 that increments h by a random sample r_1 in Figure 3 could be written as:

$$F_1(\begin{pmatrix} h \\ t \end{pmatrix}, \begin{pmatrix} r_1 \\ * \end{pmatrix}) = \underbrace{\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}}_{A_1} \underbrace{\begin{pmatrix} h \\ t \end{pmatrix}}_{x} + \underbrace{\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}}_{B_1} \underbrace{\begin{pmatrix} r_1 \\ * \end{pmatrix}}_{r} + \underbrace{\begin{pmatrix} 0 \\ 1 \end{pmatrix}}_{a_1} = \begin{pmatrix} h + r_1 \\ t + 1 \end{pmatrix}$$

For these affine PTS it is straightforward to state the restrictions for a martingale and apply some linear algebra.

Example 6.1. The PTS of the program from Example 1.2 is affine because all update functions and guards are linear. The super martingale expression we want to find has the form $c^T x + d$. Because for every (super) martingale e, also $e + k$, for some constant k, is a (super) martingale, we can choose $d = 0$. For a state $s = (l_4, (h,t))$ with $h \leq t$ the martingale has to satisfy the formula

$$\mathbb{E}_{\tau_1}(e \mid s) = \frac{1}{2}\mathbb{E}(c_1 h + c_2(t+1)) + \frac{1}{2}\mathbb{E}(c_1(h + r_1) + c_2(t+1)) = c_1 h + c_2 t$$
$$\Leftrightarrow c_1 h + c_2 t + c_2 + \frac{1}{2} c_1 \mathbb{E}(r_1) = c_1 h + c_2 t \overset{\mathbb{E}(r_1)=5}{\Leftrightarrow} c_2 + \frac{5}{2} c_1 = 0 \ .$$

The other constraints resulting from the other two transitions are always valid because the transitions do not change the program variables. Therefore, we get $p(\frac{5}{2}t - h) + q$ as martingales for any $p, q \in \mathbb{R}$. To get super martingales we need to satisfy $c_2 + \frac{5}{2}c_1 \leq 0$, what we can derive as before. This yields the super martingales $p(\frac{5}{2}t - h) - kh + q$ for any $p, q, k \in \mathbb{R}$ with $k \geq 0$.

Example 6.2. To derive linear constraints from Example 6.1, we need to encode the implication $\forall x(\phi(x)) \Rightarrow \mathbb{E}_\tau(\eta \mid (l,x)) \leq \eta(l)$ which states that $c^T x$ is a super martingale. For simplicity, we only investigate τ_1 that results in the following linear constraint:

$$\underbrace{(1,-1)\begin{pmatrix} h \\ t \end{pmatrix} \leq 0}_{\phi_1 = h \leq t} \Rightarrow \frac{1}{2} c^T F_1(x,r) + \frac{1}{2} c^T F_2(x,r) = c^T \begin{pmatrix} h + \frac{r_1}{2} \\ t + 1 \end{pmatrix} \leq c^T x$$

By using Farkas Lemma we can ensure that the resulting constraints are linear inequalities [5]:

Theorem 6.1 (Farkas Lemma)**.** The linear constraint $Ax \leq b \Rightarrow c^T x \leq d$ is valid iff its alternative is satisfiable $A^T \lambda = c \wedge b^T \lambda \geq d \wedge \lambda \geq 0$.

To find a SMRF η, we need state the linear inequalities for all transitions to introduce more variables ϵ and c_l, because every location $l \in L$ can have a different linear expression, and more constraints to satisfy the requirements of Definition 5.1: $\eta(l_f) < 0$, $\epsilon > 0$ and $\eta(l) \geq 0$ for all $l \in L \backslash \{l_f\}$. The constraints for the transitions $\tau \in \mathcal{T} \backslash \{id\}$ and states $s = (l, x)$ have to be modified to $\mathbb{E}(\eta \mid s) \leq \eta(l) - \epsilon$.

7 Related Work

The work by Chakarov and Sankaranarayanan presented in this paper is related to various other publications [5]. It is closely related to the deductive approach with *probabilistic invariants* proposed by McIver and Morgan [4, 6]. There are two types of probabilistic invariants, namely *exact* and *quantitative invariants*, that are similar to martingales and super martingales. Quantitative invariants are expressions whose post-expectations after a loop iteration is greater or equals than their values at the beginning of loop iteration. They are also used to prove almost sure termination and correspond to super martingales if the expression is negated. Similarly, exact invariants correspond to martingales because their post-expectations after a loop iteration equals their values at the beginning of loop iteration. To prove almost sure termination, they use the *probabilistic variant rule* which can be seen as a specialization of the technique based on SMRF. However, there are some differences, because in contrast to probabilistic invariants, the approach presented in this paper allows continuous probability distributions and program variables but no demonic non-determinism and has no deductive proof system.

There are also other publications which use martingale theory to derive probabilistic guarantees for randomized algorithms. For example [9] also uses the method of bounded differences provided by the Azuma-Hoeffding Theorem. However, our approach extends that method by the automatable discovery of martingales.

In a later work by Chakarov and Sankaranarayanan, they generalize the approach presented in this report with the notation of *inductive expectation invariants* to analyze probabilistic program loops [10]. In terms of this report, inductive expectation invariants are expressions whose post-expectation is non-negative in every loop iteration if the current value is non-negative. By characterizing these inductive expectation invariants as fixed points of a loop, they discover the invariants automatically.

8 Conclusion

This report presents an approach by Chakarov and Sankaranarayanan to analyze probabilistic programs with the use of martingale theory, especially to derive probabilistic assertions and prove almost sure termination. For a given probabilistic program, we construct the corresponding PTS Π. If Π is affine, we can automatically find (super) martingales and SMRF by solving some linear equations. If we can find a SMRF, the program terminates almost surely. By using the Azuma-Hoeffding Theorem, martingales can be used to derive probabilistic assertions about the program outcome and intermediate states.

The main advantages of the presented approach, in contrast to the comparable approaches, are partly automation of the program analysis and the possibility to use continuous distributions and not being limited to a fixed set of distributions. The main disadvantage, which limits the usage of the approach in practice, is the restriction to affine PTS in the discovery of martingales. For the example of the monte-carlo localization of a mobile robot mentioned in the introduction, this means that we can only analyze simple versions of the monte-carlo localization, e.g. when we only want to determine the coordinates of the robot but not its orientation. If we also want to determine the orientation of the robot and use a distance sensor, we would need trigonometric functions that violate the necessary linearity of the update functions. Furthermore, even a simple multiplication of two program variables hinders us from using the discovery approach, thus many programs can not be analyzed automatically. Another disadvantage is that we can not investigate arbitrary probabilistic assertions directly, because the approach derives the assertion bounds with the Azuma-Hoeffding Theorem from a (super) martingale that has to be found first. For example, if we want to investigate the $Pr(x \in [300, 400])$ in Example 1.1, we would have to manually discover what martingales to use, with which parameter t and how to manipulate the obtained bounds to get the required bounds.

Chakarov and Sankaranarayanan also provide an implementation of the automated martingale discovery. The implementation takes an input file which describes an affine PTS very similar to the definition in this report and generates the corresponding c++ program as well as a list of linear (super) martingale expression maps. These can be used to manually derive probabilistic assertions or find a SMRF to prove almost sure termination.

Although the approach has some limitations for practical use, e.g. due to non-linear update functions, it is an important step towards automated verification of probabilistic programs, especially because of the automated discovery of martingales.

References

[1] Chen, B., Jamieson, K., Balakrishnan, H., Morris, R.: Span: An Energy-Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks. Wireless networks **8**(5) (2002) 481–494

[2] Thrun, S., Fox, D., Burgard, W., Dellaert, F.: Robust Monte Carlo localization for mobile robots. Artificial intelligence **128**(1) (2001) 99–141

[3] Courcoubetis, C., Yannakakis, M.: The Complexity of Probabilistic Verification. Journal of the ACM (JACM) **42**(4) (1995) 857–907

[4] McIver, A., Morgan, C.C.: Abstraction, Refinement and Proof for Probabilistic Systems. Springer (2006)

[5] Chakarov, A., Sankaranarayanan, S.: Probabilistic Program Analysis with Martingales. In: Computer Aided Verification, Springer (2013) 511–526

[6] McIver, A., Morgan, C.: Developing and Reasoning about Probabilistic Programs in pGCL. In: Refinement Techniques in Software Engineering. Springer (2006) 123–155

[7] Williams, D.: Probability with Martingales. Cambridge university press (1991)

[8] Chung, F., Lu, L.: Complex Graphs and Networks. Volume 107 of CBMS-NSF regional conference series in mathematics. American Mathematical Society (2006)

[9] Dubhashi, D.P., Panconesi, A.: Concentration of Measure for the Analysis of Randomized Algorithms. Cambridge University Press (2009)

[10] Chakarov, A., Sankaranarayanan, S.: Expectation Invariants for Probabilistic Program Loops as Fixed Points. In: Static Analysis. Springer (2014) 85–100