Expectation Invariants for Probabilistic Program Loops as Fixed Points

Phillip Florian

February 3, 2015

Introduction

- Probabilistic programs use random variables
  $\Rightarrow$ Properties of probabilistic programs are hard to verify

Introduction

- Probabilistic programs use random variables
  $\Rightarrow$ Properties of probabilistic programs are hard to verify

- Expectation invariants are expressions over the program variables that stay non-negative all the time
  $\Rightarrow$ They can be used to verify properties

Introduction

- Probabilistic programs use random variables
  $\Rightarrow$ Properties of probabilistic programs are hard to verify

- Expectation invariants are expressions over the program variables that stay non-negative all the time
  $\Rightarrow$ They can be used to verify properties

- Compute expectation invariants as fixed points

Introduction

- Probabilistic programs use random variables
  $\Rightarrow$ Properties of probabilistic programs are hard to verify

- Expectation invariants are expressions over the program variables that stay non-negative all the time
  $\Rightarrow$ They can be used to verify properties

- Compute expectation invariants as fixed points
  - The presented algorithm computes a set of expectation invariants
  - Only works with some restrictions to the program

Probabilistic programs

The following notion will be used:

- $\mathcal{P}$: A probabilistic program
- $X = \{x_1, \ldots, x_m\}$: A finite set of program variables
- $R = \{r_1, \ldots, r_l\}$: A finite set of random variables
- $\mathcal{D}_R$: The joint distribution of random variables R
- $\boldsymbol{x}$,$\boldsymbol{r}$: The vectors denoting the valuation of all program and random variables respectively

Probabilistic loops

Definition (Probabilistic loops)

A probabilistic loop of $\mathcal{P}$ is a tuple $\langle \mathcal{T}, \mathcal{D}_0, n \rangle$, with

- $\mathcal{T} : \{\tau_1, \ldots, \tau_k\}$: A finite set of probabilistic transitions
- $\mathcal{D}_0$: The initial probability distribution of the program variables
- n: A loop counter

Probabilistic loops

Definition (Probabilistic loops)

A probabilistic loop of $\mathcal{P}$ is a tuple $\langle \mathcal{T}, \mathcal{D}_0, n \rangle$, with

- $\mathcal{T} : \{\tau_1, \ldots, \tau_k\}$: A finite set of probabilistic transitions
- $\mathcal{D}_0$: The initial probability distribution of the program variables
- n: A loop counter

A probabilistic transition $\tau_i : \langle \mathbf{g}_i, \mathcal{F}_i \rangle$ consists of

- A guard $\mathbf{g}_i(\boldsymbol{x})$ over X
- An update function $\mathcal{F}_i(\boldsymbol{x}, \boldsymbol{r})$ s.t. after taking the transition it holds: $\boldsymbol{x}' = \mathcal{F}_i(\boldsymbol{x}, \boldsymbol{r})$.

Example

```
int x := rand (0,2)
while (x<=10){
   x:= x + rand (0,2)
}
```

Example

```
int x := rand (0,2)
while (x<=10){
   x:= x + rand (0,2)
}
```

Can be expressed as $\langle \mathcal{T}, \mathcal{D}_0, n \rangle$

- $\mathcal{T} = \{\tau_1\}$

Example

```
int x := rand (0,2)
while (x<=10){
    x:= x + rand (0,2)
}
```

Can be expressed as $\langle \mathcal{T}, \mathcal{D}_0, n \rangle$

- $\mathcal{T} = \{\tau_1\}$
  - $\mathbf{g}_1(\boldsymbol{x}) = x \leq 10$

Example

```
int x := rand (0,2)
while (x<=10){
    x:= x + rand (0,2)
}
```

Can be expressed as $\langle \mathcal{T}, \mathcal{D}_0, n \rangle$

- $\mathcal{T} = \{\tau_1\}$
  - $\mathbf{g}_1(\boldsymbol{x}) = x \leq 10$
  - $\mathcal{F}_1(\boldsymbol{x},\boldsymbol{r}) = x + r_1$
    - $r_1 = U(0,2)$

Example

```
int x := rand (0,2)
while (x<=10){
   x:= x + rand (0,2)
}
```

Can be expressed as $\langle \mathcal{T}, \mathcal{D}_0, n \rangle$

- $\mathcal{T} = \{\tau_1\}$
  - $\mathbf{g}_1(\boldsymbol{x}) = x \leq 10$
  - $\mathcal{F}_1(\boldsymbol{x},\boldsymbol{r}) = x + r_1$
    - $r_1 = U(0,2)$
- $D_0 : \langle x \rangle = U[0,2]$

Example

```
int x := rand (0,2)
while (x<=10){
    x:= x + rand (0,2)
}
```

Can be expressed as $\langle \mathcal{T}, \mathcal{D}_0, n \rangle$

- $\mathcal{T} = \{\tau_1\}$
  - $\mathbf{g}_1(\boldsymbol{x}) = x \leq 10$
  - $\mathcal{F}_1(\boldsymbol{x},\boldsymbol{r}) = x + r_1$
    - $r_1 = U(0,2)$
- $D_0 : \langle x \rangle = U[0,2]$
- $n = 0$

Piecewise linear transitions

Definition (Piecewise linear transitions)

$\tau : \langle \mathbf{g}, \mathcal{F}(\boldsymbol{x}, \boldsymbol{r}) \rangle$ is a piecewise linear transition if:

Piecewise linear transitions

Definition (Piecewise linear transitions)

$\tau : \langle \mathbf{g}, \mathcal{F}(\boldsymbol{x}, \boldsymbol{r}) \rangle$ is a piecewise linear transition if:

- $\mathbf{g}$ is a *linear guard* over X

Piecewise linear transitions

Definition (Piecewise linear transitions)

$\tau : \langle \mathbf{g}, \mathcal{F}(\boldsymbol{x}, \boldsymbol{r}) \rangle$ is a piecewise linear transition if:

- $\mathbf{g}$ is a *linear guard* over X
- $\mathcal{F}(\boldsymbol{x}, \boldsymbol{r})$ is a *piecewise linear function* and may be written as:

$$\mathcal{F}(\boldsymbol{x}, \boldsymbol{r}) = \begin{cases} f_1 : A_1\boldsymbol{x} + B_1\boldsymbol{r} + d_1, & \text{with probability } p_1 \\ \vdots & \\ f_k : A_k\boldsymbol{x} + B_k\boldsymbol{r} + d_k, & \text{with probability } p_k \end{cases}$$

Piecewise linear transitions

Definition (Piecewise linear transitions)

$\tau : \langle \mathbf{g}, \mathcal{F}(\boldsymbol{x}, \boldsymbol{r}) \rangle$ is a piecewise linear transition if:

- $\mathbf{g}$ is a *linear guard* over X
- $\mathcal{F}(\boldsymbol{x}, \boldsymbol{r})$ is a *piecewise linear function* and may be written as:

$$\mathcal{F}(\boldsymbol{x}, \boldsymbol{r}) = \begin{cases} f_1 : A_1 \boldsymbol{x} + B_1 \boldsymbol{r} + d_1, & \text{with probability } p_1 \\ \vdots & \\ f_k : A_k \boldsymbol{x} + B_k \boldsymbol{r} + d_k, & \text{with probability } p_k \end{cases}$$

  - $f_1, \ldots, f_k$: Identifier for different outcomes of Bernoulli choices

Piecewise linear transitions

Definition (Piecewise linear transitions)

$\tau : \langle \mathbf{g}, \mathcal{F}(\boldsymbol{x},\boldsymbol{r})\rangle$ is a piecewise linear transition if:

- $\mathbf{g}$ is a *linear guard* over X
- $\mathcal{F}(\boldsymbol{x},\boldsymbol{r})$ is a *piecewise linear function* and may be written as:

$$\mathcal{F}(\boldsymbol{x},\boldsymbol{r}) = \begin{cases} f_1 : A_1\boldsymbol{x} + B_1\boldsymbol{r} + d_1, & \text{with probability } p_1 \\ \vdots & \\ f_k : A_k\boldsymbol{x} + B_k\boldsymbol{r} + d_k, & \text{with probability } p_k \end{cases}$$

  - $f_1, \ldots, f_k$: Identifier for different outcomes of Bernoulli choices
  - $p_1, \ldots, p_k$: Probabilities for choosing the corresponding fork

Piecewise linear transitions

Definition (Piecewise linear transitions)

$\tau : \langle \mathbf{g}, \mathcal{F}(\boldsymbol{x},\boldsymbol{r}) \rangle$ is a piecewise linear transition if:

- $\mathbf{g}$ is a *linear guard* over X
- $\mathcal{F}(\boldsymbol{x},\boldsymbol{r})$ is a *piecewise linear function* and may be written as:

$$\mathcal{F}(\boldsymbol{x},\boldsymbol{r}) = \begin{cases} f_1 : A_1\boldsymbol{x} + B_1\boldsymbol{r} + d_1, & \text{with probability } p_1 \\ \vdots & \\ f_k : A_k\boldsymbol{x} + B_k\boldsymbol{r} + d_k, & \text{with probability } p_k \end{cases}$$

  - $f_1, \ldots, f_k$: Identifier for different outcomes of Bernoulli choices
  - $p_1, \ldots, p_k$: Probabilities for choosing the corresponding fork
  - $A_i \in \mathbb{R}^{m \times m}$, $B_i \in \mathbb{R}^{m \times l}$, $d_i \in \mathbb{R}^m$ are used to model the changes to the program variables occurring in the loop.

Example

```
int x := rand (−5,3)
int y := rand (−3,5)
int c := 0
while (true){
  if(x+y<=10)
    if flip(3/4)
      x:= x + rand (0,2)
      y:= y + 2
    c++
  else
    do nothing
}
```

Example (continued)

The corresponding piecewise linear transition $\tau : \langle \mathbf{g}, \mathcal{F} \rangle$

- $\mathbf{g}(\boldsymbol{x}) = x + y \leq 10$

Example (continued)

The corresponding piecewise linear transition $\tau : \langle \mathbf{g}, \mathcal{F} \rangle$

- $\mathbf{g}(\boldsymbol{x}) = x + y \leq 10$
- $\mathcal{F}(\boldsymbol{x}, \boldsymbol{r}) =$

Example (continued)

The corresponding piecewise linear transition $\tau : \langle \mathbf{g}, \mathcal{F} \rangle$

- $\mathbf{g}(\boldsymbol{x}) = x + y \leq 10$
- $\mathcal{F}(\boldsymbol{x}, \boldsymbol{r}) = \begin{cases} f_1 : \begin{pmatrix} x \\ y \\ c \end{pmatrix} + \begin{pmatrix} r_1 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 2 \\ 1 \end{pmatrix} & , \; p_1 = \frac{3}{4} \\ & \end{cases}$

Example (continued)

The corresponding piecewise linear transition $\tau : \langle \mathbf{g}, \mathcal{F} \rangle$

- $\mathbf{g}(\boldsymbol{x}) = x + y \leq 10$
- $$\mathcal{F}(\boldsymbol{x}, \boldsymbol{r}) = \begin{cases} f_1 : \begin{pmatrix} x \\ y \\ c \end{pmatrix} + \begin{pmatrix} r_1 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 2 \\ 1 \end{pmatrix} & , p_1 = \frac{3}{4} \\ f_2 : \begin{pmatrix} x \\ y \\ c \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} & , p_2 = \frac{1}{4} \end{cases}$$

Restrictions for this approach

Restrictions for this approach

- Probabilistic loops are not nested

Restrictions for this approach

- Probabilistic loops are not nested
    - For simplicity
    - For all nested loops there are equivalent unnested variants

Restrictions for this approach

- Probabilistic loops are not nested
  - For simplicity
  - For all nested loops there are equivalent unnested variants
- All transitions are piecewise linear

Restrictions for this approach

- Probabilistic loops are not nested
  - For simplicity
  - For all nested loops there are equivalent unnested variants
- All transitions are piecewise linear
- Exactly one transition can be taken in every iteration
  - The loop might need to be modified

Restrictions for this approach

- Probabilistic loops are not nested
  - For simplicity
  - For all nested loops there are equivalent unnested variants
- All transitions are piecewise linear
- Exactly one transition can be taken in every iteration
  - The loop might need to be modified
- All expressions $e(\boldsymbol{x})$ are linear expressions
  - $e(\boldsymbol{x}) = c_0 + \sum_{i=0}^{m} \lambda_i \cdot x_i,\ c_0, \lambda_i \in \mathbb{R}$

Execution model

To model an execution of a probabilistic loop we use tuples $(\boldsymbol{x}_n, n)$ as states

Execution model

To model an execution of a probabilistic loop we use tuples $(\boldsymbol{x}_n, n)$ as states, where:

- $\boldsymbol{x}_n$ represents the program variables at loop-iteration n

Execution model

To model an execution of a probabilistic loop we use tuples $(\boldsymbol{x}_n, n)$ as states, where:

- $\boldsymbol{x}_n$ represents the program variables at loop-iteration n
- $(\boldsymbol{x}_0, 0)$ is an *initial state* if $\boldsymbol{x}_0$ is drawn from $\mathcal{D}_0$

Execution model

To model an execution of a probabilistic loop we use tuples $(\boldsymbol{x}_n, n)$ as states, where:

- $\boldsymbol{x}_n$ represents the program variables at loop-iteration n
- $(\boldsymbol{x}_0, 0)$ is an *initial state* if $\boldsymbol{x}_0$ is drawn from $\mathcal{D}_0$
- $(\boldsymbol{x}_i, i)$ is *predecessor* of $(\boldsymbol{x}_{i+1}, i+1)$ if for a transition $\tau : \langle \mathbf{g}, \mathcal{F}(\boldsymbol{x}, \boldsymbol{r}) \rangle$
  - $\boldsymbol{x}_i \models \mathbf{g}$
  - $\exists \boldsymbol{r} \in \mathcal{D}_R$, $\boldsymbol{x}_{i+1} = \mathcal{F}(\boldsymbol{x}_i, \boldsymbol{r})$

Execution model

To model an execution of a probabilistic loop we use tuples $(\boldsymbol{x}_n, n)$ as states, where:

- $\boldsymbol{x}_n$ represents the program variables at loop-iteration n
- $(\boldsymbol{x}_0, 0)$ is an *initial state* if $\boldsymbol{x}_0$ is drawn from $\mathcal{D}_0$
- $(\boldsymbol{x}_i, i)$ is *predecessor* of $(\boldsymbol{x}_{i+1}, i+1)$ if for a transition $\tau : \langle \mathbf{g}, \mathcal{F}(\boldsymbol{x}, \boldsymbol{r})\rangle$
  - $\boldsymbol{x}_i \models \mathbf{g}$
  - $\exists \boldsymbol{r} \in \mathcal{D}_R,\ \boldsymbol{x}_{i+1} = \mathcal{F}(\boldsymbol{x}_i, \boldsymbol{r})$

$\mathcal{D}_i = \{\boldsymbol{x}_i \mid (\boldsymbol{x}_i, i) \text{ is reachable from an initial state}\}$

- D_i is the *distribution* of program variables at iteration i

Example execution

One possible execution:

$((3,3,0)^T,0)$

Example execution

One possible execution:

$$((3,3,0)^T,0) \xrightarrow{\tau_1} ((4,5,1)^T,1)$$

Example execution

One possible execution:

$$((3{,}3{,}0)^T{,}0) \xrightarrow{\tau_1} ((4{,}5{,}1)^T{,}1) \xrightarrow{\tau_1} ((6{,}7{,}2)^T{,}2)$$

Example execution

One possible execution:

$$((3,3,0)^T,0) \xrightarrow{\tau_1} ((4,5,1)^T,1) \xrightarrow{\tau_1}$$
$$((6,7,2)^T,2) \xrightarrow{\tau_2} ((6,7,2)^T,3)$$

Example execution

One possible execution:

$$((3,3,0)^T,0) \xrightarrow{\tau_1} ((4,5,1)^T,1) \xrightarrow{\tau_1}$$
$$((6,7,2)^T,2) \xrightarrow{\tau_2} ((6,7,2)^T,3) \xrightarrow{\tau_2} \ldots$$

Pre-Expectations

- Assume we are currently in state $(\boldsymbol{x}, n)$

Pre-Expectations

- Assume we are currently in state $(\boldsymbol{x}, n)$
- What is the expected value of $e(\boldsymbol{x}')$ evaluated over all successor states $(\boldsymbol{x}', n+1)$
  - With respect to a single transition?
  - With respect to all transitions?

Pre-Expectations

- Assume we are currently in state $(\boldsymbol{x}, n)$
- What is the expected value of $e(\boldsymbol{x}')$ evaluated over all successor states $(\boldsymbol{x}', n+1)$
  - With respect to a single transition?
  - With respect to all transitions?

$\Rightarrow$ Pre-expectation of $e(\boldsymbol{x}')$

Pre-Expectation

Definition (Pre-expectation for fixed PWL transitions)

For a PWL transition τ the pre-expectation operator can be written as:

$$\mathrm{pre}\mathbb{E}_\tau(e(\boldsymbol{x}')) = \sum_{j=1}^{k} p_j \mathbb{E}_R(\mathrm{pre}(e(\boldsymbol{x}'), f_j) \mid \boldsymbol{x})$$

Pre-Expectation

Definition (Pre-expectation for fixed PWL transitions)

For a PWL transition τ the pre-expectation operator can be written as:

$$\mathrm{pre}\mathbb{E}_\tau(e(\boldsymbol{x}')) = \sum_{j=1}^{k} p_j \mathbb{E}_R(\mathrm{pre}(e(\boldsymbol{x}'), f_j) \mid \boldsymbol{x})$$

where $\mathrm{pre}(e(\boldsymbol{x}'), f_j)$ denotes the expression obtained by applying f_j to all variables of $\boldsymbol{x}$ occurring in $e(\boldsymbol{x})$.
$\mathbb{E}_R(\boldsymbol{r})$ denotes the expectation of $\boldsymbol{r}$ over $\mathcal{D}_R$.

Example

$e(\boldsymbol{x}') = 1 + 2x' - 3y'$

Example

$e(\boldsymbol{x}') = 1 + 2x' - 3y'$

$\tau_1 : \langle \mathbf{g}_1, \mathcal{F}_1 \rangle$ with:

- $\mathbf{g}_1 : x + y \leq 10$
- $\mathcal{F}_1(\boldsymbol{x}, \boldsymbol{r}) = \begin{cases} f_1 : \begin{pmatrix} x \\ y \\ c \end{pmatrix} + \begin{pmatrix} r_1 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 2 \\ 1 \end{pmatrix} & , \ p_1 = \frac{3}{4} \\ f_2 : \begin{pmatrix} x \\ y \\ c \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} & , \ p_2 = \frac{1}{4} \end{cases}$

Example (continued)

$$\mathrm{pre}\mathbb{E}_{\tau_1}(1 + 2x' - 3y') =$$

Example (continued)

$$\mathrm{pre}\mathbb{E}_{\tau_1}(1 + 2x' - 3y') = \sum_{j=1}^{2} p_j \cdot \mathbb{E}_{\mathcal{D}_R}(\mathrm{pre}(1 + 2x' - 3y',\ f_j) \mid \boldsymbol{x})$$

Example (continued)

$$
\begin{aligned}
\mathrm{pre}\mathbb{E}_{\tau_1}(1 + 2x' - 3y') &= \sum_{j=1}^{2} p_j \cdot \mathbb{E}_{\mathcal{D}_R}(\mathrm{pre}(1 + 2x' - 3y',\ f_j) \mid \boldsymbol{x}) \\
&= \frac{3}{4} \cdot \mathbb{E}_{\mathcal{D}_R}(1 + 2 \cdot (x + r_1) - 3 \cdot (y + 2)) \\
&\qquad + \frac{1}{4} \cdot \mathbb{E}_{\mathcal{D}_R}(1 + 2x - 3y)
\end{aligned}
$$

Example (continued)

$$
\begin{aligned}
\mathrm{pre}\mathbb{E}_{\tau_1}(1 + 2x' - 3y') &= \sum_{j=1}^{2} p_j \cdot \mathbb{E}_{\mathcal{D}_R}(\mathrm{pre}(1 + 2x' - 3y',\ f_j) \mid \boldsymbol{x}) \\
&= \frac{3}{4} \cdot \mathbb{E}_{\mathcal{D}_R}(1 + 2 \cdot (x + r_1) - 3 \cdot (y + 2)) \\
&\qquad + \frac{1}{4} \cdot \mathbb{E}_{\mathcal{D}_R}(1 + 2x - 3y) \\
&= -\frac{7}{2} + 2x - 3y + \frac{3}{2} \cdot \mathbb{E}_R(r_1)
\end{aligned}
$$

Example (continued)

$$\begin{aligned}
\text{pre}\mathbb{E}_{\tau_1}(1 + 2x' - 3y') &= \sum_{j=1}^{2} p_j \cdot \mathbb{E}_{\mathcal{D}_R}(\text{pre}(1 + 2x' - 3y',\ f_j) \mid \boldsymbol{x}) \\
&= \frac{3}{4} \cdot \mathbb{E}_{\mathcal{D}_R}(1 + 2 \cdot (x + r_1) - 3 \cdot (y + 2)) \\
&\quad + \frac{1}{4} \cdot \mathbb{E}_{\mathcal{D}_R}(1 + 2x - 3y) \\
&= -\frac{7}{2} + 2x - 3y + \frac{3}{2} \cdot \mathbb{E}_R(r_1) \\
r_1 &= U[0,2] \Rightarrow \mathbb{E}_R(r_1) = 1
\end{aligned}$$

Example (continued)

$$
\begin{aligned}
\mathrm{pre}\mathbb{E}_{\tau_1}(1 + 2x' - 3y') &= \sum_{j=1}^{2} p_j \cdot \mathbb{E}_{\mathcal{D}_R}(\mathrm{pre}(1 + 2x' - 3y',\ f_j) \mid \boldsymbol{x}) \\
&= \frac{3}{4} \cdot \mathbb{E}_{\mathcal{D}_R}(1 + 2 \cdot (x + r_1) - 3 \cdot (y + 2)) \\
&\qquad + \frac{1}{4} \cdot \mathbb{E}_{\mathcal{D}_R}(1 + 2x - 3y) \\
&= -\frac{7}{2} + 2x - 3y + \frac{3}{2} \cdot \mathbb{E}_R(r_1) \\
r_1 &= U[0,2] \Rightarrow \mathbb{E}_R(r_1) = 1 \\
\mathrm{pre}\mathbb{E}_{\tau_1}(1 + 2x - 3y) &= -2 + 2x - 3y
\end{aligned}
$$

Pre-expectation (continued)

Definition (Pre-expectation over all transitions)

The expected value of e over the post-state distribution starting from state $(\boldsymbol{x}_n, n)$ is the value of the pre-expectation $\mathsf{pre}\mathbb{E}(e')$ evaluated over the current state $(\boldsymbol{x}_n, n)$:

$$\mathbb{E}_{\mathcal{D}_n}(e) = \mathsf{pre}\mathbb{E}(e') = \sum_{\tau_i \in \mathcal{T}} \mathbb{1}_{\mathbf{g}_{\tau_i}}(\boldsymbol{x}_n) \cdot \mathsf{pre}\mathbb{E}_{\tau_i}(e')$$

Expectation Invariants

Definition (Expectation invariants)

An expression e over the program variables X is called an expectation invariant (EI) if and only if $\mathbb{E}_{\mathcal{D}_i}(e) \geq 0$ for all $i \geq 0$.

Expectation Invariants

Definition (Expectation invariants)

An expression e over the program variables X is called an expectation invariant (EI) if and only if $\mathbb{E}_{\mathcal{D}_i}(e) \geq 0$ for all $i \geq 0$.

Example

We show that $e(\boldsymbol{x}) = 2y - x$ is an expectation invariant.

Expectation Invariants

Definition (Expectation invariants)

An expression e over the program variables X is called an expectation invariant (EI) if and only if $\mathbb{E}_{\mathcal{D}_i}(e) \geq 0$ for all $i \geq 0$.

Example

We show that $e(\boldsymbol{x}) = 2y - x$ is an expectation invariant.

1. $\mathbb{E}_{\mathcal{D}_0}(2y - x) = 2 \cdot \mathbb{E}_{\mathcal{D}_0}(y) - \mathbb{E}_{\mathcal{D}_0}(x) = 3 \geq 0$

Expectation Invariants

Definition (Expectation invariants)

An expression e over the program variables X is called an expectation invariant (EI) if and only if $\mathbb{E}_{\mathcal{D}_i}(e) \geq 0$ for all $i \geq 0$.

Example

We show that $e(\boldsymbol{x}) = 2y - x$ is an expectation invariant.

1. $\mathbb{E}_{\mathcal{D}_0}(2y - x) = 2 \cdot \mathbb{E}_{\mathcal{D}_0}(y) - \mathbb{E}_{\mathcal{D}_0}(x) = 3 \geq 0$
2. $\mathbb{E}_{\mathcal{D}_i}(2y - x) = 2 \cdot \mathbb{E}_{\mathcal{D}_i}(y) - \mathbb{E}_{\mathcal{D}_i}(x) \geq 0$ for all $i \geq 0$ as $\mathbb{E}_{\mathcal{D}_i}(y)$ is always larger than $\mathbb{E}_{\mathcal{D}_i}(x)$

Expectation Invariants

Definition (Expectation invariants)

An expression e over the program variables X is called an expectation invariant (EI) if and only if $\mathbb{E}_{\mathcal{D}_i}(e) \geq 0$ for all $i \geq 0$.

Example

We show that $e(\boldsymbol{x}) = 2y - x$ is an expectation invariant.

1. $\mathbb{E}_{\mathcal{D}_0}(2y - x) = 2 \cdot \mathbb{E}_{\mathcal{D}_0}(y) - \mathbb{E}_{\mathcal{D}_0}(x) = 3 \geq 0$
2. $\mathbb{E}_{\mathcal{D}_i}(2y - x) = 2 \cdot \mathbb{E}_{\mathcal{D}_i}(y) - \mathbb{E}_{\mathcal{D}_i}(x) \geq 0$ for all $i \geq 0$ as $\mathbb{E}_{\mathcal{D}_i}(y)$ is always larger than $\mathbb{E}_{\mathcal{D}_i}(x)$

$\Rightarrow$ e is an expectation invariant of $\mathcal{P}$

Inductive Expectation Invariants

Definition (Inductive expectation invariants)

Let $E = \{e_1, \ldots, e_k\}$ be a set of expressions. The set E forms an inductive expectation invariant iff for each e_j, $j \in [1,k]$ the following holds:

1. $\mathbb{E}_{\mathcal{D}_0}(e_j) \geq 0$
2. $\text{pre}\mathbb{E}(e_j) = \lambda_0 + \sum_{i=1}^{k} \lambda_i e_i, \ \lambda_i, \geq 0$

Inductive Expectation Invariants

Definition (Inductive expectation invariants)

Let $E = \{e_1, \ldots, e_k\}$ be a set of expressions. The set E forms an inductive expectation invariant iff for each e_j, $j \in [1,k]$ the following holds:

1. $\mathbb{E}_{\mathcal{D}_0}(e_j) \geq 0$
2. $\text{pre}\mathbb{E}(e_j) = \lambda_0 + \sum_{i=1}^{k} \lambda_i e_i,\ \lambda_i, \geq 0$

Theorem

Let $E : \{e_1, \ldots, e_m\}$ be an inductive expectation invariant. It follows that each $e_j \in E$ is an expectation invariant.

Cones of expressions

Definition (Cones)

Let $E = \{e_1, \ldots, e_k\}$ be a finite set of program expressions over the program variables $\boldsymbol{x}$. The set of conic combinations (the finitely generated cone) of E is defined as

$$\mathrm{Cone}(E) = \left\{ \lambda_0 + \sum_{i=1}^{k} \lambda_i e_i \mid \lambda_i \in \mathbb{R}^+,\, 0 \leq i \leq k \right\}$$

Cones of expressions

Definition (Cones)

Let $E = \{e_1, \ldots, e_k\}$ be a finite set of program expressions over the program variables $\boldsymbol{x}$. The set of conic combinations (the finitely generated cone) of E is defined as

$$\text{Cone}(E) = \left\{ \lambda_0 + \sum_{i=1}^{k} \lambda_i e_i \mid \lambda_i \in \mathbb{R}^+, \, 0 \leq i \leq k \right\}$$

Theorem

If E is an inductive expectation invariant, then $e \in Cone(E)$ is an expectation invariant.

Example

$E = \{e_1 : y - x,\ e_2 : 2y + c\}$

- Without proof e_1, e_2 are EIs

Example

$E = \{e_1 : y - x,\ e_2 : 2y + c\}$

- Without proof e_1, e_2 are EIs

Consider $e = 4y - 2x + c = 2 \cdot e_1 + e_2$

Example

$E = \{e_1 : y - x,\ e_2 : 2y + c\}$

- Without proof e_1, e_2 are EIs

Consider $e = 4y - 2x + c = 2 \cdot e_1 + e_2$

$\Rightarrow e \in \text{Cone}(E)$

Example

$E = \{e_1 : y - x,\ e_2 : 2y + c\}$

- Without proof e_1, e_2 are EIs

Consider $e = 4y - 2x + c = 2 \cdot e_1 + e_2$

$\Rightarrow e \in \mathrm{Cone}(E)$

$\Rightarrow e$ is an EI

Pre-Expectation of cones

Definition (Pre-expectation over a single transitions)

Let $E = \{e_1, \ldots, e_m\}$ be a set of expressions, and let $\tau : \langle \mathbf{g}, \mathcal{F} \rangle$ be a transition. The pre-expectation of a cone I : Cone(E) with respect to τ is defined as:

$$\mathrm{pre}\mathbb{E}_\tau(I) = \{(e, \boldsymbol{\lambda}) \in \mathbb{A}(\boldsymbol{x}) \times \mathbb{R}^m \mid \boldsymbol{\lambda} \geq 0 \wedge \exists \mu \geq 0 (\mathrm{pre}\mathbb{E}_\tau(e) \equiv \sum_{j=1}^{m} \lambda_j e_j + \mu)\}$$

Pre-Expectation of cones (continued)

Definition (Pre-Expectation over all transitions)

Let I be a finitely generated cone of expressions. The pre-expectation over all transitions in $\mathcal{T} = \{\tau_1, \ldots, \tau_k\}$ can be computed as:

$$\mathrm{pre}\mathbb{E}(I) = \{e \in \mathbb{A}(\boldsymbol{x}) \mid \exists \boldsymbol{\lambda} \geq 0 (e, \boldsymbol{\lambda}) \in \bigcap_{j=1}^{k} \mathrm{pre}\mathbb{E}_{\tau_j}(I)\}$$

Fixed points for Expectation Invariants

Algorithm

Fixed points for Expectation Invariants

Algorithm

- Create the initial cone I_0

$$I_0 : \mathrm{Cone}(\{1, x_1 - \mathbb{E}_{\mathcal{D}_0}(x_1), \mathbb{E}_{\mathcal{D}_0}(x_1) - x_1, \ldots, x_n - \mathbb{E}_{\mathcal{D}_0}(x_n), \mathbb{E}_{\mathcal{D}_0}(x_n) - x_n\})$$

Fixed points for Expectation Invariants

Algorithm

- Create the initial cone I_0

$$I_0 : \mathrm{Cone}(\{1, x_1 - \mathbb{E}_{\mathcal{D}_0}(x_1), \mathbb{E}_{\mathcal{D}_0}(x_1) - x_1, \ldots, x_n - \mathbb{E}_{\mathcal{D}_0}(x_n), \mathbb{E}_{\mathcal{D}_0}(x_n) - x_n\})$$

- In each iteration
  - Compute $\mathrm{pre}\mathbb{E}(I_n)$
  - Compute $I_{n+1} = \mathrm{pre}\mathbb{E}(I_n) \cap I_0$

Fixed points for Expectation Invariants

Algorithm

- Create the initial cone I_0

$$I_0 : \mathrm{Cone}(\{1, x_1 - \mathbb{E}_{\mathcal{D}_0}(x_1), \mathbb{E}_{\mathcal{D}_0}(x_1) - x_1, \ldots, x_n - \mathbb{E}_{\mathcal{D}_0}(x_n), \mathbb{E}_{\mathcal{D}_0}(x_n) - x_n\})$$

- In each iteration
  - Compute $\mathrm{pre}\mathbb{E}(I_n)$
  - Compute $I_{n+1} = \mathrm{pre}\mathbb{E}(I_n) \cap I_0$
- Repeat until $I^* = I_{n+1} = I_n$
  - Might not converge

Fixed points for Expectation Invariants

Algorithm

- Create the initial cone I_0

$$I_0 : \mathrm{Cone}(\{1, x_1 - \mathbb{E}_{\mathcal{D}_0}(x_1), \mathbb{E}_{\mathcal{D}_0}(x_1) - x_1, \ldots, x_n - \mathbb{E}_{\mathcal{D}_0}(x_n), \mathbb{E}_{\mathcal{D}_0}(x_n) - x_n\})$$

- In each iteration
  - Compute $\mathrm{pre}\mathbb{E}(I_n)$
  - Compute $I_{n+1} = \mathrm{pre}\mathbb{E}(I_n) \cap I_0$
- Repeat until $I^* = I_{n+1} = I_n$
  - Might not converge

$\Rightarrow$ Resulting cone I^* contains only EIs

Example

$$I_0 = \mathrm{Cone}(\{1, x - \mathbb{E}_{\mathcal{D}_0}(x), \mathbb{E}_{\mathcal{D}_0}(x) - x, \mathbb{E}_{\mathcal{D}_0}(y) - y, y - \mathbb{E}_{\mathcal{D}_0}(y), \mathbb{E}_{\mathcal{D}_0}(c) - c, c - \mathbb{E}_{\mathcal{D}_0}(c)\})$$

Example

$$
\begin{aligned}
I_0 = \text{Cone}(\{&1, x - \mathbb{E}_{\mathcal{D}_0}(x), \mathbb{E}_{\mathcal{D}_0}(x) - x, \mathbb{E}_{\mathcal{D}_0}(y) - y, y - \mathbb{E}_{\mathcal{D}_0}(y), \\
&\mathbb{E}_{\mathcal{D}_0}(c) - c, c - \mathbb{E}_{\mathcal{D}_0}(c)\}) \\
= \text{Cone}(\{&1, x + 1, -1 - x, y - 1, 1 - y, c, -c\})
\end{aligned}
$$

Example

$$
\begin{aligned}
I_0 &= \mathrm{Cone}(\{1, x - \mathbb{E}_{\mathcal{D}_0}(x), \mathbb{E}_{\mathcal{D}_0}(x) - x, \mathbb{E}_{\mathcal{D}_0}(y) - y, y - \mathbb{E}_{\mathcal{D}_0}(y), \\
&\qquad \mathbb{E}_{\mathcal{D}_0}(c) - c, c - \mathbb{E}_{\mathcal{D}_0}(c)\}) \\
&= \mathrm{Cone}(\{1, x + 1, -1 - x, y - 1, 1 - y, c, -c\}) \\
\mathrm{pre}\mathbb{E}_{\tau_1}(I_0) &= \{(1, (1,0,0,0,0,0)^T), (x + 1, (0.75,1,0,0,0,0)^T), \\
&\qquad (y - 1, (2.5,0,1,0,0,0)^T), (1 - y, (1.5,0,0,1,0,0)^T), \\
&\qquad (c, (1,0,0,0,1,0)^T), \ldots\}
\end{aligned}
$$

Example

$$
\begin{aligned}
I_0 &= \text{Cone}(\{1, x - \mathbb{E}_{\mathcal{D}_0}(x), \mathbb{E}_{\mathcal{D}_0}(x) - x, \mathbb{E}_{\mathcal{D}_0}(y) - y, y - \mathbb{E}_{\mathcal{D}_0}(y), \\
&\qquad \mathbb{E}_{\mathcal{D}_0}(c) - c, c - \mathbb{E}_{\mathcal{D}_0}(c)\}) \\
&= \text{Cone}(\{1, x+1, -1-x, y-1, 1-y, c, -c\}) \\
\text{pre}\mathbb{E}_{\tau_1}(I_0) &= \{(1, (1,0,0,0,0,0)^T), (x+1, (0.75,1,0,0,0,0)^T), \\
&\quad (y-1, (2.5,0,1,0,0,0)^T), (1-y, (1.5,0,0,1,0,0)^T), \\
&\quad (c, (1,0,0,0,1,0)^T), \ldots\} \\
\text{pre}\mathbb{E}_{\tau_2}(I_0) &= \{(e, \lambda) \mid \forall e \in I_0,\ \lambda \text{ corresponding to } e\}
\end{aligned}
$$

Example

$$\begin{aligned}
I_0 &= \mathrm{Cone}(\{1,x-\mathbb{E}_{\mathcal{D}_0}(x),\mathbb{E}_{\mathcal{D}_0}(x)-x,\mathbb{E}_{\mathcal{D}_0}(y)-y,y-\mathbb{E}_{\mathcal{D}_0}(y),\\
&\qquad \mathbb{E}_{\mathcal{D}_0}(c)-c,c-\mathbb{E}_{\mathcal{D}_0}(c)\})\\
&= \mathrm{Cone}(\{1,x+1,-1-x,y-1,1-y,c,-c\})\\
\mathrm{pre}\mathbb{E}_{\tau_1}(I_0) &= \{(1,(1,0,0,0,0,0)^T),(x+1,(0.75,1,0,0,0,0)^T),\\
&\quad (y-1,(2.5,0,1,0,0,0)^T),(1-y,(1.5,0,0,1,0,0)^T),\\
&\quad (c,(1,0,0,0,1,0)^T),\ldots\}\\
\mathrm{pre}\mathbb{E}_{\tau_2}(I_0) &= \{(e,\lambda) \mid \forall e\in I_0,\ \lambda \text{ corresponding to } e\}\\
\mathrm{pre}\mathbb{E}(I_0) &= \{1,x+1,y-1,1-y,c\}
\end{aligned}$$

Example

$$
\begin{aligned}
I_0 &= \mathrm{Cone}(\{1, x - \mathbb{E}_{\mathcal{D}_0}(x), \mathbb{E}_{\mathcal{D}_0}(x) - x, \mathbb{E}_{\mathcal{D}_0}(y) - y, y - \mathbb{E}_{\mathcal{D}_0}(y), \\
&\qquad \mathbb{E}_{\mathcal{D}_0}(c) - c, c - \mathbb{E}_{\mathcal{D}_0}(c)\}) \\
&= \mathrm{Cone}(\{1, x + 1, -1 - x, y - 1, 1 - y, c, -c\}) \\
\mathrm{pre}\mathbb{E}_{\tau_1}(I_0) &= \{(1, (1,0,0,0,0,0)^T), (x + 1, (0.75,1,0,0,0,0)^T), \\
&\qquad (y - 1, (2.5,0,1,0,0,0)^T), (1 - y, (1.5,0,0,1,0,0)^T), \\
&\qquad (c, (1,0,0,0,1,0)^T), \ldots\} \\
\mathrm{pre}\mathbb{E}_{\tau_2}(I_0) &= \{(e, \lambda) \mid \forall e \in I_0,\ \lambda \text{ corresponding to } e\} \\
\mathrm{pre}\mathbb{E}(I_0) &= \{1, x + 1, y - 1, 1 - y, c\} \\
I_1 &= I_0 \cap \mathrm{pre}\mathbb{E}(I_0) = \{1, x + 1, y - 1, 1 - y, c\}
\end{aligned}
$$

Example

$$
\begin{aligned}
I_0 &= \mathrm{Cone}(\{1, x - \mathbb{E}_{\mathcal{D}_0}(x), \mathbb{E}_{\mathcal{D}_0}(x) - x, \mathbb{E}_{\mathcal{D}_0}(y) - y, y - \mathbb{E}_{\mathcal{D}_0}(y), \\
&\qquad \mathbb{E}_{\mathcal{D}_0}(c) - c, c - \mathbb{E}_{\mathcal{D}_0}(c)\}) \\
&= \mathrm{Cone}(\{1, x+1, -1-x, y-1, 1-y, c, -c\}) \\
\mathrm{pre}\mathbb{E}_{\tau_1}(I_0) &= \{(1, (1,0,0,0,0,0)^T), (x+1, (0.75,1,0,0,0,0)^T), \\
&\qquad (y-1, (2.5,0,1,0,0,0)^T), (1-y, (1.5,0,0,1,0,0)^T), \\
&\qquad (c, (1,0,0,0,1,0)^T), \ldots\} \\
\mathrm{pre}\mathbb{E}_{\tau_2}(I_0) &= \{(e, \lambda) \mid \forall e \in I_0,\ \lambda \text{ corresponding to } e\} \\
\mathrm{pre}\mathbb{E}(I_0) &= \{1, x+1, y-1, 1-y, c\} \\
I_1 &= I_0 \cap \mathrm{pre}\mathbb{E}(I_0) = \{1, x+1, y-1, 1-y, c\} \\
I^* &= I_1 = I_1 \cap \mathrm{pre}\mathbb{E}(I_0)
\end{aligned}
$$

Standard dual widening

Definition (Standard dual widening)

Let $I_1 = \mathrm{Cone}(e_1, \ldots, e_k)$ and $I_2 = \mathrm{Cone}(g_1, \ldots, g_k)$ be two finitely generated cones such that $I_1 \supseteq I_2$.
The dual widening operator $I_1 \widetilde{\nabla} I_2$ is defined as $I = \mathrm{Cone}(g_i \mid g_i \in I_2)$.
Cone I is the cone generated by generators of I_1 that are also in I_2

Standard dual widening

Definition (Standard dual widening)

Let $I_1 = \text{Cone}(e_1, \dots, e_k)$ and $I_2 = \text{Cone}(g_1, \dots, g_k)$ be two finitely generated cones such that $I_1 \supseteq I_2$.
The dual widening operator $I_1 \widetilde{\nabla} I_2$ is defined as $I = \text{Cone}(g_i \mid g_i \in I_2)$.
Cone I is the cone generated by generators of I_1 that are also in I_2

- If $\widetilde{\nabla}$ is applied to two successive cones in the algorithm the convergence is ensured

Example

Assume $\mathcal{P}$ is a probabilistic program s.t. $\pm(x-1) \in I_0$ and the pre-expectations for these expressions alternate in each iteration.

Example

Assume $\mathcal{P}$ is a probabilistic program s.t. $\pm(x-1) \in I_0$ and the pre-expectations for these expressions alternate in each iteration.

$$I_1 = \text{Cone}\{1, x-1\} = I_3 = \cdots$$
$$I_2 = \text{Cone}\{1, -x+1\} = I_4 = \cdots$$
$$I_1 \widetilde{\nabla} I_2 = \text{Cone}\{1\} = I^*$$

Example

Assume $\mathcal{P}$ is a probabilistic program s.t. $\pm(x-1) \in I_0$ and the pre-expectations for these expressions alternate in each iteration.

$$
\begin{aligned}
I_1 &= \mathrm{Cone}\{1, x-1\} = I_3 = \cdots \\
I_2 &= \mathrm{Cone}\{1, -x+1\} = I_4 = \cdots \\
I_1 \widetilde{\nabla} I_2 &= \mathrm{Cone}\{1\} = I^*
\end{aligned}
$$

$\Rightarrow$ The problem of alternation is solved

Experimental results from [2]

$|X|$: Number of program variables
$\#$: Number of needed iterations
Time: Runtime in seconds

$|\mathcal{T}|$: Number of transitions
$\widetilde{\nabla}$: Dual widening used or not
$\varepsilon = 0.05s$

Name	$\lvert X\rvert$	$\lvert\mathcal{T}\rvert$	$\#$	$\widetilde{\nabla}$	Time
MOT-EXAMPLE	3	2	2	No	$\leq \varepsilon$
MOT-EX-LOOP-INV	3	2	2	No	0.10
MOT-EX-POLY	9	2	2	No	0.18
2D-WALK	4	4	7	Yes	$\leq \varepsilon$
AGGREGATE-RV	3	2	2	No	$\leq \varepsilon$
HARE-TURTLE	3	2	2	No	$\leq \varepsilon$
COUPON5	2	5	2	No	$\leq \varepsilon$
HAWK-DOVE-FAIR	6	2	2	No	$\leq \varepsilon$
HAWK-DOVE-BIAS	6	2	2	No	$\leq \varepsilon$
FAULTY-INCR	2	2	7	Yes	$\leq \varepsilon$

Comparrison with PRINSYS

- Out of 26 tests only 3 IEI could be found by PRINSYS
- 26 IEI could be found with this approach
- Not checked whether PRINSYS finds IEI, that this approach does not find

Conclusion

+ Expectation invariants can be found fast
+ Mostly without usage of dual widening

- Unknown time complexity
- Vague descriptions in the paper
- Implementation is not sufficiently tested

Thanks for your attention!

If you have questions, feel free to ask.

Sources

- Aleksandar Chakarov and Sriram Sankaranarayanan.
  Expectation invariants for probabilistic program loops as fixed points.
  In *Static Analysis - 21st International Symposium, SAS 2014, Munich, Germany, September 11-13, 2014. Proceedings*, pages 85–100, 2014.

- Aleksander Chakarov and Sriram Sankaranarayanan.
  Expectation invariants for probabilistic program loops as fixed points (extended version).
  Technical report, University of Colerado, 2014.

- Thomas G. Dietterich.
  Ensemble methods in machine learning.
  In *Multiple Classifier Systems, First International Workshop, MCS 2000, Cagliari, Italy, June 21-23, 2000, Proceedings*, pages 1–15, 2000.

Dexter Kozen.
Semantics of probabilistic programs.
J. Comput. Syst. Sci., 22(3):328–350, 1981.

Piotr Mardziel, Stephen Magill, Michael Hicks, and Mudhakar Srivatsa.
Dynamic enforcement of knowledge-based security policies.
In *Proceedings of the 24th IEEE Computer Security Foundations Symposium, CSF 2011, Cernay-la-Ville, France, 27-29 June, 2011*, pages 114–128, 2011.

Rajeev Motwani and Prabhakar Raghavan.
Randomized Algorithms.
Cambridge University Press, 1995.

Dealing with finite loops

- Need to guarantee that exactly one transition can be taken in every iteration
- No problem for infinite loops
- Finite loops need to be modified
  1. Create an infinite loop
  2. Create an if-statement inside
  3. If the original loop-guard is valid execute original loop-body
  4. Else preserve all program variables

  $\Rightarrow$ New transition that can be taken after the original loop would have been exited

Example

```
int x := rand (−5,3)
int y := rand (−3,5)
int c := 0
while (x+y<=10){
  if flip(3/4)
    x:= x + rand (0,2)
    y:= y + 2
  c++
}
```

```
int x := rand (−5,3)
int y := rand (−3,5)
int c := 0
while (true){
  if(x+y<=10)
    if flip(3/4)
      x:= x + rand (0,2)
      y:= y + 2
    c++
  else
    do nothing
}
```