Probabilistic Programming

Lecture #8+#9: Loop Invariants

Joost-Pieter Katoen

RWTH Lecture Series on Probabilistic Programming 2018

Overview

Overview

Motivation

- Reasoning about loops is the hardest task in program verification

- Why?
  - Weakest preconditions of loops are defined as fixed points
  - They can be approximated iteratively
  - But: Recognise a pattern to yield a closed-form formula for taking a loop n times
  - Taking the limit yields the required fixed point

  These last two steps are the source of undecidability

- "Practical" approach: capture the effect of a loop by a loop invariant

A loop invariant is a property of a program loop that is true before (and after) each iteration.

Loop invariants à la Dijkstra, Floyd, Hoare

Loop invariants

Recall that for while-loops we have:

$$wlp(\text{while}(G)\{P\}, F) \;=\; \text{gfp}\, X.\ (G \wedge wlp(P, X)) \vee (\neg G \wedge F))$$

To determine the effect of a while-loop, one exploits an "invariant" $I \in \mathbb{P}$

Loop invariants

Recall that for while-loops we have:

$$wlp(\text{while}(G)\{P\}, F) \;=\; \text{gfp}\; X.\; (G \wedge wlp(P, X)) \vee (\neg G \wedge F)$$

To determine the effect of a while-loop, one exploits an "invariant" $I \in \mathbb{P}$

Loop invariant

Predicate $I \in \mathbb{P}$ is a loop invariant if it satisfies:

1. $G \Rightarrow I$
2. $\neg G \wedge I \Rightarrow F$, and
3. $G \wedge I \Rightarrow wlp(P, I)$.

Satisfaction of I is invariant under (guarded) iteration of the loop body P.

while $(x < 10)$ { $x{+}{+}$ }

$F = (x = 10)$ $\qquad$ $G = x < 10$

$I = x \leq 10$ is a loop invariant.

✓ a. $x < 10 \Rightarrow x \leq 10$

✓ b. $\neg(x < 10) \wedge x \leq 10 \Rightarrow x = 10$

✓ c. $\underbrace{x < 10 \wedge x \leq 10}_{x < 10} \Rightarrow wlp(\underbrace{P}_{x{+}{+}}, \underbrace{x \leq 10}_{= I})$

Characteristic functions for loops

Recall:

$$wp(\text{while } (G)\{\, P \,\}, f) \;=\; \text{lfp } X.\; \underbrace{([G] \cdot wp(P, X) + [\neg G] \cdot f)}_{\text{characteristic function } \Phi_f(X) \text{ for wp}}$$

and

$$wlp(\text{while } (G)\{\, P \,\}, f) \;=\; \text{gfp } X.\; \underbrace{([G] \cdot wlp(P, X) + [\neg G] \cdot f)}_{\text{characteristic function } \Psi_f(X) \text{ for wlp}}$$

Loop invariants

so these are predicates (no expectations)

wlp

For $I, F \in \mathbb{P}$ and probabilistic loop while$(G)\{P\}$ it holds:

$$\underbrace{(\neg G \wedge I \Rightarrow F \text{ and } G \wedge I \Rightarrow wlp(P, I))}_{\text{conditions 2.+3. of being an invariant}} \quad \text{iff} \quad [I] \sqsubseteq \Psi_{[F]}([I])$$

where $\Psi_{[F]}$ is the wlp-characteristic function of the probabilistic loop for postcondition $[F]$.

Proof.

On the black board. □

Example of a loop invariant.

$$[I] \leq \Psi_{[F]}([I])$$

$$\neg G \wedge I \Rightarrow F$$

iff $[\neg G] \cdot [I] \leq [F]$

iff $[\neg G] \cdot [I] \leq [\neg G] \cdot [F]$

Now derive:

$$G \wedge I \Rightarrow wlp(P, I)$$

iff $[G] \cdot [I] \leq wlp(P, [I])$

iff $[G] \cdot [I] \leq [G] \cdot wlp(P, [I])$

iff $[\neg G] \cdot [I] + [G \cdot I] \leq [G] \cdot wlp(P, [I])$

iff $[I] \leq \Phi_{[F]}([I])$ $\quad \underbrace{+ [\neg G] \cdot [F]}_{= \Phi_{[F]}([I])}$

Overview

Probabilistic invariants

[G]·wp(P,X) + [¬G]·f

Probabilistic invariants

Let Φ_f be the wp-characteristic function of $P' = \text{while}(G)\{P\}$ with respect to post-expectation $f \in \mathbb{E}$ and let $I \in \mathbb{E}$. Then:

Probabilistic invariants

Probabilistic invariants

Let Φ_f be the wp-characteristic function of $P' = \text{while}(G)\{P\}$ with respect to post-expectation $f \in \mathbb{E}$ and let $I \in \mathbb{E}$. Then:

1. I is a wp-superinvariant of P' w.r.t. f iff $\Phi_f(I) \leq I$.

Probabilistic invariants

Probabilistic invariants

Let Φ_f be the wp-characteristic function of P' = while$(G)\{P\}$ with respect to post-expectation $f \in \mathbb{E}$ and let $I \in \mathbb{E}$. Then:

1. I is a wp-superinvariant of P' w.r.t. f iff $\Phi_f(I) \leq I$.
2. I is a wp-subinvariant of P' w.r.t. f iff $I \leq \Phi_f(I)$.

Probabilistic invariants

Probabilistic invariants

Let Φ_f be the wp-characteristic function of $P' = \text{while}(G)\{P\}$ with respect to post-expectation $f \in \mathbb{E}$ and let $I \in \mathbb{E}$. Then:

1. I is a wp-superinvariant of P' w.r.t. f iff $\Phi_f(I) \leq I$.
2. I is a wp-subinvariant of P' w.r.t. f iff $I \leq \Phi_f(I)$.

Sub- and superinvariants for wlp are defined analogously (but are bounded, i.e., then $I \in \mathbb{E}_{\leq 1}$, and Φ_f is replaced by Ψ_f.)

Lemma

$[G] \cdot I \leq wp(P, I)$ iff $I \leq \Phi_{[\neg G] \cdot I}(I)$ for all $I \in \mathbb{E}$, and pGCL program P.

Proof.

Left as an exercise. □

$P ::$ while $(c=0)$ $\{ x{+}{+}\ [p]\ c:=1 \}$

Claim $I = x + [c=0] \cdot \frac{p}{1-p}$ is an invariant of P wrt. $f = x$.

subinvariant + superinvariant

we prove that I is a superinvariant

$$\Phi_f(I) \leq I.$$

$$\Phi_f(I) = [c \neq 0] \cdot x + [c=0]\ wp\ (x{+}{+}\ [p]\ c:=1, I)$$

$$= [c \neq 0] \cdot x + [c=0] \left(p \cdot I(x:=x+1) + (1-p) \cdot I(c:=1) \right)$$

$$= \ldots$$

$$[c \neq 0](x) + [c=0] \left(\cancel{px} + p + \frac{p^2}{1-p} + (x) - \cancel{px} \right)$$

$$= x + [c=0] \left(\underbrace{\frac{p(1-p)+p^2}{1-p}}_{\frac{p}{1-p}} \right)$$

$$\Rightarrow \quad x + [c=0] \frac{p}{1-p} \quad \text{is a superinvariant}$$

As in the above derivation we only have equalities

I is a wp-subinvariant too.

Induction for upper bounds on wp

Induction on complete lattices (Park's lemma)

Let $(D, \sqsubseteq)$ be a complete lattice and $\Phi : D \to D$ continuous. Then:

$$\forall d \in D. \quad \Phi(d) \sqsubseteq d \quad \text{implies} \quad \text{lfp } \Phi \sqsubseteq d.$$

Applying this induction principle yields upper bounds on weakest pre-expectations.

Upper bounds on weakest pre-expectations

For while$(G)\{P\}$ and $f, I \in \mathbb{E}$ we have:

$$\underbrace{\Phi_f(I) \leq I}_{\text{wp-superinvariant}} \quad \text{implies} \quad wp(\text{while}(G)\{P\}, f) \leq I$$

Every wp-superinvariant of a loop for f
is an upper bound to the wp of the loop (and f).

Example

$P = c:=1;\ \text{while}(c)\ \{\ c:=0\ [\tfrac{1}{2}]\ x{+}{+}\ \}$

$f = x$

$I = x + [c=1]$ is a wp-superinvariant of P w.r.t. f.

$$\Phi_x(I) = \Phi(x + [c=1])$$

$$= [c\neq 1]\cdot x + [c=1]\ \tfrac{1}{2}\left(x + [0=1] + x{+}1{+} [c=1]\right)$$

$$= x + [c=1](0+1+1)\cdot\tfrac{1}{2} = I$$

Theorem $\quad wp(\text{while}\ldots, x) \leq x + [c=1] \qquad (*)$

For P: $\quad wp\ [c:=1;\ \text{while}\ldots, x) = \ldots \leq x + [1=1] = x+1$

$wp(c:=1,\ wp(\text{while}\ldots, x))$

$\leq x + [c=1]$ by monotonicity of wp

Co-induction for lower bounds on wlp

Co-induction on complete lattices (Park's lemma)

Let $(D, \sqsubseteq)$ be a complete lattice and $\Phi : D \to D$ continuous. Then:

$$\forall d \in D. \quad d \sqsubseteq \Phi(d) \quad \text{implies} \quad d \sqsubseteq \text{gfp } \Phi.$$

Applying this induction principle yields lower bounds on weakest liberal pre-expectations.

Lower bounds on weakest liberal pre-expectations

For while$(G)\{P\}$ and $f, I \in \mathbb{E}_{\leq 1}$ we have:

$$\underbrace{I \leq \Psi_f(I)}_{\text{wlp-subinvariant}} \quad \text{implies} \quad I \leq wlp(\text{while}(G)\{P\}, f)$$

Every wlp-subinvariant of a loop for f
is a lower bound to the wlp of the loop (and f).

Example P: $\{ c:=1 \; ; \; \text{while}(c) \; \{ \; \text{skip} \; [\tfrac{1}{2}] \; c:=0 \} \; \text{div} \; [\tfrac{1}{2}] \; x{+}{+} \; ; \}$

$f = [x \text{ even}]$

claim: $I = [c \neq 1] \cdot [x \text{ even}] + [c=1] \cdot \left(\frac{2}{3} + \frac{4\,[x \text{ odd}]}{15} + \frac{[x \text{ even}]}{15} \right)$

is a wlp-subinvariant of P

need: $\Phi(I) = I$

Theorem: $I \leq wp(\text{while} \ldots, x)$

Hence $wlp(P, x) = wlp(c:=1, \underbrace{wlp(\text{while} \ldots, x)}_{I \leq \ldots})$

$$\geq \frac{2}{3} + \frac{4\,[x \text{ odd}]}{15} + \frac{[x \text{ even}]}{15}$$

Verification of loops (1)

The following procedure can be followed for induction and co-induction:

1. Find an appropriate loop invariant I
2. Push I through the characteristic function of the loop once, i.e., compute $\Phi(I)$
3. Check whether this took us down or up in the partial order $\leq$:
   - 3.1 $\Phi(I) \leq I$, for induction, or
   - 3.2 $I \leq \Phi(I)$, for co-induction.

The key difficulty is to find an appropriate invariant I. This is undecidable.

Remarks about unsound versions of (co-)induction

Let $(D, \sqsubseteq)$ be a complete lattice and $\Phi : D \to D$ continuous. The following statements are not valid:

$\Phi(d) \sqsubseteq d \implies \text{lfp}\ \Phi \sqsubseteq d$

$\forall d \in D.\quad d \sqsubseteq \Phi(d) \quad \text{implies} \quad d \sqsubseteq \text{lfp}\ \Phi.$

lower bounds of wp

and

$\forall d \in D.\quad \Phi(d) \sqsubseteq d \quad \text{implies} \quad \text{gfp}\ \Phi \sqsubseteq d.$

upper bounds of wlp

$d \sqsubseteq \Phi(d) \implies d \sqsubseteq \text{gfp}\ \Phi$

Remarks about unsound versions of (co-)induction

Let $(D, \sqsubseteq)$ be a complete lattice and $\Phi : D \to D$ continuous. The following statements are not valid:

$$\forall d \in D. \quad d \sqsubseteq \Phi(d) \quad \text{implies} \quad d \sqsubseteq \text{lfp } \Phi.$$

and

$$\forall d \in D. \quad \Phi(d) \sqsubseteq d \quad \text{implies} \quad \text{gfp } \Phi \sqsubseteq d.$$

As a consequence, co-induction for lower bounds on wp of loops and induction on upper bounds on wlp of loops is unsound.

Remarks about unsound versions of (co-)induction

Let $(D, \sqsubseteq)$ be a complete lattice and $\Phi : D \to D$ continuous. The following statements are not valid:

$$\forall d \in D. \quad d \sqsubseteq \Phi(d) \quad \text{implies} \quad d \sqsubseteq \text{lfp } \Phi.$$

and

$$\forall d \in D. \quad \Phi(d) \sqsubseteq d \quad \text{implies} \quad \text{gfp } \Phi \sqsubseteq d.$$

As a consequence, co-induction for lower bounds on wp of loops and induction on upper bounds on wlp of loops is unsound.

Counterexample

On the black board.

Counter example

while $(c=1)$ { $(c:=0 \ [\tfrac{1}{2}] \ x{+}{+})$; $k{+}{+}$ }

$f = x$

$$\Phi(X) = [c \neq 1] \cdot x + [c=1] \cdot \left(\tfrac{1}{2} \times [k:=k+1, c:=0] + \tfrac{1}{2} \times [x:=x+1, k:=k+1] \right)$$

Claim: for every $a > 0$

$$I_a = x + [c=1](2^{k+a} + 1)$$

is a fixed point of Φ. (Please verify)

Now: $a < b$ then $I_a < I_b$

Thus if e.g. $I_b \leq \Phi_f(I_b)$, we cannot conclude that $I_b \leq \mathrm{lfp}\ \Phi$ since I_a is also a fixed point

Overview

ω-invariants

ω-invariants

Let $n \in \mathbb{N}$, $f \in \mathbb{E}$ and Φ_f be the wp-characteristic function of the loop while$(G)\{P\}$.

$I_0 \leq I_1 \leq I_2 \leq I_3 \ldots$

The monotonically increasing[1] sequence $(I)_{n\in\mathbb{N}}$ is a wp-ω-subinvariant of the loop w.r.t. f iff

$$I_0 \leq \Phi_f(\mathbf{0}) \quad \text{and} \quad I_{n+1} \leq \Phi_f(I_n) \quad \text{for all } n.$$

In a similar way, wlp-ω-superinvariants are defined, where $f, I_n \in \mathbb{E}_{\leq 1}$, $(I)_{n\in\mathbb{N}}$ is monotonically decreasing, and Φ_f is replaced by Ψ_f.

$I_0 \geq I_1 \geq I_2 \ldots$

$\Psi_f(1) \leq I_0$ and $\Psi_f(I_n) \leq I_{n+1}$

[1] But not necessarily strictly increasing.

Bounds on loops using ω-invariants

$I_0 \leq \Phi_f(0)$ $\quad I_{n+1} \leq \Phi_f(I_n)$

Theorem

1. Let $(I_n)_{n\in\mathbb{N}}$ be a wp-ω-subinvariant of $\text{while}(G)\{P\}$ w.r.t. $f \in \mathbb{E}$. Then:

$$\sup_{n\in\mathbb{N}} I_n \;\leq\; wp(\text{while}(G)\{P\}, f)\ .$$

2. Let $(I_n)_{n\in\mathbb{N}}$ be a wlp-ω-superinvariant of $\text{while}(G)\{P\}$ w.r.t. f ($f \in \mathbb{E}_{\leq 1}$)! Then:

$$wlp(\text{while}(G)\{P\}, f) \;\leq\; \inf_{n\in\mathbb{N}} I_n\ .$$

Proof.

□

Claim: $\boxed{\forall n \in \mathbb{N}.\quad I_n \leq \underline{\Phi}_f^{n+1}(\underline{0})}$

for ω-cpo subinvariant $(I_n)_{n\in\mathbb{N}}$.

$$\sup_{n\in\mathbb{N}} I_n \leq \sup_{n\in\mathbb{N}} \underline{\Phi}_f^{n+1}(\underline{0})$$

$$= \sup_{n\in\mathbb{N}} \underline{\Phi}_f^{n}(\underline{0})$$

$$= \mathrm{lfp}\ \underline{\Phi}_f$$

$$= [\![\text{while } \ldots \text{ do } f]\!] \qquad \boxtimes$$

Verification of loops (2)

The following procedure can be followed using ω-sub-/superinvariants:

1. Find an appropriate ω-invariant, i.e., a sequence $(I_n)_{n\in\mathbb{N}}$
2. Check that $(I_n)_{n\in\mathbb{N}}$ is indeed an ω-invariant:
   - 2.1 Push I_n through the characteristic function
   - 2.2 Check whether this took us above I_{n+1} (for wp) or below I_{n+1} (for wlp) in the partial order $\leq$
3. Find the supremum (for wp) or the infimum (for wlp) of $(I)_{n\in\mathbb{N}}$

In addition to finding appropriate invariants I_n, we have to reason about the limits of the ω-invariants. This is undecidable too.

Overview

Total correctness proof rules for loops [McIver & Morgan 2005]

Total correctness proof rules for loops

Total correctness proof rules for loops

Let $f \in \mathbb{E}$ with $f \leq \mathbf{k}$, for some $k \in \mathbb{N}$, and let $J \in \mathbb{E}$ be $\mathbf{k}$-bounded too such that $I \in \mathbb{E}$ defined by: $I = [\neg G] \cdot f + [G] \cdot J$.

$\forall s \in \mathcal{S} .\ f(s) \leq k$

$J(s) \leq k$

Total correctness proof rules for loops

Total correctness proof rules for loops

Let $f \in \mathbb{E}$ with $f \leq \mathbf{k}$, for some $k \in \mathbb{N}$, and let $J \in \mathbb{E}$ be $\mathbf{k}$-bounded too such that $I \in \mathbb{E}$ defined by: $I = [\neg G] \cdot f + [G] \cdot J$. Then it holds:

1. If $I = [F]$ for some predicate $F \in \mathbb{P}$, then:

$$T \cdot I \leq wp(\text{while}(G)\{P\}, f)$$

where $T = wp(\text{while}(G)\{P\}, \mathbf{1})$ is the loop's termination probability.

2. If $[F] \leq T$ for some predicate $F \in \mathbb{P}$, then:

$$[F] \cdot I \leq wp(\text{while}(G)\{P\}, f)$$

3. If $\varepsilon \cdot I \leq T$ for some $\varepsilon > 0$, then:

$$I \leq wp(\text{while}(G)\{P\}, f).$$

Almost-surely terminating loops

Pr { termination of while().P.} = 1.

Proof rules for a.s.-terminating loops

Let while$(G)\{P\}$ be almost-surely terminating, i.e., $wp(\text{while}(G)\{P\}, \mathbf{1}) = 1$, and let $I \in \mathbb{E}_{\leq 1}$. Then:

for all inputs

1. $$I \leq \Phi_f(I) \quad \text{implies} \quad I \leq wp(\text{while}(G)\{P\}, f)\ .$$

2. $$\Psi_f(I) \leq I \quad \text{implies} \quad wlp(\text{while}(G)\{P\}, f) \leq I\ .$$

while (c) { c := 0 [1/2] x++ }

$I = x + [c=1] \frac{P}{1-p}$

Bound refinement

Bound refinement

Let loop while$(G)\{P\}$, $f, I \in \mathbb{E}$. Then:

1. If $\Phi_f(I) \leq I$:

$$wp(\text{while}(G)\{P\}, f) \leq I \quad \text{implies} \quad wp(\text{while}(G)\{P\}, f) \leq \Phi_f(I) \,.$$

let $wp(\text{while}\ \_\ \_\ f) \subseteq I$.

iff $\text{lfp}\ \Phi \subseteq I$

$\Leftarrow \Phi(\text{lfp}\ \Phi) \subseteq \Phi(I)$

$\Leftarrow \underbrace{\text{lfp}\ \Phi} \subseteq \Phi(I)$

$\equiv wp(\text{while}\ \ldots) \subseteq \Phi(I)$

Bound refinement

Bound refinement

Let loop while$(G)\{P\}$, $f, I \in \mathbb{E}$. Then:

1. If $\Phi_f(I) \leq I$:

$$wp(\text{while}(G)\{P\}, f) \leq I \quad \text{implies} \quad wp(\text{while}(G)\{P\}, f) \leq \Phi_f(I) \ .$$

2. If $I \leq \Phi_f(I)$:

$$I \leq wp(\text{while}(G)\{P\}, f) \quad \text{implies} \quad \Phi_f(I) \leq wp(\text{while}(G)\{P\}, f) \ .$$

If $\Phi_f(I) \neq I$, we thus obtain tighter upper and lower bounds, respectively.

The sequence $\Phi(I), \Phi^2(I), \Phi^3(I), \ldots$ converges.
For upper bounds to the greatest fixed point that is below (or equal to) I.
(This fixed point itself is an upper bound too.)

This is the Tarski-Kantorovich fixpoint principle.

Overview

Invariant synthesis for linear programs

- right-hand sides of assignments are linear expressions

$$x := x + y + 20z$$

~~$x := x^2 + y^3$~~

- guards must be linear too

Invariant synthesis for linear programs

1. Speculate that a loop invariant can be expressed as linear expression:

template $\underbrace{[\alpha_1 \cdot x_1 + \ldots + \alpha_n \cdot x_n + \alpha_{n+1} \ll 0]}_{\text{qualitative invariant}} \cdot \underbrace{(\beta_1 \cdot x_1 + \ldots + \beta_n \cdot x_n + \beta_{n+1})}_{\text{quantitative invariant}}$

- where n is at most the number of program variables in the program
- x_i are program variables
- α_i, β_i are real-valued invariant-template variables
- $\ll \in \{<, \leq\}$ is a binary comparison operator

Invariant synthesis for linear programs

$\alpha_1 \leq \alpha_2 + \alpha_3$

1. Speculate that a loop invariant can be expressed as linear expression:

$$[\alpha_1 \cdot x_1 + \ldots + \alpha_n \cdot x_n + \alpha_{n+1} \ll 0] \cdot (\beta_1 \cdot x_1 + \ldots + \beta_n \cdot x_n + \beta_{n+1})$$

   - where n is at most the number of program variables in the program
   - x_i are program variables
   - α_i, β_i are real-valued invariant-template variables
   - $\ll \in \{<, \leq\}$ is a binary comparison operator

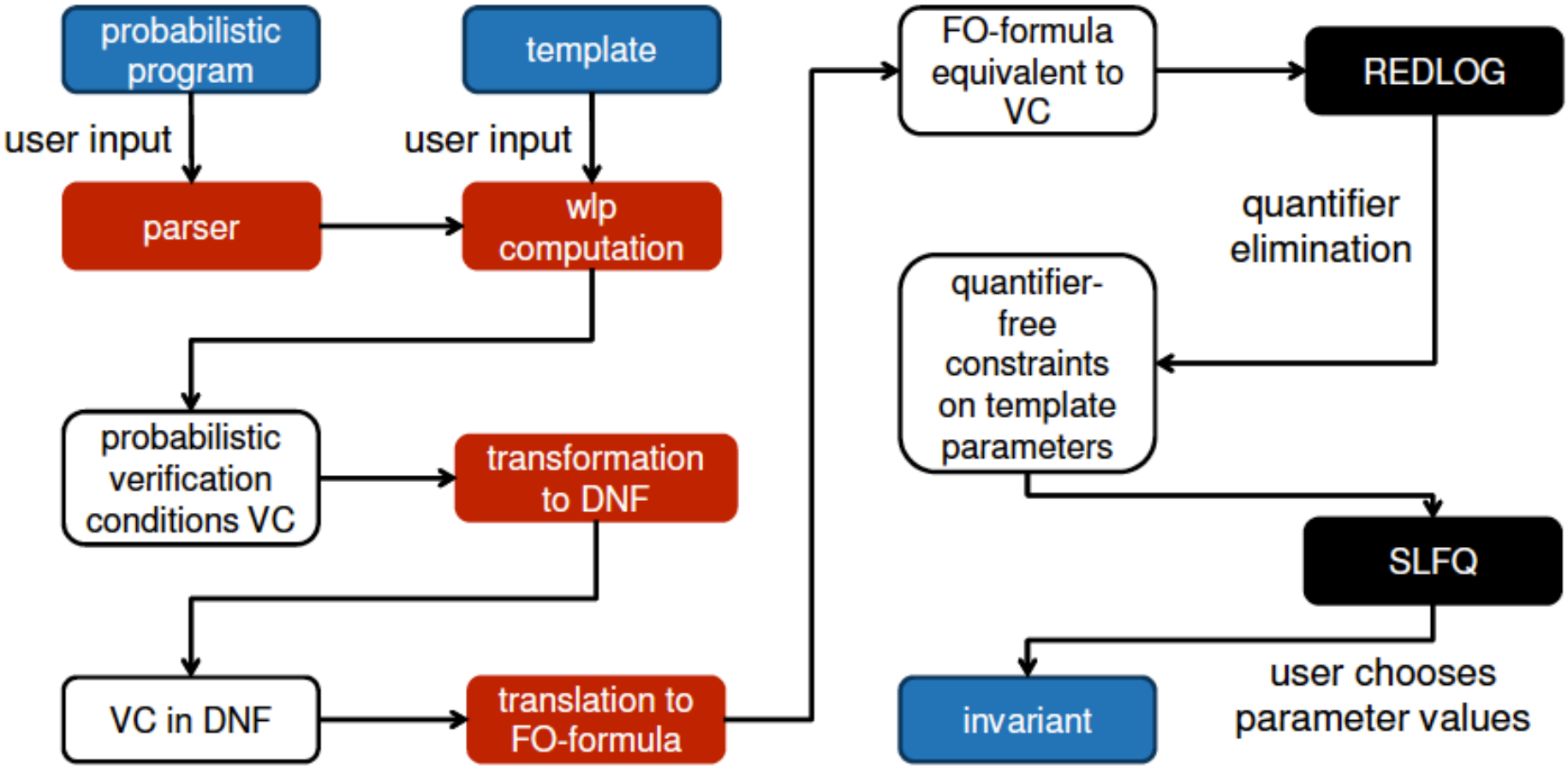
2. Transform these numerical constraints into non-linear FO formulas.
3. Use constraint solving techniques to obtain constraints on α_i, β_i.

Soundness and completeness

For any linear probabilistic program and linear loop invariant
this method will find
all parameter solutions that make the template valid, and no others.

α, β

PRINSYS tool: Probabilistic invariant synthesis



moves.rwth-aachen.de/prinsys

Duelling cowboys: when does A win?

```
int cbDuel(float a, b) {
  int t := A;
  int c := 1;
  while (c = 1) {
    if (t = A) {
      (c := 0 [a] t := B);
    } else {
      (c := 0 [b] t := A);
    }
  }
  return t;
}
```

Duelling cowboys: when does A win?

```
int cbDuel(float a, b) {
  int t := A;
  int c := 1;
  while (c = 1) {
    if (t = A) {
      (c := 0 [a] t := B);
    } else {
      (c := 0 [b] t := A);
    }
  }
  return t;
}
```

Aim: find expectation I

Satisfying $I \leq [t = A]$ upon termination.

survivor = f

Duelling cowboys: when does A win?

```
int cbDuel(float a, b) {
  int t := A;
  int c := 1;
  while (c = 1) {
    if (t = A) {
      (c := 0 [a] t := B);
    } else {
      (c := 0 [b] t := A);
    }
  }
  return t;
}
```

Aim: find expectation I

Satisfying $I \leq [t = A]$ upon termination.

Observation

On entering the loop, c = 1 and either t = A or t = B.

Template suggestion

$$I = \underbrace{[t = A \wedge c = 0]}_{A \text{ wins duel}} \cdot 1 + \underbrace{[t = A \wedge c = 1]}_{A\text{'s turn}} \cdot \alpha + \underbrace{[t = B \wedge c = 1]}_{B\text{'s turn}} \cdot \beta$$

$= 0$ $\quad + [t = B \wedge c = 0] \cdot \gamma$

Duelling cowboys: when does A win?

Invariant template: $I = [\mathsf{t} = A \wedge \mathsf{c} = 0] \cdot 1 + [\mathsf{t} = A \wedge \mathsf{c} = 1] \cdot \alpha + [\mathsf{t} = B \wedge \mathsf{c} = 1] \cdot \beta$

Initially, $\mathsf{t} = A \wedge \mathsf{c} = 1$ and thus $\alpha = Pr\{A \text{ wins duel}\}$.

Maximisation: $\beta = (1 - b) \cdot \alpha$ and $\alpha = \dfrac{a}{a + b - a \cdot b}$

Probabilistic loop invariant

$$I = [\mathsf{t} = A \wedge \mathsf{c} = 0] \cdot 1 + [\mathsf{t} = A \wedge \mathsf{c} = 1] \cdot \frac{a}{a + b - ab} + [\mathsf{t} = B \wedge \mathsf{c} = 1] \cdot \frac{(1 - b)a}{a + b - a \cdot b}$$

Annotated program for post-expectation $[t = A]$

```
1  int cowboyDuel(a, b) {
2    ⟨[illegible]⟩
3    ⟨[illegible] a/(a+b−ab), [illegible]⟩
4    (t := A [illegible];
5    ⟨[t = A] · a/(a+b−ab) + [t = B] · (1−b)a/(a+b−ab)⟩
6    c := 1;
7    ⟨[t = A ∧ c = 0] · 1 + [t = A ∧ c = 1] · a/(a+b−ab) + [t = B ∧ c = 1] · (1−b)a/(a+b−ab)⟩
8    while (c = 1) {
9      ⟨[t = A ∧ c = 1] · a/(a+b−ab) + [t = B ∧ c = 1] · (1−b)a/(a+b−ab)⟩
10     ⟨[t = A ∧ c ≠ 1] · a + [t = A ∧ c = 1] · a/(a+b−ab)
           + [t = B ∧ c = 0] · (1 − b) + [t = B ∧ c = 1] · (1−b)a/(a+b−ab)⟩
11     if (t = A) {
12       (c := 0 [a] t := B);
13     } else {
14       (c := 0 [b] t := A);
15     }
16     ⟨[t = A ∧ c = 0] · 1 + [t = A ∧ c = 1] · a/(a+b−ab) + [t = B ∧ c = 1] · (1−b)a/(a+b−ab)⟩
17   }
18   ⟨[c ≠ 1] · ([t = A ∧ c = 0] · 1 + [t = A ∧ c = 1] · a/(a+b−ab)
         + [t = B ∧ c = 1] · (1−b)a/(a+b−ab))⟩
19   ⟨[t = A]⟩
20   return t; // the survivor
21 }
```

Playing with geometric distributions

- X is a random variable, geometrically distributed with parameter p
- Y is a random variable, geometrically distributed with parameter q

Q: generate a sample x, say, according to the random variable $X - Y$

```
int XminY1(float p, q){ // 0 <= p, q <= 1
  int x := 0;
  bool flip := false;
  while (not flip) { // take a sample of X to increase x
    (x +:= 1 [p] flip := true);
  }
  flip := false;
  while (not flip) { // take a sample of Y to decrease x
    (x -:= 1 [q] flip := true);
  }
  return x; // a sample of X-Y
}
```

An alternative program

$f = [x=22]$

$f = x$

```
int XminY2(float p, q){
  int x := 0;
  bool flip := false;
  (flip := false [0.5] flip := true); // flip a fair coin
  if (not flip) {
    while (not flip) { // sample X to increase x
      (x +:= 1 [p] flip := true);
    }
  } else {
    flip := false; // reset flip
    while (not flip) { // sample Y to decrease x
      x -:= 1;
      (skip [q] flip := true);
    }
  }
return x; // a sample of X-Y
}
```

Geom (p)

Geom (q)

Program equivalence: $X - Y$

```
int XminY1(float p, q){
  int x, c := 0, 1;
  while (c) {
    (x +:= 1 [p] c := 0);
  }
  c := 1;
  while (c) {
    (x -:= 1 [q] c := 0);
  }
  return x;
}
```

$p = \frac{1}{2} \quad q = \frac{2}{3}$

```
int XminY2(float p, q){
  int x := 0;
  (c := 0 [0.5] c := 1);
  if (c) {
    while (c) {
      (x +:= 1 [p] c := 0);
    }
  } else {
    c := 1;
    while (c) {
      x -:= 1;
      (skip [q] c := 0);
    }
  }
return x;
}
```

Using wp, one can prove that the expectations of $f = x$ coincide if and only if $q = \frac{1}{2-p}$.

Program equivalence: $X - Y$

```
int XminY1(float p, q){
  int x, f := 0, 0;
  while (f = 0) {
    (x++ [p] f := 1);
  }
  f := 0;
  while (f = 0) {
    (x-- [q] f := 1);
  }
  return x;
}
```

11

12

```
int XminY2(float p, q){
  int x, f := 0, 0;
  (f := 0 [0.5] f := 1);
  if (f = 0) {
    while (f = 0) {
      (x++ [p] f := 1);
    }
  } else {
    f := 0;
    while (f = 0) {
      x--;
      (skip [q] f := 1);
    }
  }
  return x;
}
```

21

22

Using template $I = x + [f = 0] \cdot \alpha$ we find:

$\alpha_{11} = \frac{p}{1-p}$, $\alpha_{12} = -\frac{q}{1-q}$, $\alpha_{21} = \alpha_{11}$ and $\alpha_{22} = -\frac{1}{1-q}$.

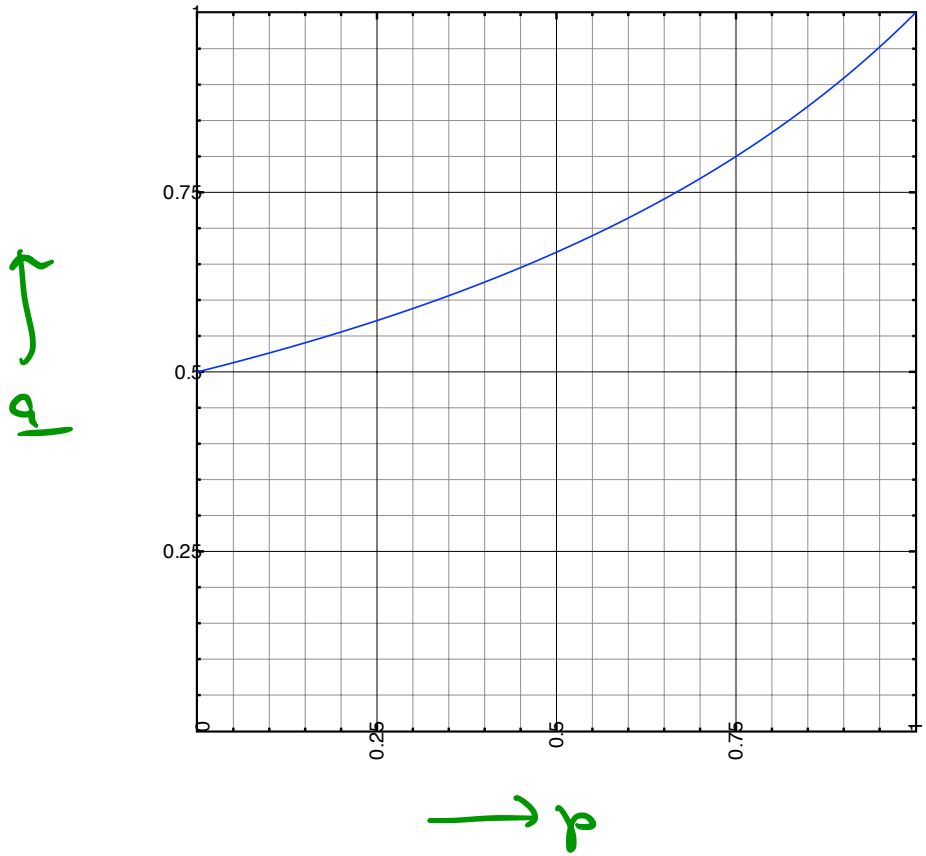
Program equivalence: $X - Y$

```
int XminY1(float p, q){
  int x, f := 0, 0;
  while (f = 0) {
    (x++ [p] f := 1);
  }
  f := 0;
  while (f = 0) {
    (x-- [q] f := 1);
  }
  return x;
}
```

```
int XminY2(float p, q){
  int x, f := 0, 0;
  (f := 0 [0.5] f := 1);
  if (f = 0) {
    while (f = 0) {
      (x++ [p] f := 1);
    }
  } else {
    f := 0;
    while (f = 0) {
      x--;
      (skip [q] f := 1);
    }
  }
  return x;
}
```

Expected value of x is $\frac{p}{1-p} - \frac{q}{1-q}$ and $\frac{p}{2(1-p)} - \frac{1}{2(1-q)}$.

Graphically this means ...



Both programs yield the same expected outcome for x all points on the curve $q = \frac{1}{2-p}$.

(and on no other points)