Linear-invariant generation for probabilistic programs

JP Katoen[a] AK McIver[b] LA Meinicke[b] CC Morgan[c]

[a] RWTH Aachen University, Germany

[b] Macquarie University, Australia

[c] University of New South Wales, Australia

Overview: Invariant generation

Inductive invariants may be used to verify iterative programs
(Floyd, 1967; Hoare 1969; Dijkstra, 1971).

Automatically generating these invariants is possible for invariants of restricted forms for restricted types of programs.

Methods include

- iterative fixed-point methods like abstract interpretation (Cousot and Cousot, 1977), and
- constraint-based approaches (e.g., Colón et al., 2003; Podelski and Rybalchenko, 2004; Cousot, 2005; Monniaux, 2000; Gulwani et al., 2008).

Overview: Constraint-based approaches

Invariants are found by

- constructing verification conditions that are sufficient to show that
- predicates of a given parameterised formula (containing only first-order unknowns) are invariant, and then
- solving them for all possible parameters of the formula using off-the-shelf constraint solvers.

Overview: Constraint-based approaches

In Colón et al. (2003) parameterised formulas are

- linear constraints on program variables,

and the programs themselves must be

- linear
- with real-valued variables.

Methods that require weaker restrictions on the form of the program and invariant have been investigated.
(E.g., Sankaranarayanan et al., 2004; Cousot, 2005; Kapur, 2005.)

Overview: Probabilistic Programs

Choices may be made both qualitatively ($S \sqcap T$) and quantitatively ($S\ {}_p{\oplus}\ T$).

Inductive quantitative invariants can be used in probabilistic program verification (Morgan, 1996; McIver and Morgan, 2005).

No methods exist for automatically generating quantitative invariants ...

Overview: Goals

The development of an automated assistant for quantitative invariant discovery to augment interactive proofs.

Overview: Goals

So far we have defined a constraint-based method for automatically generating quantitative invariants of

- linear probabilistic programs with
- real-valued variables, in which
- the parameterised invariants are structures built from linear terms.

Outline

- Qualitative and quantitative invariants.
- Constraint-solving for qualitative and qualitative invariants.
- An example.
- Comparison to other automated approaches.

Qualitative invariants: notation and semantics

Programs are interpreted using a weakest-liberal-precondition semantics:

- *wlp*.S.Q denotes the largest set of states from which S is guaranteed to either not terminate or terminate in a state satisfying Q.

- S satisfies specification $[P, Q]$ when $P \Rightarrow$ *wlp*.S.Q.

- $S \sqsubseteq T \triangleq (\forall Q \cdot \textit{wlp}.S.Q \Rightarrow \textit{wlp}.T.Q)$.

- Specifications are treated as programs.

Qualitative invariants: inductive invariants

An inductive invariant I of

$$loop \triangleq \mathsf{while}\ G\ \mathsf{do}\ S\ \mathsf{od}\ ,$$

is a predicate on the state space of the program that is preserved by iterations of the loop.

I.e., $G \wedge I \Rightarrow \mathit{wlp}.S.I$.

If I is an inductive invariant of $loop$, we have that $loop$ satisfies the specification $[I, \neg G \wedge I]$.

Qualitative invariants: inductive invariant maps

A valid inductive invariant map of a program

$loop_1 \triangleq \text{while } G_1 \text{ do } S_1 \text{ od}$

containing J program loops

$loop_j \triangleq \text{while } G_j \text{ do } S_j \text{ od}$

is a set of predicates, $\{I_j \cdot j \in [1..J]\}$, such that

I_j is an invariant of $\text{while } G_j \text{ do } S_j' \text{ od}$

where S_j is the same as S_j except that each of its inner loops $loop_i$ (if any) has been replaced by $[I_i, \neg G_i \wedge I_i]$.

Qualitative invariants: inductive invariant maps

For example, $\{I_1, I_2\}$ is a valid inductive invariant map of

$$
\begin{array}{ll}
loop_1: & \textsf{while } G_1 \textsf{ do} \\
 & \quad S_1; \\
loop_2: & \quad \textsf{while } G_2 \textsf{ do } S_2 \textsf{ od} \\
 & \textsf{od}
\end{array}
$$

(in which programs S_1 and S_2 do not contain loops) if

$$I_2 \wedge G_2 \Rightarrow \textit{wlp}.S_2.I_2 \quad \text{and}$$
$$I_1 \wedge G_1 \Rightarrow \textit{wlp}.(S_1; [I_2, \neg G_2 \wedge I_2]).I_1 \ .$$

Qualitative invariants: inductive invariant maps

If $\{I_j \cdot j \in [1..J]\}$ is a valid inductive invariant map of a loop $loop_1$, then each I_j is an invariant of $loop_j$.

Generating valid inductive invariant maps for a given qualitative loop involves (by definition) finding solutions to a set of second-order constraints.

Quantitative invariants: notation and semantics

Probabilistic programs are given a meaning in terms of a weakest-liberal-precondition semantics (McIver and Morgan, 2005).

Predicates are generalised to non-negative, one-bounded, real-valued functions, referred to as expectations.

- *wlp*.S.*expt*.σ denotes the least expected value of *expt* that may be witnessed by executing S from initial state σ.
- S satisfies specification [$\mathit{expt}_1, \mathit{expt}_2$] when $\mathit{expt}_1 \leq \mathit{wlp}.S.\mathit{expt}_2$.
- $S \sqsubseteq T \triangleq (\forall \mathit{expt} \cdot \mathit{wlp}.S.\mathit{expt} \leq \mathit{wlp}.T.\mathit{expt})$.
- Specifications are treated as programs.

Quantitative invariants

A quantitative invariant I of

$$loop \triangleq \mathsf{while}\ G\ \mathsf{do}\ S\ \mathsf{od}\ ,$$

is an expectation whose expected value does not decrease after iteration of the body of the loop.

I.e., $[G]{\times}I \quad \leq \quad \mathit{wlp}.S.I$.

If I is a quantitative invariant of $loop$ then $loop$ satisfies the specification $[I, [\neg G] \times I]$.

Quantitative invariants: Binomial update example

We have that

$$I \triangleq [0 \leq x \leq n \leq N] \times (x/N - pn/N + p)$$

is an invariant of:

$$\begin{array}{ll} init: & x, n := 0, 0; \\ loop: & \text{while } n < N \text{ do} \\ body: & \quad (x := x + 1 \,_p\!\oplus \text{skip}); n := n + 1 \\ & \text{od} \end{array}$$

since $I \leq \mathit{wlp}.body.I$.

Quantitative invariants: Binomial update example

So we can conclude that

$$
\begin{array}{lll}
 & wlp.(init; loop).(x/N) & \\
= & wlp.init.(wlp.loop.(x/N)) & \{\text{definition of sequential composition}\} \\
\geq & wlp.init.I & \{I \text{ is an invariant of } loop\} \\
= & [0 \leq N] \times p & \{\text{calculate}\} \\
= & p \,. & \{\text{assuming that } N \text{ is positive}\}
\end{array}
$$

And so $pN \leq wlp.(init; loop).x$.

Quantitative invariants: inductive invariant maps

We define a valid quantitative inductive invariant map to be a valid qualitative inductive invariant map in which expectations take the place of predicates.

If $\{I_j \cdot j \in [1..J]\}$ is a valid quantitative inductive invariant map of a probabilistic loop $loop_1$, then each I_j is a quantitative invariant of $loop_j$.

Constraint-solving for qualitative invariants

An overview of how the constraint-solving method of Colón et al. (2003) may be applied to find valid inductive invariant maps for any "linear program"

$$loop_1 \triangleq \mathsf{while}\ G_1\ \mathsf{do}\ S_1\ \mathsf{od}$$

with real-valued program variables $x_1, \ldots, x_X$, containing J loops

$$loop_j \triangleq \mathsf{while}\ G_j\ \mathsf{do}\ S_j\ \mathsf{od}\ .$$

Constraint-solving for qualitative invariants: parameterisation

Each I_j is first parameterised using an (M, N)-linear predicate

$$\bigwedge_{m \in [1..M]} \Big(\bigvee_{n \in [1..N]} \alpha_{(j,mn,1)} x_1 + \ldots + \alpha_{(j,mn,X)} x_X + \beta_{(j,mn)} \approx 0 \Big) \ ,$$

with free real-valued variables $\alpha_{(j,mn,x)}$ and $\beta_{(j,mn)}$, where $\approx$ may be instantiated with either comparison operator $\leq$ or $<$.

Constraint-solving for qualitative invariants: parameterisation

Parameterising each I_j in the definition of an inductive invariant map reduces each proof obligation

$$G_j \wedge I_j \Rightarrow \textit{wlp}.S'_j.I_j$$

to a first-order constraint on the free real-valued variables in the parametric representations.

Constraint-solving for qualitative invariants: simplification and solving

Next we:

- Evaluate the weakest-liberal precondition expressions, representing each proof obligation

  $$G_j \wedge I_j \Rightarrow \textit{wlp}.S'_j.I_j$$

  as a finite Boolean expression on linear constraints.

- Translate these universally quantified Boolean expressions to existentially quantified polynomial constraints using Motzkin's Transposition Theorem (Motzkin, 1936).

- Solve the resulting constraints using off-the-shelf constraint solvers.

Constraint-solving for quantitative invariants: parameterisation

We parameterise each I_j with an (M,N)-linear expression

$$\sum_{m\in[1..M]}[\bigwedge_{n\in[1..N]} \alpha_{(j,mn,1)}x_1 + \ldots + \alpha_{(j,mn,X)}x_X + \beta_{(j,mn)} \approx 0] \\ \times (\gamma_{(j,m,1)}x_1 + \ldots + \gamma_{(j,m,X)}x_X + \delta_{(j,m)})$$

containing free real-valued variables $\alpha_{(j,mn,x)}$, $\beta_{(j,mn)}$, $\gamma_{(j,m,x)}$ and $\delta_{(j,m)}$.

We impose the additional constraint that, for each $j \in [1..J]$, I_j is bounded.

I.e., $0 \le I_j$ and $I_j \le 1$.

Constraint-solving for quantitative invariants: parameterisation

Parameterisation reduces the constraints on each inductive invariant I_j in the quantitative inductive invariant map to the following universally quantified constraints on first-order unknowns:

$$
\begin{aligned}
& 0 \leq I_j \ , \\
& I_j \leq 1 \text{ and} \\
& [G_j] \times I_j \leq \mathit{wlp}.S'_j.I_j \ .
\end{aligned}
$$

Constraint-solving for quantitative invariants: simplification and solving

Next we

- Evaluate the weakest-liberal precondition expressions, and translate each constraint

$$0 \leq I_j \ , \tag{1}$$
$$I_j \leq 1 \text{ and} \tag{2}$$
$$[G_j] \times I_j \leq \mathit{wlp}.S'_j.I_j \ . \tag{3}$$

  to a finite Boolean expression on linear constraints

- Apply Motzkin's Transposition theorem (as for qualitative case).

- Solve the resulting constraints (as for qualitative case).

Constraint-solving for quantitative invariants: simplification and solving

We have shown that the first of these steps is possible since:

(i) Conditions (1-3) may be written as inequalities between some (M, N)-linear and (K, L)-linear expressions.

(ii) Each inequality between a (M, N)-linear and (K, L)-linear expression may be represented as a finite Boolean expression over linear constraints.

Example: Binomial Update

Given that $I_1 \triangleq [0 \leq x \leq n \leq N]$ is invariant, we search for quantitative invariants

$$I \triangleq I_1 \times (\alpha x + \beta n + \gamma) \ .$$

Example: Binomial Update

The constraints that any such $I \triangleq I_1 \times (\alpha x + \beta n + \gamma)$ must satisfy are:

$$
\begin{aligned}
0 &\leq I_1 \times (\alpha x + \beta n + \gamma) && (4) \\
I_1 \times (\alpha x + \beta n + \gamma) &\leq 1 && (5) \\
[n < N] \times I_1 \times (\alpha x + \beta n + \gamma) &\leq \textit{wlp}.body.(I_1 \times (\alpha x + \beta n + \gamma)) && (6)
\end{aligned}
$$

Example: Binomial Update

Using the fact that I_1 is invariant, these are equivalent to the following Boolean constraints

$$I_1 \Rightarrow (0 \le \alpha x + \beta n + \gamma) \quad (7)$$

$$I_1 \Rightarrow (\alpha x + \beta n + \gamma \le 1) \quad (8)$$

$$n < N \wedge I_1 \Rightarrow (\alpha x + \beta n + \gamma) \le \textit{wlp}.body.(\alpha x + \beta n + \gamma) \quad (9)$$

Example: Binomial Update

Evaluating *wlp* expression:

$$
\begin{aligned}
I_1 &\Rightarrow (0 \le \alpha x + \beta n + \gamma) && (10)\\
I_1 &\Rightarrow (\alpha x + \beta n + \gamma \le 1) && (11)\\
n < N \wedge I_1 &\Rightarrow (\alpha x + \beta n + \gamma) \le \alpha x + \beta n + p\alpha + \beta + \gamma && (12)
\end{aligned}
$$

Translating (12) to a set of existentially quantified constraints and solving reveals that parameters α, β and γ must satisfy

$$p\alpha + \beta \ge 0 \ ,$$

i.e., variable x grows at most p times the rate of n.

(The other constraints may be similarly solved.)

Alternative automated methods: comparison

Using proof-based methods in conjunction with automatic quantitative invariant generation methods we can verify programs with parameters.

(Such as the Binomial program with parameters p and N.)

Alternative automated methods: comparison

Probabilistic model checkers for MDP's, e.g.,

- PRISM:
  PCTL model checking allowing calculation of, e.g., maximal reachability probabilities.

- LiQuor:
  Maximal probability that MDP M satisfies LTL formula φ.

Can be applied to instances of probabilistic programs.

E.g., for the Binomial distribution program it can be checked:

" whether or not the probability that $x = 0$ is $(1-p)^N$, for a fixed value of p and N".

Alternative automated methods: comparison

Automated abstraction-refinement, e.g.,

- PASS:
  A SAT-based extension of PRISM. Can be used to check maximal reachability properities.

- SAT-based PRISM:
  lower and upper bounds of maximal reachability probabilities or of expected values.

can be used to verify properties of programs with unbounded unknowns.

Alternative automated methods: comparison

Other tools include

- APEX:
  checks language equivalence between probabilistic programs over finite integer datatypes but in addition allows open programs, i.e., programs in which the value of certain variables is not fixed.

- Abstract interpretation methods for probabilistic programs:
  As for non-probabilistic abstract interpretation methods, these might only produce "approximate answers".

None of these other methods produce quantitative invariants for probabilistic programs.

Conclusion: what we've done

We have defined a sound constraint-based method for generating linear quantitative invariants for linear probabilistic programs with real-valued variables.

Conclusion: what we'd like to do

We would like to

- build tool support for this approach

- extend our approach to generate polynomial forms of quantitative invariants (as in Sankaranarayanan et al. (2004), Cousot (2005) and Kapur (2005)).