Towards efficient automated analysis of probabilistic programs

MOVES Workshop

Marcin Szymczak

July 9, 2020

Abandon all hope ye who enter here...

I have **absolutely no concrete results** to present.
I will present my previous failed attempts and outline new ideas.

Motivation

We want to **automatically** and **efficiently** check properties of probabilistic programs. We focus in particular on termination complexity.

- Existing incomplete techniques (Kaminski et al.[1]) apply to general programs, but require custom invariants for while-loops, which are difficult to find.
- Automatic invariant generation (Katoen et al.[2]) works only in restricted settings.
- Automatic, efficiently decidable analysis methods only apply to a very restricted class of programs (Giesl et al.[3])

[1] Benjamin Lucien Kaminski et al. "Weakest Precondition Reasoning for Expected Runtimes of Randomized Algorithms". In: *J. ACM* 65.5 (2018), 30:1–30:68.

[2] Joost-Pieter Katoen et al. "Linear-Invariant Generation for Probabilistic Programs:" in: *Static Analysis*. Ed. by Radhia Cousot and Matthieu Martel. Springer Berlin Heidelberg, 2010.

[3] Jürgen Giesl, Peter Giesl, and Marcel Hark. "Computing Expected Runtimes for Constant Probability Programs". In: *Automated Deduction - CADE 27 - 27th International Conference on Automated Deduction, Natal, Brazil, August 27-30, 2019, Proceedings*. Vol. 11716. 2019.

Termination of probabilistic programs via pVASS

Original idea (Presented in Kleinwalsertal this year):

- Find a **complete** and **fully automated** way of analysing termination complexity of some restricted (but as broad as possible) class of programs.
- Use recent results on deciding linear termination of **probabilistic vector addition systems (pVASS)**.
- Translate probabilistic programs to pVASS and apply existing results

This was **more difficult than expected**. I will now explain why.

What are VASS?

A vector addition system with states (VASS) is a transition system consisting of:

- A finite set Q of *control states*
- n integer-valued *counters* $\mathbf{v} = (v_1, \ldots, v_n)$
  - *configuration* $p\mathbf{v}$ = control state p + counter values $\mathbf{v}$
- A finite set of *transitions* $(q, \mathbf{u}, p)$ which update the configuration $q\mathbf{v}$ to $p(\mathbf{v} + \mathbf{u})$

We assume that a VASS *terminates* when at least one counter becomes negative.

VASS $\iff$ Petri nets

VASS Fast Termination

Bràzdil et al.[4] showed the following result:

Theorem

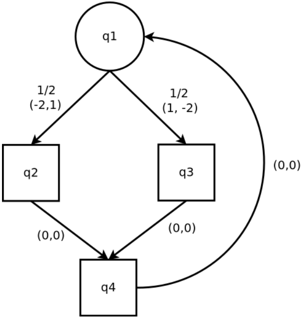
The problem of whether a strongly connected VASS terminates (under demonic nondeterminism) in linear time can be reduced to a linear programming problem, solvable in polynomial time.

[4] Tomás Brázdil et al. "Efficient Algorithms for Asymptotic Bounds on Termination Time in VASS". In: *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018*. 2018.

Probabilistic VASS

Probabilistic VASS (pVASS) include probabilistic as well as nondeterministic transitions. Can model probabilistic programs.

```
while((k>0)&&(l>0)) {
  u = flip();
  if (u) {
    k-=2;
    l++;
  }
  else {
    l-=2;
    k++;
  }
}
```



Termination in Probabilistic VASS[5]

Decidability of linear termination in *strongly connected* VASS extends to pVASS

Terminology:

- A strategy is **Markov Deterministic (MD)** if each nondeterministic state has a fixed successor
- A **strongly connected component (SCC)** $\mathcal{S}$ is a set of states s.t. for all $s, s' \in \mathcal{S}$ there is a path from s to s' which does not leave $\mathcal{S}$.
- A **Bottom Strongly Connected Components (BSCC)** $\mathcal{B}$ (for a given strategy) is a SCC s.t. there is *no* path leaving $\mathcal{B}$ in the resulting Markov chain.

Observations:

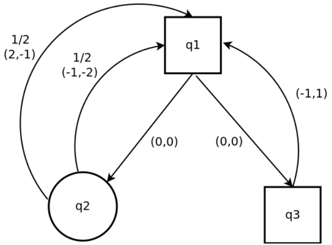
- For each MD strategy (in a strongly-connected pVASS), execution finally reaches a BSCC (potentially multiple BSCCs reachable)
- In every such BSCC we can compute the *average counter change per transition*

[5] Tomás Brázdil et al. "Deciding Fast Termination for Probabilistic VASS with Nondeterminism". In: *Automated Technology for Verification and Analysis - 17th International Symposium, ATVA 2019, Taipei, Taiwan, October 28-31, 2019, Proceedings*. 2019.

BSCCs and average changes: Example

Example:

- Strategy 1: $q1$ always goes to $q2$;
    - One BSCC: $\{q1, q2\}$
    - Average counter change: $(\frac{1}{4}, -\frac{3}{4})$
- Strategy 2: $q1$ always goes to $q3$;
    - One BSCC: $\{q1, q3\}$
    - Average counter change: $(-\frac{1}{2}, \frac{1}{2})$



Termination in Probabilistic VASS[6]

Take a strongly-connected pVASS. Let $\mathbf{i}_1, \ldots, \mathbf{i}_k$ be average counter changes corresponding to all BSCCs induced by MD strategies.

Theorem

The given pVASS terminates in linear time iff there exists a positive vector $\mathbf{w} \in \mathbb{R}_+^k$ such that $\mathbf{i}_l \cdot \mathbf{w} < 0$ for all $l \in 1..k$.
Otherwise complexity at least quadratic.

Example:
$\mathbf{i_1} = (\frac{1}{4}, -\frac{3}{4})$, $\mathbf{i_2} = (-\frac{1}{2}, \frac{1}{2})$
Take $\mathbf{w} = (2, 1)$. Then $\mathbf{i_1} \cdot \mathbf{w} = -\frac{1}{4}$ and $\mathbf{i_2} \cdot \mathbf{w} = -\frac{1}{2}$.
Hence the pVASS terminates in linear time.

[6] Brázdil et al., "Deciding Fast Termination for Probabilistic VASS with Nondeterminism".

Termination in Probabilistic VASS

How do we check if there is such a **w**?

Definition

A vector $\mathbf{v}$ is *achievable* in a pVASS iff for some strategy σ and initial state p_0, $\mathbb{E}^{\sigma}_{p_0}[\liminf_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} \mathbf{u}_i] \geq \mathbf{v}$.

where $\liminf_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} \mathbf{u}_i$ is the *expected mean-payoff* of an infinite path $p_0, \mathbf{u}_1, p_1, \mathbf{u}_2, \ldots$ with counter updates $\mathbf{u_1}, \mathbf{u_2} \ldots$.
Known result: Achievability of a rational vector is decidable in polynomial time.
Brázdil et al.[7] prove the following lemma:

Lemma

In any strongly-connected pVASS, the vector $\mathbf{0}$ *is achievable iff there is no* $\mathbf{w} > 0$ *such that* $\mathbf{i}_l \cdot \mathbf{w} < 0$ *for all* $l \in 1..k$.

The theorem can be extended to *DAG-like* pVASS.

[7] Brázdil et al., "Deciding Fast Termination for Probabilistic VASS with Nondeterminism".

Original Project Roadmap

We were planning to:

- Find a class of probabilistic programs which can be (exactly) represented as pVASS
- Try to extend the theory of pVASS

Translation - first attempt

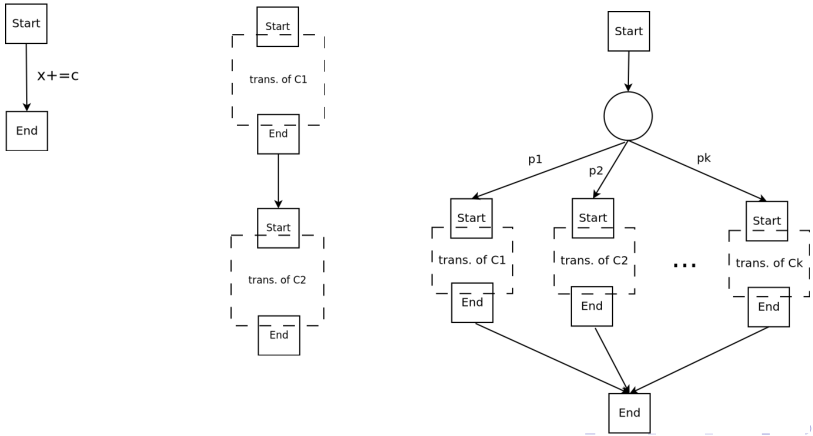
Start with the following:

$\langle C \rangle ::= \texttt{skip}$	no-operation
$\mid\ x+=c$	constant increment
$\mid\ C_1; C_2$	sequential composition
$\mid\ \texttt{if}(x>0)\{C_1\}\ \texttt{else}\{C_2\}$	conditional
$\mid\ \texttt{while}(x>0)\{C\}$	guarded loop
$\mid\ \langle p_1 : C_1, \ldots, p_k : C_k \rangle$	probabilistic choice

Translation - first attempt

Most rules straightforward:

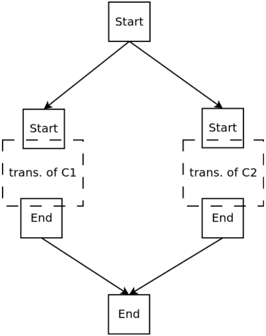
$x+=c$: $C_1; C_2$: $\langle p_1 : C_1, \ldots, p_k : C_k \rangle$:



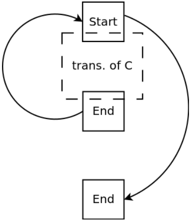
Translation - first attempt

What about control flow? We cannot test counter values!
Can try abstracting by nondeterminism:

$\texttt{if}(x > 0)\{C_1\}\ \texttt{else}\{C_2\}$:



$\texttt{while}(x > 0)\{C\}$:



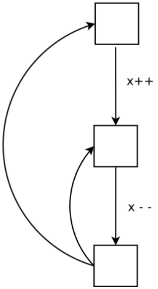
Idea: demonic nondeterminism should try to take longer branches and execute loops to the end.

Problems

Demonic nondeterminism may *overapproximate* loop runtime:

```
while(x>0) {
  x++;
  while(x>0) {
    x--;
  }
}
```

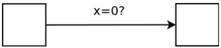
Linear termination



Never terminates!

How to fix this?

Idea 1: Extend to VASS with zero-tests



- Finkel and Sangnier[8] proved that termination for VASS with one counter tested for zero is decidable
- Bràzdil et al.[9] proved that almost-sure termination for pVASS with one counter tested for zero is decidable (in non-degenerous cases) (tested counter not causing termination). But no efficient algorithm
- No known results on linear termination

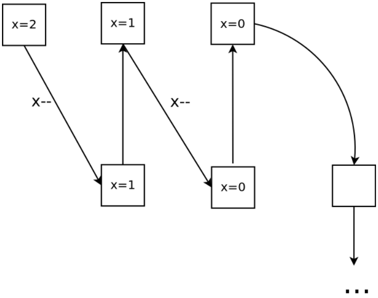
[8] Alain Finkel and Arnaud Sangnier. "Mixing Coverability and Reachability to Analyze VASS with One Zero-Test". In: *SOFSEM 2010: Theory and Practice of Computer Science, 36th Conference on Current Trends in Theory and Practice of Computer Science, Spindleruv Mlýn, Czech Republic, January 23-29, 2010. Proceedings.* 2010.

[9] Tomás Brázdil et al. "Zero-reachability in probabilistic multi-counter automata". In: *Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), CSL-LICS '14, Vienna, Austria, July 14 - 18, 2014.* 2014.

How to fix this?

Idea 2: Consider (bounded) structural counters as *parts of control states*

```
x = 2;
while(x>0) {
  x--;
}
...
```



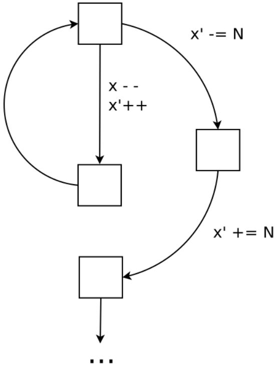
Problem: Algorithm from (Brázdil et al.[10]) only applicable to **fixed** pVASS

[10] Brázdil et al., "Deciding Fast Termination for Probabilistic VASS with Nondeterminism".

How to fix this?

Idea 3a: Add an opposite counter x' to each “structural” counter x ($x + x' = N$)

```
while(x>0) {
  x--;
}
...
```



Limitation: counter x must be bounded by initial value N

Problem: increments depend on N, so algorithm from (Brázdil et al.) **not applicable**

How to fix this?

Idea 3b: Use idea from (Czerwiński et al.[11]) to translate zero-tests to programs with constant increments:

Initially: $d = c \cdot R$, $x + x' = R$, at termination required: $d = 0$, $c \geq 0$.
Idea: $x = 0$ iff we can execute each loop R times. Initial and termination condition guarantee we executed R iteration of each of c loops.

[11] Wojciech Czerwinski et al. "The reachability problem for Petri nets is not elementary". In: June 2019, pp. 24–33.

How to fix this?

Problems:

- This approach requires a specific initial configuration ($d = c \cdot R$) and specific condition on final values ($d = 0, c \geq 0$)
- Theory from (Bràzdil et al.[12]) not applicable, difficult to extend
- And that is before we even consider probabilities!

[12] Brázdil et al., "Deciding Fast Termination for Probabilistic VASS with Nondeterminism".

How to fix this?

Idea 4: Use a result from (Leroux[13]):

- **Theorem 17.** *Let* $G = (Q, \mathbf{A}, E)$ *be a strongly connected VASS. For every* $p(\mathbf{x}) \xrightarrow{\pi} q(\mathbf{y})$, *the values* $\mathbf{y}[i]$ *where* i *is a non iterable index, and the number of occurrences of non iterable edges in* π *are bounded by:*

$$[(1 + ||\mathbf{x}||)^2 d^2 (3||\mathbf{A}||.|Q|)^{15d^4}]^{4^{d|E|}}$$

where:

An *iteration scheme* of a VASS G is a finite sequence $\sigma_1, \ldots, \sigma_k$ of cycles such that:

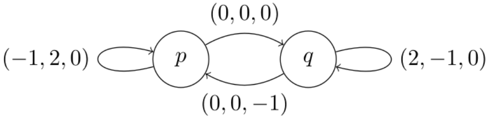
$$\bigwedge_{j=1}^{k} ||\Delta(\sigma_j)||^- \subseteq ||\Delta(\sigma_1) + \cdots + \Delta(\sigma_k)||^+$$

i.e. total scheme displacement nonnegative and negative indexes from one cycle are positive in full scheme

[13] Jérôme Leroux. "Polynomial Vector Addition Systems With States". In: *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*. 2018.

How to fix this?

Example:



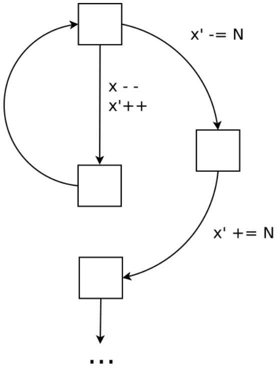
$(p, (-1, 2, 0), p), (q, (2, -1, 0), q)$ iteration scheme with displacement $(1, 1, 0)$

How to fix this?

- Idea: Reuse the idea with opposite counters
- If VASS has no iteration scheme for all N, runtime polynomial in N!

But...

- Whether VASS has iteration scheme actually depends on N



Loop acceleration (Frohn[14])

Consider the loop:

$$\texttt{while } \phi(\mathbf{x}) \texttt{ do } \mathbf{x} \leftarrow \mathbf{a}(\mathbf{x})$$

where $\mathbf{x} = (x_1, \ldots, x_n)$ integer-valued. Write $\mathbf{x} \to^n \mathbf{x}'$ if $\mathbf{x}$ becomes $\mathbf{x}'$ in n iterations.

Definition

A sound (underapproximating) **acceleration technique** computes a formula $\psi(\mathbf{x}, \mathbf{x}', n)$ over $\mathbf{x}$, $\mathbf{x}'$, $n > 0$ such that:

$$\psi(\mathbf{x}, \mathbf{x}', n) \implies \mathbf{x} \to^n \mathbf{x}'$$

If we also have $\psi(\mathbf{x}, \mathbf{x}', n) \iff \mathbf{x} \to^n \mathbf{x}'$, the technique is **exact**.

Idea: describe the behaviour of the loop by a single parametric formula
Overapproximating technique: $\psi(\mathbf{x}, \mathbf{x}', n) \impliedby \mathbf{x} \to^n \mathbf{x}'$

[14] Florian Frohn. "A Calculus for Modular Loop Acceleration". In: *Tools and Algorithms for the Construction and Analysis of Systems - 26th International Conference, TACAS 2020, Dublin, Ireland*. 2020.

Loop acceleration

Well-studied technique for deterministic programs.

Example (Frohn[15]):

$$\texttt{while}\ ((x_1 > 0) \wedge (x_2 > 0))\ \texttt{do}$$
$$x_1 \leftarrow x_1 - 1$$
$$x_2 \leftarrow x_2 + 1$$

The formula:

$$(x_1' = x_1 - n) \wedge (x_2' = x_2 + n) \wedge (x_2 > 0) \wedge (x_1 - n + 1 > 0)$$

is computed by an exact acceleration of the loop.

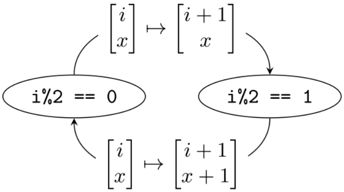
[15] Frohn, "A Calculus for Modular Loop Acceleration".

Loop acceleration via VASS

Idea from (Silverman and Kincaid[16]) using rational-valued vector addition systems with states and resets ($\mathbb{Q}$-VASRS);
Compute $\mathbb{Q}$-VASRS V and linear transformation $S \in \mathbb{Q}^{n \times m}$ overapproximating loop's reachability relation:

$$\mathbf{x} \to^* \mathbf{x}' \implies S\mathbf{x} \to_V^* S\mathbf{x}'$$

```
int x = 0; i = 1
while (*) do
  if i%2 == 0 then
    i := i + 1
  else
    i := i + 1
    x := x + 1
```



We can prove e.g. $2x \leq i$.

[16] Jake Silverman and Zachary Kincaid. "Loop Summarization with Rational Vector Addition Systems". In: *Computer Aided Verification - 31st International Conference, CAV 2019, New York City, NY, USA, July 15-18, 2019, Proceedings, Part II.* 2019.

Loop acceleration via VASS

- (V, S) computable in polynomial size
- (V, S) guaranteed to be the best $\mathbb{Q}$-VASRS approximation
- Reachability in $\mathbb{Q}$-VASRS computable in polynomial time by (Haase and Halfon[17])

Question: can we extend it to probabilistic loops?
Problem: Question: can we extend it to probabilistic loops?

[17] Christoph Haase and Simon Halfon. "Integer Vector Addition Systems with States". In: *Reachability Problems - 8th International Workshop, RP 2014, Oxford, UK, September 22-24, 2014. Proceedings.* 2014.

Reachability in probabilistic loops

Possible definition of reachability:

$$\mathcal{P}_{reach}(\mathbf{x}, S) \triangleq \sum_{\mathbf{x} \xrightarrow{p'} \mathbf{x}', \mathbf{x}' \in S} p'$$

where S set of states.

- Quantitative reachability: find p' (or a bound in p')
- Qualitative reachability: check if $p' = 1$.

Reachability in probabilistic loops

Example:

```
while(*) {
  x++ [1/2] skip;
  k++;
}
```

Let $S = \{(s, k) \mid s = k = 3\}$. Then $\mathcal{P}_{reach}((0,0), S) \leq \frac{1}{8}$

Reachability in probabilistic loops

Idea 1: use pVASS?

- Problem: no known results on reachability in pVASS, not covered by (Bràzdil et al.[18])

[18] Brázdil et al., "Deciding Fast Termination for Probabilistic VASS with Nondeterminism".

Reachability in probabilistic loops

Idea 2: use Markov chains?

- Represent loop as MC with *countablty infinite* state space (state= control state + variable values)
- Synthesise a finite-state MC *simulating* the loop MC (Baier[19])
  - Need to *aggregate* states, approximation specific to S
- Finite-state MC overapproximates reachability relation
- In finite-state MCs, reachability polynomial in # states ($=$ $|\text{control states}| \cdot B^k$ for k variables with B values)

[19] Christel Baier et al. "Comparative branching-time semantics for Markov chains". In: *Information and Computation* 200.2 (2005), pp. 149–214.

Reachability in probabilistic loops

Idea 3: use probabilistic PDA or probabilistic one-counter automata (pOC)[20]

- quantitative reachability in PSPACE
- qualitative reachability polynomial for pOC

[20] Tomás Brázdil, Stefan Kiefer, and Antonin Kucera. "Efficient Analysis of Probabilistic Programs with an Unbounded Counter". In: *J. ACM* 61.6 (2014), 41:1–41:35.

To sum up

- Translating programs to pVASS with equivalent termination complexity seems very difficult (if not impossible)
- Computing loop summaries using pVASS as a subcomponent seems more promising, but we are at a very early stage.
- What are your thoughts?