A Compositional and Monotone Approach to Termination
Towards a more dependable and scalable termination analysis

Shaowei Zhu
Advisor: Zachary Kincaid
Princeton University
May 2020
Background
program correctness = partial correctness + termination proof

It does the right thing if it terminates

It does terminate
Programs as transition systems

• A program statement defines a transition on the state space $S$

• Transition system $\langle S, R \rangle$
  
  • $S$: state space
  
  • $R \subseteq S \times S$: transition relation
**Ranking functions**

- Loop terminates iff it has a ranking function

- Ranking function $r$ for a loop with state space $S$ and body $R$
  - $r : S \rightarrow B$ is a mapping from state space $S$ into a set $B$ equipped with a well-founded relation $\succ$
  - $(s_1, s_2) \in R \Rightarrow r(s_1) > r(s_2)$

- Examples: linear ranking functions, lexicographic linear ranking functions

```plaintext
// ranking function: x
while (x > 0) {
    x = x - 1;
}
```

```plaintext
// ranking function: < i, j >
while (i > 0) {
    if (j > 0) {
        j--;
    } else {
        j = 100;
        i--;
    }
}
```
• Complete procedure for lexicographic ranking function synthesis for a single loop (Gonnord et al., 2015)

• Deciding termination of linear transition systems (Tiwari, 2004)

• Weakest terminating pre-conditions for octagonal relations (Bozga et al., 2012)

• …

• Repeatedly sample a lasso-shaped trace and synthesize ranking function for it (Cook et al., 2006)

• Extend single-loop ranking function synthesis to whole program termination by analyzing paths between cut points (Gonnord et al., 2015)

• Guess ranking function candidates, and then synthesize conditions under which they become real ranking functions (Cook et al., 2008)

• …
Motivation
“Because Infer is *compositional*, it can operate incrementally, quickly producing results on a code diff even if the diff is part of a codebase in the millions of lines”

— Peter O’Hearn
Want: a compositional analysis of termination
analyze each loop in isolation and combine the analysis results
It brings us: parallelism and scalability, incremental analysis, etc.
Want: a compositional analysis of termination

Requires: an analysis that generates sufficient conditions for termination
• Consider a compositional analysis for termination that looks at each loop in isolation

• When analyzing the while loop, we don’t know what values could \( x \) take

• The analysis need to figure out a sufficient condition for termination and checks if the context implies that condition

```c
if (__VERIFIER_nondet_int() \neq 0) {
    x = 1;
} else {
    x = -1;
}

// conditions: \( x > 0 \) or \( x < 0 \)
while (y < 100 && z < 100) {
    y = y + x;
    z = z - x;
}
```
“Tools can understand our programs, but we cannot understand our tools”

— Leino and Moskal
Heuristics

widely used in modern tools, but lead to unpredictability in the analysis
Ultimate Automizer

- #1 tool in Software Verification Competition termination category
- Supports linear, lexicographic linear, multiphase, piecewise ranking functions, etc.
- Can prove this program terminates within a few seconds, but ...

```c
int main() {
    int x, y, z, a, b;
    int n;

    while (z >= 0 && _VERIFIER_nondet_int()) {
        z = z - 1;
    }

    if (a == b) {
        while (x >= 0 && y >= 0) {
            y = y - z;
            while (z > 0) {
                z = z - 1;
                y = y + 2 * z - x;
            }
            x = x + a - b - 1;
        }
    }

    return 0;
}
```
# Ultimate Automizer

- #1 tool in Software Verification Competition termination category
- Supports linear, lexicographic linear, multiphase, piecewise ranking functions, etc.
- Can prove this program terminates within a few seconds, but it will not terminate after 30 minutes on this slightly modified program which ought to be “easier” to reason about.
Want: a predictable analysis of termination
produces a more precise result when given more precise information
This work presents a termination analysis that is:

- **Predictable**
  - Adding information to program leads to more precise analysis results

- **Compositional**
  - Analyzing on a per-loop basis
A monotone and compositional analysis for conditional termination
How we achieve compositionality and predictability?

- **Compositionality**: compute termination conditions for a program by analyzing its components and composing the results
  - Extend Tarjan’s framework to termination analysis

- **Predictability**: give a monotone procedure that computes a sufficient terminating precondition for a program
  - Exploit the compositional framework
    - Reduce termination of programs to termination of single loops
    - Access to a summary for the body of each loop
  - Utilize decision procedure for linear loops to generate terminating pre-conditions for summaries
  - Abstract transition formulas into linear models that can be handled by Tiwari’s method
An algebraic framework for program analysis

• (Tarjan, 1981) views the control flow graph of the program as a labelled graph where labels are transitions relations

• Can efficiently compute regular expressions that recognize all paths between two program locations

• Certain program analysis can be defined using interpretation of regular expressions that represent sets of paths

\[ a \in \Sigma \]
\[ e \in \text{RegExp}(\Sigma) ::= a | 0 | 1 | e_1 + e_2 | e_1e_2 | e^* \]
Interpreting path expressions as transition formulas

• Transition formula $\textbf{TF}$

  • A formula in linear integer arithmetic over program variables and primed program variables, and loop counter

• Describes a transition relation

• Compositional recurrence analysis (Kincaid et al., 2015) interprets path expressions as transition formulas
An algebraic framework for compositional termination analysis

We augment Tarjan’s framework with $\omega$-regular expressions:

- **State formula SF**
  - formula in LIA over program variables that describes a set of program states

- **Termination analysis defined as interpretation of $\omega$-path expressions**
  - atc: $\text{TF} \rightarrow \text{TF}$, approximate transitive closure operator provides the summary of loop body (Kincaid et al., 2015)
  - swf: $\text{TF} \rightarrow \text{SF}$, gives a set of initial states from which the transition is guaranteed to terminate

- Let $p$ be the set of $\omega$-paths starting at program entry. If $s$ is a state from which there is an non-terminating execution, then $s \notin \mathcal{T}(p)$

\[
\begin{align*}
a & \in \Sigma \\
e & \in \text{RegExp}(\Sigma) := a | 0 | 1 | e_1 + e_2 | e_1 e_2 | e^* \\
f & \in \omega\text{-RegExp}(\Sigma) := e^\omega | e; f | f_1 \oplus f_2 \\
S^{\mathcal{T}} & \triangleq \text{TF} \\
T_1 +^{\mathcal{T}} T_2 & \triangleq T_1 \lor T_2 \\
0^{\mathcal{T}} & \triangleq \text{False} \\
1^{\mathcal{T}} & \triangleq \bigwedge_{x \in \text{Var}} x' = x \\
T_1 .^{\mathcal{T}} T_2 & \triangleq \exists \text{Var}' . T_1[\text{Var}' \mapsto \text{Var}'] \land T_2[\text{Var} \mapsto \text{Var}'] \\
T^{\ast^{\mathcal{T}}} & \triangleq \text{atc}(T) \\
\mathcal{T}[(u, x := e, v)] & \triangleq x' = e \land \bigwedge_{y \neq x \in \text{Var}} y' = y \\
\mathcal{T}[(u, \text{assume}(c), v)] & \triangleq c \land \bigwedge_{x \in \text{Var}} x' = x \\
L^{\mathcal{T}} & \triangleq \text{SF} \\
T^{\omega^{\mathcal{T}}} & \triangleq \neg \text{swf}(T) \\
T^{\mathcal{T}} S & \triangleq \exists \text{Var}' . T \land S[\text{Var} \mapsto \text{Var}'] \\
S_1 \oplus^{\mathcal{T}} S_2 & \triangleq S_1 \lor S_2
\end{align*}
\]
How we achieve compositionality and predictability?

• **Compositionality**: compute termination conditions for a program by analyzing its components and composing the results
  - Extend Tarjan’s framework to termination analysis

• **Predictability**: give a monotone procedure that computes a sufficient terminating precondition for a program
  - Exploit the compositional framework
    - Reduce termination of programs to termination of single loops (swf operator)
    - Access to a summary for the body of each loop (atc operator)
  - Utilize decision procedure for linear loops to generate terminating pre-conditions for summaries
    - Abstract transition formulas into linear models that can be handled by Tiwari’s method
Termination analysis of the whole program

Define swf() operator that finds terminating preconditions for a transition formula

Compositional Recurrence analysis (Kincaid et al., 2015)

Reduce in a monotone manner

Terminating preconditions for linear transition systems with rational eigenvalues based on (Tiwari, 2004)
Tiwari has a way to synthesize conditions for termination, but ...
Long-term Dynamics of Linear Transition Systems

• Suppose the loop to the right does not terminate after $k$ iterations, then what are the values of $x$, $y$, and $z$?

$$
\begin{bmatrix}
x' \\
y' \\
z'
\end{bmatrix} =
\begin{bmatrix}
1 & 1 & 0 \\
0 & 1 & 1 \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
z
\end{bmatrix}
$$

while $(x > 0)$ {
  $x = x + y$;
  $y = y + z$;
}

$$
\begin{bmatrix}
x^{(k)} \\
y^{(k)} \\
z^{(k)}
\end{bmatrix} =
\begin{bmatrix}
1 & 1 & 0 \\
0 & 1 & 1 \\
0 & 0 & 1
\end{bmatrix}
^k
\begin{bmatrix}
x^{(0)} \\
y^{(0)} \\
z^{(0)}
\end{bmatrix}
$$
Long-term Dynamics of Linear Transition Systems

• Suppose the loop to the right does not terminate after $k$ iterations, then what are the values of $x$, $y$, and $z$?

  • $z^{(k)} = z^{(0)}$
  • $y^{(k)} = y^{(0)} + kz^{(0)}$
  • $x^{(k)} = x^{(0)} + ky^{(0)} + \frac{k(k - 1)}{2}z^{(0)}$

• Obviously the value of $x$ affects termination, so we look at

  • $x^{(k)} = x^{(0)} + k(y^{(0)} - \frac{1}{2}z^{(0)}) + k^2\frac{z^{(0)}}{2}$

```c
while (x > 0) {
    x = x + y;
    y = y + z;
}
```
Long-term Dynamics of Linear Transition Systems

- \( x^{(k)} = x^{(0)} + k(y^{(0)} - \frac{1}{2}z^{(0)}) + k^2 \frac{z^{(0)}}{2} \)

- Consider when \( k \) gets large enough \( k^2 \gg k \gg 1 \)
  - If \( \frac{z^{(0)}}{2} \neq 0 \) then \( x^{(k)} \approx k^2 \frac{z^{(0)}}{2} \)
  - If \( \frac{z^{(0)}}{2} = 0 \) but \( y^{(0)} - \frac{1}{2}z^{(0)} \neq 0 \), then
    \( x^{(k)} \approx k(y^{(0)} - \frac{1}{2}z^{(0)}) \)
  - If \( \frac{z^{(0)}}{2} = 0 \), \( y^{(0)} - \frac{1}{2}z^{(0)} = 0 \), then \( x^{(k)} \approx x^{(0)} \)

```java
while (x > 0) {
    x = x + y;
    y = y + z;
}
```
Long-term Dynamics of Linear Transition Systems

If the loop never terminates, \( k \) can actually get large enough and \( x^{(k)} > 0 \) still holds.

- If \( \frac{z^{(0)}}{2} \neq 0 \) then \( x^{(k)} \approx k^2 \frac{z^{(0)}}{2} > 0 \Rightarrow \frac{z^{(0)}}{2} > 0 \)

- If \( \frac{z^{(0)}}{2} = 0 \) but \( y^{(0)} - \frac{1}{2}z^{(0)} \neq 0 \), then

\[
x^{(k)} \approx k(y^{(0)} - \frac{1}{2}z^{(0)}) > 0 \Rightarrow y^{(0)} - \frac{1}{2}z^{(0)} > 0
\]

- If \( \frac{z^{(0)}}{2} = 0, \ y^{(0)} - \frac{1}{2}z^{(0)} = 0 \), then \( x^{(k)} \approx x^{(0)} > 0 \)

\[
\text{while } (x > 0) \{ \\
\quad x = x + y; \\
\quad y = y + z;
\} 
\]
Long-term Dynamics of Linear Transition Systems

Thus non-termination implies the following:

\[
\frac{z^{(0)}}{2} > 0, \text{ or } \frac{z^{(0)}}{2} = 0 \land y^{(0)} - \frac{1}{2} z^{(0)} > 0, \text{ or } \frac{z^{(0)}}{2} = 0 \land y^{(0)} - \frac{1}{2} z^{(0)} = 0 \land x^{(0)} > 0
\]

The negation of the above condition implies termination! (Tiwari, 2004)
Dominant Term Analysis (DTA) exploits long-time dynamics

- Using Tiwari’s analysis to generate terminating conditions requires
  1. Finding loop guards expressed in linear terms of program variables (polyhedral guards)
  2. Linear loop update as matrix multiplication $x' = Ax$
     - Exists a linear ordering on the eigenvalues of $A$

```plaintext
while $x > 0$
{
  $x = x + y$
  $y = y + z$
}
```

\[
\begin{bmatrix}
  x^{(k)} \\
  y^{(k)} \\
  z^{(k)}
\end{bmatrix} =
\begin{bmatrix}
  1 & 1 & 0 \\
  0 & 1 & 1 \\
  0 & 0 & 1
\end{bmatrix}^k
\begin{bmatrix}
  x^{(0)} \\
  y^{(0)} \\
  z^{(0)}
\end{bmatrix}
\]

$k^2 \gg k \gg 1$
Dominant Term Analysis (DTA) exploits long-time dynamics

• Using Tiwari’s analysis to generate terminating conditions requires
  
  1. Finding loop guards expressed in linear terms of program variables (polyhedral guards)
  
  2. Linear loop update as matrix multiplication $\mathbf{x}' = A\mathbf{x}$
  
  • Exists a linear ordering on the eigenvalues of $A$

Q: When is this possible? What if we have to compare $e^{\frac{3k\pi}{3}}i$ and $e^{\frac{3k\pi}{4}}i$?
A: That is hard to do. Want to consider only matrices with \textcolor{red}{real/rational eigenvalues}.
Tiwari has a way to synthesize conditions for termination, but …
Tiwari has a way to synthesize conditions for termination, but only applies to *linear transition systems with polyhedral guards and rational spectra*. 
Termination analysis of the whole program

Define swf() operator that finds terminating preconditions for a transition formula

Compositional Recurrence analysis (Kincaid et al., 2015)

Reduce using theory of best abstractions in a monotone manner

Terminating preconditions for linear transition systems with rational eigenvalues based on (Tiwari, 2004)
Simulation as abstraction

- Transition system $A = \langle S_A, R_A \rangle$
- Transition system $\overline{A} = \langle \overline{S_A}, \overline{R_A} \rangle$

$p : S_A \rightarrow \overline{S_A}$ is a simulation from transition system $A$ to transition system $\overline{A}$ if for all $A$-transitions $(a, a') \in R_A$, $(p(a), p(a')) \in \overline{R_A}$

- Existence of simulation $p$ from $A$ to $\overline{A}$ has the consequence that termination of $\overline{A}$ implies termination of $A$

- If have a terminating condition $C$ for $\overline{A}$ then we can get a sufficient terminating condition for $A$ as $p^{-1}(C)$
Simulation as abstraction

- Transition system $A = \langle S_A, R_A \rangle$
- Transition system $\tilde{A} = \langle \tilde{S}_A, \tilde{R}_A \rangle$
- $p : S_A \rightarrow \tilde{S}_A$ is a simulation from transition system $A$ to transition system $\tilde{A}$ if for all $A$-transitions $(a, a') \in R_A$, $(p(a), p(a')) \in \tilde{R}_A$
- Existence of simulation $p$ from $A$ to $\tilde{A}$ has the consequence that termination of $\tilde{A}$ implies termination of $A$
- If have a terminating condition $C$ for $\tilde{A}$ then we can get a sufficient terminating condition for $A$ as $p^{-1}(C)$

While $(x + y \geq 0)$ {
  \begin{align*}
  &\text{if } \text{VERIFIER\_nondet\_int()} \{
  \begin{align*}
  &x = x - 1; \\
  \end{align*}
  \}
  \text{else }
  \begin{align*}
  &y = y - 1;
  \end{align*}
  \}
  \end{align*}

Transition formula of $A$: $x + y \geq 0 \land (x' = x - 1 \land y' = y) \lor (y' = y - 1 \land x' = x)$

Simulation $s : \begin{bmatrix} x \\ y \end{bmatrix} \rightarrow x + y$

Transition formula of $\tilde{A}$: $t \geq 0 \land t' = t - 1$
Theory of best abstractions

- \( \tilde{A} \) is a best abstraction of \( A \) within class \( \mathcal{C} \) with respect to a certain class of simulations, e.g., linear simulations.
Best abstraction and monotonicity

• We want to get sufficient terminating conditions for transition system $A$

• Suppose we have a procedure that computes $\text{swf}$ for transition systems in a class $\mathcal{C}$

  • Using the best abstraction $\overline{A}$ and simulation $p : A \to \overline{A}$, we get sufficient terminating conditions $p^{-1}(\text{swf}(\overline{A}))$

  • Using some other abstraction $B$ and simulation $q : A \to B$, we get sufficient terminating conditions $q^{-1}(\text{swf}(B))$

• Under reasonable conditions, using the best abstraction yields the weakest terminating preconditions, compared to using any other abstraction in $\mathcal{C}$
Termination analysis of the whole program

Define `swf()` operator that finds terminating preconditions for a transition formula

Reduce using **theory of best abstractions** in a monotone manner

Terminating preconditions for linear transition systems with rational eigenvalues based on (Tiwari, 2004)

Composeranal Recurrence analysis (Kincaid et al., 2015)
Define \textit{swf()} operator that finds terminating preconditions for a transition formula.

**Termination analysis of the whole program**

Transition formula

**Best linear transition system abstraction with rational eigenvalues**

Terminating preconditions for linear transition systems with \textbf{rational} eigenvalues (Tiwari)

Compositional Recurrence analysis (Kincaid et al., 2015)

Example: Transition formula $x' = x \land x' = 0$ has abstractions $x' = x$, $x' = 2x$, $x' = 3x$, etc. but does not have a best abstraction.
Theorem: transition formulas in LIA have best abstractions in DATS w.r.t. linear simulations

- DATS: deterministic affine transition system
  - State space $\mathbb{Q}^n$
  - $(u, v) \in R \iff v = Tu + c \land Du = d$

- cf. TDATS (total deterministic affine transition system)
  - $(u, v) \in R \iff v = Pu + c$

- Note: transition formulas in LIA does not necessarily have best abstractions in TDATS w.r.t. linear simulations
Simplifying DATS

• DATS: deterministic affine transition system

• \((u, v) \in R \iff v = Tu + c \land Du = d\)

• Simplification 1: easy to homogenize this into a DLTS (deterministic linear transition system)

• \((u', v') \in R \iff v' = T'u' \land D'u' = 0\)

• Simplification 2: construct a representation matrix \(T_0\) that behaves exactly the same as \(T\) on the domain we care about and maps everything else to 0, then study the exponentiated version of \(T_0\) for the termination of DLTS

Define the domain of a transition relation as
\[
\text{dom}(R) \triangleq \{ x \in S : \exists y . xRy \}
\]

Define the invariant domain of transition relation \(R\) as
\[
\text{dom}^*(R) \triangleq \bigcap_{n \in \mathbb{N}} \text{dom}(R^n)
\]

• Intuition: initial states outside of \(\text{dom}^*\) are certainly terminating pre-states, so only cares about states inside the invariant domain
Termination analysis of the whole program

Define swf() operator that finds terminating pre-conditions for a transition formula

Transition formula

Best linear transition system abstraction with rational eigenvalues

Terminating preconditions for linear transition systems with rational eigenvalues (Tiwari)

Compositional Recurrence analysis (Kincaid et al.)
Define swf() operator that finds terminating pre-conditions for a transition formula.

Termination analysis of the whole program

Compositional Recurrence analysis (Kincaid et al.)

Transition formula

Best LTS abstraction (Reps et al., 2004)

Best DLTS abstraction (Kincaid et al., 2018)

Best QDLTS abstraction (new)

Terminating preconditions for linear transition systems with rational eigenvalues (Tiwari)

(u, v) ∈ R ⇔ Av = Bu

(u, v) ∈ R ⇔ v = Tu ∧ Du = 0

Next: spectral theory of DLTS
Spectral theory of DLTS

• Spectrum of total deterministic transition systems (TDLTS) $v = Tu$
  
  $$\text{spec}(T) \triangleq \{ \lambda \in \mathbb{C} : \exists v, v \neq 0. Tv = \lambda v \}$$

• Spectrum of deterministic transition systems (DLTS) $v = Tu \land Du = 0$
  
  $$\text{spec}(T, D) \triangleq \{ \lambda \in \mathbb{C} : \exists v \in \text{dom}^*(T, D), v \neq 0. Tv = \lambda v \}$$

• QDLTS: DLTS with rational spectrum

• Theorem: the representation $T_0$ for a QDLTS $v = Tu \land Du = 0$ has rational eigenvalues
Best QDLTS abstraction

• Theorem: any DLTS has a best abstraction as a QDLTS w.r.t. linear simulations

• Proof is constructive and we give an algorithm to compute the best QDLTS abstraction

• Algorithm idea: repeatedly construct DLTS with lower dimensions but remains an over-approximation of the original DLTS, until we obtain a QDLTS or we run out of dimensions
Best QDLTS abstraction restricted to invariant domain is what Tiwari needs

• Properties of a QDLTS:
  • Every iteration is a linear update restricted to a linear space $x' = Tx \land Dx = 0$
  • Spectrum of the above DLTS is all rational

• Another look:
  • Starting from $x$ within the invariant domain, the updates are characterized by $x' = T_0x$
  • Eigenvalues of $T_0$ are all rational
  • Initial states outside of the invariant domain are certainly terminating pre-states
Best QDLTS abstraction restricted to invariant domain is what Tiwari needs

- Tiwari needs:
  - Polyhedral guards
  - Linear updates in the loop body $\mathbf{x'} = A\mathbf{x}$
  - Update matrix $A$ has rational eigenvalues

Procedure to compute the best polyhedral guards given a transition formula and simulation

- Another look:
  - Starting from $\mathbf{x}$ within the invariant domain, the updates are characterized by $\mathbf{x'} = T_0\mathbf{x}$
  - Eigenvalues of $T_0$ are all rational
  - Initial states outside of the invariant domain are certainly terminating pre-states
Define swf() operator that finds terminating pre-conditions for a transition formula

Transition formula $F$

Best DLTS abstraction $G$

Best QDLTS abstraction $Q$

Terminating preconditions for linear transition systems with rational eigenvalues (Tiwari)

Compositional Recurrence analysis (Kincaid et al.)
Define swf() operator that finds terminating pre-conditions for a transition formula

Terminating preconditions for linear transition systems with rational eigenvalues (Tiwari)

Transition formula $F$

Best DLTS abstraction $G$

Best QDLTS abstraction $Q$

Monotone

Sufficient preconditions for $F$ to terminate

Sufficient preconditions for $G$ to terminate

Sufficient preconditions for $Q$ to terminate

Linear simulation $S$

Linear simulation $T$
How we achieve compositionality and predictability?

- **Compositionality:** compute termination conditions for a program by analyzing its components and composing the results
  - Extend Tarjan’s framework to termination analysis
- **Predictability:** give a monotone procedure that computes a sufficient terminating precondition for a program
  - Exploit the compositional framework
    - Reduce termination of programs to termination of single loops (swf operator)
    - Access to a summary for the body of each loop (atc operator)
  - Utilize decision procedure for linear loops to generate terminating pre-conditions for summaries
    - Abstract transition formulas into linear models that can be handled by Tiwari’s method
Preliminary Experimental Results

- SV-COMP termination benchmarks
- Not geared towards conditional termination
- Subset containing only terminating programs
- Conclusions
  - Monotonicity and compositionally come at a cost
Future work

• Create more conditional termination benchmarks and evaluate the tool on those benchmarks

• Use termination arguments to enhance loop invariant generation

• Monotone conditional termination analysis for other ranking function templates

• Explore how mathematical theory of dynamical systems could further help with program analysis
Thanks for your attention!