Termination of Polynomial Programs

Aaron R. Bradley, Zohar Manna, and Henny B. Sipma

Dept. of Computer Science

Stanford University

A Simple Terminating Loop

Loop:

$$\tau_1 : \{x \geq y\} \rightarrow \{x' = x,\ y' = y + 1\}$$

Why does it terminate?

$x - y$ is a *ranking function*:

- $x - y$ is well-founded by $x \geq y$.
- $x - y$ decreases on each iteration:

$$\begin{aligned}(x - y)' - (x - y) &= x' - y' - x + y \\ &= x - y - 1 - x + y \\ &= -1\end{aligned}$$

A Simple Change, but not-so-simple Termination

Loop:

$$\tau_1 : \{x \geq y\} \rightarrow \{x' = x + 1,\ y' = y + x\}$$

Why does it terminate?

- $x - y$ is still well-founded by $x \geq y$.
- But $x - y$ does not decrease on each iteration (when $x \leq 0$).

Idea: Look at higher-order differences.

Termination by Higher-order Differences

- $x - y$ is well-founded by $x \geq y$.
- Decreasing?
  - First difference:

    $$(x - y)' - (x - y) \;=\; 1 - x$$

    x can be nonpositive, so not decreasing.
  - Second difference:

    $$(1 - x)' - (1 - x) \;=\; -1$$

    So *eventually* only decreasing.

Multiple Paths

What about multiple paths in the loop? Example:

$$\tau_1 : \{x \geq y\} \rightarrow \{x' = x + 1,\ y' = y + x,\ z' = z\}$$
$$\tau_2 : \{x \geq y\} \rightarrow \{x' = x - z,\ y' = y + z^2,\ z' = z - 1\}$$

Does it terminate?

- $x - y$ is well-founded by $x \geq y$.
- τ_1 like before.
- In τ_2, eventually y increases more quickly than x.
- But do they *interfere* with each other?

Interference

$$\tau_1 : \{x \geq 0\} \rightarrow \{x' = x + z,\ y' = y + 1,\ z' = z - 2\}$$
$$\tau_2 : \{x \geq 0\} \rightarrow \{x' = x + y,\ y' = y - 2,\ z' = z\}$$

Does it terminate?

- $y' = y + 1$ in τ_1 definitely interferes with $y' = y - 2$ in τ_2.
- But each transition terminates by itself.

Related Work

- Synthesis of *linear ranking functions*

  Colon & Sipma, 2001 : manipulation of polyhedral cones

  Colon & Sipma, 2002 : practical applications

  Podelski & Rybalchenko, 2004 : results for single path

- Nonlinear techniques

  Tiwari, 2004 : simple linear loops decidable

- Functional programs (huge literature)
  - *Size-change principle*: **Lee, Jones, & Ben-Amram, 2001**, technique for analyzing function-call structure
- Finite differences in programs
  - construction and automatic solving of recurrence equations
  - invariants, running times, termination
  - **Cohen, 1982**; **Katz & Manna, 1976**; **Wegbreit, 1975**

The Two Main Ideas

Idea 1: *Examine higher-order differences.*
Idea 2: *Look at eventual trends.*

The rest of the talk:

- Finite difference trees and approximations
- Termination analysis
- Experimental results

Polynomial Programs

Structure:

- *initial condition*: conjunction of polynomial assertions
- set of *guarded commands*
  - *guard*: conjunction of polynomial assertions
  - *update*: simultaneous assignment of polynomial expressions to variables

Theorem: Termination of polynomial programs is not even semi-decidable.

Finite Difference Trees (FDT's)

Represent finite differences by tree:

- τ_i-child of a node with expression $E(\mathbf{x})$ is labeled with expression

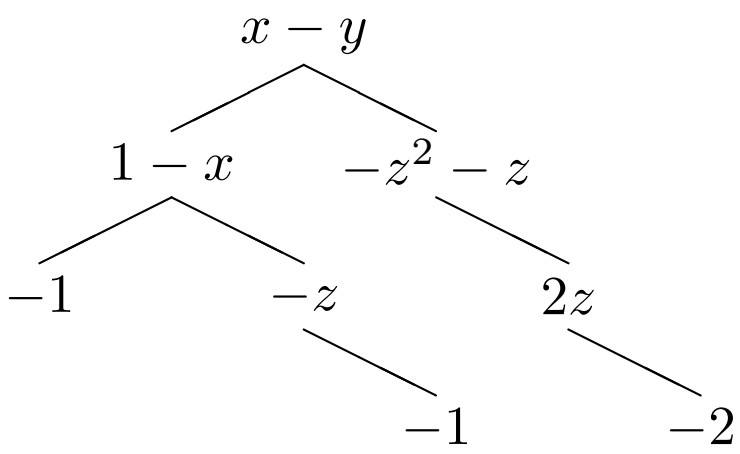
  $$E(\mathbf{x}') - E(\mathbf{x})$$

  where τ_i determines $\mathbf{x}'$.

- *Leaves* are nodes with only 0-children.

Example

$$\tau_1 : \{x \geq y\} \rightarrow \{x' = x + 1,\ y' = y + x,\ z' = z\}$$
$$\tau_2 : \{x \geq y\} \rightarrow \{x' = x - z,\ y' = y + z^2,\ z' = z - 1\}$$

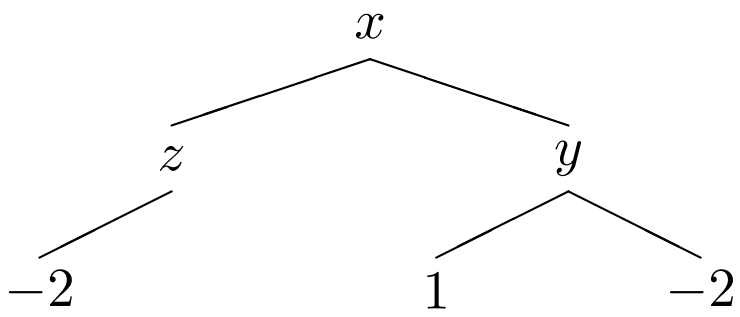


Remarks:

- τ_2 interferes with τ_1, but eventually aids progress.
- τ_1 does not interfere with τ_2.

Example

$$\tau_1 : \{x \geq 0\} \rightarrow \{x' = x + z,\ y' = y + 1,\ z' = z - 2\}$$
$$\tau_2 : \{x \geq 0\} \rightarrow \{x' = x + y,\ y' = y - 2,\ z' = z\}$$



Remarks:

- τ_1 negatively interferes with progress of τ_2.
- But we can still prove termination.

Finite *vs.* Infinite FDT

- A finite FDT indicates *polynomial* behavior.
- An infinite FDT indicates *exponential* behavior.
- Can use invariants for both cases.

More on this later.

Approximation: *Taylored* FDT

Idea: Use Taylor series expansion for analysis. But $\frac{\partial^2 f}{\partial x \partial y} \neq \frac{\partial^2 f}{\partial y \partial x}$.

Definition: *Critical nodes* are nodes $\Delta_T E(\mathbf{x})$ such that for all permutations σ, $\Delta_{\sigma(T)} E(\mathbf{x})$ is a constant expression.

Approximation: Construct *Taylored FDT* so that:

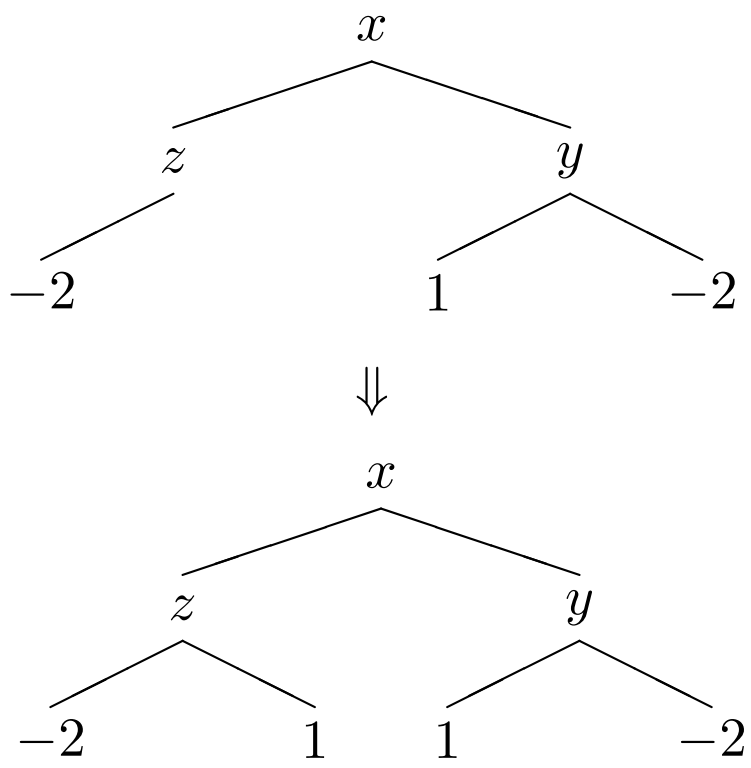
- Each *critical node* $\Delta_T E(\mathbf{x})$ has value

$$\max_{\sigma} \Delta_{\sigma(T)} E(\mathbf{x})$$

- All other nodes remain unchanged.

Example

$$\tau_1 : \{x \geq 0\} \rightarrow \{x' = x + z,\ y' = y + 1,\ z' = z - 2\}$$
$$\tau_2 : \{x \geq 0\} \rightarrow \{x' = x + y,\ y' = y - 2,\ z' = z\}$$



Analysis: Dominant Taylor Expression

Definition: *Dominant Taylor expression* (DTE) for a Taylored FDT:

$$\sum_{\Delta_T E(\mathbf{x}) \in critical\_nodes(t)} \frac{\prod_{\tau \in T} x_\tau}{|T|!} \Delta_T E(\mathbf{x})$$

where

- T are viewed as lists or multisets,
- one variable x_{τ_i} introduced per transition τ_i.

Remarks:

- x_{τ_i} counts the number of times τ_i is taken.
- Dominant terms of full Taylor series expansion for specific computations. So, $\frac{\partial^2 f}{\partial x \partial y} = \frac{\partial^2 f}{\partial y \partial x}$ where it matters.

Analysis: Dominant Taylor Expression

Idea:

- Partition set of infinite computations by transitions that occur infinitely often.
- Show that $E(\mathbf{x})$ eventually only decreases in each set.

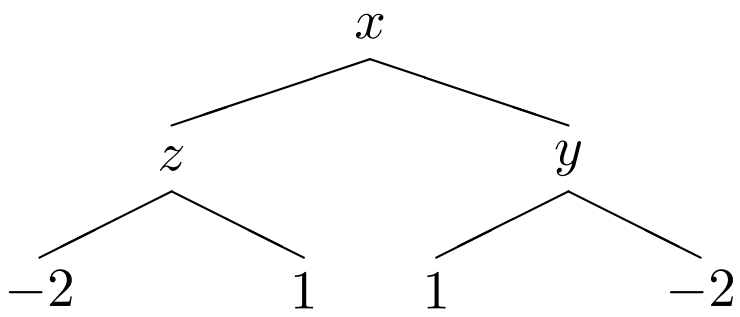
Form DTE for each nonempty subset $\mathcal{T}' \subseteq \mathcal{T}$.

- Each $\mathcal{T}'$ represents a set of infinite computations.
- Ensures that dominant terms really are dominant.

Proposition: If for each nonempty $\mathcal{T}' \subseteq \mathcal{T}$, the DTE of $E(\mathbf{x})$ with respect to $\mathcal{T}'$ *eventually only decreases as the length of the computation increases*, then $E(\mathbf{x})$ eventually only decreases.

Example

$$\tau_1 : \{x \geq 0\} \rightarrow \{x' = x + z,\ y' = y + 1,\ z' = z - 2\}$$
$$\tau_2 : \{x \geq 0\} \rightarrow \{x' = x + y,\ y' = y - 2,\ z' = z\}$$



Dominant Taylor expressions:

- $\{\tau_1\} \subseteq \{\tau_1, \tau_2\}$: $-2 \cdot \frac{1}{2!} x_1^2 = -x_1^2 \Rightarrow$ *decreases*
- $\{\tau_2\} \subseteq \{\tau_1, \tau_2\}$: $-2 \cdot \frac{1}{2!} x_2^2 = -x_2^2 \Rightarrow$ *decreases*

Example

- $\{\tau_1, \tau_2\} \subseteq \{\tau_1, \tau_2\}$:

$$\begin{aligned} -2 \cdot \tfrac{1}{2!} x_1^2 + \tfrac{1}{2!} x_1 x_2 + \tfrac{1}{2!} x_2 x_1 - 2 \cdot \tfrac{1}{2!} x_2^2 & \\ = \quad & -x_1^2 + x_1 x_2 - x_2^2 \\ = \quad & -r^2 \cos^2\theta + r^2 \cos\theta \sin\theta - r^2 \sin^2\theta \\ & \text{letting } x_1 = r\cos\theta,\ x_2 = r\sin\theta \\ = \quad & r^2(\cos\theta \sin\theta - 1) \\ = \quad & Q(r, \theta) \end{aligned}$$

But

$$\frac{\partial Q}{\partial r} = 2r(\cos\theta \sin\theta - 1) \quad \text{and} \quad \frac{\partial^2 Q}{\partial r^2} = 2(\cos\theta \sin\theta - 1)$$

Over $\theta \in [0, \frac{\pi}{2}]$, maximum of $\frac{\partial^2 Q}{\partial r^2}$ is -1. $\Rightarrow$ *decreases*

Example

- For each nonempty $\mathcal{T}' \subseteq \{\tau_1, \tau_2\}$, x was shown eventually only to decrease w.r.t. $\mathcal{T}'$ occuring i.o.
- Therefore, x eventually only decreases on all nonterminating computations.
- But x bounded from below by 0.
- So the loop terminates on all input.

Simple Analysis: Standard Condition

Definition: A τ_i-child is *decreasing* if

- its expression is at most some negative constant,
- or if its τ_i child is *decreasing* and its other children are decreasing or nonpositive.

Definition: A FDT meets the *standard condition* if for every τ_i, the root's τ_i-child is decreasing.

Simple Analysis: Standard Condition

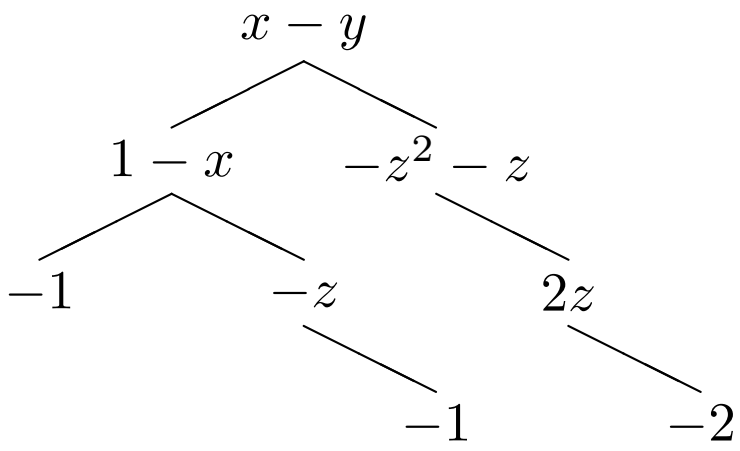
Proposition: If the FDT of $E(\mathbf{x})$ with respect to $\mathcal{T}$ meets the standard condition, then $E(\mathbf{x})$ eventually only decreases.

Remarks:

- Follows from earlier proposition.
- But also intuitive by itself.
- Simple: Form one FDT and examine.

Example

$$\tau_1 : \{x \geq y\} \rightarrow \{x' = x + 1,\ y' = y + x,\ z' = z\}$$
$$\tau_2 : \{x \geq y\} \rightarrow \{x' = x - z,\ y' = y + z^2,\ z' = z - 1\}$$



Check:

- Starting from the leaves, each node is found to be decreasing.
- Therefore, $x - y$ eventually only decreases.
- But $x \geq y$, so the loop terminates on all input.

Finite *vs.* Infinite FDT

- Finite FDT indicates qualitatively polynomial behavior: corresponds to finite number of nonzero derivatives.
- Infinite FDT indicates qualitatively exponential behavior: corresponds to infinite number of nonzero derivatives.
- But by the *standard condition*, if a node is proved to be at most some negative constant, then it is *decreasing*.

Idea: Use invariants to treat the infinite-FDT case.

Example

$\theta : \ n, x, y, N \geq 0$

$\{x \leq N\} \ \rightarrow \ \{n' = n + 1, \ x' = nx + y, \ y' = y + 1, \ N' = N\}$

Invariants: $n, x, y, N \geq 0$

$$N - x$$
$$\Big|$$
$$x - nx - y$$
$$\Big|$$
$$-1 - x + nx - n^2 x - ny$$
$$\vdots$$

Remarks:

- $n, x, y \geq 0 \rightarrow -1 - x + nx - n^2 x - ny \leq -1$
- Thus, the loop terminates on all input.

Experimental Results

Focus:

- Test scalability, applicability
- Code abstraction is currently naïve
- Suggest theoretical and practical refinements

Implementation:

- Written as a CIL feature.
- Abstracts C loops as sets of guarded commands.
- Applies **standard condition** to find *lexicographic* polyranking functions based on guards.

Experimental Results

Name	LOC	Loops	Abstract	Proved	P/A	P/L	Time (s)
`small1`	310	8	6	4	66	50	2
`vector`	361	13	13	12	92	92	2
`serv`	457	9	6	5	83	55	1
`dcg`	1K	55	53	53	100	96	3
`bcc`	4K	70	18	18	100	25	5
`sarg`	7K	108	25	25	100	23	104
`spin`	19K	652	132	121	91	18	30
`meschach`	28K	911	803	770	95	84	60
`f2c`	30K	437	113	96	84	21	52
`libavformat`	33K	452	268	216	80	47	55
`gnuplot`	50K	829	328	298	90	35	127
`gaim`	57K	594	60	52	86	8	106
`libavcodec`	75K	2227	1939	1869	96	83	183

Future Work

- Handle inequalities and more complex transition relations:
  - Can bound node expressions based on the **standard condition**.
- Rigorous code abstraction, *e.g.*,
  - aliasing,
  - interprocedural, possibly context-sensitive,
  - loop and function summaries as inequalities.
- Apply in complete verification frameworks (*e.g.*, *verification diagrams*).