Geometric Nontermination Arguments

Timo Bergerbusch

September 5, 2017

Introduction and Motivation

Example C-program

```
int main(){

  int a;
  int b=1;

  while(a+b>=4){
    a=3*a+b;
    b=2*b-5;
  }
}
```

- very basic C-program
- does it terminate?

  $\Rightarrow$ No!

  how can we prove this?

Integer Term Rewrite Systems (int-TRS)

int-TRS considered:

$$\overbrace{f_x}^{(1)} \rightarrow \overbrace{f_y}^{(2)} (v_1, \dots v_n) : | : cond_1$$

$$f_y(\underbrace{v_1, \dots v_n}_{(3)}) \rightarrow f_y (\underbrace{v'_1, \dots v'_n}_{(3)}) : | : \underbrace{cond_2}_{(4)}$$

(1) function symbol (no variables $\Rightarrow$ start)

(3) variables v'_i as linear updates of the variables v_j

(2) function symbol

(4) a set of (in)-equations mentioning v_j and v'_i

Reading: "rewrite $f_y(v_1, \dots, v_n)$ as $f_y(v'_1, \dots, v'_n)$ if *cond* holds"

Geometric Nontermination Argument (GNA)

- Idea: Split program into two parts:
  - *STEM*: variable initialization and declaration

```
int a;
int b=1;
```

  - *LOOP*: linear updates and *while*-guard

```
while(a+b>=4){
  a=3*a+b;
  b=2*b-5;
}
```

- apply the definition of a *geometric nontermination argument* by J. Leike and M. Heizmann

Example

The int-TRS of the example program would be:

$$
\begin{array}{l}
f_1 \quad\quad\quad \rightarrow f_2(1 + 3 * v_1, -3) : | : v_1 > 2 \ \ \&\& \ \ 8 < 3 * v_1 \\
f_2(v_1, v_2) \rightarrow f_2(3 * v_1 + v_2, v_3) : | : v_1 + v_2 > 3 \ \ \&\& \\
v_1 > 6 \ \ \&\& \ \ 3 * v_1 > 20 \ \ \&\& \ \ 5 + v_3 = 2 * v_2 \ \ \&\& \ \ v_3 < -10
\end{array}
$$

The first rule represents the *STEM*
Second rule represents the *LOOP*

Definition (Geometric Non Termination Argument)

A tuple of the form:

$$(x, y_1, \ldots, y_k, \lambda_1, \ldots, \lambda_k, \mu_1, \ldots, \mu_{k-1})$$

is called a *geometric nontermination argument* of size k for a program $= (STEM, LOOP)$ with n variables iff all of the following statements hold:

(domain) $x, y_1, \ldots, y_k \in \mathbb{R}^n$, $\lambda_1, \ldots \lambda_k, \mu_1, \ldots \mu_{k-1} \geq 0$

(init) x represents the *start term* ($STEM$)

(point) $A \begin{pmatrix} x \\ x + \sum_i y_i \end{pmatrix} \leq b$

(ray) $A \begin{pmatrix} y_i \\ \lambda_i y_i + \mu_{i-1} y_{i-1} \end{pmatrix} \leq 0$ for all $1 \leq i \leq k$

Note: $y_0 = \mu_0 = 0$ instead of case distinction

Definitons: Matrices

Definition (*Guard Matrix*, *Guard Constants*)

For $1 \leq i, j \leq n$ and m the number of guards not containing "$=$": The *Guard Matrix* $G \in \mathbb{Z}^{m \times n}$ is the matrix of coefficients $a_{i,j}$ of a variable v_i within the j-th guard. The *Guard Constants* $g \in \mathbb{Z}^m$ are the constant terms c_j within the j-th guard.

Definition (*Update Matrix*, *Update Constants*)

The *Update Matrix* $U \in \mathbb{Z}^{n \times n}$ and *Update Constants* $u \in \mathbb{Z}^n$ are analogously to the *Guard Matrix* and *Guard Constants*, considering the updates (right hand side) instead of the guards.

Reminder: int-TRS

$$f_1 \rightarrow f_2(1+3*v_1, -3) : | : v_1 > 2 \;\&\&\; 8 < 3*v_1$$
$$f_2(v_1, v_2) \rightarrow f_2(3*v_1+v_2, v_3) : | : v_1+v_2 > 3 \;\&\&$$
$$v_1 > 6 \;\&\&\; 3*v_1 > 20 \;\&\&\; 5+v_3 = 2*v_2 \;\&\&\; v_3 < -10$$

Example (*Guard Matrix*, *Guard Constants*)

for the stated int-TRS the *Guard Constants* G and *Guard Constants* g for the loop are:

$$G = \begin{pmatrix} -1 & -1 \\ -1 & 0 \\ -3 & 0 \\ 0 & 2 \end{pmatrix} \text{ and } g = \begin{pmatrix} -4 \\ -7 \\ -21 \\ -6 \end{pmatrix}$$

Reminder: int-TRS

$$f_1 \quad\quad\quad \to f_2(1 + 3 * v_1, -3) :|: v_1 > 2 \ \&\& \ 8 < 3 * v_1$$
$$f_2(v_1, v_2) \to f_2(3 * v_1 + v_2, v_3) :|: v_1 + v_2 > 3 \ \&\&$$
$$v_1 > 6 \ \&\& \ 3 * v_1 > 20 \ \&\& \ 5 + v_3 = 2 * v_2 \ \&\& \ v_3 < -10$$

Example (*Update Matrix*, *Update Constants*)

for the stated int-TRS the *Update Matrix* U and *Update Constants* u are:

$$U = \begin{pmatrix} 3 & 1 \\ 0 & 2 \end{pmatrix} \text{ and } u = \begin{pmatrix} 0 \\ -5 \end{pmatrix}$$

Definition (*Iteration Matrix*, *Iteration Constants*)

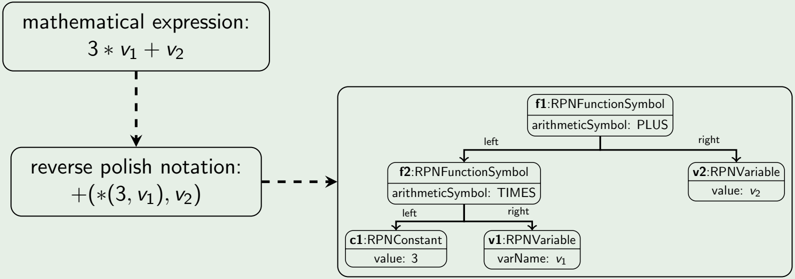
Let $\mathbf{0}$ be a matrix of the size of G with only entry's 0 and I denote the identity matrix having the same dimension as U. Then are the *Iteration Matrix* A and *Iteration Constants* b defined as:

$$A = \begin{pmatrix} G & \mathbf{0} \\ U & -I \\ -U & I \end{pmatrix} \text{ and } b = \begin{pmatrix} g \\ -u \\ u \end{pmatrix}$$

Reverse Polish Notation Tree (RPNTree)

- simple tree structure to handle <u>only</u> considered terms
- classes for variables, constants and arith. operations

Example



Sat. Modulo Theorie (SMT)

- Basic idea:

  set of assertions: (in)-equations with variables

  $\xrightarrow{\textit{SMT-solver}}$ a sat. model or unsat. core
- sat. model: a value for every variable s.t. all assertions hold
- unsat. core: a (minimal) set of assertions that can't hold simultaneously

Example

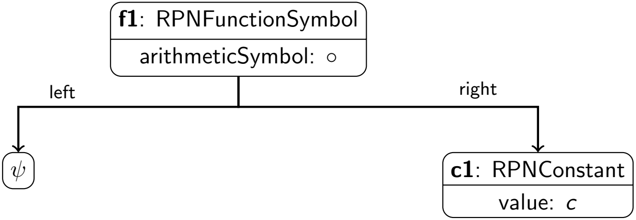
Considering the following assertions:

$$x \leq y \quad x > 5 \quad x + y \leq 20 \quad y \neq 10$$

A possible model would be $m_1 = \{x = 6, y = 6\}$.

changing the third assertion to $x + y \leq 10$:
no possible solution with unsat. core $\{x \leq y,\ x > 5,\ x + y \leq 10\}$

- assertions can be generated using *SMTFactory*
- if generated it ensures the following property:



where $\circ \in \{\leq, =\}$, $cons \in \mathbb{Z}$ and a linear update $\psi = \sum_{i=1}^{n} a_{i,j} v_i$ for variables v_i

Geometric Nontermination

Necessary steps for the derivation of a GNA:

1. derive the *STEM*
2. derive the *Guard Matrix/Constants*
3. derive the *Update Matrix/Constants*
4. compute the *Iteration Matrix/Constants*
5. add the criteria of a GNA as assertions to an *SMT-solver*
6. read of GNA (if exists)

Derivation: *STEM*

Consider two different possibilities:

constant stem: $f_x \to f_y(c_1, \ldots, c_n) : | : TRUE$
$\Rightarrow$ read of values

Example

$$f_1 \to f_2(10, -3) \Rightarrow STEM = (10, -3)^T$$

variable stem: $f_x \to f_y(c_1 + \sum_{i=1}^{n} a_{1,i} v_i, \ldots, c_n + \sum_{i=1}^{n} a_{n,i} v_i) : | : \bigwedge_{\text{guard } g} \sum_{i=1}^{n} g_{n,i} v_i \leq c_m$
$\Rightarrow$ create assertions and derive a model

Example

$$f_1 \to f_2(1 + 3v_1, -3) : | : v_1 > 2 \ \&\& \ 8 < 3v_1$$
$$\Rightarrow \text{model } m_1 = \{v = 3\} \Rightarrow STEM = (10, -3)^T$$

Derivation: Guard Matrix/Constants

conditional term given by the *Symbolic Execution Graph*

$$r = \&\&(g_1, (\&\&(\dots, (\&\&(g_{n-1}, g_n))\dots)))$$

Algorithm 1 derive set of guards

function COMPUTEGUARDSET(Rule r)
    Stack *stack* $\leftarrow r$
    Set *guards*
    while !*stack.isEmpty*() **do**
        item $\leftarrow$ *stack.pop*
        if item is of the form $\&\&(x_1, x_2)$ **then**
            add x_1 and x_2 to *stack*
        else
            add *item* to *guards*
    return *guards*

- now we have $G = \{g \mid g \text{ is a guard}\}$
- Problem: g could not be in the desired $\varphi \leq c$ form.
- Even worse: g could declare new variables using "$=$"
- Solution: bring every g in the desired form, by:

1.	filter equalities	by substituting "new" variables
2.	normalizing ($\leq$)	rewrite $<, >, \geq$ to $\leq$
3.	normalizing (c)	transfer only constant term to r.h.s.

function FILTEREQUALITIES(G)
    $V_{left} = \{v \mid \text{the left hand side of the rule contains } v\}$
    $V_{right} = \{v \mid \text{the right hand side of the rule contains } v\}$
    $V_{sub} = V_{right} - V_{left}$
    define substitution $\theta = \{\}$
    while $V_{sub} \neq \emptyset$ **do**
        select $s \in V_{sub}$
        select $g_s \in \{g \in G \mid g \text{ contains } "="\}$
        remove g_s from G
        rewrite g_s to the form $s = \psi$
        $\theta = \theta\{s/\psi\}$
        for all $g \in G$ **do**
            $g = \theta g$
        remove s from V_{sub}
    return G

Example

From the example int-TRS we get using the decat. algorithm:
$\{v_1 + v_2 > 3,\ v_1 > 6,\ 3 * v_1 > 20,\ 5 + v_3 = 2 * v_2,\ v_3 < -10\}$

1. We compute $V_{left} = \{v_1, v_2\}$, $V_{right} = \{v_1, v_2, v_3\}$ so $V_{sub} = \{v_3\}$
2. Begin with $\theta = \{\}$
3. Since obviously $V_{sub} \neq \emptyset$ we select $s = v_3$ and select $g_s \Leftrightarrow 5 + v_3 = 2 * v_2$
4. g_s rewritten to the form $s = \psi$ then follows with $v_3 = 2 * v_2 - 5$
5. $\theta = \theta\{s/2 * v_2 - 5\} = \{s/2 * v_2 - 5\}$
6. $G = \{v_1 + v_2 > 3,\ v_1 > 6,\ 3 * v_1 > 20,\ 2 * v_2 - 5 < -10\}$
7. Since $V_{sub} = \emptyset$ return G

normalization ($\leq$)

rewrite a guard g_i of the form $g_i \Leftrightarrow \psi + c_\psi \circ c$, where $\circ \in \{<, >, \leq, \geq\}$ to the form $\eta * \psi + \eta * c_\psi \leq \eta * c - \tau$ depending on $\circ$.

$\circ$	η	τ	$\eta * \psi + \eta * c_\psi \leq \eta * c - \tau$
$<$	1	1	$\psi + c_\psi \leq c - 1$
$>$	-1	1	$-\psi - c_\psi \leq -c - 1$
$\leq$	1	0	$\psi + c_\psi \leq c$
$\geq$	-1	0	$-\psi - c_\psi \leq -c$

η is the indicator of inverting the guard to convert $\geq$ ($>$) to $\leq$ ($<$)
τ is the possible subtraction of 1 to receive the $\leq$ instead of a $<$.

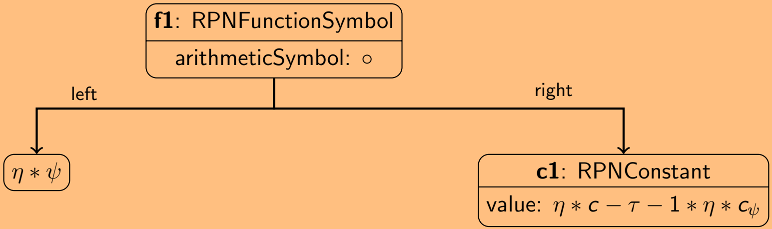
normalization (c)

Subtract the term $\eta * c_\psi$ on both sides:

final form: $\eta * \psi \leq \underbrace{\eta * c - \tau - 1 * \eta * c_\psi}_{\text{constant term}}$

Reminder: int-TRS structure

Can derive constant factors very simple using the stated structure property:



Example

Normalizing the guard $g \Leftrightarrow 3 * v_1 > 20 \Leftrightarrow \underbrace{3 * v_1}_{\psi} + \underbrace{0}_{c_\psi} > \underbrace{20}_{c}$

Looking up the row for $\circ \Leftrightarrow >$:

$\circ$	η	τ	$\eta * \psi + \eta * c_\psi \leq \eta * c - \tau$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$>$	-1	1	$-\psi - c_\psi \leq -c - 1$

Result with $\eta = -1$, $\tau = 1$ in:

$-(3 * v_1) - (0) \leq -20 - 1 \Leftrightarrow -3 * v_1 \leq -21$

- now every guard has the form $\varphi \leq c$
- deriving *Guard Constants* is very simple
- deriving *Guard Matrix* is read off the coefficients.
  (more detailed within the *Update Matrix*)

$\Rightarrow$ *Update Matrix*/*Constants* derived ✓