

Geometrische Nicht-Terminierungs Argumente

Timo Bergerbusch

September 15, 2017

- 1 Einleitung
- 2 Nötiges Vorwissen
 - Integer Transition Systems (ITS)
 - Geometrische Nicht-Terminierungs Argumente (GNA)
 - Definitionen
 - Umgekehrt Polnische Notation (RPNTree)
 - Sat. Modulo Theorie (SMT)
- 3 Geometrische Nicht-Terminierung
 - Herleitung: STEM
 - Herleitung: Guard Matrix & Konstanten
 - Herleitung: Update Matrix & Konstanten
 - Herleitung: Iteration Matrix & Konstanten
 - Herleitung: SMT-Problem
- 4 Verifizierung eines GNA
- 5 Resultate

1 Einleitung

2 Nötiges Vorwissen

- Integer Transition Systems (ITS)
- Geometrische Nicht-Terminierungs Argumente (GNA)
- Definitionen
- Umgekehrt Polnische Notation (RPNTree)
- Sat. Modulo Theorie (SMT)

3 Geometrische Nicht-Terminierung

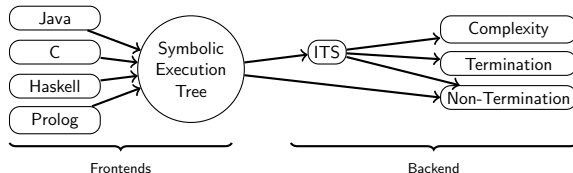
- Herleitung: STEM
- Herleitung: Guard Matrix & Konstanten
- Herleitung: Update Matrix & Konstanten
- Herleitung: Iteration Matrix & Konstanten
- Herleitung: SMT-Problem

4 Verifizierung eines GNA

5 Resultate

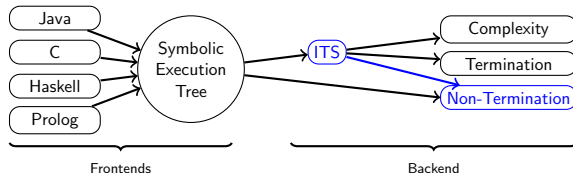
Einleitung und Motivation

- Verbreitung von Software
- Automatische Entwicklungsunterstützung \Rightarrow Halte Problem
- AProVE



Einleitung und Motivation

- Verbreitung von Software
- Automatische Entwicklungsunterstützung \Rightarrow Halte Problem
- AProVE



Beispiel C-Programm

Ein Beispiel für den gesamten Vortrag:

```
1 int main() {  
2  
3     int a;  
4     int b=1;  
5  
6     while (a+b>=4) {  
7         a=3*a+b;  
8         b=2*b-5;  
9     }  
10 }
```

- einfaches C-Programm
- terminiert es?

Beispiel C-Programm

Ein Beispiel für den gesamten Vortrag:

```
1 int main() {  
2  
3     int a;  
4     int b=1;  
5  
6     while (a+b>=4) {  
7         a=3*a+b;  
8         b=2*b-5;  
9     }  
10 }
```

- einfaches C-Programm
- terminiert es?

⇒ **Nein!**

wie kann man das
beweisen?

- 1 Einleitung
- 2 Nötiges Vorwissen
 - Integer Transition Systems (ITS)
 - Geometrische Nicht-Terminierungs Argumente (GNA)
 - Definitionen
 - Umgekehrt Polnische Notation (RPNTree)
 - Sat. Modulo Theorie (SMT)
- 3 Geometrische Nicht-Terminierung
 - Herleitung: STEM
 - Herleitung: Guard Matrix & Konstanten
 - Herleitung: Update Matrix & Konstanten
 - Herleitung: Iteration Matrix & Konstanten
 - Herleitung: SMT-Problem
- 4 Verifizierung eines GNA
- 5 Resultate

Integer Transition Systems (ITS)

ITS betrachtete Programme:

$$\begin{array}{lcl}
 & \text{(1)} & \text{(2)} \\
 1 & \underbrace{f_x} & \rightarrow \underbrace{f_y} (v_1, \dots v_n) : | : \text{cond}_1 \\
 2 & f_y(\underbrace{v_1, \dots v_n}_{(3)}) & \rightarrow f_y(\underbrace{v'_1, \dots v'_n}_{(3)}) : | : \underbrace{\text{cond}_2}_{(4)}
 \end{array}$$

- (1) Funktionssymbol (kein Variablen \Rightarrow Start)
- (2) Funktionssymbol
- (3) Variablen v'_i als linear Updates der Variablen v_j
- (4) eine Menge von (Un-)Gleichungen über v_j

Geometrische Nicht-Terminierungs Argumente (GNA)

- Idee: Teilen des Programms in **zwei** Teile:
 - STEM: Variablen Deklaration und Initialisierung

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

- LOOP: linear Updates und *while*-Bedingung

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

- Anwenden der Definition von Geometrischen Nicht-Terminierungs Argumenten von J. Leike und M. Heizmann

Beispiel

Das ITS zum Beispielprogramm:

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

erste Regel \Leftrightarrow STEM

zweite Regel \Leftrightarrow LOOP

Definition (Geometrische Nicht-Terminierungs Argumente)

Ein Tupel der Form:

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

ist ein Geometrisches Nicht-Terminierungs Argument der Größe k für ein Programm $= (STEM, LOOP)$ mit n Variablen g.d.w. alle folgenden Kriterien erfüllt sind:

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

(init) x repräsentiert den *Startterm* (STEM)

(point) $A \left(x + \sum_i y_i \right) \leq b$

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

Anmerkung: Definiere $y_0 = \mu_0 = 0$ statt einer Fallunterscheidung

Definition (Guard Matrix & Konstanten)

Die Guard Matrix G ist eine Koeffizienten Matrix für die Ungleichungen. Die Guard Konstanten g ein Vektor mit Konstanten der Ungleichungen.

Definition (Guard Matrix & Konstanten)

Die Guard Matrix G ist eine Koeffizienten Matrix für die Ungleichungen. Die Guard Konstanten g ein Vektor mit Konstanten der Ungleichungen.

Definition (Update Matrix & Konstanten)

Die Update Matrix U und Konstanten u sind analog zu der Guard Matrix und Konstanten über den linearen Updates.

Definition (Guard Matrix & Konstanten)

Die Guard Matrix G ist eine Koeffizienten Matrix für die Ungleichungen. Die Guard Konstanten g ein Vektor mit Konstanten der Ungleichungen.

Definition (Update Matrix & Konstanten)

Die Update Matrix U und Konstanten u sind analog zu der Guard Matrix und Konstanten über den linearen Updates.

Definition (Iteration Matrix & Konstanten)

Mit $\mathbf{0}$ eine Matrix aus 0-en und I der Einheitsmatrix ergeben sich die Iteration Matrix A und Konstanten b wie folgt:

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

Erinnerung: ITS

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

Beispiel: Guard Matrix & Konstanten

für das Beispiel ITS ergeben sich die Guard Matrix G und Konstanten g :

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

Erinnerung: ITS

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

Beispiel: Update Matrix & Konstanten

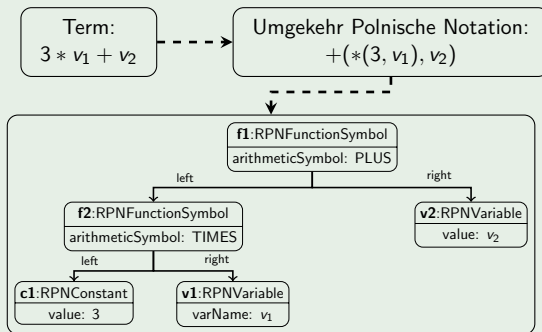
für das Beispiel ITS ergeben sich die Update Matrix U und Konstanten u :

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

Reverse Polish Notation Tree (RPNTree)

- Baumstruktur für ausschließlich berücksichtigter Terme
- Klassen für Konstanten, Variablen und Operatoren

Beispiel



Sat. Modulo Theorie (SMT)

- Grundlegende Idee:
Menge von Regeln: (Un)-Gleichungen mit Variablen
 $\xrightarrow{\text{SMT-solver}}$ ein erf. Modell oder unerf. Kern
- erf. Modell: einen Wert für jede Variable
- unerf. Modell: ein (minimale) Menge von Regeln die nicht alle zeitgleich gelten können

Beispiel

Beispiel mit folgenden Regeln:

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

Mögliches Modell $m_1 = \{x = 6, y = 6\}$.

Ändern der dritten Regel zu $x + y \leq 10$:

Kein Modell ex. mit unerf. Kern $\{x \leq y, x > 5, x + y \leq 10\}$

- 1 Einleitung
- 2 Nötiges Vorwissen
 - Integer Transition Systems (ITS)
 - Geometrische Nicht-Terminierungs Argumente (GNA)
 - Definitionen
 - Umgekehrt Polnische Notation (RPNTree)
 - Sat. Modulo Theorie (SMT)
- 3 **Geometrische Nicht-Terminierung**
 - Herleitung: STEM
 - Herleitung: Guard Matrix & Konstanten
 - Herleitung: Update Matrix & Konstanten
 - Herleitung: Iteration Matrix & Konstanten
 - Herleitung: SMT-Problem
- 4 Verifizierung eines GNA
- 5 Resultate

Herleitung: STEM

unterscheiden von **zwei** Möglichkeiten:

konstanter STEM : $f_x \rightarrow f_y(c_1, \dots, c_n) : | : TRUE$
 \Rightarrow Werte ablesen

Beispiel

$$f_1 \rightarrow f_2(10, -3) \Rightarrow \text{STEM} = (10, -3)^T$$

Herleitung: STEM

unterscheiden von **zwei** Möglichkeiten:

konstanter STEM : $f_x \rightarrow f_y(c_1, \dots, c_n) : | : TRUE$
 \Rightarrow Werte ablesen

Beispiel

$$f_1 \rightarrow f_2(10, -3) \Rightarrow \text{STEM} = (10, -3)^T$$

variabler STEM : $f_x \rightarrow f_y(c_1 + \sum_{i=1}^n a_{1,i}v_i, \dots, c_n + \sum_{i=1}^n a_{n,i}v_i) : | :$
 $\bigwedge_{\text{guard } g} \sum_{i=1}^n g_{n,i}v_i \leq c_m$
 \Rightarrow Bedingungen einem *SMT-solver* geben und
Modell ablesen

Beispiel

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

Herleitung: Guard Matrix & Konstanten

- (Un-)Gleichungen vom *Symbolic Execution Graph* der Form

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

- Aufspalten zu $G = \{g \mid g \text{ ist eine (Un-)Gleichung}\}$

Herleitung: Guard Matrix & Konstanten

- (Un-)Gleichungen vom *Symbolic Execution Graph* der Form
$$r = \&\&(g_1, (\&\&(\dots, (\&\&(g_{n-1}, g_n)) \dots)))$$
- Aufspalten zu $G = \{g \mid g \text{ ist eine (Un-)Gleichung}\}$
- **Problem:** g könnte nicht die Form $\varphi \leq c$ haben

Herleitung: Guard Matrix & Konstanten

- (Un-)Gleichungen vom *Symbolic Execution Graph* der Form
$$r = \&\&(g_1, (\&\&(\dots, (\&\&(g_{n-1}, g_n)) \dots)))$$
- Aufspalten zu $G = \{g \mid g \text{ ist eine (Un-)Gleichung}\}$
- **Problem:** g könnte nicht die Form $\varphi \leq c$ haben
- **Schlimmer:** g könnte mittels "=" neue Variablen einführen

Herleitung: Guard Matrix & Konstanten

- (Un-)Gleichungen vom *Symbolic Execution Graph* der Form

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

- Aufspalten zu $G = \{g \mid g \text{ ist eine (Un-)Gleichung}\}$
- **Problem:** g könnte nicht die Form $\varphi \leq c$ haben
- **Schlimmer:** g könnte mittels "=" neue Variablen einführen
- **Lösung:** umformen von g in die gewünschte Form:
 1. Gleichungen suchen ersetzen "neuer" Variablen
 2. Normalisierung (\leq) umschreiben $<, >, \geq$ to \leq
 3. Normalisierung (c) umschreiben der Konstanten

```
1: function FILTEREQUALITIES( $G$ )
2:    $V_{left} = \{v \mid \text{die linke Seite beinhaltet } v\}$ 
3:    $V_{right} = \{v \mid \text{die rechte Seite beinhaltet } v\}$ 
4:    $V_{sub} = V_{right} - V_{left}$ 
5:   definiere Substitutionen  $\theta = \{\}$ 
6:   while  $V_{sub} \neq \emptyset$  do
7:     wähle  $s \in V_{sub}$ 
8:     wähle  $g_s \in \{g \in G \mid g \text{ beinhaltet " = " und } s\}$ 
9:     entferne  $g_s$  aus  $G$ 
10:    umschreiben von  $g_s$  zur Form  $s = \psi$ 
11:     $\theta = \theta\{s/\psi\}$ 
12:    for all  $g \in G$  do
13:       $g = \theta g$ 
14:    entferne  $s$  aus  $V_{sub}$ 
15:  return  $G$ 
```

Beispiel

Für das Beispiel ITS erhalten wir:

$$\{v_1 + v_2 > 3, v_1 > 6, 3 * v_1 > 20, 5 + v_3 = 2 * v_2, v_3 < -10\}$$

- ① berechne $V_{left} = \{v_1, v_2\}$, $V_{right} = \{v_1, v_2, v_3\}$ so $V_{sub} = \{v_3\}$
- ② starte mit $\theta = \{\}$
- ③ Da $V_{sub} \neq \emptyset$ wähle $s = v_3$ und wähle $g_s \Leftrightarrow 5 + v_3 = 2 * v_2$
- ④ g_s umschreiben zur Form $s = \psi$ führt zu $v_3 = 2 * v_2 - 5$
- 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 Da $V_{sub} = \emptyset$ gebe G zurück

Normalisierung (\leq)

Umschreiben von g_i der Form $g_i \Leftrightarrow \psi + c_\psi \circ c$, mit
 $\circ \in \{<, >, \leq, \geq\}$ zur Form $\eta * \psi + \eta * c_\psi \leq \eta * c - \tau$ abhängig von
 \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 beschreibt die Umwandlung von \geq ($>$) zu \leq ($<$)
 τ steht für möglich Subtraktion um \leq statt $<$ zu erhalten

Normalisierung (c)

Subtrahiere $\eta * c_\psi$ auf beiden Seiten:

endgültige Form: $\eta * \psi \leq \eta * c - \tau - 1 * \eta * c_\psi$

Beispiel

Normalisierung der Ungleichung

$$g \Leftrightarrow 3 * v_1 > 20 \Leftrightarrow \underbrace{3 * v_1}_{\psi} + \underbrace{0}_{c_\psi} > \underbrace{20}_c$$

Nachschlagen des Wertes von $\circ \Leftrightarrow >$:

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

Endergebnis mit $\eta = -1, \tau = 1$ in:

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

- Jede Ungleichung hat die Form $\varphi \leq c$
 - Herleiten der Guard Konstanten ist nun trivial
 - Herleiten der Guard Matrix ist einfaches Ablesen
(genauer unter der Update Matrix)
- ⇒ Guard Matrix & Konstanten hergeleitet ✓

Herleitung: Update Matrix & Konstanten

- beinhalten keine (Un-)Gleichheiten
- haben die Form: $c + \sum_{i=1}^n a_i v_i$ (nicht $v_i a_i$)
- **Problem:** könnte "neue" Variablen aus den Guard beinhalten

Herleitung: Update Matrix & Konstanten

- beinhalten keine (Un-)Gleichheiten
- haben die Form: $c + \sum_{i=1}^n a_i v_i$ (nicht $v_i a_i$)
- **Problem:** könnte "neue" Variablen aus den Guard beinhalten
Lösung: Substitutionen auf die Updates anwenden
- **Problem:** keine Struktureigenschaft wie bei den Guards

Herleitung: Update Matrix & Konstanten

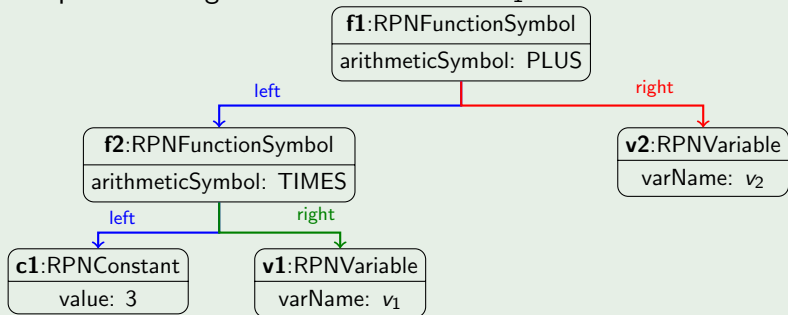
- beinhalten keine (Un-)Gleichheiten
- haben die Form: $c + \sum_{i=1}^n a_i v_i$ (nicht $v_i a_i$)
- **Problem:** könnte "neue" Variablen aus den Guard beinhalten
Lösung: Substitutionen auf die Updates anwenden
- **Problem:** keine Struktureigenschaft wie bei den Guards
Lösung: rekursive Suche mit zwei Eigenschaften:
 - 1 es ex. maximal eine Konstante
 - 2 die Konstante wird nicht multipliziert

Algorithm 1 Derivation of a coefficient

```
1: function GETCOEFFICIENT(query)
2:   if this == query then
3:     return 1
4:   else if beinhaltet die query nicht then
5:     return 0
6:
7:   if this repräsentiert PLUS then
8:     if linke Seite beinhaltet query then
9:       return getCoefficient(query)
10:    else
11:      return getCoefficient(query)
12:   if this repräsentiert TIMES then
13:     if this.right == query then
14:       return this.left.value
```

Beispiel

Beispielherleitung des Koeffizienten von v_1 :



Update Matrix & Konstanten, Iteration Matrix & Konstanten

- Koeffizient für jede Variable und Konstante pro Update
- ⇒ Update Matrix & Konstanten ✓

Update Matrix & Konstanten, Iteration Matrix & Konstanten

- Koeffizient für jede Variable und Konstante pro Update

⇒ Update Matrix & Konstanten ✓

- Iteration Matrix & Konstanten gegeben durch:

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

can be computed

⇒ Iteration Matrix & Konstanten ✓

SMT-Problem

- geg. A und b nutze *SMT-solver* zum Beweisen der (nicht) ex. eines GNA
- in Anlehnung an den Beweis der GNA-Korrektheit:

Satz

λ_i ist der i -te Eigenwert von U .

- **Problem:** $\mu_i * y_i$ ist nicht-linear

SMT-Problem

- geg. A und b nutze *SMT-solver* zum Beweisen der (nicht) ex. eines GNA
- in Anlehnung an den Beweis der GNA-Korrektheit:

Satz

λ_i ist der i -te Eigenwert von U .

- Problem:** $\mu_i * y_i$ ist nicht-linear
zwei mögliche Lösungsansätze:
 - 1 nutze *quantifier-free non-linear integer arithmetic*
 - 2 iterieren über alle μ 's

Erinnerung: Domain Kriterium

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

\Rightarrow Bedingungen für μ 's hinzufügen

Erinnerung: Domain Kriterium

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

⇒ Bedingungen für μ 's hinzufügen

Erinnerung: Initiation Kriterien

(init) x repräsentiert den *Startterm* (STEM)

⇒ keine weiteren Bedingungen

Erinnerung: Point Kriterium

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

- y_i unbekannt \Rightarrow erstelle $s_i = x_i + \sum_{j=1}^n y_{j,i}$
- $A \begin{pmatrix} x \\ s \end{pmatrix} \leq b$

$$\Leftrightarrow \begin{pmatrix} G & 0 & \dots & 0 \\ a_{1,1} & \dots & a_{1,n} & -1 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & \dots & a_{n,n} & 0 & \dots & -1 \\ -a_{1,1} & \dots & -a_{1,n} & 1 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ -a_{n,1} & \dots & -a_{n,n} & 0 & \dots & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_n \\ s_1 \\ \vdots \\ s_n \end{pmatrix} \leq \begin{pmatrix} g \\ -u_1 \\ \vdots \\ -u_n \\ u_1 \\ \vdots \\ u_n \end{pmatrix}$$

$$\begin{pmatrix} a_{1,1} * x_1 & \dots & a_{1,n} * x_n & -1 * s_1 & \dots & 0 * s_n \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ a_{n,1} * x_1 & \dots & a_{n,n} * x_n & 0 * s_1 & \dots & -1 * s_n \\ -a_{1,1} * x_1 & \dots & -a_{1,n} * x_n & 1 * s_1 & \dots & 0 * s_n \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ -a_{n,1} * x_1 & \dots & -a_{n,n} * x_n & 0 * s_1 & \dots & 1 * s_n \end{pmatrix} \leq \begin{pmatrix} -u_1 \\ \vdots \\ -u_n \\ u_1 \\ \vdots \\ u_n \end{pmatrix}$$

⇒ füge Guard Bedingungen für den STEM hinzu, n Bedingungen mit Gleichheit
und eine Bedingung pro s_i

Erinnerung: Ray Kriterium

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

$$i = 1: \Rightarrow \mu_{i-1} y_{i-1} = 0 \Rightarrow A \begin{pmatrix} y_1 \\ \lambda_1 y_1 \end{pmatrix} \leq 0$$

füge Bedingung hinzu mit $y_{1,j}$ $1 \leq j \leq n$ als neue Variablen

$i > 1$: mit λ_i als den i -ten Eigenwert

füge Bedingung hinzu mit $y_{i,j}$ $1 \leq j \leq n$
und μ_i als neue Variablen

\Rightarrow alle nötigen Bedingungen hinzugefügt✓

lasse den *SMT-solver* ein GNA herleiten (wenn eins ex.)

Erinnerung: Hergeleitete Matrix und Werte

$$A = \begin{pmatrix} -1 & -1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ -3 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 3 & 1 & -1 & 0 \\ 0 & 2 & 0 & -1 \\ -3 & -1 & 1 & 0 \\ 0 & -2 & 0 & 1 \end{pmatrix} \quad b = \begin{pmatrix} -4 \\ -7 \\ -21 \\ -6 \\ 0 \\ 5 \\ 0 \\ -5 \end{pmatrix} \quad \begin{array}{l} \lambda_1 = 3, \\ \lambda_2 = 2 \\ x = (10, -3)^T \end{array}$$

Beispiel: Bedingungen I: Point Crit.

- Guards: $-10 - (-3) \leq -4$
- Gleichheitsbedingung: $30 - 3 - s_1 = 0$
- Summenbedingungen: $s_1 = y_{1,1} + y_{2,1}$ und $s_2 = y_{1,2} + y_{2,2}$

Erinnerung: Hergeleitete Matrix und Werte

$$A = \begin{pmatrix} -1 & -1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ -3 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 3 & 1 & -1 & 0 \\ 0 & 2 & 0 & -1 \\ -3 & -1 & 1 & 0 \\ 0 & -2 & 0 & 1 \end{pmatrix} \quad b = \begin{pmatrix} -4 \\ -7 \\ -21 \\ -6 \\ 0 \\ 5 \\ 0 \\ -5 \end{pmatrix} \quad \begin{matrix} \lambda_1 = 3, \\ \lambda_2 = 2 \\ x = (10, -3)^T \end{matrix}$$

Beispiel: Bedingungen II: Ray Crit.

$i=1$: Beispielbedingung: $-y_{1,1} - y_{1,2} \leq 4$,

$$3 * y_{1,1} + y_{1,2} - 3 * y_{1,1} \leq 0$$

$i>1$: Beispielbedingung: $-y_{2,1} - y_{2,2} \leq 4$,

$$3 * y_{2,1} + y_{2,2} - 1 * (2 * y_{2,1} + \mu * y_{1,1}) \leq 0$$

- 1 Einleitung
- 2 Nötiges Vorwissen
 - Integer Transition Systems (ITS)
 - Geometrische Nicht-Terminierungs Argumente (GNA)
 - Definitionen
 - Umgekehrt Polnische Notation (RPNTree)
 - Sat. Modulo Theorie (SMT)
- 3 Geometrische Nicht-Terminierung
 - Herleitung: STEM
 - Herleitung: Guard Matrix & Konstanten
 - Herleitung: Update Matrix & Konstanten
 - Herleitung: Iteration Matrix & Konstanten
 - Herleitung: SMT-Problem
- 4 Verifizierung eines GNA
- 5 Resultate

Verifizierung eines GNA

- bekommen ein GNA vom *SMT-solver*
 - wollen testen ob dieses wirklich korrekt ist
- ⇒ Nachberechnen des GNA mit geg. Matrizen und Werten

Beispiel: Verifizierung eines GNA I

Vom *SMT-solver*: $y_1 = \begin{pmatrix} 9 \\ 0 \end{pmatrix}$, $y_2 = \begin{pmatrix} 8 \\ -8 \end{pmatrix}$, $\mu_1 = 0$

(domain) offensichtlich wahr ✓

(init) gegenrechnen STEM ✓

$$\text{(point)} \quad A \begin{pmatrix} 10 \\ -3 \\ 10 + 9 + 8 \\ -3 + 0 + (-8) \end{pmatrix} \leq b \Leftrightarrow A \begin{pmatrix} 10 \\ -3 \\ 27 \\ -11 \end{pmatrix} \leq b \quad \checkmark$$

Beispiel: Verifizierung eines GNA II

Vom *SMT-solver*: $y_1 = \begin{pmatrix} 9 \\ 0 \end{pmatrix}$, $y_2 = \begin{pmatrix} 8 \\ -8 \end{pmatrix}$, $\mu_1 = 0$

(ray)

$$i = 1: A \begin{pmatrix} 9 \\ 0 \\ 3 * 9 \\ 3 * 0 \end{pmatrix} \leq 0 \Leftrightarrow A \begin{pmatrix} 9 \\ 0 \\ 27 \\ 0 \end{pmatrix} \leq 0 \checkmark$$

$$i > 1: A \begin{pmatrix} 8 \\ -8 \\ 2 * 8 + 0 * 9 \\ 2 * (-8) + 0 * 0 \end{pmatrix} \leq 0 \Leftrightarrow A \begin{pmatrix} 8 \\ -8 \\ 16 \\ -16 \end{pmatrix} \leq 0 \checkmark$$

\Rightarrow das GNA erfüllt alle Kriterien \Rightarrow Nicht-Terminierung ist bewiesen

1 Einleitung

2 Nötiges Vorwissen

- Integer Transition Systems (ITS)
- Geometrische Nicht-Terminierungs Argumente (GNA)
- Definitionen
- Umgekehrt Polnische Notation (RPNTree)
- Sat. Modulo Theorie (SMT)

3 Geometrische Nicht-Terminierung

- Herleitung: STEM
- Herleitung: Guard Matrix & Konstanten
- Herleitung: Update Matrix & Konstanten
- Herleitung: Iteration Matrix & Konstanten
- Herleitung: SMT-Problem

4 Verifizierung eines GNA

5 Resultate

Resultate

Wie gut/schlecht ist das Programm?
 Benchmark auf den SV-COMP Programmen

SV-COMP		Vgl. zu AProVE ohne GNAs	
Anzahl Programme	847	Terminierung	0.0078 sec
Anwendbare Programme	53	Nicht-Terminierung	-1.2675 sec
false pos./neg.	0	Insgesamt	-0.76 sec

Resultate

Wie gut/schlecht ist das Programm?
 Benchmark auf den SV-COMP Programmen

SV-COMP		Vgl. zu AProVE ohne GNAs	
Anzahl Programme	847	Terminierung	0.0078 sec
Anwendbare Programme	53	Nicht-Terminierung	-1.2675 sec
false pos./neg.	0	Insgesamt	-0.76 sec

Durchschn. Berechnungszeit:

AProVE <u>ohne</u> GNAs	4.513
AProVE mit <u>lediglich</u> GNAs	3.753

aber:

```
1  int main(void){
2      int a, b, c;
3
4      while (a+b+c >= 4) {
5          a = b;
6          b = a+b;
7          c = b-1;
8      }
9      return 0;
10 }
```

aber:

```
1  int main(void){  
2      int a, b, c;  
3  
4      while (a+b+c >= 4) {  
5          a = b;  
6          b = a+b;  
7          c = b-1;  
8      }  
9      return 0;  
10 }
```

AProVE ohne GNAs \Rightarrow MAYBE \Rightarrow echte Verbesserung
AProVE mit GNAs \Rightarrow NO