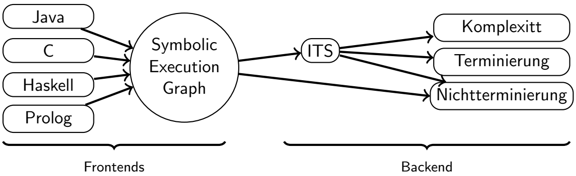
Geometrische Nichtterminierungs-Argumente

Timo Bergerbusch

18. September 2017

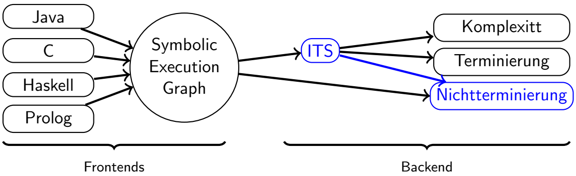
Einleitung und Motivation

- 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:

```c
int main(){

  int a;
  int b=1;

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

- einfaches C-Programm
- terminiert es?

Beispiel C-Programm

Ein Beispiel für den gesamten Vortrag:

```c
int main(){

  int a;
  int b=1;

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

- einfaches C-Programm
- terminiert es?

  ⇒ Nein!

  wie kann man das beweisen?

Integer Transition Systems (ITS)

Hier betrachtete Programme:

$$
\begin{array}{lcl}
\overbrace{f_x}^{(1)} & \rightarrow & \overbrace{f_y}^{(2)}\,(v_1, \ldots v_n) : | : cond_1 \\
f_y\underbrace{(v_1, \ldots v_n)}_{(3)} & \rightarrow & f_y\,\underbrace{(v'_1, \ldots v'_n)}_{(3)} : | : \underbrace{cond_2}_{(4)}
\end{array}
$$

(1) Startsymbol (keine Variablen)

(2) Funktionssymbol

(3) Variablen v'_i als lineare Updates der Variablen v_j

(4) eine Menge von (Un-)Gleichungen über v_j

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

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

  - LOOP: *while*-Bedingung und lineare Updates

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

- suche eines GNAs (J. Leike und M. Heizmann)

Erinnerung: ITS

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

$b' = 2 * 1 - 5 = -3$

hier: $b = 1$!

Beispiel

Das ITS zum Beispielprogramm:

$f_1 \to f_2(1 + 3 \cdot a, -3) : | : a > 2 \;\&\&\; 8 < 3 \cdot a$

$f_2(a, b) \to f_2(3 \cdot a + b, z) : | : a + b > 3 \;\&\&\; a > 6 \;\&\&\; 3 \cdot a > 20 \;\&\&\; 5 + z = 2 \cdot b \;\&\&\; z < -10$

Erinnerung: ITS

$$
\begin{aligned}
&f_1 \rightarrow f_2(1 + 3 \cdot a, -3) : | : a > 2 \;\&\&\; 8 < 3 \cdot a \\
&f_2(a, b) \rightarrow f_2(3 \cdot a + b, z) \;\; : | : a + b > 3 \;\&\&\; a > 6 \;\&\& \\
&\qquad 3 \cdot a > 20 \;\&\&\; 5 + z = 2 \cdot b \;\&\&\; z < -10
\end{aligned}
$$

Guard-Matrix & Konstanten

Vorher: Normalisieren

$$a + b > 3 \Leftrightarrow -a - b < -3 \Leftrightarrow -a - b \leq -4$$

Erinnerung: ITS

$$
\begin{aligned}
&f_1 \rightarrow f_2(1+3\cdot a, -3) : | : a > 2 \ \&\& \ 8 < 3\cdot a \\
&f_2(a,b) \rightarrow f_2(3\cdot a + b, z) : | : a + b > 3 \ \&\& \ a > 6 \ \&\& \\
&\qquad 3\cdot a > 20 \ \&\& \ 5 + z = 2\cdot b \ \&\& \ z < -10
\end{aligned}
$$

Guard-Matrix & Konstanten

Vorher: Normalisieren

$$a + b > 3 \Leftrightarrow -a - b < -3 \Leftrightarrow -a - b \leq -4$$

$\Rightarrow$ Guard-Matrix G und Konstanten g:

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

Erinnerung: ITS

$$f_1 \quad \rightarrow f_2(1 + 3 \cdot a, -3) : | : a > 2 \;\; \&\& \;\; 8 < 3 \cdot a$$

$$f_2(a, b) \rightarrow f_2(3 \cdot a + b, z) \quad : | : a + b > 3 \;\; \&\& \;\; a > 6 \;\; \&\& \;\; 3 \cdot a > 20 \;\; \&\& \;\; 5 + z = 2 \cdot b \;\; \&\& \;\; z < -10$$

Update-Matrix & Konstanten

Vorher: Ersetzen

$$5 + z = 2 \cdot b \Leftrightarrow z = 2 \cdot b - 5$$

$\Rightarrow$ Update-Matrix U und Konstanten u:

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

Definition (Iteration-Matrix & Konstanten)

$\Rightarrow$ Iteration-Matrix A und Konstanten b wie folgt:

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

Beispiel: Iteration-Matrix & Konstanten

$$A = \begin{pmatrix} -1 & -1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ -3 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 3 & 1 & -1 & 0 \\ 0 & 2 & 0 & -1 \\ -3 & -1 & 1 & 0 \\ 0 & -2 & 0 & 1 \end{pmatrix} \quad b = \begin{pmatrix} -4 \\ -7 \\ -21 \\ -6 \\ 0 \\ 5 \\ 0 \\ -5 \end{pmatrix}$$

Definition (Geometrische Nicht-Terminierungs Argumente)

Ein Tupel der Form:

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

ist ein GNA der Größe k mit n Variablen g.d.w.:

(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 repräsentiert den 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 $1 \leq i \leq k$

Anmerkung:

- Definiere $y_0 = \mu_0 = 0$
- λ_i ist der i-te Eigenwert von U.

SMT-Solving

- Grundlegende Idee:

  Menge von Regeln: (Un)-Gleichungen mit Variablen

  $\xrightarrow{\text{SMT solver}}$ Modell <u>oder</u> unerfüllbarer Kern
- Modell: einen Wert für jede Variable
- unerfüllbarer Kern: eine (minimale) unerfüllbare Menge von Bedingungen

Beispiel

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

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

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

unerfüllbarer Kern $\{x > 5, x \leq y, x + y \leq 10\}$

Herleitung: STEM

Unterscheiden von zwei Möglichkeiten:

konstanter STEM : $f_1 \rightarrow f_2(10, -3) : | : \mathit{TRUE}$

$\Rightarrow$ Werte ablesen:

$$\text{STEM} = \begin{pmatrix} 10 \\ -3 \end{pmatrix}$$

Herleitung: STEM

Unterscheiden von zwei Möglichkeiten:

konstanter STEM : $f_1 \to f_2(10, -3) : | : \mathit{TRUE}$

$\Rightarrow$ Werte ablesen:

$$\text{STEM} = \begin{pmatrix} 10 \\ -3 \end{pmatrix}$$

variabler STEM : $f_1 \to f_2(1 + 3a, -3) : | : a > 2 \;\&\&\; 8 < 3a$

$\Rightarrow$ Bedingungen einem SMT solver geben

$\Rightarrow$ Modell $m_1 = \{a = 3\}$

$$\Rightarrow \text{STEM} = \begin{pmatrix} 1 + 3 \cdot 3 \\ -3 \end{pmatrix} = \begin{pmatrix} 10 \\ -3 \end{pmatrix}$$

Herleitung: Guard-Matrix & Konstanten

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

Herleitung: Guard-Matrix & Konstanten

- Bedingungen: $G = \{g \mid g \text{ ist eine (Un-)Gleichung}\}$
- Problem: g muss nicht die Form $a_1 v_1 + \cdots + a_n v_n \leq c$ haben

Herleitung: Guard-Matrix & Konstanten

- Bedingungen: $G = \{g \mid g \text{ ist eine (Un-)Gleichung}\}$
- Problem: g muss nicht die Form $a_1 v_1 + \cdots + a_n v_n \leq c$ haben
- Schlimmer: g könnte mittels „=“ neue Variablen einführen

Herleitung: Guard-Matrix & Konstanten

- Bedingungen: $G = \{g \mid g$ ist eine (Un-)Gleichung$\}$
- Problem: g muss nicht die Form $a_1 v_1 + \cdots + a_n v_n \leq c$ 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 $<, >, \geq$ zu $\leq$
3.	Normalisierung (c)	umschreiben der Konstanten

1. filtern welche Variable nicht vorkommen soll
2. suchen einer Gleichung mit dieser Variable
3. umformen und substituieren

Beispiel

Geg. Bedingungen:
$\{a + b > 3,\ a > 6,\ 3 \cdot a > 20,\ 5 + z = 2 \cdot b,\ z < -10\}$

1. $z \in V_{sub} = V_{links} - V_{rechts}$
2. $5 + z = 2 \cdot b \Leftrightarrow z = 2 \cdot b - 5$
3. neue Bedingungen
   $\{a + b > 3,\ a > 6,\ 3 \cdot a > 20,\ 2 \cdot b - 5 < -10\}$

Zum Schluss:

schreibe alle übrigen Gleichungen als Ungleichungen

1. Normalisierung($\leq$)
   - bei $\geq / >$: Multipliziere mit -1 zu $\leq$ / $<$
   - bei $<$: Subtrahiere 1 auf der rechten Seite und nutze $\leq$
2. Normalisierung(c)
   - Subtrahiere linken konstanten Term

Beispiel

Für $g \Leftrightarrow 3a + 5 > 20$:

Schritt 1:	Multiplizieren mit -1	$\Rightarrow -3a - 5 < -20$
Schritt 2:	Subtrahieren von 1	$\Rightarrow -3a - 5 \leq -21$
Schritt 3:	Subtrahiere konstanten Term	$\Rightarrow -3a \leq -16$

- Jede Ungleichung hat die Form $a_1 v_1 + \cdots + a_n v_n \leq c$
- Herleiten der Guard-Konstanten ist nun trivial
- Herleiten der Guard-Matrix ist einfaches Ablesen
  (genauer unter der Update-Matrix)

$\Rightarrow$ Guard-Matrix & Konstanten hergeleitet ✓

Herleitung: Update-Matrix & Konstanten

- beinhalten <u>keine</u> (Un-)Gleichheiten
- haben die Form: $a_1 v_1 + \cdots + a_n v_n + c$ (<u>nicht</u> $v_i a_i$)
- Problem: könnte „neue“ Variablen aus den Guards beinhalten

Herleitung: Update-Matrix & Konstanten

- beinhalten <u>keine</u> (Un-)Gleichheiten
- haben die Form: $a_1 v_1 + \cdots + a_n v_n + c$ (<u>nicht</u> $v_i a_i$)
- Problem: könnte „neue“ Variablen aus den Guards beinhalten

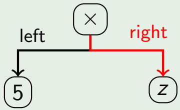
  Lösung: Substitutionen auf die Updates anwenden
- Problem: keine Struktureigenschaft wie bei den Guards

Herleitung: Update-Matrix & Konstanten

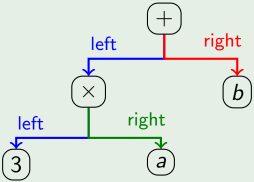
- beinhalten keine (Un-)Gleichheiten
- haben die Form: $a_1 v_1 + \cdots + a_n v_n + c$ (nicht $v_i a_i$)
- Problem: könnte „neue" Variablen aus den Guards beinhalten
  Lösung: Substitutionen auf die Updates anwenden
- Problem: keine Struktureigenschaft wie bei den Guards
  Lösung: rekursive Suche mit zwei Eigenschaften:
  1. es existiert maximal eine Konstante
  2. die Konstante wird nicht multipliziert

Beispiel: query=a, Result $= 1$

Beispiel: query=a, Result=0



Beispiel: query=a, Result=3



Iteration-Matrix & Konstanten

- Koeffizient für jede Variable und Konstante pro Update

$\Rightarrow$ Update-Matrix & Konstanten ✓

Iteration-Matrix & Konstanten

- Koeffizient für jede Variable und Konstante pro Update

$\Rightarrow$ Update-Matrix & Konstanten ✓

- Iteration-Matrix & Konstanten gegeben durch:

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

können berechnet werden

$\Rightarrow$ Iteration-Matrix & Konstanten ✓

SMT-Problem

- Geg. Iteration-Matrix A und Iteration-Konstante b
- nutze SMT-Solver zum Beweis der (Nicht-)Existenz
- Problem: Teile des Ray-Kriterium nicht-linear

SMT-Problem

- Geg. Iteration-Matrix A und Iteration-Konstante b
- nutze SMT-Solver zum Beweis der (Nicht-)Existenz
- Problem: Teile des Ray-Kriterium nicht-linear

  zwei mögliche Lösungsansätze:

  1. nutze *quantifier-free non-linear integer arithmetic*
  2. iterieren über unbekannte

Erinnerung:

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

$\Rightarrow$ Bedingungen für λ_i und μ_i hinzufügen

Erinnerung:

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

$\Rightarrow$ Bedingungen für λ_i und μ_i hinzufügen

Erinnerung: Initiation Kriterium

(init) x repräsentiert den STEM

$\Rightarrow$ keine weiteren Bedingungen

Erinnerung:

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

$$\begin{pmatrix} G & \mathbf{0} \\ U & -I \\ -U & I \end{pmatrix} \begin{pmatrix} x \\ s \end{pmatrix} \leq \begin{pmatrix} g \\ -u \\ u \end{pmatrix}$$

Erinnerung:

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

$$\begin{pmatrix} G & \mathbf{0} \\ U & -I \\ -U & I \end{pmatrix} \begin{pmatrix} x \\ s \end{pmatrix} \leq \begin{pmatrix} g \\ -u \\ u \end{pmatrix}$$

$\Rightarrow$ prüfe Guard-Bedingungen für den STEM

Erinnerung:

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

$$\begin{pmatrix} G & \mathbf{0} \\ U & -I \\ -U & I \end{pmatrix} \begin{pmatrix} x \\ s \end{pmatrix} \leq \begin{pmatrix} g \\ -u \\ u \end{pmatrix}$$

$\Rightarrow$ prüfe Guard-Bedingungen für den STEM

eine Bedingung pro s_i und

n Bedingungen mit Gleichheit

Erinnerung: Hergeleitete Matrix und Werte

$$\begin{pmatrix} -1 & -1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ -3 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 3 & 1 & -1 & 0 \\ 0 & 2 & 0 & -1 \\ -3 & -1 & 1 & 0 \\ 0 & -2 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_{1,1} \\ x_{1,2} \\ s_1 \\ s_2 \end{pmatrix} \leq \begin{pmatrix} -4 \\ -7 \\ -21 \\ -6 \\ 0 \\ 5 \\ 0 \\ -5 \end{pmatrix}$$

Beispiel: Point Kriterium ($x = (10, -3)^T$)

- Guards: $-10 - (-3) \leq -4$
- Gleichheitsbedingung: $30 - 3 - s_1 = 0$
- Summenbedingungen: $s_1 = x_1 + y_{1,1} + y_{2,1}$ und $s_2 = x_2 + y_{1,2} + y_{2,2}$

Erinnerung: Ray Kriterium

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

Füge Bedingungen hinzu:

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

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

$\Rightarrow$ alle nötigen Bedingungen hinzugefügt✓

lasse den SMT solver ein GNA herleiten

Erinnerung: Hergeleitete Matrix und Werte

$$\begin{pmatrix} -1 & -1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ -3 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 3 & 1 & -1 & 0 \\ 0 & 2 & 0 & -1 \\ -3 & -1 & 1 & 0 \\ 0 & -2 & 0 & 1 \end{pmatrix} \begin{pmatrix} y_{i,1} \\ y_{i,2} \\ \lambda_i y_{i,1} + \mu_{i-1} y_{i-1,1} \\ \lambda_i y_{i,2} + \mu_{i-1} y_{i-1,2} \end{pmatrix} \leq \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Beispiel: Ray Kriterium ($\lambda_1 = 3, \lambda_2 = 2$)

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

$3 \cdot y_{1,1} + y_{1,2} - 3 \cdot y_{1,1} = 0$

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

$3 \cdot y_{2,1} + y_{2,2} - 1 \cdot (2 \cdot y_{2,1} + \mu \cdot y_{1,1}) = 0$

Beispiel eines GNAs

Beispiel 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) neu berechnen des STEM (✓)

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

Beispiel 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)

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

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

$\Rightarrow$ das GNA erfüllt alle Kriterien

$\Rightarrow$ Nicht-Terminierung ist bewiesen

Resultate

Benchmark auf den 847 SV-COMP Programmen

SV-COMP: nur GNAs

Terminierung (21)	X sek
Nicht-Terminierung (32)	X sek
false pos./neg.	0
Insgesamt	X sek

```c
int main(void){
  int a, b, c;

  while (a+b+c >= 4) {
    a = b;
    b = a+b;
    c = b-1;
  }
  return 0;
}
```

```c
int main(void){
  int a, b, c;

  while (a+b+c >= 4) {
    a = b;
    b = a+b;
    c = b-1;
  }
  return 0;
}
```

AProVE ohne GNAs $\Rightarrow$ MAYBE
AProVE mit GNAs $\Rightarrow$ NO

$\Rightarrow$ echte Verbesserung

Zusammenfassung

- Prinzip der Programmaufteilung (STEM, LOOP)
- Matrixdarstellungen der Programmbestandteile
- SMT-Solver als Tool eingesetzt
- GNAs als Beweise der Nichtterminierung hergeleitet

Danke für die Aufmerksamkeit