Program Verification by Reduction to Semi-Algebraic System Solving

Naijun ZHAN
Lab. of Comp. Sci., Institute of Software, CAS, China
znj@ios.ac.cn

(With Yinghua Chen, Bican Xia, Lu Yang and Chaochen Zhou)

Outline

- Motivation
- Theories and Tools on Semi-Algebraic Systems
- Semi-Algebraic Transition Systems
- Polynomial Invariant Generation
- Polynomial Ranking Function Discovering
- Computing Reachable Set of Linear Hybrid Systems
- Complexity Analysis
- Conclusion

MOTIVATION

Invariant Generation and Termination Analysis

- ➜ **Termination analysis** and **invariant generation** play a central role in program verification, also are thought as the most challenging parts of program verification.

Reachability Computation

- ➜ **How to design correct embedded systems is a big challenge for computer scientists and control theorists.**
- ➜ From a computer scientist's point of view, that is how to guarantee the **correctness** of embedded software.
- ➜ Verification of embedded systems can be reduced to **reachability computation**.

Related Work on Invariant Generation

➜ **Limited success in the past attempts**

German and Wegbreit [IEEE TSE 1(1)], Karr [Acta Inf. 6], Katz and Manna [CACM 19(4)]

➜ **Based on Abstract Interpretation**

- Incomplete
- Possibly producing weak invariants
- Cousot [VMCAI05], Cousot and Halbwachs [POPL78]

➜ **Based on the Technique of Linear Algebra**

- Polynomials of bounded degree as invariants of programs with affine assignments
- M. Müller-Olm and H. Seidl, [SAS02]

➜ **Based on Ideal Theory**

- Completeness
- invariants represented as a conjunction of polynomial equations
- Rodriguez-Carbonell and Kapur [JSC 42, SCP64(1)]

➜ **Based on Gröbner Bases**

- Completeness
- invariants represented as a conjunction of polynomial equations
- Rodriguez-Carbonell and Kapur [ISSAC04], Sankaranarayanan, Sipma and Manna [POPL04]

➜ **Based on First-Order Quantifier Elimination**

- More expressive invariants, but high complexity,
- Kapur [ACA04]

Related Work on Termination Analysis

- ➜ **Kats & Manna 1975** [CACM 19(4)]
  Generate and solve constraint systems for linear ranking functions, over linear loop with affine assignments
- ➜ **Colón & Sipma 2001, 2002** [TACAS01,CAV02]
  Synthesis of linear ranking functions for linear loops.
- ➜ **Podelski & Rybalchenko 2005** [VMCAI05]
  Complete method for linear ranking functions over linear loops with one transition and without an initial condition.
- ➜ **Bradley, Manna & Sipma 2005** [CAV05,CONCUR05]
  Complete method for lexicographic linear ranking functions over linear loops.
- ➜ **Bradley, Manna & Sipma 2005** [VMCAI05]
  Incomplete but efficient method for synthesis of polynomial ranking functions over polynomial loops.
- ➜ **Cousot 2005** [VMCAI05]
  Incomplete but efficient method for synthesis of polynomial ranking functions over polynomial loops.

- **Bradley, Manna & Sipma 2005** [ICALP05]
  Complete and efficient method for synthesis of lexicographic linear polyranking functions over linear loops.
- **Gupta, Henzinger, Majundar, Rybalchenko & Xu 2008** [POPL08]
  A method to search counterexamples to termination that are infinite program executions.

Related Work on Decidability of State Reachability

- **Timed Automaton** Alur and Dill, [TCS 126]
- **Multi-rate Automata** Alur et al, [TCS 138]
- **Rectangular Hybrid Automata** Henzinger et al, [JCSS 57(1)]
- **O-Minimal Hybrid Systems** Lafferriere, Pappas and Sastry [UCB/ERL M98/29]

Summary

- ➜ Applying the theories and tools of computer algebra to program verification has made great success
- ➜ High complexity of computer algebra algorithms forms the bottleneck of such approaches

Challenges in Program Verification through Computer Algebra

Challenging Problem: To invent more powerful approaches with low complexity to program verification is still a challenging problem.

Overview

- **Goal:** Applying techniques on solving semi-algebraic systems to program verification, in particular, to automatic synthesis of more expressive invariants and ranking functions, and computing reachable set of linear hybrid systems
  - Non-linear ranking functions
  - Invariants represented as semi-algebraic systems
  - Over polynomial programs and linear hybrid systems
- **Our Solution:**
  - Reduce these problems to *semi-algebraic systems* solving
  - Then utilize our theories and tools on semi-algebraic systems to solve the resulted problems
- **Our Contributions:**
  - A complete and efficient method for invariant synthesis, which can be used to generate more expressive invariants
  - A similar method for non-linear ranking function discovering
  - Improvement of the efficiency of reachability computation of linear hybrid systems

THEORIES AND TOOLS ON SEMI-ALGEBRAIC SYSTEMS (SASS)

First-Order Quantifier Elimination

➜ **E.g.** $\exists x. ax^2 + bx + c = 0$ **iff**
$(a \neq 0 \wedge b^2 - 4ac \geq 0) \vee (a = 0 \wedge b \neq 0) \vee (a = b = c = 0)$

➜ **Tarski's Algorithm** ([Tarski51])

- To eliminate quantifications of first-order formula of polynomials
- Decidability of elementary algebra and geometry
- Complexity is non-elementary

➜ **Collins's Algorithm** ([AT&FL, LNCS 33])

- ***Cylindrical Algebra Decomposition***
- The complexity is double exponential
- **QEPCAD**

➜ **Combined First-Order Quantifiers Eli. (Weispfenning & Sturm,[..])**

- Tarski's Algebra + Presburger Arithmetics + QBF + $\cdots$
- **REDLOG**

➜ **Chinese School Led by Prof. Wu (Herbrand Award Winner) Contributed Very Much on Solving Polynomial Systems**

- ***Wu Method*** ([JSSM 4])
- ***Complete Discrimination Systems*** ([Sci in CN E(39),JSC 28])

SASs

- Let $u = (u_1, \cdots, u_t), x = (x_1, \cdots, x_s)$
- A **semi-algebraic system** is of the following form:

$$\begin{cases} p_1(\mathbf{u}, \mathbf{x}) = 0, ..., p_r(\mathbf{u}, \mathbf{x}) = 0, \\ g_1(\mathbf{u}, \mathbf{x}) \geq 0, ..., g_k(\mathbf{u}, \mathbf{x}) \geq 0, \\ g_{k+1}(\mathbf{u}, \mathbf{x}) > 0, ..., g_l(\mathbf{u}, \mathbf{x}) > 0, \\ h_1(\mathbf{u}, \mathbf{x}) \neq 0, ..., h_m(\mathbf{u}, \mathbf{x}) \neq 0, \end{cases} \tag{1}$$

- An SAS of the form (1) is usually denoted by $[\mathbb{P}, \mathbb{G}_1, \mathbb{G}_2, \mathbb{H}]$, where $\mathbb{P} = [p_1, ..., p_r], \mathbb{G}_1 = [g_1, ..., g_k], \mathbb{G}_2 = [g_{k+1}, ..., g_l]$ and $\mathbb{H} = [h_1, ..., h_m]$;
- An SAS is called *parametric* if $t \neq 0$, written PSAS, otherwise *constant*, written CSAS.

Main Concerns on SASs

- For CSASs, **real root isolation**
- For PSASs, **real solution classification**

Real Root Classification of PSASs

(Lu Yang et al, [Sci. in CN F(44), ASCM 2005])

Step 1, Triangularizing a PSAS with Ritt-Wu Method

➜ **Triangular set**

$$\begin{array}{l} T_1 = T_1(\mathbf{v}, y_1), \\ T_2 = T_2(\mathbf{v}, y_1, y_2), \\ \cdots\cdots \\ T_k = T_k(\mathbf{v}, y_1, \cdots, y_k), \end{array} \tag{2}$$

➜ **Triangular system**

$$\left\{ \begin{array}{l} f_1(\mathbf{u}, x_1) = 0, \\ \quad\cdots \\ f_s(\mathbf{u}, x_1, ..., x_s) = 0, \\ \mathbb{G}_1,\ \mathbb{G}_2,\ \mathbb{H}. \end{array} \right. \tag{3}$$

➜ With **Ritt-Wu Method**, decomposing the equations of an SAS S of (1) into triangular sets $\mathcal{T} = \{\mathbb{T}_\mathbf{1}, ..., \mathbb{T}_\mathbf{e}\}$

➜ The correspondence $\mathrm{Zero}(\mathbb{P}) = \bigcup_{i=1}^{e} \mathrm{Zero}(\mathbb{T}_i/J_i)$

Step 1, Triangularizing a PSAS with Ritt-Wu Method (Cont'd)

Example 1

Consider an SAS $S : [\mathbb{P}, \mathbb{G}_1, \mathbb{G}_2, \mathbb{H}]$ with $\mathbb{P} = [p_1, p_2, p_3]$, $\mathbb{G}_1 = \emptyset, \mathbb{G}_2 = [x, y, z, b, 2 - b], \mathbb{H} = \emptyset$, where

$$p_1 = x^2 + y^2 - z^2, \quad p_2 = (1 - x)^2 - z^2 + 1, \quad p_3 = (1 - y)^2 - b^2 z^2 + 1.$$

The equations $\mathbb{P}$ can be decomposed into two triangular sets

$$\mathbb{T}_1 : [b^4 x^2 - 2b^2(b^2 - 2)x + 2b^4 - 8b^2 + 4, -b^2 y + b^2 x + 2 - 2b^2, b^4 z^2 + 4b^2 x - 8b^2 + 4],$$
$$\mathbb{T}_2 : [x^2 - 2x + 2, y + x - 2, z],$$

with the relation

$$\mathrm{Zero}(\mathbb{P}) = \mathrm{Zero}(\mathbb{T}_1/b) \bigcup \mathrm{Zero}(\mathbb{T}_2)$$

Step 2, Computing Border Polynomial

➜
$$
\begin{aligned}
F &= a_0x^m + a_1x^{m-1} + \cdots + a_{m-1}x + a_m,\\
G &= b_0x^l + b_1x^{l-1} + \cdots + b_{l-1}x + b_l.
\end{aligned}
$$

The following $(m+l)\times(m+l)$ matrix (those entries except a_i, b_j are all zero)

$$
\begin{array}{c}
\left(\begin{array}{cccccccc}
a_0 & a_1 & \cdots & a_m & & & & \\
 & a_0 & a_1 & \cdots & a_m & & & \\
 & & \ddots & \ddots & & \ddots & & \\
 & & & a_0 & a_1 & \cdots & a_m & \\
b_0 & b_1 & \cdots & b_l & & & & \\
 & b_0 & b_1 & \cdots & b_l & & & \\
 & & \ddots & \ddots & & \ddots & & \\
 & & & b_0 & b_1 & \cdots & b_l &
\end{array}\right)
\begin{array}{l}
\left.\vphantom{\begin{array}{c}a\\a\\a\\a\end{array}}\right\} l\\
\left.\vphantom{\begin{array}{c}a\\a\\a\\a\end{array}}\right\} m
\end{array}
\end{array},
$$

is called the *Sylvester matrix* of F and G w.r.t. x. The determinant of the matrix is called the *Sylvester resultant* or *resultant* of F and G w.r.t. x and is denoted by $\mathrm{res}(F, G, x)$.

Step 2, Computing Border Polynomial (Cont'd)

➜ The **successive resultant** of f_i with respect to the triangular set $\{f_{i-1}, ..., f_1\}$, denoted by R_i, for each i $(1 \le i \le s)$, is

$$R_i = \mathrm{res}(\mathrm{res}(\cdots \mathrm{res}(\mathrm{res}(\mathrm{res}(f_i, f_i^{'}, x_i), f_{i-1}, x_{i-1}), f_{i-2}, x_{i-2}) \cdots), f_1, x_1).$$

Obviously, $R_1 = \mathrm{res}(f_1, f_1^{'}, x_1)$

➜ For each of those inequalities and inequations, the **successive resultant** of g_j (resp. h_j) w.r.t. the triangular set $[f_1, ..., f_s]$, denoted by Q_j (resp. Q_{l+j}) is

$$Q_j) = \mathrm{res}(\mathrm{res}(\cdots \mathrm{res}(\mathrm{res}(g_j, f_s, x_s), f_{s-1}, x_{s-1}) \cdots), f_1, x_1).$$

➜ For an SAS $\mathbb{T}$ of the form (3), the *border polynomial* of $\mathbb{T}$ is

$$BP = \prod_{i=1}^{s} R_i \prod_{j=1}^{l+m} Q_j.$$

Example 2

For the triangular system $\mathbb{T}_1$ in Example 1, the border polynomial is

$$BP = b(b-2)(b+2)(b^2-2)(b^4-4b^2+2)(2b^4-2b^2+1).$$

Step 3, Choosing Sample Points and Calculating Distinct Real Solutions at Each Sample Point

Theorem Let $\mathbb{T}$ be a PSAS of the form (3) and BP its border polynomial. Then, in each connected component of the complement of $BP = 0$ in parametric space $\mathbb{R}^d$, the number of distinct real solutions of $\mathbb{T}$ is constant; and each polynomial in $\mathbb{G}_1 \cup \mathbb{G}_2 \cup \mathbb{H}$ keeps the same sign.

- ➜ $BP = 0$ decomposes the parametric space into a finite number of connected region;
- ➜ Choose sample points in each connected component of the complement of $BP = 0$ with PCAD;
- ➜ Calculate the number of distinct real solutions of $\mathbb{T}$ at each sample point.

A Computer Algebra Tool: DISCOVERER

➜ **DISCOVERER** can be downloaded for free via "http://www.is.pku.edu.cn/~xbc/discoverer.html".

➜ **Main Features:**

- **Real Solution Classification of PSASs** Determines the conditions on parameters such that the given system has the given number of distinct real solutions.
- **Real Solution Isolation of CSASs** Determines the number of its distinct real solutions, say n, and moreover, can find out n disjoint cubes with rational vertices in each of which there is only one solution.

LOOP ABSTRACTION AND DEFINITIONS

Semi-Algebraic Transition Systems (SATS)

Definition: An *SATS* is a quintuple $\langle V, L, T, \ell_0, \Theta \rangle$.

- V is a set of program variables over $\mathbb{R}$
- L a set of locations
- T is a set of transitions which are of the form $\langle \ell_1, \ell_2, \rho_\tau, \theta_\tau \rangle$, where ℓ_1 and ℓ_2 are the pre- and post- locations of the transition, ρ_τ is the transition relation, and θ_τ is the guard of the transition
- ℓ_0 is the initial location, and
- Θ is the initial condition.
- θ_τ and Θ are polynomial assertion over V, while ρ_τ is polynomial assertion over $V \cup V'$.

Example 3

Integer $(x, y) := (0, 0);$

$l_0 :$ **while** $x \geq 0 \wedge y \geq 0$ **do**

$(x, y) := (x + y^2, y + 1);$

end while

(a) Program P

$$P = \{\ V = \{x, y\},$$
$$L = \{l_0\},$$
$$T = \{\tau\},\ \ l_0,$$
$$\Theta = x = 0 \wedge y = 0\}$$

where $\tau = \langle l_0, l_0, x' - x - y^2 = 0 \wedge y' - y - 1 = 0,$
$x \geq 0 \wedge y \geq 0\rangle$

(b) P's SATS

Invariant

- $PF(V)$ stands for the set of polynomial formulae in which all polynomial are in the variables of V.

Invariant at a Location Let $P = \langle V, L, T, l_0, \Theta \rangle$ be an SATS. An invariant at a location $l \in L$ is a polynomial formula $\phi \in PF(V)$ such that ϕ holds on all states that can be reached at location l.

Invariant of Program An *assertion map* for an SATS $P = \langle V, L, T, l_0, \Theta \rangle$ is a map $\eta : L \mapsto PF(V)$ that associates each location of P with a formula of $PF(V)$. An assertion map of P is said to be an *invariant* of P iff the following conditions hold:

Initiation: $\Theta(V_0) \models \eta(l_0)$.

Consecution: For each transition $\tau = \langle l_i, l_j, \rho_\tau, \theta_\tau \rangle$,

$$\eta(l_i)(V) \wedge \rho_\tau(V, V') \wedge \theta_\tau(V) \models \eta(l_j)(V').$$

Ranking Function

- A sequence of transitions $l_{11} \xrightarrow{\tau_1} l_{12}, \ldots, l_{n1} \xrightarrow{\tau_n} l_{n2}$ is called *composable* if $l_{i2} = l_{(i+1)1}$ for $i = 1, \ldots, n-1$, and written as $l_{11} \xrightarrow{\tau_1} l_{12}(l_{21}) \xrightarrow{\tau_2} \cdots \xrightarrow{\tau_n} l_{n2}$.
- A composable sequence is called *transition circle* at l_{11}, if $l_{11} = l_{n2}$.
- **Ranking Function:** Let $P = \langle V, L, T, l_0, \Theta \rangle$ be an SATS. A ranking function is a function $\gamma : Val(V) \to \mathbb{R}^+$ such that the following conditions are satisfied:

  Bounded: $\Theta(V_0) \models \gamma(V_0) \geq 0$.

  Ranking: There exists a constant $C \in \mathbb{R}^+$ such that $C > 0$ and for any transition circle at l_0 $l_0 \xrightarrow{\tau_1} l_1 \xrightarrow{\tau_2} \cdots \xrightarrow{\tau_{n-1}} l_{n-1} \xrightarrow{\tau_n} l_0$,

$$\rho_{\tau_1;\tau_2;\cdots;\tau_n}(V, V') \wedge \theta_{\tau_1;\tau_2;\cdots;\tau_n}(V) \models \gamma(V) - \gamma(V') \geq C \wedge \gamma(V') \geq 0,$$

- Existence of a ranking function implies termination of the loop

Polynomial Invariant Generation

(Y. Chen, B. Xia, L. Yang, N. Zhan, [FMRTS 07, LNCS 4711])

1. Predefining Invariant

Predefine a template of invariants as a PSAS at each of the underlining locations. All of these predefined PSASs form a parametric invariant of the program.

Example 4 For example, we can assume a template of invariants of P at l_0 in Example 3 as

$$eq(x,y) = a_1 y^3 + a_2 y^2 + a_3 x - a_4 y = 0 \tag{4}$$

$$ineq(x,y) = b_1 x + b_2 y^2 + b_3 y + b_4 > 0, \tag{5}$$

I.e. $\eta(l_0) = (4) \wedge (5)$.

Note that we can split η to η_1 and η_2 by letting $\eta_1(l_0) = (4)$ and $\eta_2(l_0) = (5)$. It is easy to prove that η exists iff η_1 and η_2 exist.

2. Deriving PSASs from Initial Condition and Solving the Resulted PSASs

➜ **Deriving PSASs**

- In Examples 3&4, $\Theta \models \eta_1(l_0)$ is equivalent to that
$$x = 0, y = 0, eq(x, y) \neq 0 \tag{6}$$
has no real solution;
- And $\Theta \models \eta_2(l_0)$ is equivalent to that
$$x = 0, y = 0, ineq(x, y) \leq 0 \tag{7}$$
has no real solution.

➜ **Solving the Resulted PSASs**

- For (6), by calling
$$\textbf{tofind}(([x, y], [\,], [\,], [eq(x, y)], [x, y], [a_1, a_2, a_3, a_4], 0)$$
we get that (6) has no real solution iff *true*.
- Similarly, (7) has no real solution iff
$$b_4 > 0. \tag{8}$$

3. Deriving PSASs from Consecutive Condition and Solving the Resulted PSASs

- For η_1 (resp. η_2), we can derive the following PSASs without real solution:

$$eq(x,y)=0 \wedge x'-x-y^2=0 \wedge y'-y-1=0 \wedge eq(x',y') \neq 0 \quad (9)$$

$$ineq(x,y)>0 \wedge x'-x-y^2=0 \wedge y'-y-1=0 \wedge ineq(x',y') \leq 0. \quad (10)$$

- By DISCOVERER, (9) (resp. (10)) has no real solution iff

$$a_3y^2+3a_1y^2+2ya_2+3a_1y-a_4+a_2+a_1=0 \wedge (a_3(a_1y^2+ya_2-a_4) \leq 0, \quad (11)$$

$$b_4+b_3+b_2+2b_2y+b_3y+b_2y^2+b_1x+b_1y^2>0). \quad (12)$$

- Simplifying (11) (resp. (12)) by QEPCAD, and obtaining

$$-a_4+a_2+a_1=0 \wedge 3a_1+2a_2=0 \wedge a_3+3a_1=0, \quad (13)$$

$$b_1+b_2 \geq 0 \wedge b_1 \geq 0 \wedge b_2+b_3+b_4>0 \wedge$$

$$(b_3+2b_2 \geq 0 \vee (b_1b_2+b_2^2 \geq 0 \wedge 4b_2b_4+4b_1b_4+4b_1b_3+4b_1b_2-b_3^2>0)) \quad (14)$$

4. Generating Invariant

- ➜ From (13), by using DISCOVERER, we get an instantiation

$$(a_1, a_2, a_3, a_4) = (-2, 3, 6, 1).$$

$\eta_1(l_0) = -2y^3 + 3y^2 + 6x - y = 0$.

- ➜ From $(8) \wedge (14)$, by PCAD of DISCOVERER, it results the following instantiation

$$(b_1, b_2, b_3, b_4) = (1, -1, 2, 1)$$

that is, $\eta_2(l_0) = x - y^2 + 2y + 1 > 0$.

- ➜ Finally, we get the following invariant for the program P:

$$\begin{cases} -2y^3 + 3y^2 + 6x - y = 0, \\ x - y^2 + 2y + 1 > 0 \end{cases}$$

Ranking Function Synthesis ([ICTAC 07])

Example 4

$\{a \geq 0\}$ $b = 0; c = 1;$

while $(c^2 \leq a)$ **do**

$c = 2c$;

end while

l_0 : **while** $c \geq 2$ **do**

$c = c/2$;

if $(b+c)^2 \leq a$ **then**

$b = b + c$;

end if

end while

return b;

$P = \{$

$V = \{a, b, c\}$

$L = \{l_0\}$

$T = \{\tau_1, \tau_2\}$

$\Theta = a \geq 0 \wedge b = 0 \wedge c \geq 1 \wedge c^2 > a$

where

$\tau_1 : \langle l_0, l_0, a' = a \wedge b' = b + c/2 \wedge c' = c/2,$
$c - 2 \geq 0 \wedge (2b + c)^2 \leq 4a\rangle$

$\tau_2 : \langle l_0, l_0, a' = a \wedge b' = b \wedge c' = c/2,$
$c \geq 2 \wedge (2b + c)^2 > 4a\rangle$

$\}$

No Linear Ranking Function

- ➜ Assume a linear ranking function $\gamma = ax + b$.

  Bounded:

$$b + 21a \geq 0 \tag{15}$$

  Decreasing Condition for First Branch: No solution

$$x \geq 1, x' = 1 - x, ax' + b < 0 \quad \text{and} \tag{16}$$
$$x \geq 1, x' = 1 - x, C > 0, ax' + b - (ax + b) < C \tag{17}$$

  Decreasing Condition for Second Branch: No solution

$$x \leq -1, x' = -x - 2, ax' + b < 0 \quad \text{and} \tag{18}$$
$$x \leq -1, x' = -x - 2, C > 0, ax' + b - (ax + b) < C \tag{19}$$

- ➜ **Completeness:** If a program has ranking function of the given template, the method indeed can discover one of them.
- ➜ **Conclusion:** The program has no linear ranking function.

Nonlinear Ranking Function

- ➜ Assume nonlinear ranking functions $\gamma = ax^2 + bx + c$, and $C = 1$.
- ➜ Applying the procedure given above to reduce and then using DISCOVERER, produce the condition

$$
\begin{aligned}
c + 21b + 441a \geq 0 \wedge a \geq 0 \wedge c \geq 0 \wedge (b \leq 0 \vee 4ac - b^2 \geq 0) \wedge \\
b + a - 1 \geq 0 \wedge a \geq 0 \wedge c + b + a \geq 0 \wedge \\
(b + 2a \leq 0 \vee 4ac - b^2 \geq 0) \wedge a \geq 0 \wedge 2b + 1 \leq 0
\end{aligned} \tag{20}
$$

- ➜ **Termination Analysis of Example 4.**
  - Using DISCOVERER, obtain a non-linear ranking function: $2x^2 - x + 3$ ($x^2 + 1$ can be another one).
  - The example terminates at all reals except integers: $x = 2 \pmod 3$.
  - For any given terminating input there exists a ranking function.
  - For input in $[-2, -1)$ (and some other intervals) it terminates but has no polynomial (even continuous) ranking function.

DISCUSSIONS

- ➜ **Completeness:** In the sense, if a program has an invariant or ranking function of the given template, the methods indeed can generate it
- ➜ The approaches can be applied to more general programs and to synthesize more expressive invariants
- ➜ Difference between Ranking Function and Invariant
  - Ranking function is global
  - Invariant may be either global or local
  - Ranking function can be seen as a global invariant
  - In general, it's difficult to handle ranking function for nested loops, but invariants can be dealt with a uniform method for all kinds of loops

Computing Reachable Set of Linear Hybrid Systems

Hybrid Systems:

- A mixture of Continuous (differential equations) and Discrete (events) states
- Software Embedded Systems
- Safety Critical Systems
- Interdisciplinary Subject: Control Theory $+$ Computer Science

Most Recent Results: Symbolic reachability computation for families of linear vector Fields (G. Lafferriere, G.J. Pappas and S. Yovine, J. Symbolic Computation 11, 2001)

➜ Linear Hybrid Systems

$$\dot{\xi} = A\xi + Bu$$

- $\xi(t) \in \mathbb{R}^n$ – state of the system at time t,
- $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$ – system matrices, and
- $u : \mathbb{R} \to \mathbb{R}^m$ – control input.

➜ Given $x = \xi(0)$ and u, the solution of the differential equation for any time $t \geq 0$ is

$$\xi(t) = \Phi(x, u, t) = e^{At}x + \int_0^t e^{A(t-\tau)}Bu(t)d\tau$$

where

$$e^{At} = \sum_{k=0}^{\infty} \frac{t^k}{k!}A^k$$

➜ Given $\mathcal{U}$, a set of control inputs, state y is said reachable from state x, if there exists control input $u \in \mathcal{U}$ and $t \geq 0$ such that $y = \Phi(x, u, t)$.

- **Decidability of Reachability**

  ① A – *nilpotent* matrix, and $\mathcal{U}$ – polynomials in t;
  ② A – *diagonalizable* matrix with rational eigenvalues, and $\mathcal{U}$ – linear combinations of exponentials;
  ③ A – *diagonalizable* matrix with purely imaginary eigenvalues, and $\mathcal{U}$ – linear combinations of sinusoids.

- To compute the reachability LPY transforms the above into Semi-Algebraic System (SAS) problem.

Example 5

- Let B be a unit matrix. Consider the diagonal matrix A and $\mathcal{U} = \{u\}$ defined as

$$A = \begin{bmatrix} 2 & 0 \\ 0 & -1 \end{bmatrix}, \quad u(t) = \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix} = \begin{bmatrix} -ae^{\frac{1}{2}t} \\ ae^{t} \end{bmatrix}, \text{ with } a \geq 0.$$

- Thus,

$$\Phi(x_1, x_2, u, t) = \begin{bmatrix} x_1 e^{2t} + \frac{2}{3}a(-e^{2t} + e^{\frac{1}{2}t}) \\ x_2 e^{-t} + \frac{1}{2}a(e^{t} - e^{-t}) \end{bmatrix}$$

- Let the initial set be $X = \{(0,0)\}$. Then the reachable set from X is:

$$\begin{aligned} \{(y_1, y_2) \in \mathbb{R}^2 \quad | \quad & \exists a \exists t : 0 \leq a \wedge t \geq 0 \\ & \wedge y_1 = x_1 e^{2t} + \tfrac{2}{3}a(-e^{2t} + e^{\frac{1}{2}t}) \\ & \wedge y_2 = x_2 e^{-t} + \tfrac{1}{2}a(e^{t} - e^{-t})\} \end{aligned}$$

➜ Let $z = e^{\frac{1}{2}t}$, thus, we get

$$\{(y_1, y_2) \in \mathbb{R}^2 \quad | \quad \exists a \exists z : 0 \leq a \wedge z \geq 1 \wedge p_1 = 0 \wedge p_2 = 0\}$$

where

$$\begin{aligned} p_1 &= y_1 - \tfrac{2}{3}a(-z^4 + z), \\ p_2 &= y_2 z^2 - \tfrac{1}{2}a(z^4 - 1). \end{aligned}$$

a, z are variables and y_1, y_2 are parameters

- Since the quantifiers cannot be eliminated using **REDLOG** or **QEPCAD** alone, LPY applied **REDLOG** to eliminate a first and then used **QEPCAD** to eliminate z, and thus obtained

$$\begin{aligned}&\{(y_1, y_2) \in \mathbb{R}^2 \mid (y_2 > 0 \wedge\ y_1 + y_2 \leq 0)\\ &\vee (y_2 < 0 \wedge\ y_1 + y_2 \geq 0) \vee\ 4y_2 + 3y_1 = 0\}\end{aligned} \tag{21}$$

- **Note that** (21) includes mistakes such as $(y_1, y_2) = (2, -1)$, $(y_1, y_2) = (1, -1)$ and $(y_1, y_2) = (4, -3)$.
- With **DISCOVERER**, state (y_1, y_2) is reachable if and only if

$$(y_2 > 0 \wedge\ y_1 + y_2 < 0) \vee (y_1 = y_2 = 0)$$

The mistakes are avoided.
(L. Yang, N. Zhan, B. Xia and C. Zhou: Program Verification by Using DISCOVERER. Proc. of VSTTE, LNCS 4171.)

Complexity Analysis

- For a PSAS S, directly applying quantifier elimination to S has complexity $\mathcal{O}((2d)^{2^{2n+8}}(s+m)^{2^{n+6}})$.
- The total cost is $\mathcal{O}(k(2d)^{2^{2n+8}}(s+m)^{2^{n+6}})$ for directly applying the technique of quantifier elimination to invariant generation.
- For a PSAS S, the cost of DISCOVERER plus QEPCAD is $\mathcal{O}(s^{n^{\mathcal{O}(1)}}(d+1)^{n^{\mathcal{O}(1)}}) + \mathcal{O}(1)\mathcal{O}(2D^{2^{2t+8}})))$, where $D = \mathcal{O}(s^{\mathcal{O}(s^2+s^2n^{\mathcal{O}(1)})}(d+1)^{\mathcal{O}(s^2n^{\mathcal{O}(1)})})$, and t is the dimension of the ideal generated by the s polynomial equations.
- The total cost of our approach is $\mathcal{O}(k*(\mathcal{O}(s^{n^{\mathcal{O}(1)}}(d+1)^{n^{\mathcal{O}(1)}}) + \mathcal{O}(1)\mathcal{O}(2D^{2^{2t+8}})))$.
- This approach can dramatically reduce the complexity, in particular when t is much less than n.

Conclusion and Future Work

Conclusion

- Proposed new approaches to program verification by applying theories and tools on solving semi-algebraic systems
- Proved that, in compared speaking, our approach is efficient by analyzing the complexity;
- The approaches for polynomial invariant generation and non-linear ranking function discovering are also **complete**;
- Similar approach can be applied to termination analysis of programs

Future Work

- How to further improve the efficiency is still a big challenge;
- How to handle programs with complicated data structures;
- how to combine our approach with other program verification techniques;
- . . .

Thank You

Questions?