

Abstraction Mechanisms across the Board: A Short Introduction.

Eric Feron

Department of Aeronautics and Astronautics,
Massachusetts Institute of Technology.

March 28th, 2004. Workshop on Robustness, Abstractions and Computations. University of Pennsylvania, Philadelphia, PA.

Outline

- Introduction
- XXXX-XXXXXXXXXX
- XXXXXX-XXX XXXXXX XXXXXX XXXXXXXX XX-XXXXX
- XXXXXX-XXX XXXXXX XXXXXX XXXXXXXXXXXXXXXX
- XXXXXX-XXX XXXXXX XXXXXX XXXXXXXX XXX
- XXXXXX-XXX XXXXXX XXXXXX XXXXXXXX XXXXXXXXXXXXXXXX
- Conclusions

Computer Programs as Dynamical Systems

- A computer program can be viewed as a rule for iterative modification of the operating memory.
- In more specifics, computer program models defined by the dynamical system $\mathcal{S} = \mathcal{S}(X, f, X_0, X_\infty)$ are considered.
- X is the state space, $X_0 \subset X$, and $X_\infty \subset X$ are sets of initial and terminal states and the set-valued function $f : X \rightarrow 2^X$ is such that $f(x) \subset X_\infty$ for all $x \in X_\infty$.
- To each dynamical system $\mathcal{S} = \mathcal{S}(X, f, X_0, X_\infty)$ corresponds the set of all sequences $x = (x(0), x(1), \dots, x(t), \dots)$ of elements of X , satisfying $x_0 \in X_0$, $x(t+1) \in f(x(t))$, $\forall t \in \mathbb{Z}^+$.

Software Analysis

- Verification of essential properties of $\mathcal{S} = \mathcal{S}(X, f, X_0, X_\infty)$.
- Finite-time termination: $\forall x(0) \in X_0, \exists T > 0$, s.t. $x(t) \in X_\infty$,
 $\forall t \geq T$.
- Absence of overflow: $\forall x(0) \in X_0, \forall t \geq 0, x(t) \in \underline{X} \subset X$.

Abstracted Model

- An Abstracted model of a computer program is a simplified but not perfectly accurate model of the original program.
- The model $\mathcal{S}(\widehat{X}, \widehat{f}, \widehat{X}_0, \widehat{X}_\infty)$ is an abstraction of $\mathcal{S}(X, f, X_0, X_\infty)$ if $X \subset \widehat{X}$, $\widehat{X}_\infty \subset X_\infty$, $X_0 \subset \widehat{X}_0$, $f(x) \subset \widehat{f}(x)$, $\forall x \in X$.
- Validity of certain properties such as finite-time termination and boundedness in the abstracted model $\mathcal{S}(\widehat{X}, \widehat{f}, \widehat{X}_0, \widehat{X}_\infty)$ imply their validity in the original model $\mathcal{S}(X, f, X_0, X_\infty)$.

An Example:

```

 $T(1, \dots, n)$ ; array of reals
 $I \in [I_{\min}, I_{\max}]$ ; uncertain integer input
 $l = l_0$ ; Integer within  $\{1, \dots, n\}$  sets the initial condition
  for  $k = 1 : K$ 
     $x = |a| \cdot T(l) + b$ ;
     $l = [c \cdot x + d \cdot l + e \cdot I]$ ;
  end

```

- Assume that M is the overflow limit. Let $T_{\max} := \max_{1 \leq i \leq n} T(i)$ and $T_{\min} := \min_{1 \leq i \leq n} T(i)$. if $[|a| \cdot T_{\min} + b, |a| \cdot T_{\max} + b] \subset [-M, M]$, then the variable x is guaranteed to remain bounded, provided that the index l , remains within the feasible set $\{1, \dots, n\}$.

- An Abstracted model of the above program is then give by

T_{\min}, T_{\max} : *real* variable

$I \in [I_{\min}, I_{\max}]$; uncertain *real* input

$x \in [|a| \cdot T_{\min} + b, |a| \cdot T_{\max} + b] \subset [-M, M]$; uncertain *real* input

$l \in [1, n]$; uncertain *real* initial condition/input

for $k = 1 : K$

$l = c.x + d.l + e.I$;

end

- Verify that subject to the above transformation, l , remains within the interval $[1, n + 1)$.
- At an abstract level, a universal language is sufficient. For practical considerations such as availability of an efficient relaxation technique and compatibility with an optimization engine, specific modeling techniques come a necessity.

Models of Computer Programs

- Linear systems with conditional switching:

$$X = \{0, 1, 2, \dots, m\} \times \mathbb{R}^n, \quad X_0 = \{0\} \times \mathbb{R}^n, \quad X_\infty = \{m\} \times \mathbb{R}^n.$$

$f : X \mapsto 2^X$ is defined by matrices $A_k, B_k, L_k, F_k, G_k, H_k, C_k, D_k$, where $k \in \{0, 1, \dots, m-1\}$, as well as by a function $p : \{0, \dots, m-1\} \mapsto \{0, \dots, m\}$, according to the following rule:

$$f(k, v) = \{(k+1, A_k v + B_k w + L_k) : w \in [-1, 1]\}$$

when $C_k v + D_k \leq 0$ and $k < m$,

$$f(k, v) = \{(p(k), F_k x + G_k w + H_k) : w \in [-1, 1]\}$$

when $C_k v + D_k > 0$ and $k < m$, and $f(k, v) = \{m, v\}$ when $k = m$.

- Mixed integer/linear systems:

$X = \mathbb{R}^n$ is the state space. The state transition map $f : X \mapsto 2^X$ is defined by two matrices F, H of appropriate dimensions, according to

$$f(x) = \{F[x; w; v; 1] : H[x; w; v; 1] = 0, w \in [-1, 1]^q, v \in \{-1, 1\}^r\}.$$

Every piecewise linear map on X can be written in this format.

- Trigonometric polynomial models:

The state space X is a direct product of sets of the form \mathbb{T}^k or \mathbb{Z}_q^k , where \mathbb{Z}_q denotes the set of all complex numbers z such that $z^q = 1$. The state transition map $f : X \mapsto 2^X$ is defined by a vector polynomial p of $2n+k$ complex variables, according to

$$f(x) = \{y \in X : p(y, x, z) = 0 \text{ for some } z \in \mathbb{T}^k\}.$$

Lyapunov-like Invariants for Software Analysis

- Consider a computer program modeled as a mixed integer/linear system. If there exists a constant $\theta > 1$ and a function $V(.) : \mathbb{R}^n \rightarrow \mathbb{R}$, which satisfies

$$1)V(x(0)) < 0, \quad 2)V(x(k+1)) < \theta V(x(k)), \quad 3)V(x(k)) > \left\| \frac{x(k)}{M} \right\| - 1$$

along any trajectory of the system, then the program will not overflow and will terminate in finite time.

- Appropriate system invariants for the two other suggested models of computer programs have a similar structure.
- Assuming a quadratic or linear form for the function $V(.)$, the search for a system invariant, reduces to solving a finite number of LMIs or an LP respectively.

Systematic Improvement of Analysis

- Increasing the depth (memory) of the model:

$$x(k+2) = FX(k), \text{ s.t. } HX(k) = 0, k = 1, 2, \dots$$

- Then, search for a function $V(.) : \mathbb{R}^n \rightarrow \mathbb{R}$, which satisfies

$$\begin{aligned} V(x(0)) &< 0, & V(x(1)) &< 0 \\ V(x(k+2)) &< \theta V(x(k)) & V(x(k-2)) &> \left\| \frac{x(k)}{M} \right\| - 1 \end{aligned}$$

- This can be extended by increasing the memory of the model even more.
- Recursive search for invariants: i.e. search for invariants that establish the desired properties partially, then use them as new information about the system to find stronger invariants.

An Example

- Consider the program:

```
 $x_1 = 0; x_2 = 0;$   
while  $x_2 \leq 100$ ,  
  if  $x_1 \geq 0$ ,  
     $x_1 = x_1 - a;$   
  else  
     $x_1 = x_1 + b;$   
  end  
 $x_2 = x_2 + 1;$   
end
```

- This program can be modeled as a mixed integer/linear system with one binary and two slack variables:

- $x(k+1) = FX(k)$ subject to the additional linear constraint $HX(k) = 0$, $k = 1, 2, \dots$
- where $X(k) = [x(k); w_1(k); w_2(k); v(k); 1]$ and

$$x_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

$$F = \begin{bmatrix} 1 & 0 & 0 & 0 & -\frac{a+b}{2} & \frac{b-a}{2} \\ 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$H = \begin{bmatrix} 1 & 0 & -\frac{M}{2} & 0 & -\frac{M}{2} & 0 \\ 0 & 1 & 0 & R & 0 & R - 100 \end{bmatrix}$$

$$R = \frac{M+100}{2}.$$

M is the overflow limit.

- The program should be correct for all $a \in \left(-\frac{M}{101}, M\right)$ and $b \in \left(-\frac{M}{101}, M\right)$.

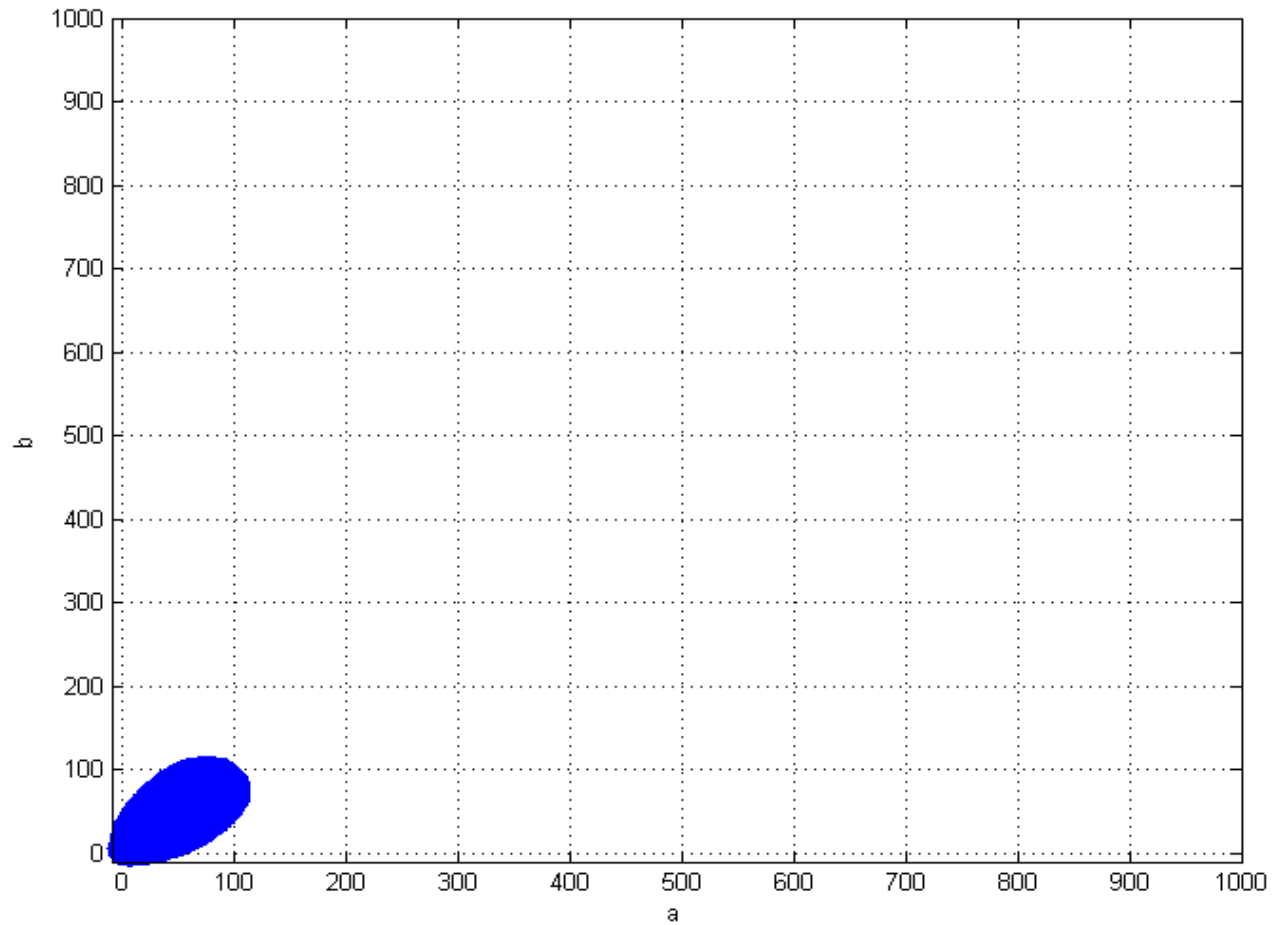


Figure 1: Shows the rather conservative results of analysis using QLF.

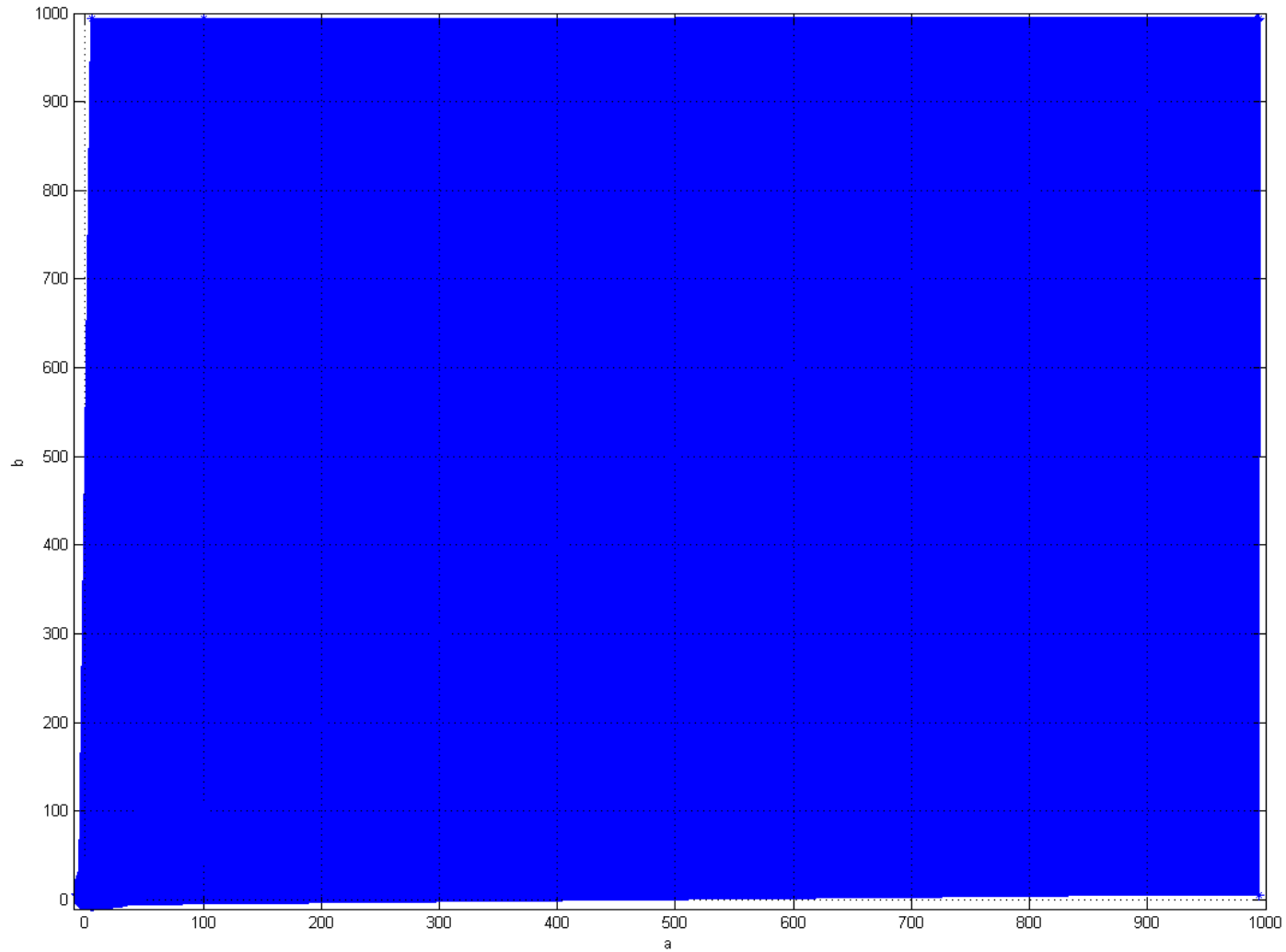


Figure 2: Analysis was significantly improved by increasing the memory of the model.