Distributed Lyapunov Functions in Analysis of Graph Models of Software *

Mardavij Roozbehani[1], Alexandre Megretski[1], Emilio Frazzoli[1], and Eric Feron[2]

[1] Laboratory for Information and Decision Systems
Massachusetts Institute of Technology (MIT), Cambridge, MA
mardavij@mit.edu, ameg@mit.edu, frazzoli@mit.edu

[2] Department of Aerospace Engineering
Georgia Institute of Technology, Atlanta, GA
feron@gatech.edu

Abstract. In previous works, the authors introduced a framework for software analysis, which is based on optimization of Lyapunov invariants. These invariants prove critical software properties such as absence of overflow and termination in finite time. In this paper, graph models of software are introduced and the software analysis framework is further developed and extended on graph models. A distributed Lyapunov function is assigned to the software by assigning a Lyapunov function to every node on its graph model. The global decremental condition is then enforced by requiring that the Lyapunov functions on each node decrease as transitions take place along the arcs. The concept of graph reduction and optimality of graphs for Lyapunov analysis is briefly discussed.

1 Introduction

Verification of safety-critical software systems presents itself with many challenges, including verification of the functional requirements, line-by-line verification of the code at the implementation level, and the need to prove absence of run-time errors. Due to its great potential to address these issues, static analysis has attracted computer scientists for decades. The book [13] provides an extensive collection of available results and techniques developed by computer scientists. Formal methods, including *Abstract Interpretation* [7], and *Model Checking* [5, 6] were developed in this endeavor to advance software verification.

While software verification has attracted little attention in the control community, recently, there have been renewed efforts at establishing properties of

* This work was supported by the National Science Foundation (NSF-0715025). Any opinions, findings, conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the supporting organization.

software systems by the combined use of abstractions and control theoretic principles. Much of the relevant literature in that regard may be found in the recent field of *hybrid systems* [10]. See for instance [8]. In general, it was found that many methods developed in system and control theory for systems driven by differential equations were in principle applicable to hybrid systems, possibly at the price of having to re-develop some elements of theory, e.g. optimal control theory on hybrid systems [12, 4, 2], computation of Lyapunov functions for hybrid systems [9], or control of hybrid systems using bisimulations [11].

The premise of using control theoretic tools for software verification is that such tools appear to adapt very well for analysis of software in aerospace, automotive, and many other safety-critical systems. For instance, in flight control systems, the software provides the control law to actuators that control the position of surfaces based on the pilot input and the current states. Our proposition is that since the embedded software implements a control law that is designed via system theoretic tools, such tools are best suited for verification at the implementation level. The analysis relies on a discrete event dynamical systems modeling of computer programs, and the verification method relies on numerical optimization in searching for system invariants. This paper complements existing results on software verification by further extending a previously established framework for transferring control theoretic tools to software analysis. This paper focuses on graph models in analysis of software systems via distributed Lyapunov functions.

1.1 Automated Software Analysis: Preliminaries

In this section we briefly review the principals of software analysis via dynamical system models. Interested readers are referred to [17],[15],[16] for more details.

Computer programs as dynamical systems

Exact and abstract representations of computer programs We will consider models defined in general by a *state space* set X with selected subsets $X_0 \subseteq X$ of *initial states* and $X_\infty \subset X$ of *terminal states*, and by a set-valued function $f : X \to 2^X$, such that $f(x) \subseteq X_\infty, \forall x \in X_\infty$. Thus, a computer program $\mathcal{P}$, represented by the dynamical system $\mathcal{S}(X, f, X_0, X_\infty)$ with parameters X, f, X_0, X_∞ is understood as the set of all sequences $\mathcal{X} := (x(0), x(1), \dots, x(t), \dots)$ of elements from X, satisfying

$$x(0) \in X_0, \qquad x(t+1) \in f(x(t)) \quad \forall t \in \mathbb{Z}_+ \tag{1}$$

Definition 1. *Consider a computer program $\mathcal{P}$ and its dynamical system representation $\mathcal{S}(X, f, X_0, X_\infty)$. Program $\mathcal{P}$ is said to terminate in finite time if every solution $\mathcal{X} \equiv x(.)$ of (1) satisfies $x(t) \in X_\infty$ for some $t \in \mathbb{Z}_+$.*

Definition 2. *Consider a computer program $\mathcal{P}$ and its dynamical system representation $\mathcal{S}(X, f, X_0, X_\infty)$. Program $\mathcal{P}$ is said to run without an overflow runtime error if for every solution $\mathcal{X} \equiv x(.)$ of (1) and for every $t \in \mathbb{Z}_+$, $x(t)$ does not belong to a certain (unsafe) subset X_- of X.*

In addition to *exact* dynamical systems models of computer programs, we also define *abstracted* models. We say that the model $\mathcal{S}(\hat{X}, \hat{f}, \hat{X}_0, \hat{X}_\infty)$ is an *abstraction* of $\mathcal{S}(X, f, X_0, X_\infty)$ (or simply an *abstraction* of $\mathcal{P}$), if $X \subseteq \hat{X}$, $X_0 \subseteq \hat{X}_0$, $\hat{X}_\infty \subseteq X_\infty$, and $f(x) \subseteq \hat{f}(x)$ for all $x \in X$.

Proposition 1. *[17] Consider a computer program $\mathcal{P}$ and its dynamical system representation $\mathcal{S}(X, f, X_0, X_\infty)$. Let $\mathcal{S}(\hat{X}, \hat{f}, \hat{X}_0, \hat{X}_\infty)$ be an abstraction of $\mathcal{P}$. Let X_- and $\hat{X}_-$, representing the overflow regions of $\mathcal{P}$ and its abstraction respectively, be such that $X_- \subseteq \hat{X}_-$. Assume that absence of overflow has been certified for the abstracted model of $\mathcal{P}$. Then, an overflow RTE will not occur during any execution of $\mathcal{P}$. In addition, if finite-time termination has been certified for the abstracted model, then program $\mathcal{P}$ will terminate in finite time.*

1.2 Lyapunov invariants as behavior certificates

We introduce Lyapunov-like invariants as *certificates* for the behavior of computer programs. We then describe the conditions under which, finding these Lyapunov-like invariants can be formulated as a convex optimization problem.

Definition 3. *A rate-θ Lyapunov invariant for system $\mathcal{S}(X, f, X_0, X_\infty)$ is defined to be a function $V : X \to \mathbb{R}$ such that*

$$V(x_+) - \theta V(x) < 0 \quad \forall x \in X,\ x_+ \in f(x) : x \notin X_\infty. \tag{2}$$

where $\theta > 0$ is a constant. Thus, a rate-θ Lyapunov invariant satisfies an invariant property ($V(x_+) - \theta V(x) < 0$) along the trajectories of (1) until they reach a terminal state.

Lemma 1. *[15] Consider a computer program $\mathcal{P}$, and its dynamical system model $\mathcal{S}(X, f, X_0, X_\infty)$ and assume that $\theta > 1$. If there exists a rate-θ Lyapunov invariant $V : X \to \mathbb{R}$, uniformly bounded on X, satisfying*

$$V(x) < 0 \quad \forall x \in X_0 \tag{3}$$

then $\mathcal{P}$ terminates in finite time.

Theorem 1. *[15] Consider a program $\mathcal{P}$, and let $\mathcal{S}(X, f, X_0, X_\infty)$ be its dynamical system model. Let θ be constant and let $\mathcal{V}$ denote the set of all rate-θ Lyapunov invariants for program $\mathcal{P}$. An overflow run-time error will not occur during any execution of $\mathcal{P}$, if there exists $V \in \mathcal{V}$ satisfying*

$$\inf_{x \in X_-} V(x) \geq \sup_{x \in X_0} V(x) \tag{4}$$

and at least one of the following three conditions holds:

$$(i)\ \ \theta = 1 \tag{5}$$

$$(ii)\ \ 0 < \theta < 1 \text{ , and } \inf_{x \in X_-} V(x) \geq 0 \tag{6}$$

$$(iii)\ \ 0 < \theta \text{ , and } 0 \geq \sup_{x \in X_0} V(x) \tag{7}$$

2 Graph Models in Analysis of Computer Programs

In this section we further extend and develop our software analysis framework on graph models. Practical considerations such as expressivity, convenience for automated parsing, existence of efficient relaxation techniques and compatibility with available numerical optimization engines render graph models an efficient and applicable model for analysis of real-time embedded software. In addition, graph models provide a convenient platform for mapping the proofs of correctness and certificates of performance from the model to the actual line of code at the implementation level. We will also see that graph models allow for trading off computational efforts at the parsing/modeling phase for computational efforts at the convex optimization phase and vice versa.

A graph model is essentially a generalized version of the model previously introduced for *Linear Programs with Conditional Switching* (LPwCS) [16], [17]. A graph model is defined on a directed graph $G(\mathcal{N}, \mathcal{E})$ with a set of nodes (*"effective"* lines of code) $\mathcal{N} := \{0, 1, \ldots, m\} \cup \{\bowtie\}$, and a set of arcs $\mathcal{E} := \{(i, j, k) \mid i \in \mathcal{N},\ j \in \mathcal{O}(i)\}$, where $\mathcal{O}(i)$ is the set of all nodes to which transition from node i is possible in one time step. Multiple arcs between nodes are allowed and the third element in the triplet (i, j, k) is the index of the k-th arc between nodes i and j. This model, has state space $X := \mathcal{N} \times \mathbb{R}^n$, with initial and terminal subsets defined as

$$X_0 := \{0\} \times v_0,\ v_0 \subseteq \mathbb{R}^n,\ X_\infty := \{\bowtie\} \times \mathbb{R}^n$$

where v_0 is a selected subset of $\mathbb{R}^n$, constrained by linear, quadratic or polynomial equalities and inequalities. On this graph, node 0 represents a (perhaps fictitious) line containing all the available information about the initial conditions of the continuous states. Node $\bowtie$ represents the terminal location and the definition of X_∞ as $\{\bowtie\} \times \mathbb{R}^n$ implies that our characterization of the terminal states depends only on the discrete component of the state space, i.e., a specific line of code, and is not (explicitly) dependent on the analog components of the state space. The set-valued map $f : X \to 2^X$, is defined by the transitions associated with the arcs on this graph, subject to certain rules associated with each arc.

The only possible transition involving node 0 is a transition from node 0 to node 1. The only possible transition from/to node $\bowtie$ is the identity transition to node $\bowtie$. Multiple arcs between nodes are allowed. The set $\mathcal{A}(i, j) := \{1, .., \overline{\kappa}_{ij}\}$ denotes the set of all indices of the arcs from node i to node j, where $\overline{\kappa}_{ij}$ denotes the total number of arcs from node i to j. We denote the k-th arc from node i to node j by (i, j, k). We attribute two labels to every arc (i, j, k) on this graph: (i) A *transition* label T^k_{ji}, to be understood as an operator defined on $\mathbb{R}^n$, which represents a set-valued function mapping the state (i, v) to all possible states $(j, \overline{v})$, where $\overline{v} \in T^k_{ji} v$. (ii) A *passport* label I^k_{ji}, to be understood as the indicator function of a semi-algebraic set, defined by a set of linear, quadratic, or polynomial equalities and inequalities.

$$P^k_{ji} := \left\{v \mid H^k_{ji}(v) = 0,\ Q^k_{ji}(v) \leq 0\right\},\ I^k_{ji}[v] := \begin{cases} 1 \text{ if } v \in P^k_{ji} \\ 0 \text{ if } v \notin P^k_{ji} \end{cases}$$

According to this definition, transition along arc (i,j,k) is possible if and only if $I_{ji}^{k}[v]=1$. A passport label $I1_{ji}^{k} \wedge I2_{ji}^{k}$ is understood as $I1_{ji}^{k} \cdot I2_{ji}^{k}$. Finally, the state transition map $f: X \to 2^{X}$ is given by

$$f(i,v)=\left\{\left(j,T_{ji}^{k}v\right) \mid j \in \mathcal{O}(i),\ I_{ji}^{k}[v]=1\right\}.$$

We have defined all the elements of the model. In reference to the graph model of a computer program, we will use the concise notation $G(\mathcal{N},\mathcal{E})$, with the convention that the nodes and arcs of G are appropriately labeled to define a valid model $\mathcal{S}(X,f,X_0,X_\infty)$ according to our previous discussion.

Remark 1. Multiple arcs between nodes enable modeling "or" or "xor" type conditional transitions. The passport labels associated with multiple arcs between nodes are not necessarily exclusive. Thus, multiple transitions along different arcs may be possible. This allows for nondeterministic modeling.

2.1 Lyapunov analysis of graph models

Consider a computer program $\mathcal{P}$, and its graph model $G(\mathcal{N},\mathcal{E})$. We are interested in finding Lyapunov functions that prove certain properties of $\mathcal{P}$. As we described before, the state in this model is defined by $x=(i,v)$ where i is the discrete component and v is the continuous component of the state vector. We define Lyapunov functions for this model in the following way:

$$V(x) \equiv V(i,v) := \sigma_i(v) \tag{8}$$

where for every $i \in \mathcal{N}$ the function $\sigma_k: \mathbb{R}^n \to \mathbb{R}$ is a quadratic, polynomial or an affine functional. This means that we assign a quadratic/linear/polynomial Lyapunov function to every node $i \in \mathcal{N}$ on graph $G(\mathcal{N},\mathcal{E})$. We will refer to Lyapunov functions defined according to (8) as node-wise Lyapunov functions.

Proposition 2. *Let $V(x)$ be defined according to* (8)*. The Lyapunov invariance condition*

$$V(x_+) < \theta V(x) \qquad \forall x \in X \setminus X_\infty,\ x_+ \in f(x)$$

holds true if and only if

$$\sigma_j(T_{ji}^{k}v) - \theta\sigma_i(v) < 0,\ \forall i \in \mathcal{N}\setminus\{\bowtie\},\ j \in \mathcal{O}(i),\ k \in \mathcal{A}(i,j),\ I_{ji}^{k}[v]=1. \tag{9}$$

Let $N_a := \sum_{(i,j)\in\mathcal{E}} |\mathcal{A}(i,j)|$ denote the total number of arcs on $G(\mathcal{N},\mathcal{E})$, excluding the identity transformation arc $(\bowtie,\bowtie)$. Then, according to proposition 2 the Lyapunov invariance condition is enforced via N_a constraints. Assume for the moment that each σ_i is a quadratic functional, and also that

$$P_{ji}^{k}[v] := \left\{v \mid E_i^k(v)=0,\ I_i^k(v) \le 0\right\}$$

where E_i^k and I_i^k are quadratic functionals. Each of the constraints in (9) expresses that a quadratic form must be negative whenever certain quadratic constraints are satisfied. Various forms of convex relaxations such as the $\mathcal{S}$-Procedure

in positivity of quadratic forms can be employed to formulate (9) as a convex optimization problem. In this case, the resulting optimization problem will be a semidefinite program [1]. In the presence of polynomial constraints, or if we allow $\sigma_i(v)$ to be polynomial functionals of v, the resulting optimization problem can be formulated as a sum of squares program [14]. Similarly, linear invariants subject to linear constraints lead to linear programming [3].

Finite-time termination In node-wise Lyapunov analysis of graph models, we often do not impose the same invariance rate θ along all the arcs, as this may lead to either infeasibility or to weak invariants. While computing the optimal value of θ per arc is neither possible nor necessary, depending on the state transitions along the arcs, certain choices of θ may be more reasonable than others. The following theorem provides a finite-time termination criterion via node-wise Lyapunov invariants defined on graph models.

Definition 4. *A cycle $\mathcal{C}_m$ on a graph $G(\mathcal{N},\mathcal{E})$ is an ordered list of m triplets (n_1,n_2,k_1), $(n_2,n_3,k_2),...,$ (n_m,n_{m+1},k_m), where $n_1=n_{m+1}$ and for all $j\in\{1,...,m\}$ we have $(n_j,n_{j+1},k_j)\in\mathcal{E}$. A simple cycle is a cycle that does not visit any node more than once. Thus, a simple cycle has the following property:*

$$\begin{gathered}\text{If } (n_i,n_{i+1},k_i)\in\mathcal{C}_m \text{ and } (n_j,n_{j+1},k_j)\in\mathcal{C}_m \text{ then}\\ n_{j+1}=n_i\Longrightarrow i=1 \text{ and } j=m\end{gathered}$$

Theorem 2. *Consider a computer program $\mathcal{P}$ and its graph model $G(\mathcal{N},\mathcal{E})$. Let $V(i,v):=\sigma_i(v)$ be a variable-rate invariant defined on G, in the sense that*

$$\sigma_0(v)<0,\ \forall v\in v_0 \tag{10a}$$

$$\begin{aligned}\sigma_j(T_{ji}^k v)<\theta_{ji}^k\sigma_i(v),\ \forall i\in\mathcal{N}\backslash\{\bowtie\},\ j\in\mathcal{O}(i),\\ k\in\mathcal{A}(i,j),\ v\in P_{ji}^k[.]\end{aligned} \tag{10b}$$

In addition, assume that V is bounded from below. Then, (10) proves that $\mathcal{P}$ terminates in finite time if and only if for every simple cycle $\mathcal{C}\in G$, we have

$$\prod_{(i,j,k)\in\mathcal{C}}\theta_{ij}^k>1,\ \mathcal{C}\in G \tag{11}$$

Proof. Proof of sufficiency proceeds by contradiction. Assume that (10) and (11) hold, but $\mathcal{P}$ does not terminate in finite time. Then, there exists a sequence $\mathcal{X}\equiv(x(0),x(1),\dots,x(t),\dots)$ of elements from X satisfying (1) that does not reach a terminal state in finite time. Let $L:X\to\mathcal{N}$ be an operator mapping every element x from X, to the discrete component of x. The sequence $L\mathcal{X}\equiv(0,1,\dots)$ is then a sequence of infinite length that takes only finitely many different values. Therefore, there exists at least one element which repeats infinitely often in $\mathcal{X}$. Let $\omega\in\mathcal{N}\backslash\{0,\bowtie\}$ be an element that repeats infinitely often in $L\mathcal{X}$ and let $\mathcal{C}[\omega]$ denote the set of all cycles on $G(\mathcal{N},\mathcal{E})$ that begin and end at ω. Define

$$\theta=\min_{\mathcal{C}\in\mathcal{C}[\omega]}\prod_{(i,j,k)\in\mathcal{C}}\theta_{ij}^k.$$

Note that (11) implies that $\theta > 1$. Let $\mathcal{W}$ be a subsequence of $\mathcal{X}$ consisting of all the elements from $\mathcal{X}$ that satisfy $Lx = \omega$, and rename the analog component of x at the k-th appearance of ω in $L\mathcal{X}$ by v_k to obtain the sequence $\mathcal{W} := ((\omega, v_1), (\omega, v_2), ..., (\omega, v_t), ...)$. Then we have $V_\omega(v_1) < 0$, and $V_\omega(v_{i+1}) < \theta V_\omega(v_i)$, and $\theta > 1$. The result follows immediately from Lemma 1. It is easy to construct counter examples to prove necessity. We do not give a counter-example here due to space limitation.

Absence of overflow The following result is a corollary of Theorems 1 and 2.

Corollary 1. *Consider a program $\mathcal{P}$ and its graph model $G(\mathcal{N}, \mathcal{E})$. Suppose that the overflow limit is defined by a positive real number M. That is, $X_- := \{x \in X \mid |x_i| \geq M\}$. Let $V(i, v) := \sigma_i(v)$ be a variable-rate invariant defined on G, as in (10a) and (10b). Assume that $V(i, v)$ additionally satisfies*

$$\sigma_i(v) > \left\| \frac{v}{M} \right\|^2 - 1 \qquad \forall v \in F_i,\ i \in \mathcal{N} \setminus \{0, \bowtie\}.$$

Then, an overflow runtime error will not occur during any execution of $\mathcal{P}$. In addition, if (11) holds, then $\mathcal{P}$ terminates in at most T steps, where

$$T = \sum_{\mathcal{C} \in G} \frac{\log M - \log \sup_{v \in v_0} |\sigma_0(v)|}{\log \theta(\mathcal{C})},\ \theta(\mathcal{C}) := \prod_{(i,j,k) \in \mathcal{C}} \theta_{ij}^k.$$

2.2 Towards Optimal Graph Models

Consider now the following two programs, $\mathcal{P}_1$ and $\mathcal{P}_2$.

program $\mathcal{P}_1$

loc 0 : % pre: $x_1, x_2 \in [-100, 100]$;
loc 1 : while *True*,
loc 2 : if $x_1^2 - x_2^2 \leq 0$
loc 3 : $x_1 = 0.99x_1 + 0.01x_2$;
$x_2 = -0.05x_1 + 0.99x_2$;
else
loc 5 : $x_1 = 0.99x_1 + 0.05x_2$;
$x_2 = -0.01x_1 + 0.99x_2$;
end
loc $\bowtie$: end

program $\mathcal{P}_2$

loc 0 : % pre: $x_1, x_2 \in [-100, 100]$;
loc 1 : while *True*,
loc 2 : while $x_1^2 - x_2^2 \leq 0$
loc 3 : $x_1 = 0.99x_1 + 0.01x_2$;
$x_2 = -0.05x_1 + 0.99x_2$;
end
loc 4 : while $x_1^2 - x_2^2 > 0$
loc 5 : $x_1 = 0.99x_1 + 0.05x_2$;
$x_2 = -0.01x_1 + 0.99x_2$;
end
loc $\bowtie$: end

The two programs $\mathcal{P}_1$ and $\mathcal{P}_2$ define exactly the same evolution path for the state variables x_1 and x_2. In other words the set of all sequences $\mathcal{X}(\mathcal{P}_1) := (x(0), x(1), \ldots, x(t), \ldots)$ of the dynamical system model $\mathcal{S}_1(X, f, X_0, X_\infty)$ of program $\mathcal{P}_1$, and the set of all sequences $\mathcal{X}(\mathcal{P}_2) := (x(0), x(1), \ldots, x(t), \ldots)$ of the dynamical system model $\mathcal{S}_2(X, f, X_0, X_\infty)$ of program $\mathcal{P}_2$ are identical. Thus, program $\mathcal{P}_1$ is correct if and only if program $\mathcal{P}_2$ is correct (indeed, both programs are correct in the sense of absence of overflow and their trajectories converge to the origin). Below, we construct the graph models of both programs and discuss analysis of the models via Lyapunov invariants. Let T_1 and T_2 be the transformations that take place upon leaving nodes 3 and 5 respectively, that is,

$$T_1 \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0.99x_1 + 0.01x_2 \\ -0.05x_1 + 0.99x_2 \end{bmatrix}, \text{ and } T_2 \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0.99x_1 + 0.05x_2x_2 \\ -0.01x_1 + 0.99x_2 \end{bmatrix}.$$

Also, define $P := \left\{x \mid x_1^2 - x_2^2 \leq 0\right\}$, $Q := \left\{x \mid x_1^2 - x_2^2 > 0\right\}$, $C_{1\bowtie} := \varnothing$.

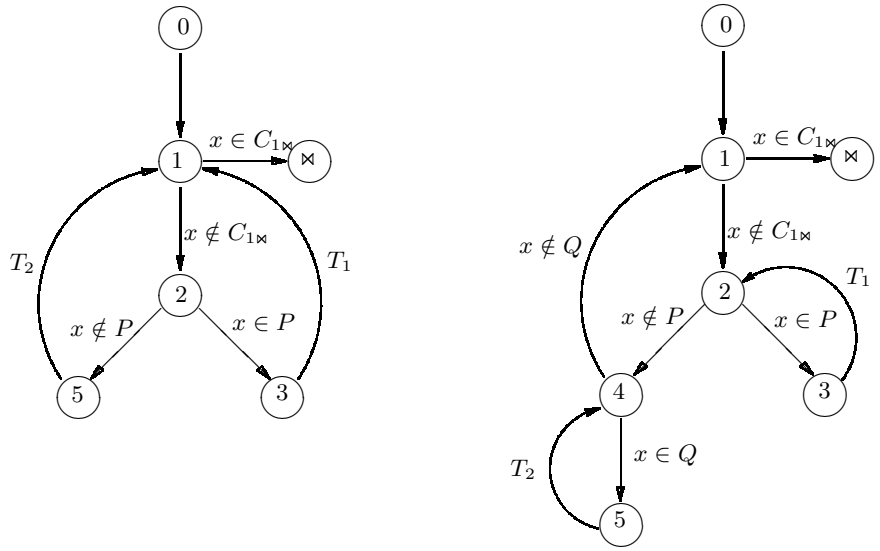


Fig. 1. Graph Models of Programs $\mathcal{P}1$ (left) and $\mathcal{P}2$ (right)

The graph models of programs $\mathcal{P}_1$ and $\mathcal{P}_2$ are shown in Figure 1. The graph model of $\mathcal{P}_1$ is of degree 4, and the graph model of $\mathcal{P}_2$ is of degree 5 (nodes 0 and $\bowtie$ are not counted). The transition labels associated with identity transitions along the arcs are dropped from the diagrams. As discussed before, we assign a quadratic Lyapunov function $\sigma_i(x) := x^T S_i x$ to every node on the graph and write the Lyapunov invariance condition according to Proposition (2). For these

programs we get

For program $\mathcal{P}_1$	For program $\mathcal{P}_2$
	$\sigma_0(x) < 0,\ \text{s.t. } x^2 \in [0, 10^4]$
$\sigma_0(x) < 0,\ \text{s.t. } x^2 \in [0, 10^4]$	$\sigma_1(x) < \sigma_0(x)$
$\sigma_1(x) < \sigma_0(x)$	$\sigma_2(x) < \sigma_1(x)\ \text{s.t. } x \notin C_{1\bowtie}$
$\sigma_2(x) < \sigma_1(x)\ \text{s.t. } x \notin C_{1\bowtie}$	$\sigma_3(x) < \sigma_2(x)\ \text{s.t. } x_1^2 - x_2^2 \leq 0$
$\sigma_3(x) < \sigma_2(x)\ \text{s.t. } x_1^2 - x_2^2 \leq 0$	$\sigma_4(x) < \sigma_2(x)\ \text{s.t. } x_1^2 - x_2^2 > 0$
$\sigma_5(x) < \sigma_2(x)\ \text{s.t. } x_1^2 - x_2^2 > 0$	$\sigma_2(T_1 x) < \sigma_3(x)$
$\sigma_1(T_1 x) < \sigma_3(x)$	$\sigma_5(x) < \sigma_4(x)\ \text{s.t. } x_1^2 - x_2^2 > 0$
$\sigma_1(T_2 x) < \sigma_5(x)$	$\sigma_4(T_2 x) < \sigma_5(x)$
$\sigma_{\bowtie}(x) < \sigma_1(x)\ \text{s.t. } x \in C_{1\bowtie}$	$\sigma_1(x) < \sigma_4(x)\ \text{s.t. } x_1^2 - x_2^2 \leq 0$
	$\sigma_{\bowtie}(x) < \sigma_1(x)\ \text{s.t. } x \in C_{1\bowtie}$

We can then use the $\mathcal{S}$-Procedure to convert the above constraints on $\sigma_i := x^T S_i x$ to semidefinite constraints and solve for the parameters of S_i. The result of this experiment is somewhat surprising. Although the two programs define the exact same trajectories for the state variables x_1 and x_2, the optimization problem arising from node-wise quadratic Lyapunov invariant analysis on graph model of $\mathcal{P}_2$ is feasible, while the optimization problem turns out infeasible for $\mathcal{P}_1$. Interestingly, this has nothing to do with the fact that there are more nodes in the graph model of $\mathcal{P}_2$, and that the Lyapunov function defined on the graph model of $\mathcal{P}_2$ has more parameters. To understand this situation better, we introduce the notions of reduction and minimality of graph models.

Definition 5. *A node $i \in \mathcal{N} \setminus \{0, \bowtie\}$ is called a focal node, if there exists a non-identity transition arc from node i to itself, that is,*

$$\exists k,\ \textit{s.t. } (i,i,k) \in \mathcal{E} \textit{ and } T_{ii}^k \neq I$$

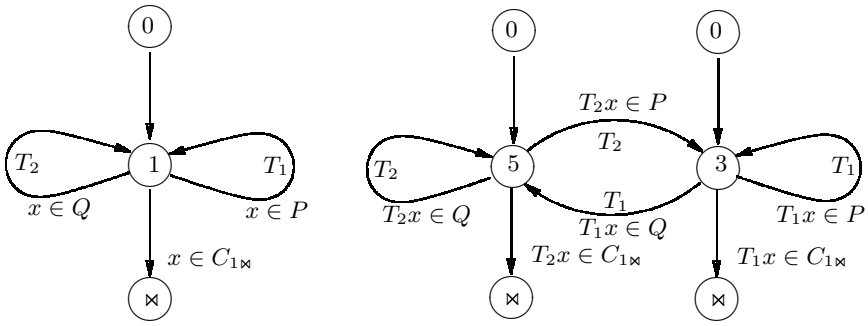
A node $i \in \mathcal{N} \setminus \{0, \bowtie\}$ is called an auxiliary node if it is not a focal node, that is,

$$\forall k : (i,i,k) \in \mathcal{E} \implies T_{ii}^k = I$$

Informally speaking, focal nodes are nodes with nontrivial self arcs and auxiliary nodes are nodes without nontrival self arcs. A graph model of a computer program $\mathcal{P}$ is called irreducible, if every node $i \in \mathcal{N} \setminus \{0, \bowtie\}$ is a focal node.

Consider a graph $G(\mathcal{N}, \mathcal{E})$ and let $\alpha \in \mathcal{N} \setminus \{0, \infty\}$ be an auxiliary node. A *reduced* graph $G_r(\mathcal{N}_r, \mathcal{E}_r)$ can be obtained from G in the following way: 1. Remove the auxiliary node α, and all the pertinent incoming and outgoing arcs 2. For every pair of arcs $\{(i, \alpha, r), (\alpha, j, s)\}$ where $i \in \mathcal{I}(\alpha)$ and $j \in \mathcal{O}(\alpha)$, add a new arc (i, j, k) with the transition label $T_{ji}^k := T_{j\alpha}^s T_{\alpha i}^r$ and passport

label $P_{ji}^{k} := P_{j\alpha}^{s} T_{\alpha i}^{r} \wedge P_{\alpha i}^{r}$. If $G_r\,(\mathcal{N}_r, \mathcal{E}_r)$ is a reduced graph model obtained from $G\,(\mathcal{N}, \mathcal{E})$, we write $G_r \sqsubseteq G$. An irreducible model of G can be obtained by repeating the above process until every auxiliary node is eliminated. Note that the irreducible graph of G is not unique, neither is its degree. Among all the irreducible offspring graphs of G, we call the one(s) with minimal degree, a minimal realization of G. The degree of a minimal realization of G, is called the effective or minimal degree of G. Similarly, among all the irreducible offsprings of G, we call the one(s) with maximal degree, a maximal realization of G. The degree of a maximal realization of G, is called the maximal degree of G, and is equal to the degree of G if and only if G is irreducible. If G is reducible, the minimal or maximal realizations of G are may not be unique either. Note that if $G_r \sqsubseteq G$, then $\mathcal{N}_r \subset \mathcal{N}$, while $\mathcal{E}_r \nsubseteq \mathcal{E}$. Also note that the set of all reduced graphs of G do not form an ordered set, in the sense that if $G_{r_1} \sqsubseteq G$ and $G_{r_2} \sqsubseteq G$, neither $G_{r_1} \sqsubseteq G_{r_2}$ nor $G_{r_2} \sqsubseteq G_{r_1}$ has to hold.



A Minimal Realization of $\mathcal{P}_1$ A Maximal Realization of $\mathcal{P}_1$

Fig. 2. Minimal and Maximal realizations of program $\mathcal{P}_1$

The graph model of Program $\mathcal{P}_1$ is of minimal degree 1, and maximal degree 2. A particular minimal and a particular maximal realization are show in Figure 2. It can be verified that the graph model of $\mathcal{P}_2$ is of minimal degree 2, and maximal degree 3. The minimal realization of $\mathcal{P}_2$ can be obtained via a reduction process that eliminates nodes 3, 5, and 1, exactly in that order. The maximal realization of $\mathcal{P}_2$ can be obtained via a reduction process that eliminates nodes 2, and 4, exactly in that order.

The following theorem states that assigning node-wise Lyapunov functions to graph models results in sufficient conditions for existence of Lyapunov invariants within a specific class of functions, e.g. quadratic functions, that are weaker than (i.e. always imply) the sufficient conditions imposed by assigning node-wise Lyapunov invariants to reduced models of the same graph. In other words, existence of node-wise Lyapunov invariants within a specific class of functions for the reduced model is a necessary but not sufficient condition for existence of node-wise Lyapunov invariants within the same class for the original graph model of a computer program.

Theorem 3. *Consider a computer program $\mathcal{P}$, and its graph model $G(\mathcal{N},\mathcal{E})$. Let $G_r(\mathcal{N}_r,\mathcal{E}_r) \sqsubseteq G(\mathcal{N},\mathcal{E})$ be any reduced model of G. If*

$$V(i,v) := \sigma_i(v),\ i \in \mathcal{N}$$

is a nodewise quadratic Lyapunov invariant on graph $G(\mathcal{N},\mathcal{E})$, then there exists a nodewise quadratic Lyapunov invariant $V_r(i,v)$ that is valid on $G_r(\mathcal{N}_r,\mathcal{E}_r)$. However, if

$$V_r(i,v) := \sigma_i(v),\ i \in \mathcal{N}_r$$

is a node-wise quadratic Lyapunov invariant on graph $G_r(\mathcal{N}_r,\mathcal{E}_r)$, a node-wise quadratic Lyapunov invariant that is valid on $G(\mathcal{N},\mathcal{E})$ may not even exist.

Proof. If $G_r \sqsubseteq G$, then there exists a sequence of reduced graph models G_i, $i = 1...q$, where $G_1 = G$, $G_{i+1} \sqsubseteq G_i$, and $G_q = G_r$ with the property that $|\mathcal{N}_{i+1}| = |\mathcal{N}_i| - 1$, that is, G_{i+1} is obtained by removing one auxiliary node from G_i. Further, assume that G_{i+1} is derived from G_i by eliminating node n, and that V is a Lyapunov invariant for G. Then,

$$V_n(T^r_{nm}v) - \theta V_m(v) < 0,\ m \in \mathcal{I}(n),\ r \in \mathcal{A}(m,n),\ I^r_{nm}[v] = 1,$$

$$V_l(T^s_{ln}v) - \theta V_n(v) < 0,\ l \in \mathcal{O}(n),\ s \in \mathcal{A}(n,l),\ I^s_{ln}[v] = 1.$$

Necessary conditions for the latter conditions to hold are that:

$$V_l(T^s_{ln}T^r_{nm}v) - \theta V_n(T^r_{nm}v) < 0,\ I^s_{ln}[T^r_{nm}v] = 1,$$
$$m \in \mathcal{I}(n),\ l \in \mathcal{O}(n),\ s \in \mathcal{A}(n,l),\ r \in \mathcal{A}(m,n).$$

This, and the first set of conditions imply that:

$$V_l(T^s_{ln}T^r_{nm}v) - \theta^2 V_m(v) < 0,\ I^r_{nm}[v] = 1,\ I^s_{ln}[T^r_{nm}v] = 1,$$
$$r \in \mathcal{A}(m,n),\ s \in \mathcal{A}(n,l).$$

By definition, this implies that V_l and V_m satisfy the Lyapunov conditions along all the arcs that were added in the reduction process. Since V_l and V_m satisfy all Lyapunov conditions along all the existing arcs (before reduction), we conclude that V is also a Lyapunov invariant for the reduced model. The result holds by induction. The proof also shows that

$$V_r(i,v) := \sigma_i(v),\ i \in \mathcal{N}_r$$

is a valid Lyapunov invariant on G_r.

In analysis of programs via Lyapunov invariants, an important issue is to determine whether a computer program admits certain type of Lyapunov invariants, e.g. quadratic, piece-wise quadratic, etc. For instance, consider program $\mathcal{P}_1$, which is known not to admit a quadratic Lyapunov invariant, while a piecewise quadratic Lyapunov invariant is known to exist. Recall that the graph model of this program is of degree 4 (not counting nodes 0, and $\bowtie$), of minimal degree

1, and maximal degree 2. Theorem 3 states that a Lyapunov invariant cannot be found by assigning four different quadratic Lyapunov functions to the four nodes on the graph. However, a Lyapunov function may be found by assigning two Lyapunov functions to each of the two nodes on the maximal realization of $\mathcal{P}_1$. As far as existence of Lyapunov functions is concerned, assigning different Lyapunov functions to an immediate graph model of a program is only as good as assigning fewer many Lyapunov functions to its minimal realization. In other words, more Lyapunov functions assigned to auxiliary nodes do not add more flexibility/power in Lyapunov analysis. The latter statement of the theorem is even more interesting since it states that performing analysis on reduced models may even be beneficial. This is indeed the case for the program $\mathcal{P}_1$. Since $\mathcal{P}_1$ does not admit a quadratic Lyapunov invariant, the optimization problem arising from the original graph of $\mathcal{P}_1$ is infeasible. So is the optimization problem arising from analysis of the minimal graph of $\mathcal{P}_1$. However, the optimization problem arising from analysis of the maximal graph of $\mathcal{P}_1$ (which is of degree 2) is feasible and a Lyapunov invariant was indeed found. On the other hand, since the minimal graph of $\mathcal{P}_2$ is of degree 2, the optimization problem arising from the original graph of $\mathcal{P}_2$ is readily feasible and a Lyapunov invariant is found. Same is true for analysis of minimal and maximal realization of $\mathcal{P}_2$.

So far, we have established that, at least from a theoretical point of view, it is beneficial to search for Lyapunov invariants on the reduced graph models rather than the original graph models of computer programs. From an optimization point of view, Theorem 3 compares two generally nonconvex optimization problems. It states that if the nonconvex optimization problem associated with the original graph model is feasible, then so is the nonconvex optimization problem associated with the reduced graph model. A natural question that arises here is about the computational procedure that will be used to compute the Lyapunov invariants on the two graph models. More specifically, the effects of convex relaxations on the computation of such invariants on the original and the reduced models must be investigated. It is interesting that the statement of Theorem 3 remains valid even after convex relaxations are applied to these nonconvex optimization problems. More specifically, the Lyapunov invariant $V_r(i, v)$ can be computed using the same convex relaxations that render the computation of $V(i, v)$ a feasible convex optimization problem. To make this concept clearer, let us consider a specific case. Consider a graph G, and assume that G_r is obtained by eliminating node 2, where, $\mathcal{I}(2) := \{1\}$, $\mathcal{O}\{2\} := \{3, 5\}$ (this is like the graph of program $\mathcal{P}_1$). Assume that each transition label T_{ji} is a finite-dimensional linear operator defined by the matrix of T_{ji}, and each passport label is defined by a single quadratic constraint: $P_{ji}[u] = \left\{u \mid u^T Q_{ji} u \leq 0\right\}$. The Lyapunov conditions as imposed on G, are:

$$\sigma_2(T_{21}v) - \theta\sigma_1(v) < 0, \text{ s.t. } v^T Q_{21} v \leq 0, \tag{12a}$$

$$\sigma_3(T_{32}v) - \theta\sigma_2(v) < 0, \text{ s.t. } v^T Q_{32} v \leq 0, \tag{12b}$$

$$\sigma_5(T_{52}v) - \theta\sigma_2(v) < 0, \text{ s.t. } v^T Q_{52} v \leq 0. \tag{12c}$$

The Lyapunov conditions as imposed on the reduced model G_r, are:

$$\sigma_3\left(T_{32}T_{21}v\right) - \theta\sigma_1\left(v\right) < 0,\ \text{s.t. } v^T Q_{21} v \leq 0,\ v^T T_{21}^T Q_{32} T_{21} v \leq 0, \tag{13a}$$

$$\sigma_4\left(T_{42}T_{21}v\right) - \theta\sigma_1\left(v\right) < 0,\ \text{s.t. } v^T Q_{21} v \leq 0,\ v^T T_{21}^T Q_{52} T_{21} v \leq 0. \tag{13b}$$

Now, let each σ_i be a quadratic functional, $\sigma_i\left(v\right) := v^T P_i v$. Note that each of the conditions in (12) express negativity of a quadratic form subject to a single quadratic constraint, while each of the conditions in (13), express negativity of a quadratic form subject to two quadratic constraints. It may be misleading to think that using the $\mathcal{S}$-Procedure as the convex relaxation method for (13) would be more conservative than for (12). However, as we have already suggested, this is not the case. Using the $\mathcal{S}$-Procedure, conditions (12) are converted to LMIs in the following way:

$$T_{21}^T P_2 T_{21} - \theta P_1 - \tau_{21} Q_{21} < 0,\ \tau_{21} > 0, \tag{14a}$$

$$T_{32}^T P_3 T_{32} - \theta P_2 - \tau_{32} Q_{32} < 0,\ \tau_{32} > 0, \tag{14b}$$

$$T_{52}^T P_5 T_{52} - \theta P_2 - \tau_{52} Q_{52} < 0,\ \tau_{52} > 0, \tag{14c}$$

while (13) becomes:

$$T_{21}^T T_{32}^T P_3 T_{32} T_{21} - \theta P_1 - \tau_{21} Q_{21} - \tau_{32} T_{21}^T Q_{32} T_{21} < 0,\ \tau_{21} > 0,\ \tau_{32} > 0, \tag{15a}$$

$$T_{21}^T T_{52}^T P_5 T_{52} T_{21} - \theta P_1 - \tau_{21} Q_{21} - \tau_{52} T_{21}^T Q_{52} T_{21} < 0,\ \tau_{21} > 0,\ \tau_{52} > 0. \tag{15b}$$

It is not difficult to see that if P_1, P_2, P_3, P_5, τ_{21}, τ_{32}, τ_{52} are a feasible solution to the set of LMIs in (14), then P_1, θP_3, θP_5, τ_{21}, $\theta\tau_{32}$, $\theta\tau_{52}$ are a feasible solution to the set of LMIs is (15): To obtain (15a) from (14), multiply (14b) on both sides by $T_{21}^T/\sqrt{\theta}$ and $T_{21}\sqrt{\theta}$, and add it to (14a). Inequality (15b) can be obtained similarly. We have shown for a special case, that computation of the Lyapunov invariants on the reduced graph is not more difficult than the original graph. The result is true in general, and the same type of relaxations that make convex optimization of the Lyapunov invariants feasible on the original graph, are applicable to the reduced graph. In light of Theorem (3), the conclusion of the above discussion is that analysis of the reduced models are always beneficial, regardless of the convex relaxations that are used at the optimization phase.

3 Conclusions and Future Work

Concepts and tools from control and optimization can be exploited to build a framework for software verification. This framework is particularly appealing for analysis of safety-critical software in embedded systems. Our framework consists of the following four procedures: 1. Model the software as a dynamical system. 2. Given the functional and the safety specifications, formulate Lyapunov invariants whose existence prove the desired properties of the model and hence, the program

itself. 3. Apply convex relaxations such that finding the proposed invariants can be formulated as a convex optimization problem. 4. Use the relevant numerical optimization tools, e.g. semidefinite programming, to compute these invariants. In this paper, we focused on graph models of software and further developed the framework on such models. Some future works include in-depth study of optimality of graph models in the reduction process and improving scalability. Computing the minimal and maximal degrees, and the corresponding realizations of a computer program are interesting problems that arise in this context.

References

1. S. Boyd, L.E. Ghaoui, E. Feron, and V. Balakrishnan. Linear Matrix Inequalities in Systems and Control Theory. Society for Industrial and Applied Math., 1994.
2. A. Bemporad, D. Mignone, M. Morari. Moving horizon estimation for hybrid systems and fault detection. In Proc. American Control Conf., 1999, pp. 2471−2475.
3. D. Bertsimas, J. Tsitsikilis. Introduction to Linear Optimization. Athena Scientific, 1997.
4. M. S. Branicky, V. S. Borkar, S. K. Mitter. A unified framework for hybrid control: model and optimal control theory. IEEE Trans. Automatic Control, 43(1):31-45, 1998.
5. E. M. Clarke, E. A. Emerson, A. P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. ACM Trans. on Programming Languages and Systems, 8(2): 244 - 263, 1986.
6. E. M. Clarke, O. Grumberg, D. A. Peled. Model Checking, MIT Press, 1999.
7. P. Cousot, R. Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In Proc. 4th Symposium on Principles of Programming Languages, pp. 238–252, 1977.
8. S. Prajna, A. Jadbabaie, G. Pappas. A Framework for Worst-Case and Stochastic Safety Verification Using Barrier Certificates. IEEE Trans. on Automatic Control, 52(8): 1415 - 1428, 2007.
9. M. Johansson, A. Rantzer. Computation of piecewise quadratic Lyapunov functions for hybrid systems. IEEE Trans. on Automatic Control, 43(4):555-559, 1998.
10. R. Alur, G. J. Pappas (Eds.): Hybrid Systems: Computation and Control, Lecture Notes in Computer Science, v. 2993, Springer Verlag, March 2004.
11. G. Lafferriere, G. J. Pappas, S. Sastry. Hybrid systems with finite bisimulations. Hybrid Systems V, Lecture Notes in Computer Science, v. 1567, Springer 1999.
12. J. Lygeros, C. Tomlin, S. Sastry. Controllers for reachability specifications for hybrid systems. Automatica, 35(3):349-370, 1999.
13. D. A. Peled. Software Reliability Methods. Springer-Verlag, New York, NY, 2001.
14. P. A. Parrilo. Minimizing Polynomial Functions. In Algorithmic and Quantitative Real Algebraic Geometry, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Vol. 60, pp. 83–99, AMS.
15. M. Roozbehani, É. Feron & A. Megrestki. Modeling, Optimization and Computation for Software Verification. In M. Morari & L. Thiele (Eds.): Hybrid Systems: Computation and Control, Lecture Notes in CS 3414, Springer 2005, pp. 606–622.
16. M. Roozbehani, A. Megretski, E. Feron. Convex optimization proves software correctness. In Proc. American Control Conf., 2005, pp. 1395 - 1400.
17. M. Roozbehani, A. Megretski, E. Feron. Optimization of Lyapunov Invariants for Certification of Software Systems. Submitted, IEEE Trans. Automatic Control, 2007.