On Generating Safe Controllers for Discrete-Time Linear Systems

Adam Cataldo

EECS 290N Report
December 17, 2004

University of California at Berkeley
Berkeley, CA, 94720, USA

acataldo@eecs. berkeley.edu

Abstract

In this paper I describe a implementation procedure for generating controllers to guarantee safety constraints for controllable discrete-time linear systems. The first two sections are a summary of the work of [6]. The contribution of this paper is the discussion of the limitations of this method and future research directions. We omit any discussion of complexity and focus instead on functionality questions.

1 Introduction

The classic control system problem is to design a controller which drives some error function to zero as time goes to infinity. Finding a controller of this type is called the stability problem in control systems. These controllers have had huge impact in many engineering disciplines including circuits, mechanical systems, and chemical process control. Solving the stability problem does not let us say anything about the transient behavior of a system. In many cases, this is inadequate. We may for example wish to guarantee that a robot never collides with an obstacle at any time or that the change in concentration of a chemical is bounded by a certain amount at each time. As computers become embedded in more physical systems, computational methods for designing such controllers will become increasingly relevant.

In [6], Tabuada and Pappas give a method for generating such controllers for controllable discrete-time linear systems. We focus almost exclusively on extending the results of this paper, because this paper, along with a nonlinear extension in [5], are the only papers (to the author's knowledge) that give methods for generating safe control laws on discrete-time control systems over a broad class of safety constraints.

Let $x : \mathbb{N}_0 \to \mathbb{R}^n$ describe the evolution of a system's state with respect to time and $u : \mathbb{N}_0 \to \mathbb{R}^m$ describe the time evolution of the control input. The design problem is to chose a control input such that the system's state follows a "safe" evolution.. The system is discrete-time, linear, and time-invariant if there exist matrices $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times m}$ such that $\forall k \in \mathbb{N}_0$,

$$x(k+1) = Ax(k) + Bu(k). \tag{1}$$

The system is controllable if for any initial state $x(0) \in \mathbb{R}^n$, for any time $k \geq n$, and for any desired state $p \in \mathbb{R}^n$, there exists a $u : \mathbb{N}_0 \to \mathbb{R}^m$, such that the solution to Equation 1 satisfies

$$x(k) = p. \tag{2}$$

See [2] for an introduction to controllability.

A feedback control law is a map $c : \mathbb{R}^n \to \mathbb{R}^m$. If the state at time t is $x(t)$, we will apply $u(t) = c\big(x(t)\big)$. Then the next state will only depend on the current state. This feedback connection is always well defined [4].

For the system in Equation 1, there is transition system over which our safety properties are defined. In our context, a transition system is a tuple $S = (Q, Q^0, V, f, h)$, where:

- Q is a set of states,
- $Q^0 \subseteq Q$ is a set of initial states,
- V is a set of values,
- $f : Q \to \mathcal{S}(Q)$ is the state-update map assigning the set of possible next states $f(q)$ to each $q \in Q$,

- $h : Q \to V$ is a map assigning value $h(q)$ tob each $q \in Q$.

Thought of as a concurrent model of computation, the set of tags is $T = \mathbb{N}_0$, and the values are V.

In our case $Q = \mathbb{R}^n$, $Q^0 \subset \mathbb{R}^n$, and

$$q_2 \in f(q_2) \Leftrightarrow \exists p \in \mathbb{R}^m, q_2 = Aq_1 + Bp. \tag{3}$$

That is q_1 can take a transition to q_2 if there is a control input p which will push state q_1 to q_2 in one step. We will later define V and h for our example.

A (initialized) run of a transition system is a finite or infinite sequence $r = (r_0, r_1, \ldots) \in Q^{**}$ where $r_0 \in Q^0$ and $r_i \to r_{i+1}$. In our case, given any $u : \mathbb{N}_0 \to \mathbb{R}^m$ and any initial state $x(0) \in Q^0$ the solution $\big(x(0), x(1), \ldots\big)$ to Equation 1 is a run. The language $L(S)$ of the transition system is the set of all possible finite observation sequences, that is

$$L(S) = \left\{ \bar{o} \in V^* \;\middle|\; \begin{array}{c} \exists \text{ a run } r \in Q^* \text{ of } S, \\ \forall t \leq \text{length}(r), \\ \bar{o}_t = h(r_t) \end{array} \right\}. \tag{4}$$

We similarly define the ω-language $L^\omega(S)$ as the set of all possible infinite observation sequences. A transition system output can be thought of as a sequence in the language or the ω-language. An input can be thought of a decision maker which chooses which state transition in $f(q)$ to make at each time t and each state r_t, although the decision making procedure is not included in the model.

Given two transition systems S_1 and S_2 with the same value set V, their parallel composition $S_1 \parallel S_2 = (Q, Q^0, V, f, h)$ is defined by:

- $Q = \big\{(q_1, q_2) \in Q_1 \times Q_2 \big| h_1(q_1) = h_2(q_2)\big\}$,
- $Q^0 = \big\{(q_1, q_2) \in Q_1^0 \times Q_2^0 \big| h_1(q_1) = h_2(q_2)\big\}$,
- $(p_1, p_2) \in f(q_1, q_2) \Leftrightarrow p_1 \in f_1(q_1)$ and $p_2 \in f(q_2)$,
- $h(q_1, q_2) = h_1(q_1) = h_2(q_2)$.

Note that $L(S_1 \parallel S_2) = L(S_1) \cap L(S_2)$. Thus parallel composition can be thought of as requiring the outputs of the transition systems to be equal. $S_1 = (Q_1, Q_1^0, V, f_1, h_1)$ and $S_2 = (Q_2, Q_2^0, V, f_2, h_2)$ are bisimilar with the same value V space are bisimilar if $L(S_1) = L(S_2)$.

1.1 Linear Temporal Logic

Let P be a set of predicates over the values V for a transition system. Each $p \in P$ is a map from V to $\mathbb{B} = \{\text{true}, \text{false}\}$. We let **true** : $V \to \mathbb{B}$ be the constant function with **true**$(v) =$ true and **false** : $V \to \mathbb{B}$ be the constant function with **false**$(v) =$ false.

Let $\mathcal{P}(P)$ be the powerset of P. Given an infinite word $w = (w_0, w_1, \ldots) \in L^\omega(S)$, define a string $s : \mathbb{N}_0 \to \mathcal{P}(P)$ as follows:

$$p \in s(t) \Leftrightarrow p(w_t) = \text{true}. \tag{5}$$

Now define LTL formulae as:

- **true**, **false** are LTL formulae
- Any $p \in P$ is an LTL formula.
- If φ_1 and φ_2 are LTL formulae, then $\varphi_1 \wedge \varphi_2$, $\neg\varphi_1$, $\bigcirc\varphi_1$, and $\varphi_1 \mathbf{U} \varphi_2$ are LTL formulae.

Given $s : \mathbb{N}_0 \to \mathcal{P}(P)$, we say s satisfies LTL formula φ at time t, $s(t) \vDash \varphi$, under the following conditions, where $p \in P$ is a predicate and φ_1 and φ_2 are LTL formula:

- $s(t)$ satisfies p:
  $s(t) \vDash p \Leftrightarrow p \in s(t)$,
- $s(t)$ does not satisfy φ_1:
  $s(t) \vDash \neg\varphi_1 \Leftrightarrow s(t) \nvDash \varphi_1$,
- $s(t)$ satisfies φ_1 and φ_2:
  $s(t) \vDash \varphi_1 \wedge \varphi_2 \Leftrightarrow \big(s(t) \vDash \varphi_1\big) \wedge \big(s(t) \vDash \varphi_2\big)$,
- $s(t)$ satisfies φ_1 next:
  $s(t) \vDash \bigcirc\varphi_1 \Leftrightarrow s(t+1) \vDash \varphi$,
- $s(t)$ satisfies φ_1 until φ_2:
  $s(t) \vDash \varphi_1 \mathbf{U} \varphi_2 \Leftrightarrow$
  $\exists N \in \mathbb{N}_0, \big(\forall k < N, s(t+k) \vDash \varphi_1\big) \wedge \big(s(t+N) \vDash \varphi_2\big)$.

By convention, $s(t) \vDash$ **true** and $s(t) \nvDash$ **false**.

In our case, the strings will correspond to properties that x will satisfy as it varies in time. For example, the state always remains in some set $G \subset \mathbb{R}^n$ is equivalent to LTL formula

$$\neg\Big(\textbf{true}\ \mathbf{U} \neg \text{inG}\Big),$$

where inG is a predicate corresponding to the state $q \in G$. Here our value $v \in V$ would indicate whether or not the state is in G or not.

A Büchi automata is a tuple $A = (Q, Q^0, V, f, h, F)$ where $S = (Q, Q^0, V, f, h)$ is a transition system and $F \subset Q$ is a set of accepting states. A Büchi automata accepts only infinite sequences in its behavior. An initialized run $r = (r_0, r_1, \ldots) \in Q^\omega$ is accepted by A if there are infinitely many $i \in \mathbb{N}_0$ with $r_i \in F$. We can convert any LTL constraints over a finite observation set V into a Büchi automata which only accepts runs satisfying our LTL constraints. See [7] for a nice explanation of how to do this. By composing the resulting transition system with our original system, we restrict the behaviors of the original system to satisfy the LTL constraint.

1.2 Semilinear Sets over the Observation Space

For a controllable discrete-time linear system, a particular m-dimensional subspace O of $\mathbb{R}^n$, has a nice property for defining safety properties over. Let b_i denote the i^{th} column of B. Let

$$\mu_i = \min \left\{ k < n \left| \begin{matrix} \text{span}\{b_i, Ab_i, \ldots, A^{n-1}b_i\} = \\ \text{span}\{b_i, Ab_i, \ldots, A^{k}b_i\} \end{matrix} \right. \right\} \quad (6)$$

Then we define O as

$$O = \text{span}\{A^{\mu_1}b_1, A^{\mu_2}b_2, \ldots, A^{\mu_m}b_m\}. \quad (7)$$

We call this the observation space. From [4], we know that this space will in fact be m-dimensional. Let $\pi_O : \mathbb{R}^n \rightarrow O$ give the projection of a point onto this subspace. The importance of this subspace is that given any desired state trajectory $x_d : \mathbb{R} \rightarrow \mathbb{R}^n$ and any initial condition, there exists a control input such that $\pi_O\big(x(t)\big)\pi_O\big(x_d(t)\big)$ for all

$$t > \max\{\mu_1, \ldots, \mu_m\}.$$

A set is semilinear if it can be described through unions, intersections, and complements of the following primitive sets:

$$\big\{x \in \mathbb{R}^n \big| f^T x + c \sim 0\big\}, \quad (8)$$

where $f \in \mathbb{Q}^n$, $c \in \mathbb{Q}$, and $\sim \in \{=, >\}$. We use rational numbers here instead of real numbers, so that we can always perform exact numerical computations and not be subject to roundoff error. This does not limit us in any practical sense, since any real number can be approximated arbitrarily close by a rational number. Thus, we will also assume that $A \in \mathbb{Q}^{n\times n}$ and $B \in \mathbb{Q}^{n\times m}$.

Suppose we partition O into a finite number of semilnear sets. We let our values V be the collection of these sets. We let $\pi_V : O \rightarrow V$ be the projection operator that projects each point in V into its observation. Then we can let our transition system be $S = (\mathbb{R}^n, Q^0, V, f, \pi_V \circ \pi_O)$. The main result of [6] is that for any linear temporal logic constraints over this transition system, we can always compute a set of control laws that guarantee our desired LTL property or decide that no such control law exists. These LTL properties will be encodings of safety properties, such as collision avoidance, so our control laws will be able to guarantee or safety properties.

2 Implementation

As of the writing of this paper, I have began a Ptolemy II [1] implementation of the procedure to generate safe control laws for discrete-time controllable linear systems. The idea is to have a Ptolemy actor which takes the state as input and produces a control input as output which guarantees safety of the system. The parameters to this actor would include the A and B matrix as well as the observation space O and the desired LTL constraints that guarantee safety.

After the parameters are given, the control actor would refine itself to generate the control law $c : \mathbb{R}^n \rightarrow \mathbb{R}^m$. Since the method of [6] only gives a method for generating a set of possible control laws, some decision would have to be made as to what control input to chose (minimum norm, minimum change in value, etc.).

Our first step in computing the control law is to compute a transition system S_Σ which is bisimilar to our discrete-time control system with only a finite number of states. [6] gives us the bisimulation algorithm which computes S_Σ.

To compute our controller, we must compute a Büchi automata A_C which accepts our LTL formula. The next step for our computation is to compute the parallel composition of the corresponding transition system S_C with S_Σ. This allows only behaviors which satisfy the LTL formula. [3] gives an algorithm for computing such a controlled system which allows the maximum number of possible behaviors. This composition can be thought of as disabling possible transitions in S_Σ.

Each state in S_Σ corresponds to some subset of $\mathbb{R}^n$. For each disabled transition emanating from a given state, there will be a restriction on the allowable values of the control input $c(p)$ for p in the corresponding subset of $\mathbb{R}^n$.

If a transition controller exists in this case, [6] gives us a method to compute valid control laws. The control laws will be described through semilinear sets which depend on the states. There will be a finite number of possible control laws. As was mentioned earlier, we are left with a choice in deciding which particular control law to use.

3 Limitations

This method has some limitations which make it impractical for many applications. Many real-world systems cannot be modelled as linear. Extensions of this method to nonlinear systems has been addressed in [5]. Even for linear systems, this method has some severe restrictions.

3.1 Dimension

Many control systems have a state space dimension n which is larger than the number of inputs m. Using this method, we only have control over the projection of our state on an m dimensional subspace of our state space, so we are limited in the type of constraints we can impose on our system.

We may be able to transform the observation space through an equivalent feedback system. That is given some $K \in R^{m\times n}$, we can let our control input be given by

$u(k) = Kx(k) + v(k)$, and v becomes our new control input. Then

$$x(k+1) = (A + BK)x(k) + v(k). \tag{9}$$

As an example from [6], if

$$A = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, B = \begin{pmatrix} 1/2 \\ 1 \end{pmatrix}, \text{ then } V = \begin{pmatrix} 3/2 \\ 1 \end{pmatrix}. \tag{10}$$

If we apply $K = (0 \ -1)$, then our new V becomes $(1\ 0)^T$

An interesting question is if it is possible to find a K which transforms V into an arbitrary m dimensional subspace of $\mathbb{R}^n$. Even if we can, we are always limited to m dimensional constraints.

3.2 Bounded Control Inputs

In many systems, there are physical limitations to the amount of control we can apply at each time. We have assumed that we can apply any control input $u : \mathbb{N}_0 \to \mathbb{R}^m$, but in practice we can only apply $u : \mathbb{N}_0 \to U$, where U is some bounded subset of $\mathbb{R}^m$. What if we can describe U through semilinear sets? Are we still able to find safe control laws? The answer is in general no.

Suppose we have the following control system:

$$x(k+1) = Ax(k) + Bu(k) = \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix} x(k) + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u(k). \tag{11}$$

Suppose we can describe U through a semilnear subset of $\mathbb{R}^m$. We might try and transfer these constraints to the state by adding states to measure the control input. We define our new system as

$$\begin{pmatrix} x(k+1) \\ y(k+1) \end{pmatrix} = \begin{pmatrix} A & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} x(k) \\ y(k) \end{pmatrix} + \begin{pmatrix} B \\ I \end{pmatrix} u(k) \tag{12}$$

The problem here is that this system may no longer be controllable, so we cannot use the results of [6] to find a bisimilar transition system with finite state space representation.

In some cases we may not lose controllability using this technique. Consider again the case where

$$x(k+1) = x(k) + u(k). \tag{13}$$

We can then define

$$\begin{pmatrix} x(k+1) \\ y(k+1) \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} x(k) \\ y(k) \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \end{pmatrix} u(k) \tag{14}$$

We still only have control over one dimension, since $m = 1$, so if our constraints on the state x conflict with our bounds on the control input, we have no way of finding a finite bisimilar system.

4 Conclusions

We can automate the process of generating safe controllers for controllable discrete-time linear systems using the method of [6]. We are trying to implement this in Ptolemy II. There are some fundamental limits to this method. First, we are limited in the dimension of constraints we can apply on the states. Second, we have no way of handling constraints on the control input. The method of [6] relies on being able to transform our control system into a transition system with a finite representation. It remains to be seen if we can find a finite representation of systems with more general constraints.

References

[1] C. Brooks, E. Lee, X. Liu, S. Neuendorffer, Y. Zhao, and H. Zheng. Heterogeneous concurrent modeling and design in java. Memorandum UCB/ERL M04/27, University of California, Berkeley, July 2004.

[2] F. M. Callier and C. A. Desoer. *Linear System Theory*. Springer Verlag, 1991.

[3] P. Ramadge. Some tractable supervisory control problems for discrete-event systems modeled by Büchi automata. *IEEE Transactions on Automatic Control*, 34(1):10–19, January 1989.

[4] E. Sontag. *Mathematical Control Theory*. Springer-Verlag, 2 edition, 1998.

[5] P. Tabuada. Nonlinear flat systems admit flat bisimulations. *Submitted for publication*, 2004. available at http://www.nd.edu/ ptabuada/papers/FinFlat.pdf.

[6] P. Tabuada and G. Pappas. Linear time logic control of linear systemsb. *IEEE Transactions on Automatic Control*,, Submitted 2004.

[7] P. Wolper. Constructing automata from temporal logic formulas: A tutorial. In *Lectures on Formal Methods and Performance Analysis : First EEF/Euro Summer School on Trends in Computer Science*, volume 2090 of *Lecture Notes in Computer Science*. Springer-Verlag, 2001.