BISIMULATIONS OF DISCRETE, CONTINUOUS AND HYBRID SYSTEMS

George J. Pappas
University of Pennsylvania, Philadelphia, PA 19102. USA.

Keywords: Transition systems, linear temporal logic, verification, reachability, model checking, language equivalence, bisimulation, hybrid systems, decidability

Contents

Summary

Hybrid systems unify discrete-event and continuous-time dynamics in a manner that enables the modeling of software control or embedded systems. The complexity of hybrid systems naturally requires the use of algorithmic approaches for analysis and design. For analysis and design purposes, it is often useful to abstract a system in a way that preserves the desired properties while hiding modeling details that are of no interest. This is achieved using state space partitions, such as bisimulations, that preserve reachability properties as well as a finite set of propositions. In this chapter, we survey notions of bisimulations for discrete, continuous, and hybrid systems. We show that interesting classes of hybrid systems can be abstracted to purely discrete systems while preserving all properties that are definable in linear temporal logic.

1. Introduction

Hybrid systems combine both digital and analog components, in a way that is useful for the analysis and design of embedded control systems. Hybrid systems have been used as mathematical models for many applications such as automated highway systems, air

traffic management systems, embedded automotive controllers, and chemical processes. Their wide applicability has inspired research from both control theory and theoretical computer science.

The complexity of hybrid system models requires the development of a formal theory of modeling abstractions. In the literature, the notions of *abstraction* or *aggregation* refer to grouping the system states into equivalence classes. Depending on the grouping of the original state space, we may have *discrete*, or *continuous* abstractions. With this notion of abstraction, the abstracted system will be defined as the induced quotient dynamics. Theoretical computer science, and, in particular, the areas of concurrency theory, and computer aided verification have established formal notions of abstraction and model refinement which are used to tackle the state explosion arising in purely discrete systems. Given a discrete system, an *abstraction* is a quotient system that preserves some properties of interest while ignoring detail. Properties of interest include reachability, safety, liveness, and other properties expressible in various temporal logics.

The notion of bisimulation is one such formal notion of abstraction that has been used for reducing the complexity of finite state systems such as labeled transition systems. Bisimulations are partitions of the state space that preserve observations and reachability properties. In addition to reachability, bisimulations of finite transition systems preserve all properties that are expressible in temporal logics such as linear temporal logic (LTL). The notion of bisimulation has been also instrumental in obtaining decidability results for various classes of hybrid systems, by considering finite bisimulations of hybrid systems. In the control community, notions that are similar to bisimulation have been considered in the hierarchical, supervisory control of discrete event systems, and hybrid systems (see survey). Furthermore, bisimulations have also been used as a controller synthesis tool for discrete-event systems.

In this chapter, we survey the notion of bisimulation in the context of purely discrete, purely continuous, and, eventually, hybrid systems. We first review the well established notion of bisimulation for transition systems along with linear temporal logic, a popular logic for specifying properties of transitions systems. We then apply the framework of bisimilar transition systems to transition systems that are generated by linear control systems. Given a continuous-time linear system, and a finite number of propositions, we characterize linear quotient maps that result in quotient transition systems that are bisimilar to the original system. We show that computing the coarsest bisimulation, which results in maximum complexity reduction, corresponds to computing the maximal reachability invariant subspace inside a subspace which is related with the propositions.

We then focus on finite bisimulations of hybrid systems. Given a hybrid system and some desired property, one extracts a finite, discrete system while preserving all properties of interest. This is achieved by constructing suitable, *finite* and *computable* partitions of the state space of the hybrid system. By obtaining discrete abstractions which are finite, and preserve properties of interest, analysis can be equivalently performed on the finite system, which requires only a finite number of steps. Checking the desired property on the abstracted system should be either *equivalent to* or *sufficient*

for checking the property on the original system. In this chapter, we focus on *equivalent discrete abstractions of hybrid systems* along with some properties they preserve. We show that there are many interesting classes of hybrid systems which can be abstracted by finite systems for analysis purposes.

2. Bisimulations of Transition Systems

Transition systems are graph models, possibly with an infinite number of states or transitions.

Definition 2.1 (Transition Systems) *A transition system* $T = (Q, \Pi, \rightarrow, \models, Q_0)$ *consists of:*

- *A* (*possibly infinite*) *set Q of states,*
- *A finite alphabet* Π *of propositions,*
- *A transition relation* $\rightarrow \subseteq Q \times Q$ *, and*
- *A satisfaction relation* $\models \subseteq Q \times \Pi$ *, and*
- *A set* $Q_0 \subseteq Q$ *of initial states.*

A state q_1 is *predecessor* of a state q_2, and q_2 is a *successor* of q_1, written $q_1 \rightarrow q_2$, if the transition relation $\rightarrow$ contains the pair (q_1, q_2). A state q *satisfies* a proposition π, written $q \models \pi$, if the satisfaction relation $\models$ contains the pair (q, π). Given a proposition $\pi \in \Pi$, we write $[\![\pi]\!] = \{q \in Q \mid q \models \pi\}$ for the set of states that satisfy π.

The transition system *T* is *finite* if the cardinality of *Q* is finite, and it is infinite otherwise. A *region* is a subset $P \subseteq Q$ of the states. The sets of predecessor and successor states of *P* respectively are

$$\mathrm{Pre}(P) = \{q \in Q \mid \exists p \in P.\ q \rightarrow p\} \quad (1)$$

$$\mathrm{Post}(P) = \{q \in Q \mid \exists p \in P.\ p \rightarrow q\} \quad (2)$$

The set of states that are accessible from *P* in two transitions is Post(Post(*P*)), and is denoted $\mathrm{Post}^2(P)$. In general, $\mathrm{Post}^i(P)$ consists of the states that are accessible from *P* in *i* transitions. $\mathrm{Pre}^i(P)$ is defined similarly. Then

$$\mathrm{Pre}^*(P) = \bigcup_{i \in \mathbb{N}} \mathrm{Pre}^i(P) \quad (3)$$

$$\mathrm{Post}^*(P) = \bigcup_{i \in \mathbb{N}} \mathrm{Post}^i(P) \quad (4)$$

are the set of states that are *backward* and *forward reachable* from *P*, that is, accessible in any number of transitions. In particular, $\mathrm{Post}^*(Q_0)$ is the set of *reachable states* of the transition system *T*, and is denoted by $\mathrm{Reach}(T)$. A problem that is of great interest

for transition systems is the reachability problem.

Problem 2.2 (Reachability Problem) *Given a transition system* $T = (Q, \Pi, \rightarrow, \models, Q_0)$ *and a proposition* $\pi \in \Pi$, *is* $Reach(T) \cap [\![\pi]\!] \neq \emptyset$ *?*

If the proposition π encodes an undesirable or unsafe region of the state space, then solving reachability corresponds to checking if the system is safe. In this chapter, we are interested in computational approaches to the solution of the reachability problem. The following algorithm computes the reachable space until either a state satisfying π is reached, or no more reachable states can be added.

Algorithm 1 (Forward Reachability Algorithm)

initially $R := Q_0$;

while *true* **do**

if $R \cap [\![\pi]\!] \neq \emptyset$ **then return** "unsafe" **end if;**

if $\text{Post}(R) \subseteq R$ **then return** "safe" **end if;**

$R := R \cup \text{Post}(R)$

end while

A backward reachability algorithm which starts with $[\![\pi]\!]$ and checks whether $\text{Pre}^*([\![\pi]\!]) \cap Q_0 \neq \emptyset$ can be similarly constructed. Such iterative algorithmic approaches to checking system properties are guaranteed to terminate if the state space of the transition system is finite. If the state space is infinite, then there is, in general, no guarantee that the forward reachability algorithm will terminate within a finite number of iterations of the loop. It could continue adding states forever without ever reaching the target region $[\![\pi]\!]$ or reaching a fixed point R such that $\text{Post}(R) \subseteq R$. A practically useful goal is to find classes of infinite transition systems whose analysis can be performed on *equivalent* but *finite* transition systems. This is accomplished by constructing suitable finite quotients of the original system which partition the state space in a manner that preserves the properties of interest while omitting modeling detail.

In addition to reachability, the desired system specification may require more detailed system properties. For example, one may wish to encode the requirement that a system failure is eventually followed by a return to the normal mode of operation. More abstractly, if the transition system visits a region P_1, encoding a failure, then eventually it will reach a region P_2, encoding normal operation. Such properties can be encoded as formulas in temporal logic. Formulas of temporal logic are thus used to formally specify

properties of systems, such as reachability, invariance, or response properties. In the sequel, after defining the notion of quotient transition systems, two kinds of equivalence relations, *language equivalences* and *bisimulations*, are considered along with the popular temporal logic Linear Temporal Logic (LTL) whose properties they preserve.

An equivalence relation $\sim \subseteq Q \times Q$ on the state space is *proposition preserving* if for all states $p, q \in Q$ and all propositions $\pi \in \Pi$, if $p \sim q$ and $p \models \pi$, then $q \models \pi$; that is, the region $[\![\pi]\!]$ is a union of equivalence classes. Given a proposition-preserving equivalence relation $\sim$, the definition of *quotient transition system* $T/\sim$ is natural. Let $Q/\sim$ denote the quotient space, that is, the set of equivalence classes. For a region *P*, we denote by $P/\sim$ the collection of all equivalence classes which intersect *P*. The transition relation $\rightarrow_\sim$ on the quotient space is defined as follows: for $P_1, P_2 \in Q/\sim$, we have $P_1 \rightarrow_\sim P_2$ if and only if (iff) there exist two states $q_1 \in P_1$ and $q_2 \in P_2$ such that $q_1 \rightarrow q_2$. The satisfaction relation $\models\sim$ on the quotient space is defined as follows: for $P \in Q/\sim$, we have $P \models\sim \pi$ iff there exists a state $q \in P$ such that $q \models \pi$. The quotient transition system is then $T/\sim = (Q/\sim, \Pi, \rightarrow_\sim, \models_\sim, Q_0/\sim)$.

2.1. Language Equivalence and Linear Temporal Logic

Let $q \in Q$ be a state of the transition system $T = (Q, \Pi, \rightarrow, \models, Q_0)$. Given a state $q \in Q$, let $\Pi_q = \{\pi \in \Pi \mid q \models \pi\}$ be the set of propositions that are satisfied by *q*. A *trajectory* generated from *q* is an infinite sequence $q_0 q_1 q_2 \ldots$ such that $q_0 = q$ and for all $i \in \mathbb{N}$, we have $q_i \rightarrow q_{i+1}$. This trajectory defines the *word* $\Pi_{q0}\Pi_{q1}\Pi_{q2}\ldots$ The set of words that are defined by trajectories generated from *q* is denoted by *L*(*q*), and called the *language* of the state *q*. The set $\bigcup_{q \in Q_0} L(q)$ of words that are defined by trajectories generated from initial stats is denoted by $L(T)$, and called the *language* of the transition system *T*.

Definition 2.3 (Language Equivalences) *Let T be a transition system with state space Q. An equivalence relation* $\sim_L$ *on Q is a language equivalence of T if for all states* $p, q \in Q$, *if* $p \sim_L q$, *then* $L(p) = L(q)$.

Note the every language equivalence is proposition preserving. Every language equivalence $\sim_L$ partitions the state space and gives rise to the quotient transition system $T/\sim_L$, which is called a *language equivalence quotient* of *T*. The formulas of Linear Temporal Logic (LTL) are interpreted over words, and hence the properties expressed in LTL are preserved by language equivalence quotients.

Definition 2.4 (Linear Temporal Logic) *The formulas of Linear Temporal Logic* (*LTL*) *are defined inductively as follows:*

- **Propositions** *Every proposition* π *is a formula*.

- **Formulas** *If* ϕ_1 *and* ϕ_2 *are formulas, then the following are also formulas:*

$\phi_1 \vee \phi_2 \qquad \neg\phi_1 \qquad \bigcirc\phi_1 \qquad \phi_1 \mathcal{U} \phi_2$

The formulas of LTL are interpreted over infinite sequences of sets of propositions. Consider a word $w = \Pi_0\Pi_1\Pi_2...$, where each Π_i is a set of propositions. The satisfaction of a proposition π at position $i \in \mathbb{N}$ of word *w* is denoted by $(w,i) \models_L \pi$ (which should not be confused with the satisfaction relation $\models$ which tells us whether a state satisfies a proposition), and holds iff $\pi \in \Pi_i$. We can then recursively define the semantics for any LTL formula as follows:

- $(w,i) \models_L \phi_1 \vee \phi_2$ if either $(w,i) \models_L \phi_1$ or $(w,i) \models_L \phi_2$
- $(w,i) \models_L \neg\phi_1$ if $(w,i) \not\models_L \phi_1$
- $(w,i) \models_L \bigcirc\phi_1$ if $(w,i+1) \models \phi_1$
- $(w,i) \models_L \phi_1 \mathcal{U} \phi_2$ if there is a $j \geq i$ such that $(w, j) \models_L \phi_2$ and for all $i \leq k < j$, we have $(w,k) \models_L \phi_1$

A word *w satisfies* an LTL formula ϕ if $(w,0) \models_L \phi$. From $\neg$ and $\vee$, which stand for negation and disjunction, respectively, we can also define conjunction $\wedge$, implication $\Rightarrow$, and equivalence $\Leftrightarrow$. The *temporal operators* $\bigcirc$ and $\mathcal{U}$ are called the *next* and *until* operators. The $\bigcirc\phi_1$ formula holds for a word $\Pi_0\Pi_1\Pi_2...$ iff the sub-formula ϕ_1 is true for the suffix $\Pi_1\Pi_2...$ The formula $\phi_1 \mathcal{U} \phi_2$ intuitively expresses the property that ϕ_1 is true until ϕ_2 becomes true. Using the next and until operators, we can also define the following temporal operators in LTL:

- Eventually: $\Diamond\phi = true\ \mathcal{U}\phi$
- Always: $\Box\phi = \neg\Diamond\neg\phi$

Therefore, $\Diamond\phi$ indicates that ϕ becomes eventually true, whereas $\Box\phi$ indicates that ϕ is true at all positions of a word. The LTL formula $\Box\Diamond\phi$ is true for words that satisfy ϕ infinitely often, whereas a word satisfies $\Diamond\Box\phi$ if ϕ becomes eventually true and then stays true forever.

A transition system *T satisfies* an LTL formula ϕ if all words in the language *L*(*T*) satisfy ϕ. For example, if π is a proposition encoding an unsafe region, then safety can be simply expressed as $\Box\neg\pi$, or equivalently, as $\neg\Diamond\pi$. The more elaborate requirement that visiting region $[\![\pi_1]\!]$ will eventually be followed by visiting region $[\![\pi_2]\!]$, is expressed by the formula $\Box(\pi_1 \Rightarrow \Diamond\pi_2)$.

Problem 2.5 (LTL Model Checking Problem) *Given a transition system T and an LTL formula ϕ, determine if T satisfies ϕ.*

Since reachability can be expressed by an LTL formula of the form $\Diamond\pi$, it is immediate that Problem 2.2 is contained in Problem 2.5. Given the definition of language equivalence, the following theorem should come as no surprise.

Theorem 2.6 (Language equivalences preserve LTL properties) *Let T be a transition system and let $\sim_L$ be a language equivalence of T. Then T satisfies the LTL formula ϕ iff the language equivalence quotient $T/\sim_L$ satisfies ϕ.*

Therefore, given a transition system *T* and an LTL formula ϕ, we can equivalently perform the model checking problem on $T/\sim_L$. In general, language equivalence quotients are not finite. If, however, we are *given* a finite language equivalence quotient of a transition system *T*, then using the above theorem, LTL model checking can be decided for *T*.

2.2. Bisimulation Partitions

We now define a different way of partitioning the state space which has computational advantages over language equivalence.

Definition 2.7 (Bisimulation) *Let $T = (Q, \Pi, \rightarrow, \models, Q_0)$ be a transition system. A proposition-preserving equivalence relation $\sim_B$ on Q is a bisimulation of T if for all states $p, q \in Q$, if $p \sim_B q$, then for all states $p' \in Q$, if $p \rightarrow p'$, then there exists a state $q' \in Q$ such that $q \rightarrow q'$ and $p' \sim_B q'$.*

If $\sim_B$ is a bisimulation, then the quotient transition system $T/\sim_B$ are called a *bisimulation quotient* of *T*. The crucial property of bisimulations is that for every equivalence class $P \in Q/\sim_B$, the predecessor region *Pre(P)* is a union of equivalence classes. Therefore, if $P_1, P_2 \in Q/\sim_B$, then $Pre(P_1) \cap P_2$ is either the empty set or all of P_2. It is not difficult to check that every bisimulation is a language equivalence, but a language equivalence is not necessarily a bisimulation. Therefore, LTL model checking for *T* can be performed equivalently on $T/\sim_B$.

Theorem 2.8 (Bisimulation preserves LTL properties) *Let T be a transition system and let $\sim_B$ be a bisimulation of T. Then T satisfies the LTL formula ϕ iff the bisimulation quotient $T/\sim_B$ satisfies ϕ.*

Even though we focus on LTL properties in this chapter, it should be noted that bisimulations also preserve properties expressed in more expressive logics such as CTL^* and μ-calculus. But even for LTL properties, bisimulation partitions are easier to compute than language equivalence quotients. In particular, for finite *T*, computing

$T/\sim_B$ can be performed in polynomial time, while computing $T/\sim_L$ is PSPACE-complete. Bisimulations can be computed using the following algorithm. If the algorithm terminates within a finite number of iterations of the loop, then there is a finite bisimulation quotient, and the algorithm returns a finite partition of the state space which is the coarsest bisimulation (i.e., the bisimulation with the fewest equivalence classes).

Algorithm 2 (Bisimulation Algorithm)

initially $Q/\sim_B := \{[\![\pi]\!] \mid \pi \in \Pi\}$ **;**

while there exist $P, P' \in Q/\sim_B$ such that $\emptyset \subsetneq P \cap \mathrm{Pre}(P') \subsetneq P$ **do**

$$P_1 := P \cap \mathrm{Pre}(P'); P_2 = P \setminus \mathrm{Pre}(P');$$

$$Q/\sim_B := (Q/\sim_B \setminus \{P\}) \cup \{P_1, P_2\}$$

end while;

return $Q/\sim_B$

Therefore, in order to show that LTL model checking can be decided for a transition system *T*, it suffices to show that the bisimulation algorithm terminates on *T*, and that each step of the algorithm is *computable* or *effective*. This means that we must be able to represent (possibly infinite) state sets symbolically, perform Boolean operations, check emptiness, and compute the predecessor operation *Pre* on the symbolic representation of state sets.

-
-
-

Bibliography

Alur R., Courcoubetis C., Halbwachs N., Henzinger T., Ho P.H., Nicollin X., Olivero A., Sifakis J., Yovine S. (1995). The algorithmic analysis of hybrid systems. *Theoretical Computer Science* **138**, 3-34.

Alur R., Dill D. (1994). A theory of timed automata. *Theoretical Computer Science* **126**, 183-235.

Alur R., Henzinger T., Lafferriere G., Pappas G. (2000). Discrete abstractions of hybrid systems. *Proceedings of the IEEE* **88**(7), 971-984.

Balluchi A., Benvenuti L., DiBenedetto M., Pinello C., Sangiovanni-Vincentelli A. (2000). Automotive engine control and hybrid systems : Challenges and opportunities. *Proceedings of the IEEE* **88**(7), 888-

912.

Caines P., Wei Y. (1995). The hierarchical lattices of a finite state machine. *Systems and Control Letters* **25**, 257-263.

Cerans K., Viksna J. (1996). Deciding reachability for planar multi-polynomial systems. In R. Alur, T. Henzinger, E. Sontag, eds., *Hybrid Systems III*, vol. 1066 of *Lecture Notes in Computer Science*, pp. 389-400, Berlin, Germany: Springer Verlag.

Daws C., Olivero A., Tripakis S., Yovine S. (1996). The tool KRONOS. In *Hybrid Systems III*, vol. 1066 of *Lecture Notes in Computer Science*, pp. 208-219, Springer-Verlag.

Engell S., Kowalewski S., Schulz C., Stursberg O. (2002). Simulation, analysis and optimization of continuous-discrete interactions in chemical processing plants. *Proceedings of the IEEE* **88**(7), 1050-1068.

Henzinger T., Kopke P., Puri A., Varaiya P. (1998). What's decidable about hybrid automata? *Journal of Computer and System Sciences* **57**, 94-124.

Koutsoukos X., Antsaklis P., J.Stiver, Lemmon M. (2000). Supervisory control of hybrid systems. *Proceedings of the IEEE* **88**(7), 1026-1049.

Lafferriere G., Pappas G., Sastry S. (1998a). Subanalytic stratifications and bisimulations. In T. Henzinger, S. Sastry, eds., *Hybrid Systems : Computation and Control*, vol. 1386 of *Lecture Notes in Computer Science*, pp. 205-220, Berlin: Springer Verlag.

Lafferriere G., Pappas G.J., Sastry S. (1998b). Hybrid systems with finite bisimulations. In P. Antsaklis, W. Kohn, M. Lemmon, A. Nerode, S. Sastry, eds., *Hybrid Systems V*, Lecture Notes in Computer Science, New York: Springer Verlag.

Lafferriere G., Pappas G.J., Sastry S. (2000). O-minimal hybrid systems. *Mathematics of Control, Signals, and Systems* **13**(1), 1-21.

Lafferriere G., Pappas G.J., Yovine S. (1999a). A new class of decidable hybrid systems. In *Hybrid Systems : Computation and Control*, vol. 1569 of *Lecture Notes in Computer Science*, pp. 137-151, Springer Verlag.

Lafferriere G., Pappas G.J., Yovine S. (1999b). reachability computation for linear hybrid systems. In *Proceedings of the* 14th *IFAC World Congress*, vol. E, pp. 7-12, Beijing, P.R. China.

Livadas C., Lygeros J., Lynch N. (2000). High-level modeling and analysis of tcas. *Proceedings of the IEEE* **88**(7), 926-948.

Lygeros J., Godbole D., Sastry S. (1998). Verified hybrid controllers for automated vehicles. *IEEE Transactions on Automatic Control* **43**(4), 522-539.

Milner R. (1989). *Communication and Concurrency*. Prentice Hall.

Pnueli A. (1977). The temporal logic of programs. In I.C.S. Press, ed., *Proceedings of the* 18th *Annual Symposium on Foundations of Computer Science*, pp. 46-57.

Tarski A. (1951). *A decision method for elementary algebra and geometry*. University of California Press, second edn.

Tomlin C., Pappas G.J., Sastry S. (1998). Conflict resolution for air traffic management : A study in muti-agent hybrid systems. *IEEE Transactions on Automatic Control* **43**(4), 509-521.

Van den Dries L. (1998). *Tame Topology and o-minimal structures*. Cambridge University Press.

Varaiya P. (1993). Smart cars on smart roads: problems of control. *IEEE Transactions on Automatic Control* **38**(2), 195-207.

Wong K., Wonham W. (1995). Hierarchical control of discrete-event systems. *Discrete Event Dynamic Systems* **6**, 241-273.

Biographical Sketch

George J. Pappas received the B.S. degree in Computer and Systems Engineering in 1991, the M.S. degree in Computer and Systems Engineering in 1992, both from Rensselaer Polytechnic Institute, Troy, NY. In 1994, he was a Graduate Fellow at the Division of Engineering Science of Harvard University. In December 1998, he received the Ph.D degree from the Department of Electrical Engineering and Computer Sciences at the University of California at Berkeley. He was a postdoctoral researcher at the University of California at Berkeley and the University of Pennsylvania. His is currently an Assistant Professor and Graduate Group Chair in the Department of Electrical Engineering at the University of Pennsylvania, where he also holds a secondary appointment in the Department of Computer and Information Sciences.

George Pappas is the recipient the NSF CAREER award in 2002, and the 1999 Eliahu Jury Award for Excellence in Systems Research from the Department of Electrical Engineering and Computer Sciences at the University of California at Berkeley. He was also a finalist for the Best Student Paper Award at the 1998 IEEE Conference on Decision and Control.

His research interests include hierarchical control systems, embedded hybrid systems, distributed control systems, nonlinear control systems, geometric control theory, with applications to flight management systems, robotics, and unmanned aerial vehicles.