Replace this file with `prentcsmacro.sty` for your meeting, or with `entcsmacro.sty` for your meeting. Both can be found at the ENTCS Macro Home Page.

Numerically-aided Deductive Safety Proof for a Powertrain Control System

Nikos Aréchiga[a,1], James Kapinski[b], Jyotirmoy V. Deshmukh[b], André Platzer[a] and Bruce Krogh[a]

[a] *Carnegie Mellon University, 5000 Forbes Ave Pittsburgh, PA, USA*

[b] *Toyota Technical Center, 1630 W. 186th, Gardena, CA, USA*

Abstract

The use of deductive techniques, such as theorem provers, has several advantages in safety verification of hybrid systems. There is often a gap, however, between the type of assistance that a theorem prover requires to make progress on a proof task and the assistance that a system designer is able to provide. To address this deficiency we present an extension to the deductive verification framework of differential dynamic logic that allows the theorem prover KeYmaera to *locally reason* about behaviors by leveraging forward invariant sets provided by external methods, such as numerical techniques and designer insights. Our key contribution is a new inference rule, the forward invariant cut rule, introduced into the proof calculus of KeYmaera. We demonstrate the cut rule in action on an example involving an automotive powertrain control systems, in which we make use of a simulation-driven numerical technique to compute a local barrier function.

1 Introduction

Most cyberphysical systems are *hybrid* in nature, i.e., have both continuous state evolution governed by differential equations and discrete mode transitions. Unfortunately, the problem of verifying safety properties for hybrid systems is undecidable [6], and most techniques that are used to verify software are not directly applicable. Many approaches to hybrid system verification focus on creating an overapproximation of the set of system states reachable over a fixed time horizon [9],[3],[4],[2]. While these approaches enjoy a high degree of automation, they are restricted in scope and scalability. An alternative is to employ deductive techniques that attempt to construct a symbolic proof of safety using a semi-interactive theorem prover [10]. Unlike reachable-set computation techniques, theorem provers can handle nonlinear dynamics directly, without introducing approximation artifacts. Further, theorem provers can handle proof tasks that involve symbolic parameters.

Safety verification of hybrid systems via theorem proving may incorporate human insight in the form of a *safety certificate*, i.e., a symbolic expression representing

[1] This work partially supported by the National Science Foundation under Grant NSF EXPEDITION CNS-0926181.

©2015 Published by Elsevier Science B. V.

a set containing all reachable states from a given initial set, while excluding unsafe states [1,10]. However, a designer usually has better insight about local behaviors in different operating regimes rather than overarching knowledge about the entire system. In [8], we demonstrated a numerical technique for discovering local invariants. This technique can be used to search for local invariants in specific regions of interest. In contrast to previous work focused on obtaining a global safety certificate, our approach encourages *local reasoning* and *lazy construction* of such certificates.

To support local reasoning, we introduce a new proof rule called the *forward invariant cut rule* in the calculus of the theorem prover KeYmaera. This rule is similar to an inductive invariant; given a region of operation and a proposed local safety certificate (in the form of a forward invariant), the rule allows us to decompose the overall proof into three proof obligations: (1) a proof of invariance of the proposed certificate, (2) a proof that the local certificate guarantees safety, and (3) a proof of safety of everything *excluding* the behaviors associated with the region pertaining to the local certificate.

We demonstrate this rule in a case study on safety verification for a powertrain control system. This system is a simplified model of a control system responsible for maintaining the air-to-fuel (A/F) ratio in a gasoline engine near an optimal setpoint. We describe the overall proof, but primarily describe the role of the forward invariant cut rule to prove that the A/F ratio remains within 10% of the optimal setpoint in KeYmaera.

2 Hybrid systems and hybrid programs

A hybrid dynamical system consists of a set of continuous-valued state variables $\mathbf{x}$ that take values from a domain $X \subseteq \mathbb{R}^n$ and a discrete-valued state variable q (also known as a *mode*) taken from a finite set Q. The system evolves in continuous or discrete time, and the configuration of a hybrid system at time t can be described by the values of its continuous and discrete state variables. In mode q, the evolution of the continuous-valued state variables is typically described using ordinary differential equations (ODEs) of the form $\dot{\mathbf{x}}(t) = f_q(\mathbf{x}(t))$, where f_q is a function from X to X. Though hybrid systems can have external inputs, here, we consider only *autonomous systems*, i.e., systems in which all transitions depend only on the system states. The state-dependent conditions that allow the system to transition from one discrete state to another (possibly same) discrete state are called *guards*.

While hybrid automata are often a convenient modeling formalism for such systems, here we use the *hybrid programs* notation to facilitate the use of the KeYmaera theorem prover, the workhorse for our deductive approach. To specify hybrid programs and specifications, KeYmaera uses the formalism of *differential dynamic logic*[2] denoted by d$\mathcal{L}$.

2.1 The logic d$\mathcal{L}$

A *hybrid program* is specified by the grammar

$$\alpha, \beta ::= \; x := \theta \mid x := * \mid \{x_1' = \theta_1, \ldots, x_n' = \theta_n \& H\} \mid ?H \mid \alpha \cup \beta \mid \alpha; \beta \mid \alpha^* \qquad (1)$$

[2] The syntax and semantics of d$\mathcal{L}$ are described in detail in [10]; we provide only a minimal overview here.

where α, β are hybrid programs, $\theta, \theta_1, \ldots, \theta_n$ are terms, and H is a logical formula. The program $x := \theta$ means that x is assigned the value of the term θ. The program $x := *$ means that x is nondeterministically assigned an arbitrary real value. The program $\{x_1' = \theta_1, \ldots, x_n' = \theta_n \& H\}$ means that the x_i variables evolve continuously for some duration, with derivatives θ_i subject to the constraint that the values taken by the x_i variables satisfy H during the entire flow. The hybrid program $?H$ behaves as a *skip* if the logical formula H is true, and as an *abort* otherwise. The rest follows notation similar to that for regular expressions, $\cup$ denotes nondeterministic choice, ; denotes sequential composition, and $*$ denotes arbitrary repetition. The formulas of d$\mathcal{L}$ are described by the grammar:

$$\phi, \psi ::= \ \theta_1 = \theta_2 \mid \theta_1 \geq \theta_2 \mid \neg\phi \mid \phi \wedge \psi \mid \phi \vee \psi \mid \phi \rightarrow \psi \mid [\alpha]\phi \mid \langle\alpha\rangle\, \phi \tag{2}$$

where ϕ, ψ are formulas of d$\mathcal{L}$, θ_1, θ_2 are terms, and α is a hybrid program. The box modality $[\alpha]\phi$ means that ϕ holds after all traces of the hybrid program α, and $\langle\alpha\rangle\, \phi$ means that ϕ holds after some execution of hybrid program α.

3 Safety verification with the forward invariant cut rule

The safety verification problem. The safety verification problem is to decide whether the state of a hybrid program α is always contained within a given safe set S when starting from a designated initial set I (i.e., to decide if $I \rightarrow [\alpha^*]S$).

We say that a set is *initialized* if it includes the initial set, *safe* if it excludes the unsafe set, and *invariant* if whenever a system behavior enters it, the behavior remains in the set for all future time. A *global safety certificate* is an initialized, invariant and safe set. Arguments with safety certificates are captured in d$\mathcal{L}$ using the invariant proof rule, where G is a global safety certificate:

$$\frac{I \rightarrow G \quad G \rightarrow [\alpha]G \quad G \rightarrow S}{I \rightarrow [\alpha^*]S} \tag{3}$$

The general task of finding a global safety certificate is difficult. In this work, we instead use knowledge of the system structure to propose sets that are invariant and safe, but not necessarily initialized, and leverage them in the proof procedure. We call such sets *local safety certificates*.

The forward invariant cut rule. A cut in a logical proof allows introducing a lemma. We introduce a new cut rule that, in our experience, simplifies the proof procedure by leveraging knowledge of local invariance properties. The following proof rule asserts that if a set $C \subseteq I$ is locally invariant ($C \rightarrow [\alpha]C$) and safe ($C \rightarrow S$), the remaining conditions ($I \wedge \neg C$) can be separately addressed to prove safety. We remark that we can prove that this proof rule is sound, but omit the proof for brevity.

Rule 3.1 (Forward Invariant Cut Rule)

$$\frac{I \wedge \neg C \rightarrow [(\alpha; ?\neg C)^*]S \quad C \rightarrow [\alpha]C \quad C \rightarrow S}{I \rightarrow [\alpha^*]S} \tag{4}$$

We note that the first branch in (3) concerns a global invariant set G that must hold for all initial states I, but the forward invariant cut set C need not hold for

all initial states I. Thus, the first branch in (4) corresponds to the states in I that are not included in C.

Numerical methods to obtain forward invariants. We briefly mention some existing techniques to generate safe forward invariant sets. In the hybrid systems community, barrier certificates have been proposed as a Lyapunov-like analysis technique to prove that starting from an initial set of states X_0, no system trajectory ever enters an unsafe set U [11]. To discover barrier certificates, we employ a modification of a technique from [12], which uses concrete system executions to generate a series of candidate barrier functions. Our technique, which is based on [8], uses concrete executions to generate a set of linear constraints. The linear constraints are used to construct the series of candidates. The final candidate is verified using a satisfiability modulo theories (SMT) solver that is capable of handling nonlinear theories over the reals, dReal [5].

4 Engine Fuel Control Case Study

Model. We present a case study involving a hybrid system representing an automotive fuel control system. Environmental concerns and government legislation require that the fuel economy be maximized and the exhaust gas emissions be minimized. At the ideal air-to-fuel (A/F) ratio, also known as the *stoichiometric* ratio, both quantities are optimized. We study an automotive control system whose purpose is to accurately regulate the A/F ratio.

The system dynamics and parameters were derived from a published model [7] and then simplified, as in [8]. The model consists of a simplified version of the physics of engine subsystems responsible for air intake and A/F ratio measurement, along with a computer control system tasked with regulating the A/F ratio. The objective of the controller is to maintain the A/F ratio within 10% of the nominal operating conditions. The experiment that we model involves an engine running at a fixed speed. The controller has two modes of operation: (1) a recovery mode, which controls fuel in an open-loop manner, i.e., with only feedforward control action, where the system runs for at most 8ms, and (2) a normal run mode, which uses feedback control to regulate the A/F ratio.

The controller measures both the air flow through the air-intake manifold, which it uses to estimate the air pressure in the manifold, and the oxygen content of the exhaust gas, which it uses to compute the A/F ratio. The recovery mode represents the behavior of the controller when recovering from a sensor fault (e.g., aberrant sensor readings or environmental conditions that cause suspicion of the sensor readings). During the recovery mode, the controller has no access to oxygen sensor measurements and must operate in a feedforward manner (i.e., using only the manifold air flow rate). The normal mode is the typical mode of operation, where the oxygen sensor measurements are used for feedback control.

Algorithm 1 is a hybrid program representing this system[3]. The ODEs representing the continuous dynamics in each mode and the model parameters are omitted for brevity, but they can be found in [8]. The state variables $\hat{p}, \hat{r}, \hat{p}_{est}$, and $\hat{i}$ represent the manifold pressure, the ratio between actual air-fuel ratio and the stoichiometric value, the controller estimate of the manifold pressure, and the internal

[3] Note that apparent redundancies appear in the hybrid program due to the explicit conversion from the representation of the system as a hybrid automaton to a hybrid program.

Algorithm 1: A d$\mathcal{L}$ model of a closed-loop fuel control system

1 $EFC \equiv I \rightarrow [(\mathtt{m}_1 \cup \mathtt{m}_2 \cup \mathtt{s}_{1\mapsto 2} \cup \mathtt{s}_{\{1,2\}\mapsto fail} \cup \mathtt{m}_{fail})^*]S$

2 $I \equiv ((\tau = 0) \ \wedge \ (M = recovery) \ \wedge \ (p_{est} = 0) \ \wedge \ (i = 0) \wedge (-10^{-3} \leq p \leq 10^{-3}) \ \wedge \ (-10^{-3} \leq r \leq 10^{-3}))$

3 $\mathtt{m}_1 \equiv (?M = recovery; ?\tau \leq 0.008; \{\exists \ell_1.\exists \ell_2.\exists \ell_3.(-0.86 \leq \ell_1 \leq 0.74) \ \wedge \ (-0.17 \leq \ell_2 \leq 0.18) \ \wedge \ (-0.81 \leq \ell_3 \leq 0.68) \wedge (p' = \ell_1) \ \wedge \ (r' = \ell_2) \ \wedge \ (p'_{est} = \ell_3) \ \wedge \ (i' = 0) \ \wedge \ (\tau' = 1) \ \& \ \tau \leq 0.008\})$

4 $\mathtt{s}_{1\mapsto 2} \equiv (?M = recovery; ?\tau \geq 0.008; M := normal;)$

5 $\mathtt{m}_2 \equiv (?M = normal; \{(p' = f_p) \ \wedge \ (r' = f_r) \ \wedge \ (p'_{est} = f_{p_{est}}) \ \wedge \ (i' = f_i) \ \& \ (-0.02 \leq p \leq 0.02) \ \wedge \ (-0.02 \leq r \leq 0.02) \wedge (-0.02 \leq p_{est} \leq 0.02) \ \wedge \ (-0.02 \leq i \leq 0.02)\})$

6 $\mathtt{s}_{\{1,2\}\mapsto fail} \equiv (?r < -0.1 \vee r > 0.1; M := fail;)$

7 $\mathtt{m}_{fail} \equiv (?r < -0.1 \vee r > 0.1; M := fail)$

8 $S \equiv M \neq fail$

state of the PI controller; these variables have all been translated so that the equilibrium point coincides with the origin. In the *recovery* mode, the continuous-time state $\mathbf{x}$ is the tuple $(\hat{p}, \hat{r}, \hat{p}_{est}, \hat{i}, \tau)$. The additional state variable in the *recovery* mode represents the state of a timer that evolves according to the ODE $\dot{\tau} = 1$. In the *normal* mode, the state is given by $(\hat{p}, \hat{r}, \hat{p}_{est}, \hat{i})$. We assume the system is within 1.0% of the nominal value at the initialization of the *recovery* mode. This represents the case where the system was previously in a mode of operation that accurately regulated the A/F ratio to the desired setpoint. A domain of interest for the state variables is given by $\|\mathbf{x}\|_\infty < 0.2$, which contains the set of reasonable values for the state variables.

Safety proof using forward invariant cut. The verification goal is to ensure that in the given experimental setting, the system always remains within 10% of the nominal A/F ratio after a fixed recovery time of 8.0 ms has passed. We describe an interactive proof using KeYmaera. To assist the proof procedure, we generated a barrier function, $B : \mathbb{R}^n \rightarrow \mathbb{R}$, using the numeric technique outlined in Sec. 3. We then use the set enclosed by the barrier, $\{\mathbf{x}|B(\mathbf{x}) \leq 0\}$, to formulate the forward invariant cut:

$$C \equiv (M = normal) \wedge (B(\mathbf{x}) \leq 0). \tag{5}$$

We apply the forward invariant cut inference rule (4), producing three proof obligations that KeYmaera has to discharge.

Obligation 1: $C \rightarrow [\alpha]C$**.** From the definition of C, the hybrid programs $\mathtt{m}_1$ and $s_{1\mapsto 2}$ may be ignored, as both have the hybrid program $?M = recovery$ as their first item, which is inconsistent with C. Thus, this obligation only needs to be proved for the programs $\mathtt{m}_2$, $\mathtt{s}_{\{1,2\}\mapsto fail}$, and $\mathtt{m}_{fail}$. To discharge this obligation for the program $\mathtt{m}_2$ we use the barrier certificate rule that we have added to KeYmaera's proof calculus, below.

$$\frac{I \rightarrow (B(\mathbf{x}) \leq 0) \qquad (B(\mathbf{x}) = 0) \rightarrow \frac{\partial B}{\partial \mathbf{x}} \cdot f(\mathbf{x}) < 0 \qquad (B(\mathbf{x}) \leq 0) \rightarrow S}{I \rightarrow [\{x' = f(\mathbf{x})\}]S} \tag{6}$$

To apply the barrier certificate rule, we use the initial states I defined in Algorithm 1 and substitute S with C. The first and the third proof obligations in the barrier

certificate rule are trivially satisfied due to our choice of the barrier function and the nature of the forward invariant cut. For the middle proof obligation KeYmaera asks dReal if the query $(B(\mathbf{x}) = 0) \wedge (\frac{\partial B}{\partial \mathbf{x}} \cdot f_{normal}(\mathbf{x}) > -\epsilon)$ is unsatisfiable.

To discharge the proof obligation for $\mathtt{m}_2$, $\mathtt{s}_{\{1,2\}\mapsto fail}$, KeYmaera needs to show that if $B(\mathbf{x}) < 0$ holds, either of these programs cannot invalidate C by transitioning to mode $fail$. It proves this by showing that the set $B(\mathbf{x}) < 0$ is a subset of the safe set using dReal.

Obligation 2: $C \rightarrow S$**.** This obligation is trivial as S requires the mode to anything but $fail$, while C says that the mode is *normal* mode.

Obligation 3: $I \wedge \neg C \rightarrow [(\alpha; ?\neg C)^*]S$**.** Here we introduce $C1$, which is invariant for all initial conditions in $I \wedge \neg C$.

$$C1 \equiv (M \neq fail) \wedge (0 \leq \tau \leq 0.008) \wedge (x \in S_{reach}) \tag{7}$$

Here S_{reach} is an overapproximation of reachable sets by using upper and lower bounds on $\dot{p}$ and $\dot{r}$ computed using dReal. There is one additional barrier certificate application to show that the normal mode, when starting in this set, lands within the barrier certificate and hence also respects this invariant. This requires a derivative negativity argument, which KeYmaera resolves by querying dReal. The rest is handled by KeYmaera's standard deduction procedures.

Conclusion. We demonstrated how to leverage local knowledge of the powertrain control system to propose a set that is invariant and safe. We used the set to apply the forward invariant cut rule to prove a safety property for the system.

References

[1] Bernhard Beckert, Reiner Hähnle, and Peter H Schmitt. *Verification of object-oriented software: The KeY approach*. Springer-Verlag, 2007.

[2] Xin Chen, Erika Abraham, and Sriram Sankaranarayanan. Flow*: An Analyzer for Non-Linear Hybrid Systems. In *CAV*, 2013.

[3] Goran Frehse. PHAVer: Algorithmic verification of hybrid systems past HyTech. *STTT*, 10(3):263–279, 2008.

[4] Goran Frehse, Colas Le Guernic, Alexandre Donzé, Scott Cotton, Rajarshi Ray, Olivier Lebeltel, Rodolfo Ripado, Antoine Girard, Thao Dang, and Oded Maler. SpaceEx: Scalable verification of hybrid systems. In *CAV*, pages 379–395, 2011.

[5] Sicun Gao, Jeremy Avigad, and Edmund M Clarke. δ-complete decision procedures for satisfiability over the reals. In *J. Automated Reasoning*, pages 286–300, 2012.

[6] Thomas A. Henzinger, Peter W. Kopke, Anuj Puri, and Pravin Varaiya. What's Decidable about Hybrid Automata? . *JCSS*, 57(1):94 – 124, 1998.

[7] Xiaoqing Jin, Jyotirmoy V. Deshmukh, James Kapinski, Koichi Ueda, and Ken Butts. Powertrain Control Verification Benchmark. In *Hybrid Systems: Computation and Control*, 2014.

[8] James Kapinski, Jyotirmoy V. Deshmukh, Sriram Sankaranarayanan, and Nikos Aréchiga. Simulation-guided lyapunov analysis for hybrid dynamical systems. In *Hybrid Systems: Computation and Control*, 2014.

[9] James Kapinski and Bruce H Krogh. Verifying asymptotic bounds for discrete-time sliding mode systems with disturbance inputs. In *ACC*, pages 2852–2857, 2004.

[10] André Platzer. *Logical Analysis of Hybrid Systems*. Springer, 2010.

[11] Stephen Prajna. *Optimization-based methods for nonlinear and hybrid systems verification*. PhD thesis, California Institute of Technology, Caltech, Pasadena, CA, USA, 2005.

[12] U. Topcu, P. Seiler, and A. Packard. Local stability analysis using simulations and sum-of-squares programming. *Automatica*, 44:2669–2675, 2008.