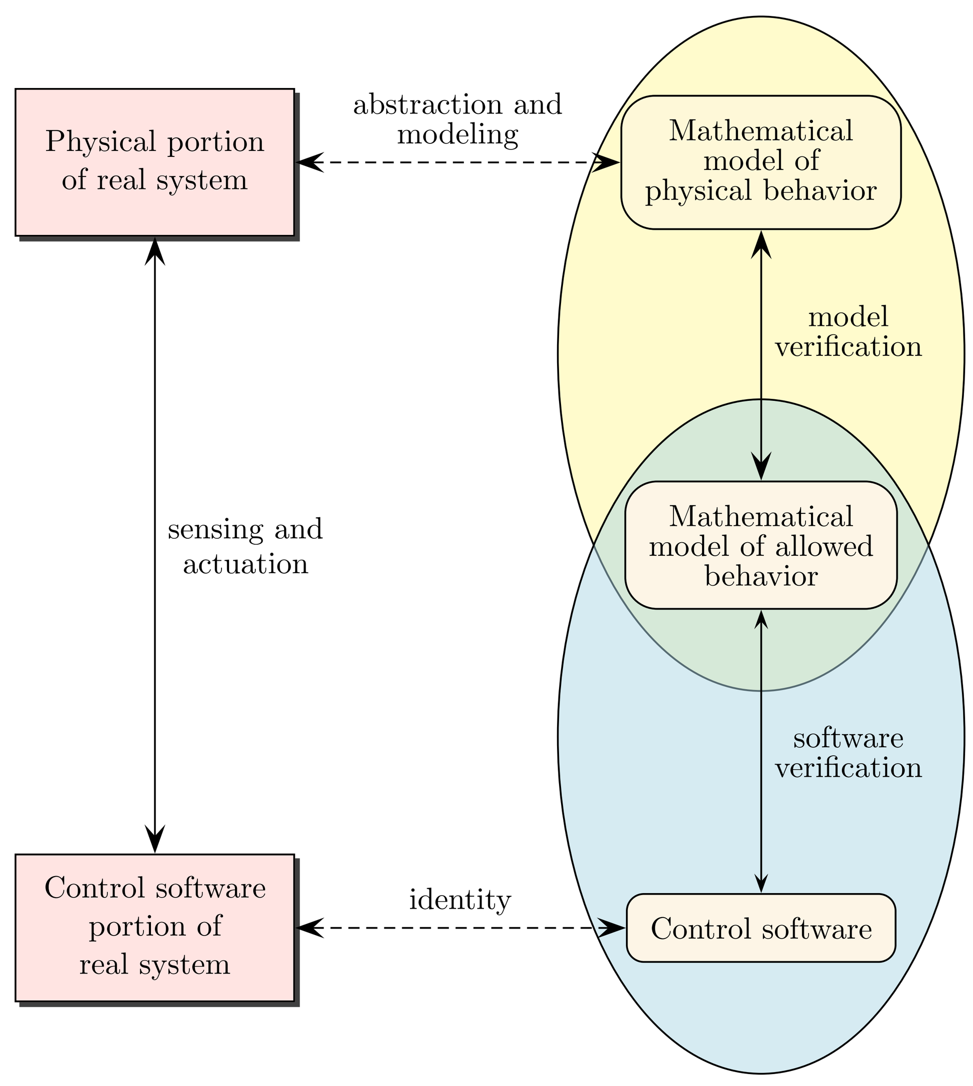
Verification of Smooth and Close Collision-Free Cruise Control

Theodore P. Pavlic,
Paolo A. G. Sivilotti, Alan D. Weide, and Bruce W. Weide
Department of Computer Science & Engineering
The Ohio State University
{pavlic.3, sivilotti.1, weide.3, weide.1}@osu.edu

Abstract

Modern adaptive cruise control technologies are designed to improve the comfort or safety of the driver; however, no safety guarantees are asserted by these designs. Furthermore, existing theoretical work in the safety verification of adaptive cruise control algorithms require both discrete braking modes and overly conservative separation distances to make such safety guarantees. Thus, existing work in safety verification both risks reducing driver comfort while also eliminating any of the performance gains typically associated with automated highways. Our work extends verification of automated highway systems to mitigate both of these problems. Motivated by optimal control and verification of software systems, we have developed safety conditions for adaptive cruise control algorithms that do not require discontinuous braking and also allow for substantially lower following distances than existing work in the verification of autonomous highway systems. Moreover, we demonstrate a novel approach for verifying software in hybrid systems by embedding the continuous dynamics into the software specifications. The result is a verified software paradigm consistent with the vision of Hoare's verifying compiler.

Tools for Verifying Cyber-Physical Systems



CPS Concrete–Abstract Correspondence

```
havoc dt
assume  0.0 < dt  and  dt < rho

physical loop
  maintains
    bl = #bl  and  bf = #bf  and
    afMax = #afMax  and rho = #rho  and
    af = #af  and dt = #dt  and
    0.0 <= t  and  t < rho + dt  and
    vl = VEL(#vl, -bl, t)  and
    xl = POS(#xl, #vl, -bl, t)  and
    vf = VEL(#vf, af, t)  and
    xf = POS(#xf, #vf, af, t)  and
    xl >= xf
while IsGreater (rho, t) do
  variable zero, dv, dx: Real

  dv := Replica (dt)
  Multiply (dv, bl)
  Subtract (vl, dv)
  if IsGreater (zero, vl) then
      Clear (vl)
  end if
  dx := Replica (dt)
  Multiply (dx, vl)
  Add (xl, dx)

  dv := Replica (dt)
  Multiply (dv, af)
  Add (vf, dv)
  if IsGreater (zero, vf) then
      Clear (vf)
  end if
  dx := Replica (dt)
  Multiply (dx, vf)
  Add (xf, dx)

  Add (t, dt)
end loop
```

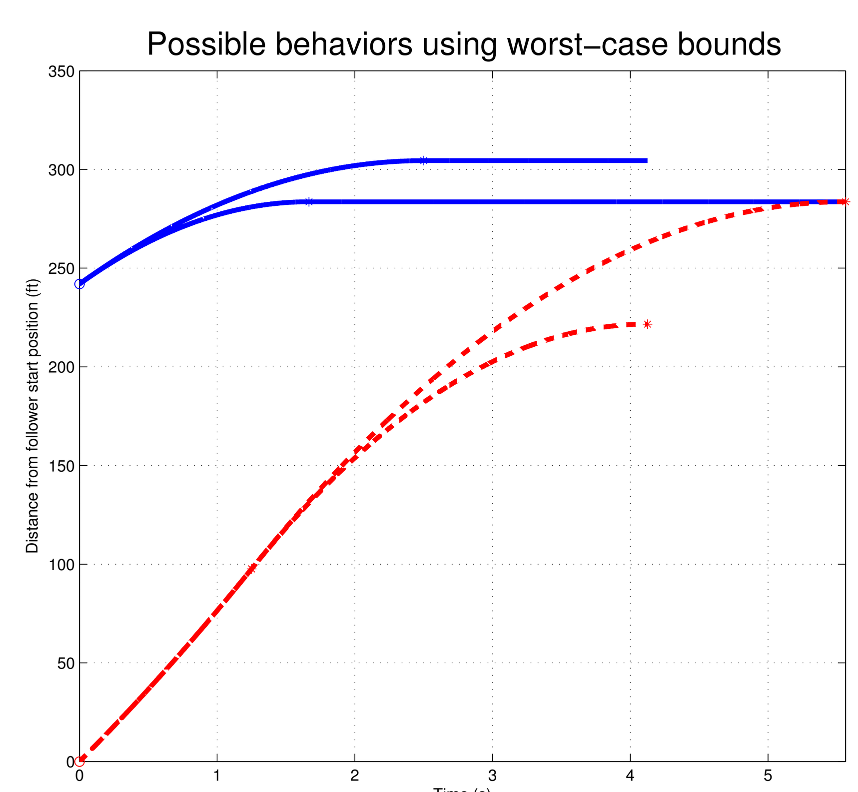
Augment Annotated Code with Physical Loop

```
Prove:
VEL (vl_4, -bl_0, t_11) - dt_9 × bl_0
   = VEL (vl_4, -bl_0, t_11 + dt_9)
Given:
0.0 < bl_0
0.0 < bf_0
0.0 < afMax_0
bf_0 ≤ bl_0
0.0 < rho_0
MINGAP (vl_2, bl_0, vf_2, bf_0, afMax_0, rho_0)
   ≤ xl_2 - xf_2
0.0 ≤ vl_2
0.0 ≤ vf_2
0.0 ≤ vl_4
0.0 ≤ vf_4
MINGAP (vl_4, bl_0, vf_4, bf_0, afMax_0, rho_0)
   ≤ xl4 - xf4 -bf_0 ≤ af_8
af_8 ≤ afMax_0
MINGAP (VEL (vl_4, -bl_0, rho_0),
         bl_0, VEL (vf_4, af_8, rho_0),
         bf_0, afMax_0, rho_0)
   ≤ POS (xl_4 - xf_4, vl_4, -bl_0, rho_0)
   - POS (0.0, vf_4, af_8, rho_0)
0.0 < dt_9
dt_9 < rho_0
t_11 < rho_0
0.0 ≤ t_11
t_11 < rho_0 + dt_9
POS (xf_4, vf_4, af_8, t_11)
   ≤ POS (xl_4, vl_4, -bl_0, t_11)
0.0 ≤ VEL (vl_4, -bl_0, t_11) ≤ dt_9 × bl_0
VEL (vf_4, af_8, t_11) + dt_9 × af_8 < 0.0
```

Example Verification Condition (VC)

Conventional Verification of ACC

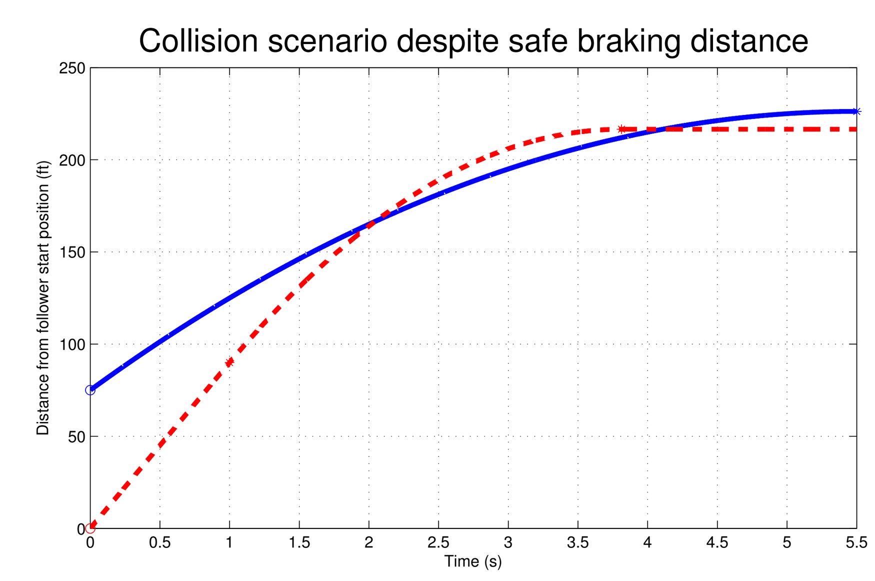
- Assume global upper and lower braking bounds
- Must apply minimum brake whenever worst-case collision scenario is possible
- Acceleration-safe distance grows as distance between bounds grow



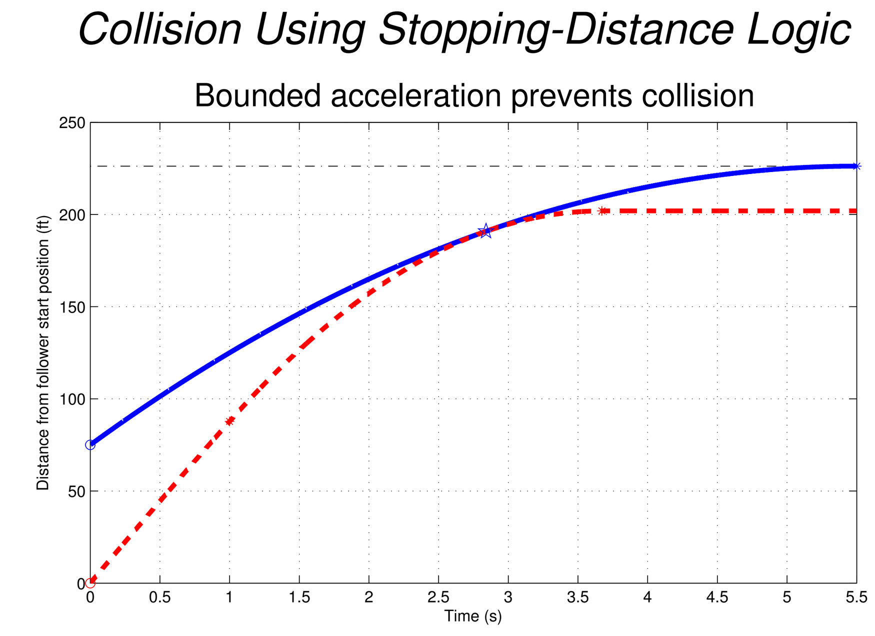
Specifications using worst-case stopping distances

Heterogeneous Smooth-and-Close ACC

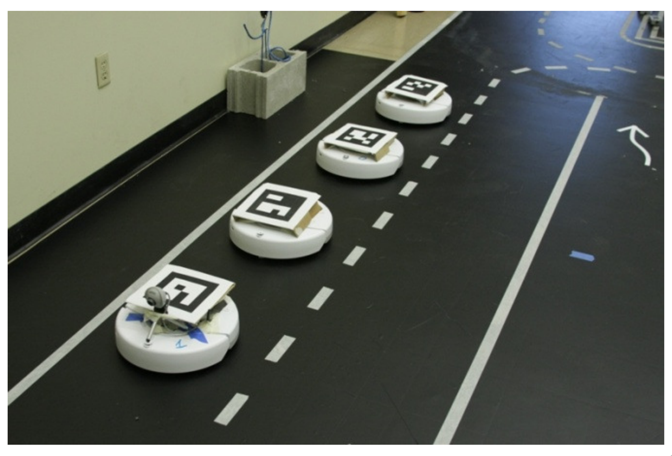
- Local braking is known
- Upper bound on leader is known (e.g., plate tag)
- Adjust upper bound on safe local acceleration
- Stopping-distance condition not sufficient

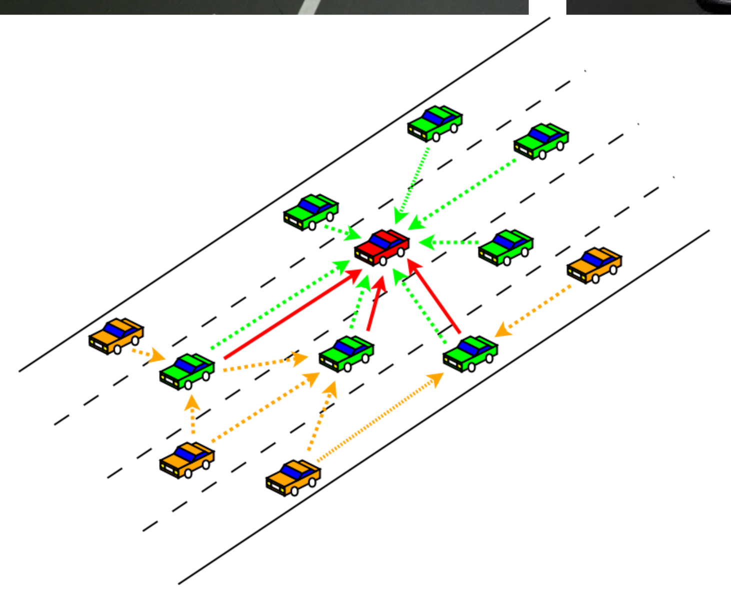


Collision Using Stopping-Distance Logic



Marginally Safe Stop after Evasive Acceleration





Mixed-Traffic Adaptive Cruise Control (ACC)

Supported by NSF Grant ECCS-0931669.