Hierarchical Hybrid Modeling of Embedded Systems

Rajeev Alur

Systems Design Research Lab

University of Pennsylvania

www.cis.upenn.edu/~alur/

EMSOFT, Tahoe City, October 2001

Programming Interacting Autonomous Robots

Low level

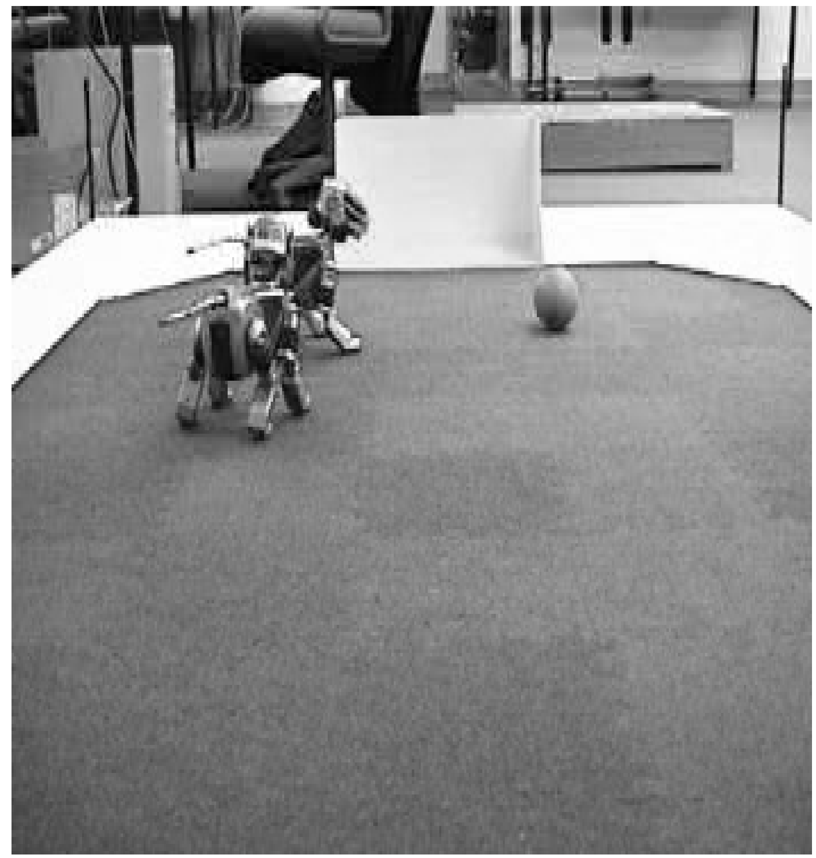
- Analysis of vision data
- Control laws for legs
- Wireless cards

Current programming

- How to implement Go-to-ball
- Real-time scheduling

High level

- Modes: Attack, Defend
- How to switch? Strategies
- Communication to collaborate



Trends in Model-Based Design

- Emerging notations: UML-RT,Stateflow
  - Visual, Hierarchical, Object oriented
  - Simulation, code generation
- Formal models and Model checking tools
- Programming languages (Esterel, FRP…)
- Control engineer's tools (Matlab…)
- Design/Simulation environments
  - SHIFT, Ptolemy-II, Modelica ….

Guiding Themes for CHARON

- Integrated modeling of control program and physical environment (hybrid)
  - Programming language technology in Control tools
  - Continuous modeling in Programming environments
- Foundations for hybrid systems in presence of concurrency, hierarchy, exceptions …
  - Compositionality, refinement, ….
- Models need to be analyzable
  - Model checking
  - Exploit modeling constructs for efficiency

CHARON Team

Faculty

Rajeev Alur (CIS)

Vijay Kumar (MEAM)

Insup Lee (CIS)

George Pappas (EE)

PhD Students

Joel Esposito

Yerang Hur

Franjo Ivancic

P K Mishra

Usa Sammapun

Research Associates

Thao Dang

Salvatore La Torre

Supratik Mukhopadhyay

Oleg Sokolsky

Collaborators

Rafael Fierro (U Oklahoma)

Radu Grosu (SUNY StonyBrook)

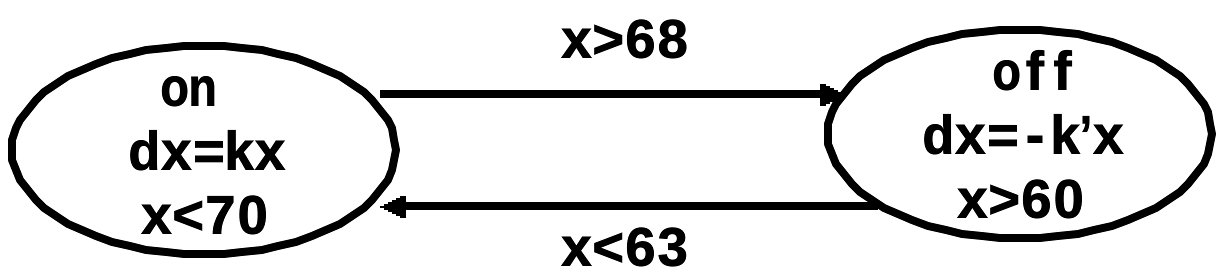
Funding: DARPA Mobies, NSF

Talk Outline

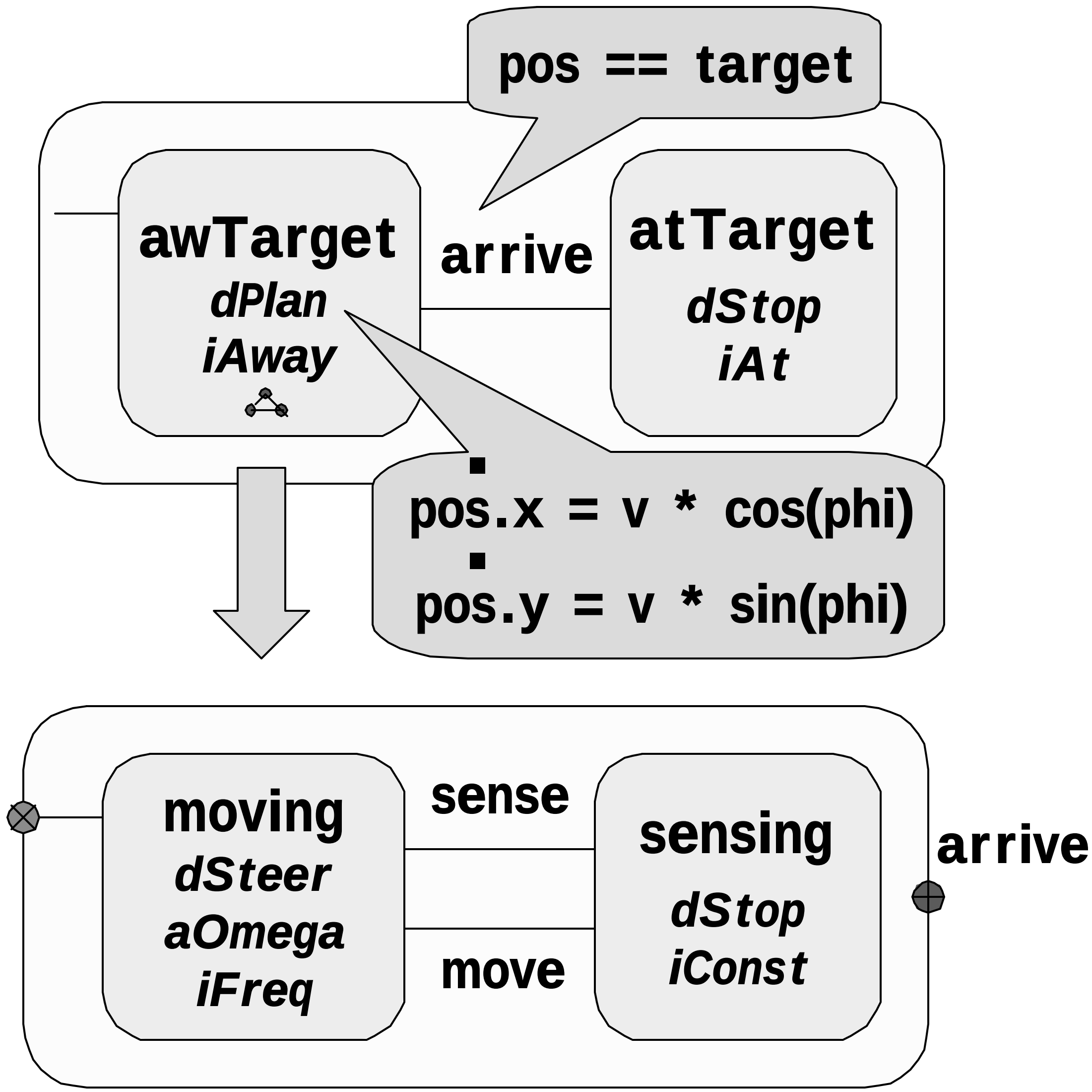
- ✓ Motivation
- ➲ CHARON Summary
- ❑ Compositional Refinement
- ❑ Model Checking via Predicate Abstraction
- ❑ Conclusions

Hybrid Modeling

State machines + Dynamical systems



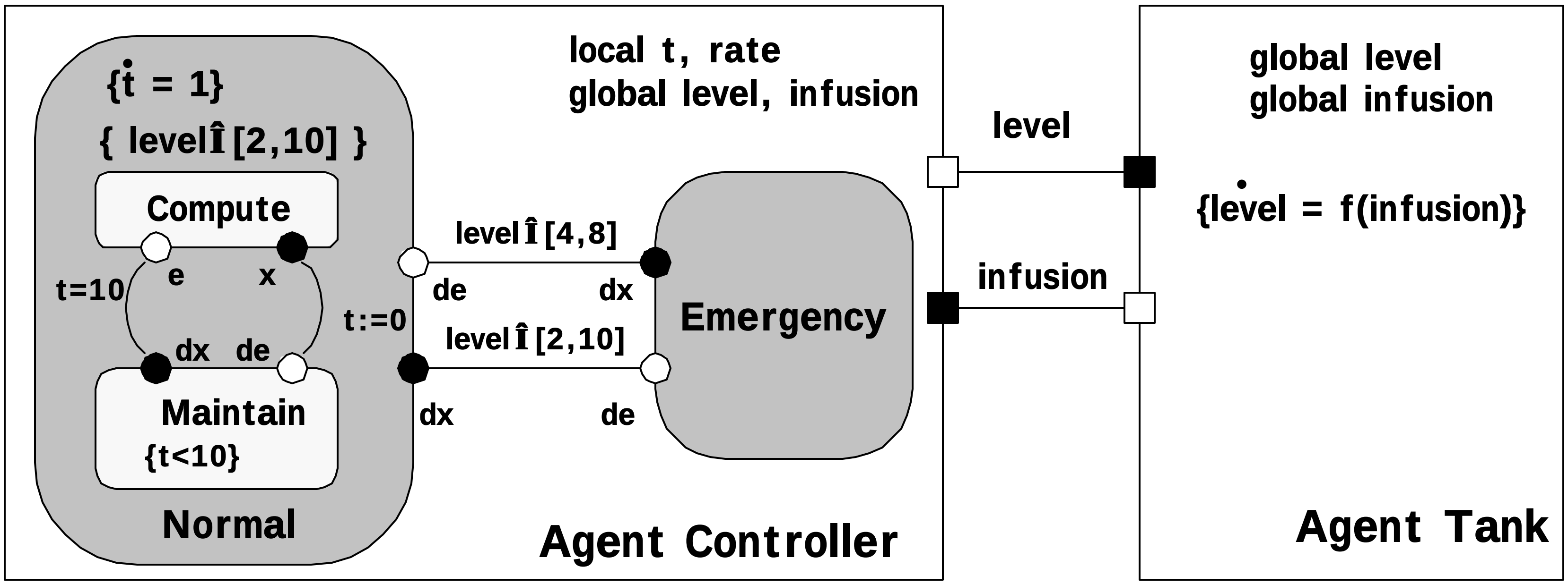
Behavioral Hierarchy



CHARON Language Features

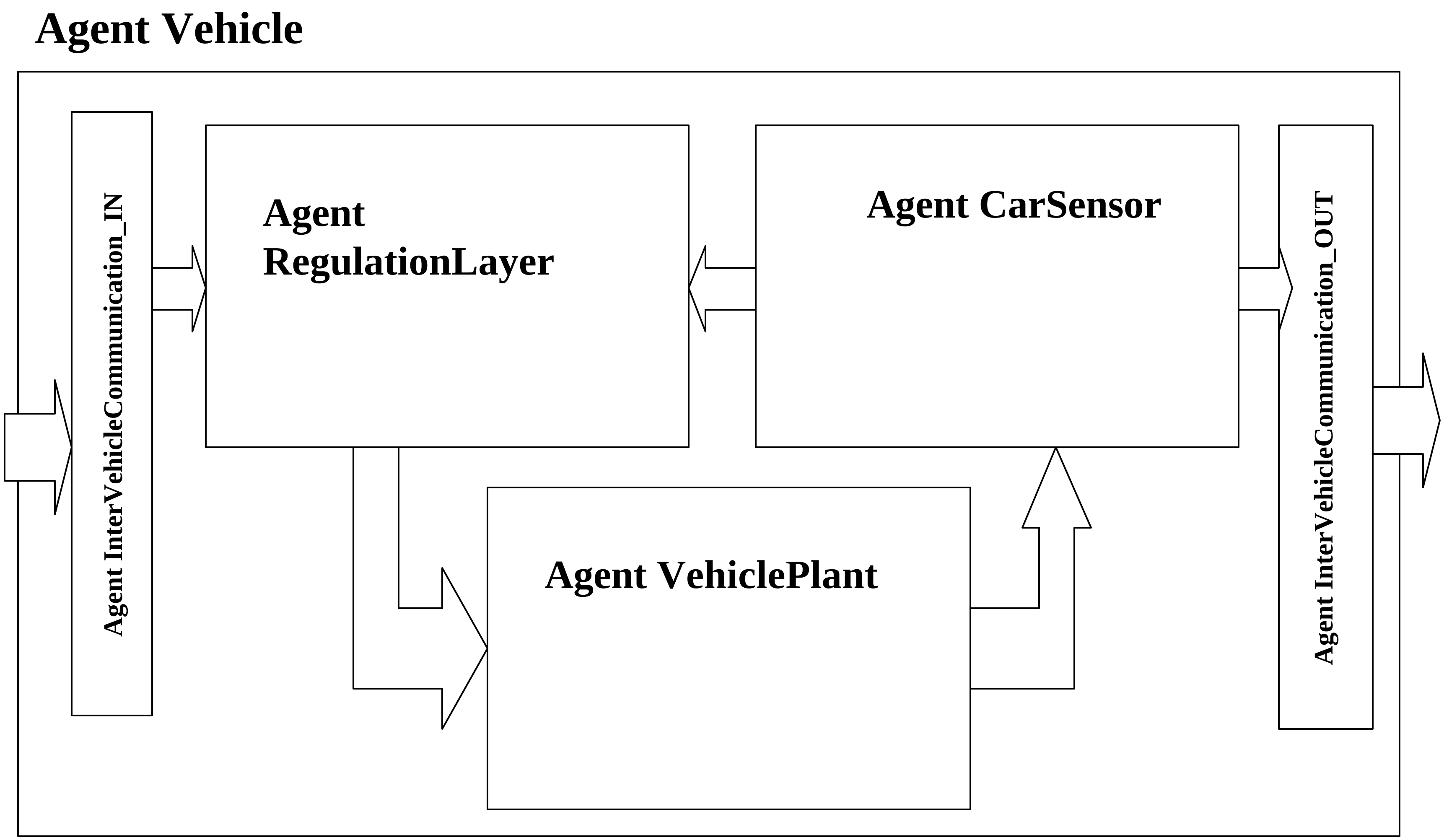
- Individual components described as agents
  - Composition, instantiation, and hiding
- Individual behaviors described as modes
  - Encapsulation, instantiation, and Scoping
- Support for concurrency
  - Shared variables as well as message passing
- Support for discrete and continuous behavior
  - Differential as well as algebraic constraints
  - Discrete transitions can call Java routines

Syntax: Modes and Agents

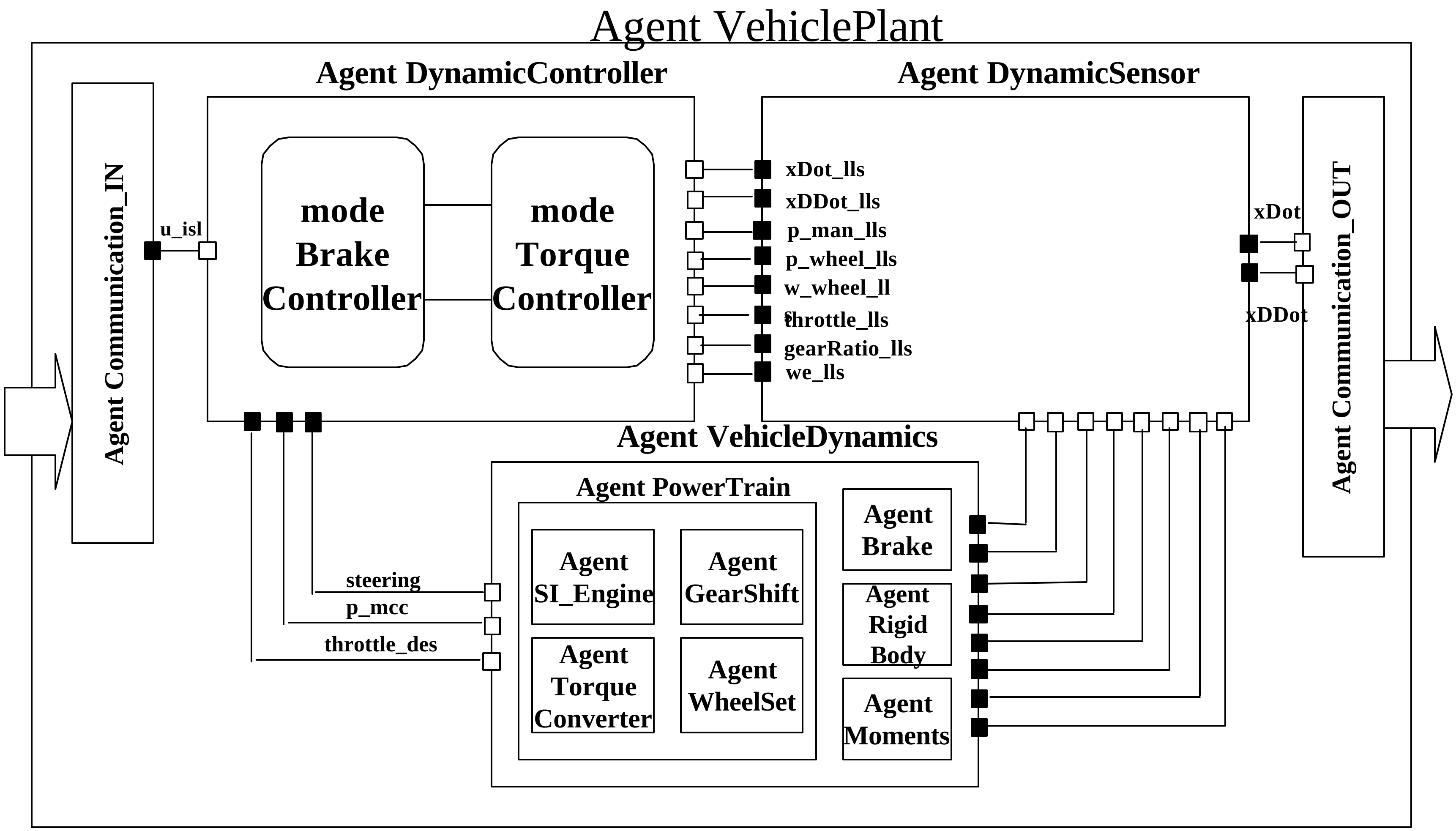


- ❑ Modes describe sequential behavior
- ❑ Agents describe concurrency

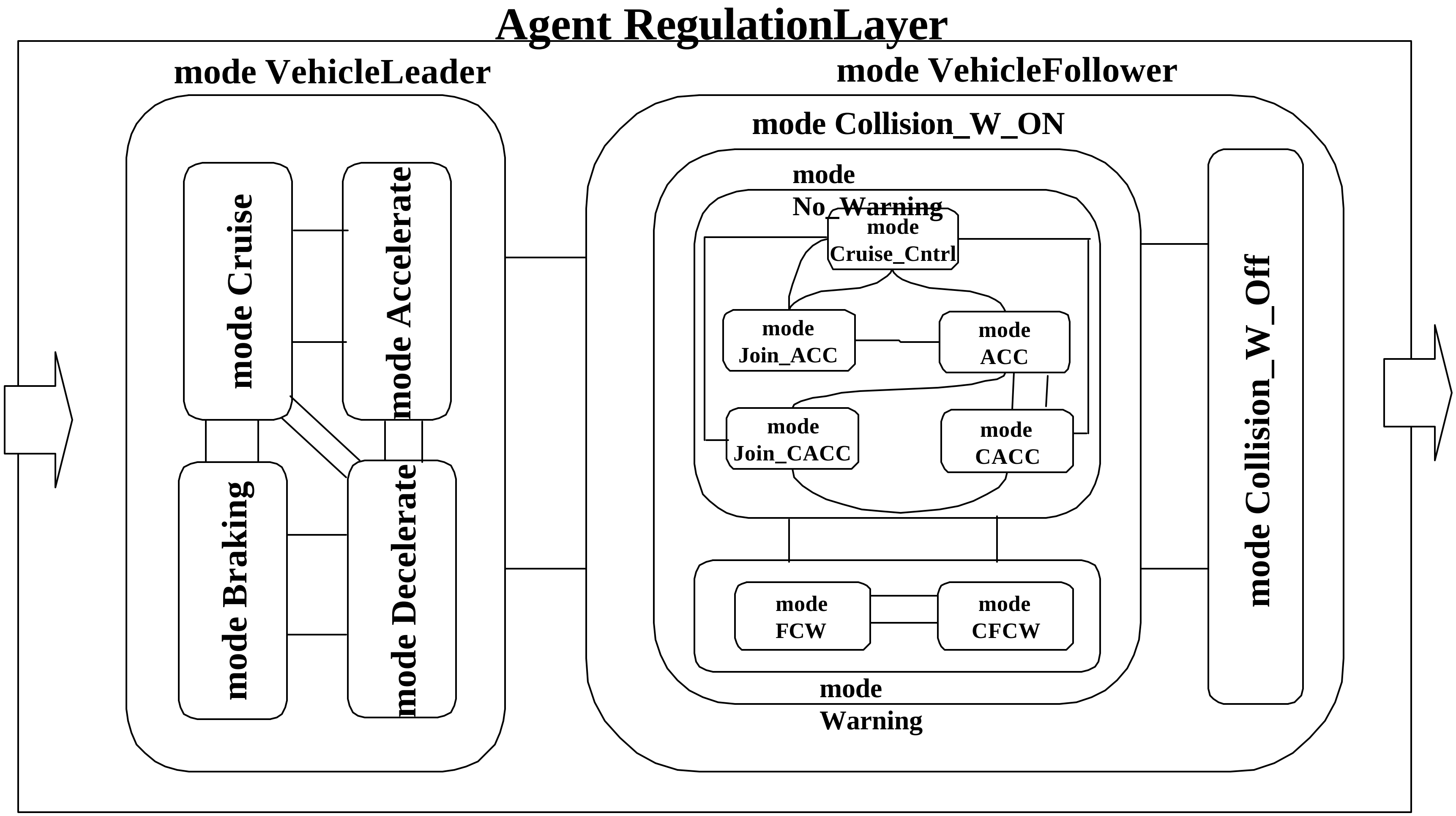
Example: V2V model



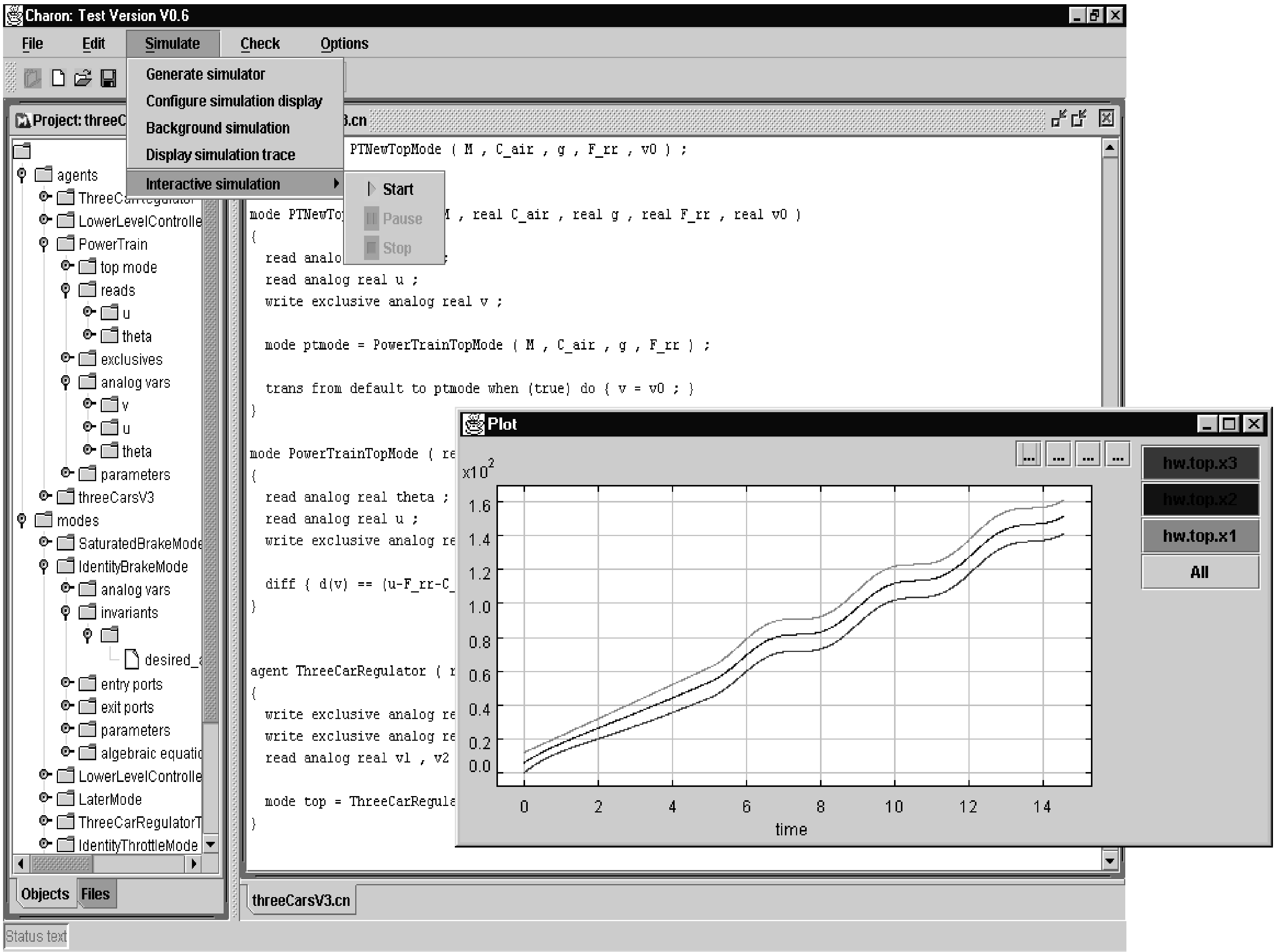
Agent VehiclePlant



Agent RegulationLayer



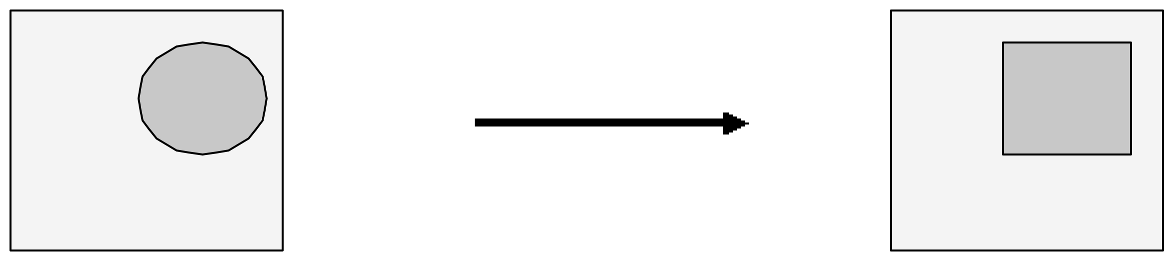
CHARON Toolkit



Talk Outline

- ✓ Motivation
- ✓ CHARON Summary
- ➲ Compositional Refinement
- ❑ Model Checking via Predicate Abstraction
- ❑ Conclusions

What is Compositionality ?

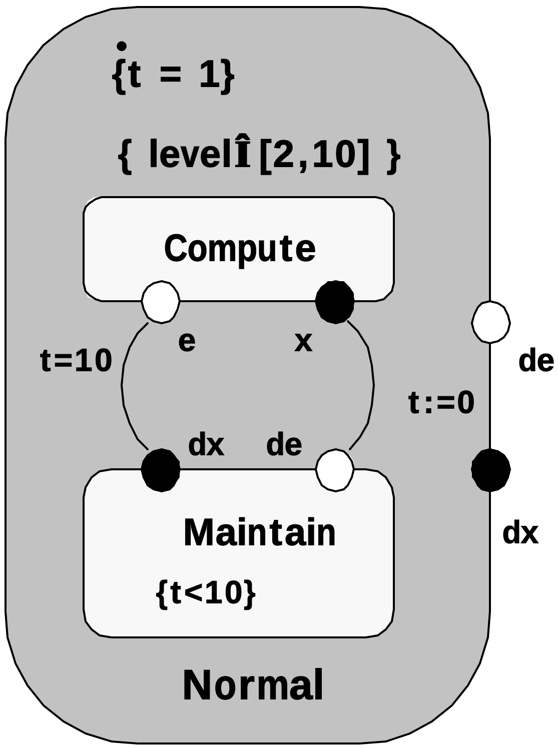


Which properties are preserved?

Can we restrict reasoning to modified parts of design?

Mode should have a precise interface spec that would permit composition of behaviors

Mode Executions



(ctl,t,level,infusion,rate,h)

(dx,0,5.1,1,0.2,Maintain)

Flow Step

(dx,10,15.1,3,0.2,Maintain)

Env Step

(de,10,15.1,5,0.2,Maintain)

Discrete Mode Step

(dx,10,15.1,5,0.1,Compute)

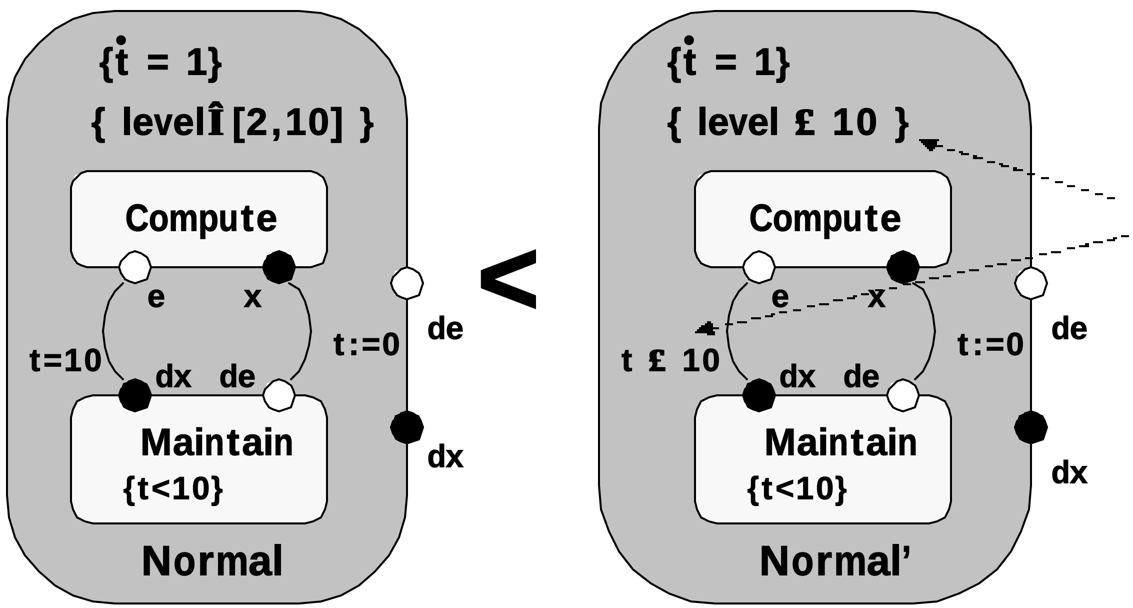
Semantics of modes

- Semantics of a mode consists of:
  - **control interface: entry and exit points**
  - **data interface: global variables**
  - **traces (sequences over observable states)**
- Key Thm: Semantics is compositional
  - **traces of a mode can be computed from traces of its sub-modes**

Refinement

Refinement is trace inclusion

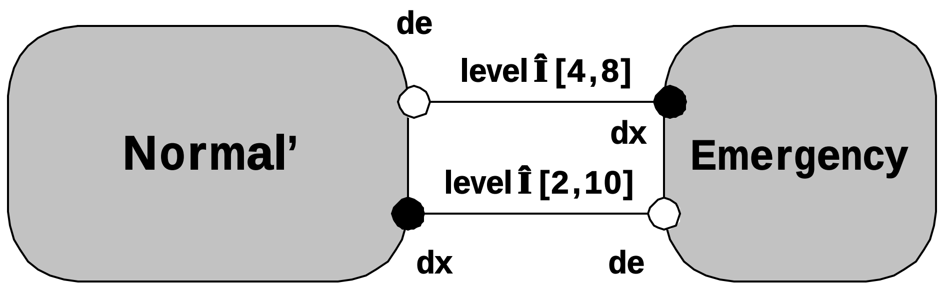
Normal < **Normal'**



- Same control points and global variables
- Guards and constraints are relaxed

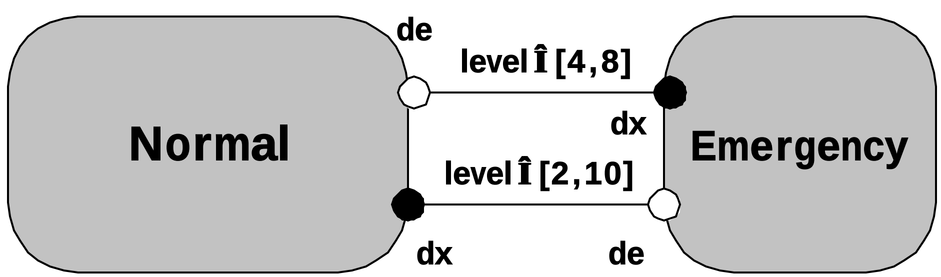
Sub-mode refinement

Controller’

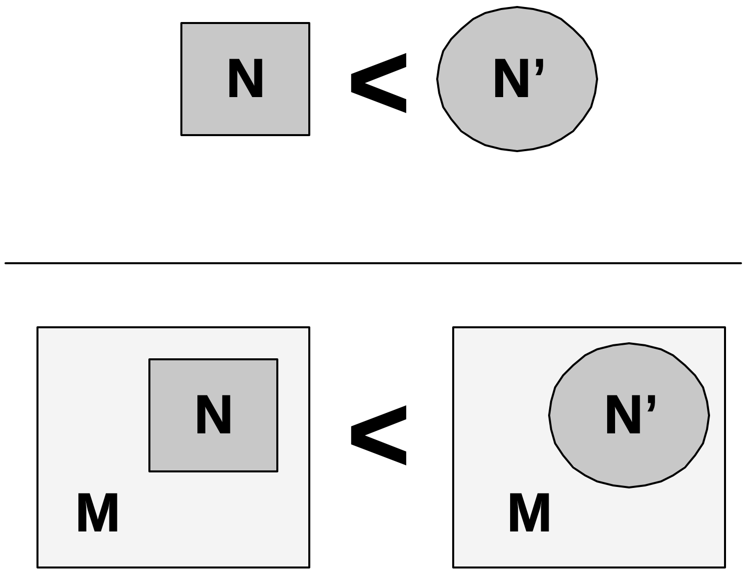


Refines

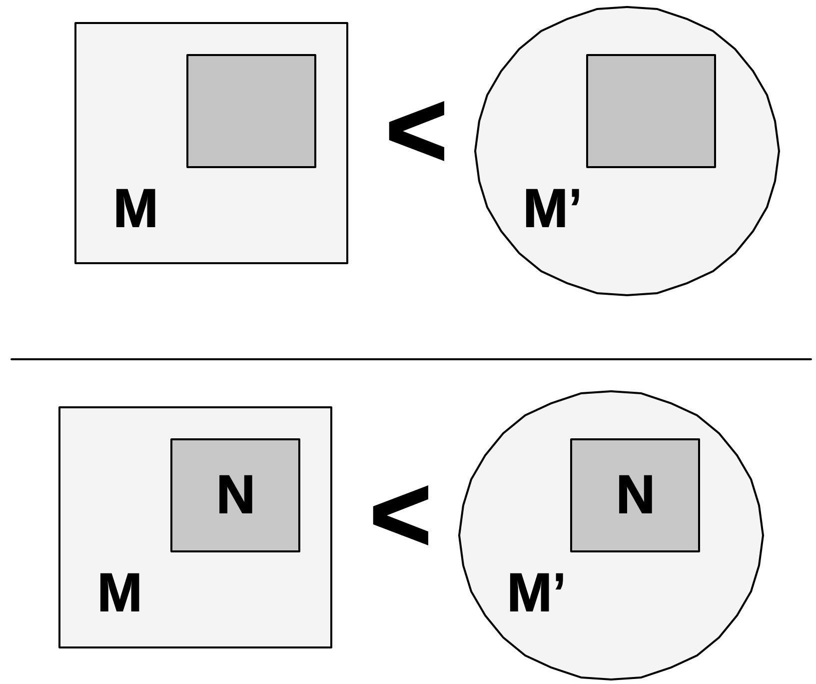
Controller



Compositional Reasoning



Sub-mode refinement



Context refinement

Automated refinement checking for discrete systems
Pipelined processors, Network protocols

Talk Outline

✓ Motivation

✓ CHARON Summary

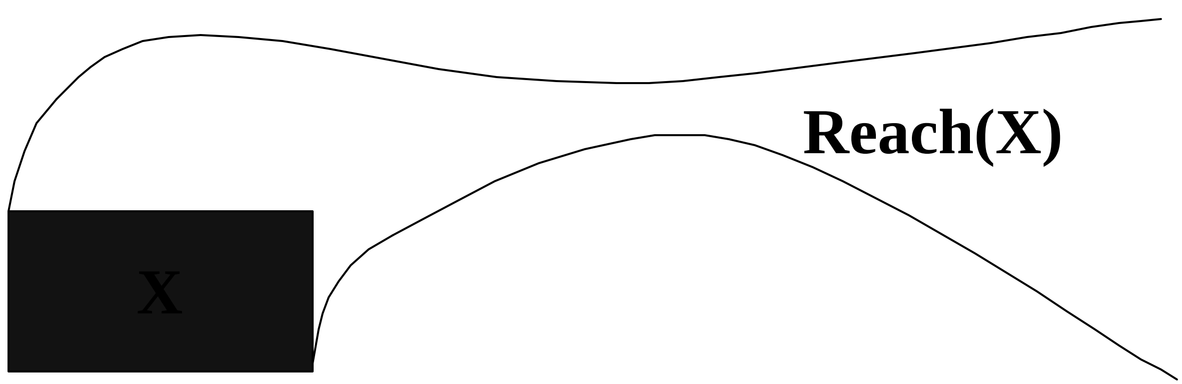
✓ Compositional Refinement

➲ Model Checking via Predicate Abstraction

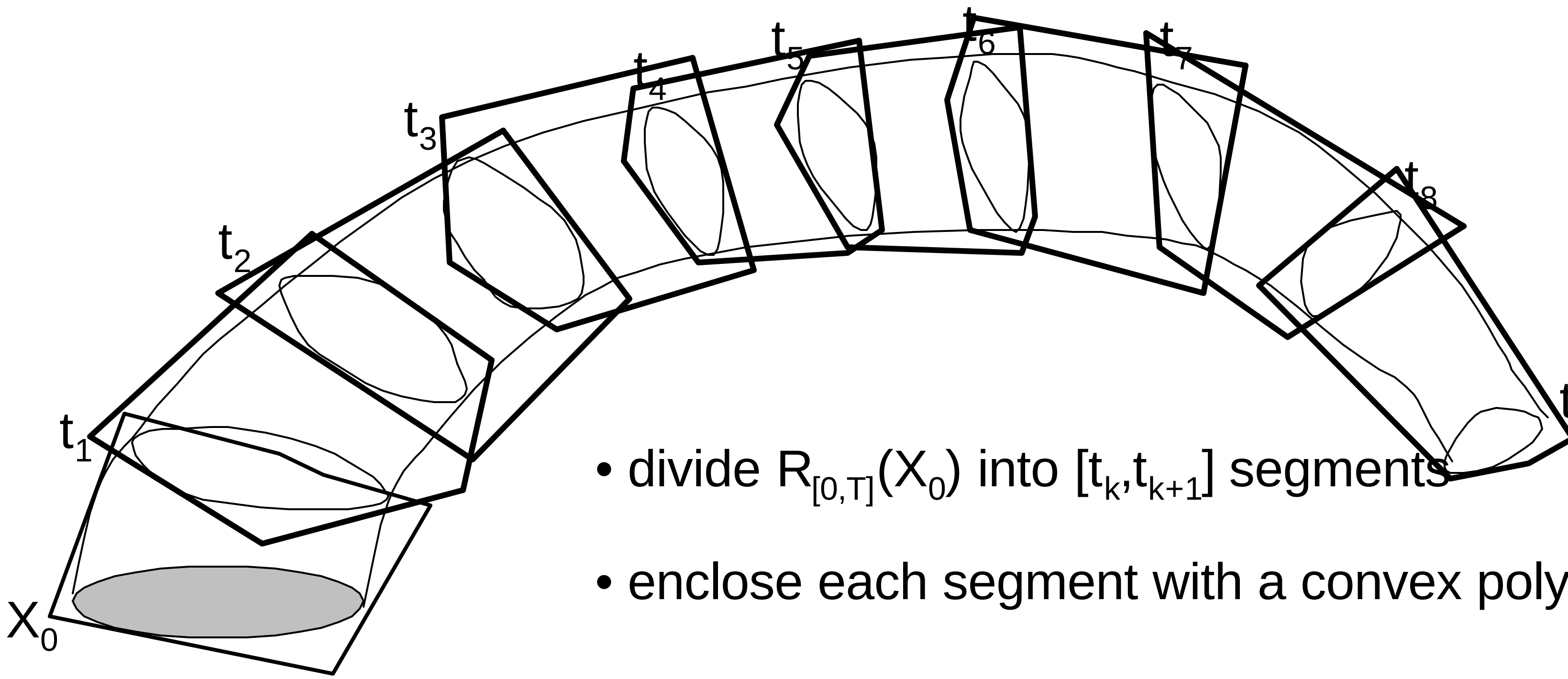
❑ Conclusions

Model Checking of Hybrid Systems

- Goal: Given an initial region, compute whether a bad state can be reached
- Existing tools: HyTech, d/dt, Checkmate
- Key step is to compute Reach(X) for a given set X under dx = Ax+Bu (expensive !!!)



Polyhedral Flow Pipe Approximations

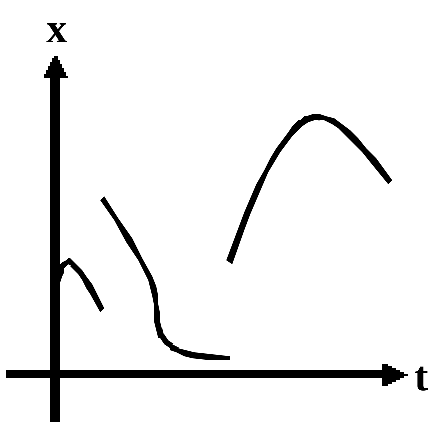


- divide $R_{[0,T]}(X_0)$ into $[t_k, t_{k+1}]$ segments
- enclose each segment with a convex polytope
- $R^M_{[0,T]}(X_0)$ = union of polytopes

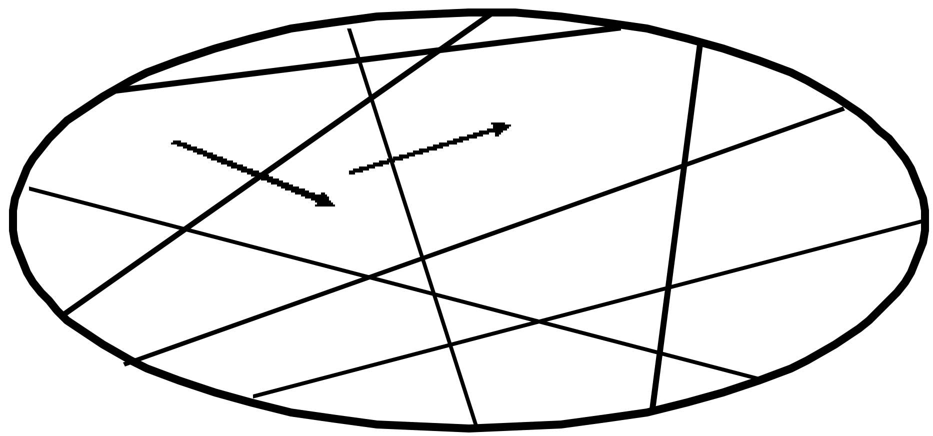
A. Chutinan and B. H. Krogh, Computing polyhedral approximations to dynamic flow pipes, IEEE CDC, 1998

Predicate Abstraction

- Input is a hybrid automaton and a set of *k* boolean predicates, e.g. $x+y > 5-z$.
- The partitioning of the concrete state space is specified by the user-defined *k* predicates.

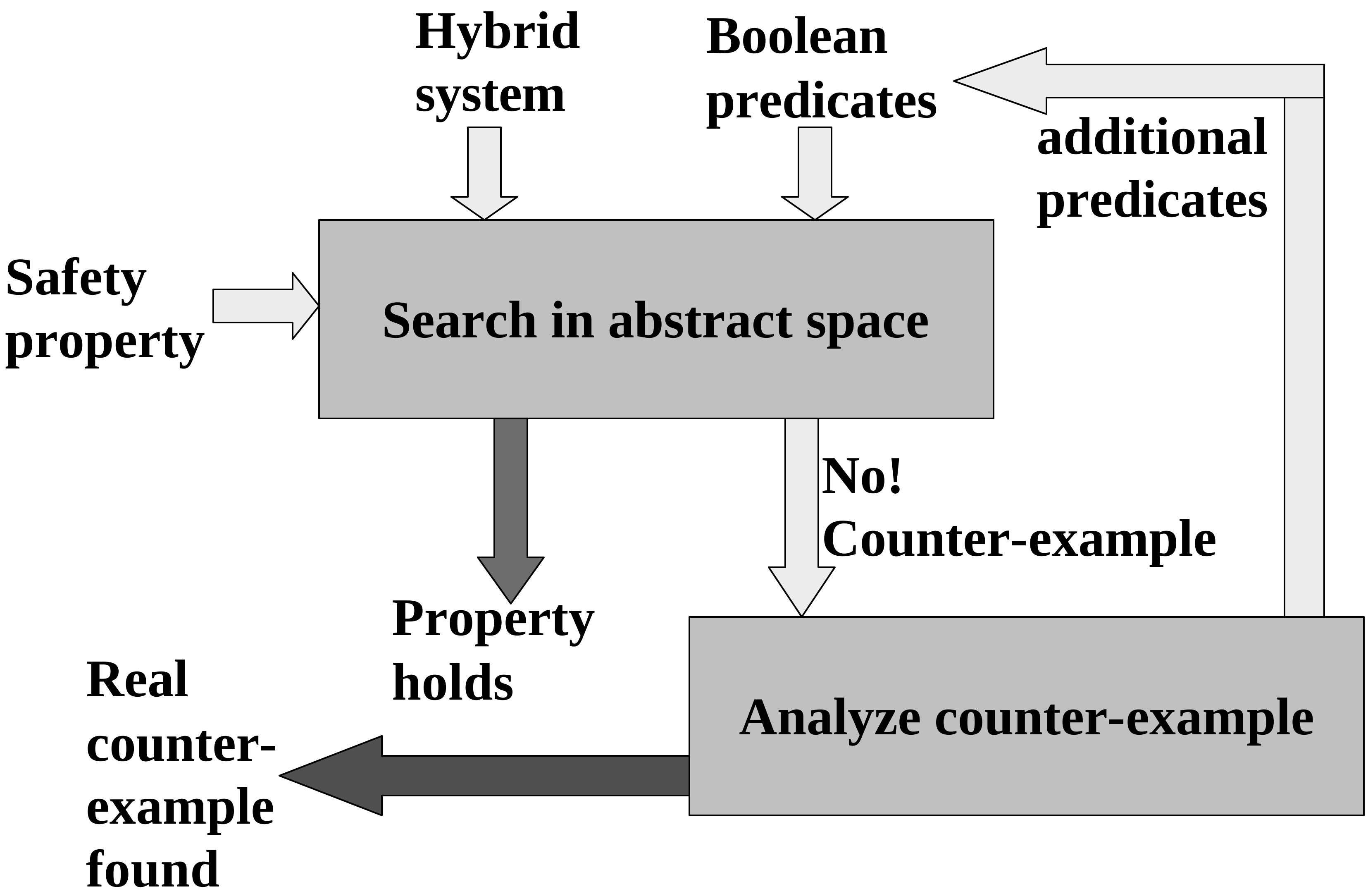


Concrete Space:
L x R ^{n}



Abstract Space:
L x {0,1} ^{k}

Overview of the Approach



Why use this approach?

- Reach(X) needs to be computed only for abstract states X and not intermediate regions of unpredictable shapes/complexity
- No need to compute Reach (X). Goal is to find one new abstract state reachable from X, partial results are of great use
  - Simulate vertices
  - Consider time-slices at discrete times
- Our focus is on search strategies to make progress in the abstract state-space
- Initial implementation in C++ with promising results

Case-study: V2V

- Platoon controller for 3 vehicles scenario
- First step: make the model linear (feedback linearization)
- Initial predicates from model description. E.g.
  - **from LowLevelController:**
    - **u_isl < 0, u_isl > 0**
  - **from BrakeControl:**
    - **u_isl < max_brake_pressure**
  - **from ThrottleControl:**
    - **u_isl < max_throttle**
- Safety of the controller verified using the tool (17 predicates, 4 continuous vars, < 1 min)

Talk Outline

✓ Motivation

✓ CHARON Summary

✓ Compositional Refinement

✓ Model Checking via Predicate Abstraction

➲ Conclusions

Ongoing Activities

- **Research**
  - Distributed simulation
  - Qualitative abstractions of hybrid systems
  - Model-based test generation
  - Accurate event detection for simulation
  - Exploiting hierarchy for efficient simulation
- **Applications/Case-studies**
  - Multiple autonomous robots
  - MoBIES challenge problems
  - Animation
  - Biomolecular networks…

Wrap-Up

- Modeling and Analysis in symbiosis
- Common themes in many modeling proposals
  - Hierarchy
  - Concurrency and Communication
  - Component interfaces
  - Formal semantics and compositionality
  - Integration: Discrete + Continuous + Stochastic
- Automating formal verification is hard, but not impossible, and there is steady progress
- Biological systems: emerging application for modeling with similarities to embedded software