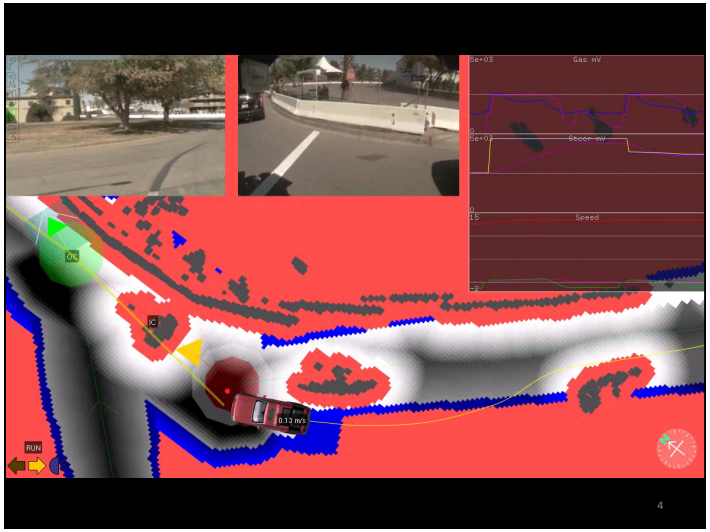
Motivation

2



A Driver's License for Autonomous Vehicles

- Earning a driver's license implies competency and safety in a *large variety* of *driving scenarios*.
- It is a skill assessment used to ***bound the risk*** *of operating a vehicle*.

6

A Driver's License for Autonomous Vehicles

- Under what *criteria* can we determine that an *autonomous vehicle is* ***safe***?
- How can we **generalize** its actions ***beyond simple tests***?

7

The APEX Approach

How can we provide a driver's license for autonomous vehicles?

The APEX Approach

How can we provide a driver's license for autonomous vehicles?

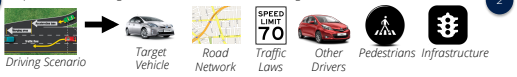
1 Explore the modeling of subsets of autonomous driving missions as scenarios

2 Map driving scenarios to collections of agents represented as hybrid systems

The APEX Approach

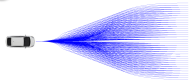
How can we provide a driver's license for autonomous vehicles?

1 Explore the modeling of subsets of autonomous driving missions as scenarios



2 Map driving scenarios to collections of agents represented as hybrid systems

3 Implement planning stack in C++



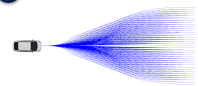
The APEX Approach

How can we provide a driver's license for autonomous vehicles?

1 Explore the modeling of subsets of autonomous driving missions as scenarios

2 Map driving scenarios to collections of agents represented as hybrid systems

3 Implement planning stack in C++

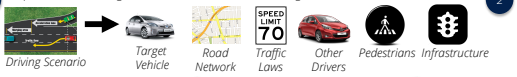


4 Integrate ROS based vehicle OS

The APEX Approach

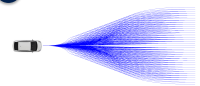
How can we provide a driver's license for autonomous vehicles?

1 Explore the modeling of subsets of autonomous driving missions as scenarios



2 Map driving scenarios to collections of agents represented as hybrid systems

3 Implement planning stack in C++



4 Integrate ROS based vehicle OS

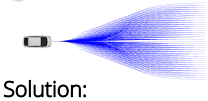
5 Validation of algorithms & software

The APEX Approach

How can we provide a driver's license for autonomous vehicles?

1. Explore the modeling of subsets of autonomous driving missions as scenarios

2. Map driving scenarios to collections of agents represented as hybrid systems
3. Implement planning stack in C++

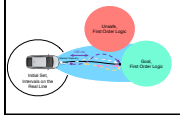


4. Integrate ROS based vehicle OS

5. Validation of algorithms & software

Solution:

6. Reachability Problem

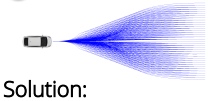


The APEX Approach

How can we provide a driver's license for autonomous vehicles?

1. Explore the modeling of subsets of autonomous driving missions as scenarios

2. Map driving scenarios to collections of agents represented as hybrid systems
3. Implement planning stack in C++

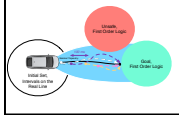


4. Integrate ROS based vehicle OS

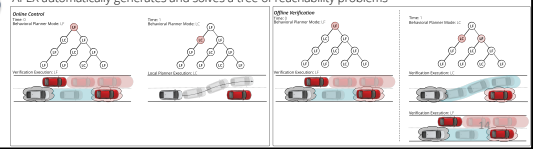
5. Validation of algorithms & software

Solution:

6. Reachability Problem



7. APEX automatically generates and solves a tree of reachability problems



Outline

Part 1: APEX Framework (SAE 2016) *10:00-11:30 AM*
- Development of an AV model
- Development of hybrid systems framework to accommodate AV model
- Scenario: lane change
- Technical details, proofs, and discussion

Part 2: Demo: Agile AV Platform (RTSS WIP 2015) *11:30-12:00*
- F1/10 demo
- Hardware and algorithms overview

Lunch

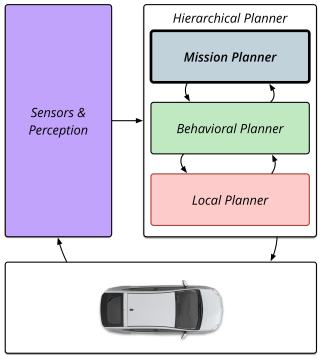
Part 3: APEX Benchmarks and Complexity (ARCH 2016) *1:30-2:30*
- Development of AV model part 2
- Pure Pursuit
- Differential Inclusions
- Composition of Agent Automatons
- Complex Scenario from Agents

Part 4: Co-design of Control and Perception (RTSS 2015) *2:30-3:00*
- Algorithms for Low Cost AV

Part 5: Next Steps and Discussion *3:00-4:00*
- Final Report Format
- Development of framework for scenario creation from Open Street Maps.
- Scenario authoring tools and agent building blocks.
- Support for AV model in Unity and RViz.

Development of an AV Model

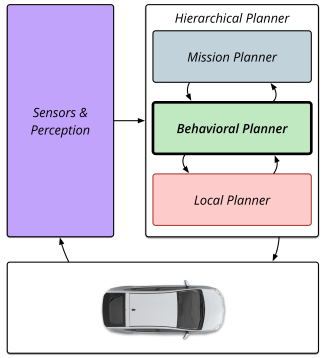
Basic Architecture: Mission Planner



- The ***mission planner*** determines the series of road network links that the AV will follow in order to accomplish the ***mobility goals*** of the vehicle's occupants.
- Such information defines the ***set of waypoints*** that the behavioral planner will reason about. This is **not a current focus of our research.**

17

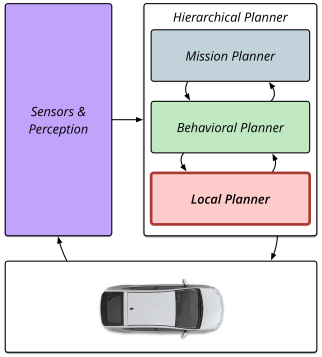
Basic Architecture: Behavioral Planner

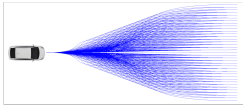


- The ***behavioral planner*** makes local decisions regarding the short term goals of the vehicle. For example the behavioral planner may ***initiate a lane change*** based on the need to exit the highway.
- There are a **huge number of potential scenarios** and **no canonical implementation** of this layer.

18

Basic Architecture: Local Planner





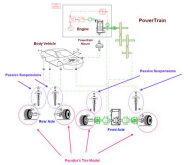
- The ***local planner*** is responsible for trajectory generation and evaluation.
- Based on the goal selection provided by the behavioral planner, the local planner may use a variety of *optimal control techniques* to compute ***dynamically*** and ***kinematicaly feasible trajectories***.
- It is **essential to our framework** because **bad events** are *defined by the* ***vehicle state***.

19

Which Vehicle Model?

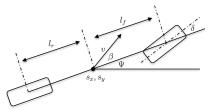
1. **Real Car:** ✓
   Testing and Validation

2. **Systems Integration:** ✗
   Multi-domain Simulation

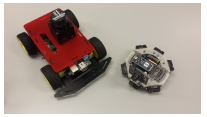


3. **Algorithm Prototype:**
   Agile AV Platform ✓

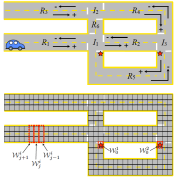
4. **Nonlinear Bicycle Model:**
   Simulation, Verification, and Control ✓



5. **Test Platform:** ✗
   Differential Drive Robots



6. **Discrete Transition System** ✗



Vehicle Model

- The ***bicycle model*** to is used as the intermediate representation (IR) because it supports verification of ***low level trajectory trackers*** and analysis with respect to first order logic necessary for ***formal verification.***
- Model captures the effect of steering input and longitudinal acceleration on the vehicle's trajectory.

$$\dot{\beta} = \left(\frac{C_r l_r - C_f l_f}{mv^2}\right)\dot{\psi} + \left(\frac{C_f}{mv}\right)\delta - \left(\frac{C_f + C_r}{mv}\right)\beta$$

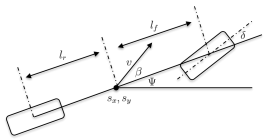
$$\ddot{\psi} = \left(\frac{C_r l_r - C_f l_f}{I_z}\right)\beta - \left(\frac{C_f l_f^2 - C_r l_r^2}{I_z}\right)\left(\frac{\dot{\psi}}{v}\right) + \left(\frac{C_f l_f}{I_z}\right)\delta$$

$$\dot{v} = a_x$$

$$\dot{s}_x = v\cos(\beta + \psi)$$

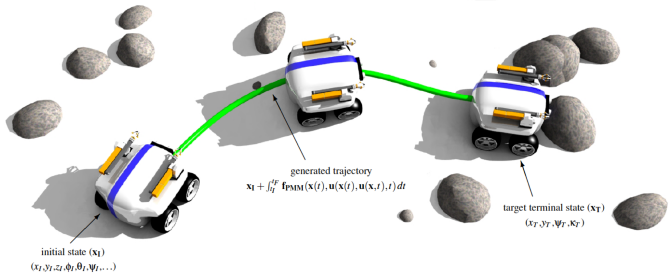
$$\dot{s}_y = v\sin(\beta + \psi)$$

$$\dot{\delta} = v_w$$



21

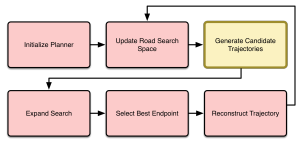
The Local Planning Problem



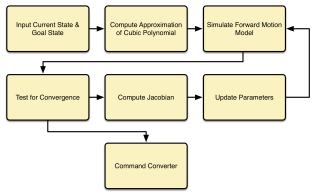
22

Local Planning Details

Local planning high-level overview



Trajectory generation high-level overview



Problem Definition

- The state x of a vehicle is described by the following vector:

$$x = [s_x, s_y, s_z, \phi, \theta, \rho, v_x, v_y, v_z, \omega_x, \omega_y, \omega_z, \kappa]^t$$

- Define the initial state as x_i and the target state as x_t
- Note that κ, the curvature is simply the rate of change of θ with respect to distance traveled
- For this exercise we will consider a vehicle simplified to a 2D model:

$$x = [s_x, s_y, \theta, v, \kappa]$$

- Which evolves according to the following set of ordinary nonlinear differential equations:

$$\dot{x} = f(x, u(p, x))$$

- Here, $u(p, x)$ is the parameterized control input

Methodology

- We will be concerned with computing trajectories which lead from an initial state x_i to a final state x_t within a finite time horizon.
- The final state state boundary constraints which are given by the mission planner are defined by the following vector:

$$x_c = [s_x, s_y, \theta]$$

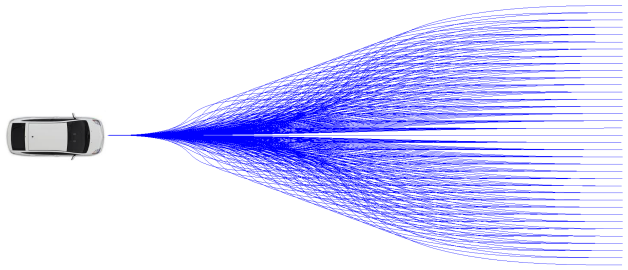
- A prediction of the vehicles final state (using Euler's method) due to the control input is:

$$x_f(p, x) = x_i + \int_{t_0}^{t_f} \dot{x}(x, p)dt$$

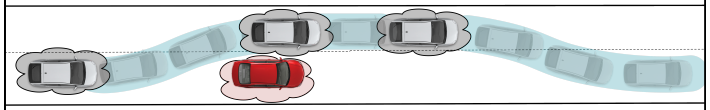
- The goal of the algorithm is to drive the following equation to zero:

$$C(x, p) = x_c - x_f(p, x)$$

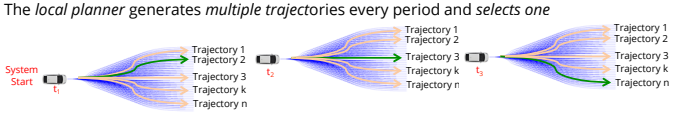
Trajectory Generation



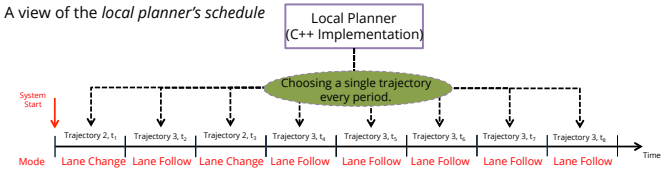
Trajectory Generation



The *local planner* generates *multiple trajectories* every period and *selects one*

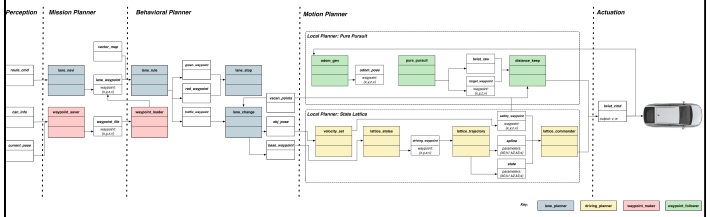


A view of the *local planner's schedule*

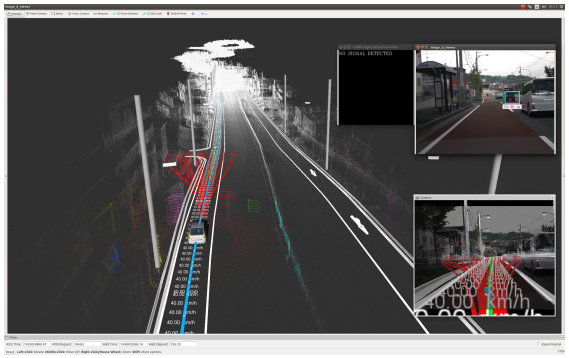


27

Planning Architecture

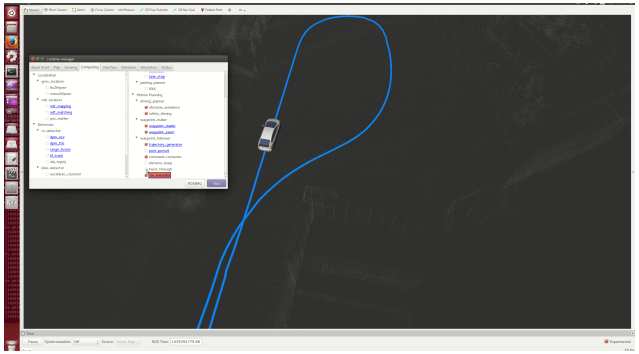


Simulation Environment



29

Vehicle Test



30

Vehicle Test

31

Development of Hybrid Systems Framework

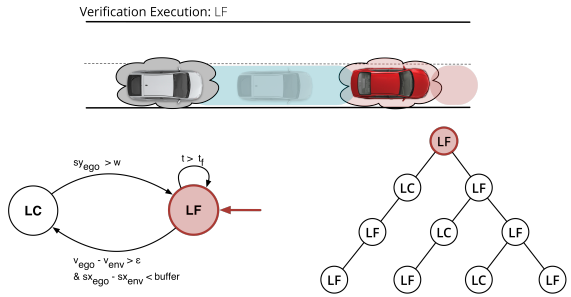
32

Why go this route?

- Autonomous vehicles are ***mixed discrete-continuous systems***.
- Their controllers make discrete decisions which cause the *vehicle* to *evolve in a continuous state space* according to ordinary differential equations.
- Hybrid systems are the natural framework to express this behavior.
- Utilizing the formal semantics of hybrid systems will *allow the definition of simulations and verification tasks in the same language and framework*.
- Hybrid systems provide *a rich enough* ***intermediate representation*** to allow for further *abstraction* if necessary.
- This modeling framework most *closely corresponds* with the *controllers* which will run on real vehicles.

33

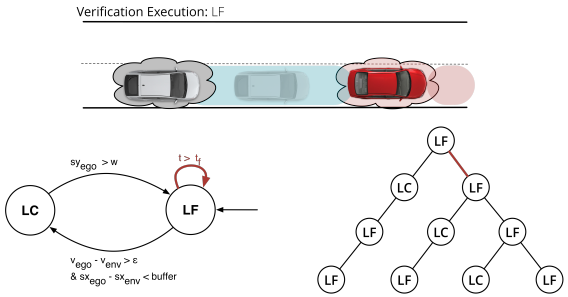
Behavior Controller Execution



Start in the ***Lane Following*** mode.

34

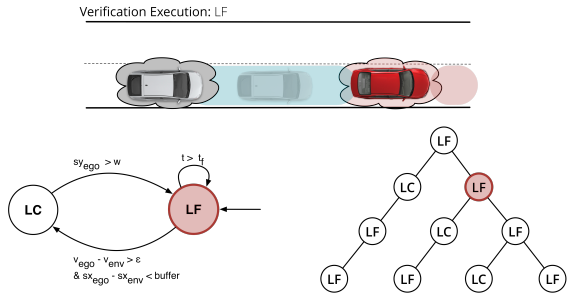
Behavior Controller Execution



If the current time is greater than the ***schedule period*** *update* the ***reference trajectory***…

35

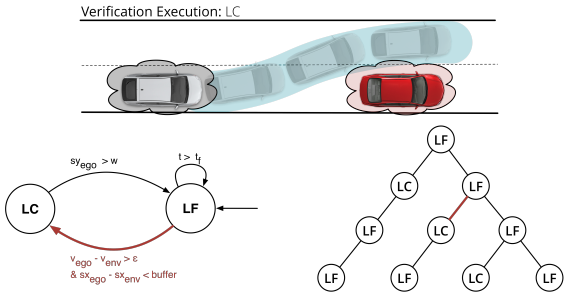
Behavior Controller Execution



Continue in the ***Lane Following*** mode following updated *reference trajectory*.

36

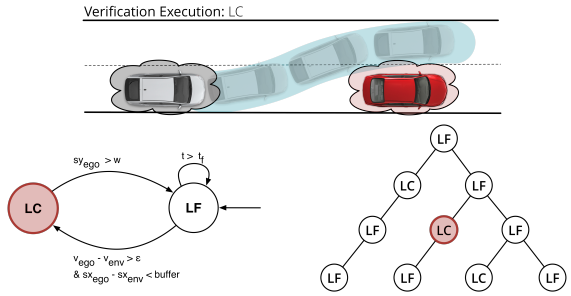
Behavior Controller Execution



If the *ego-vehicle* is within the ***buffer*** and the difference in velocities is big enough *transition*.

37

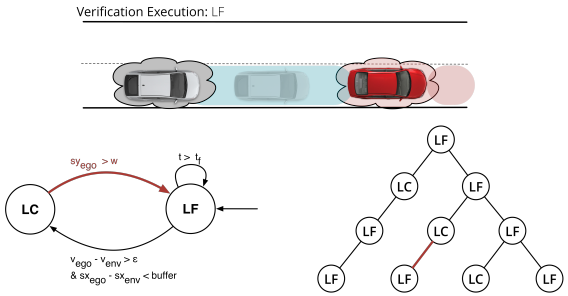
Behavior Controller Execution



Enter ***Lane Change*** mode and follow new *reference trajectory*.

38

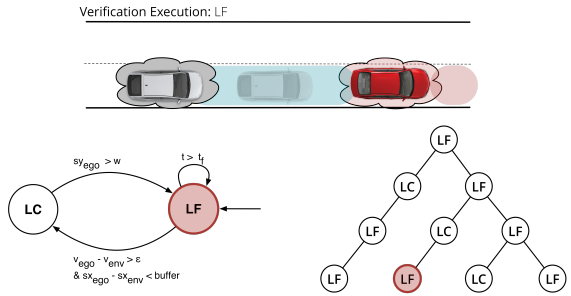
Behavior Controller Execution



If the ego-vehicle has entered new lane update *reference trajectory* and execute *transition*.

39

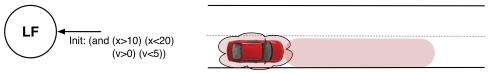
Behavior Controller Execution



Continue in the ***Lane Following*** mode following updated *reference trajectory*.

40

Other Drivers



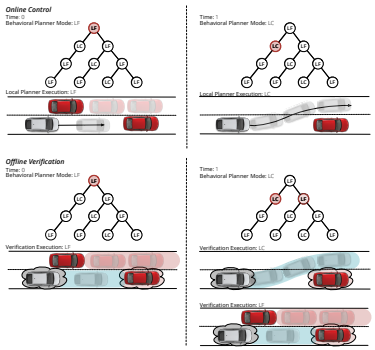
- For this study we begin with a simple model of the other cars in the environment.
- Initially expressivity of modeling language limited us to this representation.
- We do not know the control inputs for the other vehicles so here it is fruitless to include a more expressive dynamical model
- We only care about the path which the other vehicle will take
- In this case we allow non-determinism in the initial estimate of the vehicles velocity.
- Real multi-object tracking perception algorithms assume constant velocity between each position update, thus this is not a bad model of the ego-vehicles view of the world.
- We will show later how more complexity can be added to the non-deterministic behavior of the other agents in the environment by taking unknown control inputs as differential inclusions.

41

Verifying a Lane Change

42

Lane Change



- The ego vehicle is driving in the right lane of a uni-directional two lane road network.
- Another car is driving in front of the ego vehicle at a lower speed.
- We include the extreme case where the environmental vehicle stops.
- The ego vehicle's planner must eventually initiate a lane change maneuver, which involves moving to the left lane, in order to enter a left hand turn lane at a four way stop.
- We highlight that when there is significant uncertainty regarding the ego vehicles orientation and that it may deviate (initially) from the reference trajectory

43

Specification

- The ego vehicle travels at a velocity less than or equal to the speed limit

$$\Box\,(v_{ego} \leq v_{limit}) \tag{29}$$

- The ego vehicle does not drive backwards

$$\Box\,(v_{ego} \geq 0) \tag{30}$$

- The ego vehicle does not collide with any of the n other objects in the environment

$$\Box\left(\sqrt{(s_{x_{ego}} - s_{x_{env_i}})^2 + (s_{y_{ego}} - s_{y_{env_i}})^2} \geq r\right) \quad \forall i = 1...n \tag{31}$$

- If a timed lane change request is invoked, the ego vehicle completes the lane change on time.

$$\Box\,(LC \rightarrow (s_{y_{ego}} > w) \wedge (t \leq t_{max})) \tag{32}$$

- Acceleration ceases when some maximum velocity is reached.

$$\Box\,(v_{env} \geq v_{max} \rightarrow a = 0) \tag{33}$$

- Other agents must drive in the proper direction according to their lane.

$$\Box\,(v_{env} \geq 0) \tag{34}$$

- The accelerations of other agents are within those rates achievable by maximum engine power

$$\Box\,(a_{env} \leq a_{max}) \tag{35}$$

- Other agents maintain their lanes unless explicitly specified not to.

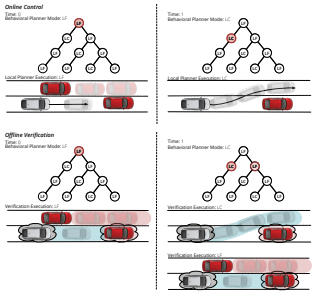
$$\Box\,(\neg LC \rightarrow (y_{min} \leq s_{y_{env}}) \wedge (y_{max} \geq s_{y_{env}})) \tag{36}$$

- Lane changes by other agents are only permitted if the alternate lane is unoccupied or unless a degenerate scenario is being modeled.

$$\Box\,(LO \rightarrow \neg LC) \tag{37}$$

44

Verification Algorithm



Algorithm: External Trajectory Generation

safety_var := SAFE
search_depth :=x

WHILE (depth<search_depth)
safety_var = Check kth trajectory
IF safety_var= SAFE, **then** initialize k+1th trajectory with goal set of kth trajectory and update depth_counter if all nodes at a depth have been visited…
ELSE IF safety_var:=UNSAFE, **then return** safety_var
END WHILE, return safety_var

dReach Verification Instance

An example dReach file that is *automatically* converted from one trajectory generated from the local planner

```
//Trajectory
#define s (22.681867) #define kappa_0 (0.000000) #define kappa_1 (0.033194)
#define kappa_2 (-0.033194) #define kappa_3 (0.000000)

// Constants
#define m 2273.0  #define Iz 4423.0  #define Cf 108000.0 #define Cr 108000.0
#define lf 1.292  #define lr 1.515  #define k1 2.0  #define k2 12.0  #define k3 4.0
#define k4 2.0  #define k5 1.0  #define k6 10.0

// Scenario parameters
#define w 3.7  // Width of road
#define v_d 11.11  // Desired velocity
#define a_env 0.0  // Acceleration of environmental vehicle.

// Helpers to clean up equations...
#define cl (Cr*lr-Cf*lf)

// Tracking controller inputs
// Rate of change of front wheel angle
#define vw k1*(cos(Psi_d)*(sy_d - sy) - sin(Psi_d)*(sx_d - sx)) +k2*(Psi_d - Psi)
+k3*(Psi_dot_d - Psi_dot) -k4*(delta)

// Longitudinal acceleration
#define ax  k5*(cos(Psi_d)*(sx_d - sx) + sin(Psi_d)*(sy_d - sy)) +k6*(v_d-v)

// Cubic spline inputs
#define a (kappa_0)
#define b ((-0.50)*(-2*kappa_3 + 11*kappa_0 - 18*kappa_1 + 9*kappa_2)/s)
#define c ((4.50)*(-kappa_3 + 2*kappa_0 - 5*kappa_1 +4*kappa_2)/(s*s))
#define e ((-4.50)*(-kappa_3 + kappa_0 - 3*kappa_1 + 3*kappa_2)/(s*s*s))

// Bounds on variables
[0,3.1] tau;  [-2.2,0.3] Beta;  [-2,2.04] Psi;  [-2,2.05] Psi_dot;
[6,12] v;  [0,50] sx;  [0,10] sy;  [-2, 2.04] delta;  [0, 3.1] time;

[-2,2.06] kappa_d;  [-2,2.07] Psi_d;  [0,50] sx_d;  [0,10] sy_d;  [-2,2.08] Psi_dot_d;
[9, 50] sx_env;
```

```
// Dynamics
{
   // Lane change motion primitive
   mode 1;

   invt:
   // X position must be greater than or equal to 0
   (sx>=0);
   // Velocity must be greater than or equal to 0
   (v>=0);

   flow:
   /* TARGET VEHICLE */
   /* Plant equations */
   d/dt[tau]=1; // Time
   // Compute rate of change of the slip angle
   d/dt[Beta]=((cl/(m*v*v))-1.0)*Psi_dot + (2*delta/(m*v)) - (Cf+Cr)*Beta/(m*v);
   d/dt[Psi]=Psi_dot; // Rate of change of the heading angle is equal to yaw rate

   // Rate of change of yaw rate
   d/dt[Psi_dot]=(c*Beta/Iz) - ((lf*lf*Cf+lr*lr*Cr)/Iz)*(Psi_dot/v) + ((lf*Cf)*delta/Iz;
   d/dt[v]=ax; // Longitudinal acceleration
   d/dt[sx]=v*cos(Beta+Psi); // X component of velocity
   d/dt[sy]=v*sin(Beta+Psi); // Y component of velocity
   d/dt[delta]=vw; // Rate of change of front wheel angle.

   /*Trajectory and Tracking Equations*/
   d/dt[kappa_d] = b*v_d + 2*c*v_d*v_d*tau + 3*e*v_d*v_d*v_d*tau*tau ;
   d/dt[Psi_d] = v_d*kappa_d;
   d/dt[sx_d] = v_d*cos(Psi_d);
   d/dt[sy_d] = v_d*sin(Psi_d);
   d/dt[Psi_dot_d] = v_d*(b*v_d + 2*c*v_d*v_d*tau + 3*e*v_d*v_d*v_d*tau*tau);

   // Other Vehicle
   d/dt[sx_env]=0.0;

   jump:
}
init:
@1 (and (sx>=0) (sx<=0.5) (sy>=0) (sy<=0.1) (v>=10.8) (v<=11.1) (sx_d=0) (sy_d=0)
   (Psi>=0) (Psi<=0.01) (Psi_dot=0) (Beta=0) (delta=0) (tau=0)
   (Psi_d=0) (Psi_dot_d=0)
   (sx_env=15) (kappa_d=0));

goal:
@1 (or (and (sy<w) (tau>2) (tau<3)) (and (sy<w) (sx>sx_env)));
```


Lane Change

Symbol	Note	Old Assignment	New Assignment
w	Lane Width	-	3.7m
ε	Threshold of the relative speed	-	1 m/s
Buffer	Distance from the ahead vehicle	-	15 m
tf	Max timestep to generate the trajectories	-	3.0 s
V	Velocity of Ego Vehicle	-	[10.8, 11.1]
Sx	Position of Ego Vehicle, X	-	[0.0, 0.5]
Sy	Position of Ego Vehicle, Y	-	[0.0, 0.1]
Φ	Orientation of Ego Vehicle, rad	-	[0.0, 0.1]

- Scenario
  - The vehicle ahead suddenly stops during lane following
- Verification Result: UNSAFE
  - (i.e.) the vehicle may crash the ahead vehicle)
- Analysis (Based on the counter example)
  - Buffer (15m) is too small to change the lane
  - Initial position of the vehicle is heading slightly toward right before starting lane changing
  - The dotted line shows the intended trajectory, the solid line shows the actual trajectory
  - Error due to poor state estimation

Lane Change: Part 2

Symbol	Note	Old Assignment	New Assignment
w	Lane Width	3.7m	3.7m
ε	Threshold of the relative speed	1 m/s	1 m/s
Buffer	Distance from the ahead vehicle	15 m	20 m
tf	Max timestep to generate the trajectories	3.0 s	3.0 s
V	Velocity of Ego Vehicle	[10.8, 11.1]	[10.9, 11]
Sx	Position of Ego Vehicle, X	[0.0, 0.5]	[0.0, 0.5]
Sy	Position of Ego Vehicle, Y	[0.0, 0.1]	[0.0, 0.05]
Φ	Orientation of Ego Vehicle, rad	[0.0, 0.1]	[0.0, 0.1]

- Scenario
  - The vehicle ahead suddenly stops during lane following
- Verification Result: SAFE
  - (i.e.) the vehicle may crash the ahead vehicle)
- Analysis (Based on the counter example)
  - Buffer (20 m) is enough to change the lane
  - Initial position of the vehicle is heading slightly toward right before starting lane changing

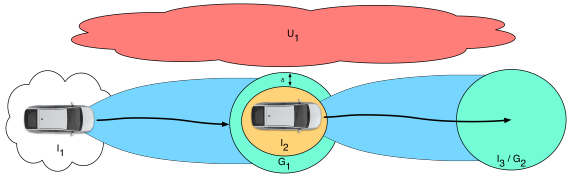
Lane Change: Part 3

Symbol	Note	Old Assignment	New Assignment
w	Lane Width	3.7m	3.7m
ε	Threshold of the relative speed	1 m/s	1 m/s
Buffer	Distance from the ahead vehicle	15 m	20 m
tf	Max timestep to generate the trajectories	3.0 s	3.0 s
V	Velocity of Ego Vehicle	[10.8, 11.1]	[10.9, 11]
Sx	Position of Ego Vehicle, X	[0.0, 0.5]	[0.0, 0.5]
Sy	Position of Ego Vehicle, Y	[0.0, 0.1]	[0.0, 0.05]
Φ	Orientation of Ego Vehicle, rad	[0.0, 0.1]	[0.0, 0.2]

- Scenario
  - The vehicle ahead suddenly stops during lane following
- Verification Result: UNSAFE
  - (i.e.) the vehicle may crash the ahead vehicle)
- Analysis (Based on the counter example)
  - Uncertainty about ego vehicle orientation causes tracking controller to deviate from reference trajectory
  - Demonstrates ability to validate supplier specifications on test scenarios

49

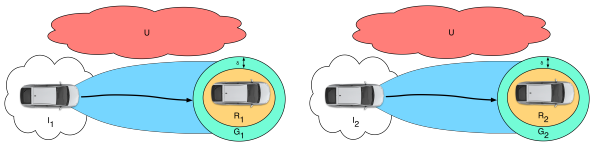
Composition of Trajectories



- The ego-vehicle starts in I_1, I_1 does not intersect U_1
- If verification instance 1 returns SAFE, then X, the ego vehicle's state, never intersects U_1 and the ego-vehicle finishes the run within G_1
- Verification instance 2 must begin within I_2 , I_2 is a subset of G_1. Thus, I_2 does not intersect U_1.
- If verification instance 2 returns SAFE, then X, the ego vehicle's state, never intersects U_1 and the ego-vehicle finishes the run within G_2
- Thus, the ego vehicle never intersects the unsafe region U_1 and we show that the composition of two SAFE bounded time verification instances must imply that the ego vehicle can never intersect U_1 over the combined horizon.
- Key takeaway: If we verify a sequence of two trajectories, the composition of the trajectories is SAFE.
- Key takeaway: The goal sets do not grow.
- Note: Delta is fixed through all verification instances.

50

Verification Instances are Independent



Theorem 1. *Given, I_1, I_2 the initial sets, G_1, G_2 the goal sets, and δ the relaxation parameter of the verification instances V_1, V_2. The verification instances are independent if the initialization, $I_2 \supseteq R_1$. Where R_i is the reachable set of V_i.*

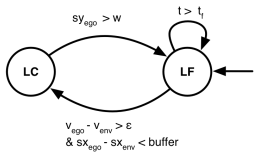
Remark. For each verification instance we call dReach to prove that $R_i \subseteq G_i$. If dReach returns that the program is SAFE, then we are assured that $R_i \subseteq G_i$ and x_t never intersects U.

Proof. We ensure by construction that $I_2 \supseteq R_1$ if $I_2 = G_1^{-\delta}$. Via the verification results we know that G_i does not intersect U, the unsafe region. Furthermore, we know that as the x evolves from t_0 to t it never intersects U and ends in G_{i+1} □

Remark. Independence in this sense implies that we can compose trajectories to develop safe trajectory sequences. Furthermore, it implies that verification instances may be run in parallel.

51

Results: Infinite Time Property



- Behavior planner can express runs of the form LF* or (LF* LC LF*)*
- If the ego-vehicle were to operate on an infinitely long, straight road then the runs could be of infinite length.
- For a run (LF LC) we demonstrate that both LF and LC are safe.
- We have already shown that we can compose LF and LC if sequential trajectories are initialized in the goal set of the previous trajectory. Thus, a finite run can certainly be shown to be safe.
- Because the representation of the behavior planner and motion planner here only admits a finite set of trajectories which are exactly the same in every execution we can show LF, LF, LF, … is safe.
  - ie on a straight road lane following and lane changing trajectories never change.
- Furthermore, it is easy to see that (LF* LC LF*)* is also safe if LF, LC is safe.
- This property will not hold on arbitrarily curved roads.

52

Runtime and Complexity

Table 5: Verification Results

Symbol	Scenario 1	Scenario 2	Scenario 3
w	3.7	3.7	3.7
B	15	20	20
δ	0.1	0.1	0.1
v_{ego}	[10.8, 11.1]	[10.9, 11]	[10.9, 11]
$s_{x_{ego}}$	[0.0, 0.5]	[0.0, 0.5]	[0.0, 0.5]
$s_{y_{ego}}$	[0.0, 0.1]	[0.0, 0.05]	[0.0, 0.05]
Ψ	[0.0, 0.1]	[0.0, 0.1]	[0.0, 0.2]
Search Depth	2	2	2
Verification Time (s)	30.821	373.924	36.166
Result	δ-UNSAFE	SAFE	δ-UNSAFE

53

Conclusions

- Why go to all the trouble to make an AV model?
  - Cannot perform verification without a model
  - Modeling using formal semantics such as "hybrid systems" will allow for bottom up approach to building more scenarios from agent blocks.
  - Modeling using formal semantics such as "hybrid systems" will allow for translation to simulation environments.
- First fully automatic verification of realistic lane changing algorithms and planning stack !
- The counterexamples are intuitive…
  - At the time of writing the modeling language of dReach and dReal did not support key features for true non-determinism.
    - This leaves us to hand engineer strange environment behaviors
  - Yes, this gives us confidence that we are finding real bugs
  - No, it is not good enough yet.
  - We need more complex scenarios, but building CTL like trees with external trajectory generation is not efficient.
  - In the sequel we will show how to get around these two deficits.

54

F1/10

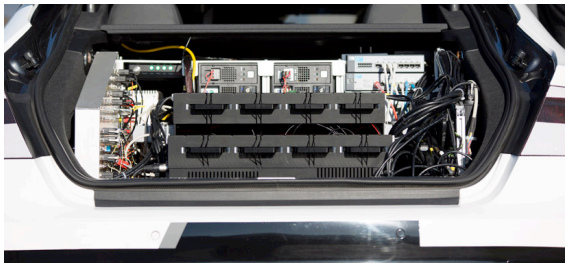
55

Early Autonomous Vehicles

In 2008 AV prototypes, were ***ad-hoc***, ***complex***, and ***expensive***…

56

State of the Art AV



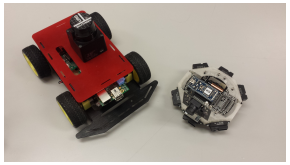
Today industrial grade prototypes have gotten better, but platforms are still ***inaccessible*** and ***dangerous***…

57

Agile AV Platform

1

- We began with simple differential drive robots which did not capture realistic dynamics.
- Progressed from line following to map based navigation.

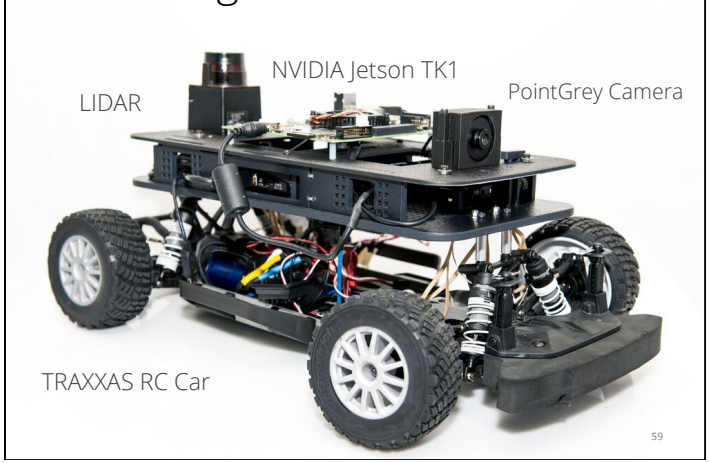


2

- Powerful non-holonomic robot has more realistic dynamics.
- Utilizes embedded GPU to run realistic planning, perception, and control algorithms.

58

Agile AV Platform



59

Demo

60