Automatic Testing with Dynamically Constrained Reinforcement Learning

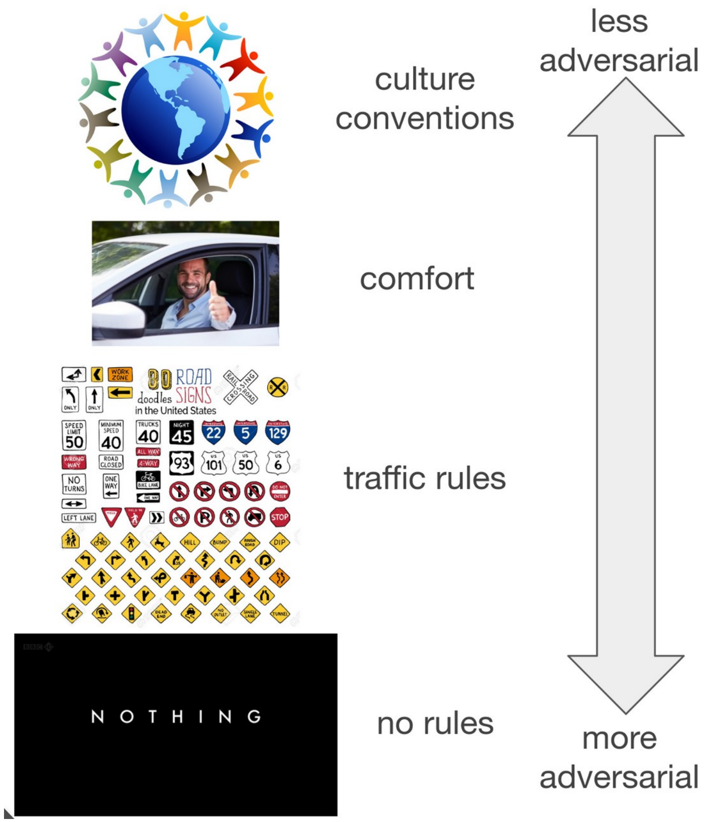
Xin Qin, Nikos Arechiga, Andrew Best, Jyotirmoy Deshmukh

Paper preprint: https://arxiv.org/pdf/1910.13645.pdf

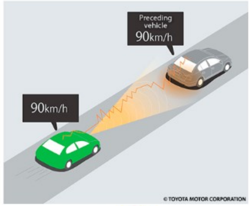
TOYOTA RESEARCH INSTITUTE

Controllable AI for automatic testing

- Want an adversarial agent that can automatically learn to cause ego to make a mistake.
- Learn to respect rules chosen from hierarchical rulebooks (a la nutonomy):
  - Specify which rules ado must follow,
  - Prevent unreasonable behavior, such as driving the wrong way down the freeway.
- We call these logical scaffolds.
  - Logical structures around which the AI grows and becomes stronger
- Use logical scaffold to declaratively control the behaviors of the agent.



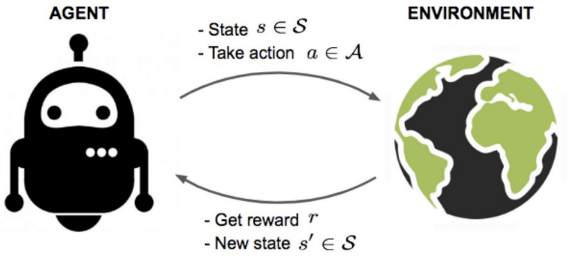
Formal problem statement



- Given
  - A scenario
    - “follow lead car on freeway”, “intersection with yellow light”
  - A target specification
    - “make ego rear-end you”, “make ego run red light”
  - Logical scaffolds that constrain allowed behavior
- Intelligent agent should find an ego specification violation while respecting its constraints

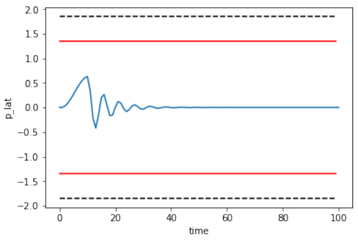
Solution components

- Train: reinforcement learning
  - Tabular
  - Neural network
- Represent logical scaffolds as STL formulas
- Implementations:
  - PyTorch
  - CARLA



"I always stay in my lane"

$$\mathcal{A}(-\ell + \mu \leq x \leq \ell - \mu)$$



Signal Temporal Logic (STL)

s "signals", functions that evolve over time

Functions and inequalities

$$f(s) < c$$

Boolean operators

$$\wedge, \vee, \neg, \rightarrow$$

Temporal operators

$\mathcal{A}$ always

$\mathcal{E}$ eventually

$\mathcal{U}$ until

$\mathcal{A}(-\ell + \mu \leq x \leq \ell - \mu)$

"I always stay in my lane"

$\mathcal{A}(brake = 1 \rightarrow \mathcal{E}_{[0,0.1]} a = decel)$

"If the driver presses brake pedal, eventually after at most 0.1 seconds, I apply deceleration"

Boolean and quantitative semantics

- Boolean: is the property true at each point in time?
- Quantitative: how true is the property? (“robust satisfaction”)
  - For inequalities, difference between left and right
  - For “and”, smallest robustness of subformulas
  - For “or”, largest robustness of subformulas
  - For “eventually”, largest robustness over the trace
  - For “always”, smallest robustness over the trace
  - “Until” is an “always”, until the second condition is true

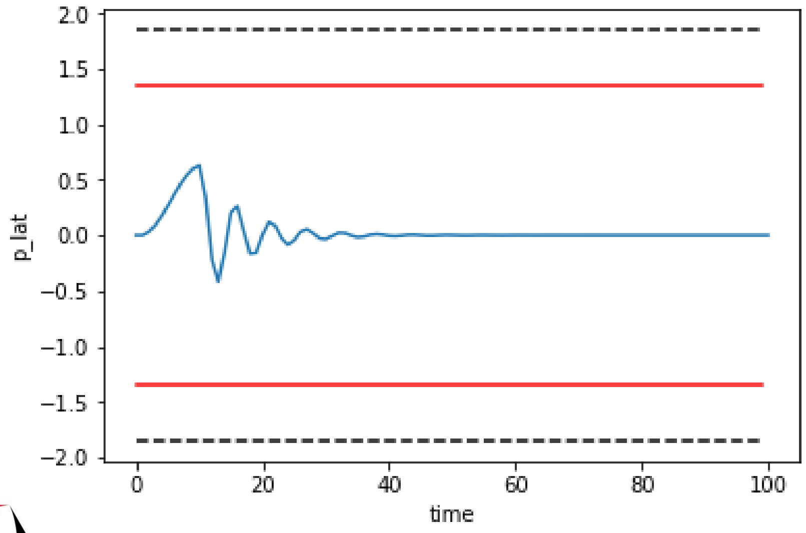
TOYOTA
RESEARCH INSTITUTE

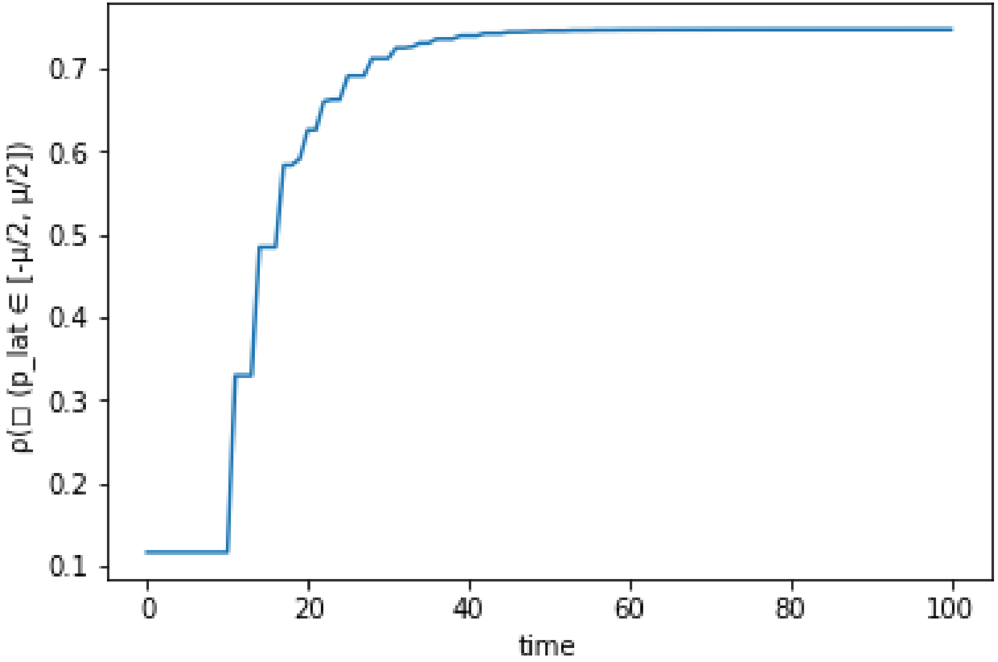
Monitors can be automatically synthesized

```
drive_in_lane = ((left_lane_boundary + margin <= x_lat - 0.5*car_width)
                 & (x_lat + 0.5*car_width <= right_lane_boundary - margin))

aw_drive_in_lane = stl.Always( drive_in_lane )

aw_drive_in_lane.plot(trace)
```





TOYOTA
RESEARCH INSTITUTE

STL @ Toyota

Mining Requirements from Closed-Loop Control Models

Xiaoqing Jin
Univ. of California Riverside
jinx@cs.ucr.edu

Alexandre Donzé
Univ. of California Berkeley
donze@eecs.berkeley.edu

Jyotirmoy V. Deshmukh
Toyota Technical Center
jyotirmoy.deshmukh@tema.toyota.com

Sanjit A. Seshia
Univ. of California Berkeley
sseshia@eecs.berkeley.edu

Robust Online Monitoring of Signal Temporal Logic

Jyotirmoy V. Deshmukh[1], Alexandre Donzé[2], Shromona Ghosh[2]
Xiaoqing Jin[1], Garvit Juniwal[2], and Sanjit A. Seshia[2]

[1] Toyota Technical Center, firstname.lastname@tema.toyota.com
[2] University of California Berkeley,
{donze, shromona.ghosh, garvitjuniwal, sseshia}@eecs.berkeley.edu

Property-Driven Runtime Resolution of Feature Interactions

Santhana Gopalan Raghavan[1], Kosuke Watanabe[2], Eunsuk Kang[3], Chung-Wei Lin[4], Zhihao Jiang[5], and Shinichi Shiraishi[2]

[1] University of Southern California, USA santhanr@usc.edu
[2] Toyota InfoTechnology Center, USA {kwatanabe,sshiraishi}@us.toyota-itc.com
[3] Carnegie Mellon University, USA eskang@cmu.edu
[4] National Taiwan University, Taiwan cwlin@csie.ntu.edu.tw
[5] ShanghaiTech University, China jiangzhh@shanghaitech.edu.cn

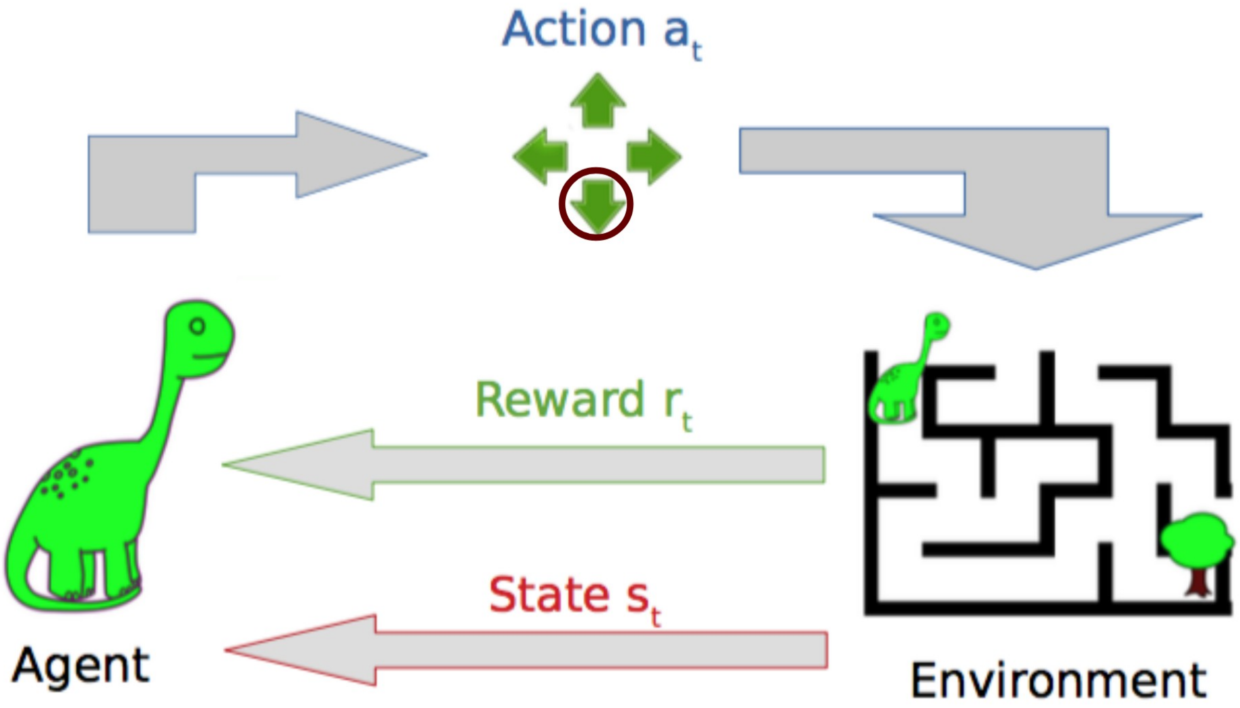
Backpropagation for Parametric STL

Karen Leung,[1] Nikos Aréchiga[2] and Marco Pavone[1]

- Our “weapon of choice” for logical specifications.

- Some groups in Japan also use it.

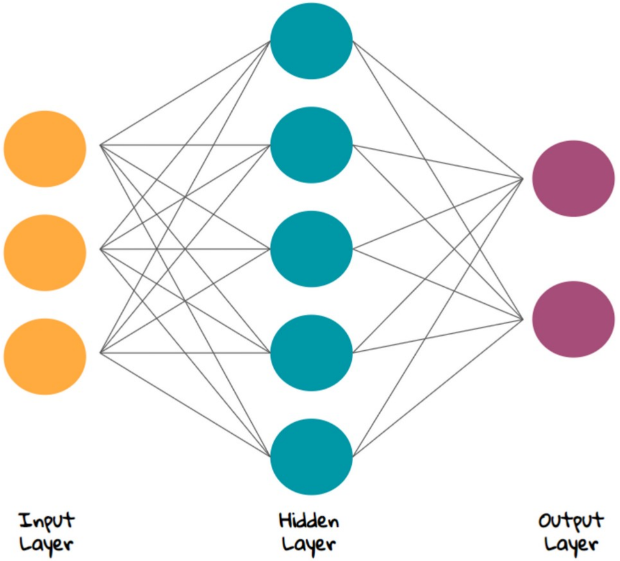
Reinforcement Learning

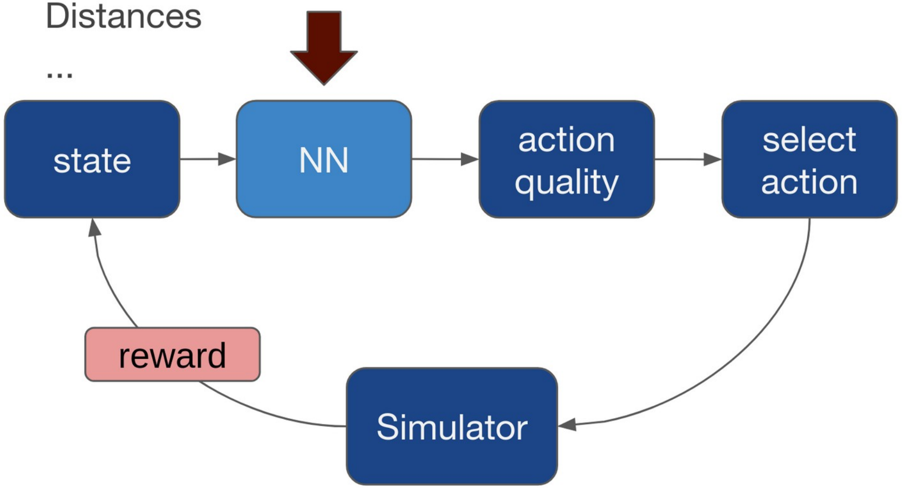


- State
- Actions
- Reward

DQN

- Continuous State:
  - Velocities
  - Distances
  - ...





$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \Big[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t) \Big].$$

Neural networks (hopefully) generalize to unseen states

Implementation

- Implementation on TRI’s internal simulator
  - Due to time constraints of Xin’s internship, used a simplified ego controller
- CARLA implementation
  - For research publication purposes

(a) Case Study I: Driving in lane with lead vehicle.

(b) Case Study II: Left vehicle merges in front.

(c) Case Study III: Yellow light running.

Fig. 1: Simulation environments for case studies

Philosophical results

- A unified framework to declaratively specify *what* should be done
- Techniques to automatically monitor that it is being done
- Use of RL to synthesize agents that automatically learn *how* to do what is required

The future of Loki

- Goal: “To build an evil AI and hire it as a consultant”.
- Algorithmic flexibility: Loki should have a “bank” of algorithms it tries, and selects the one that performs the best on that particular scenario
- Investigate transferability of adversaries across parametric variations of driving stack, and composability of agents across scenarios
- Further case studies: working on integrating Carla and OpenPilot for further development

TOYOTA RESEARCH INSTITUTE