Cyber-Physical Systems

Laura Nenzi

Università degli Studi di Trieste
II Semestre 2019

Lecture 12: Examples

[Many Slides due to J. Deshmukh]

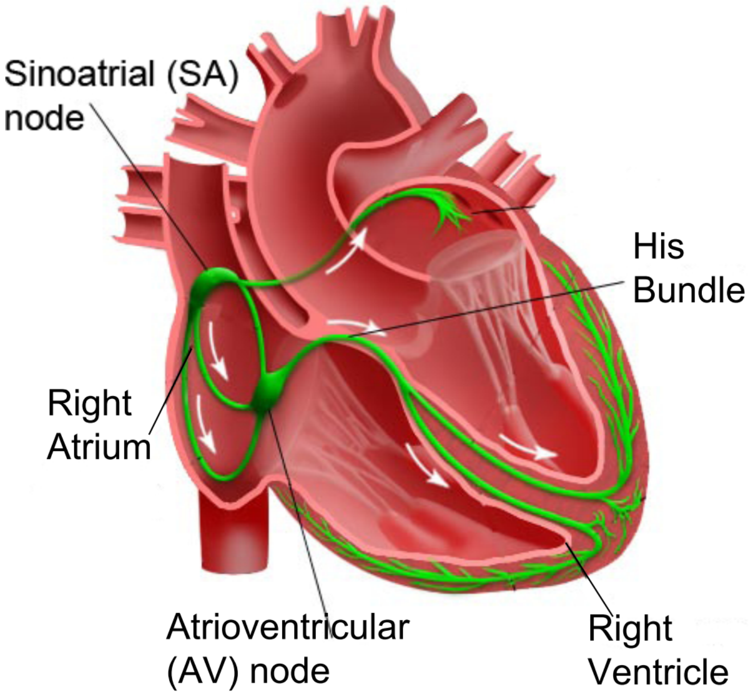
Pacemaker Modeling as a Timed Process

- Most material that follows is from this paper:

Z. Jiang, M. Pajic, S. Moarref, R. Alur, R. Mangharam, *Modeling and Verification of a Dual Chamber Implantable Pacemaker*, In Proceedings of Tools and Algorithms for the Construction and Analysis of Systems (TACAS), 2012.

- The textbook has detailed descriptions of some other pacemaker components

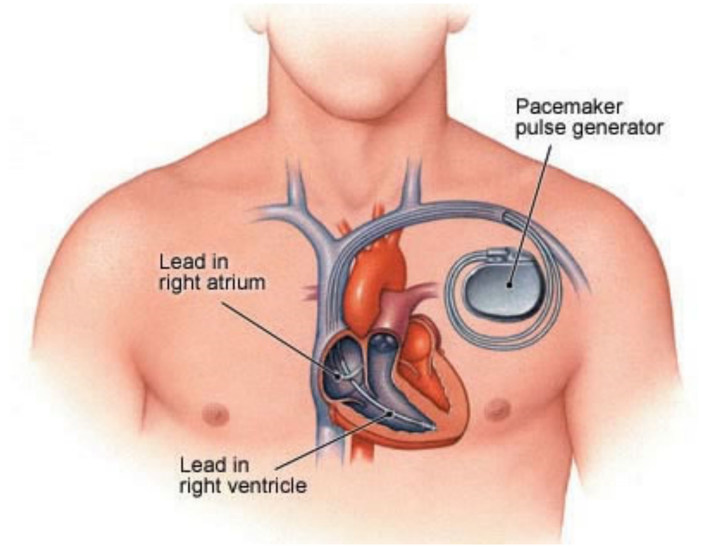
How does a healthy heart work?



Electrical Conduction System of the Heart

- SA node (controlled by nervous system) periodically generates an electric pulse
- This pulse causes both atria to contract pushing blood into the ventricles
- Conduction is delayed at the AV node allowing ventricles to fill
- Finally the His-Pukinje system spreads electric activation through ventricles causing them both to contract, pumping blood out of the heart

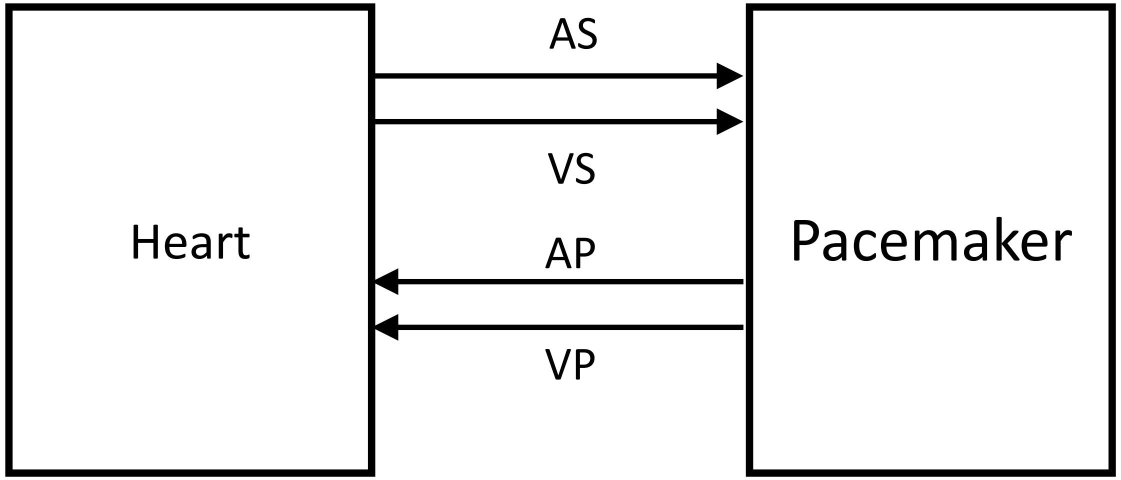
What do pacemakers do?



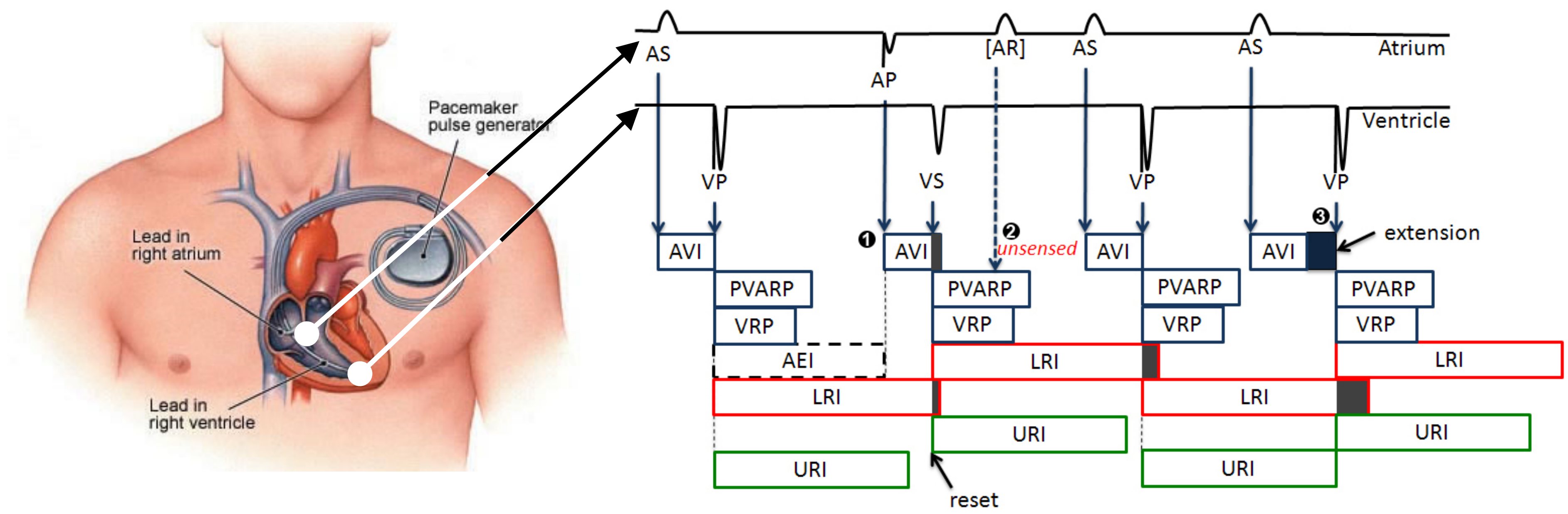
- Aging and/or diseases cause conduction properties of heart tissue to change leading to changes in heart rhythm
- Tachycardia: faster than desirable heart rate impairing hemo-dynamics (blood flow dynamics)
- Bradycardia: slower heart rate leading to insufficient blood supply
- Pacemakers can be used to treat bradycardia by providing pulses when heart rate is low

How dual-chamber pacemakers work

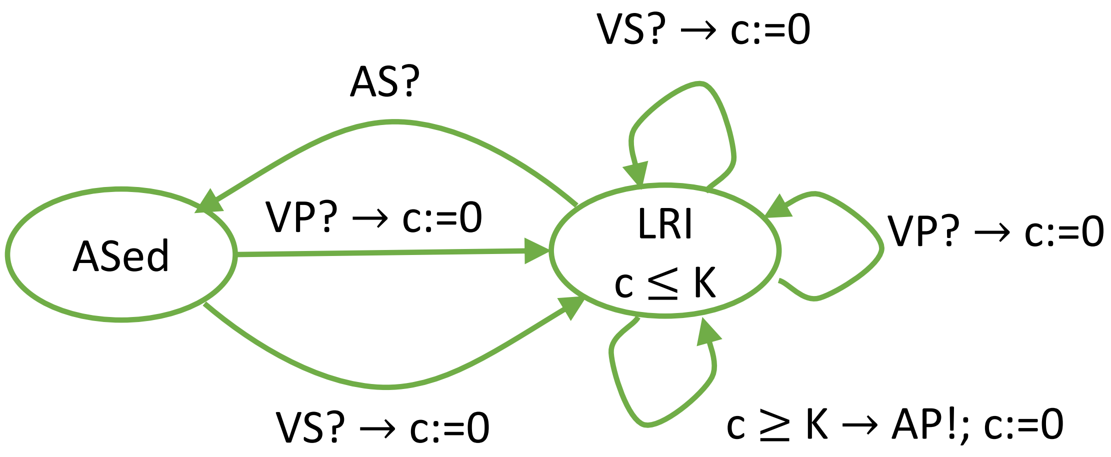
- Two fixed leads on wall of right atrium and ventricle respectively
- Activation of local tissue sensed by the leads (giving rise to events Atrial Sense (AS) and Ventricular Sense (VS))
- Atrial Pacing (AP) or Ventricular Pacing (VP) are delivered if no sensed events occur within deadlines



Implantable Pacemaker modeling



The LRI mode of operation explained



K= 850ms

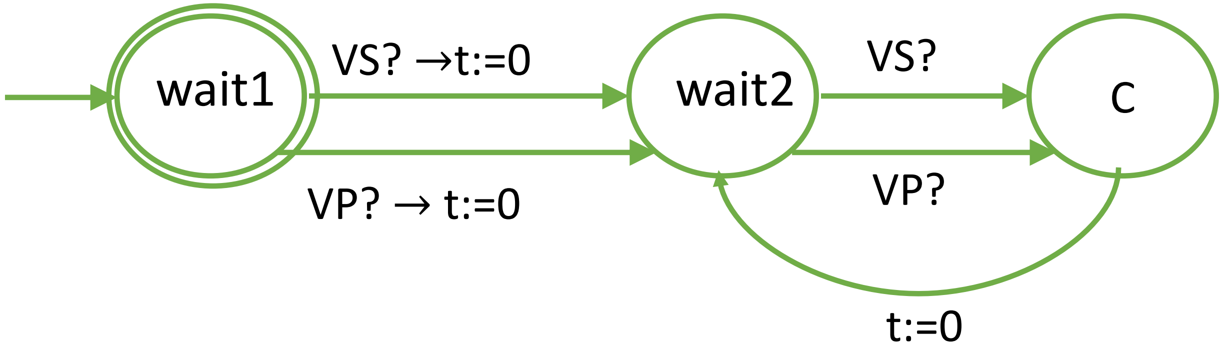
- LRI (Low Rate Interval) component keeps heart rate above minimum level
- One of the pacemaker modes of operation that models the basic timing cycle
- Measures the longest interval between ventricular events
- Clock reset when VS or VP received
- No AS received ⇒ LRI outputs AP after K (TLRI-TAVI) time units

Random Heart Model (RHM)

- RHM is designed to cover open-loop heart behaviors. It non-deterministically generates an intrinsic heart event Xget! within [Xminwait, Xmaxwait] after each intrinsic heart event Xget or pacing XP.

Property

Monitor Pvv



CTL formula : $AG\ (Pvv.\text{wait2} \Rightarrow\ Pvv.c \leq TLRI)$

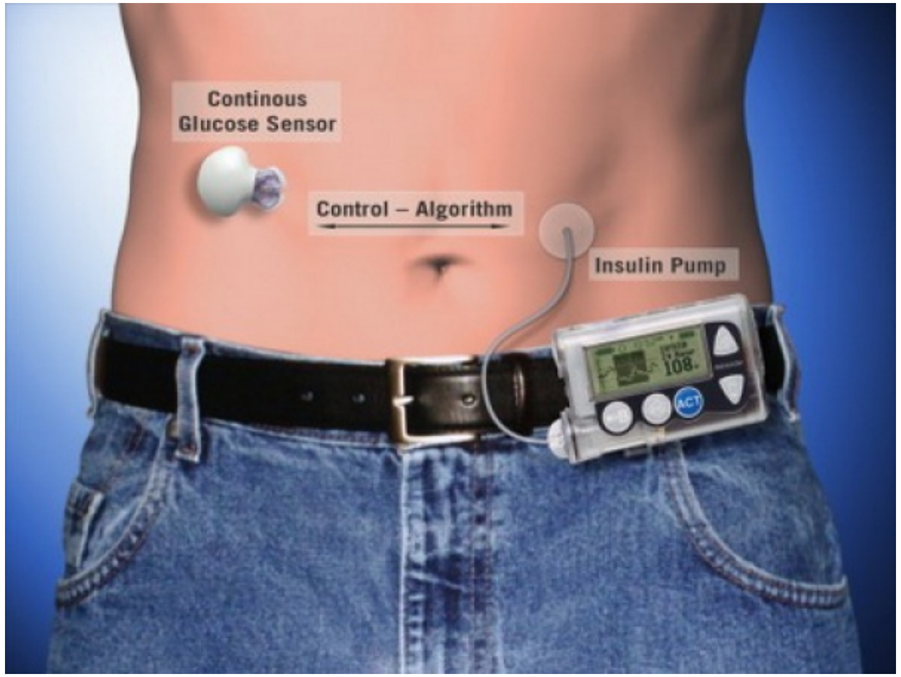
Timed Automata

- Useful tool to do timing analysis and explore properties of timed processes
- Finite-state timed automaton: a machine where all state variables other than clock variables have finite types (e.g. Boolean, enums)
- State-space of timed automata is infinite (clocks can become arbitrarily large!)
- But some questions about timed automata behavior can still be answered exactly

Artificial Pancreas

Type 1 diabetes occurs when the pancreas produces little or none of the insulin needed to regulate blood glucose

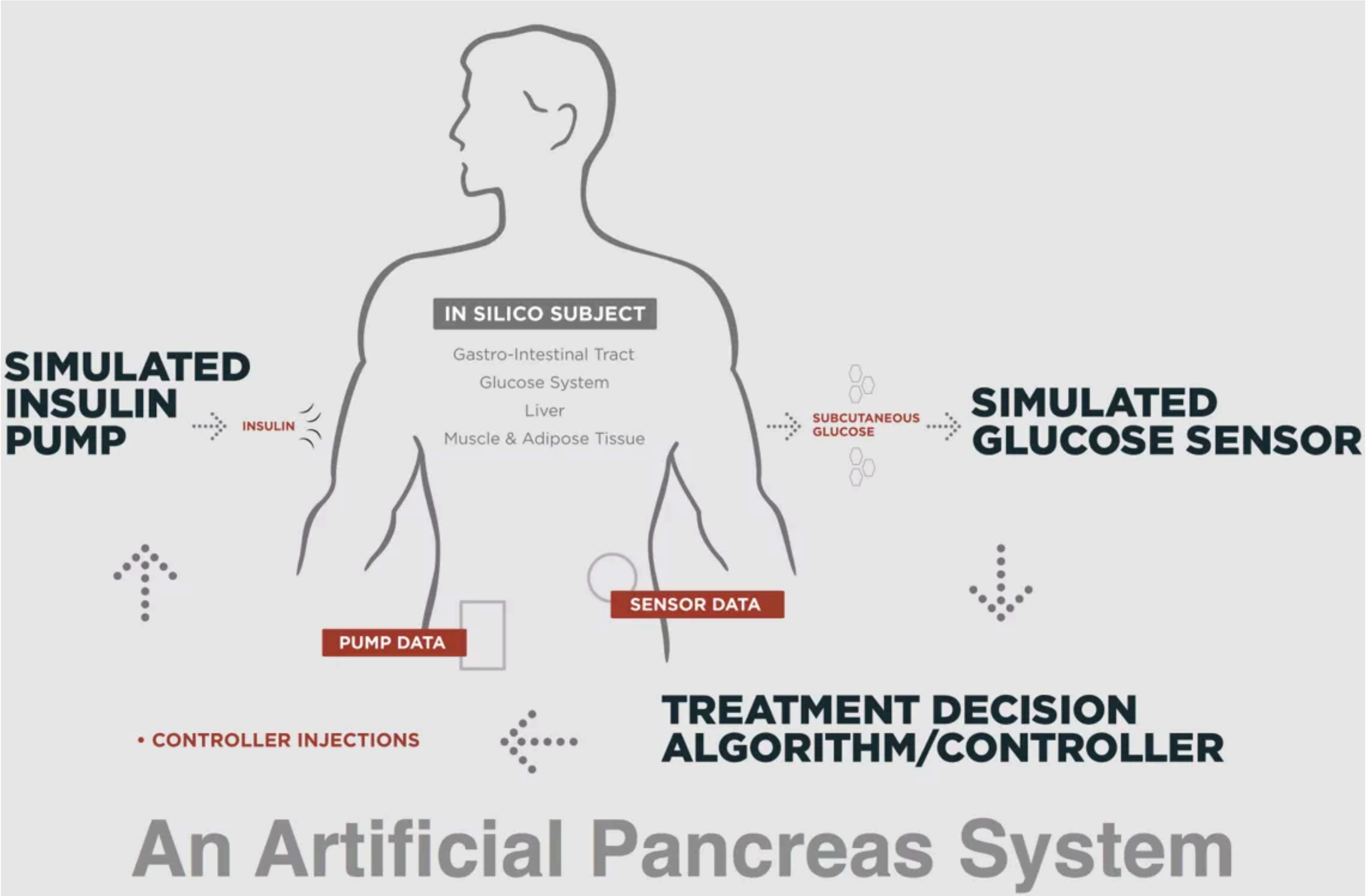
They rely on external ad- ministration of insulin to manage their blood glucose levels.



Most material that follows is from these papers:

- Fedor Shmarov, Nicola Paoletti, Ezio Bartocci, Shan Lin, Scott A. Smolka, Paolo Zuliani: SMT-based Synthesis of Safe and Robust PID Controllers for Stochastic Hybrid Systems. Haifa Verification Conference 2017: 131-14
- Simone Silvetti, Laura Nenzi, Ezio Bartocci, Luca Bortolussi: **Signal Convolution Logic.** CoRR abs/1806.00238 (2018)

Artificial Pancreas



Stochastic Hybrid Systems Of Glucose

$$\frac{d}{dt}\boldsymbol{x}(t) \ = \ F(\boldsymbol{x}(t);\ u(t); \Theta); \qquad y(t) \ = x_{Q1}(t)$$

glucose concentration

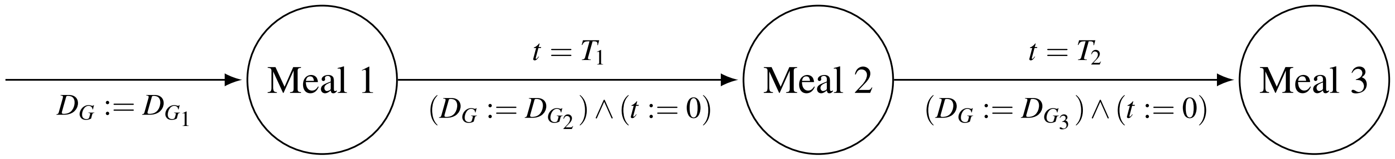
the control parameters

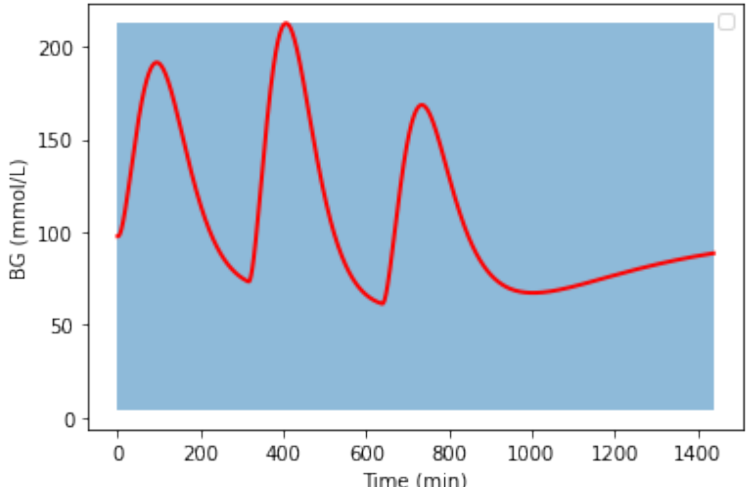
Infusion rate of bolus insulin

$\Theta \ = \ (D_{G_1}; D_{G_2}; D_{G_3};\ T_1;\ T_2)$ are the control parameter

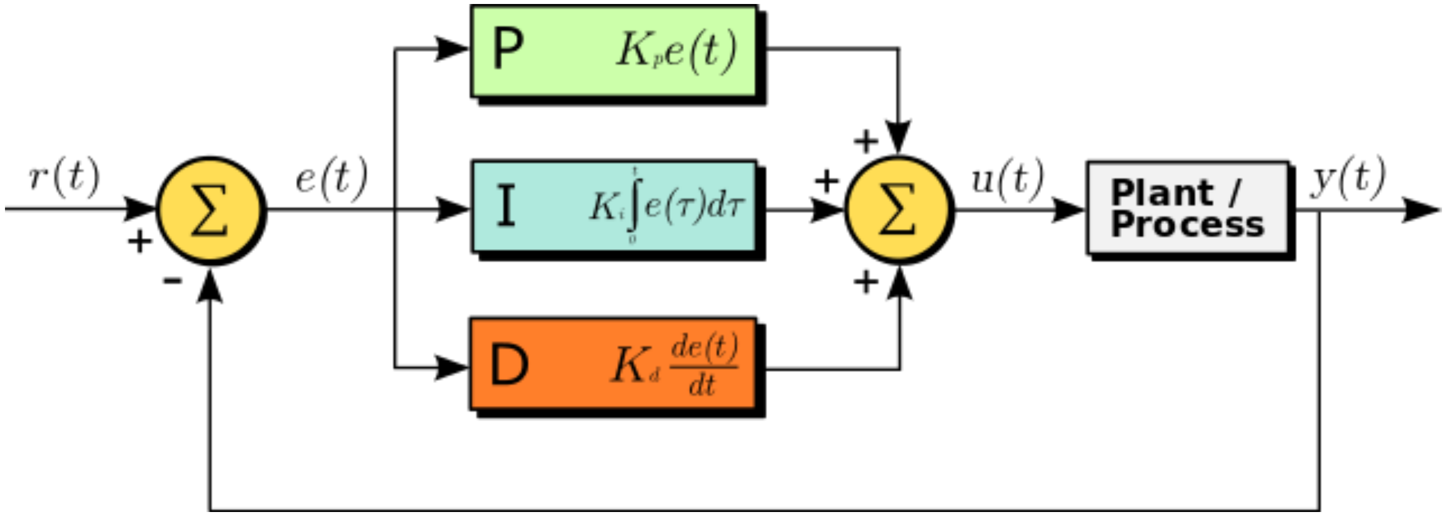
$(D_{G_1}; D_{G_2}; D_{G_3}) \in (N(40;\ 10); N(90;\ 10); N(60;\ 10))$ are the three daily meals

$(T_1;\ T_2) \in \sim N$ (300, 10) and *T2* ~ *N* (300, 10) are the inter-times between each of them


$D_G := D_{G_1}$
Meal 1
$t = T_1$
$(D_G := D_{G_2}) \wedge (t := 0)$
Meal 2
$t = T_2$
$(D_G := D_{G_3}) \wedge (t := 0)$
Meal 3



PID Control



$$u(t) = K_p e(t) + Ki \int_0^t e(\tau)\, d\tau + Ke^{\cdot}(t)\,, \qquad e(t) = r\,(t) - y(t)$$

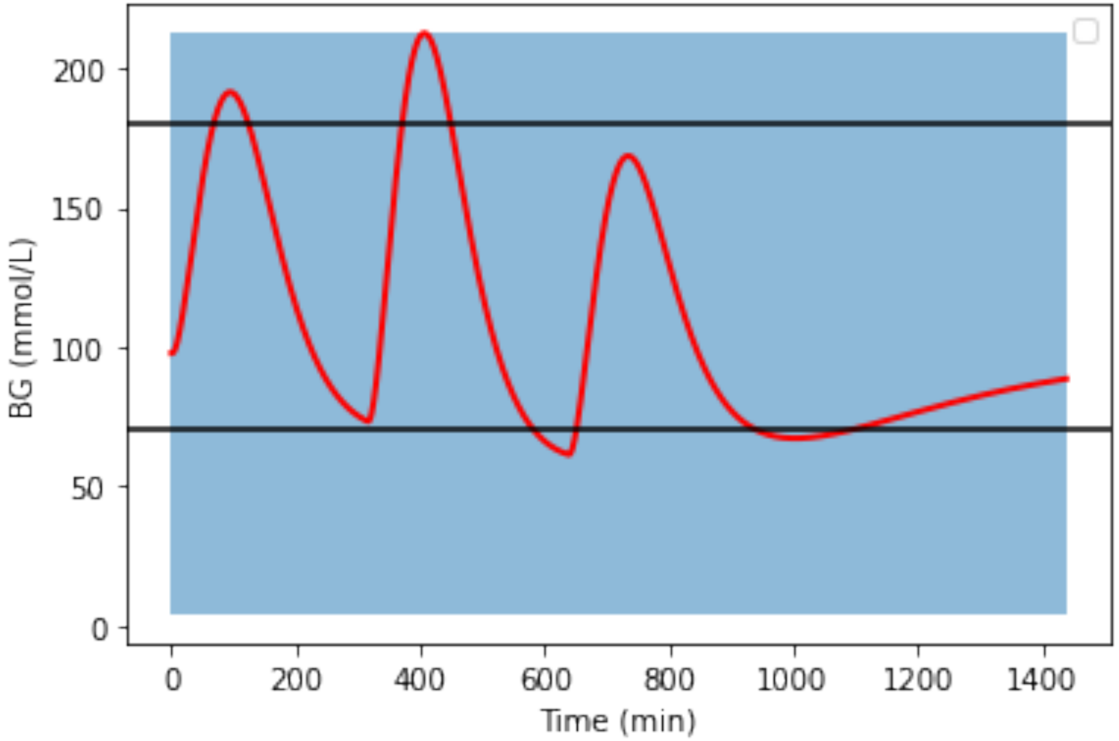
Artificial Pancreas

- Hyperglycemia
  - “during the day the level of glucose goes above 180mg/dl”

$$F_{[0,24h]}(BG(t) > 180)$$

- Hypoglycemia
  - “during the day the level of glucose goes below 70mg/dl”

$$F_{[0,24h]}(BG(t) < 70)$$



Falsification

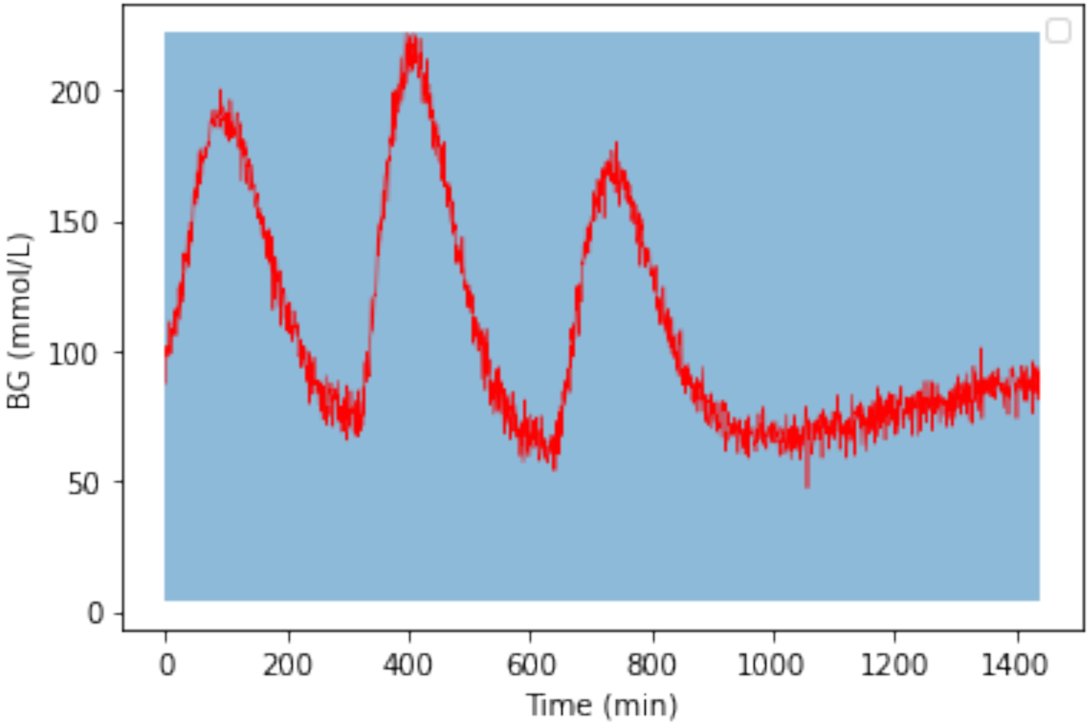
The most simple way to do falsification with respect a property ϕ is minimizing the robustness over N iterations considering random samples on control parameters, i.e:

```
$\mathrm{minSTL} = \text{‘inf’}$
For $\mathrm{i} = 1, \ldots, \mathrm{N}$:
    $\Theta \ = \mathrm{sampling}\ (D_{G_1}, D_{G_2}, D_{G_3}, T_1, T_2)$
    $\mathrm{t}\ ,\mathrm{y} = \mathrm{simulation}(\Theta)$
    $\mathrm{stl} = \mathrm{computeRobustness}(\mathrm{y}, \phi)$
    if $(\mathrm{stl} < \mathrm{minSTL})$:
        $\mathrm{minSTL} = \mathrm{stl}$
        $\mathrm{vSTL} = [D_{G_1}, D_{G_2}, D_{G_3}, T_1, T_2]$
```

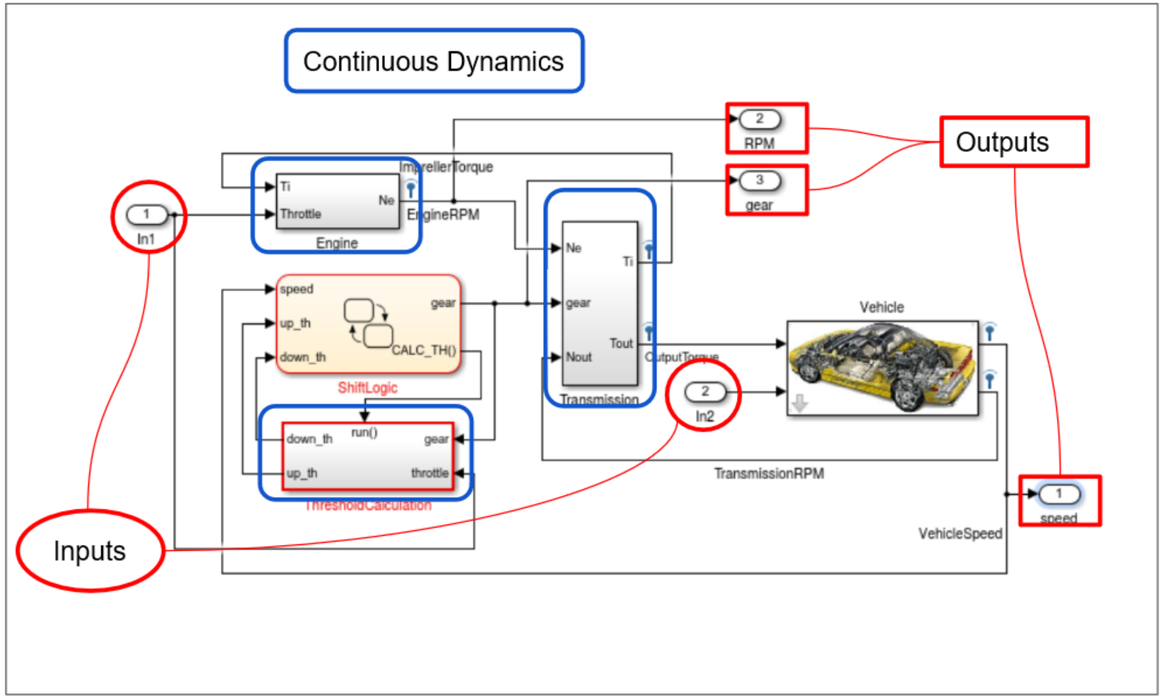
For fixed control parameter spaces you can consider to sample with respect on grids over it.

Noise Robustness

- To consider noisy sensor we can add a Gaussian noise to the generated glucose trajectory, i.e. $GB(t) + \gamma$ with $\gamma \in N(0;\ 5)$



Automatic Transmission

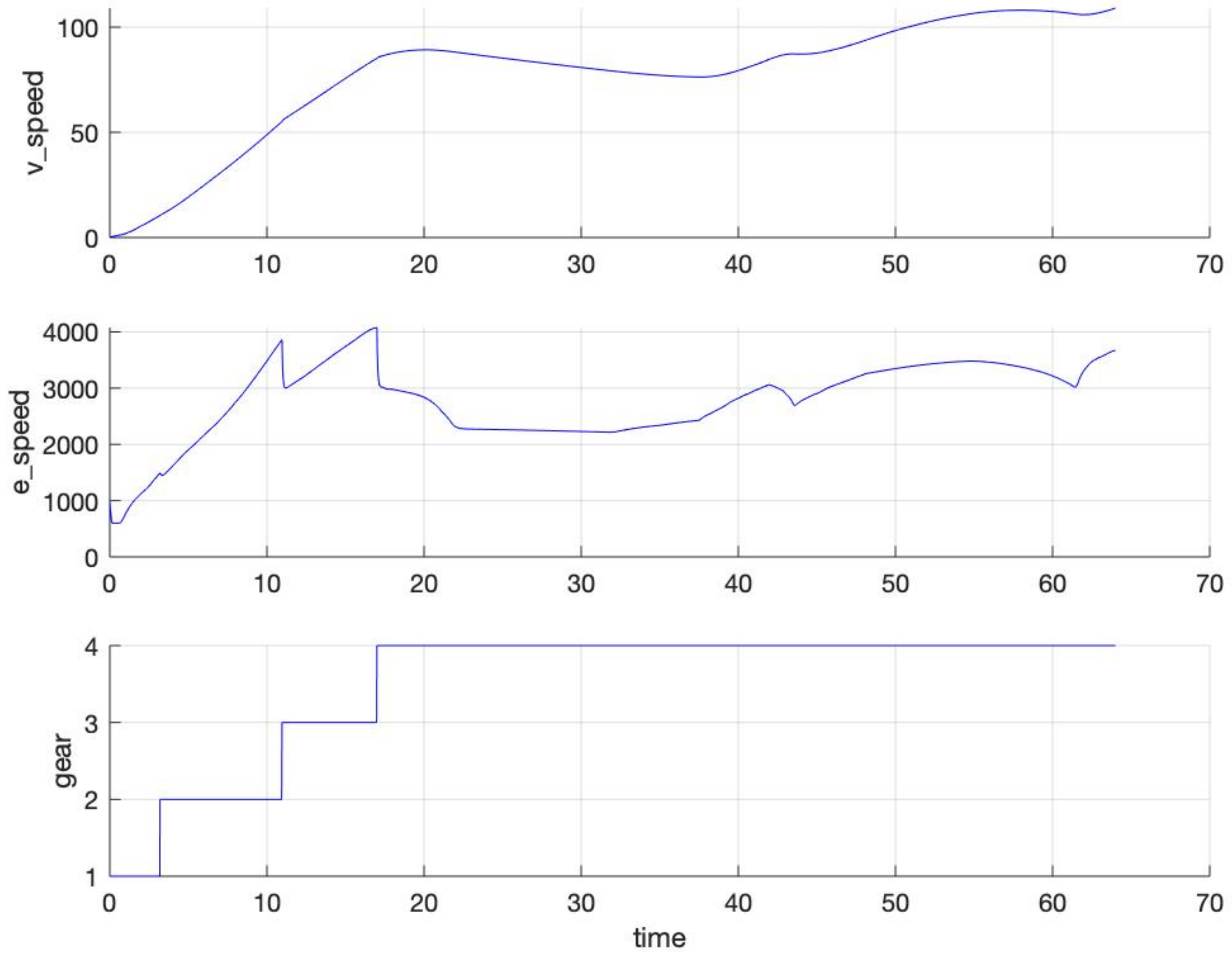


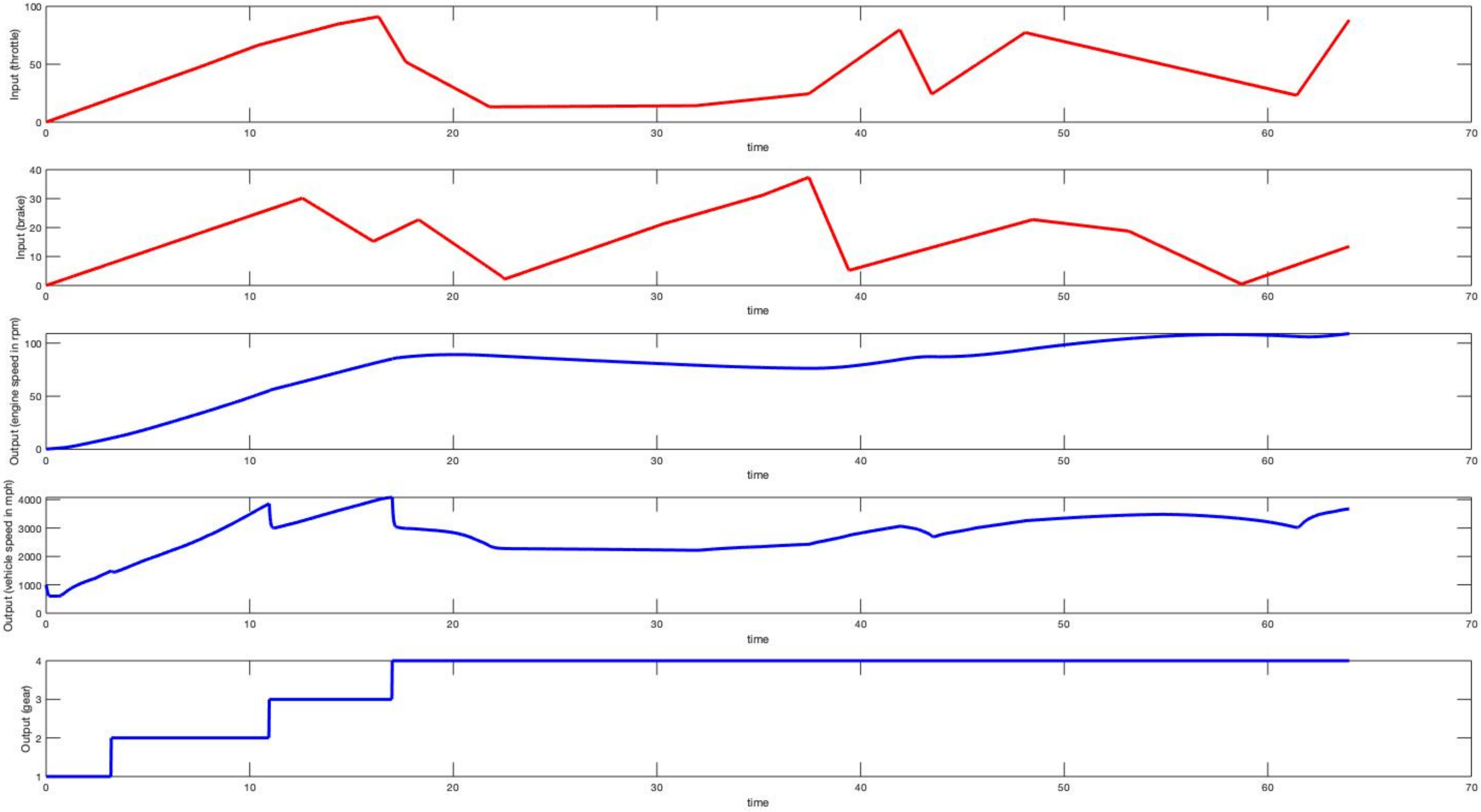
Most material that follows is from this paper:

- Bardh Hoxha, Houssam Abbas, Georgios E. Fainekos: Benchmarks for Temporal Logic Requirements for Automotive Systems. ARCH@CPSWeek 2014: 25-30

Automatic Transmission

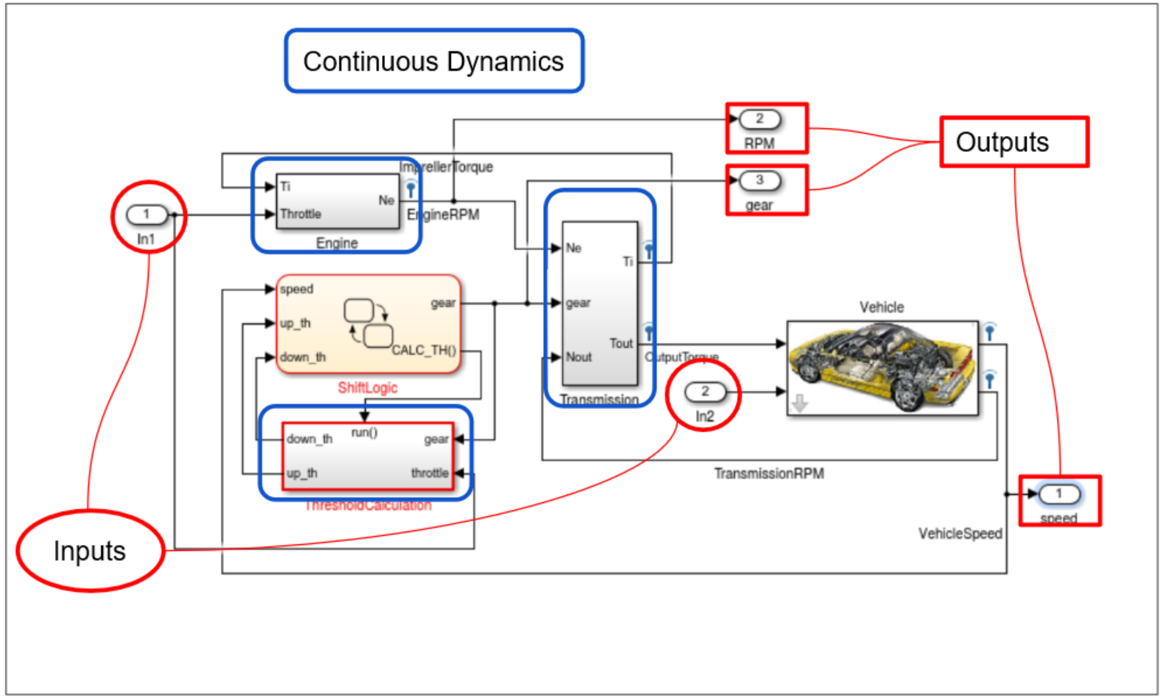
- Inputs: the throttle and break
- Outputs: the speed of the engine ω (RPM), the speed of the vehicle v (mph) and the gear.
- Initially, the vehicle is at rest at time 0, i.e. the speed v = 0 and engine speed ω = 0
- Therefore, the output trajectories depend only on the input signals ut and ub which model the throttle and break inputs.
- The throttle and break, at each point in time, can take any value between 0 (fully closed) to 100 (fully open).

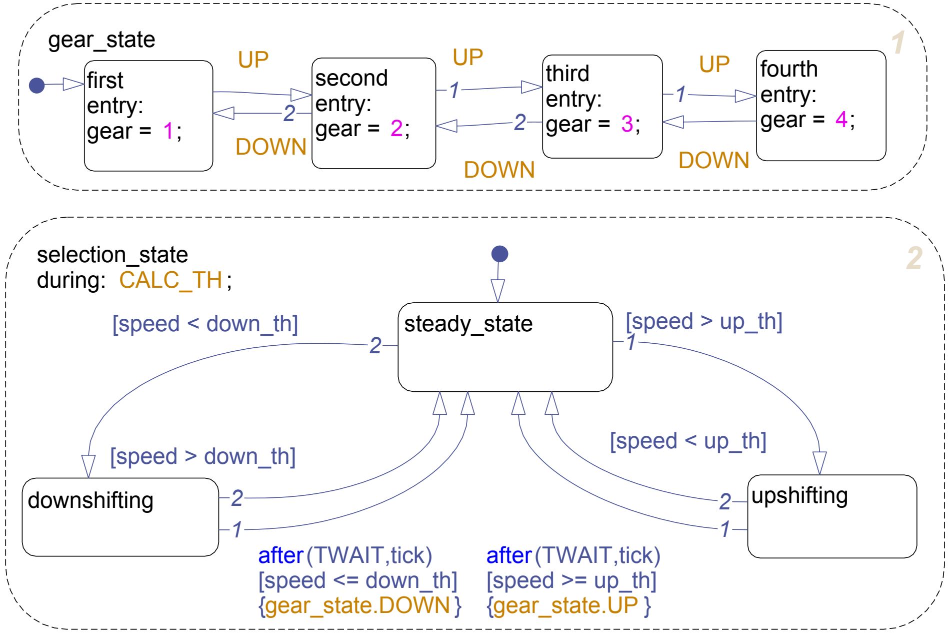



Input (throttle)
100
50
0
Input (brake)
40
30
20
10
0
Output (engine speed in rpm)
100
50
0
Output (vehicle speed in mph)
4000
3000
2000
1000
0
Output (gear)
4
3
2
1
0
10
20
30
40
50
60
70
time

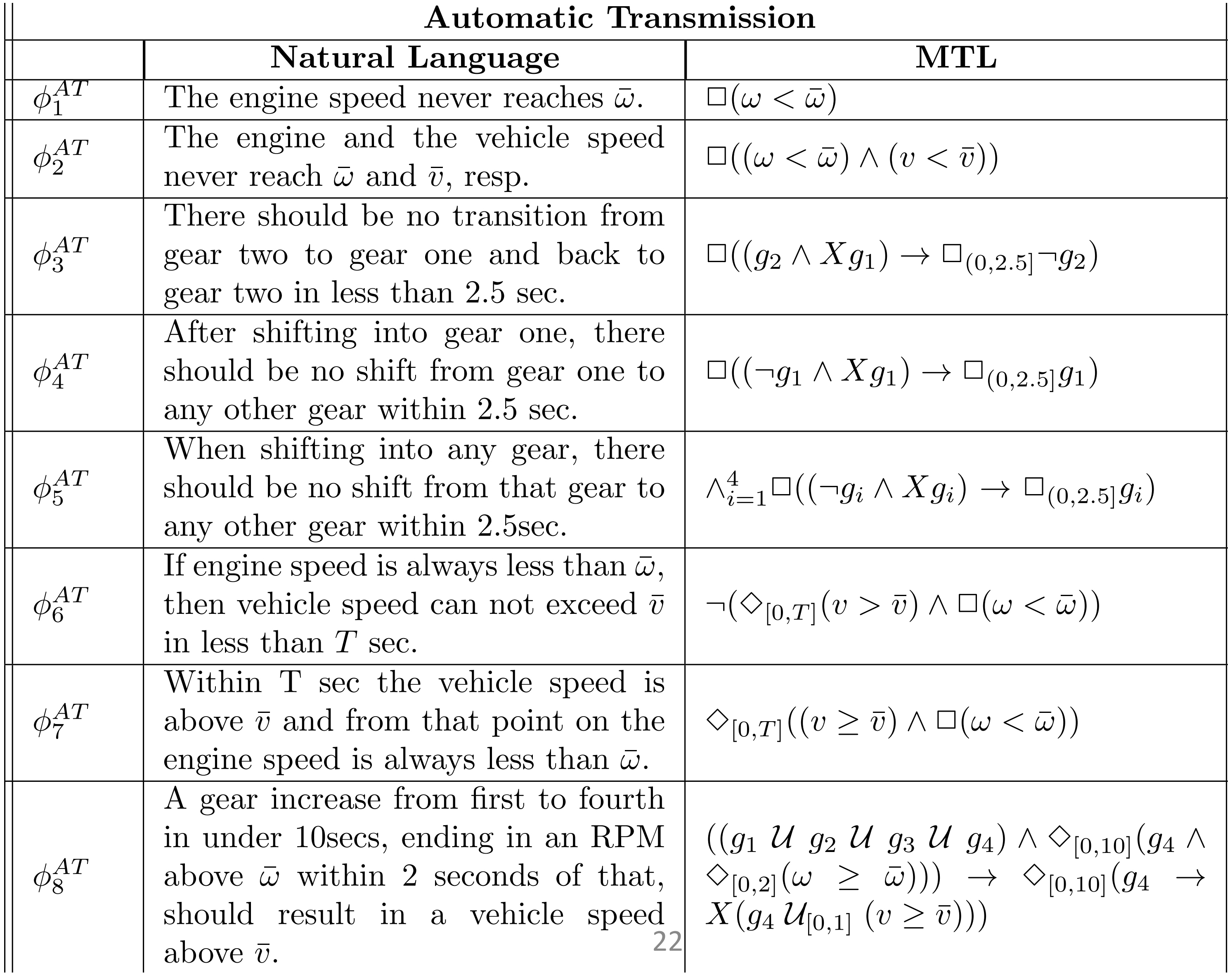
Automatic Transmission

- The model contains 69 blocks among which there are 2 integrators (i.e., 2 continuous state variables), and a Stateflow chart. The Stateflow chart contains two concurrently executing Finite State Machines with 4 and 3 states, respectively.

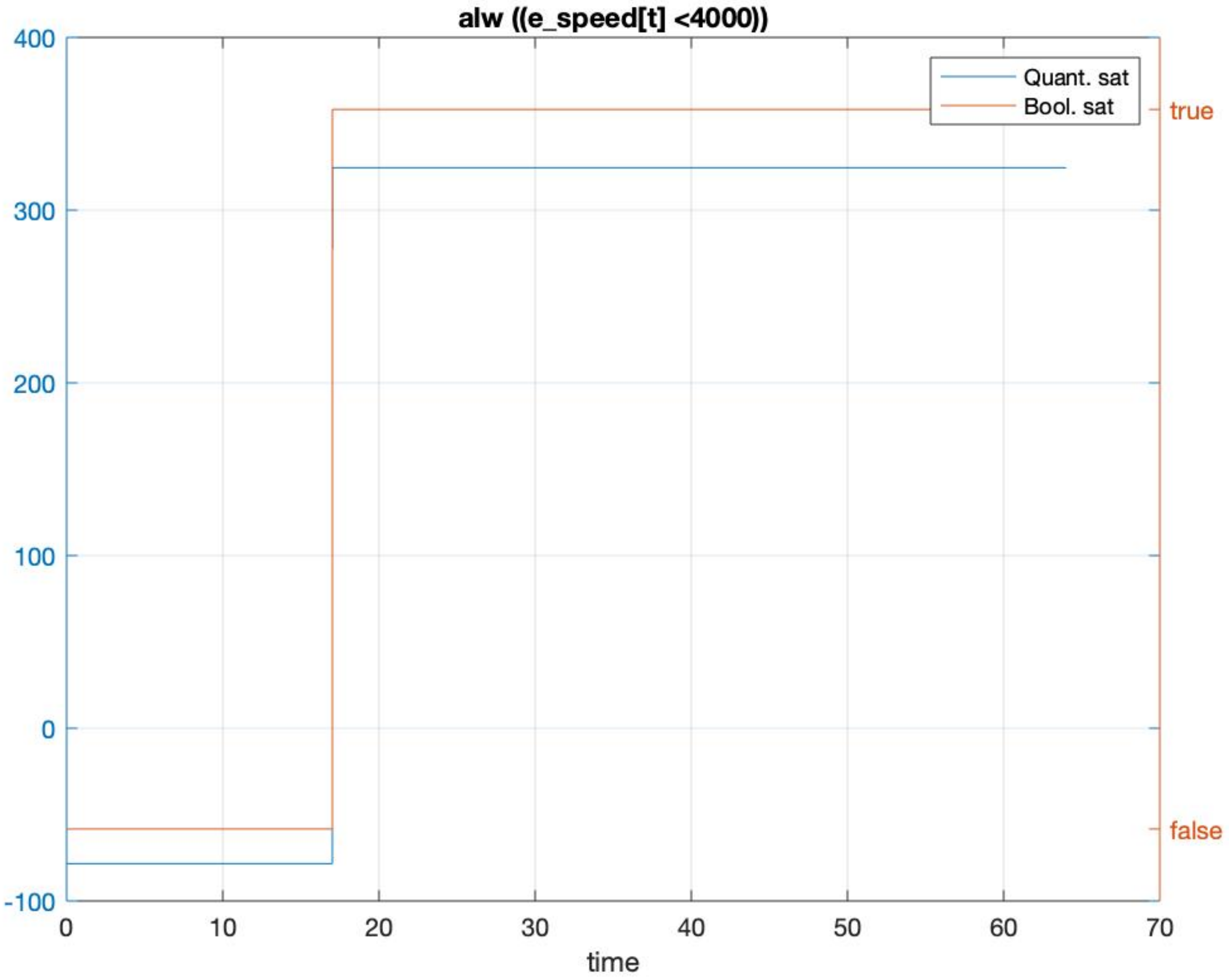




Properties

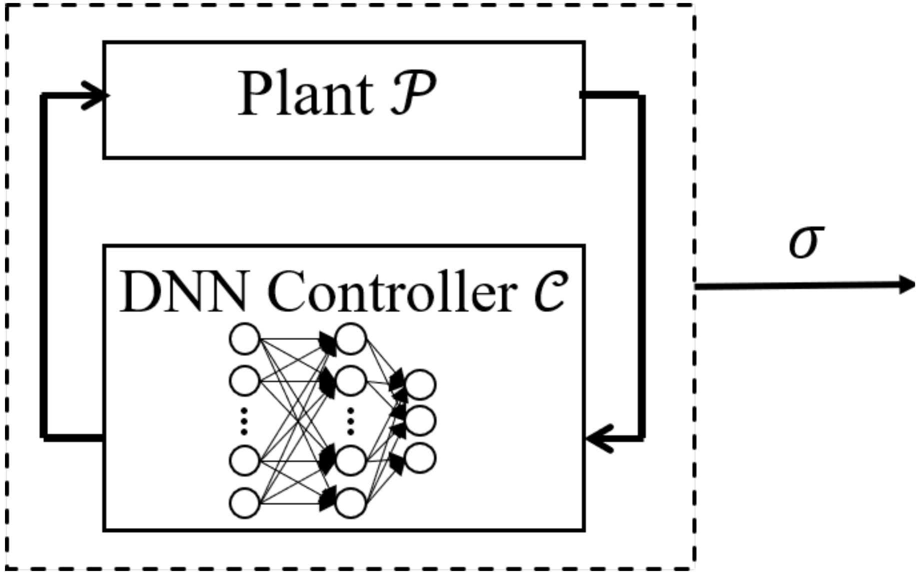


Automatic Transmission		
	Natural Language	**MTL**
ϕ_1^{AT}	The engine speed never reaches $\bar{\omega}$.	$\Box(\omega < \bar{\omega})$
ϕ_2^{AT}	The engine and the vehicle speed never reach $\bar{\omega}$ and $\bar{v}$, resp.	$\Box((\omega < \bar{\omega}) \wedge (v < \bar{v}))$
ϕ_3^{AT}	There should be no transition from gear two to gear one and back to gear two in less than 2.5 sec.	$\Box((g_2 \wedge X g_1) \rightarrow \Box_{(0,2.5]} \neg g_2)$
ϕ_4^{AT}	After shifting into gear one, there should be no shift from gear one to any other gear within 2.5 sec.	$\Box((\neg g_1 \wedge X g_1) \rightarrow \Box_{(0,2.5]} g_1)$
ϕ_5^{AT}	When shifting into any gear, there should be no shift from that gear to any other gear within 2.5sec.	$\wedge_{i=1}^{4} \Box((\neg g_i \wedge X g_i) \rightarrow \Box_{(0,2.5]} g_i)$
ϕ_6^{AT}	If engine speed is always less than $\bar{\omega}$, then vehicle speed can not exceed $\bar{v}$ in less than T sec.	$\neg(\Diamond_{[0,T]}(v > \bar{v}) \wedge \Box(\omega < \bar{\omega}))$
ϕ_7^{AT}	Within T sec the vehicle speed is above $\bar{v}$ and from that point on the engine speed is always less than $\bar{\omega}$.	$\Diamond_{[0,T]}((v \geq \bar{v}) \wedge \Box(\omega < \bar{\omega}))$
ϕ_8^{AT}	A gear increase from first to fourth in under 10secs, ending in an RPM above $\bar{\omega}$ within 2 seconds of that, should result in a vehicle speed above $\bar{v}$.	$((g_1 \ \mathcal{U} \ g_2 \ \mathcal{U} \ g_3 \ \mathcal{U} \ g_4) \wedge \Diamond_{[0,10]}(g_4 \wedge \Diamond_{[0,2]}(\omega \geq \bar{\omega}))) \rightarrow \Diamond_{[0,10]}(g_4 \rightarrow X(g_4 \ \mathcal{U}_{[0,1]} \ (v \geq \bar{v})))$


alw ((e_speed[t] <4000))
Quant. sat
Bool. sat
400
300
200
100
0
-100
true
false
0
10
20
30
40
50
60
70
time

A Deep Neural Network controller

$S = P||C$



Mojtaba Zarei, Yu Wang, Miroslav Pajic: ***Statistical verification of learning-based cyber-physical systems.*** *HSCC 2020: 12:1-12:7*

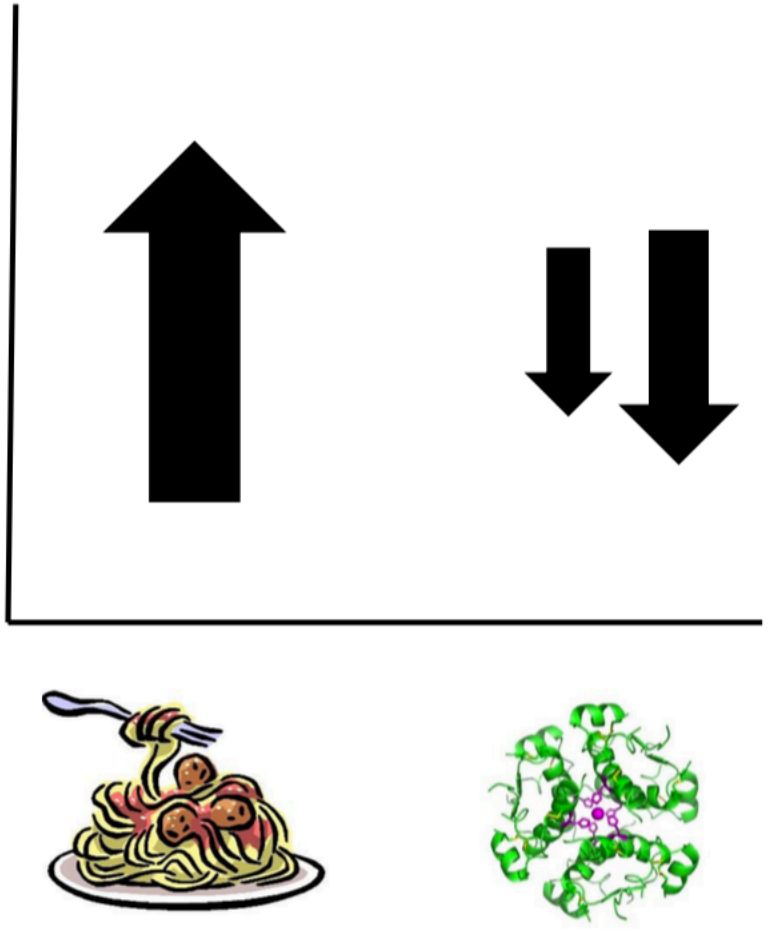
Reinforcement Learning and Temporal Logic

Challenges:

- Safe RL
- Complex Tasks
- Reward Hacking

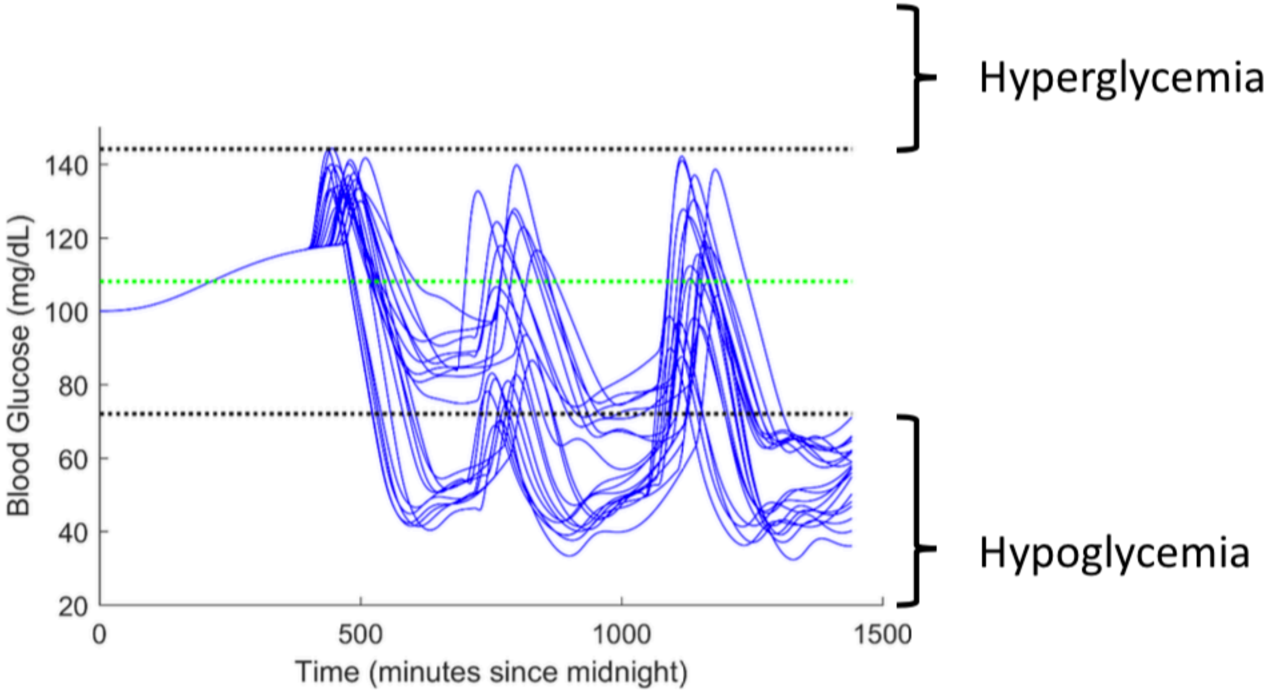
Safe Reinforcement Learning

Blood Glucose (sugar)



Eat Carbohydrates

Release Insulin



Reward Hacking

A policy that achieves high returns but against the designer’s intentions

https://www.youtube.com/watch?v=92qDfT8pENs

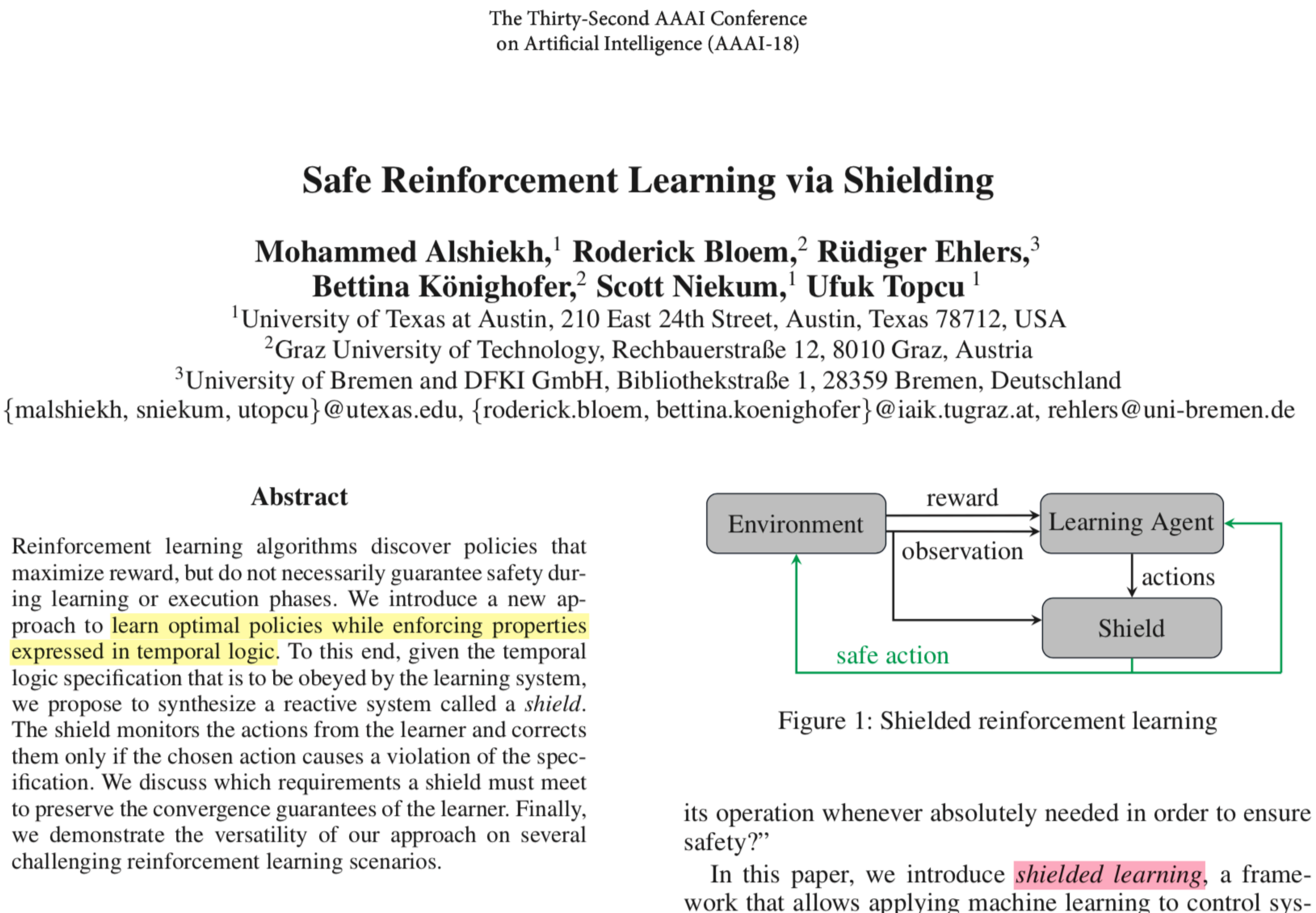
Reward function is not enough

Description using a language can help..

- To define task better
- To learn more efficiently and precisely
- To transfer learning between tasks
- To be “safe”

Several Works with different motivations

- **LTL constrained, Reward function remained the same**
- Reward shaping using probability of average robusntess satisfaction
- multi-task-RL



The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

Safe Reinforcement Learning via Shielding

Mohammed Alshiekh,[1] Roderick Bloem,[2] Rüdiger Ehlers,[3]
Bettina Könighofer,[2] Scott Niekum,[1] Ufuk Topcu [1]
[1]University of Texas at Austin, 210 East 24th Street, Austin, Texas 78712, USA
[2]Graz University of Technology, Rechbauerstraße 12, 8010 Graz, Austria
[3]University of Bremen and DFKI GmbH, Bibliothekstraße 1, 28359 Bremen, Deutschland
{malshiekh, sniekum, utopcu}@utexas.edu, {roderick.bloem, bettina.koenighofer}@iaik.tugraz.at, rehlers@uni-bremen.de

Abstract

Reinforcement learning algorithms discover policies that maximize reward, but do not necessarily guarantee safety during learning or execution phases. We introduce a new approach to learn optimal policies while enforcing properties expressed in temporal logic. To this end, given the temporal logic specification that is to be obeyed by the learning system, we propose to synthesize a reactive system called a *shield*. The shield monitors the actions from the learner and corrects them only if the chosen action causes a violation of the specification. We discuss which requirements a shield must meet to preserve the convergence guarantees of the learner. Finally, we demonstrate the versatility of our approach on several challenging reinforcement learning scenarios.

Figure 1: Shielded reinforcement learning

its operation whenever absolutely needed in order to ensure safety?”

In this paper, we introduce *shielded learning*, a framework that allows applying machine learning to control sys-

Safe RL via Shield

How can we let a learning agent do whatever it is doing, and also monitor and interfere with its operation whenever absolutely needed in order to ensure safety?

- The shield is computed upfront from the safety part of the given system specification and an abstraction of the agent's environment dynamics

- ***Minimum interference**:* monitors the actions selected by the learning agent and corrects them if and only if the chosen action is unsafe.

- Boundary helps to separate the concerns, e.g., safety and correctness on one side and convergence and optimality on the other

- Compatible with mechanisms such as function approximation, employed by learning algorithms in order to improve their scalability

Safe RL via Shield

- Safety fragment of LTL
  (something bad should never happen, e.g. no safety G(r → Fg), every request is eventually granted)

- A faithful, yet precise enough, abstraction of the physical environment is required

- Independent of the state space components of the system to be controlled

- The shield is the product between specification automaton and the MDP abstraction

Safe RL via Shield

If the property is violated there are two approaches:

- Assign a punishment : negative reward
- Assign the reward: positive reward

Then the shield selects an action in a “rank” that is safe

Grid world Example

With tabular Q-learning with an ε-greedy explorer

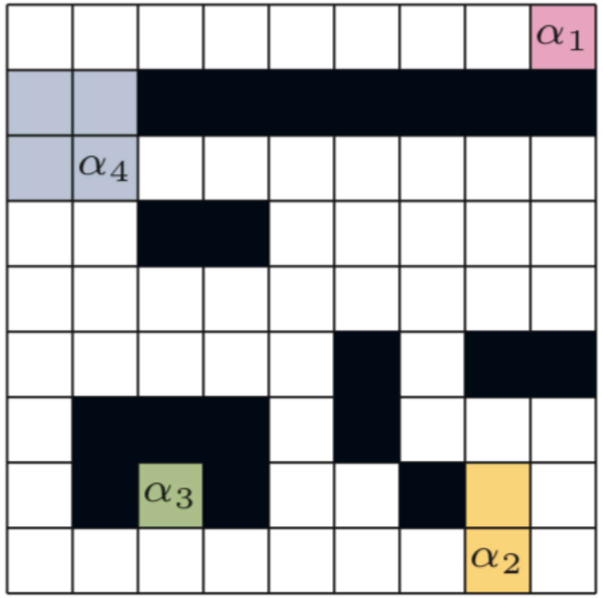


Figure 6: 9x9 grid world.

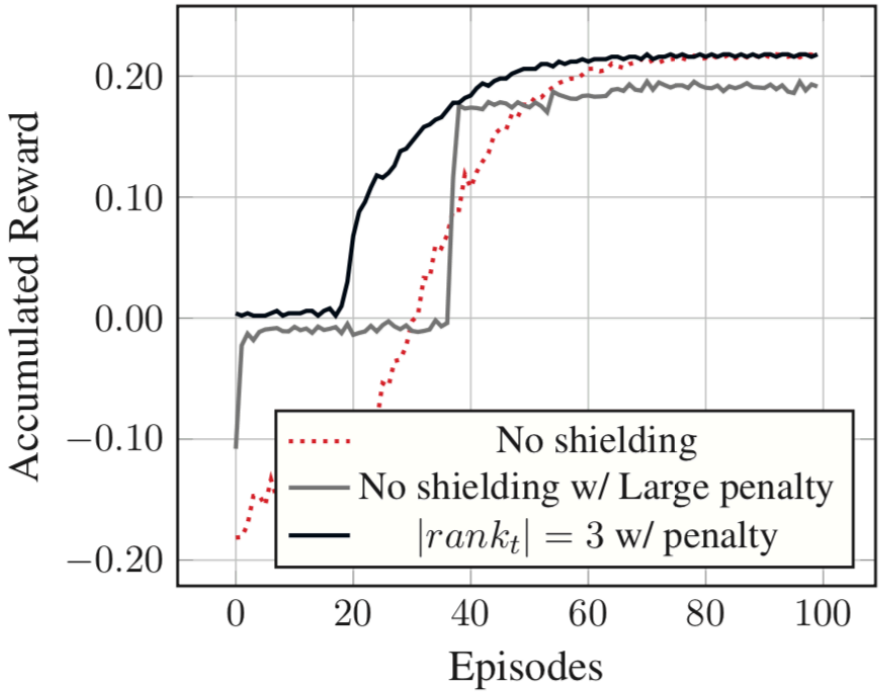


Figure 7: The accumulated reward per episode for the 9x9 grid world example.

The PacMan Example

Approximate Q-learning agent

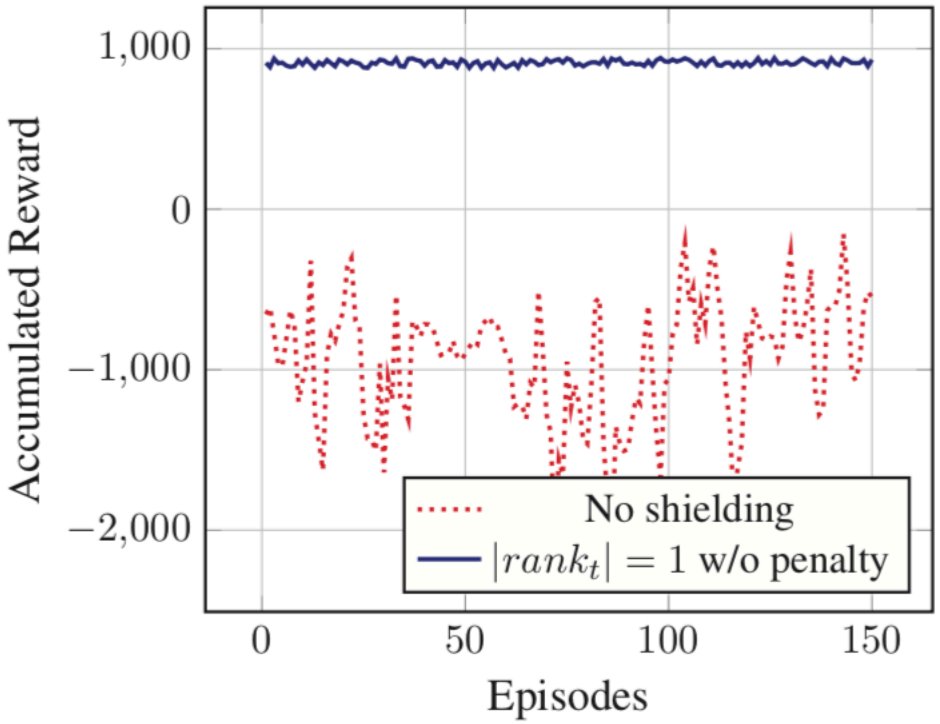


Figure 13: The accumulated reward per episode for the pacman example.

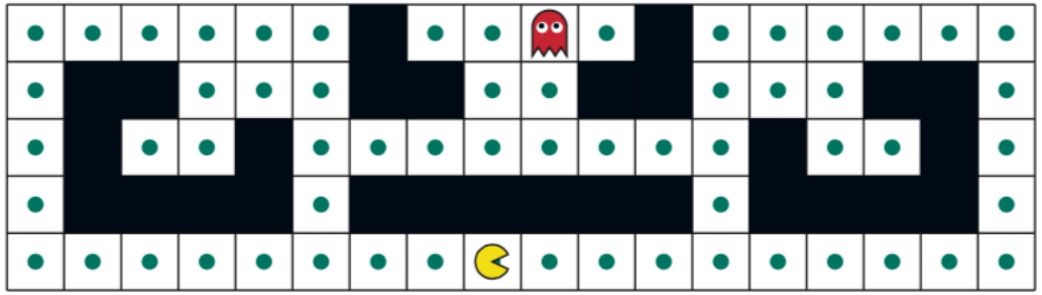


Figure 12: The 5x18 grid world of the pacman example.

Several Works with different motivations

- LTL constrained, Reward function remained the same

- **Reward shaping using probability of average robusntess satisfaction**

- multi-task-RL

General Idea

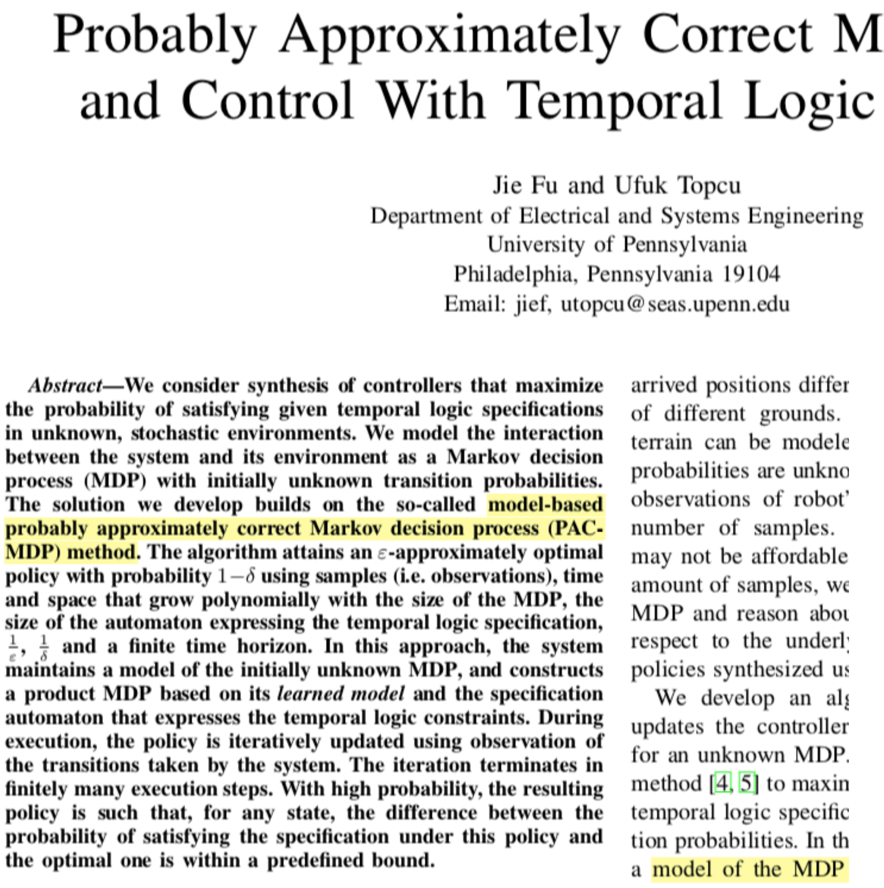
Reward Shaping problem:

Design $R(s, a\ )$ s.t. I can find $\pi^*\ s.t. \forall\ x, \pi^*(x)$ the ”satisfaction” of x is maximised

Why important?

- Poorly design -> poorly convergence
- Learning unsafe or unrealistic action

LTL constrained to discrete state and action



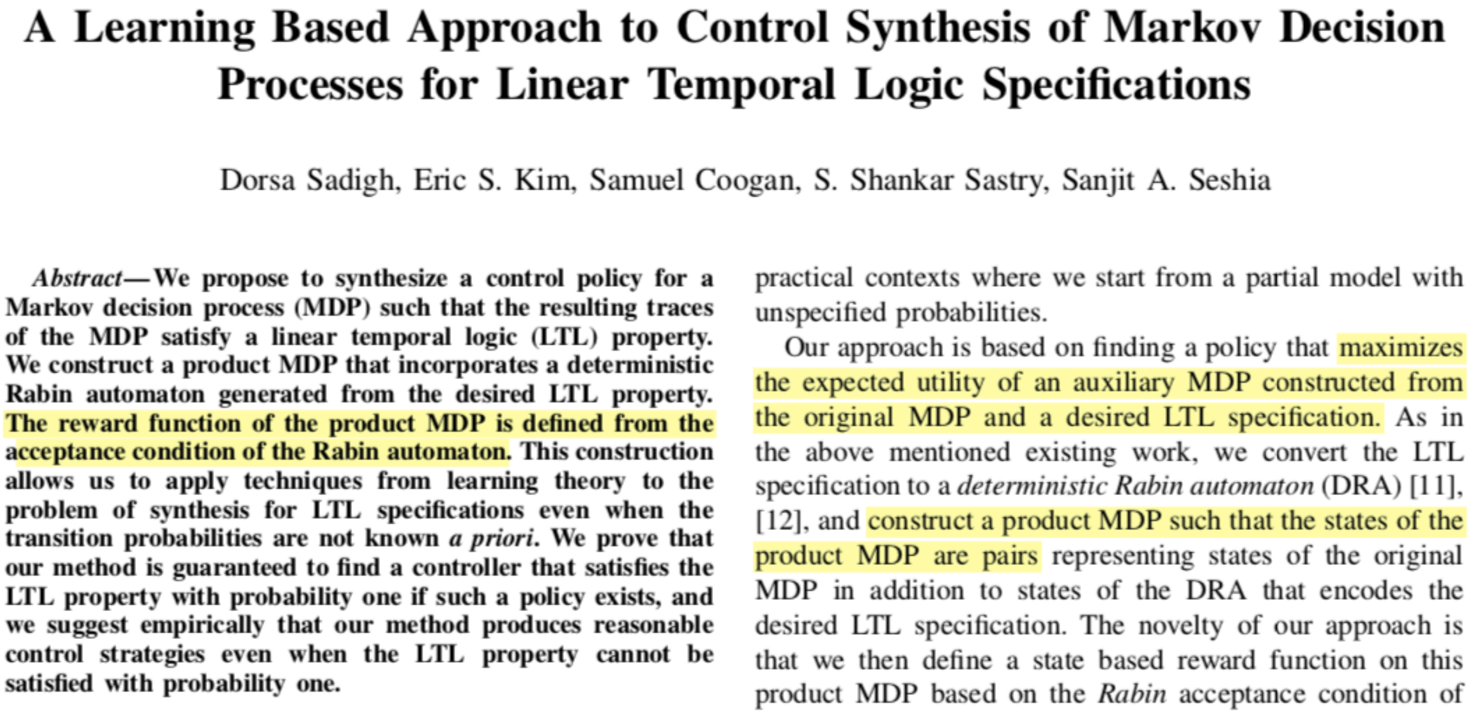
Probably Approximately Correct MDP Learning and Control With Temporal Logic Constraints

Jie Fu and Ufuk Topcu
Department of Electrical and Systems Engineering
University of Pennsylvania
Philadelphia, Pennsylvania 19104
Email: jief, utopcu@seas.upenn.edu

***Abstract*—We consider synthesis of controllers that maximize the probability of satisfying given temporal logic specifications in unknown, stochastic environments. We model the interaction between the system and its environment as a Markov decision process (MDP) with initially unknown transition probabilities. The solution we develop builds on the so-called model-based probably approximately correct Markov decision process (PAC-MDP) method. The algorithm attains an ε-approximately optimal policy with probability $1-\delta$ using samples (i.e. observations), time and space that grow polynomially with the size of the MDP, the size of the automaton expressing the temporal logic specification, $\frac{1}{\varepsilon}$, $\frac{1}{\delta}$ and a finite time horizon. In this approach, the system maintains a model of the initially unknown MDP, and constructs a product MDP based on its *learned model* and the specification automaton that expresses the temporal logic constraints. During execution, the policy is iteratively updated using observation of the transitions taken by the system. The iteration terminates in finitely many execution steps. With high probability, the resulting policy is such that, for any state, the difference between the probability of satisfying the specification under this policy and the optimal one is within a predefined bound.**

arrived positions differ
of different grounds.
terrain can be modele
probabilities are unkno
observations of robot'
number of samples.
may not be affordable
amount of samples, we
MDP and reason abou
respect to the underl
policies synthesized us

We develop an alg
updates the controller
for an unknown MDP.
method [4, 5] to maxin
temporal logic specific
tion probabilities. In th
a model of the MDP



A Learning Based Approach to Control Synthesis of Markov Decision Processes for Linear Temporal Logic Specifications

Dorsa Sadigh, Eric S. Kim, Samuel Coogan, S. Shankar Sastry, Sanjit A. Seshia

***Abstract*—We propose to synthesize a control policy for a Markov decision process (MDP) such that the resulting traces of the MDP satisfy a linear temporal logic (LTL) property. We construct a product MDP that incorporates a deterministic Rabin automaton generated from the desired LTL property. The reward function of the product MDP is defined from the acceptance condition of the Rabin automaton. This construction allows us to apply techniques from learning theory to the problem of synthesis for LTL specifications even when the transition probabilities are not known *a priori*. We prove that our method is guaranteed to find a controller that satisfies the LTL property with probability one if such a policy exists, and we suggest empirically that our method produces reasonable control strategies even when the LTL property cannot be satisfied with probability one.**

practical contexts where we start from a partial model with unspecified probabilities.

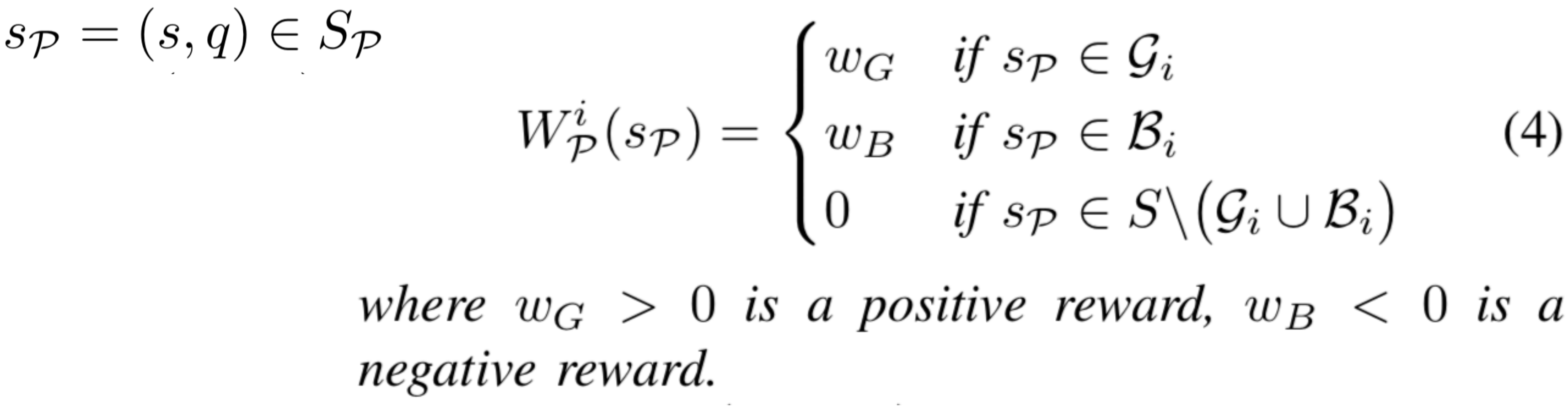
Our approach is based on finding a policy that maximizes the expected utility of an auxiliary MDP constructed from the original MDP and a desired LTL specification. As in the above mentioned existing work, we convert the LTL specification to a *deterministic Rabin automaton* (DRA) [11], [12], and construct a product MDP such that the states of the product MDP are pairs representing states of the original MDP in addition to states of the DRA that encodes the desired LTL specification. The novelty of our approach is that we then define a state based reward function on this product MDP based on the *Rabin* acceptance condition of

LTL constrained to discrete state and action

- For MDPs with unknown transition probability
- LTL -> Deterministic Rabin Automata (DRA)
- Translation breakes the history- dependence
- select the reward function on the product MDP so it corresponds to the *Rabin* acceptance condition of the LTL specification.

LTL constrained to discrete state and action

- select the reward function on the product MDP so it corresponds to the *Rabin* acceptance condition of the LTL specification.



$s_{\mathcal{P}} = (s, q) \in S_{\mathcal{P}}$

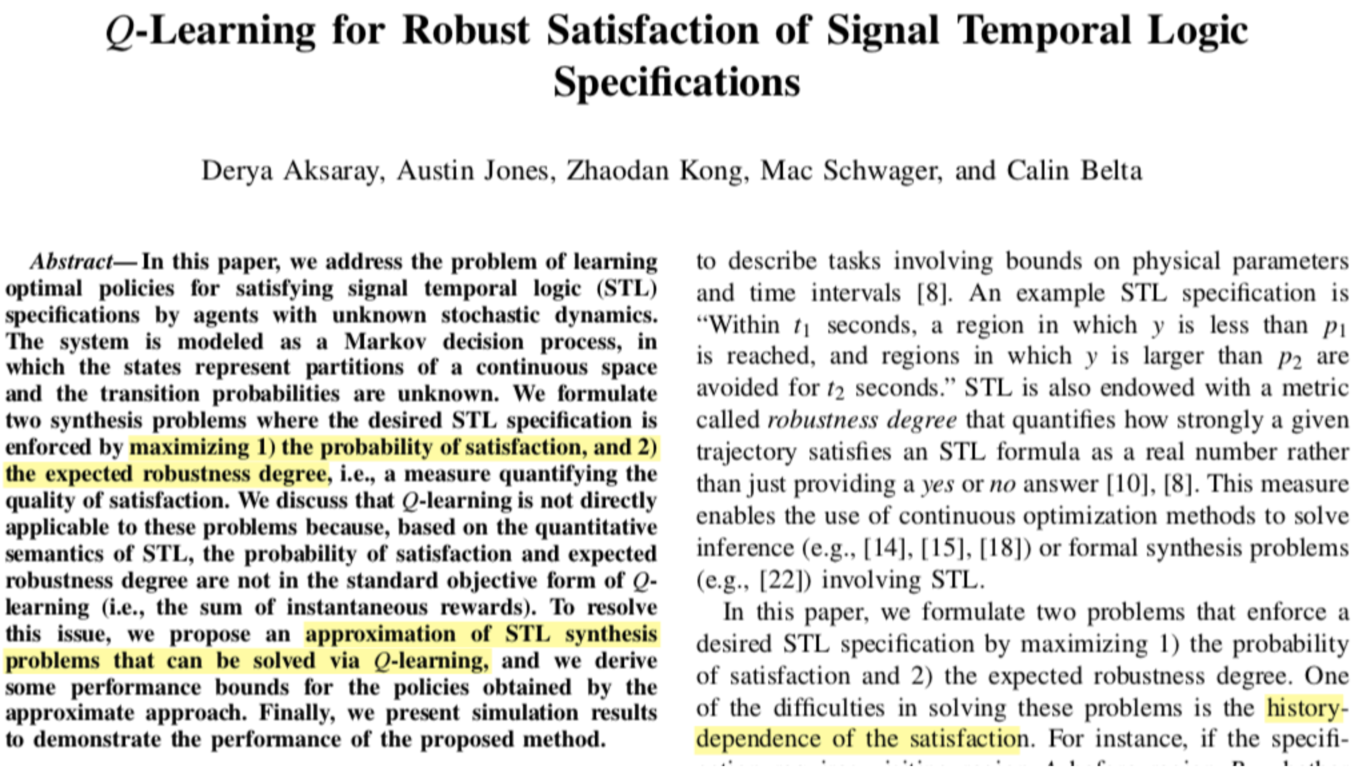
$$W_{\mathcal{P}}^{i}(s_{\mathcal{P}}) = \begin{cases} w_G & \textit{if } s_{\mathcal{P}} \in \mathcal{G}_i \\ w_B & \textit{if } s_{\mathcal{P}} \in \mathcal{B}_i \\ 0 & \textit{if } s_{\mathcal{P}} \in S \backslash (\mathcal{G}_i \cup \mathcal{B}_i) \end{cases} \tag{4}$$

where $w_G > 0$ *is a positive reward,* $w_B < 0$ *is a negative reward.*

- Prove convergence if policy exist s.t. it satisfies property with probability 1

- 1) Learn the transition probabilities and 2) Optimize the expected utility. E.g. with a modified active temporal difference learning algorithm

STL and discrete space



Q-Learning for Robust Satisfaction of Signal Temporal Logic Specifications

Derya Aksaray, Austin Jones, Zhaodan Kong, Mac Schwager, and Calin Belta

***Abstract*— In this paper, we address the problem of learning optimal policies for satisfying signal temporal logic (STL) specifications by agents with unknown stochastic dynamics. The system is modeled as a Markov decision process, in which the states represent partitions of a continuous space and the transition probabilities are unknown. We formulate two synthesis problems where the desired STL specification is enforced by maximizing 1) the probability of satisfaction, and 2) the expected robustness degree, i.e., a measure quantifying the quality of satisfaction. We discuss that Q-learning is not directly applicable to these problems because, based on the quantitative semantics of STL, the probability of satisfaction and expected robustness degree are not in the standard objective form of Q-learning (i.e., the sum of instantaneous rewards). To resolve this issue, we propose an approximation of STL synthesis problems that can be solved via Q-learning, and we derive some performance bounds for the policies obtained by the approximate approach. Finally, we present simulation results to demonstrate the performance of the proposed method.**

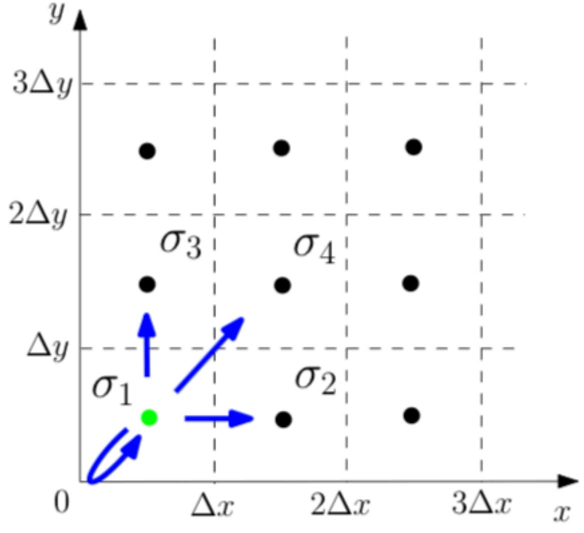
to describe tasks involving bounds on physical parameters and time intervals [8]. An example STL specification is "Within t_1 seconds, a region in which y is less than p_1 is reached, and regions in which y is larger than p_2 are avoided for t_2 seconds." STL is also endowed with a metric called *robustness degree* that quantifies how strongly a given trajectory satisfies an STL formula as a real number rather than just providing a *yes* or *no* answer [10], [8]. This measure enables the use of continuous optimization methods to solve inference (e.g., [14], [15], [18]) or formal synthesis problems (e.g., [22]) involving STL.

In this paper, we formulate two problems that enforce a desired STL specification by maximizing 1) the probability of satisfaction and 2) the expected robustness degree. One of the difficulties in solving these problems is the history-dependence of the satisfaction. For instance, if the specifi-

STL and discrete space

- Partition of a Continuous Space

- Uknown stochastic dynamics



(a)

Fig. 2. (a) Discretized state-space,

Problem: history- dependence of the satisfaction

- Fragment of STL such that the progress towards satisfaction is checked with a sufficient number of (i.e., τ) state measurements.

$$\varphi \quad := f(\mathbf{s}) < d \,|\, \neg\varphi \,|\, \varphi \wedge \varphi \,|\, \varphi \vee \varphi \,|\, F_{[a,b]}\varphi \,|\, G_{[a,b]}\varphi,$$

$$\pi_1^* = \arg\max_{\pi} Pr^{\pi}[s_{0:T} \models \Phi] \qquad \pi_2^* = \arg\max_{\pi} E^{\pi}[r(s_{0:T}, \Phi)]$$

Problem: history- dependence of the satisfaction

- τ −MDP where $\tau = \frac{hrz(\psi)}{\Delta t} + 1$ for $F_{[0,T]}\psi, G_{[0,T]}\psi$
- Each state corresponds to a τ-length trajectory
- Probability remains Markovian

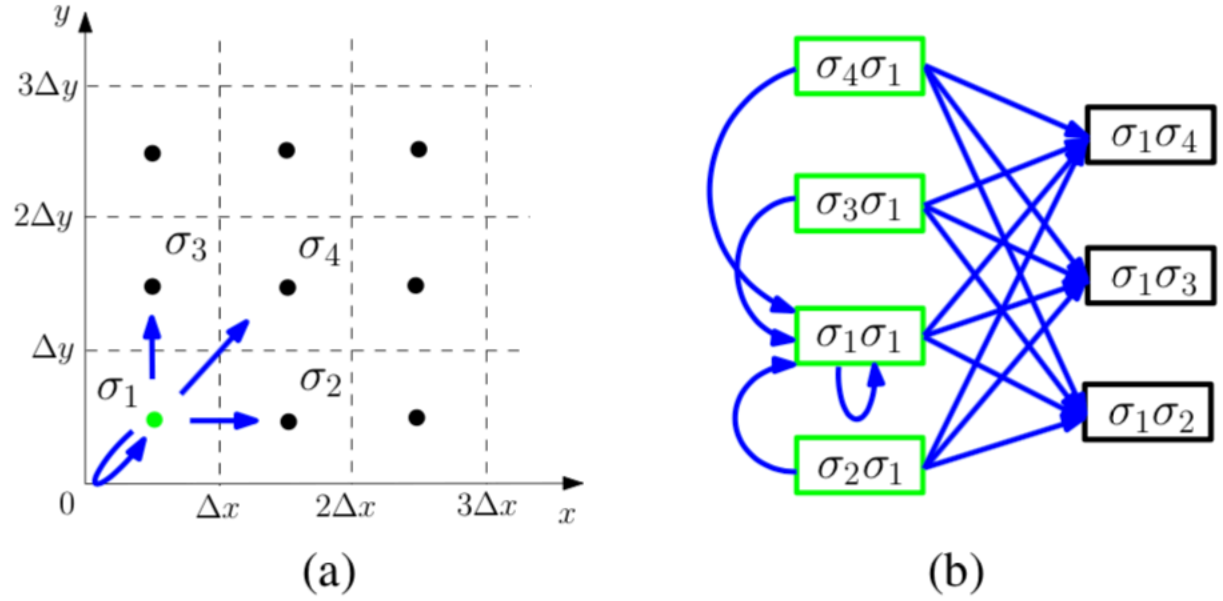


Fig. 2. (a) Discretized state-space, (b) Representation of σ_1 over $2 - MDP$.

Problem: robustness shape

$$\max_{\pi} E^{\pi}[r(s_{0:T},\Phi)] = \begin{cases} \max_{\pi} E^{\pi}\Big[\max_{\tau-1\leq t\leq T}\big(r(s_t^{\tau},\phi)\big)\Big], & \text{if } \Phi = F_{[0,T]}\phi \\ \max_{\pi} E^{\pi}\Big[\min_{\tau-1\leq t\leq T}\big(r(s_t^{\tau},\phi)\big)\Big], & \text{if } \Phi = G_{[0,T]}\phi \end{cases}$$

- log-sum-exp approximation to adapt the Robustness of Q-learning

$$\max(x_1,\ldots,x_n) \sim \frac{1}{\beta}\log\sum_{i=1}^{n} e^{\beta x_i},$$

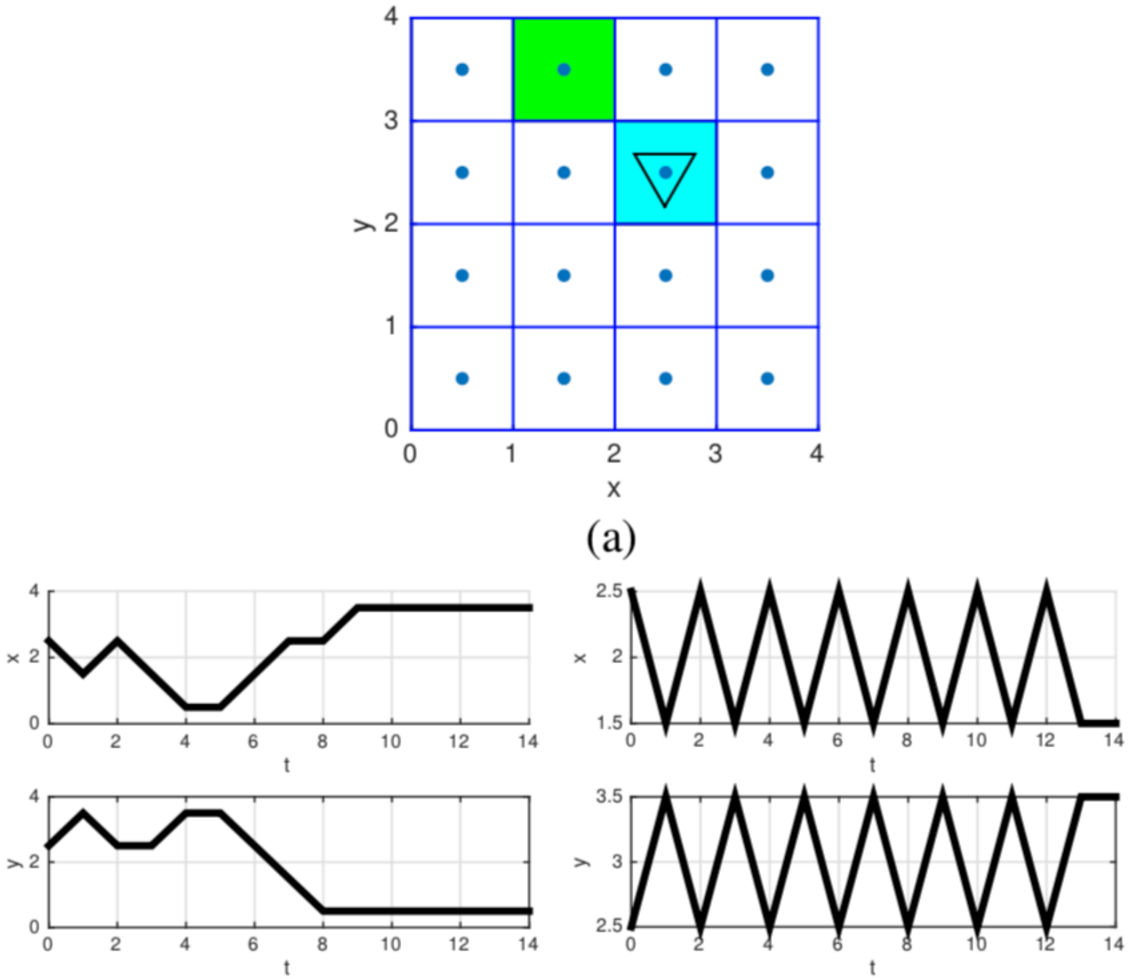
Finally…

$$\pi_{2A}^{*} = \begin{cases} \arg\max\limits_{\pi} E^{\pi}\left[\sum\limits_{t=\tau-1}^{T} e^{\beta r(s_t^{\tau},\phi)}\right], & \text{if } \Phi = F_{[0,T]}\phi \\ \arg\max\limits_{\pi} E^{\pi}\left[-\sum\limits_{t=\tau-1}^{T} e^{-\beta r(s_t^{\tau},\phi)}\right], & \text{if } \Phi = G_{[0,T]}\phi \end{cases}$$

The immediate reward is :

$$R = \begin{cases} e^{\beta I(r(s_j^{\tau},\phi))}, & \text{if Problem 1A with } \Phi = F_{[0,T]}\phi \\ -e^{-\beta I(r(s_j^{\tau},\phi))}, & \text{if Problem 1A with } \Phi = G_{[0,T]}\phi \\ e^{\beta r(s_j^{\tau},\phi)}, & \text{if Problem 2A with } \Phi = F_{[0,T]}\phi \\ -e^{-\beta r(s_j^{\tau},\phi)}, & \text{if Problem 2A with } \Phi = G_{[0,T]}\phi \end{cases}$$

Experiments



(a)

(b)

(c)

Fig. 5. (a) The initial state and the desired regions in case study 2 for which a sample trajectory by (b) π^*_{1A} and (c) π^*_{2A}.

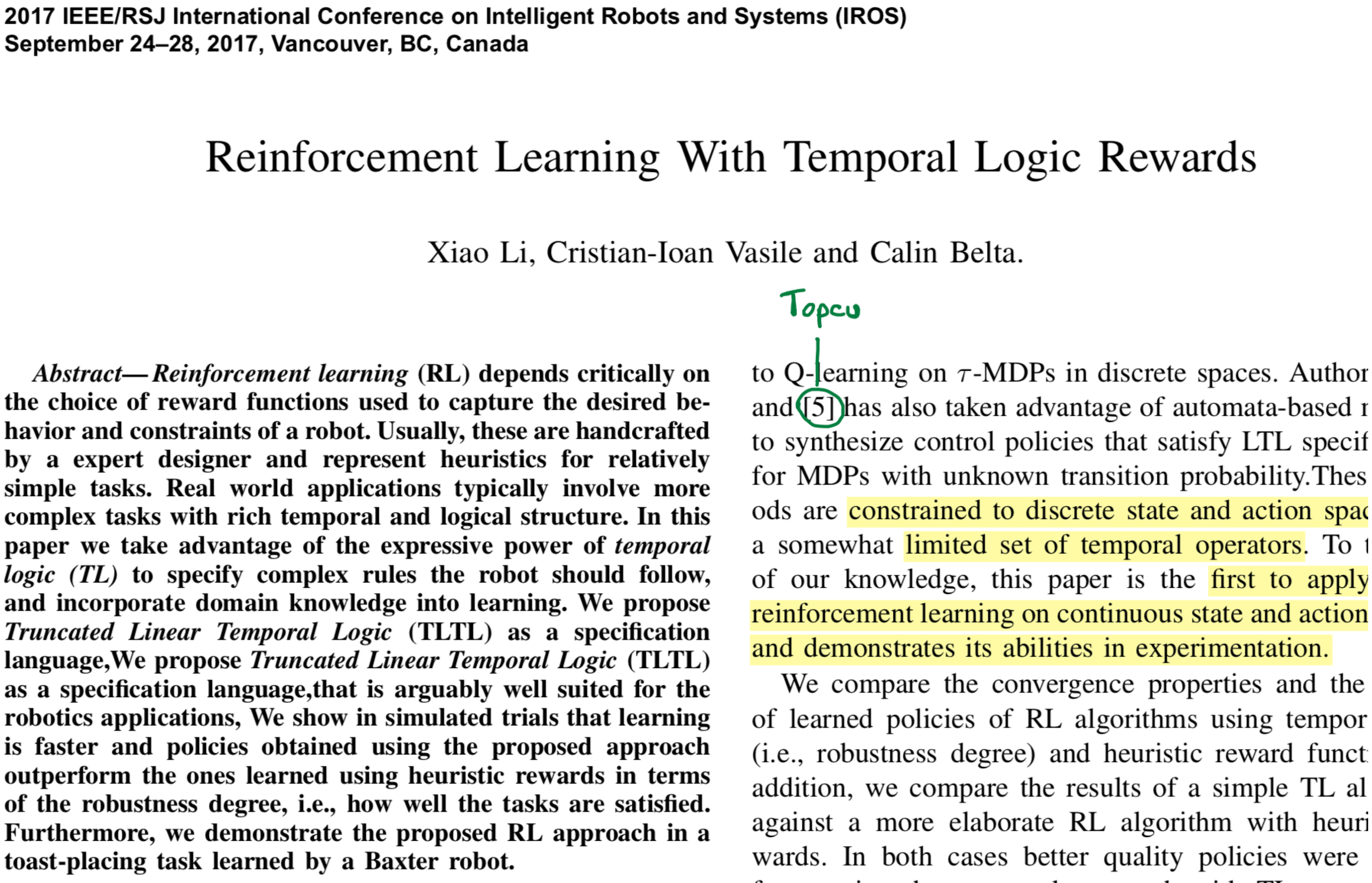
$$\Phi_2 = G_{[0,12]}(F_{[0,2]}(region\ A) \wedge F_{[0,2]}(region\ B))$$

$|S|$= 19 and $|S^{\tau}|$ = 676

the robustness degree gives “partial credit” for trajectories that are close to satisfaction

For the prop satisfaction, instead, Q-learning algorithm is essentially performing a random search

STL and continuous space



2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
September 24–28, 2017, Vancouver, BC, Canada

Reinforcement Learning With Temporal Logic Rewards

Xiao Li, Cristian-Ioan Vasile and Calin Belta.

Abstract—*Reinforcement learning* (RL) depends critically on the choice of reward functions used to capture the desired behavior and constraints of a robot. Usually, these are handcrafted by a expert designer and represent heuristics for relatively simple tasks. Real world applications typically involve more complex tasks with rich temporal and logical structure. In this paper we take advantage of the expressive power of *temporal logic (TL)* to specify complex rules the robot should follow, and incorporate domain knowledge into learning. We propose *Truncated Linear Temporal Logic* (TLTL) as a specification language,We propose *Truncated Linear Temporal Logic* (TLTL) as a specification language,that is arguably well suited for the robotics applications, We show in simulated trials that learning is faster and policies obtained using the proposed approach outperform the ones learned using heuristic rewards in terms of the robustness degree, i.e., how well the tasks are satisfied. Furthermore, we demonstrate the proposed RL approach in a toast-placing task learned by a Baxter robot.

Topcu

to Q-learning on τ-MDPs in discrete spaces. Author
and [5] has also taken advantage of automata-based r
to synthesize control policies that satisfy LTL specif
for MDPs with unknown transition probability.Thes
ods are constrained to discrete state and action spac
a somewhat limited set of temporal operators. To t
of our knowledge, this paper is the first to apply
reinforcement learning on continuous state and action
and demonstrates its abilities in experimentation.

We compare the convergence properties and the
of learned policies of RL algorithms using tempor
(i.e., robustness degree) and heuristic reward funct
addition, we compare the results of a simple TL al
against a more elaborate RL algorithm with heuri
wards. In both cases better quality policies were

Truncated Linear Temporal Logic (TLTL)

- Specifically for robots
- Unbounded
- Atomic propositions
- Evaluated against finite time sequences

$$\begin{aligned}
\rho(s_{t:t+k}, \top) &= \rho_{max}, \\
\rho(s_{t:t+k}, f(s_t) < c) &= c - f(s_t), \\
\rho(s_{t:t+k}, \neg\phi) &= -\rho(s_{t:t+k}, \phi), \\
\rho(s_{t:t+k}, \phi \Rightarrow \psi) &= \max(-\rho(s_{t:t+k}, \phi), \rho(s_{t:t+k}, \psi)) \\
\rho(s_{t:t+k}, \phi_1 \wedge \phi_2) &= \min(\rho(s_{t:t+k}, \phi_1), \rho(s_{t:t+k}, \phi_2)), \\
\rho(s_{t:t+k}, \phi_1 \vee \phi_2) &= \max(\rho(s_{t:t+k}, \phi_1), \rho(s_{t:t+k}, \phi_2)), \\
\rho(s_{t:t+k}, \bigcirc\phi) &= \rho(s_{t+1:t+k}, \phi) \ (k > 0), \\
\rho(s_{t:t+k}, \Box\phi) &= \min_{t' \in [t,t+k)} (\rho(s_{t':t+k}, \phi)), \\
\rho(s_{t:t+k}, \Diamond\phi) &= \max_{t' \in [t,t+k)} (\rho(s_{t':t+k}, \phi)), \\
\rho(s_{t:t+k}, \phi \,\mathcal{U}\, \psi) &= \max_{t' \in [t,t+k)} (\min(\rho(s_{t':t+k}, \psi), \\
&\quad \min_{t'' \in [t,t')} \rho(s_{t'':t'}, \phi))), \\
\rho(s_{t:t+k}, \phi \,\mathcal{T}\, \psi) &= \max_{t' \in [t,t+k)} (\min(\rho(s_{t':t+k}, \psi), \\
&\quad \max_{t'' \in [t,t')} \rho(s_{t'':t'}, \phi))),
\end{aligned}$$

STL and continuous space

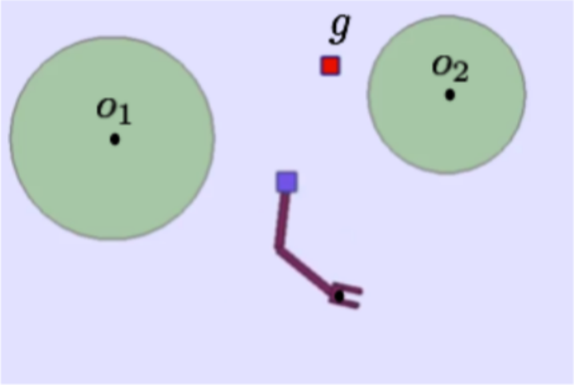
- Parametrized policy $\pi(s, a|\theta)$

- $\theta^* = argmax_\theta \, E_{p^{\pi_\theta(\tau)}}[R(\tau)]$,
where $p^{\pi_\theta}(\tau)$ is trajectory distribution from following policy π

- Relative Entropy Policy Search (REPS) :
constrained optimization problem that can be solved by Lagrange multipliers method

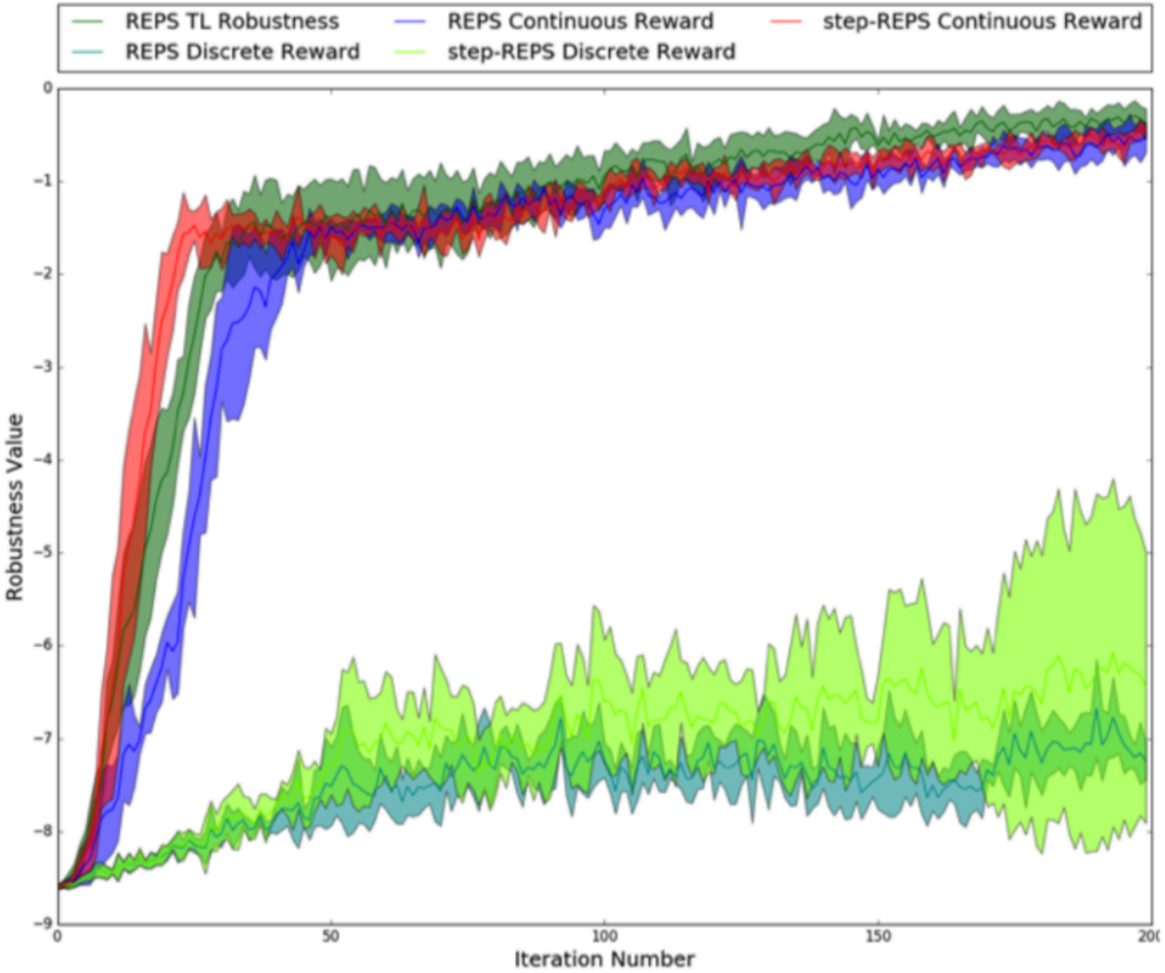
- Tlinear-Gaussian policies and weighted maximum-likelihood estimation to update the policy parameters

Experiments

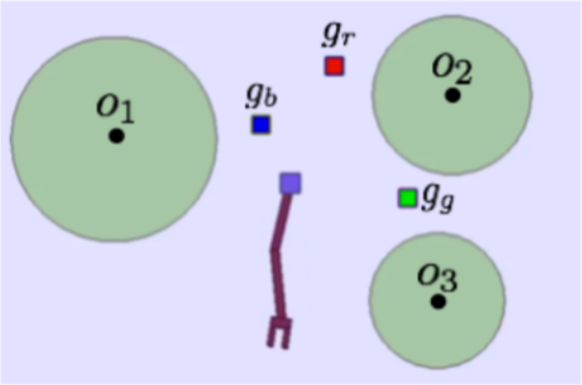


$$r_1^{discrete} = \begin{cases} 5 & d_g \leq 0.2 \\ -2 & d_{o_{1,2}} \leq r_{o_{1,2}} \\ 0 & \text{everywhere else} \end{cases}$$

$$r_1^{continuous} = -c_1 d_g + c_2 \sum_{i=1}^{2} d_{o_i}.$$

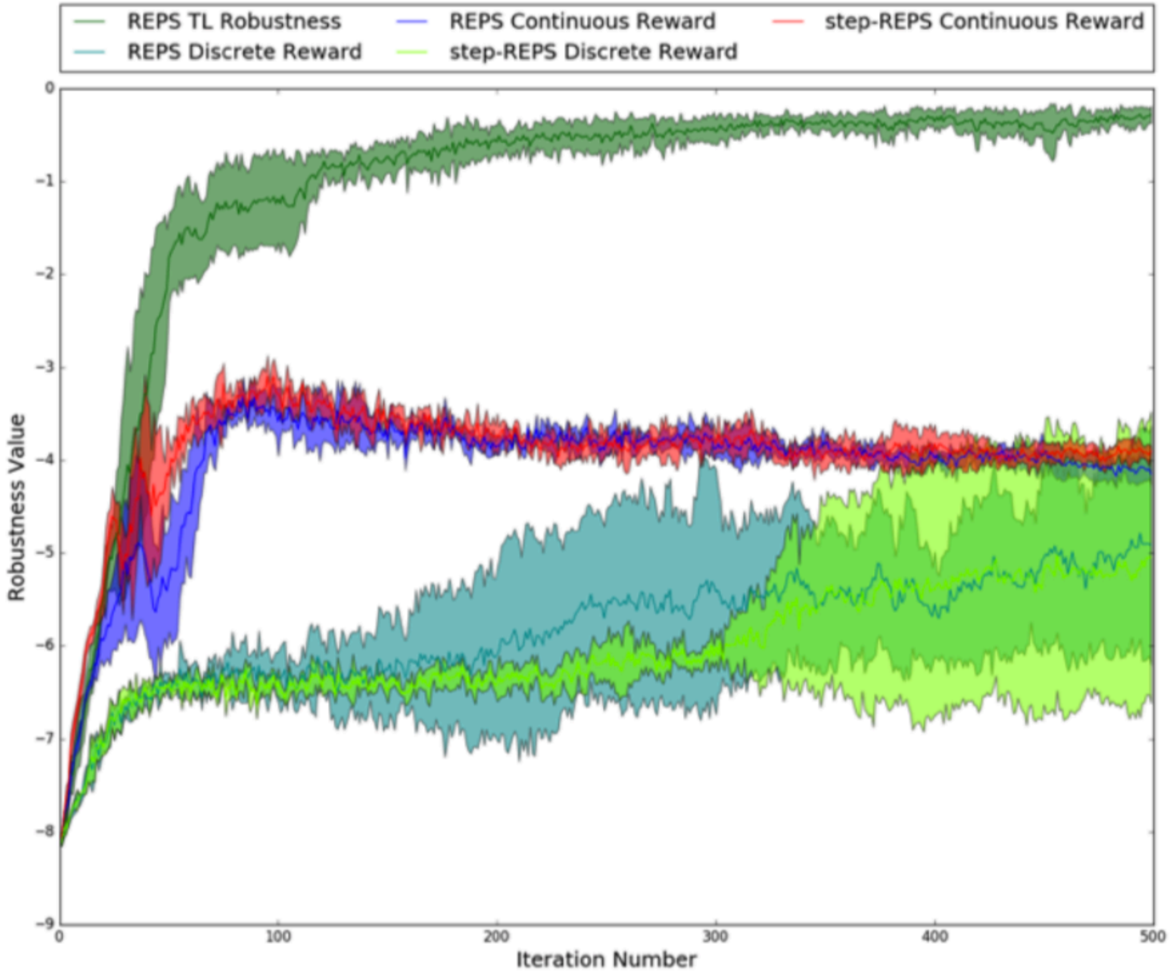


Experiments

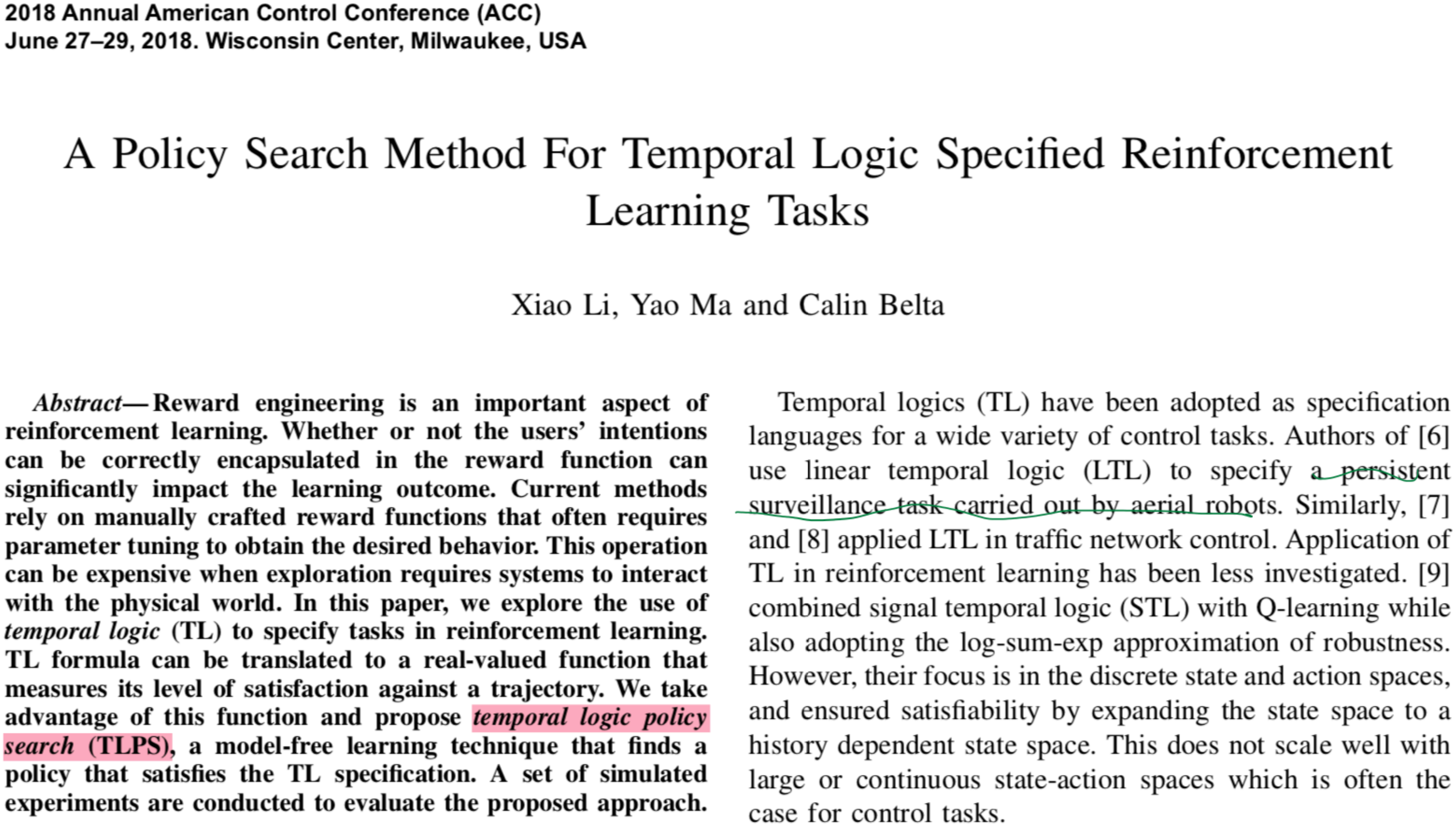


$$r_2^{discrete} = \begin{cases} 5 & \text{goals visited in the right order} \\ -5 & \text{goals visited in the wrong} \\ -2 & d_{o_{1,2,3}} \leq r_{o_{1,2,3}} \\ 0 & \text{everywhere else} \end{cases}$$

$$r_2^{continuous} = -c_1 d_{g_i} + c_2(d_{g_j} + d_{g_k}) + c_3 \sum_{i=1}^{3} d_{o_i}.$$



Smooth Robustness and continuous space



2018 Annual American Control Conference (ACC)
June 27–29, 2018. Wisconsin Center, Milwaukee, USA

A Policy Search Method For Temporal Logic Specified Reinforcement Learning Tasks

Xiao Li, Yao Ma and Calin Belta

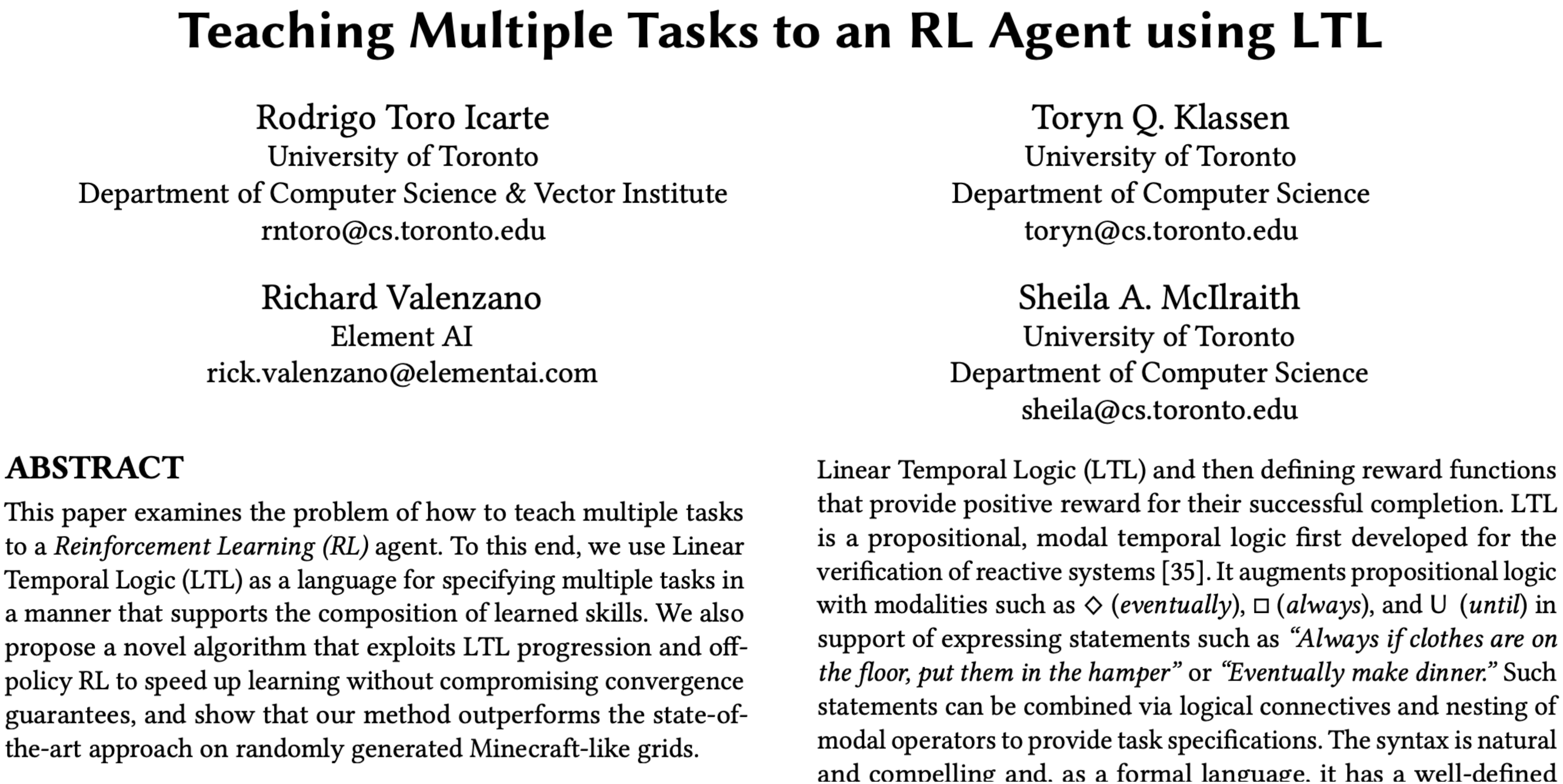
***Abstract*—Reward engineering is an important aspect of reinforcement learning. Whether or not the users' intentions can be correctly encapsulated in the reward function can significantly impact the learning outcome. Current methods rely on manually crafted reward functions that often requires parameter tuning to obtain the desired behavior. This operation can be expensive when exploration requires systems to interact with the physical world. In this paper, we explore the use of *temporal logic* (TL) to specify tasks in reinforcement learning. TL formula can be translated to a real-valued function that measures its level of satisfaction against a trajectory. We take advantage of this function and propose *temporal logic policy search* (TLPS), a model-free learning technique that finds a policy that satisfies the TL specification. A set of simulated experiments are conducted to evaluate the proposed approach.**

Temporal logics (TL) have been adopted as specification languages for a wide variety of control tasks. Authors of [6] use linear temporal logic (LTL) to specify a persistent surveillance task carried out by aerial robots. Similarly, [7] and [8] applied LTL in traffic network control. Application of TL in reinforcement learning has been less investigated. [9] combined signal temporal logic (STL) with Q-learning while also adopting the log-sum-exp approximation of robustness. However, their focus is in the discrete state and action spaces, and ensured satisfiability by expanding the state space to a history dependent state space. This does not scale well with large or continuous state-action spaces which is often the case for control tasks.

Several Works with different motivations

- LTL constrained, Reward function remained the same
- Reward shaping using probability of average robusntess satisfaction
- **Multi-task-RL**

Multi-task-RL



Teaching Multiple Tasks to an RL Agent using LTL

Rodrigo Toro Icarte
University of Toronto
Department of Computer Science & Vector Institute
rntoro@cs.toronto.edu

Toryn Q. Klassen
University of Toronto
Department of Computer Science
toryn@cs.toronto.edu

Richard Valenzano
Element AI
rick.valenzano@elementai.com

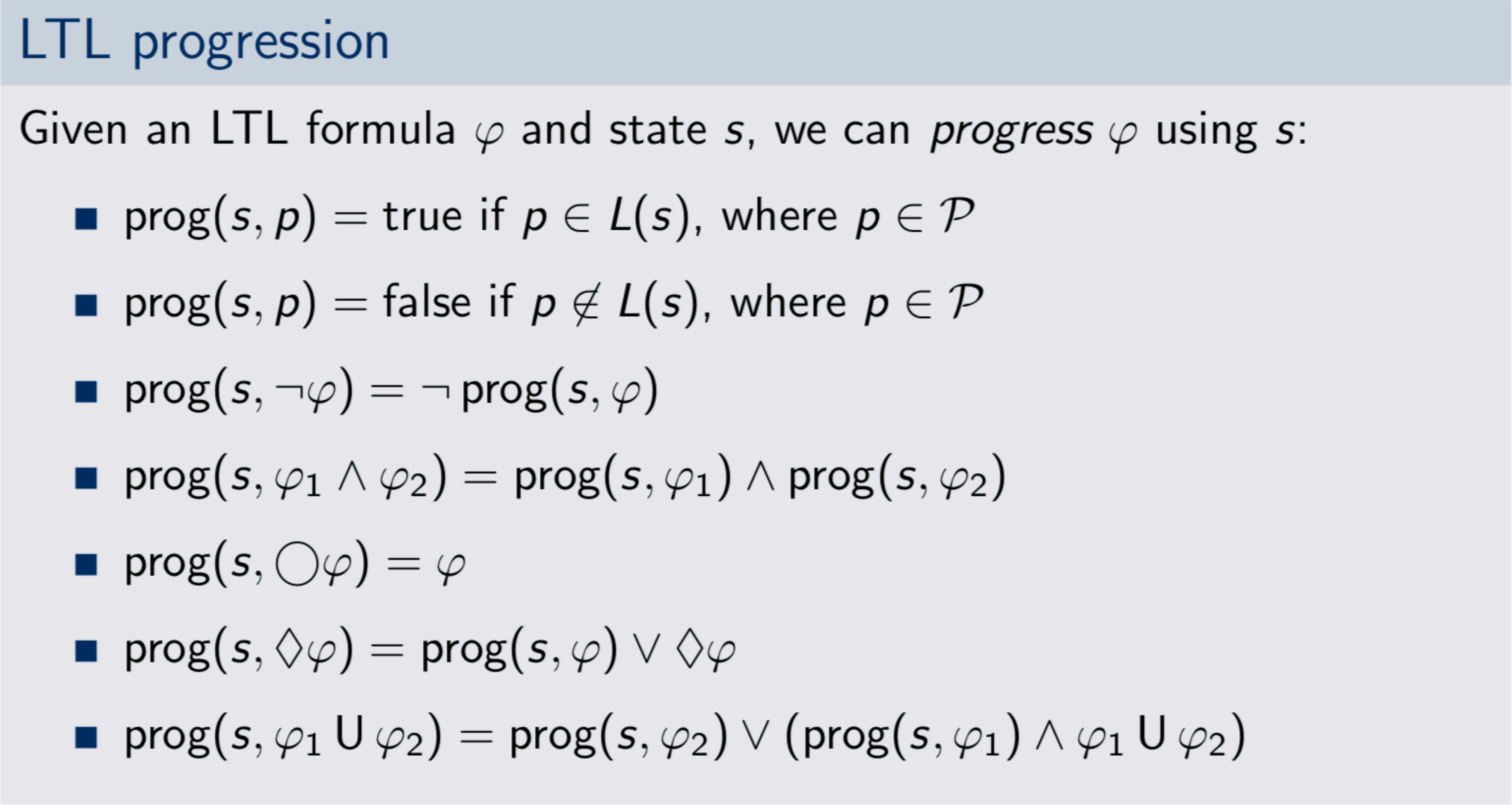
Sheila A. McIlraith
University of Toronto
Department of Computer Science
sheila@cs.toronto.edu

ABSTRACT

This paper examines the problem of how to teach multiple tasks to a *Reinforcement Learning (RL)* agent. To this end, we use Linear Temporal Logic (LTL) as a language for specifying multiple tasks in a manner that supports the composition of learned skills. We also propose a novel algorithm that exploits LTL progression and off-policy RL to speed up learning without compromising convergence guarantees, and show that our method outperforms the state-of-the-art approach on randomly generated Minecraft-like grids.

Linear Temporal Logic (LTL) and then defining reward functions that provide positive reward for their successful completion. LTL is a propositional, modal temporal logic first developed for the verification of reactive systems [35]. It augments propositional logic with modalities such as $\Diamond$ (*eventually*), $\Box$ (*always*), and U (*until*) in support of expressing statements such as *"Always if clothes are on the floor, put them in the hamper"* or *"Eventually make dinner."* Such statements can be combined via logical connectives and nesting of modal operators to provide task specifications. The syntax is natural and compelling and, as a formal language, it has a well-defined

Decompose tasks into subtasks with LTL progression



LTL progression

Given an LTL formula φ and state s, we can *progress* φ using s:

- $\text{prog}(s, p) = \text{true if } p \in L(s)$, where $p \in \mathcal{P}$
- $\text{prog}(s, p) = \text{false if } p \notin L(s)$, where $p \in \mathcal{P}$
- $\text{prog}(s, \neg\varphi) = \neg\,\text{prog}(s, \varphi)$
- $\text{prog}(s, \varphi_1 \wedge \varphi_2) = \text{prog}(s, \varphi_1) \wedge \text{prog}(s, \varphi_2)$
- $\text{prog}(s, \bigcirc\varphi) = \varphi$
- $\text{prog}(s, \Diamond\varphi) = \text{prog}(s, \varphi) \vee \Diamond\varphi$
- $\text{prog}(s, \varphi_1 \,\mathsf{U}\, \varphi_2) = \text{prog}(s, \varphi_2) \vee (\text{prog}(s, \varphi_1) \wedge \varphi_1 \,\mathsf{U}\, \varphi_2)$

Task with finite-episode -> restriction to co-safe properties