Coupling Reinforcement Learning with Temporal Logic Rewards

Taisa Kushner & Allie Morgan

December 14th 2017

Overview

- Motivation
  - optimization
  - temporal logic
- What is **Reinforcement Learning (RL)**?
- What is **S-TaLiRo**?
- Coupling RL with temporal logic via S-TaLiRo
- Demo — application to a simple pendulum
- Results:
  - (1) RL, (2) S-TaLiRo, (3) RL + S-TaLiRo
- Conclusions and future work

Optimization

What are we doing?

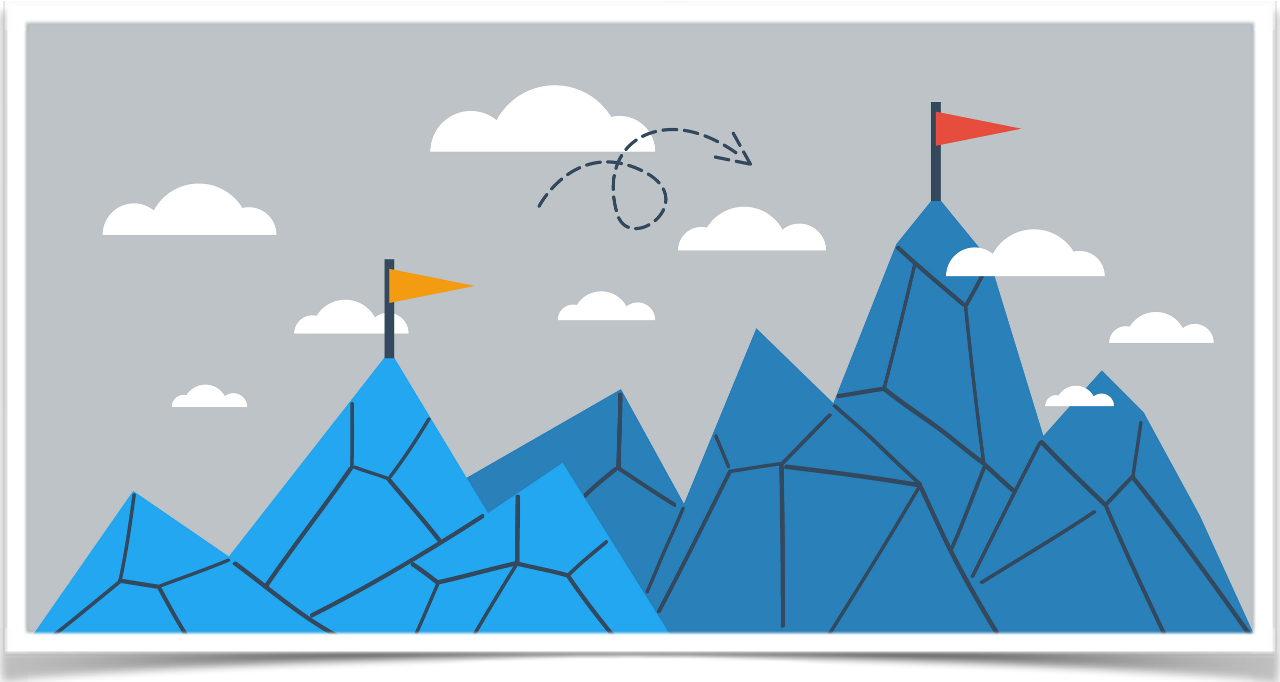
- trying to determine an optimal sequence of actions
- maximize a reward function

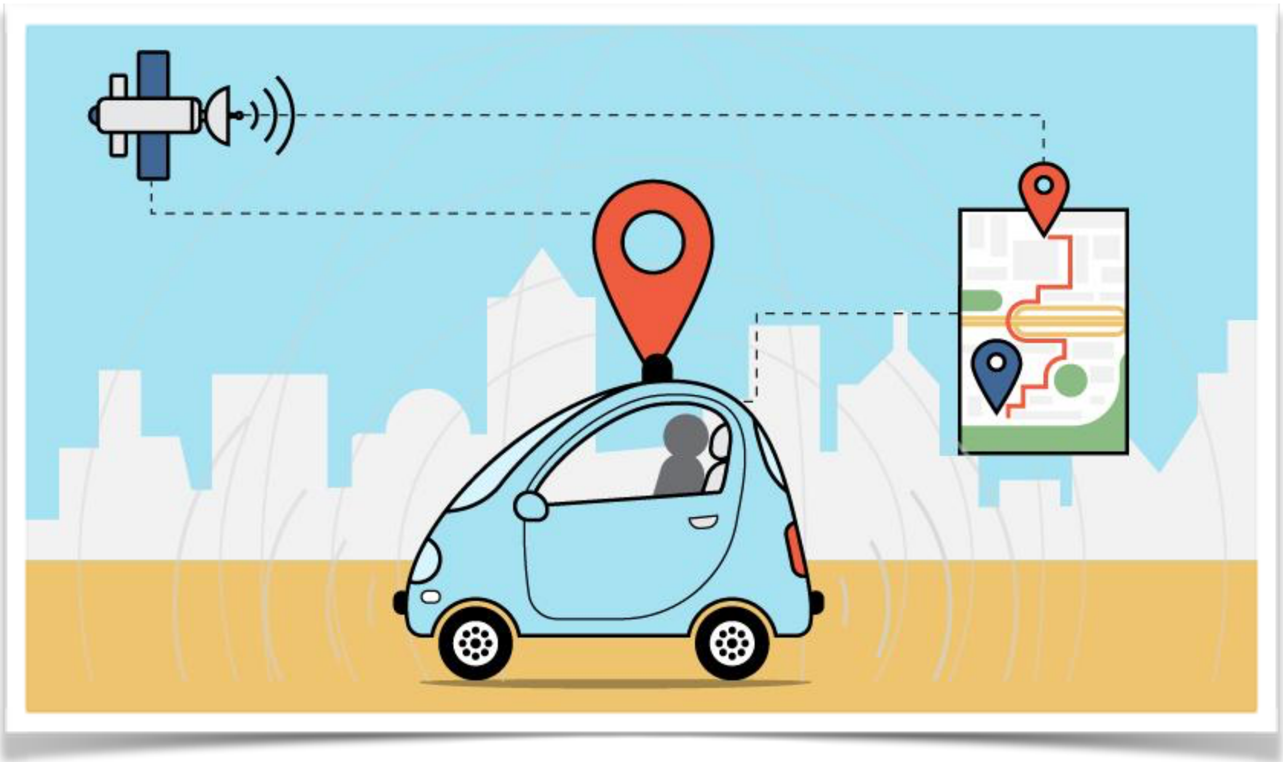
Many approaches

- convex programming, quadratic programming, nonlinear programming, combinatorial optimization, space mapping, etc.

Reinforcement learning as optimization

- optimizing, but not necessarily optimal





https://unbounce.com/landing-pages/next-level-landing-page-optimization/,
https://www.pcmag.com/feature/348851/9-ways-driverless-cars-will-change-your-life

Temporal logic

Reasoning about qualities of a program in **terms of time**

$$\forall s.(f(s) \wedge g(s)) \leftrightarrow (\forall s.f(s)) \wedge (\forall s.g(s))$$

$$\downarrow$$

$$\begin{aligned}
s_{t:t+k} &\models f(s) < c & \Leftrightarrow \quad & f(s_t) < c, \\
s_{t:t+k} &\models \neg\phi & \Leftrightarrow \quad & \neg(s_{t:t+k} \models \phi), \\
s_{t:t+k} &\models \phi \Rightarrow \psi & \Leftrightarrow \quad & (s_{t:t+k} \models \phi) \Rightarrow (s_{t:t+k} \models \psi), \\
s_{t:t+k} &\models \phi \wedge \psi & \Leftrightarrow \quad & (s_{t:t+k} \models \phi) \wedge (s_{t:t+k} \models \psi), \\
s_{t:t+k} &\models \phi \vee \psi & \Leftrightarrow \quad & (s_{t:t+k} \models \phi) \vee (s_{t:t+k} \models \psi), \\
s_{t:t+k} &\models \bigcirc\phi & \Leftrightarrow \quad & (s_{t+1:t+k} \models \phi) \wedge (k > 0), \\
s_{t:t+k} &\models \Box\phi & \Leftrightarrow \quad & \forall t' \in [t, t+k)\ s_{t':t+k} \models \phi, \\
s_{t:t+k} &\models \Diamond\phi & \Leftrightarrow \quad & \exists t' \in [t, t+k)\ s_{t':t+k} \models \phi, \\
s_{t:t+k} &\models \phi\, \mathcal{U}\, \psi & \Leftrightarrow \quad & \exists t' \in [t, t+k)\ s.t.\ s_{t':t+k} \models \psi \\
& & & \wedge (\forall t'' \in [t, t')\ s_{t'':t'} \models \phi), \\
s_{t:t+k} &\models \phi\, \mathcal{T}\, \psi & \Leftrightarrow \quad & \exists t' \in [t, t+k)\ s.t.\ s_{t':t+k} \models \psi \\
& & & \wedge (\exists t'' \in [t, t')\ s_{t'':t'} \models \phi).
\end{aligned}$$

Li, Xiao, Cristian-Ioan Vasile, and Calin Belta. "Reinforcement Learning With Temporal Logic Rewards." arXiv preprint arXiv:1612.03471 (2016).

Advantages

- can easily express complicated statements & rules to follow

Heuristic reward function

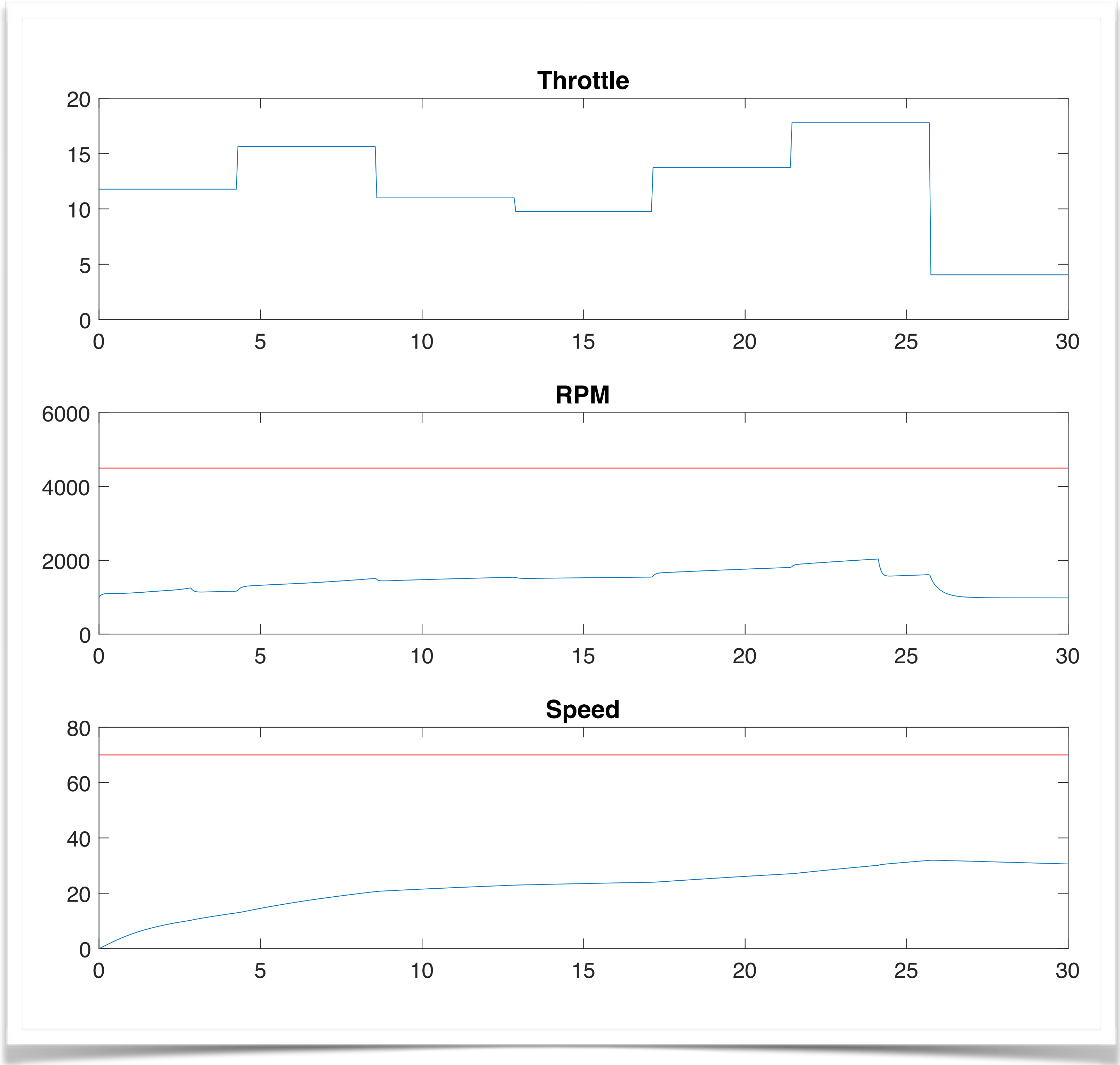
- “go from a to b in shortest path”

Temporal logic

- “go from a to b in shortest path $\wedge$ keep speed between 60-70 MPH $\wedge$ acceleration under 4500 RMP $\wedge$ drive safe”

Motivating Question

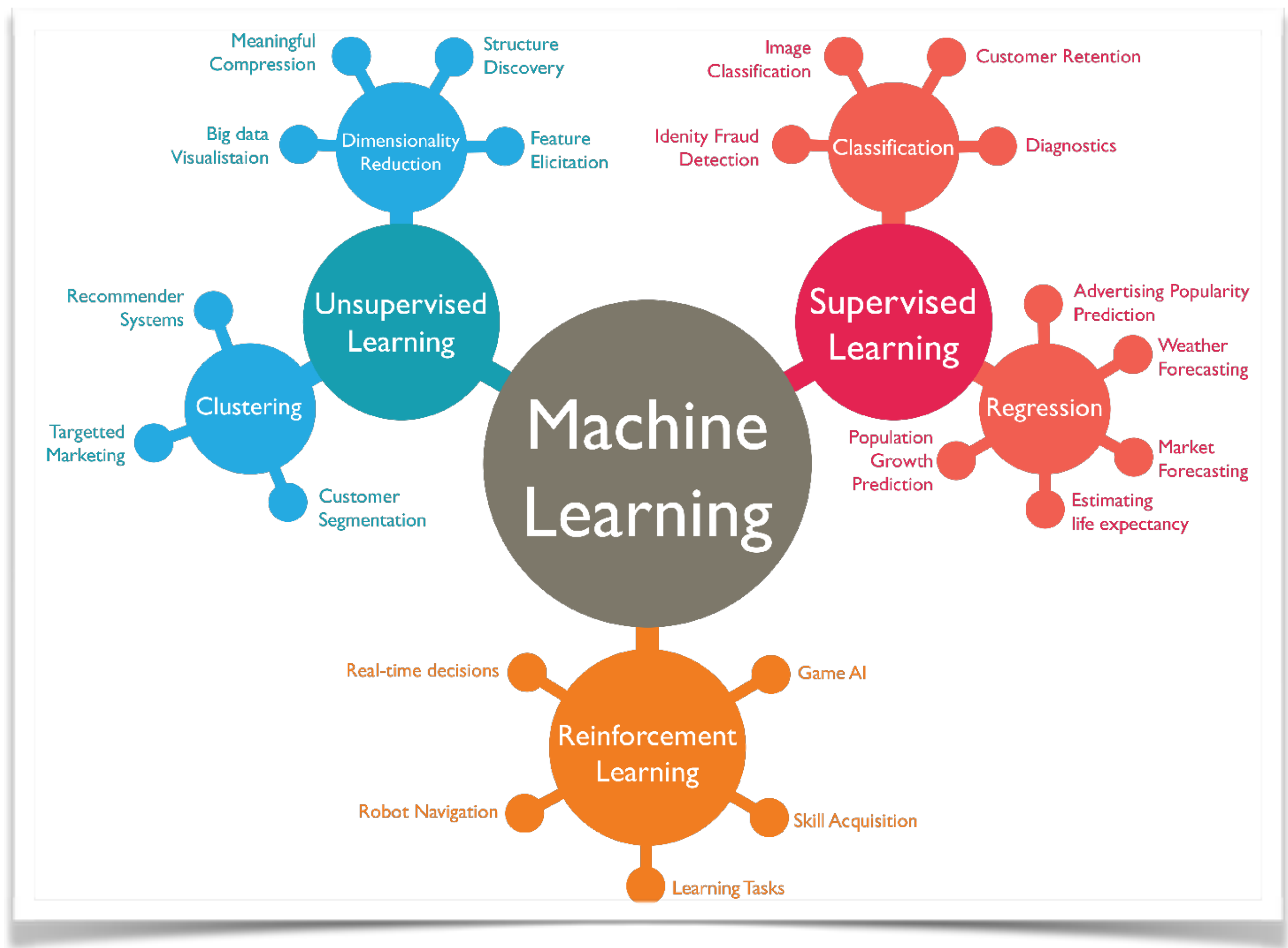
Can we couple temporal logic constraints to reinforcement learning?



Machine Learning

Three main categories

- Supervised learning
  - have training data
- Unsupervised Learning
  - no training data, try to learn **structure** in your data
- Reinforcement learning
  - no training data, try to learn **optimal action**

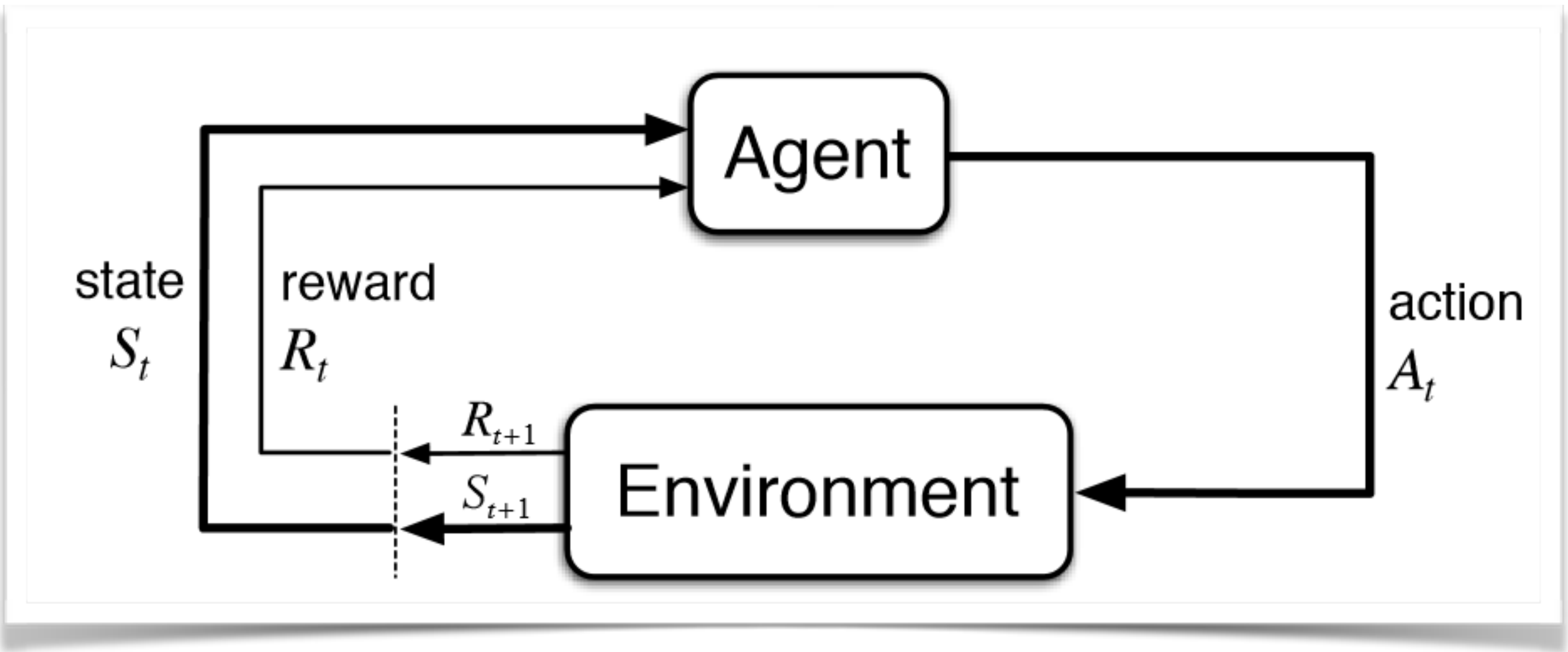


http://www.wordstream.com/images/machine-learning.png

Reinforcement Learning

Main elements

- sensation
- action
- goal



http://web.stanford.edu/class/cs234

How you carry it out

- policy
- reward signal
- value function
- model of the environment (optional)

(Greedy) Q-Learning Outline

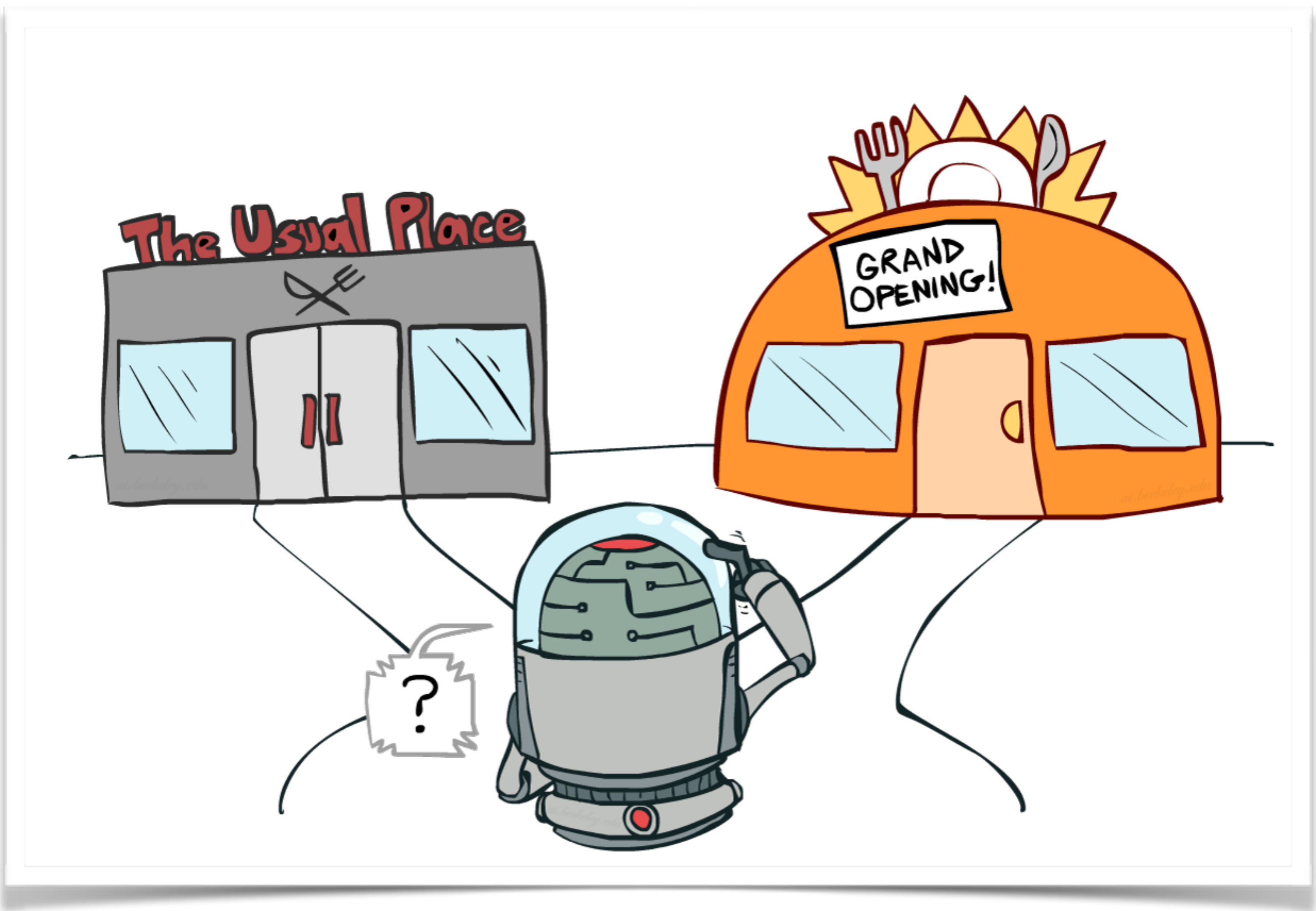
```
s_t = current state, a = action, r = reward
q = values associated with (s_t, a)-pair
epsilon = initial probability exploration:exploitation
epsilonDecay = decay per action (req to converge)
learnRate = confidence in new trials (1=all new)
discount = amount we value future:current q values
while not end of interaction do:
    if rand() > epsilon:
        a = argmax_x(q(s_t, x))
    else:
        a = choose random action
    execute(a)
    s_t+1 = update(s_t, a)

    update(q)
    q(s_t, a) = q(s_t, a) + learnRate * (r(s_t+1) + discount*max(q(s_t+1, )) - q(s_t, a) + bonus)
    s_t = s_t+1
    epsilon = epsilon*epsilonDecay
```

Effective RL

Key elements to manage

- exploration vs exploitation
- defining a reward function



http://ai.berkeley.edu/lecture_slides.html

S-TaLiRO

Performs **temporal logic falsification**

- search for trajectories of minimal robustness in system models

Important for safety verification

How we want to incorporate it:

- improve exploration and convergence of RL
- define a new reward function via robustness measure from S-TaLiRo

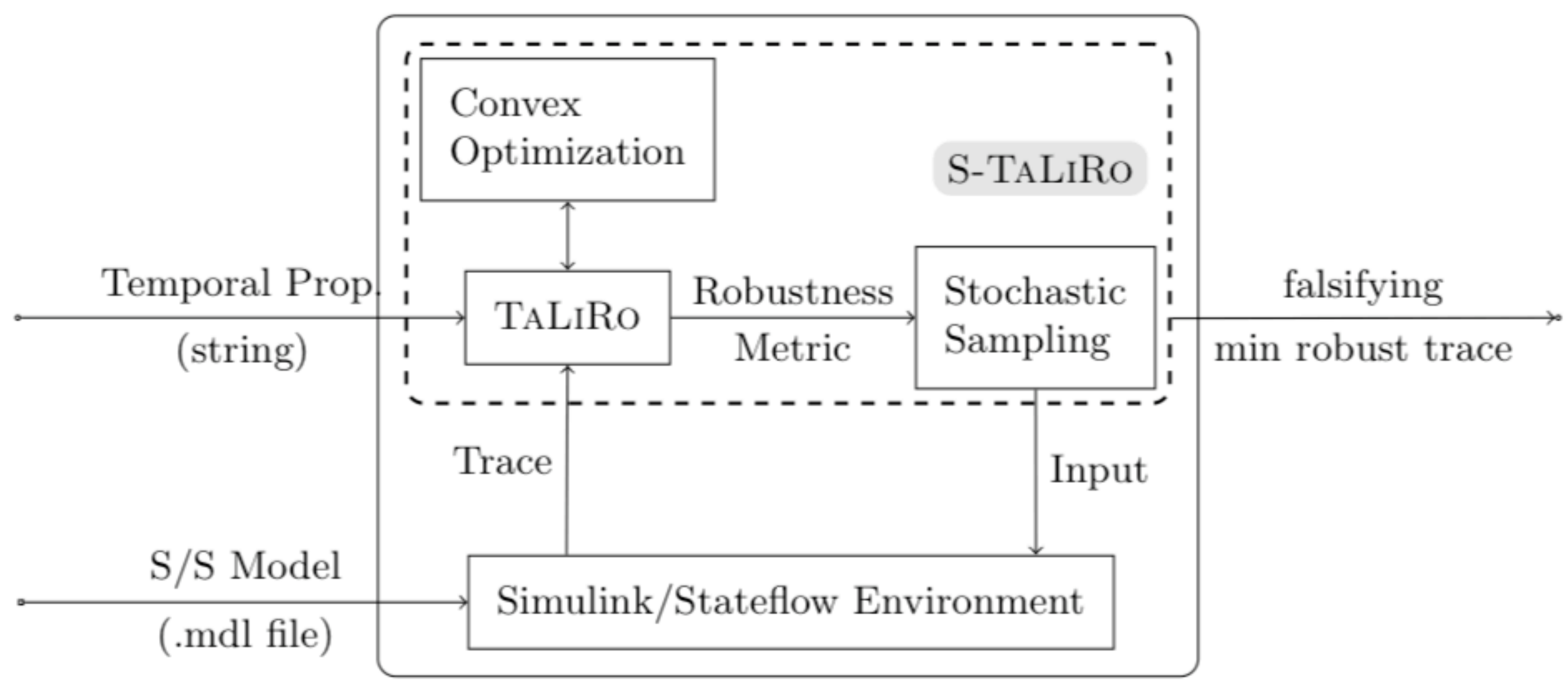
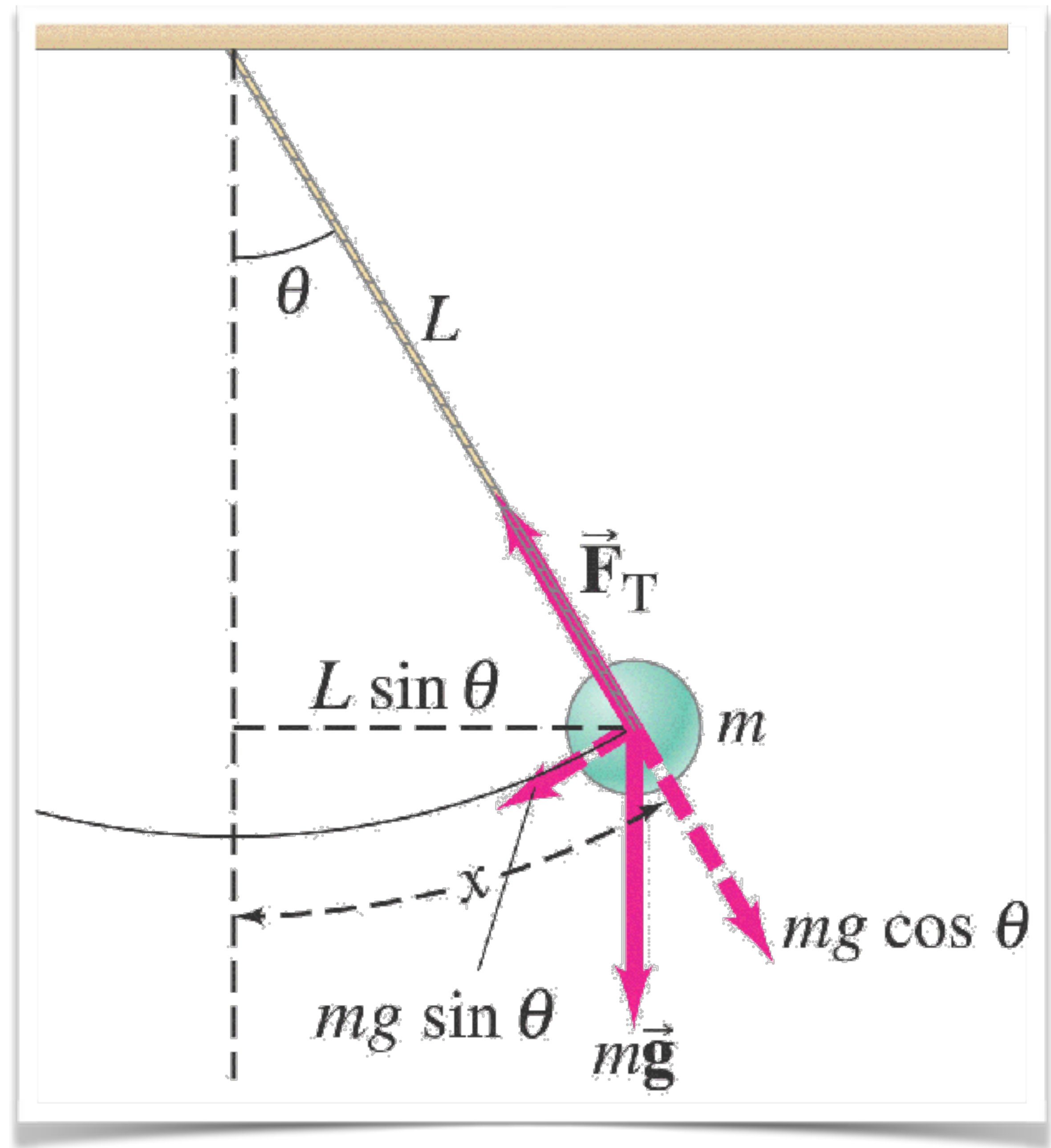


Fig. 1. The architecture of the S-TaLiRo tool.

Example Problem

Simple Pendulum

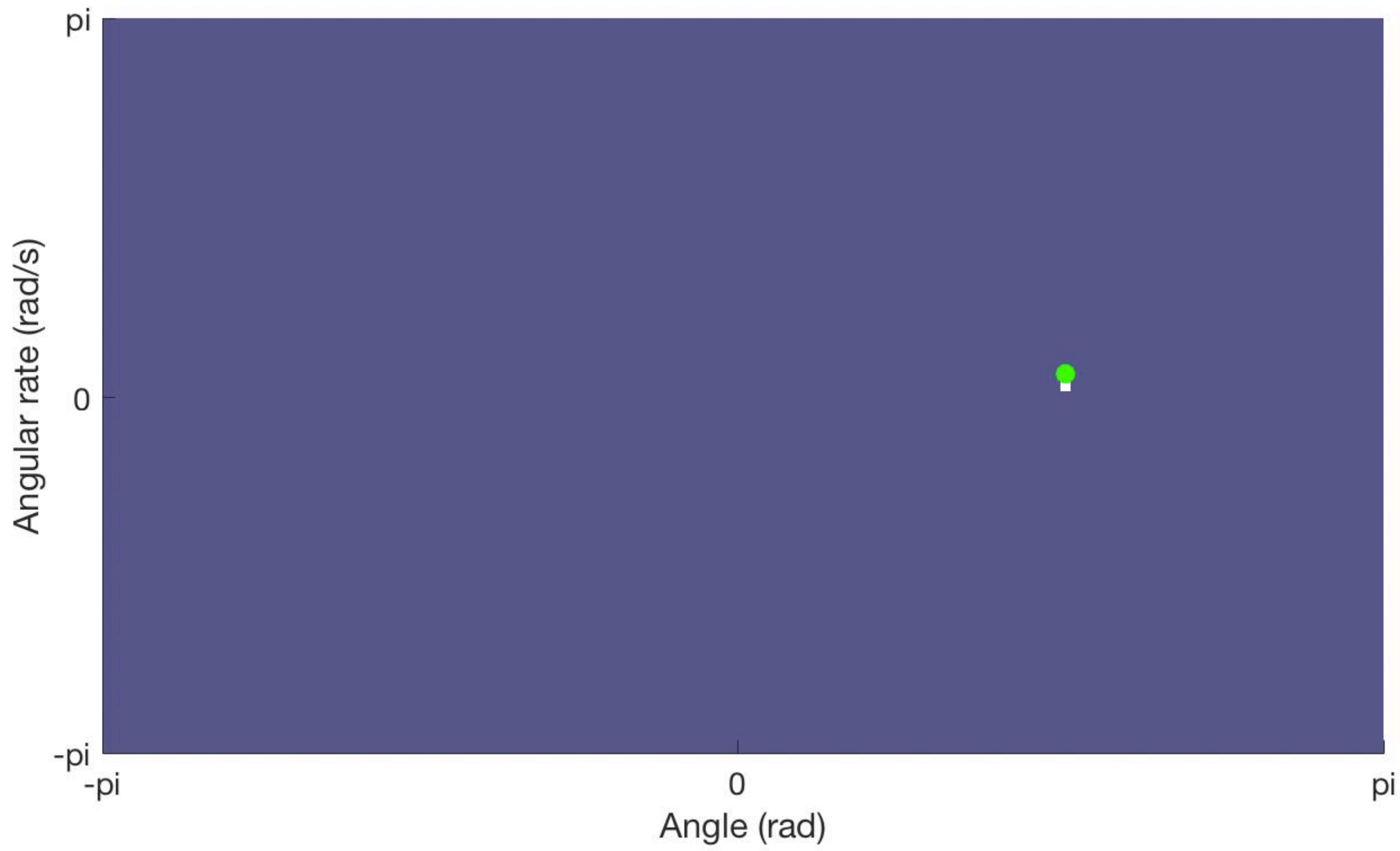
- **Objective: Control the speed**
- **Actions: Push right, left, or stay**



Giancoli, Physics: Principles with Applications. Chapter 11

RL approach

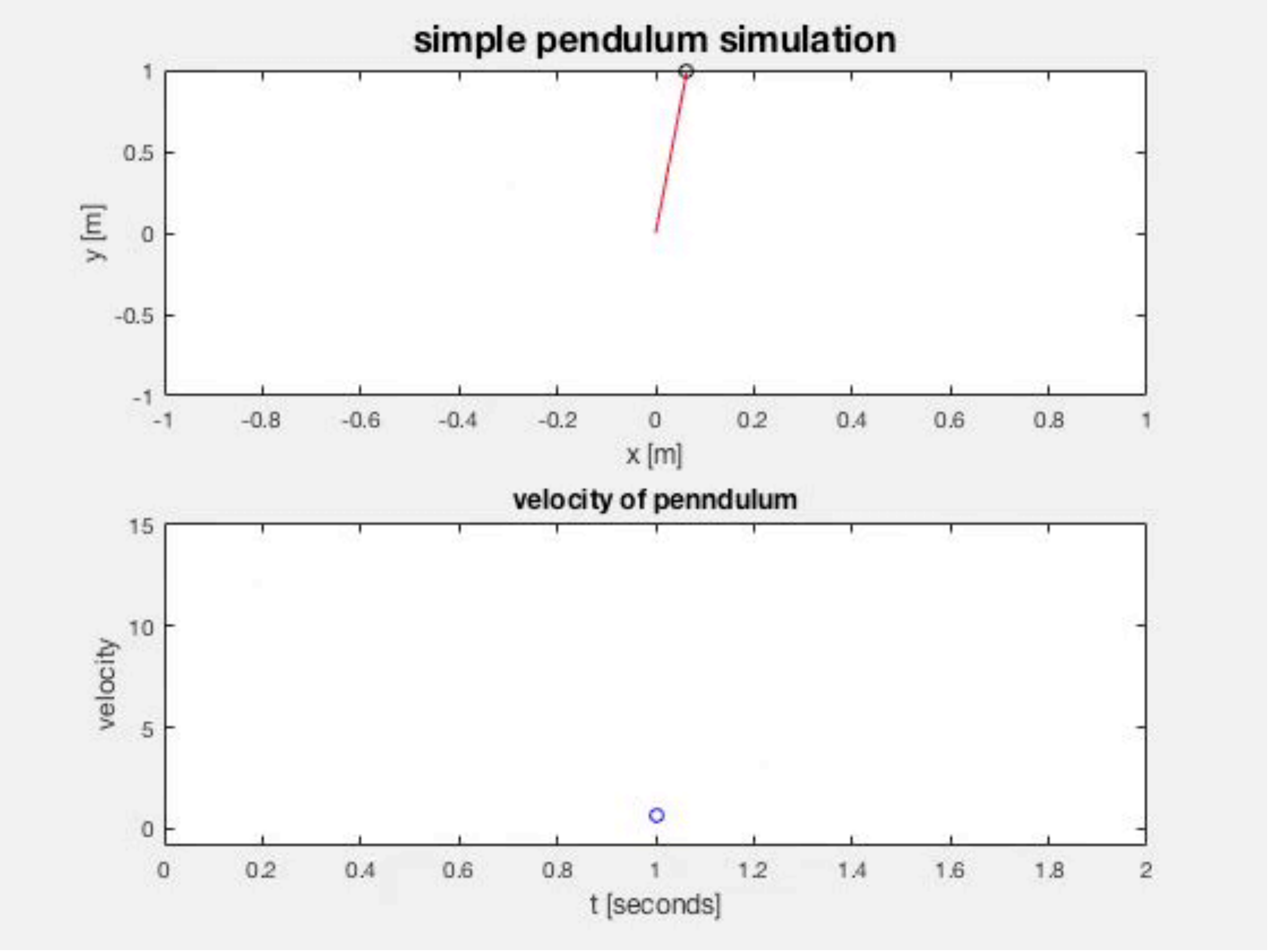
- Keep the speed at 2



S-TaLiRo approach

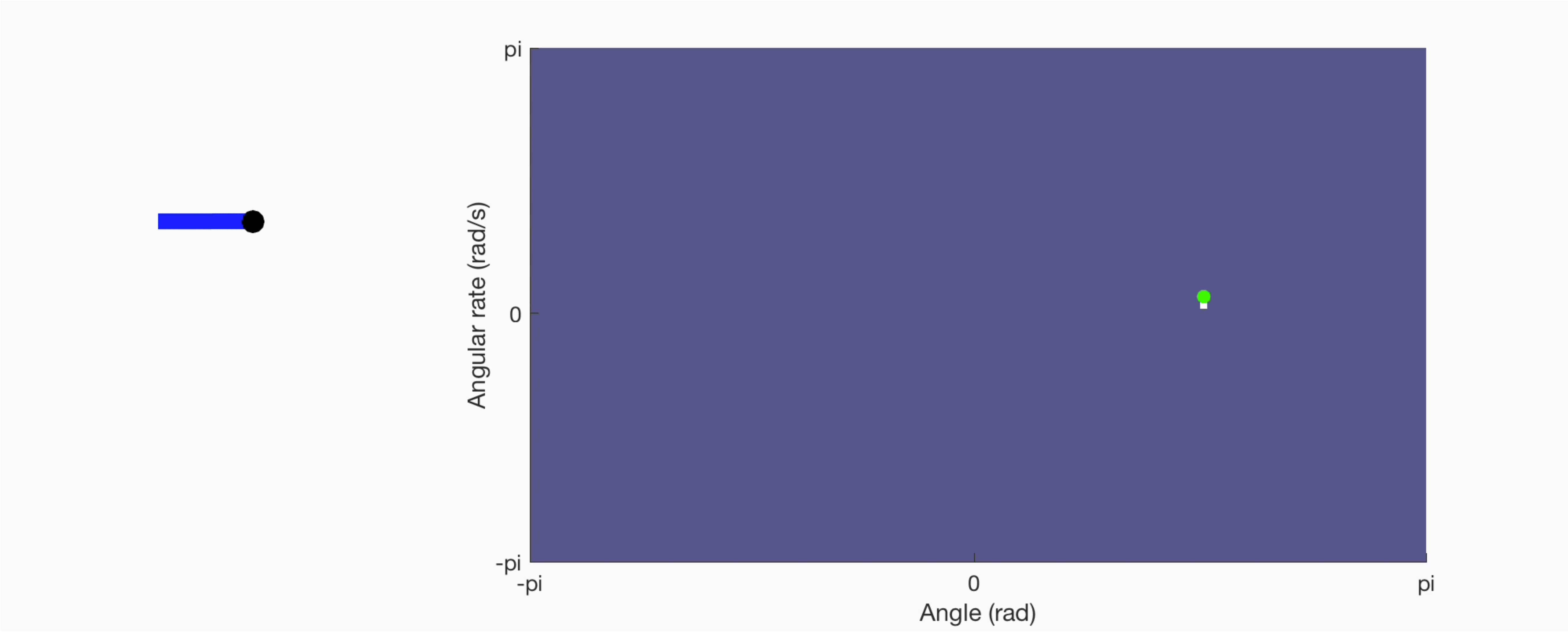
Keep the speed
under 2 until t=30,
then at 2 until 2=40,
then anything

More exciting!



RL with S-TaLiRo

- Add S-TaLiRo into the exploration step
- Note improved convergence rate



Conclusions & future work

- More exciting examples
- Efficient coupling
  - Exploit that S-TaLiRo can take sequences of actions
- Define a reward function via robustness
- Quantify tradeoff between explore & exploit
  - Time steps to converge