FORMAL METHODS FOR MACHINE LEARNED SYSTEMS

CS60030 Formal Systems

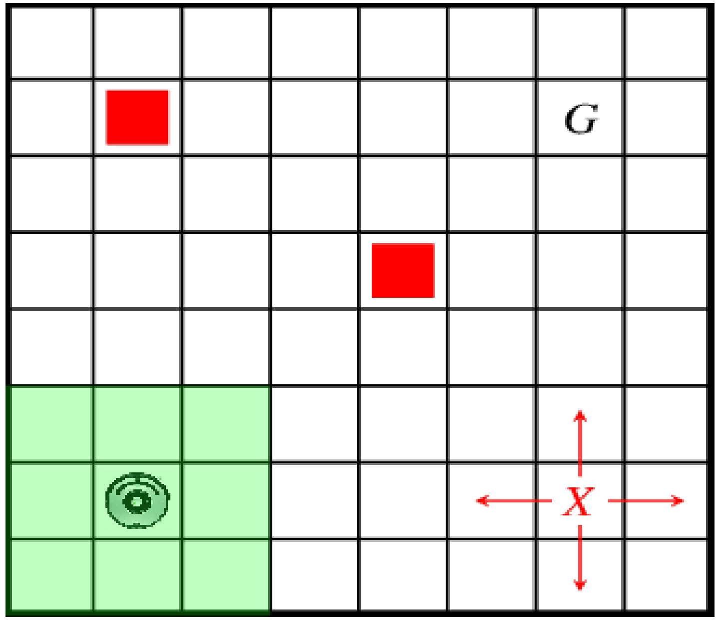
Briti Gangopadhyay
PhD Student,
Department of Computer Science and Engineering,
IIT Kharagpur
Email: briti_gangopadhyay@iitkgp.ac.in

INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

FMSAFE

FORMAL METHODS FOR SAFETY CRITICAL SYSTEMS

Machine Learned Controllers

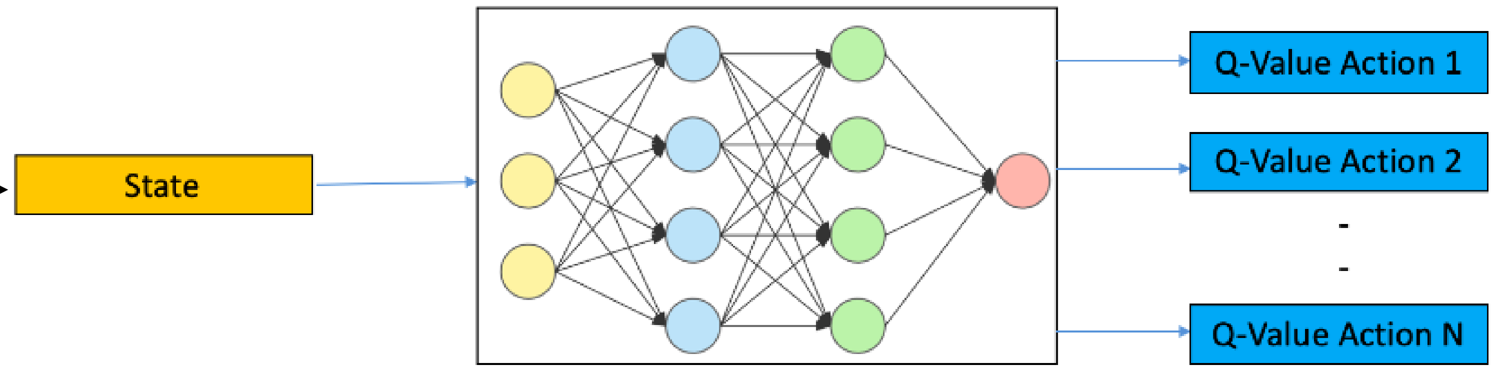


Reinforcement Learning can learn control strategies for toy problems like moving an agent in gridworld to more complex domains like autonomous driving.

Large state and action space

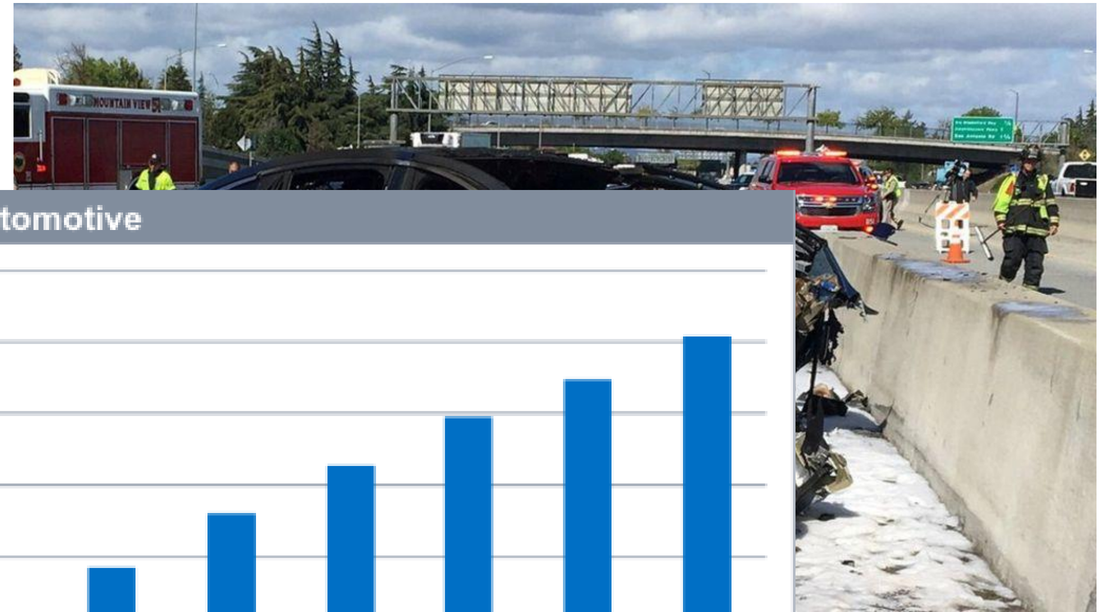
Action	↑	↓	→	←
Start	0	0	1	0
Idle	0	0	0	0
Correct Path	0	0	0	0
Wrong Path	0	0	0	0
End	0	0	0	0

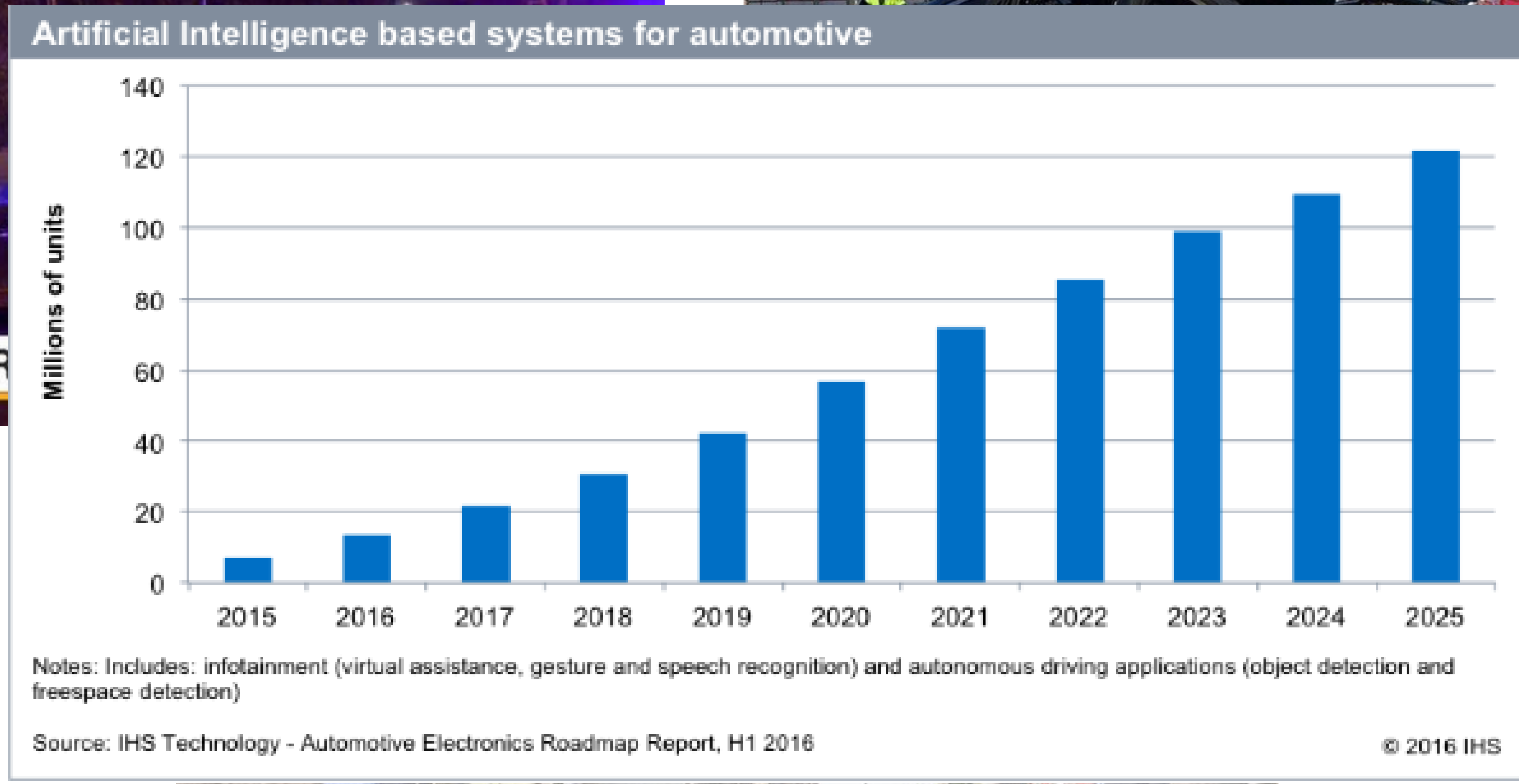
Q Table



Deep Q Neural Networks

Why Should Machine Learned Controllers be Verified?



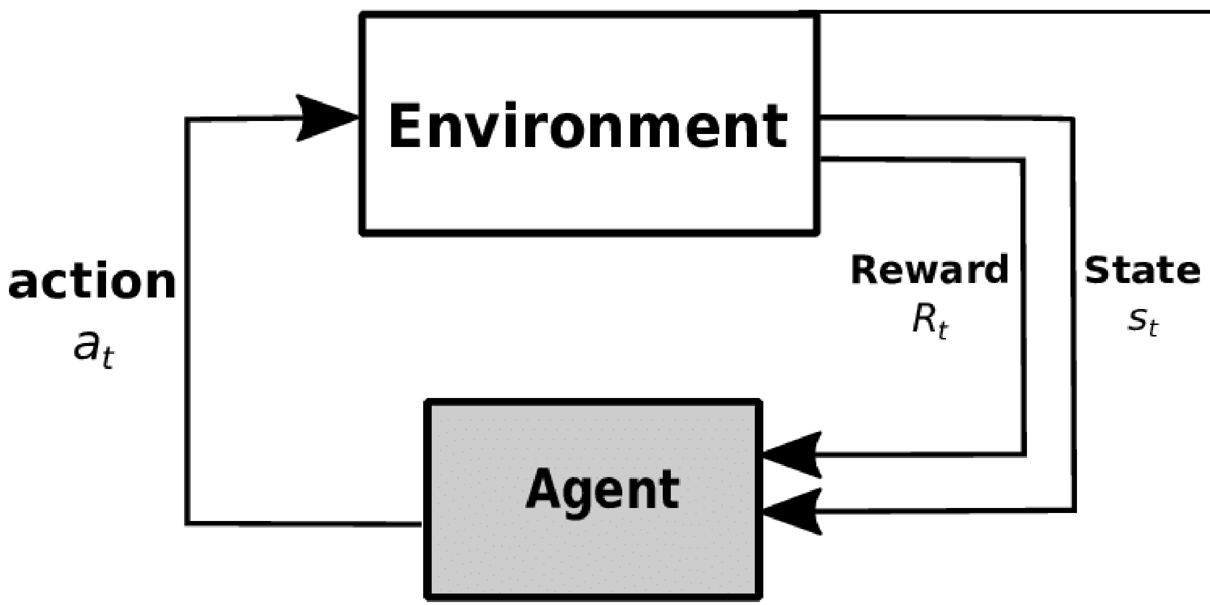


Uber Autonomous Crash March 2018

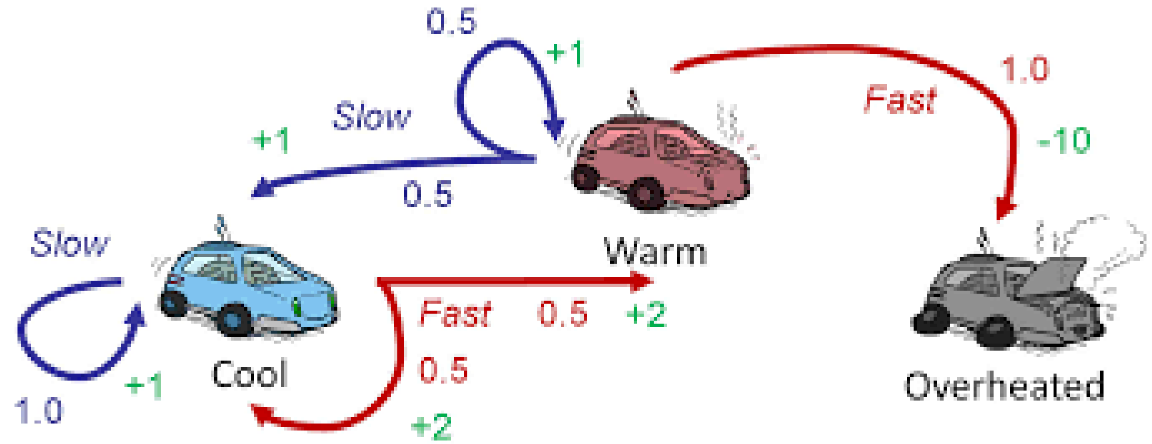
utopilot Failure 018

Boeing 737 crash 2018, 2019

Reinforcement Learning Overview



Modelled using Markov Decision Process (MDP)



Learns a policy π ($s_t a_t, s_{t+1} a_{t+1} \ldots s_{t+n} a_{t+n}$) to maximizing reward. Policy is learnt using different algorithms with Neural Network Components such as Deep Q Learning, Actor Critic, Deep Deterministic Policy Gradient Methods

$$(S, A, P_a, R_a)$$

Reinforcement Learning Overview

Learning in Reinforcement Learning is guided by the Bellman Equations which depends on two main factors:

1. **The immediate reward given for visiting a state using a particular action.**
2. **The expected reward for taking action a_{t+1} from state s_{t+1}**

$$\underbrace{q_\pi(s, a)}_{\text{Q Value}} = \mathbb{E}_\pi\left[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}) \mid S_t = s, A_t = a\right]$$

Policy

Reward

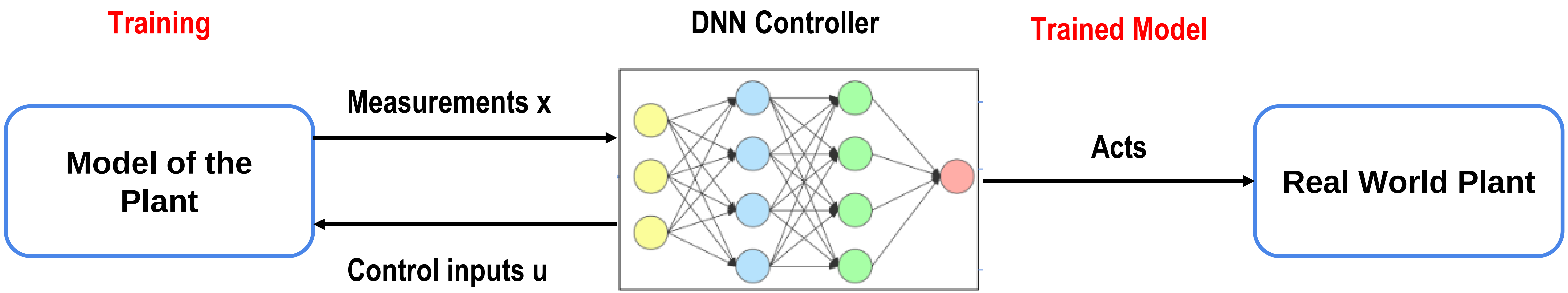
Discounted Q value over future states

State and action at time t

Reward shaping can prevent the agents from visiting unsafe states.

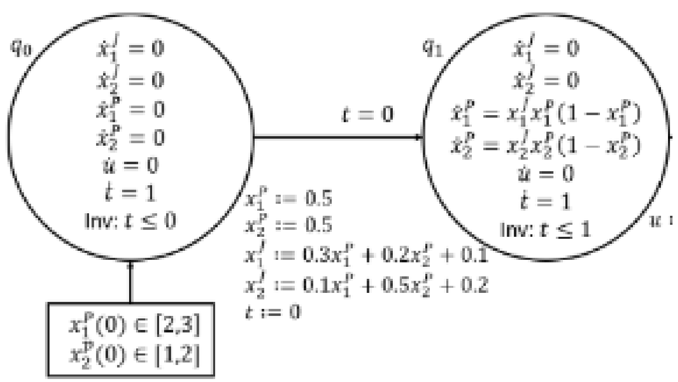
How Is Formal Methods Relevant for Reinforcement Learning?

Many control tasks are solved using Neural Networks due to their ability to learn from data and generalization capabilities. These controllers are also used in safety critical domain like autonomous driving where verification is mandatory.



Non Interpretable hence difficult to verify

DNN with L layers and N neurons per layer can be represented as a hybrid system with L + 1 modes and 2N states. [Verisig : Ivanov et el 2019]



Analyse the hybrid models with tools like Flow* to answer reachability question. Works only with DNN's having sigmoid activation. Relu activations can be verified with Reluplex. Not scalable.

How Is Formal Methods Relevant for Reinforcement Learning?

Training

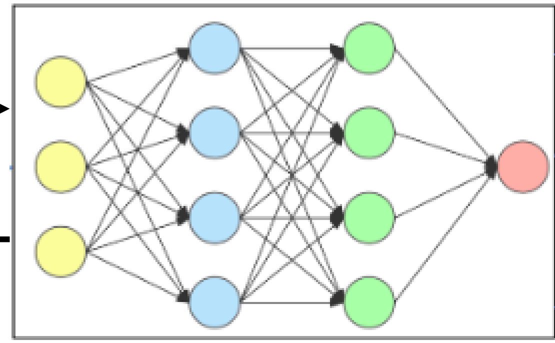
Trained Model

DNN Controller

Measurements x

Model of the Plant

Control inputs u



Non Interpretable hence difficult to verify

SAFETY LAYER

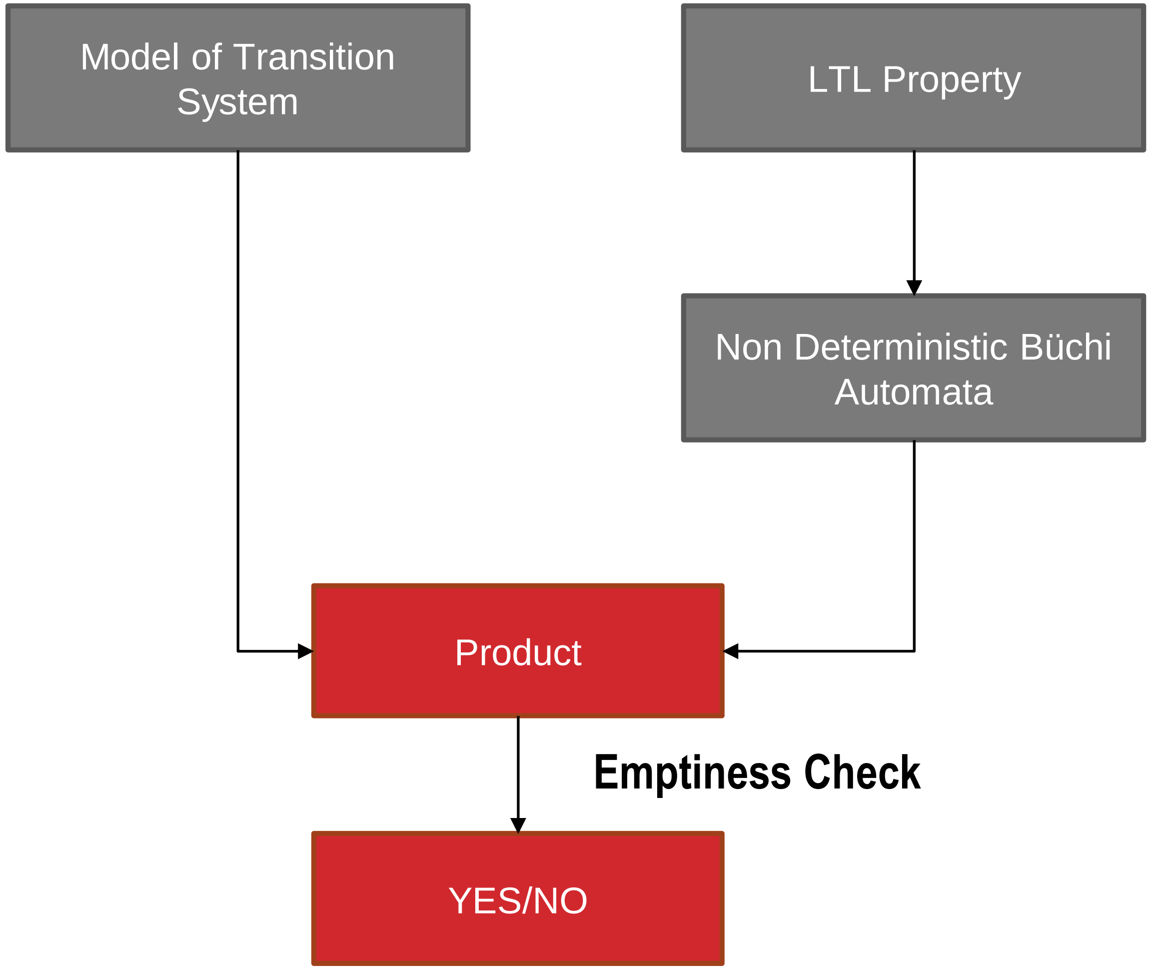
Real World Plant

Constraint the markov decision process using LTL properties such that the agent only learns to visit safe states.

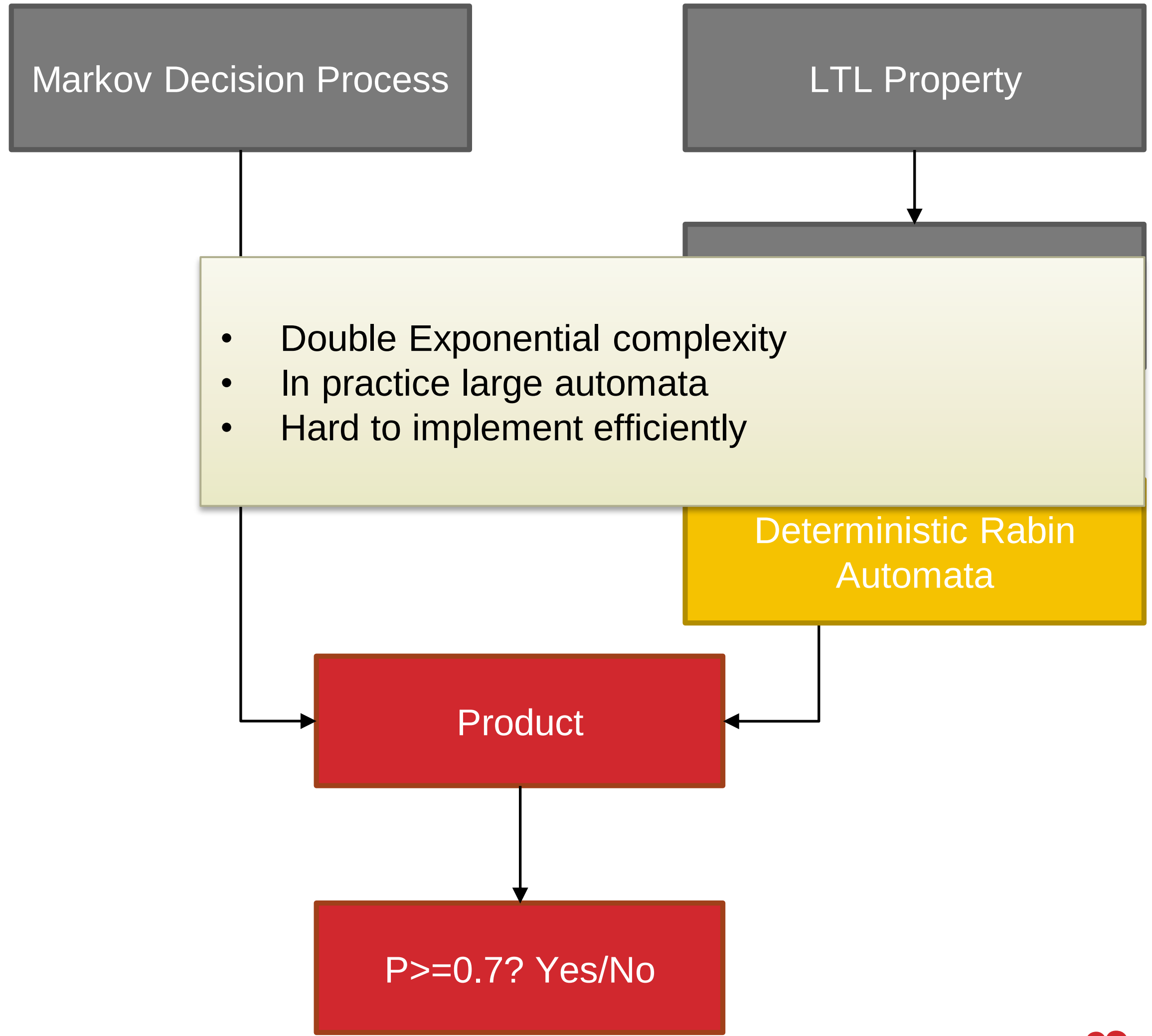
A safety layer that analytically solves an action correction formulation at each state. [Dvijotham et el 2018]

Model Checking On Markov Decision Process's

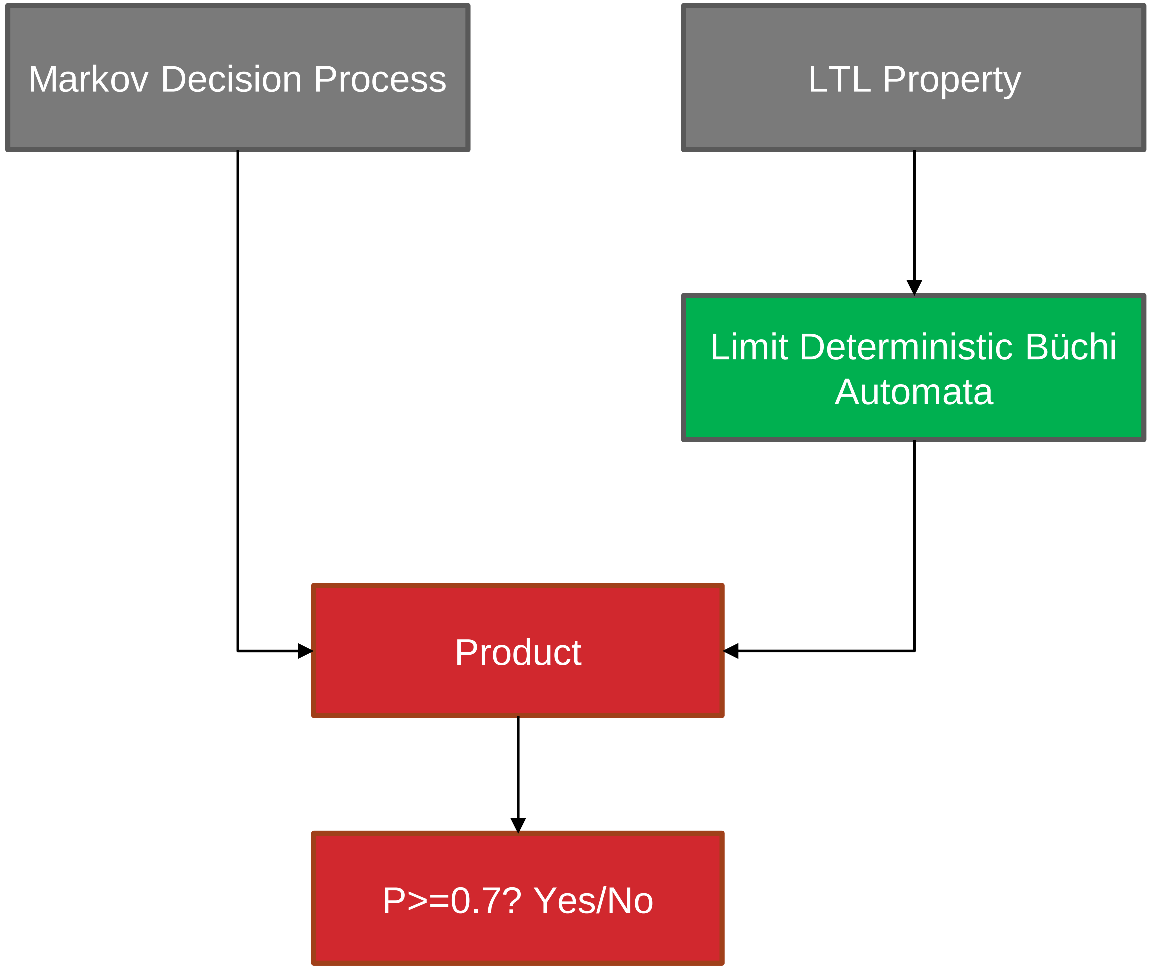
Standard Model Checking

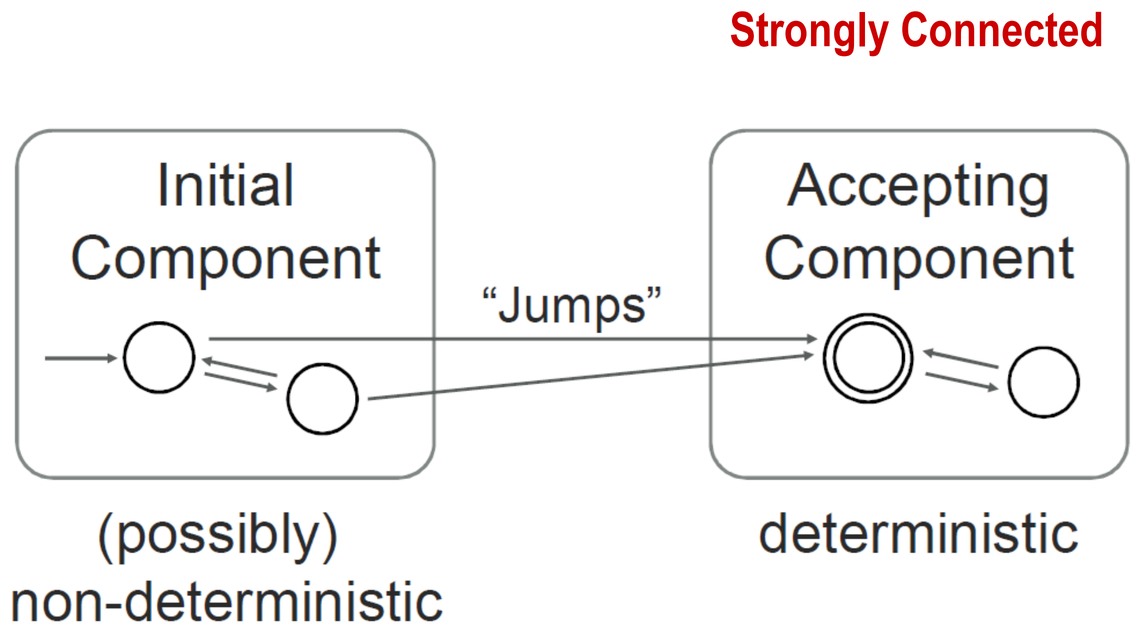


Model Checking on Probabilistic Models



Model Checking On Markov Decision Process's



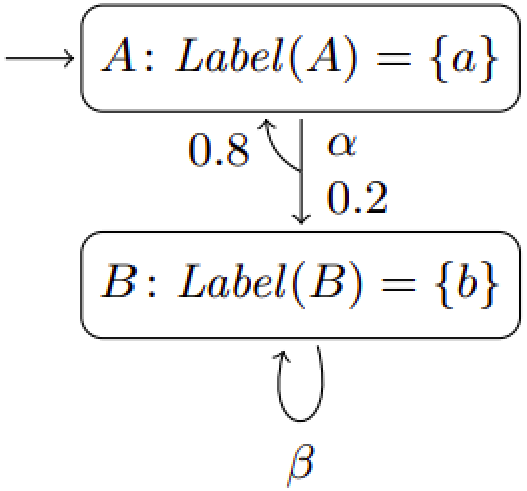


- Simpler Construction
- Smaller Automata

Markov Decision Process (MDP)

A Markov Decision Process can be defined as a tuple M = (S, A, s_0, P, AP, L) where:

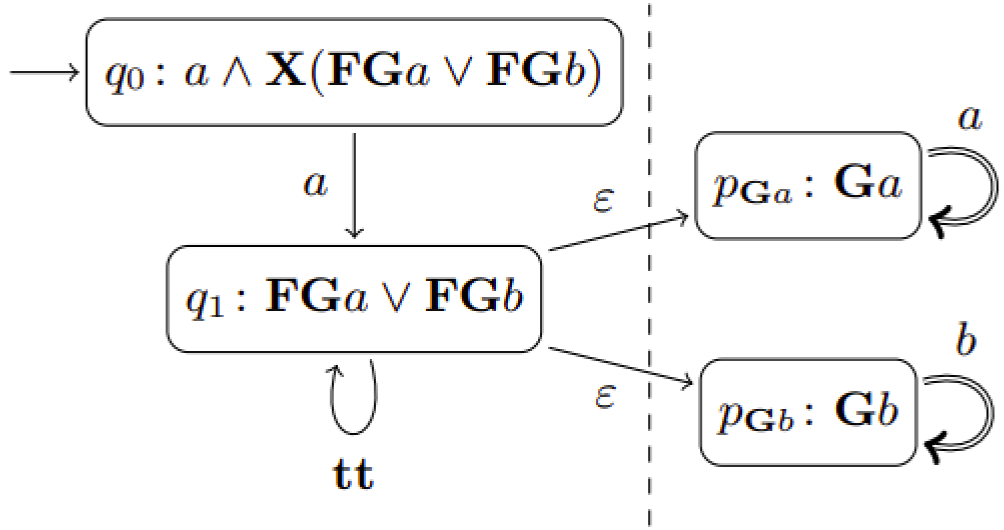
- S is a finite set of states
- A is a finite set of actions
- s_0 is the initial state
- P : S × A × S → [0, 1] is the transition probability function which determines probability of moving from a given state to another by taking an action
- AP is a finite set of atomic propositions
- L : S → 2^{AP} assigns to each state s ∈ S a set of atomic propositions L(s) ⊆ 2^{AP}.
- We use $s_i \rightarrow a \rightarrow s_j$ to denote a transition from state $s_i \in S$ to state $s_j \in S$ by action a ∈ A.



Limit Deterministic Büchi Automata (LDBA)

A Generalized Non Deterministic Büchi Automata can be defined as a tuple N = $(Q, q_0, \Sigma, F, \Delta)$ where

- Q is a finite set of states
- $q_0 \in Q$ is the initial state
- $\Sigma = 2^{AP}$ is a finite alphabet
- $F = \{F_1, ..., F_f\}$ is the set of accepting conditions
- $\Delta : Q \times \Sigma \rightarrow 2^Q$ is a transition relation.



GNBA is limit deterministic if Q can be partitioned into two disjoint sets $Q = Q_N \cup Q_D$, such that : –

For every state $q \in Q_D$ and for every $\alpha \in \Sigma : \Delta(q, \alpha) \subset Q_D$ and $|\Delta(q, \alpha)| = 1$,

And for every $F_j \in F$, $F_j \subset Q_D$

LDBA and MDP Product

Given an MDP $M = (S, A, s_0, P, AP, L)$ and an LDBA $N = (Q, q_0, \Sigma, F, \Delta)$ with $\Sigma = 2^{AP}$, the product MDP is defined as

$M \otimes N = MN = (S^{\otimes}, A, s^{\otimes}_0, P^{\otimes}, AP^{\otimes}, L^{\otimes})$, where

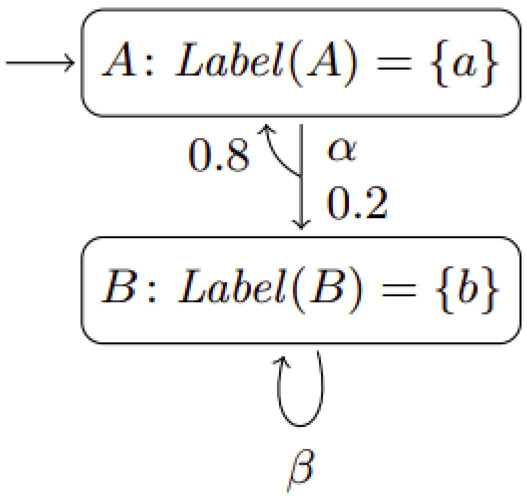
- $S^{\otimes} = S \times Q$ is a set of product states,
- $s^{\otimes}_0 = (s_0, q_0)$ is the initial state of the product MDP,
- $AP^{\otimes} = Q$,
- $L^{\otimes} = S \times Q \rightarrow 2^{Q}$ such that $L^{\otimes}(s, q) = q$,
- $P^{\otimes} : S^{\otimes} \times A \times S^{\otimes} \rightarrow [0, 1]$ is the transition probability function such that $(s_i \rightarrow a \rightarrow s_j) \wedge (q_i \rightarrow L(s_i) \rightarrow q_j) \Rightarrow P^{\otimes}((s_i, q_i), a, (s_j, q_j)) = P(s_i, a, s_j)$.
- Over the states of the product MDP we also define accepting condition $F^{\otimes} = \{F^{\otimes}_1, ..., F^{\otimes}_f\}$ where $F^{\otimes}_j = S \times F_j$.

[Sickert et el 2016]

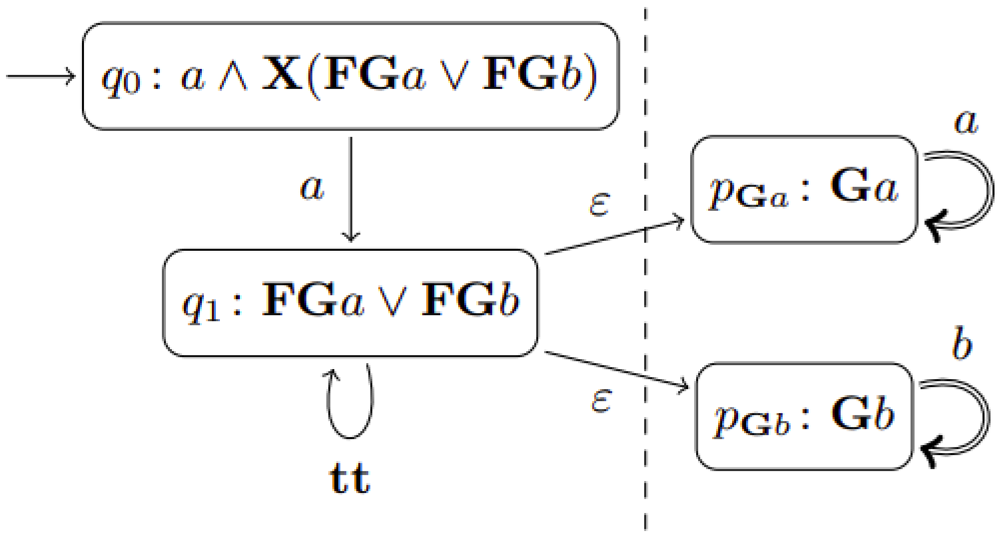
LDBA and MDP Product Example

1. $s^{\otimes}_0 = (s_0, q_0)$,
2. $AP^{\otimes} = Q$, $L^{\otimes} = S \times Q \rightarrow 2^Q$ such that $L^{\otimes}(s, q) = q$,
3. $P^{\otimes} : S^{\otimes} \times A \times S^{\otimes} \rightarrow [0, 1]$ such that $(s_i \rightarrow a \rightarrow s_j) \wedge (q_i \rightarrow L(s_i) \rightarrow q_j) \Rightarrow P^{\otimes}((s_i, q_i), a, (s_j, q_j)) = P(s_i, a, s_j)$.

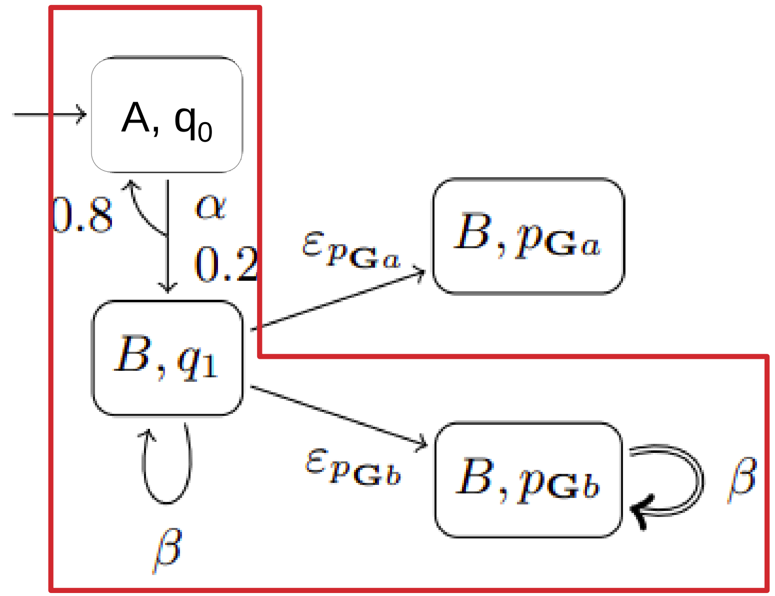
MDP



LDBA



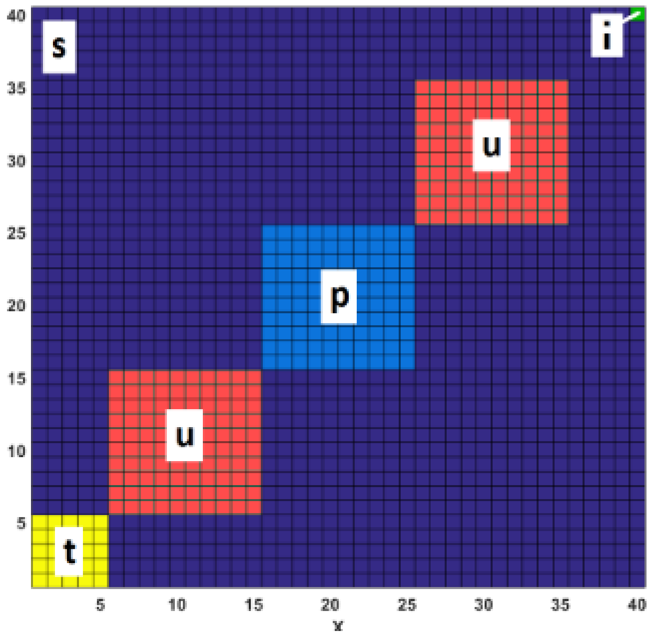
Product MDP



Logically Constrained Reinforcement Learning

1. The Reward function is defined over the product Markov Decision Process.
2. The product MDP is a synchronous structure encompassing transition relations of the original MDP and also the structure of the Büchi automaton.
3. A always contains those accepting states that are needed to be visited at a given time.
4. The Reinforcement Learning agent learns on the product Markov decision process using the Bellman equation.

$$R(s^{\otimes}, a) = \begin{cases} r_p & \text{if } q' \in \mathbb{A},\ s^{\otimes\prime} = (s', q'), \\ r_n & \text{otherwise.} \end{cases}$$



Example of Slippery Gridworld: A = {left, right, up, down, stay}

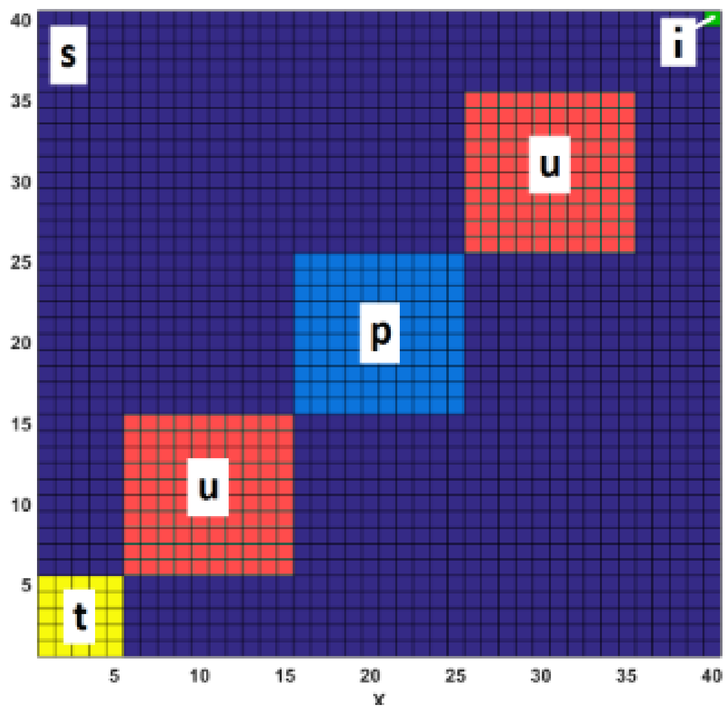
There is a probability of 85% that the action takes the robot to the correct state and 15% that the action takes the robot to a random state in its neighbourhood.

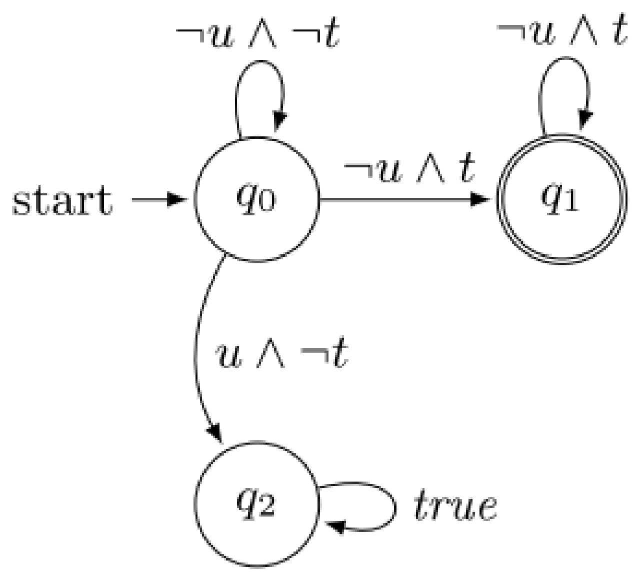
t stands for “target”, u stands for “unsafe”, and p refers to the area that has to be visited before visiting the area with label t.

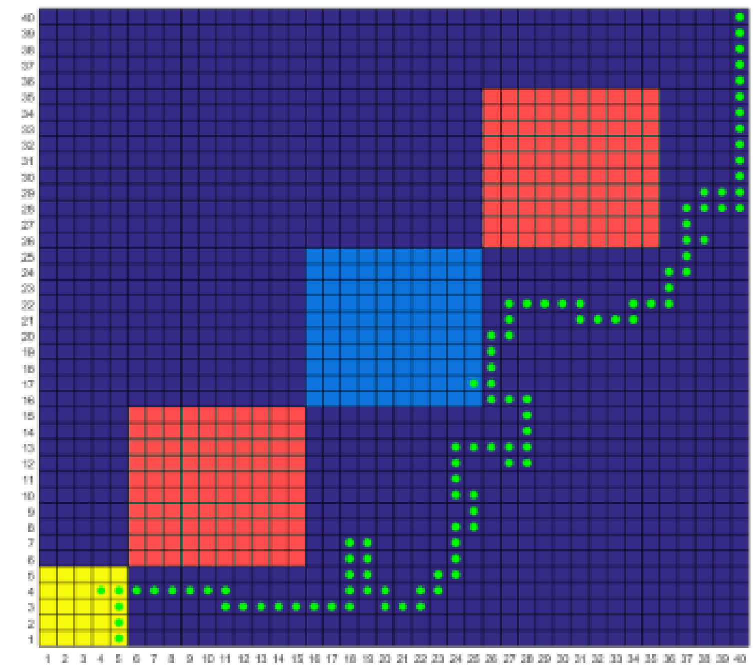
Example of Logically Constrained RL

Safety Specification : Eventually find the target Ft and stay there G(t → Gt) while avoiding the unsafe otherwise the agent is going to be trapped there G(u → Gu). [Hasanbeig et el 2018]

$$\Diamond t \wedge \Box(t \rightarrow \Box t) \wedge \Box(u \rightarrow \Box u)$$





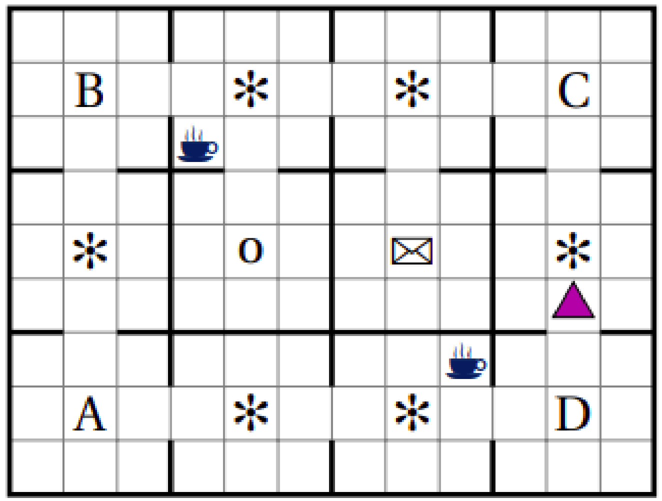


Reward Machines

- **Reward machines (RM's) are a formal representation for reward functions.**

- **LTL formulas and other regular languages can be used to specify reward-worthy behaviour that is automatically converted into RM's (via DFAs).**

- **RM structure can be exploited by Q-learning and automated reward shaping to learn policies faster, solving problems that cannot reasonably be solved otherwise. [Camacho et al. (2019)]**



Symbol	Meaning
▲	Agent
✻	Furniture
☕	Coffee machine
✉	Mail room
o	Office
A, B, C, D	Marked locations

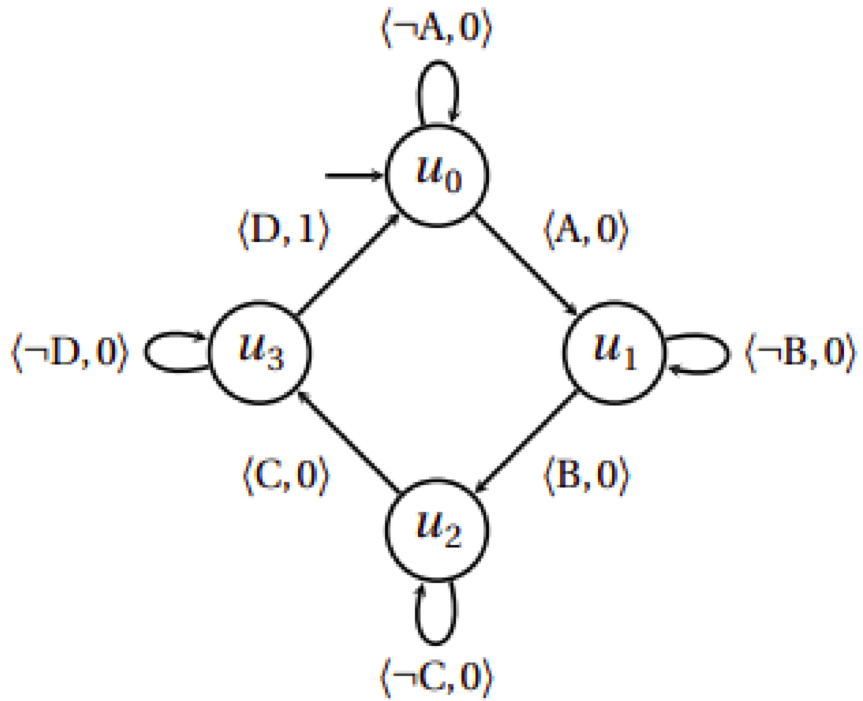
Task: Patrol A, B, C, and D.

Reward Machines

Reward Machines (RM) are Mealy machines where the input alphabet is the set of possible labels and the output alphabet is a set of reward functions. They consist of the following elements:

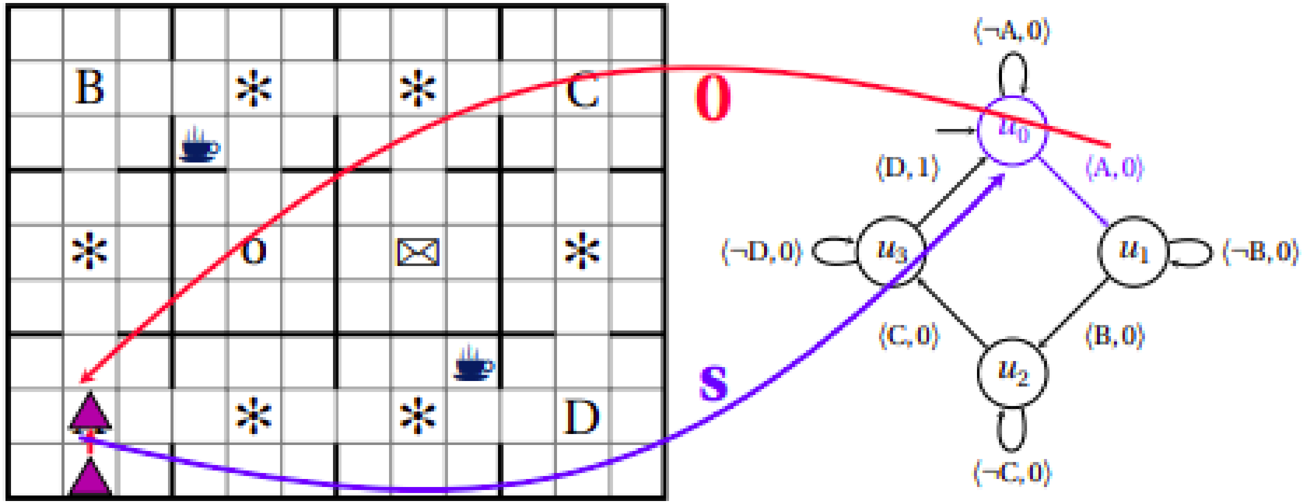
- A finite set of states U .
- An initial state $u_0 \in U$.
- A set of transitions, each labelled by:
    - a logical condition defined over the vocabulary
    - and a reward function

```
m = 0 # global variable
def get_reward(s):
  if m == 0 and s.at("A"):
    m = 1
  if m == 1 and s.at("B"):
    m = 2
  if m == 2 and s.at("C"):
    m = 3
  if m == 3 and s.at("D"):
    m = 0
    return 1
  return 0
```



Reward Machines

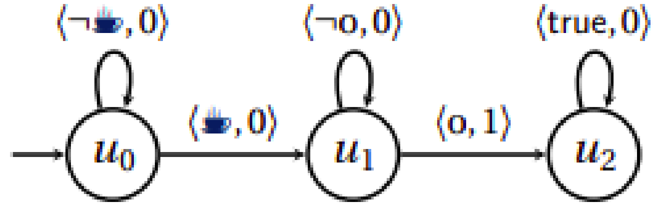
This RM starts in u_0 and transitions to u_1 when A is reached.
The agent gets reward 0 from that transition's reward function.



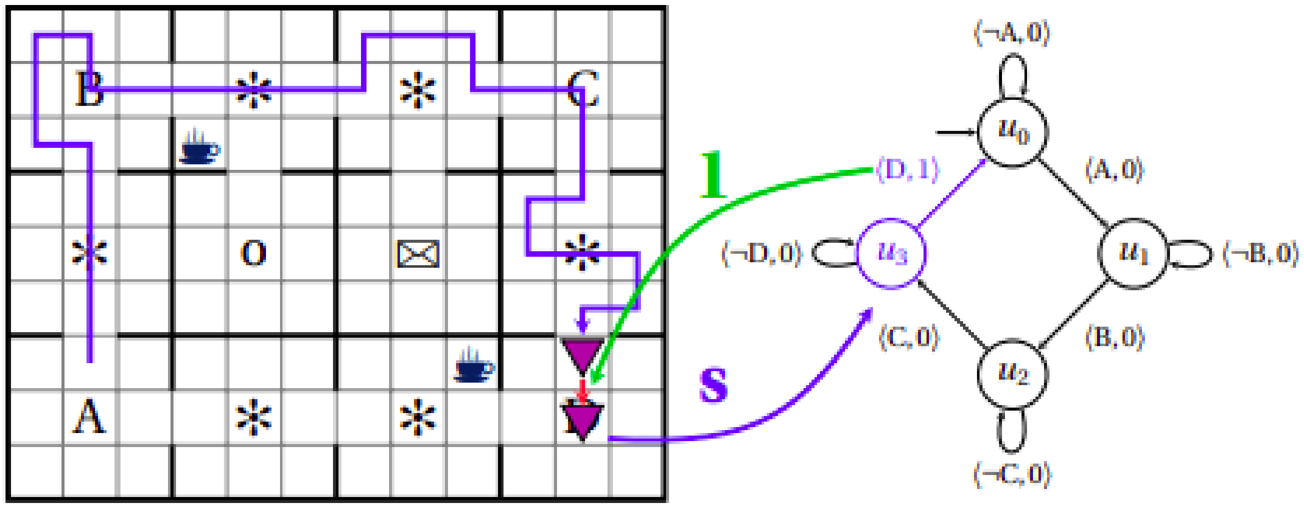
Example: "*Get coffee and bring it to the office.*"

LTL: **Eventually**[☕ $\wedge$ **Next** [**Eventually** o]]

RM:



Positive reward is given only when the agent completes a cycle.



- Observe state $\langle s, u\rangle$ and execute action $a \sim \pi(a|\langle s, u\rangle)$.
- Observe next state $\langle s', u'\rangle$ and the reward r.
- Improve policy π using experience $\langle\langle s, u\rangle, a, r, \langle s', u'\rangle\rangle$.
- $\langle s, u\rangle \leftarrow \langle s', u'\rangle$.

THANK YOU

References

1. Radoslav Ivanov, James Weimer, Rajeev Alur, George J. Pappas, and Insup Lee. 2019. Verisig: verifying safety properties of hybrid systems with neural network controllers. In Proceedings of the 22nd ACM International Conference on Hybrid Systems.
2. Kroening, D, A Abate, and M Hasanbeig. n.d. “Towards Verifiable and Safe Model-Free Reinforcement Learning.” In . Vol. 2509. CEUR Workshop Proceedings
3. Limit-Deterministic Büchi Automata for Linear Temporal Logic Computer Aided Verification, 2016, Volume 9780 SBN : 978-3-319-41539-0 Salomon Sickert, Javier Esparza, Stefan Jaax
4. MoChiBA: Probabilistic LTL Model Checking Using Limit-Deterministic Büchi Automata Automated Technology for Verification and Analysis, 2016, Volume 9938 ISBN : 978-3-319-46519-7 Salomon Sickert, Jan Křetínský
5. Mohammadhosein Hasanbeig, Alessandro Abate, and Daniel Kroening . 2018. Logically-Constrained Reinforcement Learning. arXiv preprint arXiv:1801.08099 (2018)
6. Dalal, G., Dvijotham, K., Vecerík, M., Hester, T., Paduraru, C., & Tassa, Y. (2018). Safe Exploration in Continuous Action Spaces. ArXiv, abs/1801.08757.
7. A. Camacho, R. Toro Icarte, T. Q. Klassen, R. Valenzano, and S. A. McIlraith, “LTL and beyond: Formal languages for reward function specification in reinforcement learning,” in IJCAI’19. International Joint Conferences on Artificial Intelligence Organization, 7 2019, pp. 6065–6073