Slide 1

# Learning to Plan with Logical Automata

Brandon Araki[1]*, Kiran Vodrahalli[2]*, Thomas Leech[1,3], Mark Donahue[3], Cristian-Ioan Vasile[1], Daniela Rus[1]

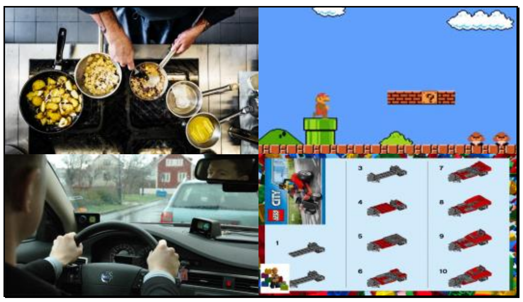
[1]Massachusetts Institute of Technology

[2]Columbia University

[3]MIT Lincoln Laboratory

*Equal contributors

Slide 2



Many environments have simple rules – for example cooking from a recipe, playing games, driving, and assembly. People are able to learn how to perform tasks like these by observing an expert. When observing an expert, people don't learn to just mimic the expert. They learn the rules that the expert is following.

This allows a person who has, for example, learned to cook a dish to modify the ingredients they put in the dish or the order in which they add ingredients.

Slide 3

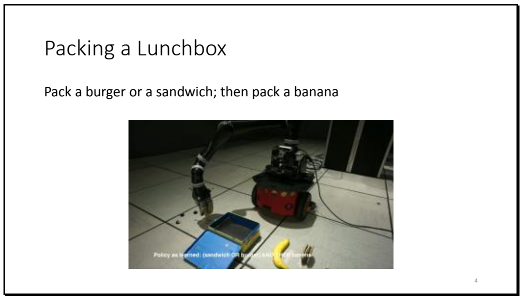
## Goals

Learn to plan in an environment with rules

1. Learn the rules in a way that they can be easily **interpreted** by humans
2. Incorporate the rules into planning so that **modifying** the rules results in predictable changes in behavior

Our goal is to replicate this ability algorithmically using model-based imitation learning.

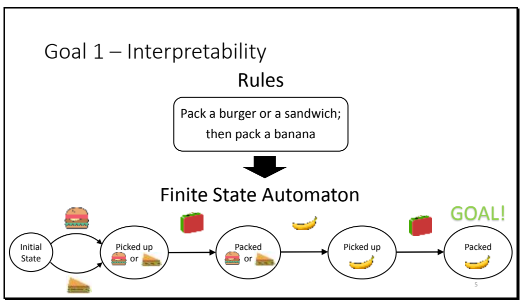
Slide 4




Let’s say you have a robot that has to pack a lunchbox.

The rules are that it has to first pack a burger or a sandwich, and then pack the banana.

Slide 5

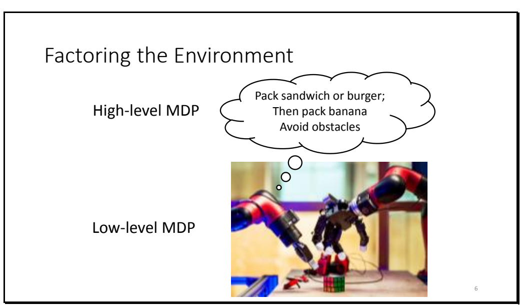



We can make these rules both useful and interpretable by representing them as a finite state automaton.

…

We assume that the environment can be factored into a high-level Markov Decision Process which is equivalent to the FSA, and a low-level MDP of the sort usually used in reinforcement learning.

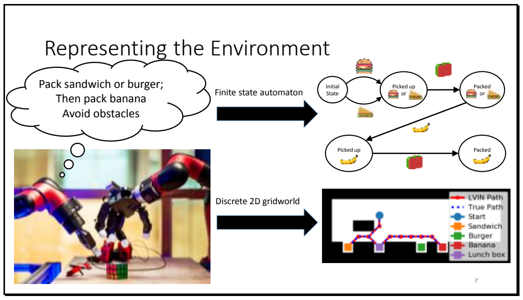
Slide 6



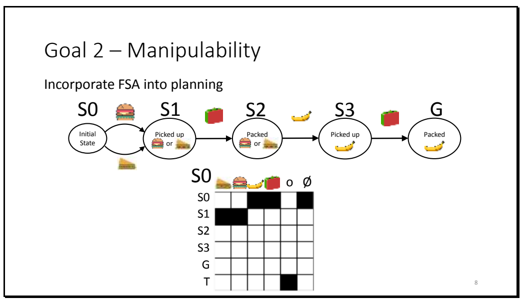

So you can imagine that we have a reinforcement learning robot arm simulator with state x, y, theta, etc, and actions such as torques or commanded positions.

We assume that there is also a high-level MDP, which embodies the rules that the robot must follow.

Slide 7




Slide 8

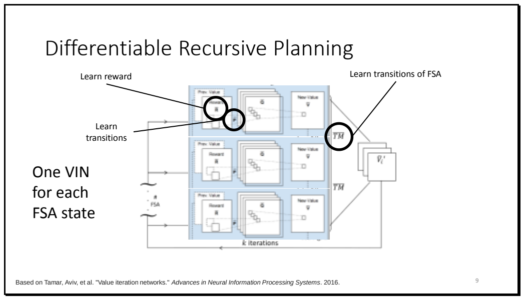



We want to be able to modify the behavior of the agent in order to make it perform similar but new tasks to the one it has learned.

We achieve this by incorporating the FSA into a recursive planning step. Since FSAs are graphs, they can be converted into a transition matrix. First it is useful to label each FSA state with a name.

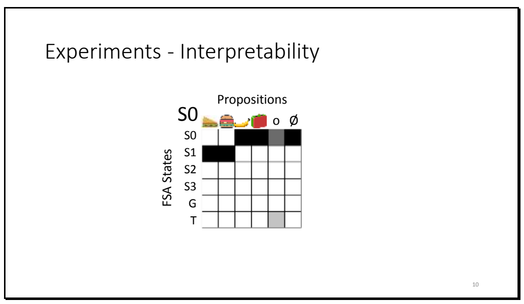
And here is the transition matrix of the first FSA state. The columns are associated with features of the environment, and the rows correspond to FSA states. You can see by looking at the graph that the sandwich and the burger cause a transition to state S1, whereas the other items do not cause a transition to a new state.

Slide 9



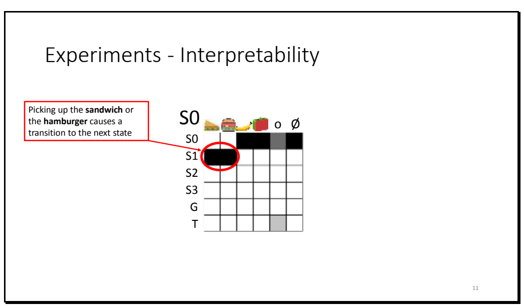

We use differentiable recursive planning to approximate value iteration and calculate a policy for the agent. The matrix form of the FSA allows us to embed the FSA as a convolution in the planning step – for more details, come to our poster session.

Slide 10

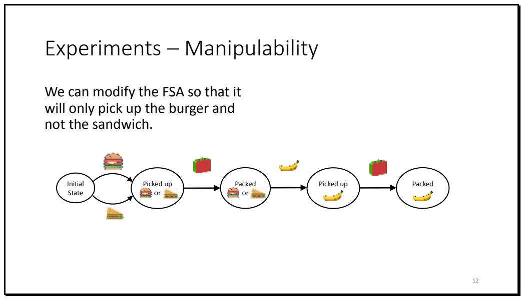



This is the learned transition matrix of the first state of the FSA.
Columns correspond to propositions, or important features of the environment.
Rows correspond to the other FSA states.

Slide 11

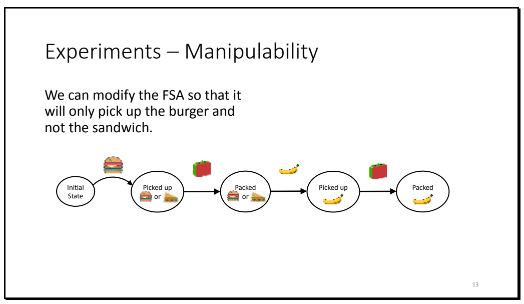


Slide 12

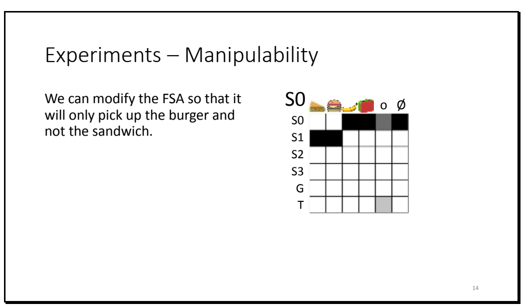



Since we have learned an interpretable model of the rules, we can easily modify the rules to change the behavior of the agent. In terms of the FSA, this means just deleting this edge between the initial state and the next state.

Slide 13




Slide 14

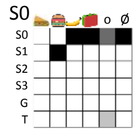



This is also easy to express using the transition matrix of the FSA; we can change the values in the matrix to change the form of the FSA.

Slide 15

## Experiments – Manipulability

We can modify the FSA so that it will only pick up the burger and not the sandwich.

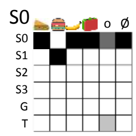





Slide 16

## Experiments – Manipulability

We can modify the FSA so that it will only pick up the burger and not the sandwich.






Slide 17

# Learning to Plan with Logical Automata

Brandon Araki[1]*, Kiran Vodrahalli[2]*, Thomas Leech[1,3], Mark Donahue[3], Cristian-Ioan Vasile[1], Daniela Rus[1]
[1]Massachusetts Institute of Technology
[2]Columbia University
[3]MIT Lincoln Laboratory
*Equal contributors

MIT CSAIL CS@CU