The Logical Options Framework

Brandon Araki[1], Xiao Li[1], Kiran Vodrahalli[2], Jonathan DeCastro[3], Daniela Rus[1]

[1]CSAIL MIT

[2]Columbia University

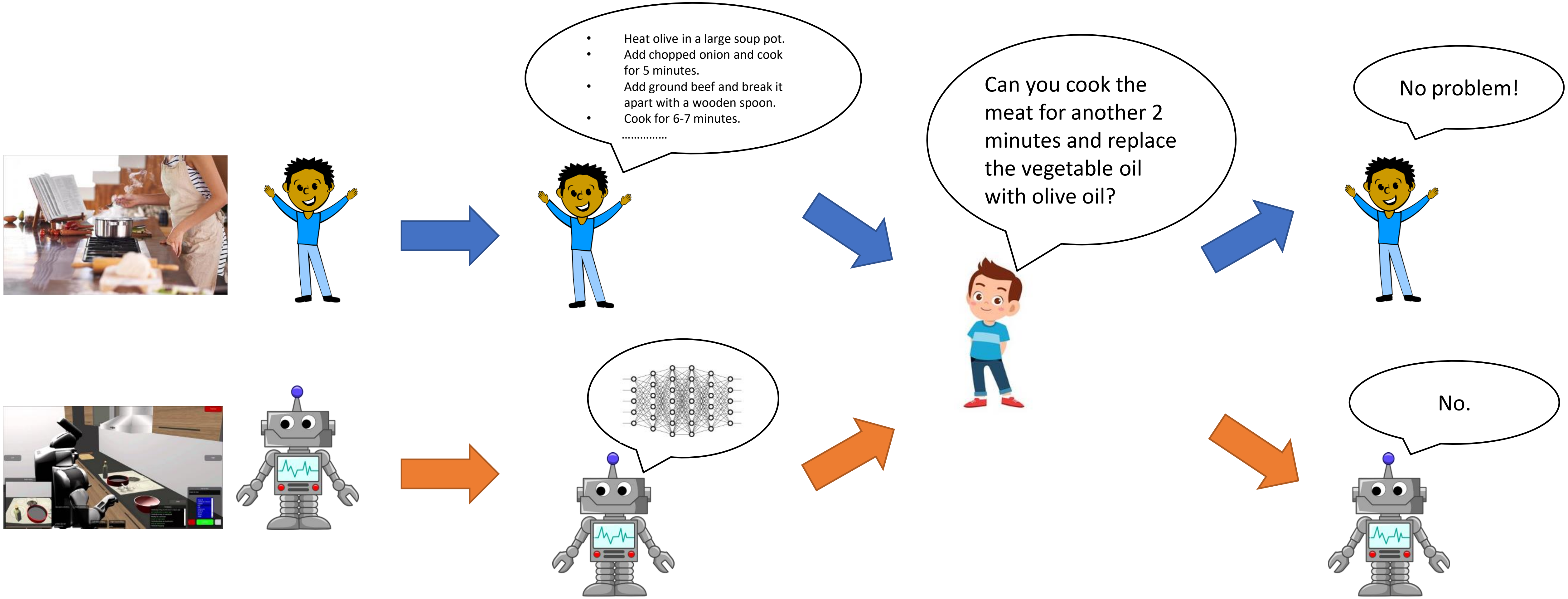
[3]Toyota Research Institute

Deep RL vs. Human Intelligence

Given a task and environment….

What are you doing?

Can you modify it?



- Heat olive in a large soup pot.
- Add chopped onion and cook for 5 minutes.
- Add ground beef and break it apart with a wooden spoon.
- Cook for 6-7 minutes.

……………

Can you cook the meat for another 2 minutes and replace the vegetable oil with olive oil?

No problem!

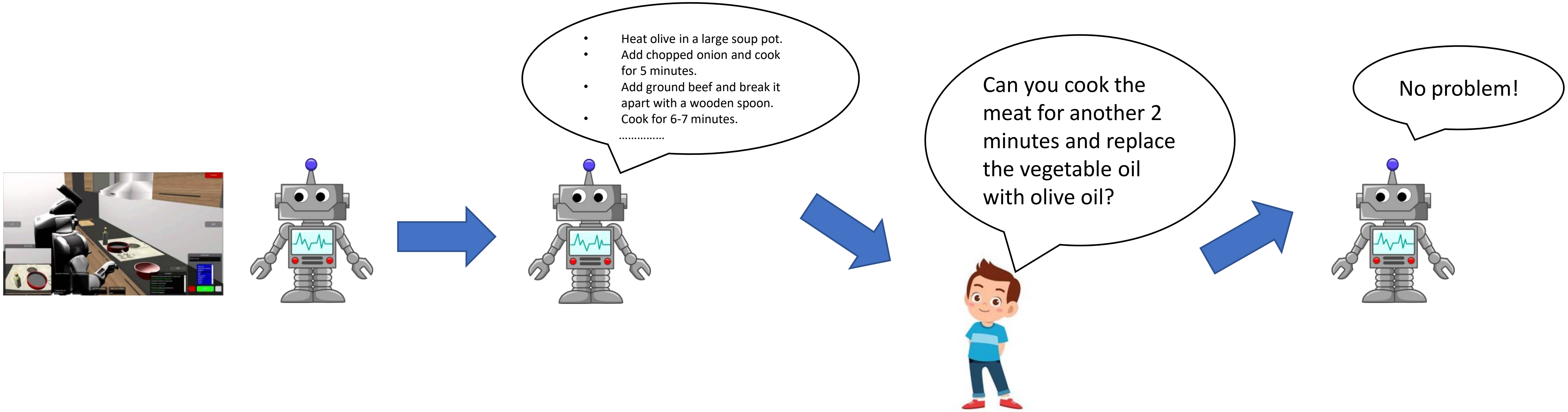
No.

Goals

Given a task and environment….

What are you doing?

Can You Modify It?



- Heat olive in a large soup pot.
- Add chopped onion and cook for 5 minutes.
- Add ground beef and break it apart with a wooden spoon.
- Cook for 6-7 minutes.

…………..

Can you cook the meat for another 2 minutes and replace the vegetable oil with olive oil?

No problem!

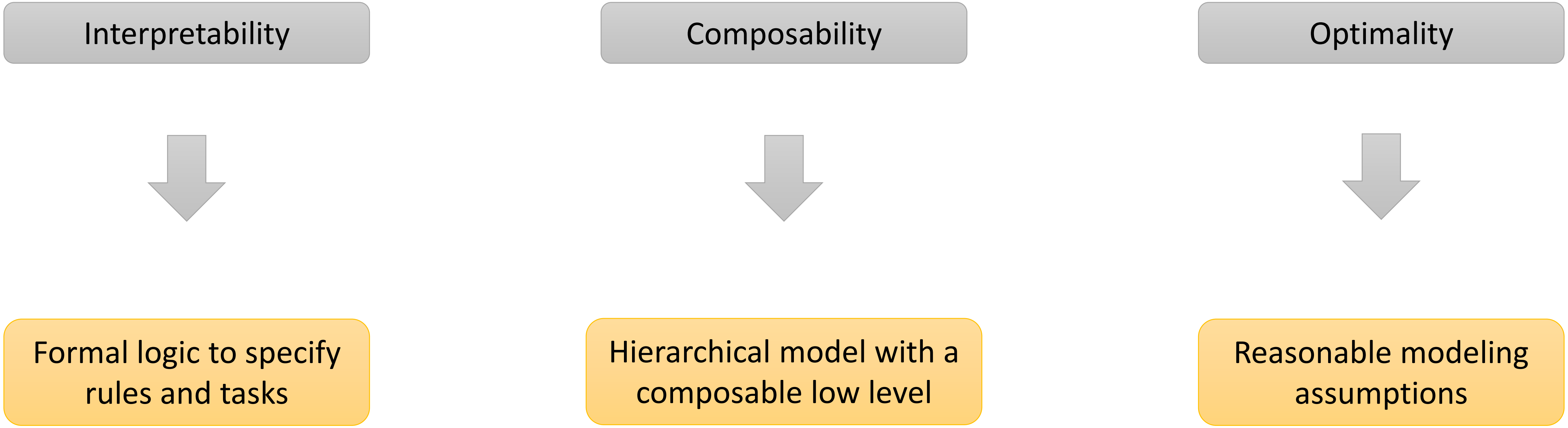
Make plans that are…

Interpretable

Composable

(and **optimal**!)

The Logical Options Framework



Interpretability

Formal logic to specify rules and tasks

Composability

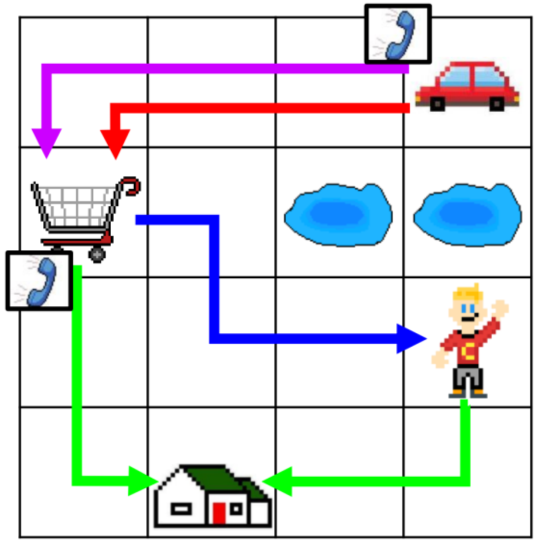
Hierarchical model with a composable low level

Optimality

Reasonable modeling assumptions

Working Example

"Go **grocery shopping**, pick up the **kid**, and go **home**, unless your partner **calls** telling you that they will pick up the kid, in which case just go **grocery shopping** and then go **home**. And don't drive into the **lake**."

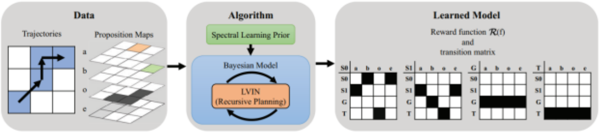


(F ☎ & F(🛒 & F 🏠)) |
(G ! ☎ & F(🛒 & F(🧒 & F 🏠)))
& G ! 💧

Related Work

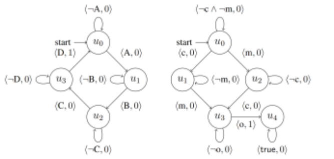
Not Composable

Probabilistic Automata



Araki et al., Deep Bayesian Nonparametric Learning of Rules and Plans from Demonstrations with a Learned Automaton Prior. *AAAI 2020*.

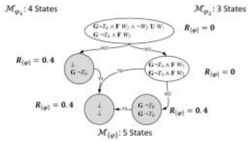
Reward Machines



(a) Patrol A, B, C, and D (b) Deliver a coffee and the mail

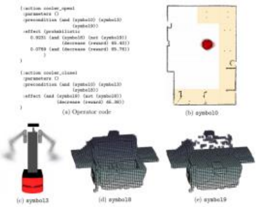
Toro Icarte et al., Using Reward Machines for High-Level Task Specification and Decomposition in Reinforcement Learning. *ICML 2018*.

LTL Formulas


$\mathcal{M}_{\varphi_1}$: 4 States
$\mathcal{M}_{\varphi_2}$: 3 States
$\mathcal{M}_{(\varphi)}$: 5 States

Ankit Shah et al., Planning with uncertain specifications (PUnS). *R-AL 2020*.

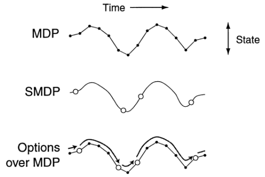
PDDL Operators



George Konidaris et al., From skills to symbols: Learning symbolic representations for abstract high-level planning. *JAIR 2018*.

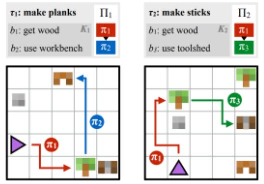
Not Satisfying

The Options Framework



Richard Sutton et al., Between mdps and semi-mdps: Learning, planning, and representing knowledge at multiple temporal scales. *JAIR 1998*.

Policy Sketches



Jacob Andreas et al., Modular Multitask Reinforcement Learning with Policy Sketches. *ICML 2017*.

MAXQ

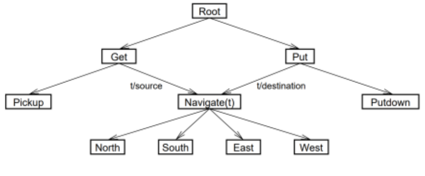
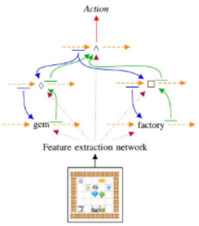


Figure 2: A task graph for the Taxi problem.

Thomas Dietterich et al., The MAXQ Method for Hierarchical Reinforcement Learning. *ICML 1998*.

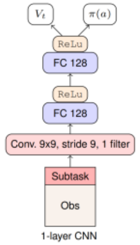
Not Optimal

Composing LTL Operators



Yen-Ling Kuo et al., Encoding formulas as deep networks: Reinforcement learning for zero-shot execution of LTL formulas. *IROS 2020.*

Neuro-Symbolic Planning



Borja Leon et al., Systematic Generalisation through Task Temporal Logic and Deep Reinforcement Learning. *arXiv 2020.*

The Logical Options Framework

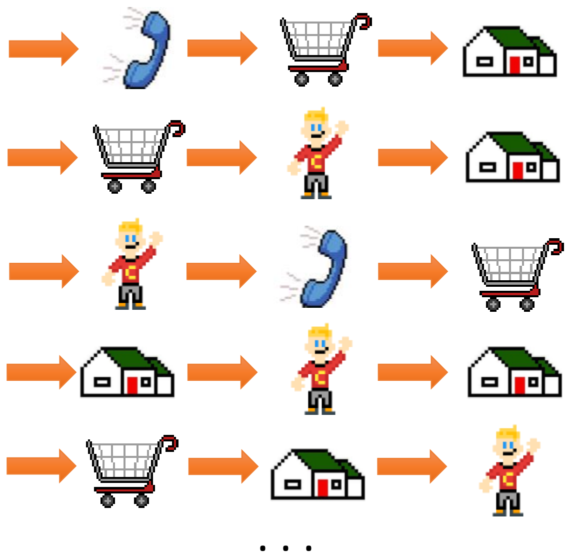
Interpretability

Formal logic to specify rules and tasks

(F ☎ & F(🛒& F 🏠)) |
(G ! ☎ & F(🛒& F(🧍 & F 🏠)))
& G !💧

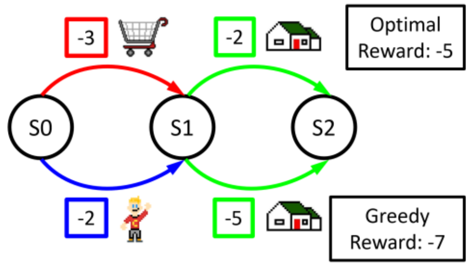
Composability

Hierarchical model with a composable low level

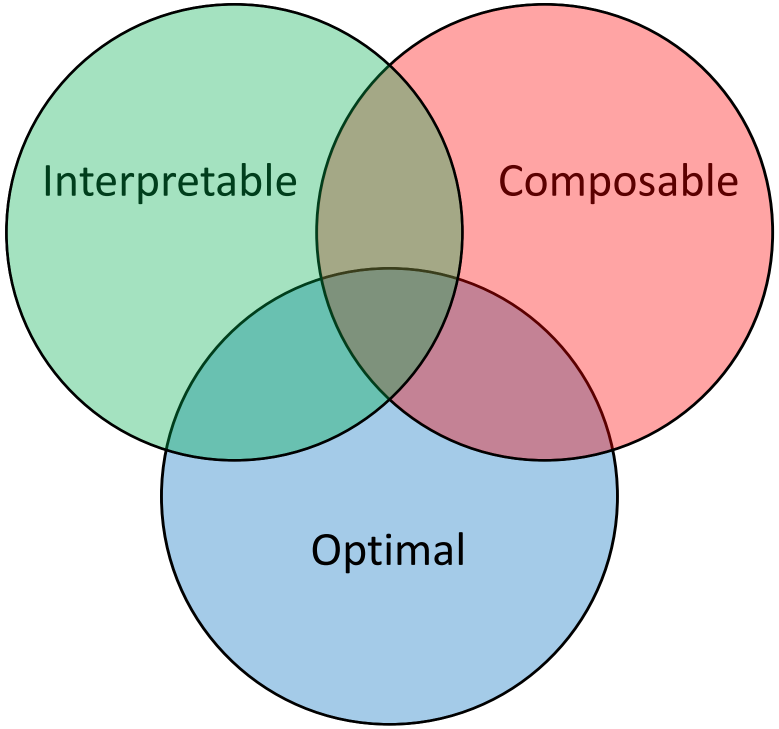


Optimality

Reasonable modeling assumptions



How to Unify these Three Goals?



1. Model the high-level as an automaton derived from an LTL formula
2. Model the environment as a composable semi-MDP
3. Place reasonable restrictions on the model and solve using value iteration

Formal logic to specify rules and tasks

Hierarchical models with a composable low level

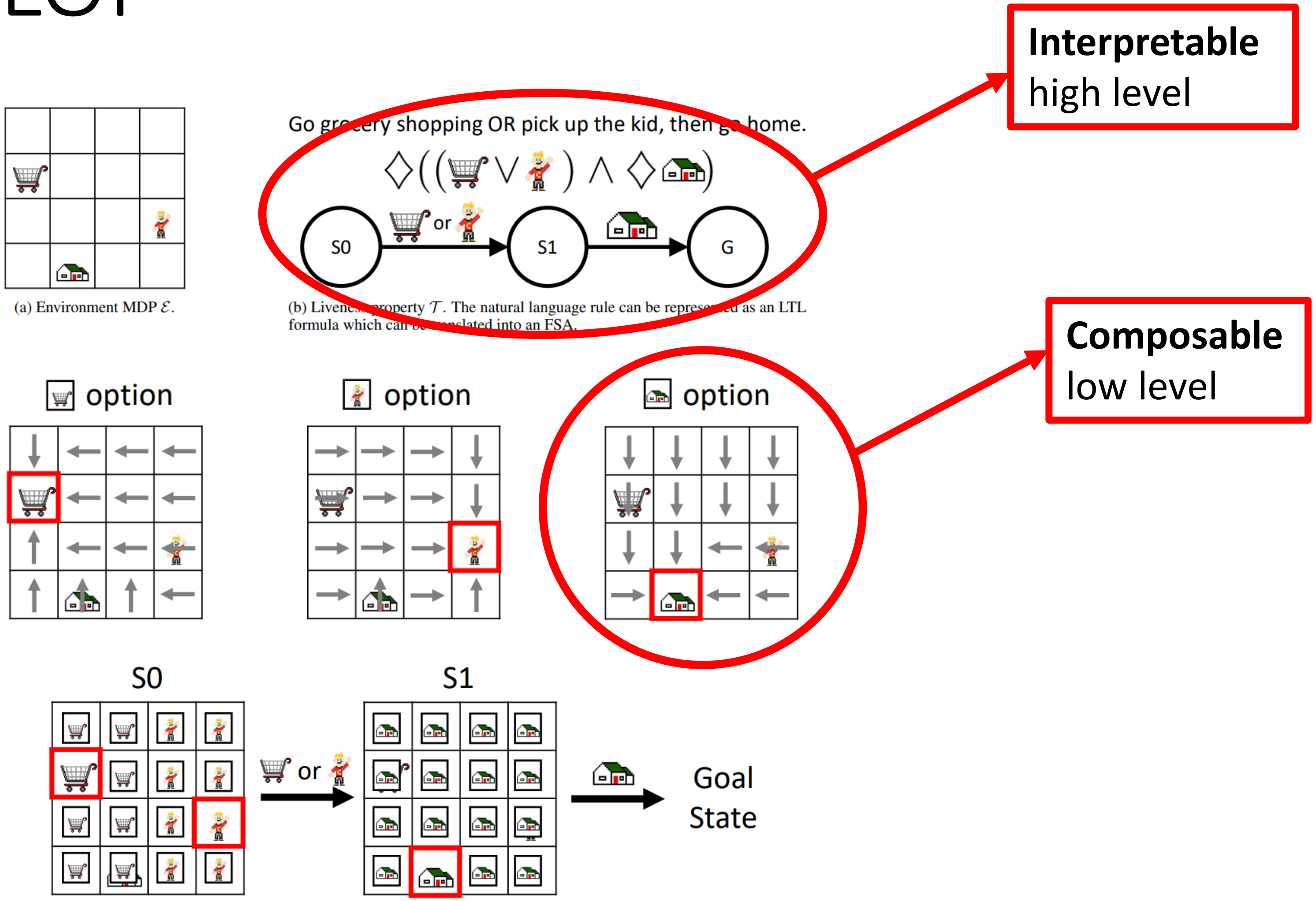
LVI and assumptions for optimality

Overview of LOF

Step 0: Define the SMDP

Step 1: Learn an option for each subgoal

Step 2: Make a meta-policy



(a) Environment MDP ℰ.

(b) Liveness property 𝒯. The natural language rule can be represented as an LTL formula which can be translated into an FSA.

Interpretable high level

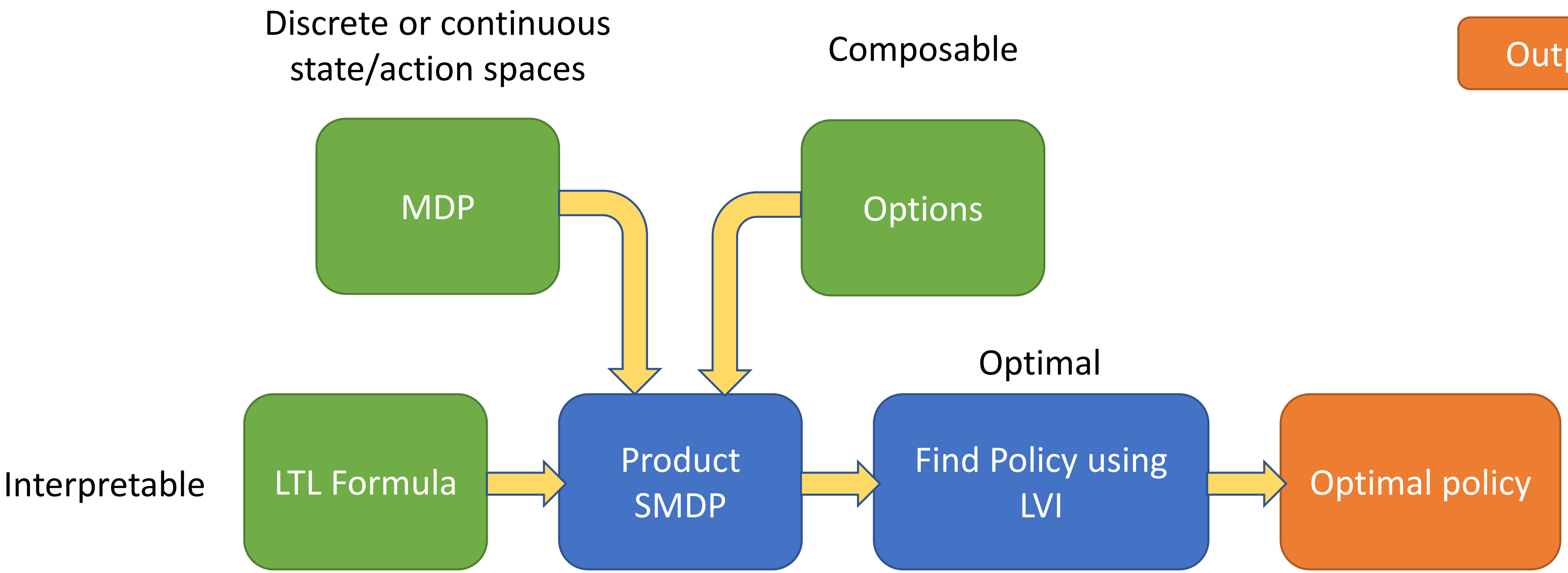
Composable low level

Overview of LOF

🟩 Inputs

🟦 Algorithm

🟧 Output



Linear Temporal Logic

- Set of atomic propositions Π

- Syntax: $\phi ::= p \mid \neg p \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid F\phi \mid X\phi \mid G\phi \mid \phi_1 \mathcal{U}\, \phi_2$

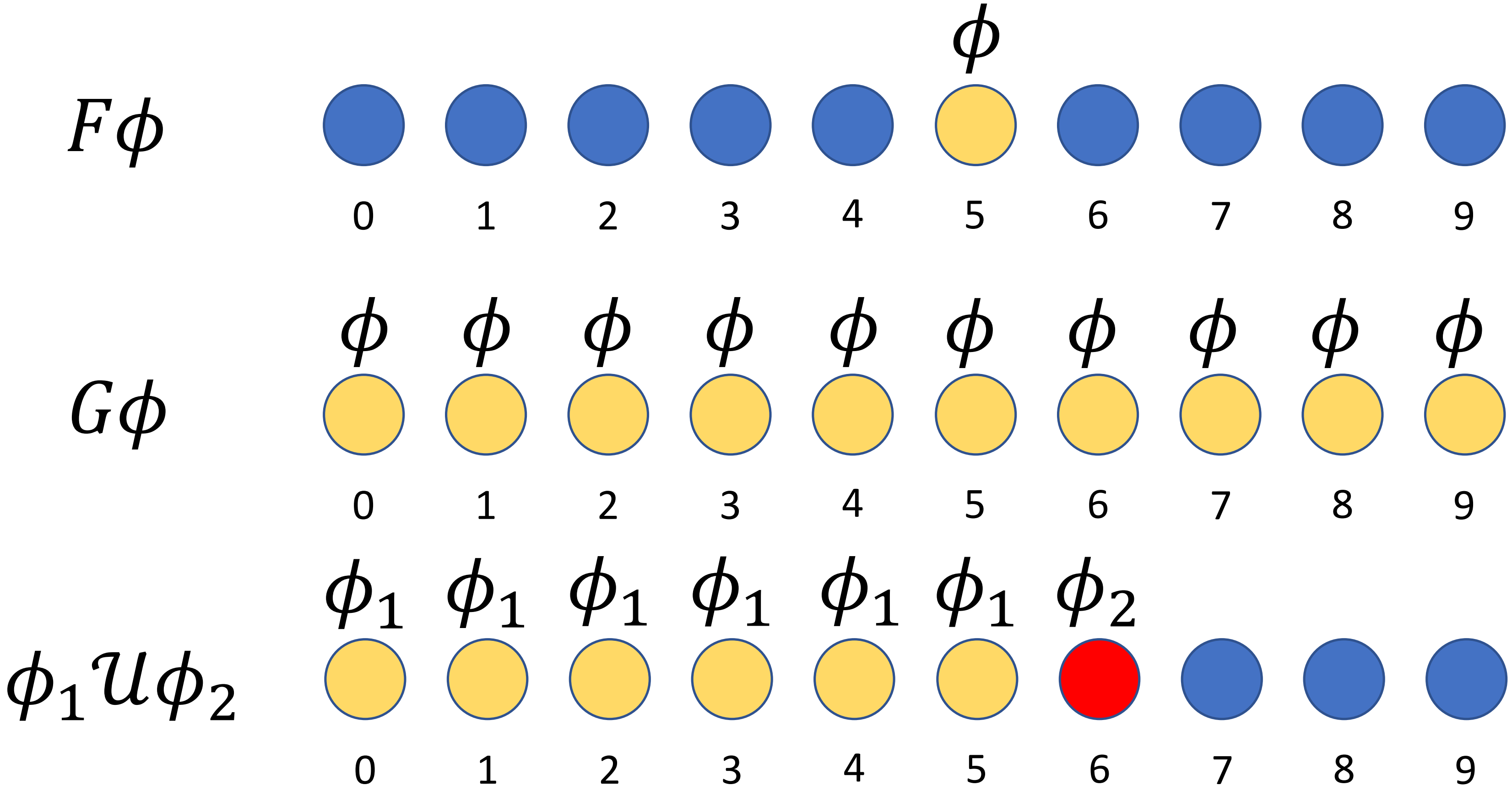
- Semantics interpreted infinite words over 2^{Π}
- Boolean operators: $\neg$ (negation), $\wedge$ (conjunction), $\vee$ (disjunction)
- Temporal operators: F (eventually), X (next), G(always), $\mathcal{U}$ (until)

Formal logic to specify rules and tasks

Hierarchical models with a composable low level

LVI and assumptions for optimality

Temporal operators


$F\phi$
ϕ
0 1 2 3 4 5 6 7 8 9
$G\phi$
ϕ ϕ ϕ ϕ ϕ ϕ ϕ ϕ ϕ ϕ
0 1 2 3 4 5 6 7 8 9
$\phi_1 \mathcal{U} \phi_2$
ϕ_1 ϕ_1 ϕ_1 ϕ_1 ϕ_1 ϕ_1 ϕ_2
0 1 2 3 4 5 6 7 8 9

Formal logic to specify rules and tasks

Hierarchical models with a composable low level

LVI and assumptions for optimality

Representing a Task

“Go **grocery shopping**, pick up the **kid**, and go **home**, unless your partner **calls** telling you that they will pick up the kid, in which case just go **grocery shopping** and then go **home**. And don’t drive into the **lake**.”

(F ☎ & F(🛒 & F 🏠)) |
(G ! ☎ & F(🛒 & F(🧒 & F 🏠)))
& G ! 💧

Formal logic to specify rules and tasks

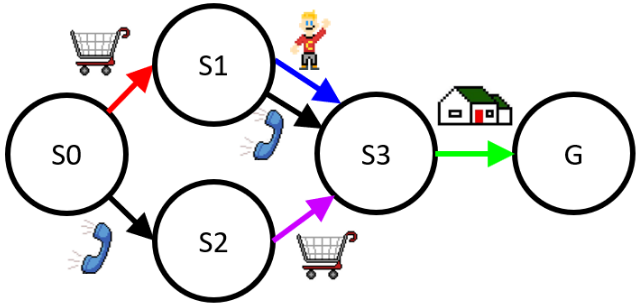
Hierarchical models with a composable low level

LVI and assumptions for optimality

LTL to Automata

- All LTL formulas can be converted to Buchi automata

(F ☎ & F(🛒& F 🏠)) |
(G ! ☎ & F(🛒& F(🧍 & F 🏠)))
& G ! 💧



Formal logic to specify rules and tasks

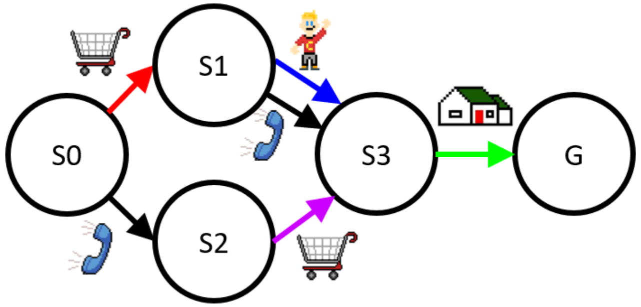
Hierarchical models with a composable low level

LVI and assumptions for optimality

Liveness and Safety Properties

- All Buchi automata can be decomposed into **liveness** and **safety** properties
- Liveness property: tasks that the agent **must achieve**
- Safety property: things that the agent **must avoid**

(F ☎ & F(🛒 & F 🏠)) |
(G ! ☎ & F(🛒 & F(🧍 & F 🏠)))
& G ! 💧



Penalty on 💧

Formal logic to specify rules and tasks

Hierarchical models with a composable low level

LVI and assumptions for optimality

Propositions

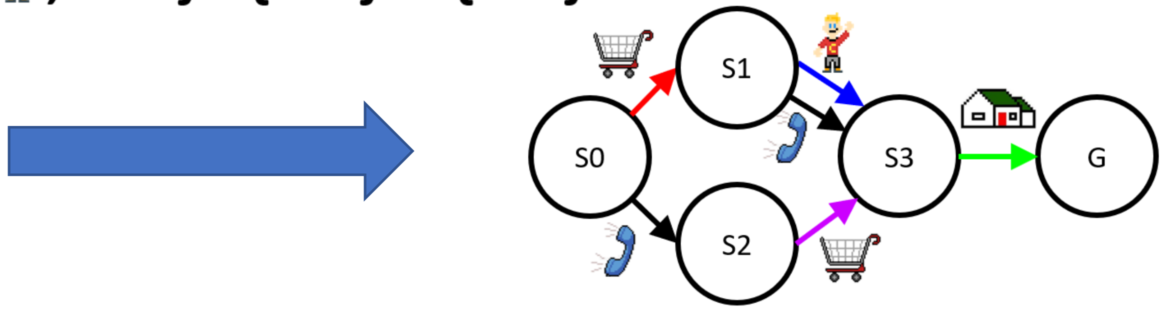
- Three types of propositions – **subgoal**, **event**, and **safety** propositions
- Every subgoal is associated with an ***option***

Subgoal propositions {🛒, 🧍, 🏠}

Safety proposition {💧}

Event proposition { ☎ }

(F ☎ & F(🛒 & F 🏠)) |
(G ! ☎ & F(🛒 & F(🧍 & F 🏠)))
& G ! 💧



Penalty on 💧

Formal logic to specify rules and tasks

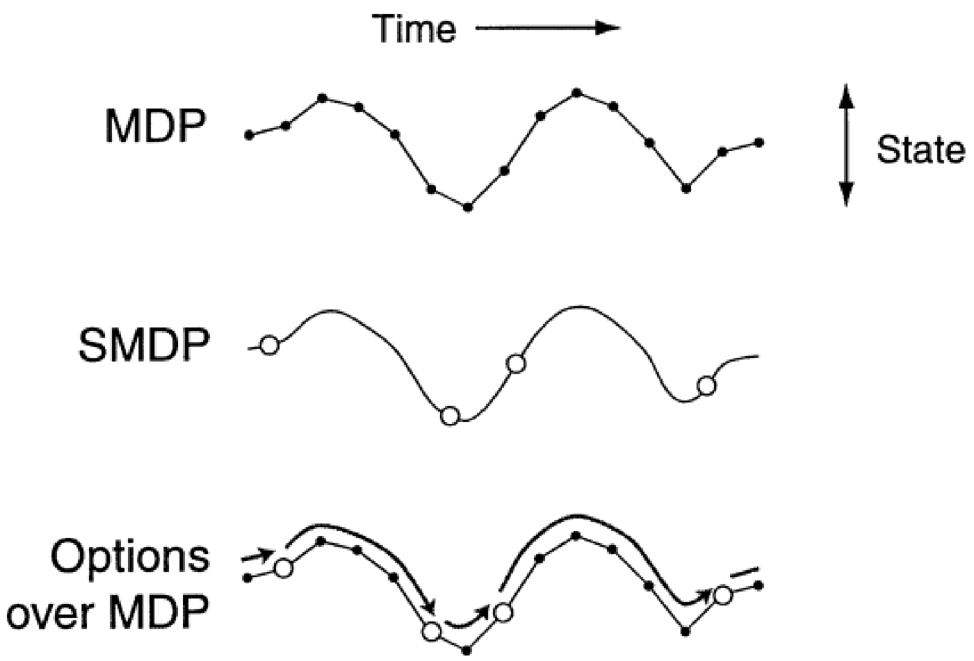
Hierarchical models with a composable low level

LVI and assumptions for optimality

MDPs vs. Semi-MDPs

- Current state depends on previous state/action
- Actions take variable amounts of time
- High-level actions called **options** take variable amounts of time. The current state/action depends on the identity of the option, which may have been chosen multiple time steps ago.

MDPs vs. Semi-MDPs



- The Options Framework extends MDP planning to SMDP planning
  - Introduces hierarchical action space with high-level actions called **options**
  - Options can be trained on **continuous** state/action spaces
  - Options can be **composed** arbitrarily

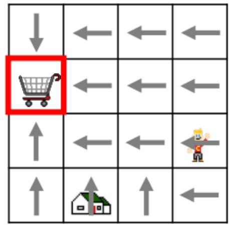
Formal logic to specify rules and tasks

Hierarchical models with a composable low level

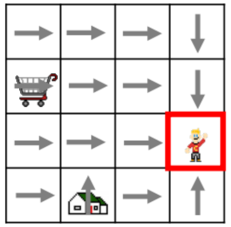
LVI and assumptions for optimality

Logical Options

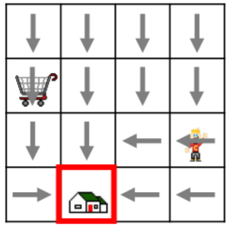
🛒 option



🧍 option



🏠 option



For every p in $\mathcal{P}_G$, learn an option for achieving p, $o_p = (\mathcal{I}_{o_p}, \pi_{o_p}, \beta_{o_p}, R_{o_p}(s), T_{o_p}(s'|s))$

Initiation set

$$\mathcal{I}_{o_p} = \mathcal{S}$$

Termination condition

$$\beta_{o_p} = \begin{cases} 1 & \text{if } T_P(s,p) = 1 \\ 0 & \text{otherwise} \end{cases}$$

Sub-policy

$$\pi_{o_p} = \text{optimal policy on } \mathcal{E} \text{ with rollouts terminating when } T_P(s) = p$$

Transition model

$$T_{o_p}(s'|s) = \begin{cases} \gamma^k & \text{if } T_P(s') = p\text{, where } k \text{ is number of time steps to reach } p \\ 0 & \text{otherwise} \end{cases}$$

Reward model

$$R_{o_p}(s) = \mathbb{E}[\mathcal{R}_\mathcal{E}(s,a) + \gamma\mathcal{R}_\mathcal{E}(s',a') + \gamma^2\mathcal{R}_\mathcal{E}(s'',a'') + \ldots]$$

Formal logic to specify rules and tasks

Hierarchical models with a composable low level

LVI and assumptions for optimality

Transition and Reward Models

- Reward model is equivalent to a value function

$$R_o(s) = \mathbb{E}\left[r_{t+1} + \gamma r_{t+2} + \ldots \gamma^{k-1} r_{t+k}\right]$$

Note: Safety propositions must be assigned costs and incorporated into the reward function of the environment when learning the policy and value function

- Transition model can be simplified by setting gamma=1 and by assuming the option always reaches its subgoal

$$T_o(s'|s) = \sum_{k=1}^{\infty} p(s', k)\gamma^k$$

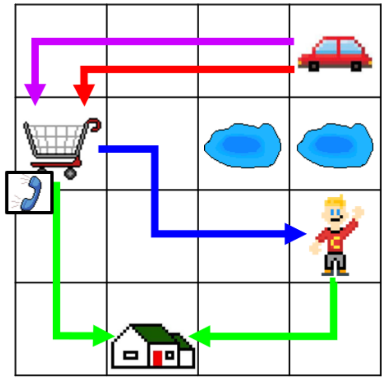
Formal logic to specify rules and tasks

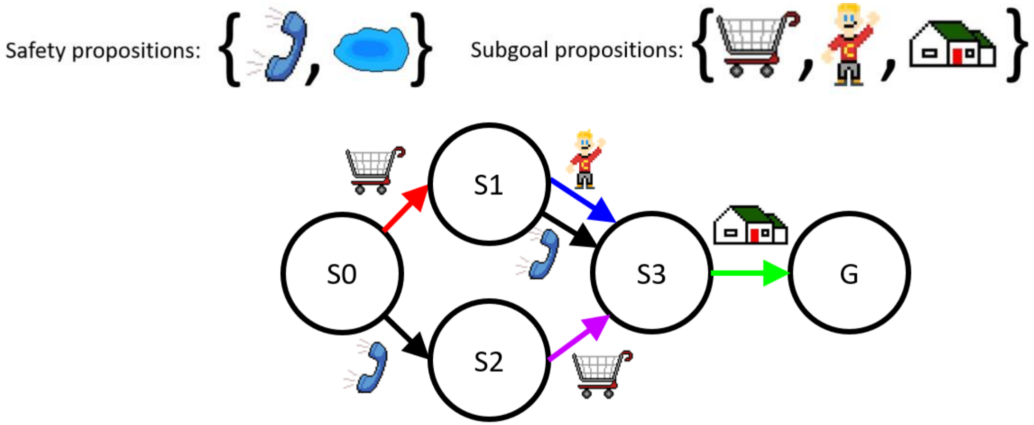
Hierarchical models with a composable low level

LVI and assumptions for optimality

Review: How LTL Fits into the Picture

- Three types of propositions – **subgoals, event** and **safety** propositions
- Specification divided into **liveness** and **safety** properties
- Associate every subgoal with an **option**
- Find highest-reward path through the liveness FSA



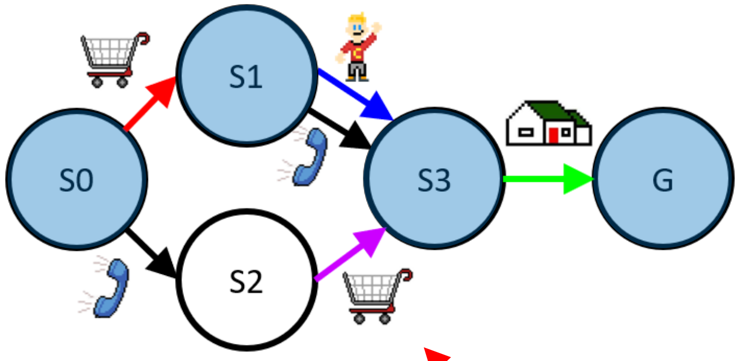


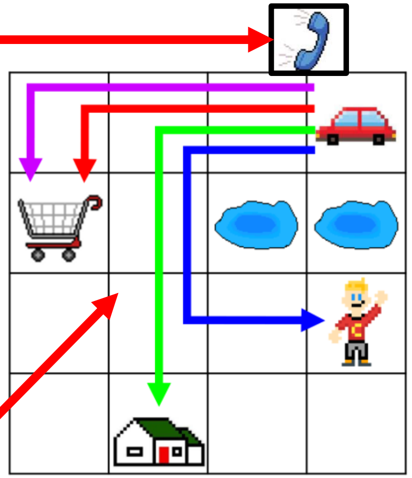
Formal logic to specify rules and tasks

Hierarchical models with a composable low level

LVI and assumptions for optimality

Logical Value Iteration





$$Q_k(f,s,o) \leftarrow R_F(f)R_o(s) + \sum_{f'\in\mathcal{F}} \sum_{\bar{p}_e\in 2^{\mathcal{P}_E}} \sum_{s'\in\mathcal{S}} \boxed{T_F(f'|f,T_{P_G}(s'),\bar{p}_e} \boxed{T_{P_E}(\bar{p}_e} \boxed{T_o(s'|s)} V_{k-1}(f',s') \quad (3)$$

$$V_k(f,s) \leftarrow \max_{o\in\mathcal{O}} Q_k(f,s,o) \quad (4)$$

Formal logic to specify rules and tasks

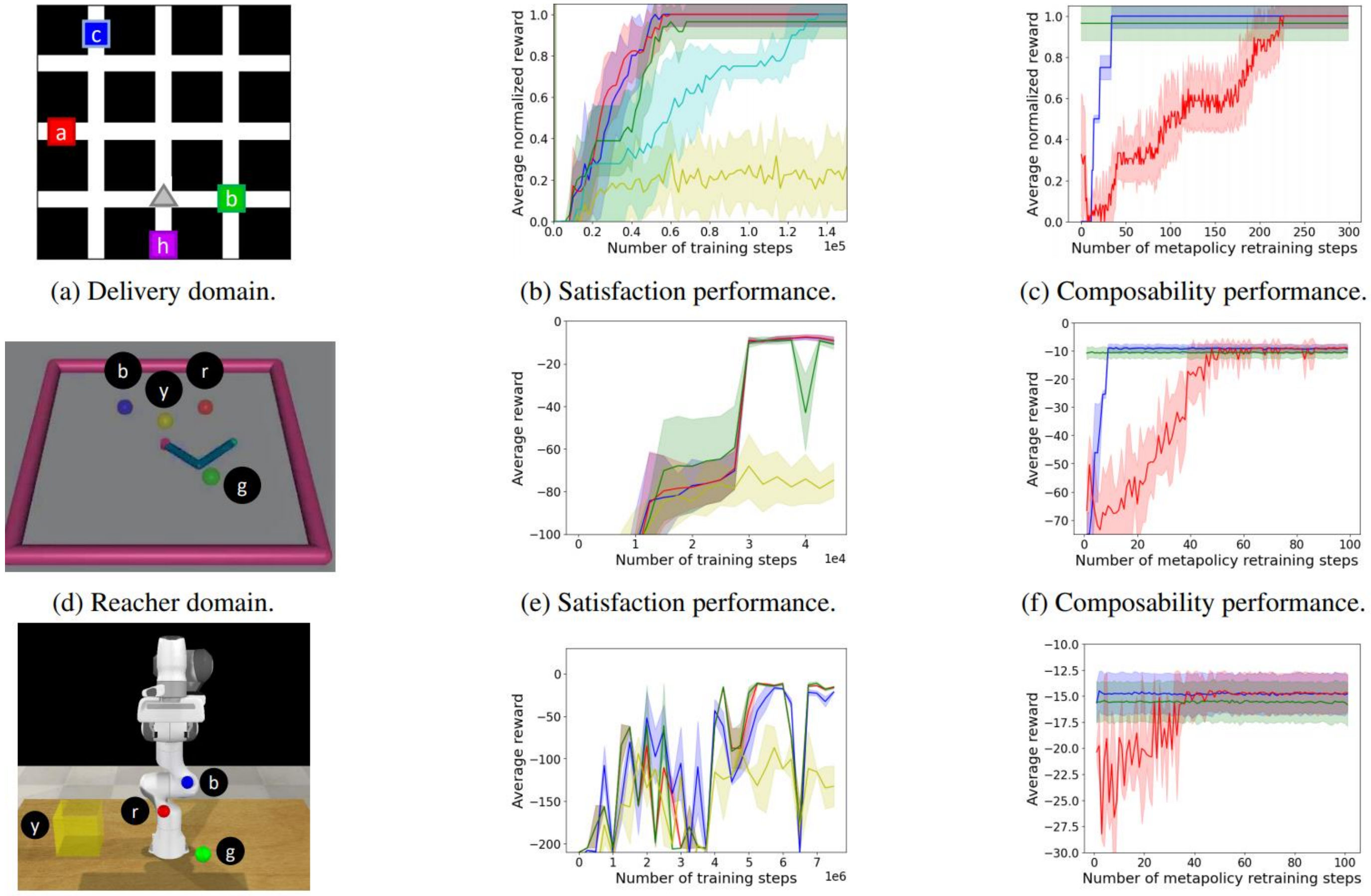
Hierarchical models with a composable low level

LVI and assumptions for optimality

Assumptions for Optimality

- Every subgoal is associated with a *single state*
- Every option can reach its associated subgoal from any other state in the environment
- The goal state of the automaton is reachable from every other automaton state via subgoals

Experiments



(a) Delivery domain.

(b) Satisfaction performance.

(c) Composability performance.

(d) Reacher domain.

(e) Satisfaction performance.

(f) Composability performance.

(g) Pick-and-place domain.

(h) Satisfaction performance.

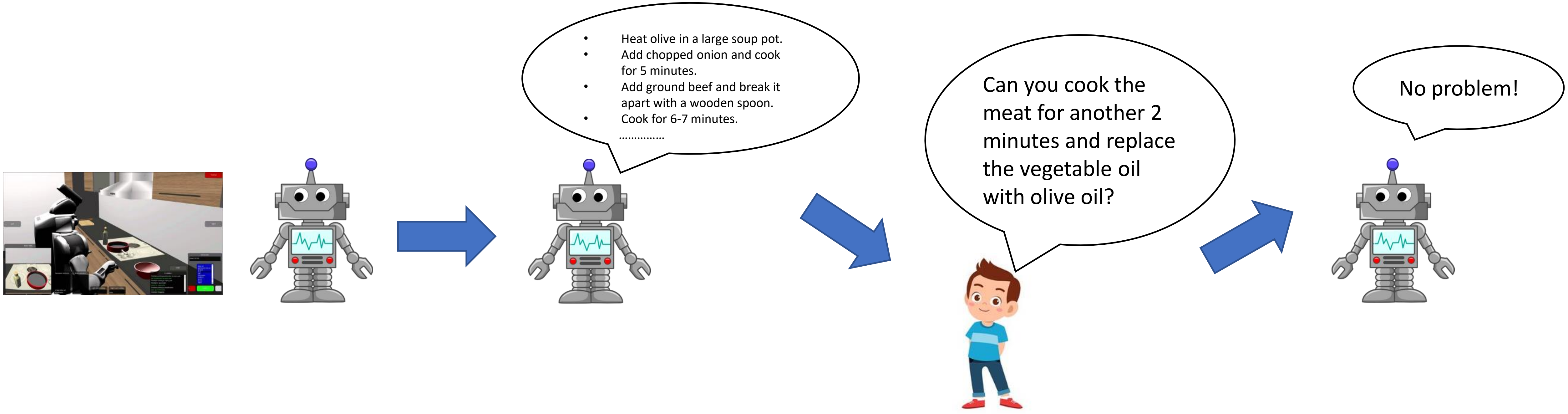
(i) Composability performance.

Conclusion

Given a task and environment….

What are you doing?

Can You Modify It?



- Heat olive in a large soup pot.
- Add chopped onion and cook for 5 minutes.
- Add ground beef and break it apart with a wooden spoon.
- Cook for 6-7 minutes.

……………

Can you cook the meat for another 2 minutes and replace the vegetable oil with olive oil?

No problem!

Make plans that are…

Interpretable

Composable

(and **optimal**!)

Conclusion

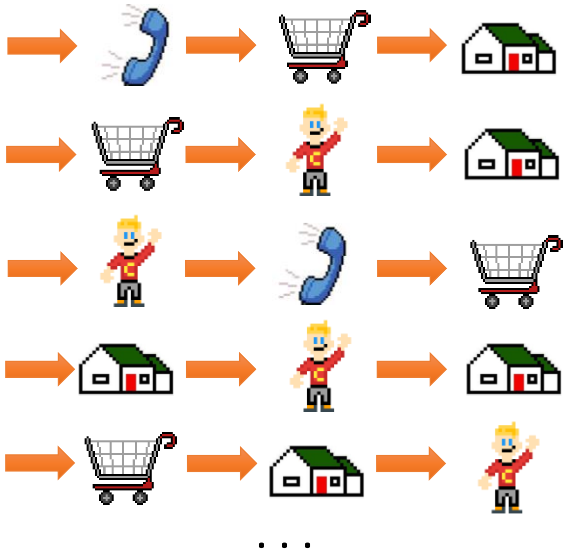
Interpretability

Formal logic to specify rules and tasks

(F ☎ & F(🛒& F 🏠)) |
(G ! ☎ & F(🛒& F(🧍 & F 🏠)))
& G ! 💧

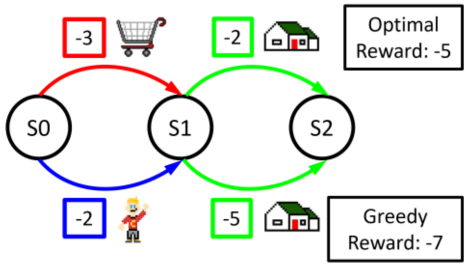
Composability

Hierarchical model with a composable low level



Optimality

Reasonable modeling assumptions


-3
-2
Optimal Reward: -5
S0
S1
S2
-2
-5
Greedy Reward: -7