The Logical Options Framework

Brandon Araki\textsuperscript{1}, Xiao Li\textsuperscript{1}, Kiran Vodrahalli\textsuperscript{2}, Jonathan DeCastro\textsuperscript{3}, Daniela Rus\textsuperscript{1}

\textsuperscript{1}CSAIL MIT
\textsuperscript{2}Columbia University
\textsuperscript{3}Toyota Research Institute
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?

• 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?

Can You Modify It?

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
"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."

\[(F \land F(\text{Grocery Shopping} \land F \text{Pick up Kid}))) \mid (G ! \land F(\text{Grocery Shopping} \land F(\text{Pick up Kid} \land F \text{Drive into Lake}))) \land G !\]
Related Work

Not Composable

Probabilistic Automata


Reward Machines


LTL Formulas


PDPL Operators

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

Not Satisfying

The Options Framework


Policy Sketches


Not Optimal

Composing LTL Operators


Neuro-Symbolic Planning

The Logical Options Framework

Interpretability
Formal logic to specify rules and tasks

(F & F( & F )) | (G ! & 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

Interpretable high level

Composable low level
Overview of LOF

Discrete or continuous state/action spaces

Interpretable

LTL Formula

MDP

Options

Composable

Optimal

Product SMDP

Find Policy using LVI

Optimal policy

Inputs

Algorithm

Output
Linear Temporal Logic

• Set of atomic propositions $\Pi$

• Syntax: $\phi ::= p | \neg p | \phi_1 \land \phi_2 | \phi_1 \lor \phi_2 | F\phi | X\phi | G\phi | \phi_1 \mathcal{U} \phi_2$

• Semantics interpreted infinite words over $2^\Pi$

• Boolean operators: $\neg$ (negation), $\land$ (conjunction), $\lor$ (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

\[ F\phi \]

\[ G\phi \]

\[ \phi_1 \mathcal{U} \phi_2 \]

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**.”

\[(F \& F(\& F(\& F(\& G))) | (G ! \& F(\& F(\& F(\& F(\& G))))) \& G !)\]
LTL to Automata

• All LTL formulas can be converted to Buchi automata

\[(F & F(\text{phone} & F(\text{bottle}))) \mid (G \neg & F(\text{bottle} & F(\text{car} & F(\text{house})))) & G \neg)\]
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**

\[(F \land F(\mathcal{W} \land F(\mathcal{H}))) \lor (G \land G(\mathcal{W} \land F(\mathcal{H})))\land G !\]

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**

\[
(F \land G) \lor (G \land F)
\]

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.

Formal logic to specify rules and tasks
Hierarchical models with a composable low level
LVI and assumptions for optimality
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

For every $p$ in $\mathcal{P}_G$, learn an option for achieving $p$, $o_p = (I_o, \pi_o, \beta_o, R_o(s), T_o(s'|s))$

$$I_o = S$$

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

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

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

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

- **Initiation set**
- **Termination condition**
- **Sub-policy**
- **Transition model**
- **Reward model**

**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

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

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

Formal logic to specify rules and tasks

Hierarchical models with a composable low level

LVI and assumptions for optimality

\[
Q_k(f, s, o) \leftarrow R_f(f)R_o(s) + \sum_{f' \in F} \sum_{\bar{p}_c} \sum_{s' \in S} T_F(f'|f, T_{P_G}(s'), \bar{p}_c) T_{P_E}(\bar{p}_c) T_O(s'|s) V_{k-1}(f', s')
\]

\[
V_k(f, s) \leftarrow \max_{o \in O} Q_k(f, s, o)
\]
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

Formal logic to specify rules and tasks
Hierarchical models with a composable low level
LVI and assumptions for optimality
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?

- 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 Modify It?

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

Composability

Hierarchical model with a composable low level

Optimality

Reasonable modeling assumptions