Designing Behavior Trees from Goal-Oriented LTLf Formulas

Aadesh Neupane & Michael A. Goodrich
Computer Science Department
Brigham Young University

Goal Specification and Planning

Specify a Goal

Natural Language, State-machine, LTLf, Rewards, Objective Functions

Plan

RRT*, A*, Policy Iteration, MPC, PPO, Q-Learning

Measure Performance

F-Score, Precision, Recall, PAC, Resilience, Success Rate

Goal Specification and Planning

Specify a Goal:
Finite Trace LTL

Plan:
Behavior Tree

Measure Performance:
Successful BT ⇒ Satisfied Goal

LTL_f Overview

Finite Trace Linear Temporal Logic (LTL_f)

- Define state transition systems $S \times A \rightarrow S$
- Each state is a vector of propositions
- Define a *trace* is a sequence of states $< S_0, S_1, \ldots, S_T >$.

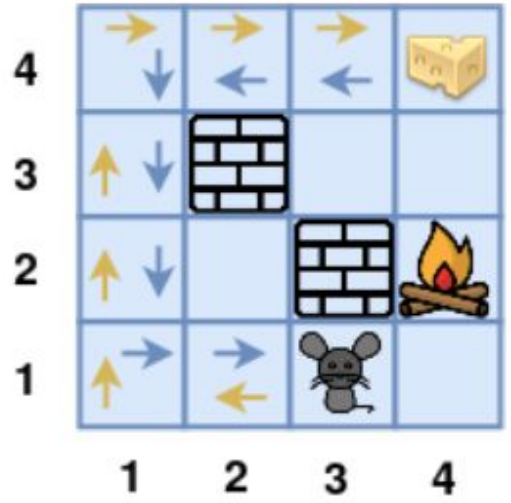
$$\sigma = [S_0 , S_1 , \ldots , S_n]$$

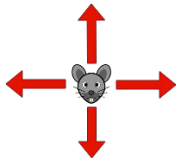
- Trace is **valid** only if LTL formula satisfied

Finite Trace Linear Temporal Logic (LTL_f)

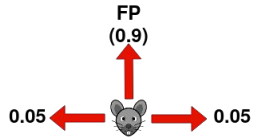
- Propositions:
  - **c** (cheese)
  - **f** (fire)
  - **h** (home)
- Transitions: Mouse movements in Markov process

- Start: S_0 = {'h': True, 'c': False, 'f': False}
- Goal: S_n = {'h': True, 'c': True, 'f': False}
- Constraint: $\forall i\ S_i$ = {'h': ??, 'c': ??, 'f': False}





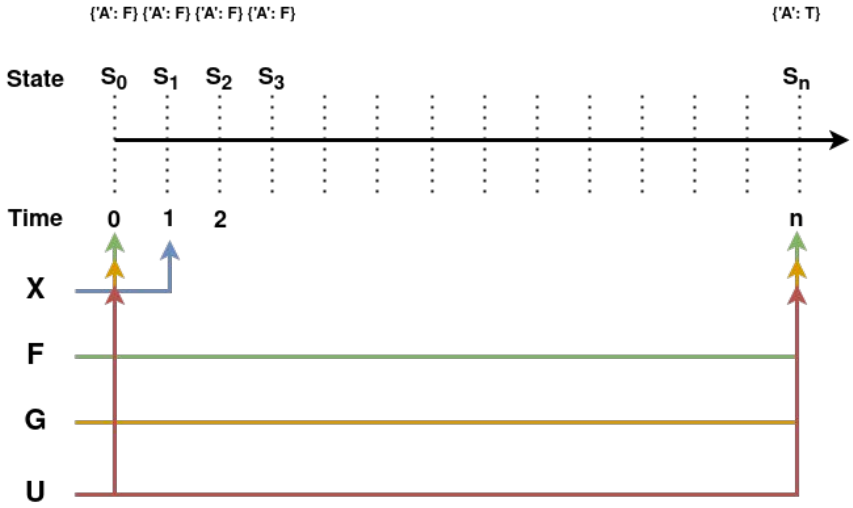
Agent Actions



Intended Action Probability

LTL_f Syntax and Semantics

- Given a trace σ of some finite length *m* and LTL_f goal specification φ
- φ= **A**, where A is a proposition variable that checks if the current state is True or False.

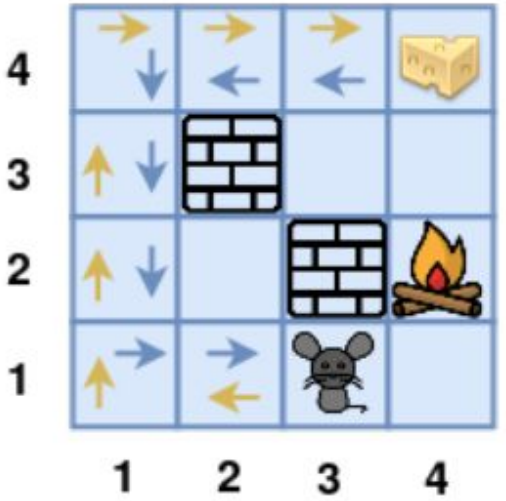


Textual	Explanation	Example
X φ	neXt: φ has to hold at the next state	σ = [{'A':F},{'A':F},{'A':F}]. X φ is False
F φ	Finally: φ eventually has to hold	σ = [{'A':F},{'A':T},{'A':F}]. F φ is True
G φ	Globally: φ has to hold on the entire trace	σ = [{'A':F},{'A':T},{'A':F}]. G φ is False
φ_1 U φ_2	Until: φ_1 has to hold at least until φ_2 becomes true, which must hold at current or future positions	Suppose $\varphi = \varphi_1 = \varphi_2$ σ = [{'A':F, 'B':F},{'A':F, 'B':T}]. φ_1 U φ_2 is False

Color Code : Trace Scope | Valid | Invalid

Finite Trace Linear Temporal Logic (LTL_f)

- Propositions:
  - **c** (cheese)
  - **f** (fire)
  - **h** (home)
- Transitions: Mouse movements in Markov process



- Start: S_0 = {'c': False, 'h': True, 'f': False}
- Goal: S_n = {'c': True, 'h': True, 'f': False}
- Constraint: $\forall i\ S_i$ = {'c': ??, 'h': ??, 'f': False}

$$\psi^{Cheese} = \wedge \mathbf{F}(\wedge Cheese\ Home)\mathbf{G}(\neg Fire)$$

Behavior Tree Overview

Behavior Tree

BT Node	BT Symbol	Behavior	Operator
Selector	?	Or	v
Sequence		And	∧
Condition		True/False	
Action		User Defined	

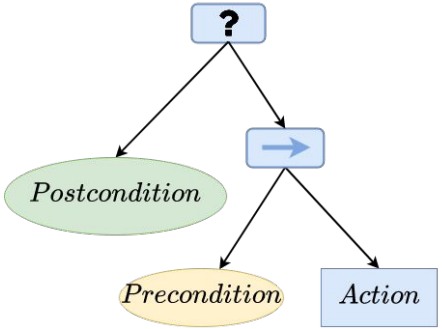
- BTs are modular, readable and reactive
- Formally equivalent to the Hierarchical State Machine and Or-And Trees.

Behavior Tree: PPA Form

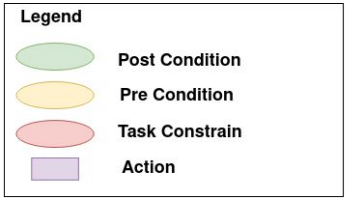
BT Node	BT Symbol	Behavior	Operator
Selector	?	Or	v
Sequence	→	And	∧
Condition	⬭	True/False	
Action	▭	User Defined	

Postcondition-Precondition-Action (PPA) BT

- Postcondition OR plan execution
- Ensures planner is not (re)executed if post condition is already met



a) Standard PPA BT

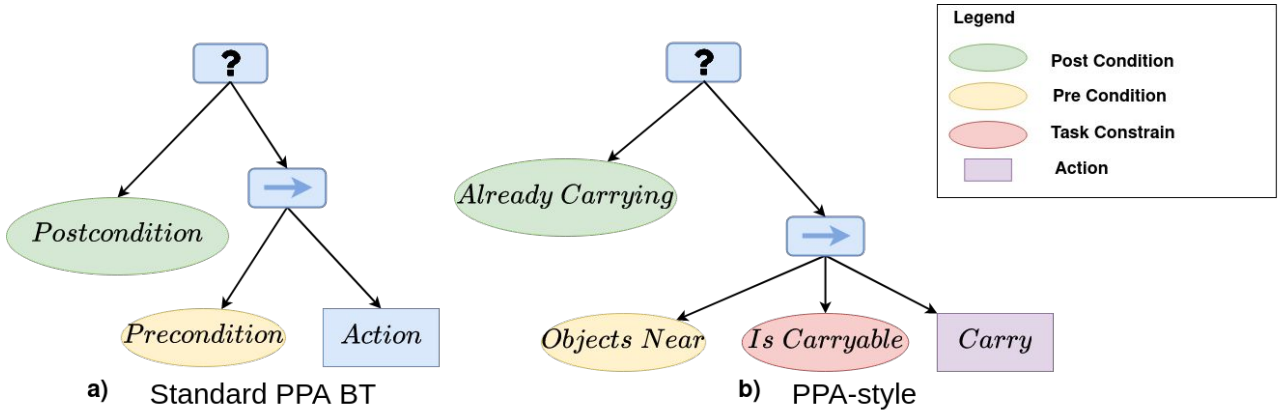


Behavior Tree:
PPA Form w/ constraint

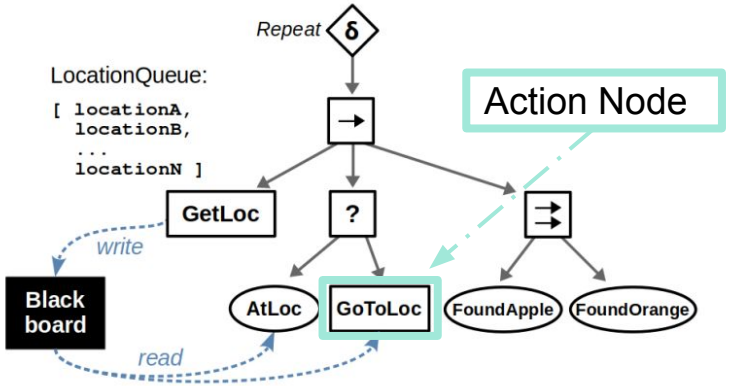
BT Node	BT Symbol	Behavior	Operator
Selector	?	Or	v
Sequence	→	And	∧
Condition	⬭	True/False	
Action	▭	User Defined	

Postcondition-Precondition-Action (PPA) BT

- Postcondition OR plan execution
- Ensures planner is not (re)executed if post condition is already met

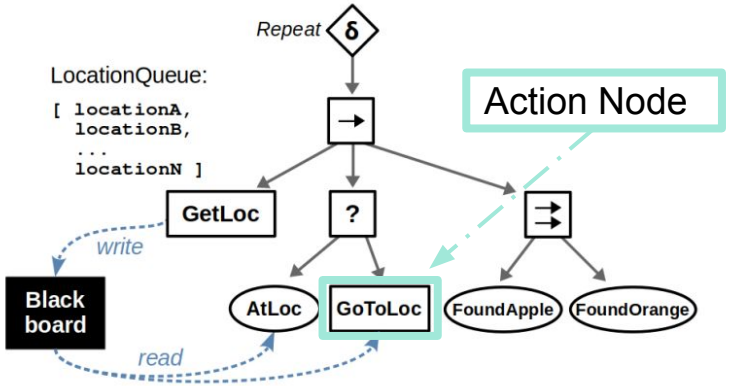


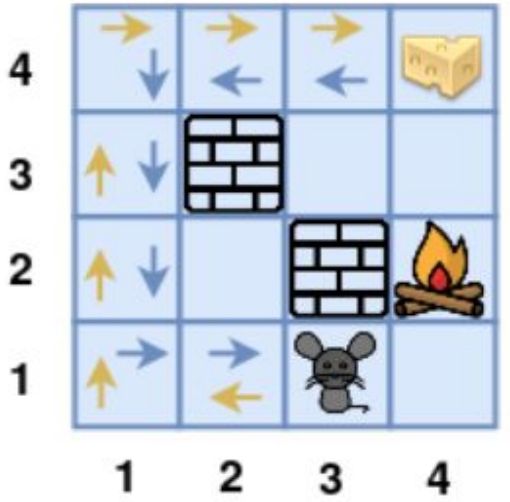
BT as Trace Generator



- Plans are stored in **action** nodes
- State changes when plans execute actions
- Trace is sequential accumulation of states

BT as Trace Generator





- Plans are stored in **action** nodes
- State changes when plans execute actions
- Trace is sequential accumulation of states

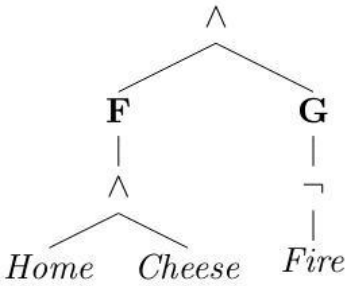
σ = [{‘c’: F, ‘h’:F, ‘f’:F}, {‘c’: F, ‘h’:F, ‘f’:F}, {‘c’: F, ‘h’:F, ‘f’:F},..., {‘c’: F, ‘h’:F, ‘f’:T}]

Convert Goal in LTL_f to a PPA-Style Behavior Tree …

Idea: LTL_f to BT

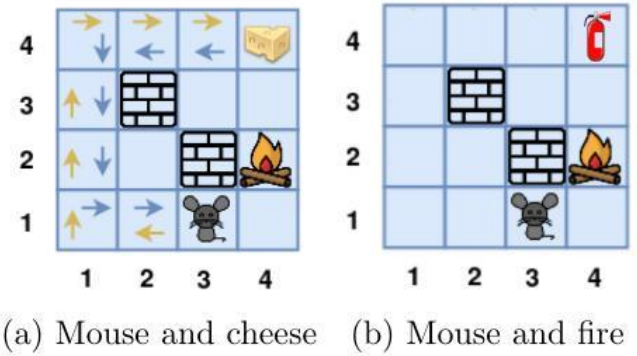
Mouse and Cheese

$$\psi^{Cheese} = \wedge \mathbf{F}(\wedge Cheese\ Home)\mathbf{G}(\neg Fire)$$



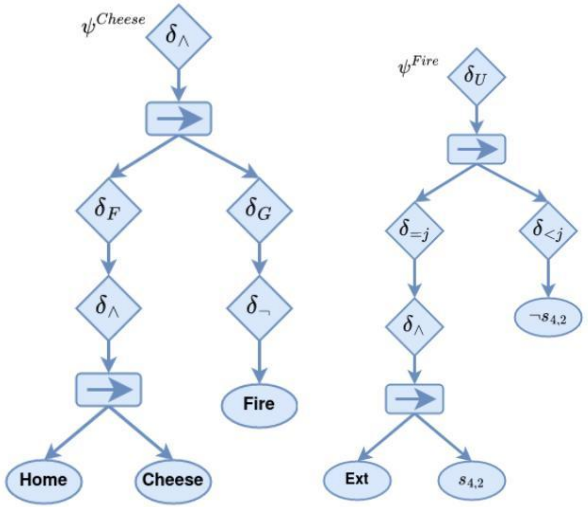
Mouse and Fire

$$\psi^{\text{Fire}} = \mathbf{U}\neg s_{4,2} \quad (\wedge \ Ext\ s_{4,2}).$$



(a) Mouse and cheese (b) Mouse and fire

Decomposition



(a) Mouse and Cheese BT (b) Mouse and Fire BT

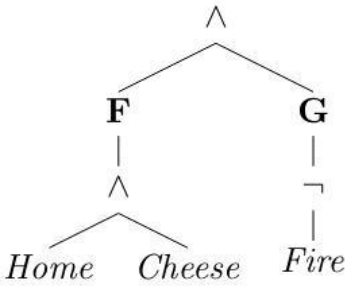
Existing LTLf to BT decomposition algorithm problems

- Coupled algorithm for verification and synthesis
- Unable to use existing planners

Idea: LTL_f to BT

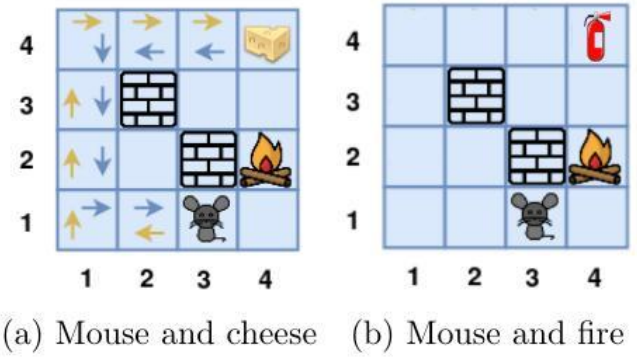
Mouse and Cheese

$$\psi^{Cheese} = \wedge \mathbf{F}(\wedge Cheese\ Home)\mathbf{G}(\neg Fire)$$



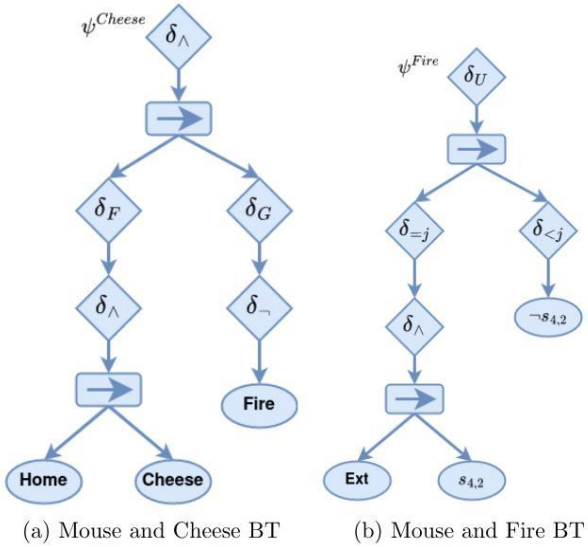
Mouse and Fire

$$\psi^{\text{Fire}} = \mathbf{U}\neg s_{4,2} \quad (\wedge \ Ext\ s_{4,2}).$$



(a) Mouse and cheese (b) Mouse and fire

Decomposition



(a) Mouse and Cheese BT (b) Mouse and Fire BT

True or False

Success, Running, or Failure

Creating PPA-Style BT from LTL_f formula

Task grammar uses LTL_v style operators

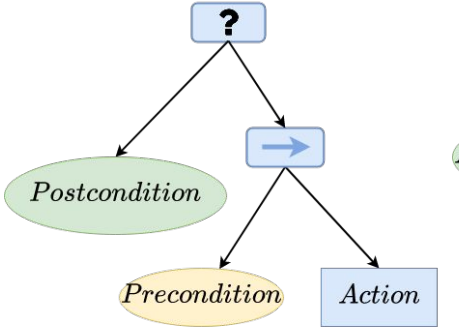
Mission grammar uses LTL_f style operators to combine tasks

LTL_f Formula constrained to structures compatible with robot plans

Robot *Task* Grammar and PPA-Style BT

$\langle \text{PPATask} \rangle :== \vee(\wedge(\mathbf{G}\ GC)\ PoC)(\wedge(\wedge(\mathbf{G}\ GC)\ (PrC))(\mathbf{U}\ (TC)\ (\wedge\ (Action)\ (\mathbf{G}\ GC))))$

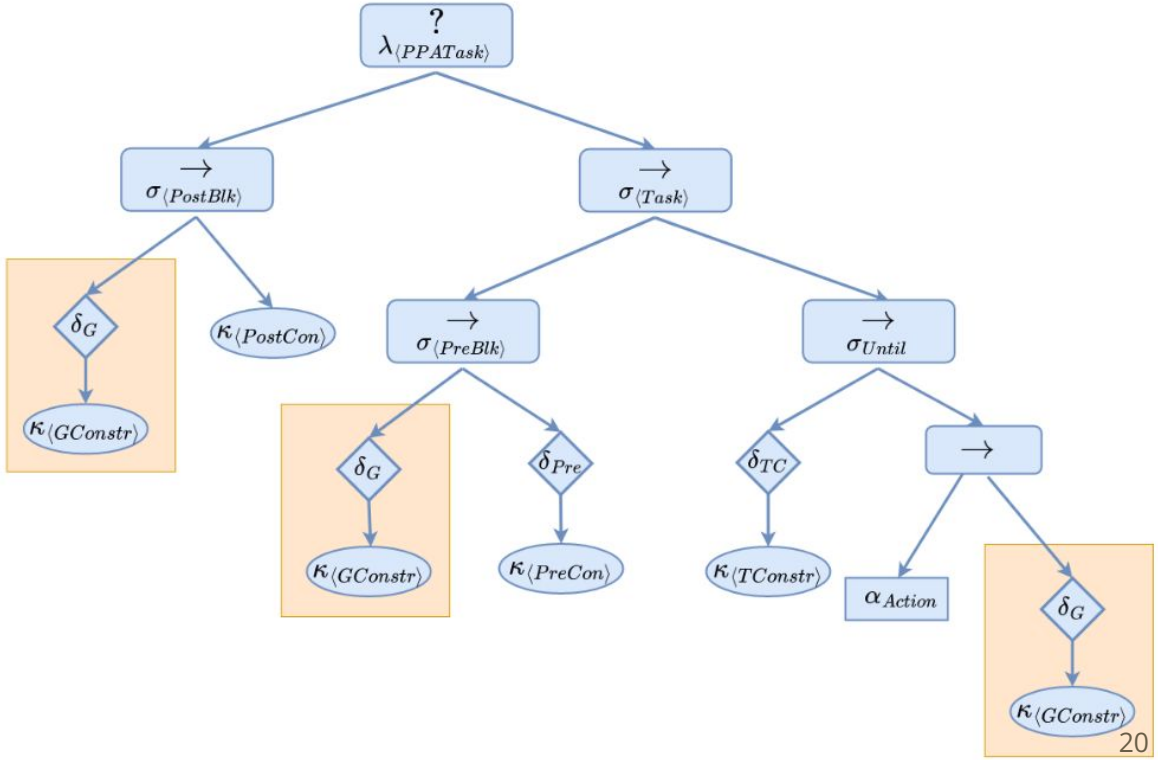
$$
\begin{aligned}
\langle \text{PPATask} \rangle &::= \vee\langle \text{PostBlk} \rangle \langle \text{Task} \rangle \\
\langle \text{PostBlk} \rangle &::= \wedge\langle \text{GCnstr} \rangle \langle \text{PostCon} \rangle \\
\langle \text{Task} \rangle &::= \wedge\langle \text{PreBlk} \rangle \langle \text{ExecBlk} \rangle \\
\langle \text{PreBlk} \rangle &::= \wedge\langle \text{GCnstr} \rangle\ \texttt{PreCon} \\
\langle \text{ExecBlk} \rangle &::= \mathbf{U}\ (\texttt{TCnstr})(\wedge\texttt{Action}\ \langle \text{GCnstr} \rangle) \\
\langle \text{A} \rangle &::= a \mid \langle \text{A} \rangle \wedge \langle \text{A} \rangle \mid \langle \text{A} \rangle \vee \langle \text{A} \rangle \\
\langle \text{PostCon} \rangle &::= \langle \text{A} \rangle \\
\langle \text{GCnstr} \rangle &::= \mathbf{G}\ \langle \text{A} \rangle
\end{aligned}
$$



Robot *Task* Grammar and PPA-Style BT

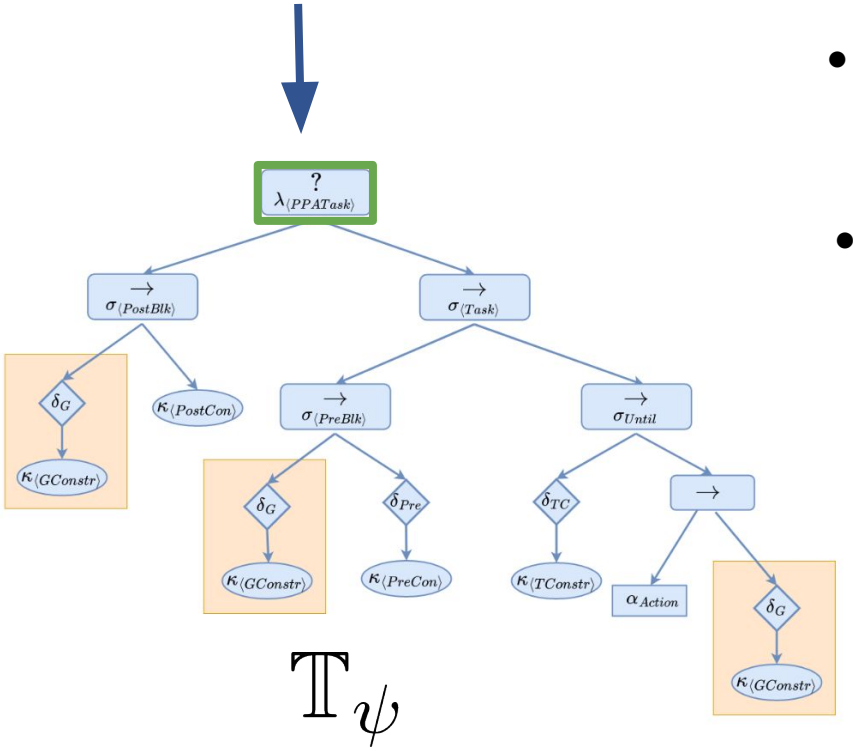
$\langle \text{PPATask} \rangle :== \vee(\wedge(\mathbf{G}\ GC)\ PoC)(\wedge(\wedge(\mathbf{G}\ GC)\ (PrC))(\mathbf{U}\ (TC)\ (\wedge\ (Action)\ (\mathbf{G}\ GC))))$

$$
\begin{aligned}
\langle \text{PPATask} \rangle &::= \vee\langle \text{PostBlk} \rangle \langle \text{Task} \rangle \\
\langle \text{PostBlk} \rangle &::= \wedge\langle \text{GCnstr} \rangle \langle \text{PostCon} \rangle \\
\langle \text{Task} \rangle &::= \wedge\langle \text{PreBlk} \rangle \langle \text{ExecBlk} \rangle \\
\langle \text{PreBlk} \rangle &::= \wedge\langle \text{GCnstr} \rangle\ \texttt{PreCon} \\
\langle \text{ExecBlk} \rangle &::= \mathbf{U}\ (\texttt{TCnstr})(\wedge\texttt{Action}\ \langle \text{GCnst} \\
\langle \text{A} \rangle &::= a \mid \langle \text{A} \rangle \wedge \langle \text{A} \rangle \mid \langle \text{A} \rangle \vee \langle \text{A} \rangle \\
\langle \text{PostCon} \rangle &::= \langle \text{A} \rangle \\
\langle \text{GCnstr} \rangle &::= \mathbf{G}\ \langle \text{A} \rangle
\end{aligned}
$$



Robot *Task* Grammar Theorem

$$\langle \text{PPATask} \rangle :== \vee(\wedge(\mathbf{G}\ GC)\ PoC)(\wedge(\wedge(\mathbf{G}\ GC)\ (PrC))(\mathbf{U}\ (TC)\ (\wedge\ (Action)\ (\mathbf{G}\ GC))))$$



$\mathbb{T}_\psi$

- PPATask BT structures ensure that when BT returns **Success**, the trace is a **satisfies** the LTLf formula.

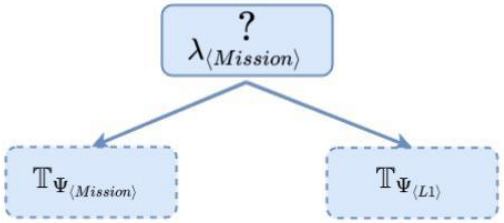
$$success \leftarrow \lambda_{PPATask} \implies \tau \models \psi$$

- PPATask BT structure and system constraints only agenerate a subset of valid traces

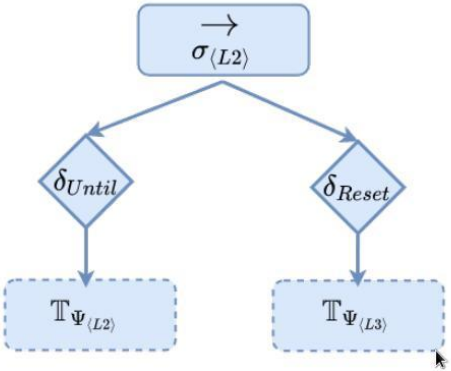
$$L(\mathbb{T}_\psi) \subseteq L(\psi))$$

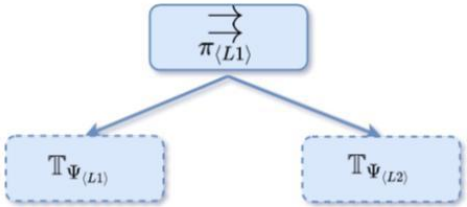
Mission Grammar BT

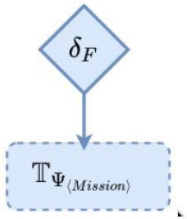
Connects PPATask with LTLf operators



$$
\begin{aligned}
\langle \text{Mission} \rangle &::= \langle \text{L1} \rangle \\
&::= \vee \langle \text{Mission} \rangle \langle \text{L1} \rangle \\
\langle \text{L1} \rangle &::= \langle \text{L2} \rangle \\
&::= \wedge \langle \text{L1} \rangle \langle \text{L2} \rangle \\
\langle \text{L2} \rangle &::= \langle \text{L3} \rangle \\
&::= \mathbf{U} \langle \text{L2} \rangle \langle \text{L3} \rangle \\
\langle \text{L3} \rangle &::= \mathbf{F} \left(\langle \text{Mission} \rangle \right) \\
&::= \left(\langle \text{PPATask} \rangle \right)
\end{aligned}
$$







Mission Grammar BT and Theorem

Connects PPATask with LTLf operators

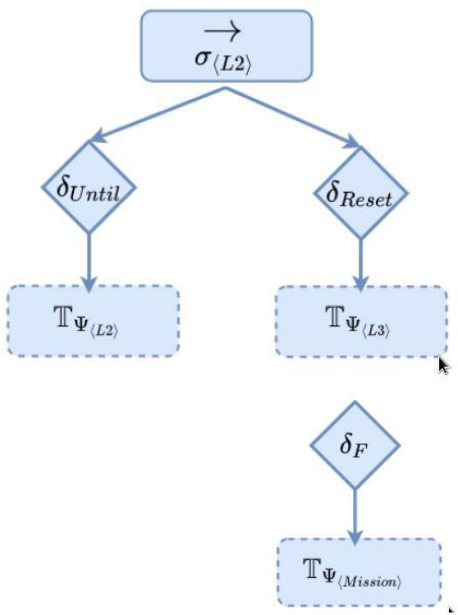
- Mission BT structures ensures that when mission BT returns **Success**, the trace **satisfies** the mission LTLf formula.

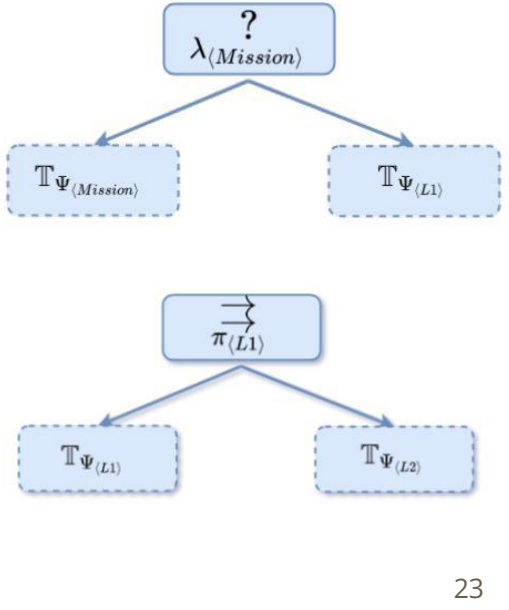
$$success \leftarrow \mathbb{T}_\Psi \implies \tau \models \Psi$$

- Mission structure and environmental constraints only allow a subset of valid traces to be generated

$$L(\mathbb{T}_\Psi) \subseteq L(\Psi))$$

$$\begin{aligned}
\langle \text{Mission} \rangle &::= \langle \text{L1} \rangle && \text{(M1)} \\
&::= \vee \langle \text{Mission} \rangle \langle \text{L1} \rangle && \text{(M2)} \\
\langle \text{L1} \rangle &::= \langle \text{L2} \rangle && \text{(M3)} \\
&::= \wedge \langle \text{L1} \rangle \langle \text{L2} \rangle && \text{(M4)} \\
\langle \text{L2} \rangle &::= \langle \text{L3} \rangle && \text{(M5)} \\
&::= \mathbf{U} \langle \text{L2} \rangle \langle \text{L3} \rangle && \text{(M6)} \\
\langle \text{L3} \rangle &::= \mathbf{F} \left(\langle \text{Mission} \rangle \right) && \text{(M7)} \\
&::= \left(\langle \text{PPATask} \rangle \right) && \text{(M8)}
\end{aligned}$$





Creating BT from LTL_f formula

Task grammar uses LTL_v style operators

Mission grammar uses LTL_f style operators to combine tasks

LTL_f Formula constrained to structures compatible with robot plans

Theorem: Every successful trace generated by the Behavior Tree satisfies the LTL_f formula …

… but not conversely

Goal Specification and Planning

Specify a Goal:
Finite Trace LTL

Plan:
Behavior Tree

Measure Performance:
Successful BT ⇒ Satisfied Goal

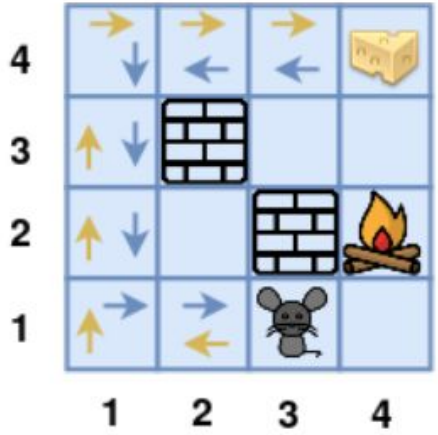
Applications

Reward Misalignment

$$\Psi^{\text{C2H}} = \mathbf{U}\ \mathbf{F}\psi^{\text{Cheese}}\ \mathbf{F}\psi^{\text{Home}}$$

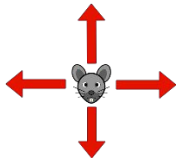
$$\psi^{\text{Cheese}} = \vee(\wedge\neg Fire\ Cheese)(\wedge(\wedge\neg Fire\ True)$$
$$(\mathbf{U}\ True\ \wedge\ Action_{\text{Cheese}}\neg Fire))$$

$$\psi^{\text{Home}} = \vee(\wedge\neg Fire\ Home)(\wedge(\wedge\neg NotFire\ Cheese)$$
$$(\mathbf{U}\ True\ \wedge\ Action_{\text{Home}}))$$

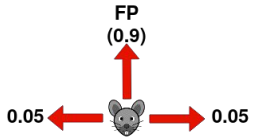


(a) Mouse and cheese

- Separate policies were trained using policy iteration
  - Various reward structures
  - Various intended action probabilities
- Policies are used by the action node to take action in the environment



Agent Actions



Intended Action Probability

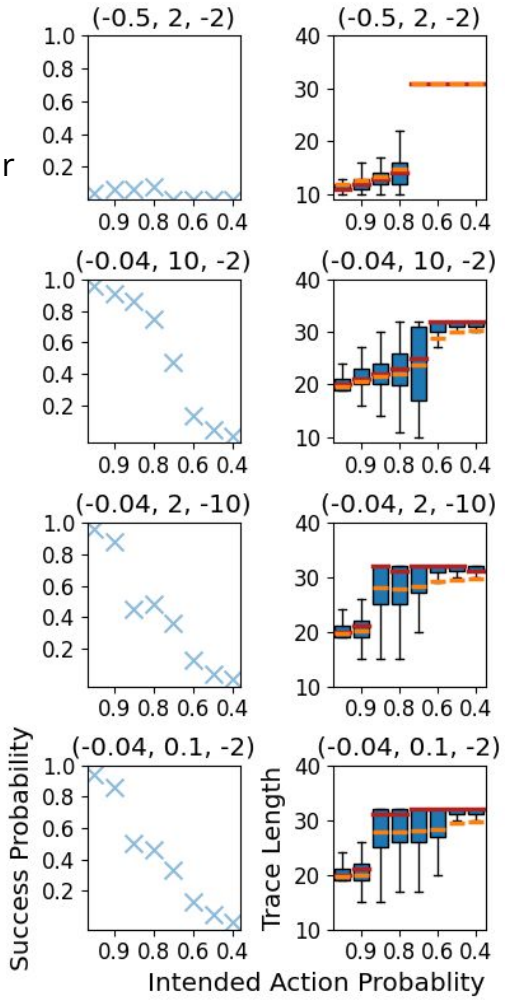
Reward Misalignment

$$\Psi^{\text{C2H}} = \mathbf{U}\ \mathbf{F}\psi^{\text{Cheese}}\ \mathbf{F}\psi^{\text{Home}}$$

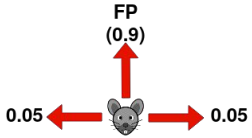
$$\psi^{\text{Cheese}} = \vee(\wedge\neg Fire\ Cheese)(\wedge(\wedge\neg Fire\ True)(\mathbf{U}\ True \wedge Action_{\text{Cheese}}\neg Fire))$$

$$\psi^{\text{Home}} = \vee(\wedge\neg Fire\ Home)(\wedge(\wedge\neg NotFire\ Cheese)(\mathbf{U}\ True \wedge Action_{\text{Home}}))$$

- Points in **First column**, **Higher** is Better
- Boxes in **Second column**, **Lower** is Better
- Each row, different reward structures



- Higher negative rewards leads to lower mission accomplishment rate
- Designing rewards is tricky



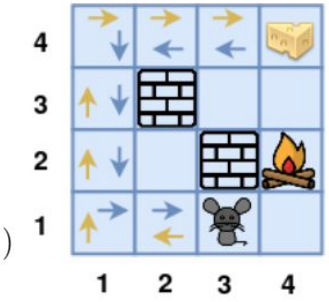
Intended Action Probability

Learning using Mission BT

$$\Psi^{\text{C2H}} = \mathbf{U}\ \mathbf{F}\psi^{\text{Cheese}}\ \mathbf{F}\psi^{\text{Home}}$$

$$\psi^{\text{Cheese}} = \vee(\wedge\neg Fire\ Cheese)(\wedge(\wedge\neg Fire\ True)(\mathbf{U}\ True\ \wedge\ Action_{\text{Cheese}}\ \neg Fire))$$

$$\psi^{\text{Home}} = \vee(\wedge\neg Fire\ Home)(\wedge(\wedge\neg NotFire\ Cheese)(\mathbf{U}\ True\ \wedge\ Action_{\text{Home}}))$$

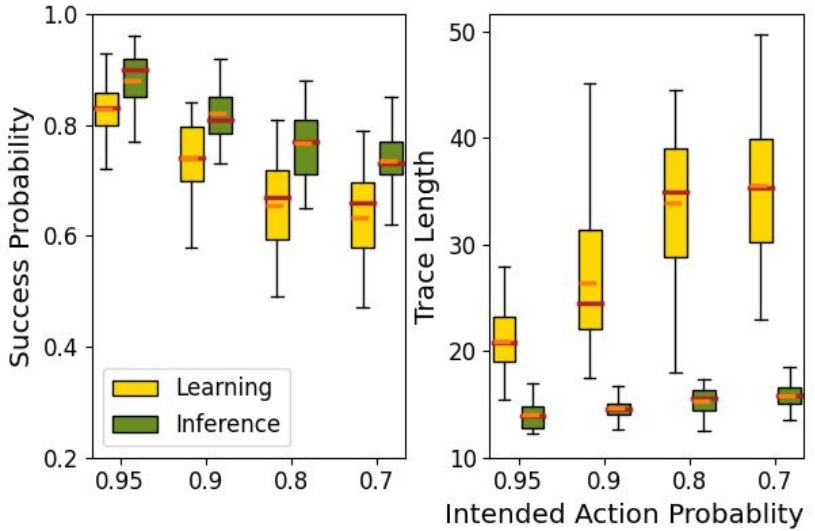


(a) Mouse and cheese

GenRecProp Learning Algorithm

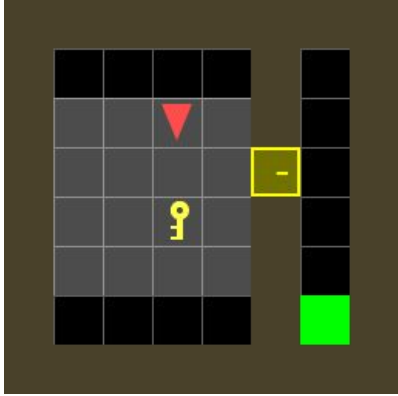
```
For episode in max_run:
    For step in episode:
        state,done = env.step(action)
        If done:
            Break
    For t in len(trace):
```

$$\pi(a(t)|s(t)) \leftarrow \pi(a(t)|s(t)) + \mu^{m-t} * b$$



- **Left** plot, **Higher** is better
- **Right** plot, **Lower** is better

Resilient Robot Execution



Key Door Problem

Mission: use the key to open the door and then get to the goal

Mission LTLf: $\Psi^{KD} = \mathbf{U}(\ \mathbf{F}\ \psi^{Key})(\ \mathbf{U}\ \mathbf{F}\ \psi^{Door}\ \mathbf{F}\ \psi^{Prize})$

Tasks LTLf :

$$\psi^{Key} = \vee(\wedge NoErr\ KeyStacked)(\wedge(\wedge NoErr\ IsKeyDoor)\ (\mathbf{U}\ VisibleKeyDoor\ \wedge\ Action_{KeyStacked}\ NoErr))$$

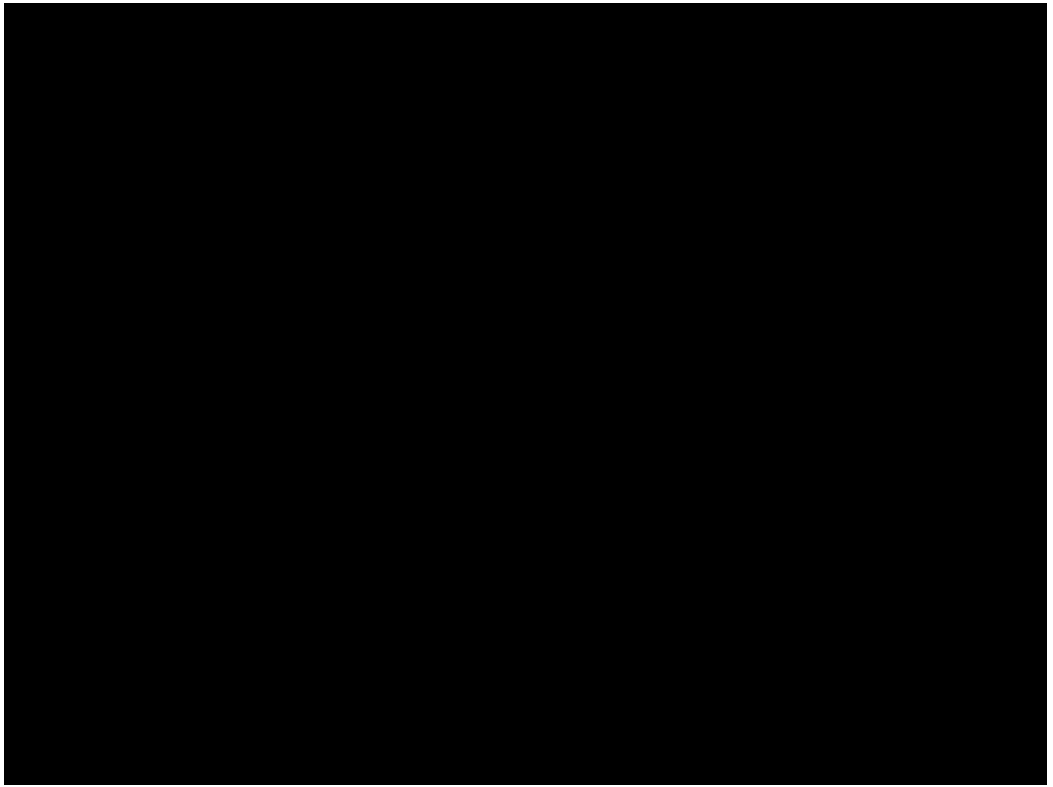
$$\psi^{Door} = \vee(\wedge NoErr\ KeyDoorPassive)(\wedge(\wedge NoErr\ KeyStacked)\ (\mathbf{U}\ KeyStacked\ \wedge\ Action_{KeyDoorPassive}\ NoErr))$$

$$\psi^{Prize} = \vee(\wedge NoErr\ PrizePassive)(\wedge(\wedge NoErr\ PrizeVisible)\ (\mathbf{U}\ KeyDoorPassive\ \wedge\ Action_{PrizePassive}\ NoErr))$$

Fetch Robot in Action

Mission: Move the Key block and stack up the Door block Until carry both blocks to staging area until carry the prize block to staging area

Mission LTLf: $\Psi^{KD} = \mathbf{U}(\ \mathbf{F}\ \psi^{Key})(\ \mathbf{U}\ \mathbf{F}\ \psi^{Door}\ \mathbf{F}\ \psi^{Prize})$



Thank you. I'm happy to take questions.

This work was supported by ONR grant number N000141613025.