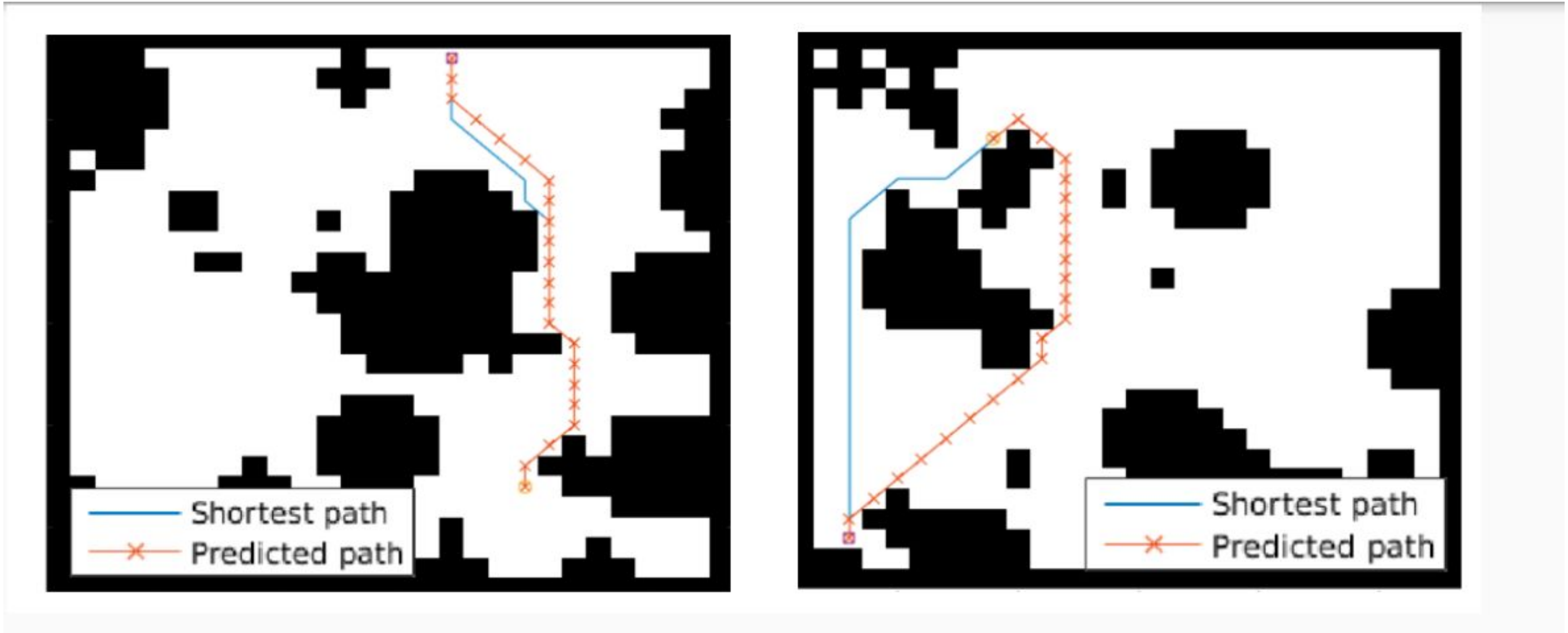
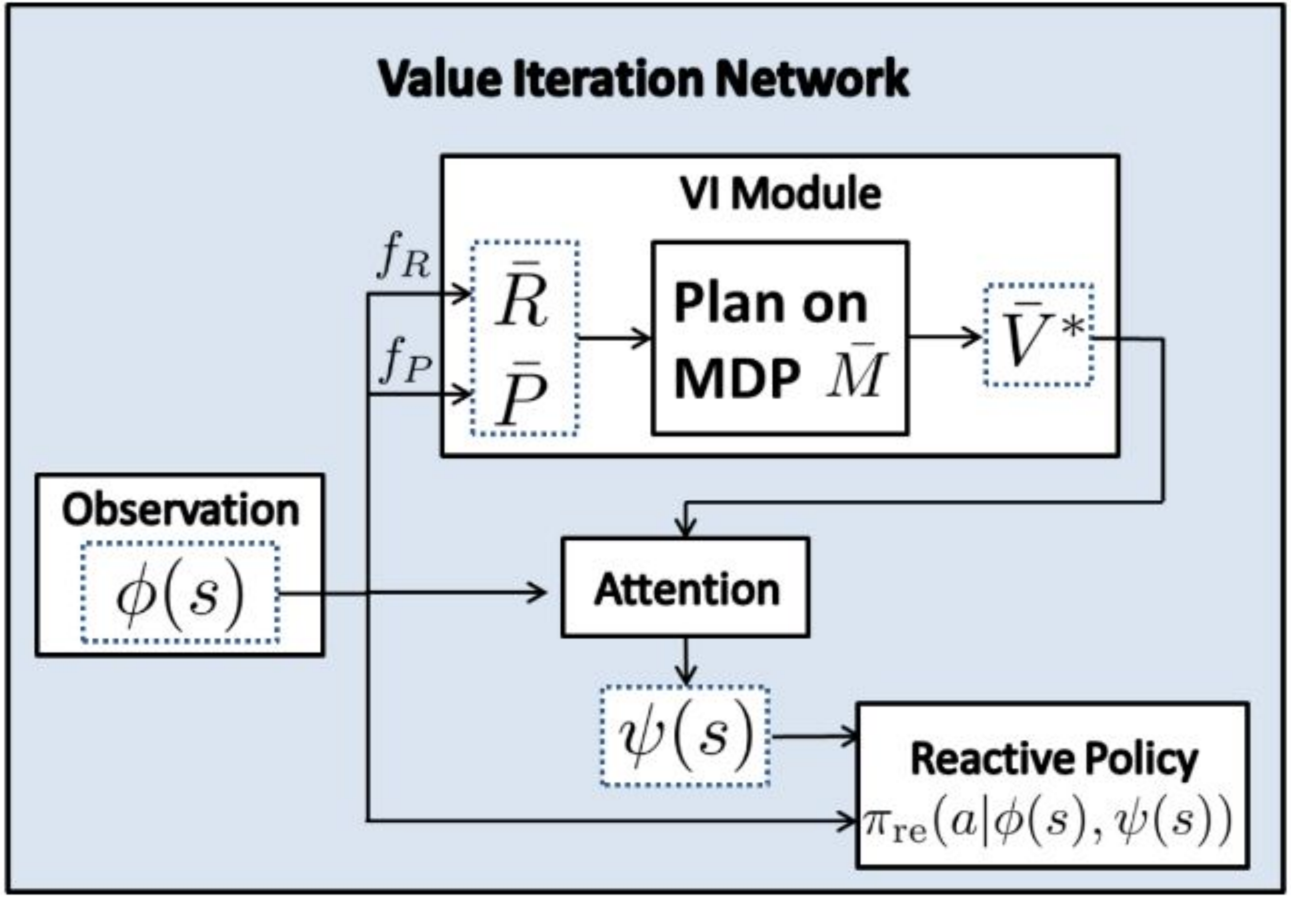
A Planning Problem




Value Iteration Network
VI Module
f_R
f_P
$\bar{R}$
$\bar{P}$
Plan on MDP $\bar{M}$
$\bar{V}^*$
Observation
$\phi(s)$
Attention
$\psi(s)$
Reactive Policy
$\pi_{\mathrm{re}}(a|\phi(s), \psi(s))$

Modular Multitask Reinforcement Learning with Policy Sketches

Jacob Andreas [1] Dan Klein [1] Sergey Levine [1]

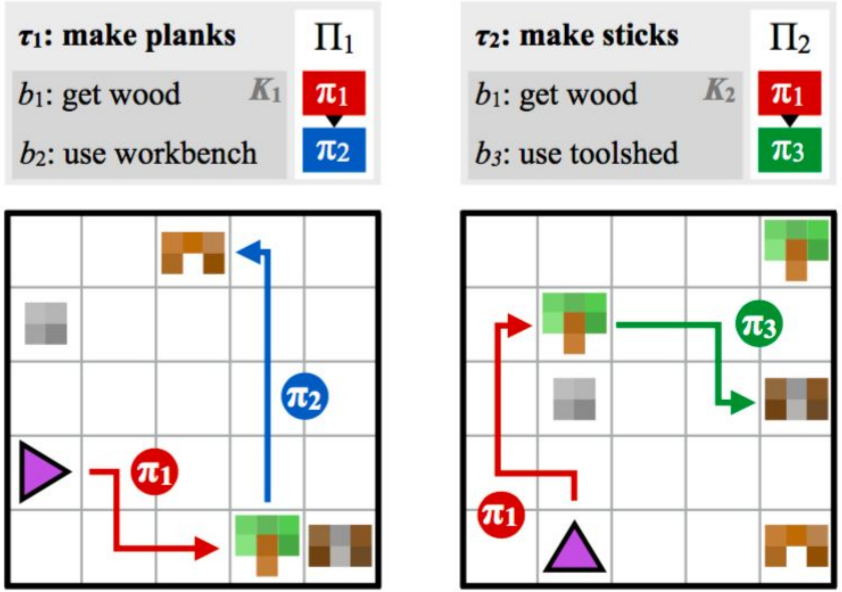


Figure 1: Learning from policy sketches. The figure shows simplified versions of two tasks (*make planks* and *make sticks*, each associated with its own policy (Π_1 and Π_2 respectively). These policies share an initial high-level action b_1: both require the agent to *get wood* before taking it to an appropriate crafting station. Even without prior information about how the associated behavior π_1 should be implemented, knowing that the agent should initially follow the same subpolicy in both tasks is enough to learn a reusable representation of their shared structure.

Algorithm 1 TRAIN-STEP($\boldsymbol{\Pi}$, curriculum)

$\mathcal{D} \leftarrow \emptyset$
while $|\mathcal{D}| < D$ **do**
  // sample task τ from curriculum (Section 3.3)
  $\tau \sim \text{curriculum}(\cdot)$
  // do rollout
  $d = \{(s_i, a_i, (b_i = K_{\tau,i}), q_i, \tau), \ldots\} \sim \Pi_\tau$
  $\mathcal{D} \leftarrow \mathcal{D} \cup d$
// update parameters
for $b \in \mathcal{B}, \tau \in \mathcal{T}$ **do**
  $d = \{(s_i, a_i, b', q_i, \tau') \in \mathcal{D} : b' = b, \tau' = \tau\}$
  // update subpolicy
  $\theta_b \leftarrow \theta_b + \frac{\alpha}{D} \sum_d \left(\nabla \log \pi_b(a_i|s_i)\right)\left(q_i - c_\tau(s_i)\right)$
  // update critic
  $\eta_\tau \leftarrow \eta_\tau + \frac{\beta}{D} \sum_d \left(\nabla c_\tau(s_i)\right)\left(q_i - c_\tau(s_i)\right)$

How can we express logic?

In this section, we provide definitions for TLTL (refer to our previous work [17] for a more elaborate discussion of TLTL). A TLTL formula is defined over predicates of form $f(s) < c$, where $f : \mathbb{R}^n \to \mathbb{R}$ is a function of state and c is a constant. We express the task as a TLTL formula with the following syntax:

$$\begin{aligned} \phi := \top \mid f(s) < c \mid \neg\phi \mid \phi \wedge \psi \mid \phi \vee \psi \mid \\ \Diamond\phi \mid \Box\phi \mid \phi\,\mathcal{U}\,\psi \mid \phi\,\mathcal{T}\,\psi \mid \bigcirc\phi \mid \phi \Rightarrow \psi, \end{aligned} \tag{1}$$

where $\top$ is the boolean constant true, $f(s) < c$ is a predicate, $\neg$ (negation/not), $\wedge$ (conjunction/and), and $\vee$ (disjunction/or) are Boolean connectives, and $\Diamond$ (eventually), $\Box$ (always), $\mathcal{U}$ (until), $\mathcal{T}$ (then), $\bigcirc$ (next), are temporal operators. Implication is denoted by $\Rightarrow$ (implication). TLTL formulas are evaluated against finite time sequences of states $\{s_0, s_1, \ldots, s_T\}$.

Algorithm 1 Temporal Logic Policy Search

Inputs: Episode horizon T, batch size N, KL constraint parameter ϵ, smoothed robustness function $\hat{\rho}(s_{0:T}, \phi)$, softmax parameter $\alpha > 0$
Initialize policy $\pi \leftarrow (K_t, k_t, C_t)$
Initialize trajectory buffer $\mathcal{B} \leftarrow \emptyset$
for $m = 1$ to *number of training episodes* **do**
    τ_m = SampleTrajectories(π, T)
    Store τ_m in $\mathcal{B}$
    if Size($\mathcal{B}$) $\geq N$ **then**
        $\bar{\tau}^i \leftarrow$ GetUpdatedTrajectories(τ^i) **for** $i = 1$ to N
end for ▷ Using Equation (6)
        $\bar{\mu}_\tau, \bar{\Sigma}_\tau \leftarrow$ FitTrajectoryDistribution($\{\tau_1, ..., \tau_N\}$)
▷ Using Equation (7)
        for i=1 to N **do**
            $p^i \leftarrow \mathcal{N}(\tau^i | \bar{\mu}_\tau, \bar{\Sigma}_\tau)$
            $w^i = \frac{e^{\alpha p^i}}{\sum_{i=1}^N e^{\alpha p^i}}$
        end for
        for t = 0 to T-1 **do**
            $k'_t \leftarrow \sum_i^N w^i k_t^i$
            $C'_t \leftarrow \sum_i^N w^i (k_t^i - k'_t)(k_t^i - k'_t)^T$
        end for
        Clear buffer $\mathcal{B} \leftarrow \emptyset$
    end if
end for

A Policy Search Method For Temporal Logic Specified Reinforcement Learning Tasks

Xiao Li, Yao Ma and Calin Belta

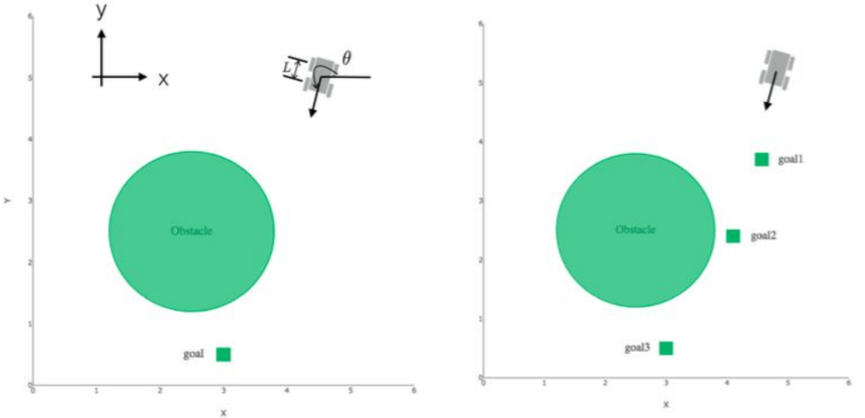


Fig. 1 : Vehicle navigation task using TLTL specifications. The vehicle is shown in brown, the obstacle is shown as the green circle and the goals are shown as the green squares. *left*: Task 1 is to reach the goal while avoiding the obstacle. *right*: Task 2 is to visit goals 1,2,3 in this order while avoiding the obstacle

$$
\begin{aligned}
\rho(s_{t:t+k}, \top) &= \rho_{max}, \\
\rho(s_{t:t+k}, f(s_t) < c) &= c - f(s_t), \\
\rho(s_{t:t+k}, \neg\phi) &= -\rho(s_{t:t+k}, \phi), \\
\rho(s_{t:t+k}, \phi \Rightarrow \psi) &= \max(-\rho(s_{t:t+k}, \phi), \rho(s_{t:t+k}, \psi)) \\
\rho(s_{t:t+k}, \phi_1 \wedge \phi_2) &= \min(\rho(s_{t:t+k}, \phi_1), \rho(s_{t:t+k}, \phi_2)), \\
\rho(s_{t:t+k}, \phi_1 \vee \phi_2) &= \max(\rho(s_{t:t+k}, \phi_1), \rho(s_{t:t+k}, \phi_2)), \\
\rho(s_{t:t+k}, \bigcirc\phi) &= \rho(s_{t+1:t+k}, \phi) \ (k > 0), \\
\rho(s_{t:t+k}, \Box\phi) &= \min_{t' \in [t, t+k)} (\rho(s_{t':t+k}, \phi)), \\
\rho(s_{t:t+k}, \Diamond\phi) &= \max_{t' \in [t, t+k)} (\rho(s_{t':t+k}, \phi)), \\
\rho(s_{t:t+k}, \phi\, \mathcal{U}\, \psi) &= \max_{t' \in [t, t+k)} (\min(\rho(s_{t':t+k}, \psi), \\
&\quad \min_{t'' \in [t, t')} \rho(s_{t'':t'}, \phi))), \\
\rho(s_{t:t+k}, \phi\, \mathcal{T}\, \psi) &= \max_{t' \in [t, t+k)} (\min(\rho(s_{t':t+k}, \psi), \\
&\quad \max_{t'' \in [t, t')} \rho(s_{t'':t'}, \phi))),
\end{aligned}
$$

where ρ_{max} represents the maximum robustness value.

Measuring conformity to logic.