Bellman Value Decomposition for Task Logic in Safe Optimal Control

Anonymous

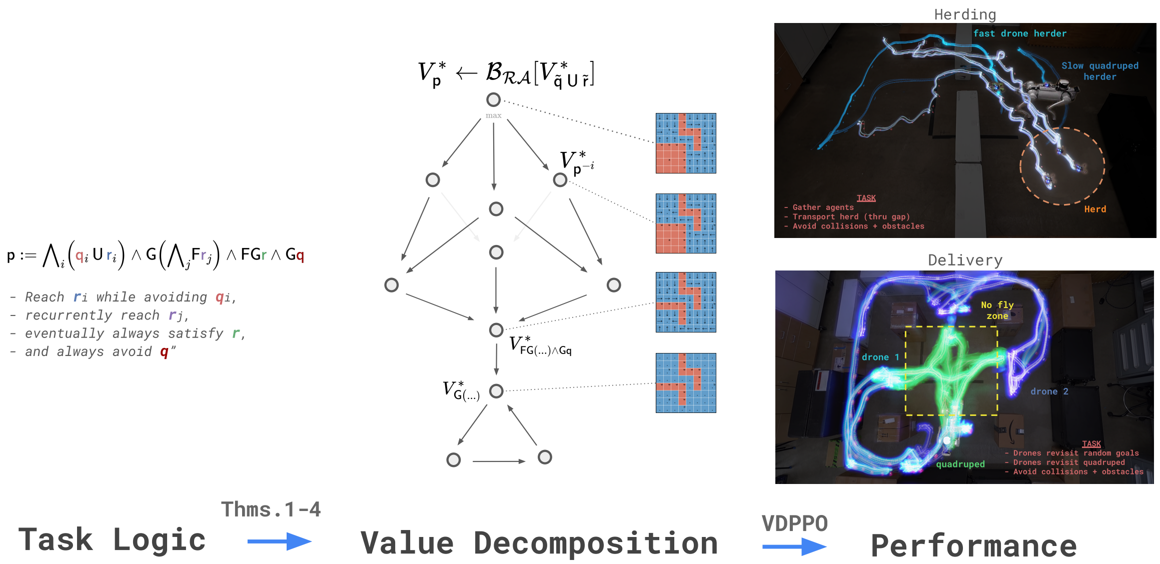

Fig. 1: **Value-Decomposition PPO (VDPPO).** The Bellman value for a wide range of temporal logic formulae (e.g., multi-goal, recurrence, stability, safety) decomposes into a value graph connected by atomic Bellman equations (Thms. 1–4). We propose VDPPO, an algorithm that exploits the structure of the value graph to learn policies for complex, high-dimensional tasks. Our approach is validated on hardware with `Herding` and `Delivery`, two tasks involving a heterogeneous team of drones and a quadruped.

I. Abstract

Real-world tasks involve nuanced combinations of goal and safety specifications, which often directly compete. In high dimensions, the challenge is exacerbated: formal automata become cumbersome, and the combination of sparse rewards tends to require laborious tuning. In this work, we consider the structure of the Bellman Value as a means to naturally organize the problem for improved automatic performance without introducing additional abstractions. Namely, **we prove the Bellman Value for a complex task defined in temporal logic can be decomposed into a graph of Bellman Values**, where the graph is connected by a set of well-studied Bellman equations (BEs): the Reach-Avoid BE, the Avoid BE, and a novel type, the Reach-Avoid-Loop BE. From this perspective, we design a specialized PPO variant, Value-Decomposition PPO (**VDPPO**) that uses a single learned representation by embedding the decomposed Value graph. We conduct a variety of simulated and real multi-objective experiments, including delivery and herding, to test our method on diverse high-dimensional systems involving heterogeneous teams and complex agents. Ultimately, we find this approach greatly improves performance over existing baselines, balancing safety and liveness automatically.

II. Introduction and Related Work

Reinforcement Learning (RL) typically optimizes expected cumulative reward [1], making it ill-suited for safety-critical or temporally structured tasks that require worst-case guarantees or satisfaction at specific times. Such objectives are naturally expressed using Temporal Logic (TL) [2], but TL itself does not prescribe how to act. Existing RL–TL methods therefore face a trade-off between sparse binary rewards that slow learning and hand-crafted dense rewards that can misalign objectives.

Hamilton–Jacobi Reachability (HJR) [3, 4] provides optimal controllers for basic safety and liveness tasks via max–min Bellman equations, yielding dense and informative learning signals. Recent work showed that certain TL tasks can be solved exactly by decomposing their value functions into sequences of simple HJR problems [5]. We generalize this idea to a broad class of TL specifications, introduce a value-function decomposition algebra and a corresponding PPO variant, and demonstrate effectiveness in simulation and real-world drone and quadruped experiments.

RL with TL specifications A large body of work study RL with TL specifications [6, 7, 8, 9, 10, 11], including approaches based on Non-Markovian Reward Decision Processes [12, 13, 14, 15, 2], approximated quantitative semantics [16, 17, 18], modified Bellman equations [19, 20, 21], or multiple discounted rewards [22, 23, 24]. In contrast, our method exactly decomposes TL value functions into simpler objectives solved via HJR, avoiding semantic approximation and long-horizon reward sparsity. Additional discussion appears in the Appendix and [5].

Constrained, Multi-Objective, and Goal-Conditioned RL.

Constrained MDPs (CMDPs) maximize discounted rewards subject to constraints, typically via Lagrangian relaxation [25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38], but require careful tuning and are ill-suited to general TL objectives. Multi-objective RL instead Pareto-optimizes multiple reward sums [39, 40, 41, 42, 43, 44, 45], yet does not naturally encode TL structure. Goal-conditioned RL learns policies over a family of goals [46, 47, 48, 49, 50, 51, 52, 49, 50, 53], but differs fundamentally from TL settings, where all specifications must be jointly satisfied.

Hamilton–Jacobi Reachability. HJR was originally developed to compute value functions for reach, avoid, and reach-avoid problems in continuous time and space [3, 4], corresponding to the quantitative semantics of eventually, always, and until predicates [54]. Recent work has successfully integrated HJR into RL frameworks [55, 56, 57, 58, 59, 60]. Our work builds on these results by decomposing value functions for complex TL objectives into sequences of simpler HJR problems.

III. Contributions

1) We establish a formal connection between Temporal Logic and Bellman Value theory, characterizing both equivalences (Lem. 1) and divergences (e.g., Rem. 1).
2) We prove that a broad class of TL predicates admits an exact decomposition of the Value function into a directed graph of atomic Bellman equations (Thms. 1, 3, 4), including a novel Reach–Avoid–Loop Bellman equation for always–eventually specifications (Lem. 2).
3) We introduce **VDPPO**, an algorithm that solves the decomposed value graph, and demonstrate its effectiveness through extensive simulation and real-world hardware experiments, achieving improved speed and success over existing methods.

IV. Preliminaries

Given a discrete-time system $x_{t+1} = f(x_t, a_t)$ with state $x_t \in \mathcal{X} \subseteq \mathbb{R}^n$ and action $a_t \in \mathcal{A} \subseteq \mathbb{R}^m$, a trajectory beginning at x is a sequence of states $\xi_x^\alpha := (x, ...) \in \mathbb{X} := \mathcal{X}^\mathbb{N}$ arising from actions $\alpha = (a, ...) \in \mathbb{A} := \mathcal{A}^\mathbb{N}$. We let $\xi_x(t)$ and $\alpha(t)$ be the state and action at time t.

To specify desired properties of a trajectory, let an atomic predicate $\mathsf{r} : \mathbb{R}^n \to \{\text{true}, \text{false}\}$ be defined by a bounded predicate function $r : \mathbb{R}^n \to \mathbb{R}$, also known as a target or reward function in HJR or RL. Given a trajectory and time (ξ_x, t), r is satisfied (written $(\xi_x, t) \models \mathsf{r}$) iff $r(\xi_x(t)) \geq 0$, and thus, r is employed to represent the arrival of a trajectory at a goal or obstacle (defined by the 0-level-set of r).

To represent complex tasks, TL defines a logic for modular composition of predicates [61]. Namely, predicates may be composed via negation ($\neg$), conjunction/and ($\wedge$), the *Until* operator (U) and the next operator (X). With these operations, one may also define disjunction/or ($\vee$), finally/eventually (F), and globally/always (G). We give these operators via the robustness score $\rho : \mathbb{R}^n \to \mathbb{R}$ [62] because this is the payoff used in the corresponding HJB optimal control problem [3, 4]. See the Appendix for more details.

Definition 1. *For any predicate p composed of atomic predicates r_i, let the robustness score $\rho[\mathsf{p}] : \mathbb{X} \times \mathbb{N} \to \mathbb{R}$ be defined inductively with*

$$\begin{aligned}
\rho[\mathsf{r}_i](\xi_x, t) &:= r_i(\xi_x(t)) \\
\rho[\neg\mathsf{p}](\xi_x, t) &:= -\rho[\mathsf{p}](\xi_x, t) \\
\rho[\mathsf{p} \wedge \mathsf{p}'](\xi_x, t) &:= \min\left\{\rho[\mathsf{p}](\xi_x, t), \rho[\mathsf{p}'](\xi_x, t)\right\} \\
\rho[\mathsf{p} \vee \mathsf{p}'](\xi_x, t) &:= \max\left\{\rho[\mathsf{p}](\xi_x, t), \rho[\mathsf{p}'](\xi_x, t)\right\} \\
\rho[\mathsf{X}\mathsf{p}](\xi_x, t) &:= \rho[\mathsf{p}](\xi_x, t+1) \\
\rho[\mathsf{F}\mathsf{p}](\xi_x, t) &:= \max_{\tau \geq t} \rho[\mathsf{p}](\xi_x, \tau) \\
\rho[\mathsf{G}\mathsf{p}](\xi_x, t) &:= \min_{\tau \geq t} \rho[\mathsf{p}](\xi_x, \tau) \\
\rho[\mathsf{p} \,\mathsf{U}\, \mathsf{p}'](\xi_x, t) &:= \\
&\max_{\tau \geq t} \min\left\{\rho[\mathsf{p}'](\xi_x, \tau), \min_{\kappa \in [t, \tau]} \rho[\mathsf{p}](\xi_x, \kappa)\right\}
\end{aligned}$$

with $(\xi_x, t) \models \mathsf{p} \iff \rho[\mathsf{p}](\xi_x, t) \geq 0.$

Note that $\mathsf{F}\mathsf{p} = \top \,\mathsf{U}\, \mathsf{p}$, where $\top$ is true, and thus it suffices to consider only the U and G operations in the analysis. Similarly, $\mathsf{G}\mathsf{q} \wedge \mathsf{G}\mathsf{q}' = \mathsf{G}(\mathsf{q} \wedge \mathsf{q}') = \mathsf{G}\mathsf{q}''$ so we will write our always-specifications together. With this syntax, one may express the satisfaction of complex specifications over trajectories formally and succinctly.

V. Problem Formulation

In this work, we consider the problem of synthesizing optimal actions α and a policy $\pi : \mathcal{X} \to \mathcal{A}$ (Appendix), such that for any initial state x the resulting trajectory ξ_x^α maximizes the payoff ρ for a given predicate. We assume the system begins at $t = 0$ and evolves indefinitely. For brevity, we let $\rho[\mathsf{p}](\xi) := \rho[\mathsf{p}](\xi, 0)$. This leads to the following infinite-horizon Safe Optimal Control Problem (SOCP),

$$\begin{aligned}
\text{maximize}_\alpha \quad & \rho[\mathsf{p}](\xi_x^\alpha), \\
\text{s.t.} \quad & \xi_x^\alpha(t+1) = f\left(\xi_x^\alpha(t), \alpha(t)\right).
\end{aligned}$$

Note, because ρ is defined by temporal extrema (max/min over time), this program induces behavior characterized by its *outlying performance*, in contrast with a sum-based SOCP (in canonical RL [1]) which selects for average behavior. This objective is explicitly captured by the Bellman Value function, the "high score" function for the given SOCP.

Definition 2. *For a predicate p, we aim to solve the Bellman Value function*

$$V^*[\mathsf{p}](x) := \max_\alpha \rho[\mathsf{p}](\xi_x^\alpha). \tag{1}$$

We have defined the Bellman Value for a general TL predicate p, or V_p^* for brevity, but in fact, for the operations $\mathsf{G}\mathsf{q}$ and $\mathsf{q} \,\mathsf{U}\, \mathsf{r}$ this object has been extensively studied in the HJR literature [3, 57, 63]. Namely, the Value for these operations are known as the Avoid ($\mathcal{A}$) and Reach-Avoid ($\mathcal{RA}$) Values. In this context, the following contractive Bellman operations for these extrema-based Values have been derived [57].

Definition 3. *The $\mathcal{A}$ and $\mathcal{RA}$ Bellman operators [57],*

$$\mathcal{B}^{\gamma}_{\mathcal{A}}[V] := (1-\gamma)q + \gamma \min\{V^{+}, q\},$$
$$\mathcal{B}^{\gamma}_{\mathcal{RA}}[V] := (1-\gamma)\min\{r, q\} + \gamma \min\{\max\{V^{+}, r\}, q\},$$

where $V^{+}(x) := \max_a V(f(x,a))$, are contractive. For $V^*[\mathsf{Gq}]$ and $V^*[\mathsf{qUr}]$ defined in (1), the fixed points

$$V^{\gamma}[\mathsf{Gq}] = \mathcal{B}^{\gamma}_{\mathcal{A}}[V^{\gamma}[\mathsf{Gq}]] \quad \& \quad V^{\gamma}[\mathsf{qUr}] = \mathcal{B}^{\gamma}_{\mathcal{RA}}[V^{\gamma}[\mathsf{qUr}]],$$

satisfy $\lim_{\gamma\to 1} V^{\gamma} = V^*$ by Thm. 1 of [57].

These Bellman operators differ from those which arise with a discounted-sum [1], as they propagate maximum or minimum (extremum) values, thus encouraging behavior defined by outlying performance. This has proved to make the RL algorithms based on these equations significantly better at safety and achievement tasks [55, 64].

In a recent work [5], it was demonstrated that for simple conjunctions $\mathsf{Fr} \wedge \mathsf{Gq}$ and $\mathsf{Fr}_1 \wedge \mathsf{Fr}_2$, one may decompose the corresponding Bellman Values into these "atomic" BE, which in some ways resembles the base case for what follows. In this work, we generalize this principle, demonstrating that the $\mathcal{A}$-BE and $\mathcal{RA}$-BE, along with the novel REACH-AVOID-LOOP BE (Lem. 2), serve as a set of "atomic" building blocks to decompose the Bellman Value of complex TL predicates.

VI. MOTIVATION

A. Why the Value function?

Above all, the Value function serves to define an optimal policy for autonomy. Moreover, this Value function has several properties which motivate the work, and we discuss them here.

Value functions are stable, policies need not be. While the value function is Lipschitz continuous, its gradient—and thus the optimal policy—may be discontinuous. Consequently, nearby states can induce very different optimal trajectories, making direct policy learning unstable under noise.

The value remains informative even for infeasible tasks. $V^*_{\mathsf{p}}(x)$ characterizes both satisfiability (≥ 0) and degree of violation. Hence, maximization produces policies that minimize failure when satisfaction is impossible.

Value decomposition yields dense, aligned learning signals. Sparse binary rewards provide little guidance, while dense rewards under discounted sums often conflate with TL objectives. Decomposing the value function produces a hierarchy of dense rewards that directly reflect the structure of the TL specification.

Extremum-based decomposition enforces safety without tuning. Because each subproblem is governed by an extremum-based Bellman equation, worst-case and best-case outcomes propagate without additive trade-offs. This naturally prioritizes safety and goal achievement, avoiding the Lagrangian tuning required by constrained RL methods [28].

B. Optimality versus Satisfaction

The decomposition of formal logic is well-studied in several contexts, including formal verification [65], automata theory [66], and temporal logic trees (TLT) [67]. This body of work has established a rich framework for understanding the structure and properties of temporal logic formulas, and has led to performant decompositional learning methods for complex tasks [68]. However, the algebra of TL, which is equivalent to the algebra over the robustness score, is fundamentally distinct from that of the Value function due to the presence of the maximum over action sequences or control policies in (2). This distinction is not only relevant to theoretical analysis but can lead to safety failures and sub-optimality in real world applications. We illustrate this with the following remark and offer concrete counter-examples in the Appendix.

Remark 1. *The following TL identity always holds:*

$$\rho[\mathsf{Fr} \wedge \mathsf{Gq}](\xi^{\alpha}_{x}) = \min\{\rho[\mathsf{Fr}](\xi^{\alpha}_{x}), \rho[\mathsf{Gq}](\xi^{\alpha}_{x})\}.$$

By contrast, for the corresponding Value, we have

$$V^*[\mathsf{Fr} \wedge \mathsf{Gq}](x) \leq \min\{V^*[\mathsf{Fr}](x), V^*[\mathsf{Gq}](x)\},$$

where the inequality is indeed strict when no single choice of action sequence can both reach r *and avoid* q.

VII. RESULTS

In this section, we present our main results regarding the decomposition of the Bellman Value for complex TL predicates. We begin by discussing the relationship between the Value and TL algebra, and then proceed to present a series of decomposition theorems culminating in a general decomposition result for a class of TL predicates. In general, we seek to express the Bellman Value for a complex predicate in terms of simpler components that are themselves composed with the fundamental Bellman equations of HJR (and thus may be solved similarly), and we will observe that these are associated with subsets of the overall logic. We give all proofs in the Appendix.

A. Agreeable Algebra

We begin by noting the similarity between the decomposition of the Bellman Value and TL algebra. The presence of the $\max_{\alpha}$ in (1) does not always distinguish the Value algebra from that of TL, namely when the TL is also defined by maxima, as with $\vee$ and a "right-side" U (for which F, is a special case). The commutativity of $\max$ in this case yields a decomposition that mirrors that of the TL, giving the following results.

Lemma 1. *Let* $\mathsf{v_p}$ *be the predicate for* $V[\mathsf{p}]$, *i.e.* $(\xi_x, t) \models \mathsf{v_p} \iff V[\mathsf{p}](\xi_x(t)) \geq 0$. *Recall that*

$$\rho[\mathsf{v_p}](\xi_x, t) := V[\mathsf{p}](\xi_x(t)). \tag{2}$$

The following properties hold:

1) $V[\mathsf{a} \vee \mathsf{b}](x) = V[\mathsf{v_a} \vee \mathsf{v_b}](x)$
2) $V[\mathsf{a}\,\mathsf{U}\,\mathsf{b}](x) = V[\mathsf{a}\,\mathsf{U}\,\mathsf{v_b}](x)$

This result makes some compositions of the Value simple to consider. For example, we may know that the Value for a series of Until predicates is equivalent to a chain of Until Values, i.e. a chain of $\mathcal{RA}$ Values. Moreover, the Value for FG, also known

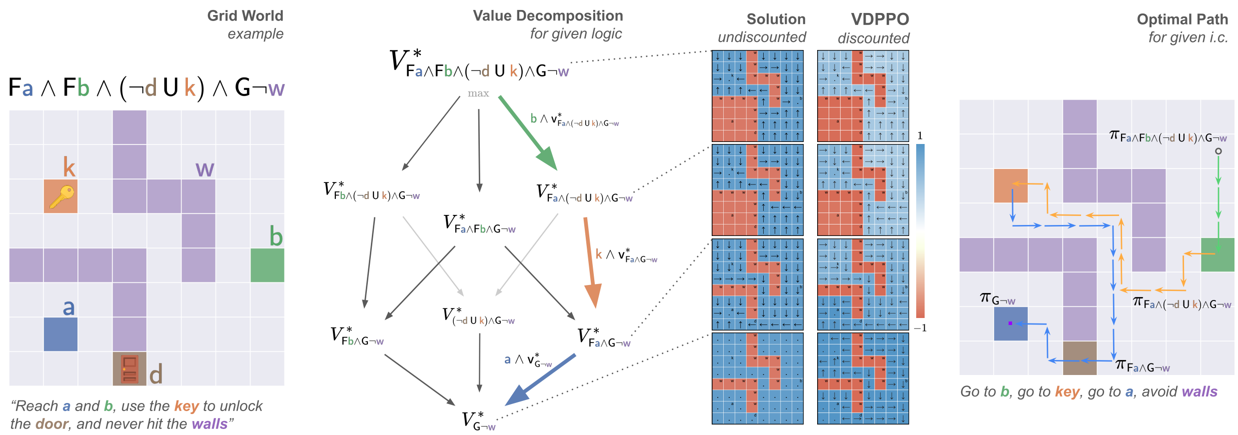

Fig. 2: **E.g. N-Until-Conjunction Value Decomposition.** Here we illustrate the primary decomposition result (Thm. 1 extension, Appendix), with a `GridWorld` example (left) for a given specification. The corresponding DVG is shown (center left) with each node representing a decomposed Value, and edges representing dependencies. In the center right, a subset of decomposed Values solved with dynamic programming are shown, along with the discounted solution produced by VDPPO. On the right, the optimal path for a given initial condition is shown.

as the reach-stay problem, is simply a $\mathcal{R}$ Value where the target is the $\mathcal{A}$ Value associated with the always predicate. See the Appendix for more details. These results, however, do not apply when the TL is defined by $\min$ as with $\wedge$, and thus are insufficient to decompose the Value for many common TL predicates.

B. N-Until-Conjunction Decomposition

We next present the first major result of the work concerning the decomposition of the Bellman Value for the conjunction of N Until predicates, or equivalently, the N-$\mathcal{RA}$ Value. This result is a generalization of the RR Value function decomposition in [5], which explored the independent pairwise combination of two reach tasks.

Theorem 1. *For the predicate* $\mathsf{p} := \bigwedge_{i\in\mathcal{I}}(\mathsf{q}_i \;\mathsf{U}\; \mathsf{r}_i)$, *the corresponding Bellman Value satisfies*

$$V^*\big[\bigwedge_i(\mathsf{q}_i \;\mathsf{U}\; \mathsf{r}_i)\big](x) = V^*\big[\tilde{\mathsf{q}} \,\mathsf{U}\, \tilde{\mathsf{r}}\big](x)$$

where,

$$\tilde{\mathsf{r}} := \bigvee_i \left(\mathsf{r}_i \wedge \mathsf{v}^*_{\mathsf{p}^{-i}}\right), \qquad \tilde{\mathsf{q}} := \bigwedge_i \mathsf{q}_i,$$

and $\mathsf{p}^{-i} := \bigwedge_{j\in\mathcal{I}\setminus\{i\}} \mathsf{q}_j \;\mathsf{U}\; \mathsf{r}_j$.

This result gives an equivalence between the N-$\mathcal{RA}$ Value and the Value function of a single $\mathcal{RA}$ task, which has abstract reach and avoid predicates in the sense that they no longer represent physical goals or obstacles. Instead, the new reach predicate $\tilde{\mathsf{r}}$ is defined by the disjunction of N conjunctions that each correspond to reaching one of the predicates r_i and being able to satisfy the remaining logic p^{-i}, i.e. having $V^*_{\mathsf{p}^{-i}} > 0$. The new avoid predicate $\tilde{\mathsf{q}}$ is defined by the conjunction of all N-avoid predicates and hence implies that we need to avoid all q_j. Intuitively, Thm. 1 breaks down the optimal value for the conjunction of N Untils into the goal of reaching any of the predicates while being able to satisfy the rest of the predicate of $N-1$ Until operations, denoted p^{-i}, where r_i has been 'popped off' the original predicate.

Notably, Thm. 1 is recursive, and, therefore, we may reapply the result iteratively to the N-$\mathcal{RA}$ Value to break it into N decomposable sub-Values and so forth, giving $2^N - 1$ Values in total. Crucially, as each of these Values is equivalent to a special Until Value, *they may each be solved with the discounted $\mathcal{RA}$-BE* with their respective rewards. We demonstrate this result in Fig.2 with a simple `GridWorld` problem, where the true solution may be solved via dynamic programming.

Analogous to the proof of the Reach-Always-Avoid Value in [5], this result can in fact be extended to the case where $\mathsf{p} := \bigwedge_{i\in\mathcal{I}} (\mathsf{q}_i \;\mathsf{U}\; \mathsf{r}_i) \wedge \mathsf{G}\mathsf{q}$. In this case, the only difference is that the presence of $\mathsf{G}\mathsf{q}$ persists to the ultimate sub-Value, which is at this point equivalent to the RAA Value posed in [5]. We give this in the Appendix.

C. Recursive Decompositions

In this section, we consider the family of recurrence relation operations corresponding to the composition of G with U (for which GF is a special case). To always-eventually satisfy a predicate implies that a trajectory must continue to satisfy it indefinitely. These compositions are particularly important as they encompass the liveness property, arising in safety-critical applications where certain states or tasks must be revisited or regenerated in some sense. Moreover, this operation is significantly less strict than the FG (which requires that we eventually satisfy the predicate continuously), and thus more desirable, when the possibility of satisfaction is unknown.

The temporal coupling of the outer G with the inner TL makes the Value of these compositions more challenging to characterize and decompose, and in general *may not be unique*. We begin with a formal characterization of the Value in this situation for the base-case predicate $\mathsf{G}(\mathsf{q}\,\mathsf{U}\,\mathsf{r})$.

Theorem 2. *For the predicate* $\mathsf{p} := \mathsf{G}\,(\mathsf{q}\;\mathsf{U}\;\mathsf{r})$ *the*

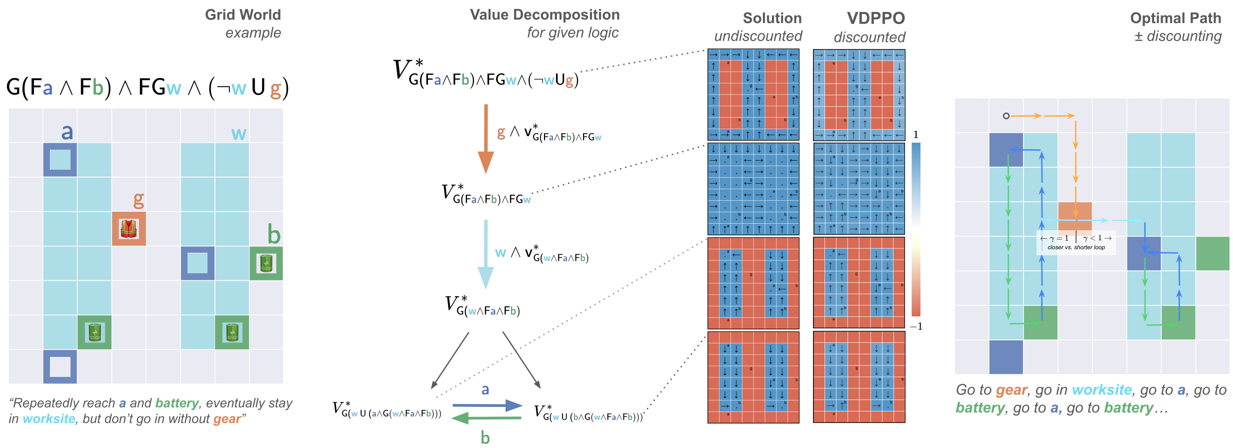

Fig. 3: **E.g. G(N-Until-Conjunction) Value Decomposition.** We illustrate the recursive decomposition result (Thm. 3), with a `GridWorld` example (left) for a given specification. The plots here are analogous to those of Fig. 2, with the DVG (center left), decomposed Values (center right), and optimal path (right). Note, the optimal path for the discounted case differs due to the subtle effect of discounting the Value associated with a G composition, which selects for shorter loops (Sec. VII-C).

> *corresponding Bellman Value satisfies*
>
> $$V^*[\mathsf{G}\,(\mathsf{q}\;\mathsf{U}\;\mathsf{r})](x) = V^*\left[\mathsf{q}\;\mathsf{U}\,(\mathsf{r}\wedge \mathsf{X}\mathsf{v}^*_{\mathsf{p}})\right](x).$$

This result demonstrates that the Value function associated with the predicate $\mathsf{G}\,(\mathsf{q}\;\mathsf{U}\;\mathsf{r})$ can be characterized recursively. Intuitively, one may consider this Value as a special $\mathcal{RA}$ Value that aims to reach an intersection of the target predicate r and its own satisfiable set (denoted by $\mathsf{v}^*_{\mathsf{p}}$) at the next step, and hence, maintain the ability to satisfy it again in the future. More generally, we may expand this result to the case involving a composition fo G with N-Until-Conjunctions, formalized in the following result.

> **Theorem 3.** *Given the set of coupled Bellman Values of length $J = |\mathcal{J}|$,*
>
> $$V^*_j(x) := V^*\left[\tilde{\mathsf{q}}_j\;\mathsf{U}\,(\tilde{\mathsf{r}}_j \wedge \mathsf{X}\mathsf{v}^*_{j+1})\right](x)$$
>
> *where $J+1 := 1$, $\tilde{\mathsf{q}}_j := \mathsf{q}_j \wedge (\mathsf{q}_{j+1} \vee \mathsf{r}_{j+1})$, and $\tilde{\mathsf{r}}_j := \mathsf{r}_j \wedge (\mathsf{q}_{j+1} \vee \mathsf{r}_{j+1})$, then $\forall j$, defined by*
>
> $$V^*\left[\mathsf{G}\left(\bigwedge_{j\in\mathcal{J}} (\mathsf{q}_j\;\mathsf{U}\;\mathsf{r}_j)\right)\right](x) = V^*_j(x).$$

This result allows us to consider the problem of recurrently reach-avoiding N tasks as a loop of N coupled $\mathcal{RA}_\ell$ Values. Note, in this case, the fixed iteration order is equivalent to any ordering given in Thm. 1 because of the infinite nature of the problem (see Appendix).

Although, these results appear like the previous decompositions, it is important to note that they are fundamentally different due to the implicit definition of the Value. These characterizations do *not* guarantee uniqueness or existence of the Value, and in continuous state spaces, they may be ill-defined. To certify the existence in certain scenarios (e.g. finite state spaces), we show in the Appendix that these Values are equivalent to the limit of finite recurrence, however, this is not generally a practical procedure.

Moreover, a straightforward application of the discounted $\mathcal{RA}$-BE yields a BE that is *not guaranteed to be contractive*, due to the appearance of the Value in both $(1-\gamma)$ and γ terms. To address these challenges, we propose a novel contractive Bellman Equation, which we call the $\mathcal{RA}$-Loop ($\mathcal{RA}_\ell$) BE, which is guaranteed to solve the family of $\mathsf{G}(\ldots)$ predicates in the limit of discounting.

> **Lemma 2.** *For the set of J Values defined in Thm. 3, let the $\mathcal{RA}_\ell$-BE be defined as*
>
> $$\mathcal{B}^\gamma_{\mathcal{RA}_\ell}[V_j] := (1-\gamma)\min\{\tilde{r}_j, \tilde{q}_j\} + \gamma \min\left\{\max\left\{\min\left\{\tilde{r}_j, V^+_{j+1}\right\}, V^+_j\right\}, \tilde{q}_j\right\}.$$
>
> *This is contractive such that $V^\gamma_j = \mathcal{B}^\gamma_{\mathcal{RA}_\ell}[V^\gamma_j]$ has a unique fixed point, satisfying $\lim_{\gamma\to 1} V^\gamma_j = V^*[\mathsf{G}(\bigwedge_{j\in\mathcal{J}}(\mathsf{q}_j\;\mathsf{U}\;\mathsf{r}_j))]$.*

Equipped with the $\mathcal{RA}_\ell$-BE, we can now tackle the problem of computing the Value function for the family of $\mathsf{G}(\ldots)$ predicates effectively.

D. A general result for a class of predicates

Here, we give the final decompositional result of the paper, combining several of the previous results. Note, we present this as a culmination of the different algebraic decompositions of the Value to certify the decomposition of a general class of TL predicates, including all of those involved in the work.

> **Theorem 4.** *For the predicate*
>
> $$\mathsf{p} := \left(\bigwedge_{i\in\mathcal{I}} (\mathsf{q}_i\;\mathsf{U}\;\mathsf{r}_i)\right) \wedge \mathsf{G}\left(\bigwedge_{j\in\mathcal{J}} (\mathsf{q}_j\;\mathsf{U}\;\mathsf{r}_j)\right) \wedge \mathsf{G}\mathsf{q}$$
>
> *the corresponding optimal Value satisfies*

$$V^*[\mathsf{p}](x) = V^*\left[\tilde{\mathsf{q}} \,\mathsf{U}\, \tilde{\mathsf{r}}\right](x) \textit{ where}$$

$$\tilde{\mathsf{r}} := \bigvee_i \left(\mathsf{r}_i \wedge \mathsf{v}^*_{\mathsf{p}^{-i}}\right), \qquad \tilde{\mathsf{q}} := \bigwedge_{k \in \mathcal{I} \times \mathcal{J}} \tilde{\mathsf{q}}_k \wedge \mathsf{q},$$

$$\mathsf{p}^{-i} := \bigwedge_{k \in \mathcal{I} \setminus \{i\}} (\mathsf{q}_k \,\mathsf{U}\, \mathsf{r}_k) \wedge \mathsf{G}\left(\bigwedge_{j \in \mathcal{J}} (\mathsf{q}_j \,\mathsf{U}\, \mathsf{r}_j)\right) \wedge \mathsf{G}\mathsf{q}.$$

Akin to previous results, Thm. 4 demonstrates that the given predicate p, involving the conjunction of N-Until predicates and the composition of G with N-Until predicates, may be rewritten as a single $\mathcal{RA}$ Value. The residual Value of this decomposition is the Value associated with the composition of G with N-Until predicates, and can thus be recursively decomposed with Thm. 3. See the Appendix for the complete proof.

VIII. Algorithm(s)

In this section, we introduce Value-Decomposition PPO, a variant of PPO that solves the Bellman value associated with the class of TL predicates in Sec. VII using the decomposed value graph (DVG). We also describe the tools required to generate the DVG and to solve it via dynamic programming for low-dimensional problems. A graphical overview is shown in Fig. 4, and all relevant code is provided in the Appendix.

`valtr`: Generating the DVG. We introduce `valtr`, a tool that converts a parsed temporal logic specification into the general predicate form of Thm. 4 by recursively applying standard TL rules. This representation is then transformed into the directed acyclic graph (DAG) of the DVG, where nodes correspond to predicates, negations, max, min, and value functions, and edges encode their dependencies. Cyclic G compositions are handled via a special node, enabling efficient parsing and transformation of arbitrary predicates into DVGs. See the Appendix for details.

Dynamic Programming with the DVG. With the DVG, one may compute the Value of a given predicate by performing a topological sort of the DAG and applying dynamic programming to compute the Value of each subformula in the correct order. This allows us to compute the dynamic programming solution for the low-d test cases given in Figs. 2 and 3.

VDPPO. Finally, we propose Value-Decomposition PPO (VDPPO), a special variant of PPO which solves the Bellman Value associated with the class of TL predicates in Sec. VII by using the DVG. In this method, we use a shared trunk for each decomposed Value in the DVG by embedding the node representations with a one-hot vector. Depending on the embedding value, the trunk is trained with the corresponding discounted $\mathcal{A}$-BE, $\mathcal{RA}$-BE or $\mathcal{RA}_\ell$-BE by using the appropriate BE to compute the advantage estimate. Note, by definition this requires boot-strapping the current Value estimate for each node, which is represented by the feedback loop in Fig. 4. The policy also uses a shared trunk with the embedded value and is trained with the standard PPO objective, using the advantage estimate corresponding to the embedding. This allows us to leverage the decomposed structure of the Value functions to efficiently learn policies that satisfy complex TL specifications without sequentially approximating the Value. See the Appendix for further details.

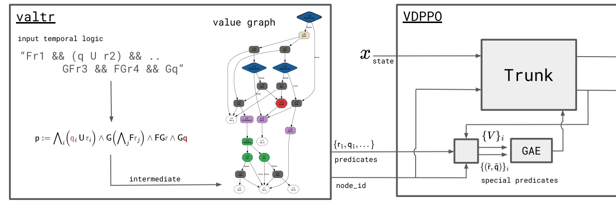

Fig. 4: **Graphical Depiction of Algorithms.**

IX. Simulation Results

To better understand the performance of VDPPO, we design simulation experiments to answer the following research questions:

- **(Q1):** Does value decomposition help with satisfying more complex temporal logic specifications (in both breadth and depth)?
- **(Q2):** Does value decomposition help with scaling to multiple agents?
- **(Q3):** Can VDPPO scale to more complex dynamics?

Additional ablation studies are provided in the Appendix.

A. Setup

Environments. We evaluate on four simulated domains: `DoubleInt` (toy double integrator environment to focus on TL challenges), `Herding` (a team of herders collaborates to herd multiple targets to a designated location while avoiding obstacles), `Delivery` (agents must continuously pick up and deliver packages to a special agent while avoiding collisions with each other and static obstacles), and `Manipulator` (a robotic arm interacts with a cube and a drawer as specified by TL formulas).

Baselines. We compare VDPPO with other model-free methods that can solve TL specifications with black-box dynamics. These include LCRL [69], a deep RL method that solves TL tasks by augmenting the state space with an automata representation of the TL formula, and an extension of Model Predictive Path Integral (MPPI) [70] to tackle TL problems [71], which we denote TL-MPPI. For each environment, LCRL and VDPPO are run for the same number of update steps, while for TL-MPPI we follow the hyperparameters chosen in [71].

Evaluation criteria. Performance is measured by success rate on finite-horizon TL satisfaction; we additionally report satisfaction rates of individual subformulas. All methods are trained with three seeds and evaluated on 256 initial conditions.

B. Results

(Q1): Value decomposition improves scalability with TL complexity. We study two TL families of increasing complexity in a single-agent double-integrator environment. Breadth specifications combine a safety constraint with an increasing number of unordered `Finally` goals, while depth specifications contain nested `Finally` operators enforcing a fixed order. Results are shown in Fig. 9.

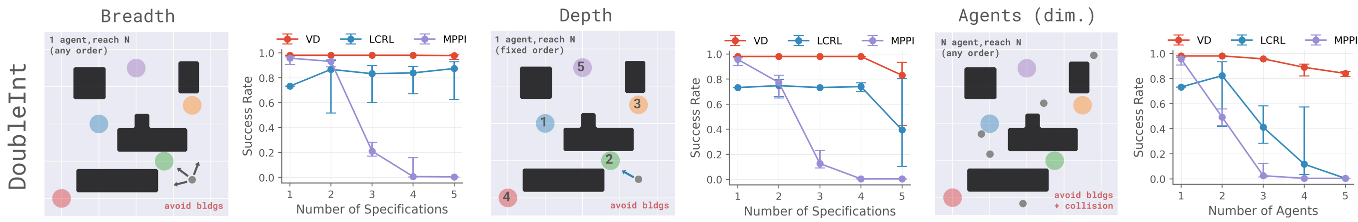

Fig. 5: **Performance scaling with TL complexity.** Value decomposition enables VDPPO to better scale by tackling smaller problems.

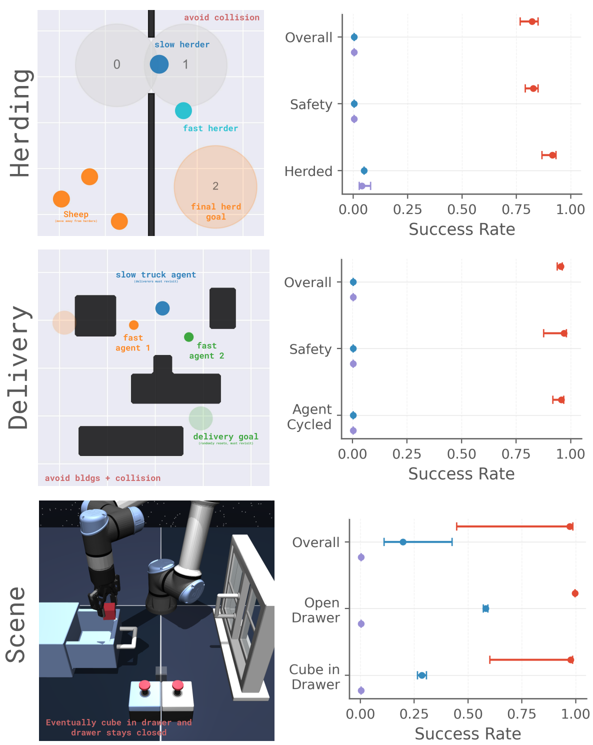

Fig. 6: **Complex high-dimensional tasks**. VDPPO greatly outperforms baseline methods on more complex tasks.

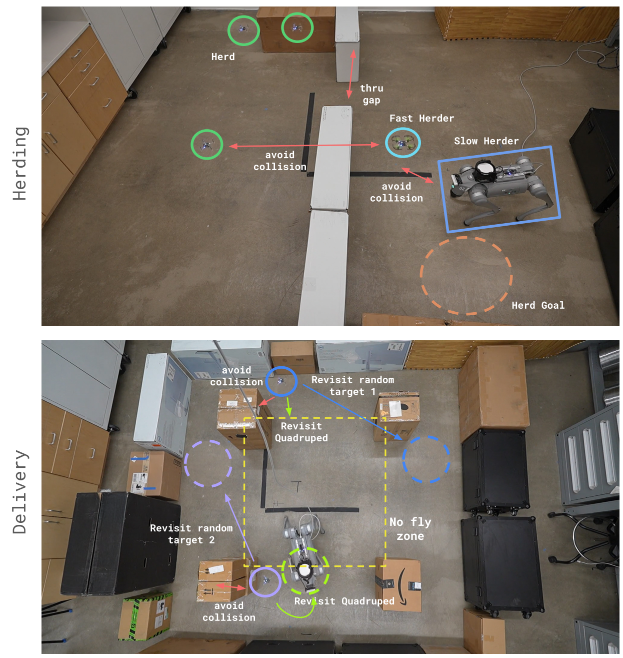

Fig. 7: **Hardware Overview for `Herding` and `Delivery` Tasks**

All methods solve the singular specification but degrade as the number of specifications increases. VDPPO consistently outperforms both baselines as the complexity of the TL specifications increases in both breadth and depth, demonstrating the effectiveness of value decomposition in handling complex TL tasks. This is particularly true in the depth case, where both baselines achieve $\leq 40\%$ success rate for a depth of $n = 5$. This is because the probability of satisfying nested TL specifications by luck decreases exponentially with depth, making it difficult for non-decompositional methods to learn effective policies.

(Q2): Value decomposition strongly helps with increasing number of agents. Compared to the Breadth plot where we only increase the number of specifications, we scale both the number of agents and the number of specifications simultaneously and show the results in Fig. 9. Increasing the number of agents increases the action dimension, which increases the difficulty of exploration. This degrades the performance of all methods. However, VDPPO is least impacted by this and is the only method that solves the problem with 5 agents.

(Q3): VDPPO shines on problems with difficult dynamics. We now consider more challenging problems, either due to complex interactions with uncontrolled agents (`Herding`), needing to collaborate (`Delivery`), or complex dynamics (`Manipulator`) and show the results in Fig. 6. In all three tasks, VDPPO achieves the highest success rate by a significant margin. See the Appendix for ablations.

X. Hardware Results

Lastly, we perform hardware experiments corresponding to the `Herding` and `Delivery` environments using a swarm of Crazyflie (CF) drones collaborating with the Unitree Go2 to demonstrate the ability of VDPPO to solve complex task specifications in high-dimensional real-world settings with heterogeneous collaboration. See Fig. 7 for an overview.

A. *`Herding`*

In this experiment, we consider a team of one CF and the Go2 tasked with herding three "sheep" CFs through a narrow gap to a target location while avoiding obstacles and collisions. The sheep CFs have a fixed nominal policy, using the softmin to drive them away from the nearest obstacle or agent, and thus will move only when approached.

The TL specification for the task is given by,

$$\mathsf{p}_{\text{herding}} := \mathsf{G}(\neg \mathsf{c}) \wedge \mathsf{F}(\mathsf{r}_0 \wedge \mathsf{F}\mathsf{r}_1) \wedge \mathsf{FG}(\mathsf{r}_h),$$

where c denotes collisions, r_0 the herd reaching the pre-gate region, r_1 passage through the gate, and r_h arrival at the target.

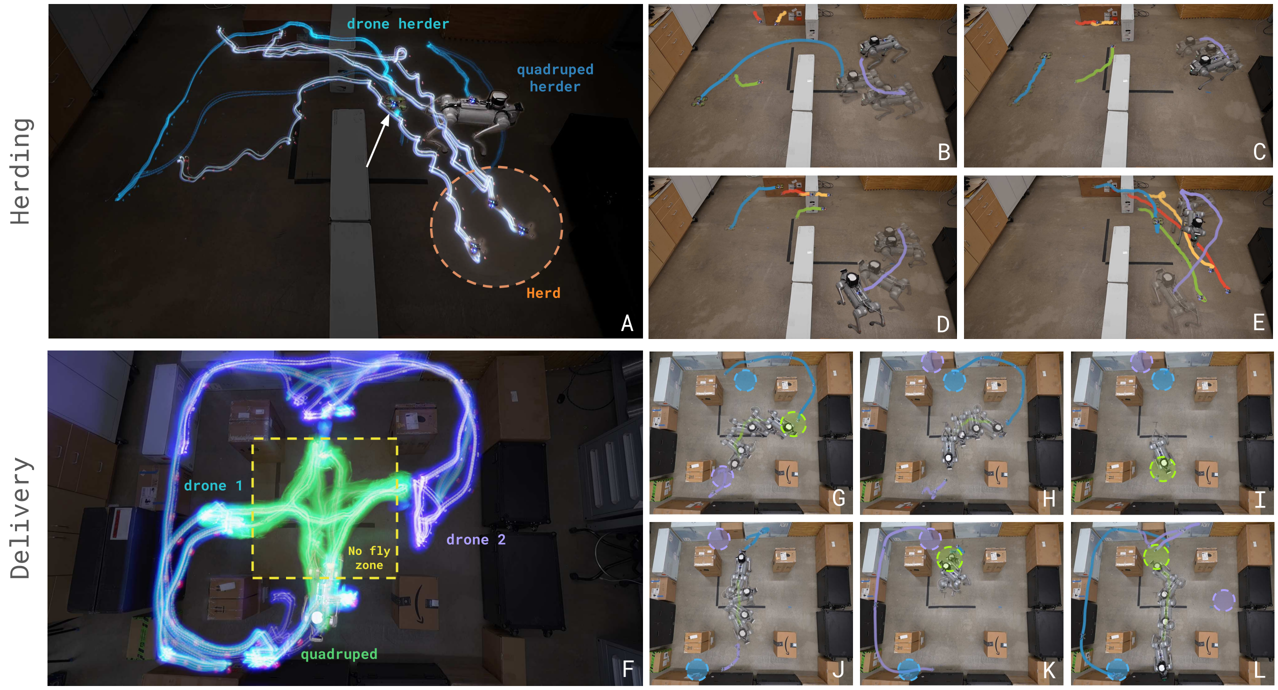

Fig. 8: **Trajectory snapshots from `Herding` and `Delivery` hardware tasks**. We show a long-exposure photo (left), and stills from independent times (right), with depictions corresponding to those of the overview in Fig. 7.

This encodes a sequence of reach–avoid objectives followed by a reach–stabilize objective requiring indefinite herding.

The herders (CF and Go2) are initialized opposite the gap from the herd and have asymmetric dynamics, with the Go2 moving more slowly. To satisfy the specification, the herders must coordinate to pass through the gap, collect the herd, and guide it to the target while avoiding obstacles. We train a VDPPO policy using the DVG and deploy it on hardware, where the agents adapt online to real-time state feedback.

Ultimately, we observe that the CF and Go2 **learn to divide the labor** of the task such that the CF passes through the gap to gather the agents (Fig.8.B), while the Go2 waits to receive on the herding side (Fig.8.C). When the herd passes through the narrow gap, the Go2 initially moves out of the way (Fig.8.C) and then transitions to providing support, rapidly shifting position to block the Herd from distributing across the new space (Fig.8.E). This behavior is entirely emergent and demonstrates the wide-ranging ability of VDPPO to solve complex tasks automatically.

B. `Delivery`

In this experiment, we consider a team of two CFs and the Go2 tasked with recurrently visiting agent-specific target locations and recurrently revisiting the Go2 agent (to model package delivery and resupply), while avoiding building obstacles, collisions, and a "no fly zone" (for the CFs).

The TL specification for the task is given by,

$$\mathsf{p}_{\text{delivery}} := \bigwedge_i \mathsf{GF}(\mathsf{r}_i) \wedge \bigwedge_i \mathsf{GF}(\mathsf{rs}_i) \wedge \mathsf{G}\neg\mathsf{ac} \wedge \mathsf{G}\neg\mathsf{ob} \wedge \mathsf{G}\neg\mathsf{nf}$$

where the predicate r_i captures CF i visiting target i, rs_i captures CF i visiting the Go2 (resupplying), ac captures aerial collision, ob captures obstacle collision, and nf captures the no-fly-zone (for the CFs only). Here, the task logic is dominated by GF, and hence is largely solved with the $\mathcal{RA}_\ell$-BE.

In this environment, the CF targets jump to a new random location after an agent has visited it, requiring a policy that is conditioned to various target locations. The real difficulty of this problem arises in the tightness of the layout; the obstacles confine the Go2 to the central area where the CFs are not allowed to fly (modeling a busy intersection), yet they must visit one another to "resupply". We again implement VDPPO to learn a policy to solve the complex task and deploy it live.

Ultimately, we observe sophisticated coordination between the three agents to **distribute the difficulty of the task evenly**. Namely, as the CFs move around the outskirts of the arena, avoiding one another carefully but not too cautiously (Fig.8.L), the Go2 anticipates their movements, moving between each of the agents (Fig.8.G-I) to be in position to resupply them as close to their target as possible. This complex collaboration generated by VDPPO allows the agents to rapidly meet deliveries and resupply without crashing at all.

XI. Conclusion

In this work, we propose a novel approach to solving the Bellman Value associated with complex temporal specifications via decomposition. Namely, we demonstrate that for a large class of TL predicates, the corresponding Bellman Value may be decomposed into a graph of Values connected by a set of "atomic" Bellman equations. With this perspective, we propose VDPPO that is shown to solve optimal policies in complex tasks well beyond existing methods. This work highlights a novel and powerful approach to tackling complex task logic for real-world autonomy.

APPENDIX

Contents

Useful Properties and Notation

We give here properties and notations for simplifying the following proofs. For a given action sequence α,

$$\alpha := (a_1, a_2, \dots) \in \mathbb{A} := \mathcal{A}^{\mathbb{N}}$$

let a portion beginning at i and ending at j be written

$$\alpha_{i:j} := (a_i, \dots a_j).$$

Moreover, for a trajectory ξ_x^α,

$$\xi_x^\alpha := (x, x_1, \dots) \in \mathbb{X} := \mathcal{X}^{\mathbb{N}},$$

where $x_{i+1} = f(x_i, \alpha_i)$, it follows then that for α divided into $\alpha_{t^-} := \alpha_{1:t}$ & $\alpha_{t^+} := \alpha_{t+1:\infty}$,

$$\xi_x^\alpha = \xi_y^{\alpha_{t^+}},$$

where $y = \xi_x^{\alpha_{t^-}}(t)$. We then have the following result corresponding to the decomposition of a controlled trajectory, which will be used ubiquitously.

Lemma 3. *Let $\mathcal{X}$ s.t. $|\mathcal{X}| < \infty$. Then for $t \in \mathbb{N}$, $\alpha \in \mathbb{A}$, $\xi_x^\alpha \in \mathbb{X}$, and, $x \in \mathcal{X}$,*

$$\max_\alpha \max_t f(\xi_x^\alpha, t) = \max_t \max_{\alpha_{t^-}} \max_{\alpha_{t^+}} f(\xi^{\alpha_{t^+}}_{\xi_x^{\alpha_{t^-}}(t)}, t).$$

A. More Related Works

We here give a slightly more expanded description of the related works compared to the main text. We refer the reader to [5] for additional discussion of many of these works.

Reinforcement Learning with TL Objectives. Many works have explored ways to optimize objectives that encode TL specifications [6, 7, 8, 9, 10, 11, 72] (or conversely learn TL specifications from agent behavior [73]). One line of such works uses Non-Markovian-Reward Decision Processes (NMRDPs), which allow for history-dependent rewards [2, 13, 14, 15, 74]. Other works optimize the quantitative semantics associated with an STL objective, approximating the maximums and minimums in a sum-of-discounted rewards fashion, which are then solved with traditional methods [16, 17], or otherwise encoding TL objectives through expectations [18]. Several other methods also exist that attempt to optimize general objective functions using non-traditional Bellman equations [19, 20, 21] or handle discounted sums of multiple rewards or penalties [22, 23, 24]. We also refer the reader to [75] for an approach that proceeds by composing learned sub-tasks into higher level ones using an additional planning algorithms rather than breaking a high-level task down into subtasks. By contrast to most of these previous approaches, our approach proceeds by decomposition of a TL-specified problem in an exact manner. Specifically, we decompose the value function associated with a quantitative semantic for a TL predicate into value functions associated with simpler objectives. These simpler objectives are then solved by leveraging powerful recent Hamilton-Jacobi Reachability (HJR) methods. (Note that these decompositions of the value functions are fundamentally different from decompositions of the quantitative semantics themselves.) This approach allows one to avoid approximations of the objective function or issues associated with sparsity of long-horizon rewards, which commonly afflict the previous methods.

Constrained, Multi-Objective, and Goal-Conditioned RL A number of techniques in RL have arisen to handle constraints or multiple goals. Constrained MDPs (CMDPs) attempt to maximize sums of discounted rewards subject to a safety or liveness condition, which is often handled via a Lagrangian term in the objective function [25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38]. For CMDPs, the Lagrangian term involved typically requires substantial tuning for desired behavior, severly limiting its use for satisfying general TL tasks. Multi-objective RL techniques, by contrast Pareto-optimize multiple sums of discounted rewards [39, 40, 41, 42, 43, 44, 45]. This allows users to balance multiple objectives, but generally are not built for handling TL-like specifications. Goal-conditioned RL, by contrast, simultaneously learns policies for a range of possible task specifications [46, 47, 48, 49, 50, 51, 52, 49, 50, 53]. At the time of deployment, a user can then decide which specification is most appropriate. This is fundamentally different from TL tasks, where all specifications must be satisfied.

Hamilton-Jacobi Reachability Hamilton-Jacobi Reachability (HJR) methods were initially designed to solve value functions associated with "reach", "avoid", or "reach-avoid" problems using traditional dynamic programming for continuous space and times [3, 4]. The objectives for these tasks are precisely the quantitative semantics for eventually, never, and until predicates. HJR approaches have recently been adapted to solve these same problems in RL settings, with exciting performance [55, 56, 57, 58, 59, 60]. Our work builds on such advancements, using the RL algorithms developed by these building blocks to accomplish higher-level tasks.

B. Temporal Logic

In this section, we give further background on the temporal logic used in the main text. We begin with the logical definitions of the operators $\vee, \wedge, \neg, \mathsf{X}, \mathsf{F}, \mathsf{G}, \mathsf{U}$, alternatively defined by their robustness metric in the main text.

Definition 4. *Let p, p' be predicates, $\xi_x \in \mathbb{X}$ a trajectory beginning at $x \in \mathcal{X}$, and $t \in \mathbb{N}$ a starting time. The relation $(\xi_x, t) \models \mathsf{p}$ is defined as follows,*

$$\begin{aligned}
(\xi_x, t) \models \mathsf{r}_i &\iff r_i(\xi_x(t)) \geq 0, \\
(\xi_x, t) \models \neg\mathsf{p} &\iff (\xi_x, t) \not\models \mathsf{p}, \\
(\xi_x, t) \models \mathsf{p} \wedge \mathsf{p}' &\iff (\xi_x, t) \models \mathsf{p} \text{ and } (\xi_x, t) \models \mathsf{p}', \\
(\xi_x, t) \models \mathsf{p} \vee \mathsf{p}' &\iff (\xi_x, t) \models \mathsf{p} \text{ or } (\xi_x, t) \models \mathsf{p}', \\
(\xi_x, t) \models \mathsf{X}\mathsf{p} &\iff (\xi_x, t+1) \models \mathsf{p}, \\
(\xi_x, t) \models \mathsf{F}\mathsf{p} &\iff \exists \tau \geq t \text{ s.t. } (\xi_x, \tau) \models \mathsf{p}, \\
(\xi_x, t) \models \mathsf{G}\mathsf{p} &\iff \forall \tau \geq t,\ (\xi_x, \tau) \models \mathsf{p}, \\
(\xi_x, t) \models \mathsf{p} \,\mathsf{U}\, \mathsf{p}' &\iff \exists \tau \geq t \text{ s.t. } (\xi_x, \tau) \models \mathsf{p}' \text{ and } \\
&\qquad \forall \kappa \in [t, \tau],\ (\xi_x, \kappa) \models \mathsf{p}.
\end{aligned}$$

From these definitions, we may certify a few equivalence relations for rearranging certain combinations of operators, which will later prove to be useful. Note, for the interested reader all of the following equivalences may be automatically verified with the tool Spot [76].

Lemma 4.

$$(\mathsf{q}_1 \,\mathsf{U}\, \mathsf{r}_1) \wedge (\mathsf{q}_2 \,\mathsf{U}\, \mathsf{r}_2) \equiv (\mathsf{q}_1 \wedge \mathsf{q}_2) \,\mathsf{U}\, \big((\mathsf{r}_1 \wedge \mathsf{q}_2 \,\mathsf{U}\, \mathsf{r}_2) \vee (\mathsf{r}_2 \wedge \mathsf{q}_1 \,\mathsf{U}\, \mathsf{r}_1)\big)$$

Proof. We show this via double entailment.

1. LHS $\models$ RHS:
Suppose $\sigma, 0 \models \mathsf{q}_1 \,\mathsf{U}\, \mathsf{r}_1 \wedge \mathsf{q}_2 \,\mathsf{U}\, \mathsf{r}_2$. Then,

1) Since $\sigma, 0 \models \mathsf{q}_1 \,\mathsf{U}\, \mathsf{r}_1$, there exists $t_1 \geq 0$ such that $\sigma, t_1 \models \mathsf{r}_1$, and for all $0 \leq k < t_1$, $\sigma, k \models \mathsf{q}_1$.
2) Since $\sigma, 0 \models \mathsf{q}_2 \,\mathsf{U}\, \mathsf{r}_2$, there exists $t_2 \geq 0$ such that $\sigma, t_2 \models \mathsf{r}_2$, and for all $0 \leq k < t_2$, $\sigma, k \models \mathsf{q}_2$.

Let $t = \min(t_1, t_2)$. Since $\sigma, k \models \mathsf{q}_1$ and $\sigma, k \models \mathsf{q}_2$ for all $0 \leq k < t$, we have $\sigma, k \models \mathsf{q}_1 \wedge \mathsf{q}_2$ for all $0 \leq k < t$.

We now show that the goal is reached at time t.

- ($t_1 \leq t_2$): Then, $t = t_1$, and $\sigma, t \models \mathsf{r}_1$. Since $t_2 \geq t_1$ and $\sigma, k \models \mathsf{q}_2$ for all $t \leq k < t_2$, we have $\sigma, t \models \mathsf{q}_2 \,\mathsf{U}\, \mathsf{r}_2$. Hence, $\sigma, t \models \mathsf{r}_1 \wedge \mathsf{q}_2 \,\mathsf{U}\, \mathsf{r}_2$.
- ($t_2 < t_1$): Then, $t = t_2$, and $\sigma, t \models \mathsf{r}_2$. Since $t_1 > t_2$ and $\sigma, k \models \mathsf{q}_1$ for all $t \leq k < t_1$, we have $\sigma, t \models \mathsf{q}_1 \,\mathsf{U}\, \mathsf{r}_1$. Hence, $\sigma, t \models \mathsf{r}_2 \wedge \mathsf{q}_1 \,\mathsf{U}\, \mathsf{r}_1$.

Thus, $\sigma, 0 \models (\mathsf{q}_1 \wedge \mathsf{q}_2) \,\mathsf{U}\, \big((\mathsf{r}_1 \wedge \mathsf{q}_2 \,\mathsf{U}\, \mathsf{r}_2) \vee (\mathsf{r}_2 \wedge \mathsf{q}_1 \,\mathsf{U}\, \mathsf{r}_1)\big)$.

2. RHS $\models$ LHS:
Suppose $\sigma, 0 \models (\mathsf{q}_1 \wedge \mathsf{q}_2) \,\mathsf{U}\, \big((\mathsf{r}_1 \wedge \mathsf{q}_2 \,\mathsf{U}\, \mathsf{r}_2) \vee (\mathsf{r}_2 \wedge \mathsf{q}_1 \,\mathsf{U}\, \mathsf{r}_1)\big)$. Then, there exists $t \geq 0$ such that

- $\sigma, t \models (\mathsf{r}_1 \wedge \mathsf{q}_2 \,\mathsf{U}\, \mathsf{r}_2) \vee (\mathsf{r}_2 \wedge \mathsf{q}_1 \,\mathsf{U}\, \mathsf{r}_1)$
- For all $0 \leq k < t$, $\sigma, k \models \mathsf{q}_1 \wedge \mathsf{q}_2$.

We now split into two cases.

1) ($\sigma, t \models \mathsf{r}_1 \wedge \mathsf{q}_2 \,\mathsf{U}\, \mathsf{r}_2$):
   - $\sigma, t \models \mathsf{r}_1$
   - Since $\sigma, k \models \mathsf{q}_1$ for all $0 \leq k < t$, we have $\sigma, 0 \models \mathsf{q}_1 \,\mathsf{U}\, \mathsf{r}_1$.
   - There exists $t_2 \geq t$ such that $\sigma, t_2 \models \mathsf{r}_2$, and $\sigma, k \models \mathsf{q}_2$ for all $t \leq k < t_2$.
   - Since $\sigma, k \models \mathsf{q}_2$ for all $0 \leq k < t_2$, we have $\sigma, 0 \models \mathsf{q}_2 \,\mathsf{U}\, \mathsf{r}_2$.
   - Thus, $\sigma, 0 \models \mathsf{q}_1 \,\mathsf{U}\, \mathsf{r}_1 \wedge \mathsf{q}_2 \,\mathsf{U}\, \mathsf{r}_2$.
2) ($\sigma, t \models \mathsf{r}_1 \wedge \mathsf{q}_2 \,\mathsf{U}\, \mathsf{r}_2$): The reasoning is symmetric to the previous case, yielding $\sigma, 0 \models \mathsf{q}_1 \,\mathsf{U}\, \mathsf{r}_1 \wedge \mathsf{q}_2 \,\mathsf{U}\, \mathsf{r}_2$.

Thus, $\sigma, 0 \models \mathsf{q}_1 \,\mathsf{U}\, \mathsf{r}_1 \wedge \mathsf{q}_2 \,\mathsf{U}\, \mathsf{r}_2$.

Since we have shown both directions, the equivalence holds. $\square$

Lemma 5.

$$\mathsf{p} := \bigwedge_{i=1}^{n} (\mathsf{q}_i \,\mathsf{U}\, \mathsf{r}_i) \equiv \Big(\bigwedge_{i=1}^{n} \mathsf{q}_i\Big) \,\mathsf{U}\, \Big(\bigvee_{i=1}^{n} \big(\mathsf{r}_i \wedge \mathsf{p}^{-i}\big)\Big)$$

where $\mathsf{p}^{-i} := \bigwedge_{j=1, j\neq i}^{n} (\mathsf{q}_j \,\mathsf{U}\, \mathsf{r}_j)$.

Proof. We prove this using induction on n.

Base Case ($n = 2$): This is exactly the previous lemma 4.

Inductive Step: Assume the statement holds for $n = k$, i.e.,

$$\bigwedge_{i=1}^{k} (\mathsf{q}_i \,\mathsf{U}\, \mathsf{r}_i) \equiv \underbrace{\Big(\bigwedge_{i=1}^{k} \mathsf{q}_i\Big)}_{:=\tilde{\mathsf{q}}} \,\mathsf{U}\, \underbrace{\Big(\bigvee_{i=1}^{k} \big(\mathsf{r}_i \wedge \bigwedge_{j=1, j\neq i}^{n} (\mathsf{q}_j \,\mathsf{U}\, \mathsf{r}_j)\big)\Big)}_{:=\tilde{\mathsf{r}}}.$$

We need to show it holds for $n = k+1$.

$$\begin{aligned}\bigwedge_{i=1}^{k+1}(\mathsf{q}_i \,\mathsf{U}\, \mathsf{r}_i) &= \Big(\bigwedge_{i=1}^{k}(\mathsf{q}_i \,\mathsf{U}\, \mathsf{r}_i)\Big) \wedge (\mathsf{q}_{k+1} \,\mathsf{U}\, \mathsf{r}_{k+1}) \\ &\equiv \tilde{\mathsf{q}} \,\mathsf{U}\, \tilde{\mathsf{r}} \wedge (\mathsf{q}_{k+1} \,\mathsf{U}\, \mathsf{r}_{k+1}) \\ &\equiv (\tilde{\mathsf{q}} \wedge \mathsf{q}_{k+1}) \,\mathsf{U} \\ &\quad\; ((\tilde{\mathsf{r}} \wedge \mathsf{q}_{k+1} \,\mathsf{U}\, \mathsf{r}_{k+1}) \vee (\mathsf{r}_{k+1} \wedge \tilde{\mathsf{q}} \,\mathsf{U}\, \tilde{\mathsf{r}}))\end{aligned}$$

Note that $\tilde{\mathsf{q}} \wedge \mathsf{q}_{k+1} = \bigwedge_{i=1}^{k+1} \mathsf{q}_i$. For the first part,

$$\begin{aligned}\tilde{\mathsf{r}} \wedge \mathsf{q}_{k+1} \,\mathsf{U}\, \mathsf{r}_{k+1} &= \bigvee_{i=1}^{k}\Big(\mathsf{r}_i \wedge \bigwedge_{j=1, j\neq i}^{k} (\mathsf{q}_j \,\mathsf{U}\, \mathsf{r}_j)\Big) \wedge \mathsf{q}_{k+1} \,\mathsf{U}\, \mathsf{r}_{k+1} \\ &= \bigvee_{i=1}^{k}\Big(\mathsf{r}_i \wedge \bigwedge_{j=1, j\neq i}^{k} (\mathsf{q}_j \,\mathsf{U}\, \mathsf{r}_j) \wedge \mathsf{q}_{k+1} \,\mathsf{U}\, \mathsf{r}_{k+1}\Big) \\ &= \bigvee_{i=1}^{k}\Big(\mathsf{r}_i \wedge \bigwedge_{j=1, j\neq i}^{k+1} (\mathsf{q}_j \,\mathsf{U}\, \mathsf{r}_j) \wedge \mathsf{q}_{k+1} \,\mathsf{U}\, \mathsf{r}_{k+1}\Big).\end{aligned}$$

For the second part,

$$\begin{aligned}\mathsf{r}_{k+1} \wedge \tilde{\mathsf{q}} \,\mathsf{U}\, \tilde{\mathsf{r}} &= \mathsf{r}_{k+1} \wedge \bigwedge_{i=1}^{k}(\mathsf{q}_i \,\mathsf{U}\, \mathsf{r}_i), \\ &= \mathsf{r}_{k+1} \wedge \bigwedge_{j=1, j\neq k+1}^{k+1} (\mathsf{q}_j \,\mathsf{U}\, \mathsf{r}_j).\end{aligned}$$

Combining these two parts completes the inductive step:

$$\bigvee_{i=1}^{k+1}\Big(\mathsf{r}_i \wedge \bigwedge_{j=1, j\neq i}^{k+1} (\mathsf{q}_j \,\mathsf{U}\, \mathsf{r}_j)\Big).$$

Since the base case and inductive step hold, the statement holds for all $n \geq 2$. □

Corollary 1.

$$\mathsf{p} := \bigwedge_{i=1}^{n}(\mathsf{q}_i \,\mathsf{U}\, \mathsf{r}_i) \wedge \mathsf{G}\mathsf{q} \equiv \Big(\bigwedge_{i=1}^{n} \mathsf{q}_i \wedge \mathsf{q}\Big) \,\mathsf{U}\, \Big(\bigvee_{i=1}^{n} (\mathsf{r}_i \wedge \mathsf{p}^{-i})\Big)$$

where $\mathsf{p}^{-i} := \bigwedge_{j=1, j\neq i}^{n}(\mathsf{q}_j \,\mathsf{U}\, \mathsf{r}_j) \wedge \mathsf{G}\mathsf{q}$.

Proof. It suffices to show that $\mathsf{G}\mathsf{q} = \mathsf{q} \,\mathsf{U}\, \tilde{\mathsf{r}}$ where $\tilde{\mathsf{r}} = \mathsf{G}\mathsf{q}$. This follows directly from the definition of G and U,

$$\begin{aligned}\sigma, 0 \models \mathsf{G}\mathsf{q} &\iff \forall t \geq 0, \sigma, t \models \mathsf{q} \\ &\iff \exists t' \geq 0 \text{ s.t. } \sigma, t' \models \mathsf{G}\mathsf{q} \\ &\qquad\quad \text{and } \forall 0 \leq t < t', \sigma, t \models \mathsf{q} \\ &\iff \sigma, 0 \models \mathsf{q} \,\mathsf{U}\, \tilde{\mathsf{r}}.\end{aligned}$$

□

Additionally, we can show this kind of rearrangement for the $\mathsf{G}\mathsf{U}$ composition as well, given by the following result.

Lemma 6.

$$\mathsf{G}(\mathsf{q} \,\mathsf{U}\, \mathsf{r}) \equiv \mathsf{q} \,\mathsf{U}\, (\mathsf{r} \wedge \mathsf{X}\mathsf{G}(\mathsf{q} \,\mathsf{U}\, \mathsf{r}))$$

Proof. We show this via double entailment.

1. (LHS $\models$ RHS) Suppose $\sigma, 0 \models \mathsf{G}(\mathsf{q} \,\mathsf{U}\, \mathsf{r})$.

- For all $t \geq 0$, there exists $s_t \geq t$ such that $\sigma, s_t \models \mathsf{r}$ and $\forall 0 \leq t' < s_t, \sigma, t' \models \mathsf{q}$. In particular, for $t = 0$, there exists $s_0 \geq 0$ such that $\sigma, s_0 \models \mathsf{r}$.
- Since $\mathsf{G}(\mathsf{q}\,\mathsf{U}\,\mathsf{r})$ is a tail property, we have $\sigma, s_0 + 1 \models \mathsf{G}(\mathsf{q}\,\mathsf{U}\,\mathsf{r})$.
- Thus, $\sigma, s_0 \models \mathsf{r} \wedge \mathsf{XG}(\mathsf{q}\,\mathsf{U}\,\mathsf{r})$.
- Hence, $\sigma, 0 \models \mathsf{q}\,\mathsf{U}\,(\mathsf{r} \wedge \mathsf{XG}(\mathsf{q}\,\mathsf{U}\,\mathsf{r}))$.

2. (RHS $\models$ LHS) Suppose $\sigma, 0 \models \mathsf{q}\,\mathsf{U}\,(\mathsf{r} \wedge \mathsf{XG}(\mathsf{q}\,\mathsf{U}\,\mathsf{r}))$.
   - By definition of U, there exists $t_0 \geq 0$ such that $\sigma, t_0 \models \mathsf{r} \wedge \mathsf{XG}(\mathsf{q}\,\mathsf{U}\,\mathsf{r})$ s.t. $\forall 0 \leq t < t_0, \sigma, t \models \mathsf{q}$.
   - The conjunction implies that $\sigma, t_0 + 1 \models \mathsf{G}(\mathsf{q}\,\mathsf{U}\,\mathsf{r})$.
   - Since $\mathsf{G}(\mathsf{q}\,\mathsf{U}\,\mathsf{r})$ is a tail property, this implies that $\sigma, 0 \models \mathsf{G}(\mathsf{q}\,\mathsf{U}\,\mathsf{r})$.

Since we have shown both directions, the equivalence holds. □

Next, we may extend this to the multi-Until case, in order to capture the behavior of multiple recurrent Until operators. Notably, in this case, the order does not matter, as all must be satisfied infinitely often. This is formalized in the following result.

Lemma 7. *Given*

$$\mathsf{p} := \mathsf{G}\big((\mathsf{q}_1\,\mathsf{U}\,\mathsf{r}_1) \wedge (\mathsf{q}_2\,\mathsf{U}\,\mathsf{r}_2)\big),$$

it holds that

$$\begin{aligned}\mathsf{p} &\equiv \tilde{\mathsf{q}}_1\,\mathsf{U}\left(\tilde{\mathsf{r}}_1 \wedge \big(\tilde{\mathsf{q}}_2\,\mathsf{U}\,(\tilde{\mathsf{r}}_2 \wedge \mathsf{p})\big)\right)\\ &\equiv \tilde{\mathsf{q}}_2\,\mathsf{U}\left(\tilde{\mathsf{r}}_2 \wedge \big(\tilde{\mathsf{q}}_1\,\mathsf{U}\,(\tilde{\mathsf{r}}_1 \wedge \mathsf{p})\big)\right),\end{aligned}$$

where $\tilde{\mathsf{q}}_i := \mathsf{q}_i \wedge (\mathsf{q}_j \vee \mathsf{r}_j)$, $\tilde{\mathsf{r}}_i := \mathsf{r}_i \wedge (\mathsf{q}_j \vee \mathsf{r}_j)$.

Proof. We show this via double entailment. For brevity, let $\mathsf{w}_1 := (\mathsf{q}_1 \vee \mathsf{r}_1)$, $\mathsf{w}_2 := (\mathsf{q}_2 \vee \mathsf{r}_2)$.

1. (LHS $\models$ RHS) Assume $\sigma, 0 \models \mathsf{p}$.
   - For all $t \geq 0$, $\sigma, t \models (\mathsf{q}_1\mathsf{U}\mathsf{r}_1) \wedge (\mathsf{q}_2\mathsf{U}\mathsf{r}_2)$. Choose $k_1 \geq 0$ with $\sigma, k_1 \models \mathsf{r}_1$ and $\sigma, t \models \mathsf{q}_1$ for $t < k_1$. Then $\sigma, t \models \mathsf{w}_2$ for $t \leq k_1$, so $\sigma, t \models \mathsf{q}_1 \wedge \mathsf{w}_2$ for $t < k_1$.
   - From $\sigma, k_1 \models \mathsf{q}_2\mathsf{U}\mathsf{r}_2$, choose $k_2 \geq k_1$ with $\sigma, k_2 \models \mathsf{r}_2$ and $\sigma, t \models \mathsf{q}_2$ for $k_1 \leq t < k_2$. Since p holds globally, $\sigma, t \models \mathsf{w}_1$ on $[k_1, k_2]$ and $\sigma, k_2 \models \mathsf{p}$.

   Thus $\sigma, k_1 \models \tilde{\mathsf{q}}_2\,\mathsf{U}\,(\tilde{\mathsf{r}}_2 \wedge \mathsf{p})$, so $\sigma, 0$ satisfies the RHS.

2. (RHS $\models$ LHS) Assume $\sigma, 0$ satisfies the RHS.
   - There exists $k_1 \geq 0$ with $\sigma, k_1 \models \tilde{\mathsf{r}}_1 \wedge \Psi$ and $\sigma, t \models \mathsf{q}_1 \wedge \mathsf{w}_2$ for $t < k_1$, where

   $$\Psi := \tilde{\mathsf{q}}_2\,\mathsf{U}\,(\tilde{\mathsf{r}}_2 \wedge \mathsf{p}).$$

   - From Ψ there exists $k_2 \geq k_1$ with $\sigma, k_2 \models \tilde{\mathsf{r}}_2 \wedge \mathsf{p}$ and $\sigma, t \models \mathsf{q}_2 \wedge \mathsf{w}_1$ for $k_1 \leq t < k_2$.

Since $\sigma, k_2 \models \mathsf{p}$, the property $(\mathsf{q}_1\mathsf{U}\mathsf{r}_1) \wedge (\mathsf{q}_2\mathsf{U}\mathsf{r}_2)$ holds for all $t \geq k_2$. Using the witnesses k_1 and k_2 and the safety conditions above, it also holds for all $t < k_2$. Hence $\sigma, 0 \models \mathsf{p}$.

Both directions hold, so the equivalence follows. □

We now give a logical equivalence for the general class of predicates considered in Thm. 4.

Lemma 8. *Consider the formula* $\mathsf{p}_{\mathcal{I},\mathcal{J}}$ *defined as*

$$\mathsf{p}_{\mathcal{I},\mathcal{J}} := \bigwedge_{i \in \mathcal{I}} \mathsf{G}(\mathsf{q}_i\,\mathsf{U}\,\mathsf{r}_i) \wedge \bigwedge_{j \in \mathcal{J}} (\mathsf{q}_j\,\mathsf{U}\,\mathsf{r}_j) \wedge \mathsf{Gq}$$

Then, $\mathsf{p}_{\mathcal{I},\mathcal{J}}$ *can be equivalently written as a single nested until formula:*

$$\mathsf{p}_{\mathcal{I},\mathcal{J}} \equiv \tilde{\mathsf{q}}_{\mathcal{I},\mathcal{J}}\,\mathsf{U}\,\tilde{\mathsf{r}}_{\mathcal{I},\mathcal{J}},$$

where

$$\tilde{\mathsf{q}}_{\mathcal{I},\mathcal{J}} \coloneqq \bigwedge_{j\in\mathcal{J}} \mathsf{q}_j \wedge \mathsf{q} \wedge \bigwedge_{i\in\mathcal{I}} (\mathsf{q}_i \vee \mathsf{r}_i),$$
$$\tilde{\mathsf{r}}_{\mathcal{I},\mathcal{J}} \coloneqq \bigvee_{j\in\mathcal{J}} \mathsf{r}_j \wedge \Phi_{\mathcal{I},\mathcal{J}\setminus\{j\}}$$

and

$$\mathsf{p}_{\mathcal{I},\emptyset} \equiv \mathsf{G}\Big(\bigwedge_{i\in\mathcal{I}} (\mathsf{q}_i \wedge \mathsf{q})\ \mathsf{U}\ (\mathsf{r}_i \wedge \mathsf{q})\Big).$$

Proof. We start by proving $\mathsf{p}_{\mathcal{I},\emptyset}$. Then, we prove $\mathsf{p}_{\mathcal{I},\mathcal{J}}$.

$$\begin{aligned}\mathsf{p}_{\mathcal{I},\emptyset} &\coloneqq \bigwedge_{i\in\mathcal{I}} \mathsf{G}(\mathsf{q}_i\ \mathsf{U}\ \mathsf{r}_i) \wedge \mathsf{G}\mathsf{q},\\ &\equiv \mathsf{G}\Big(\bigwedge_{i\in\mathcal{I}} (\mathsf{q}_i\ \mathsf{U}\ \mathsf{r}_i) \wedge \mathsf{q}\Big),\\ &\equiv \mathsf{G}\Big(\bigwedge_{i\in\mathcal{I}} (\mathsf{q}_i \wedge \mathsf{q})\ \mathsf{U}\ (\mathsf{r}_i \wedge \mathsf{q})\Big).\end{aligned}$$

Now we prove $\mathsf{p}_{\mathcal{I},\mathcal{J}}$. Define $\tilde{\mathsf{r}}^{\mathrm{U},\mathcal{J}}$ as the reward function obtained when applying the transformation to a conjunction of until formulas, i.e.,

$$\tilde{\mathsf{r}}^{\mathrm{U},\mathcal{J}} \coloneqq \bigvee_{j\in\mathcal{J}} \Big\{\mathsf{r}_j \wedge \bigwedge_{j\in\mathcal{J}\setminus\{j\}} (\mathsf{q}_j\ \mathsf{U}\ \mathsf{r}_j)\Big\}.$$

Then,

$$\begin{aligned}\mathsf{p}_{\mathcal{I},\mathcal{J}} &\coloneqq \bigwedge_{i\in\mathcal{I}} \mathsf{G}(\mathsf{q}_i\ \mathsf{U}\ \mathsf{r}_i) \wedge \bigwedge_{j\in\mathcal{J}} (\mathsf{q}_j\ \mathsf{U}\ \mathsf{r}_j) \wedge \mathsf{G}\mathsf{q},\\ &\equiv \bigwedge_{i\in\mathcal{I}} \mathsf{G}(\mathsf{q}_i\ \mathsf{U}\ \mathsf{r}_i) \wedge \tilde{\mathsf{q}}^{\mathrm{U},\mathcal{J}}\ \mathsf{U}\ \tilde{\mathsf{r}}^{\mathrm{U},\mathcal{J}} \wedge \mathsf{G}\mathsf{q},\\ &\equiv \Big(\tilde{\mathsf{q}}^{\mathrm{U},\mathcal{J}} \wedge \mathsf{q} \wedge \bigwedge_{i\in\mathcal{I}} (\mathsf{q}_i \vee \mathsf{r}_i)\Big)\ \mathsf{U}\ \Big(\tilde{\mathsf{r}}^{\mathrm{U},\mathcal{J}} \wedge \bigwedge_{i\in\mathcal{I}} \mathsf{q}_i\ \mathsf{U}\ \mathsf{r}_i \wedge \mathsf{G}\mathsf{q}\Big).\end{aligned}$$

Examining the right argument of the U operator, we see that

$$\begin{aligned}&\tilde{\mathsf{r}}^{\mathrm{U},\mathcal{J}} \wedge \bigwedge_{i\in\mathcal{I}} \mathsf{q}_i\ \mathsf{U}\ \mathsf{r}_i \wedge \mathsf{G}\mathsf{q}\\ &= \Big(\bigvee_{j\in\mathcal{J}} \Big\{\mathsf{r}_j \wedge \bigwedge_{j\in\mathcal{J}\setminus\{j\}} (\mathsf{q}_j\ \mathsf{U}\ \mathsf{r}_j)\Big\}\Big) \wedge \bigwedge_{i\in\mathcal{I}} \mathsf{q}_i\ \mathsf{U}\ \mathsf{r}_i \wedge \mathsf{G}\mathsf{q},\\ &= \bigvee_{j\in\mathcal{J}} \Big(\mathsf{r}_j \wedge \underbrace{\bigwedge_{j\in\mathcal{J}\setminus\{j\}} (\mathsf{q}_j\ \mathsf{U}\ \mathsf{r}_j) \wedge \bigwedge_{i\in\mathcal{I}} \mathsf{q}_i\ \mathsf{U}\ \mathsf{r}_i \wedge \mathsf{G}\mathsf{q}}_{\coloneqq \Phi_{\mathcal{I},\mathcal{J}\setminus\{j\}}}\Big).\end{aligned}$$

Plugging this back in completes the proof. □

C. Logic vs. Value Examples

In this section we reproduce an argument from [5] to demonstrate the following point: the algebraic relations that apply to the quantitative semantics in TL do not generally apply to the optimal value functions associated with the quantitative semantics. Many previous works have explored and leveraged the algebraic relations dictating quantitative semantics, while we focus on building an algebra for the value functions. An example highlighting the difference between the two is as follows.

Consider a reach-always-avoid (RAA) problem (i.e. reach a target set while avoiding an obstacle both before and after the target is reached) in which an agent would like to canoe across a river without hitting any rocks. Let r represent reaching the other side of the river and q represent not hitting a rock. The TL formula for the RAA problem is then $\mathsf{F}\mathsf{r} \wedge \mathsf{G}\mathsf{g}$. By definition, the following algebraic decomposition of the quantitative semantic for this predicate always holds:

$$\rho[\mathsf{F}\mathsf{r} \wedge \mathsf{G}\mathsf{g}](\xi_x^\alpha) = \min\left\{\rho[\mathsf{F}\mathsf{r}](\xi_x^\alpha), \rho[\mathsf{G}\mathsf{g}](\xi_x^\alpha)\right\}. \tag{3}$$

However, the analogous relation does not generally hold for the optimal value functions. To see this point, recall that

$$\begin{aligned} V^*[\mathsf{Fr}](x) &:= \max_\alpha \rho[\mathsf{Fr}](\xi_x^\alpha), \\ V^*[\mathsf{Gq}](x) &:= \max_\alpha \rho[\mathsf{Gq}](\xi_x^\alpha), \\ V^*[\mathsf{Fr} \wedge \mathsf{Gq}](x) &:= \max_\alpha \rho[\mathsf{Fr} \wedge \mathsf{Gq}](\xi_x^\alpha) \\ &= \max_\alpha \min \left\{ \rho[\mathsf{Fr}](\xi_x^\alpha), \rho[\mathsf{Gg}](\xi_x^\alpha) \right\}. \end{aligned}$$

It is always the case that

$$\begin{aligned} &\max_\alpha \min \left\{ \rho[\mathsf{Fr}](\xi_x^\alpha), \rho[\mathsf{Gg}](\xi_x^\alpha) \right\} \\ &\qquad \leq \min \left\{ \max_\alpha \rho[\mathsf{Fr}](\xi_x^\alpha), \max_\alpha \rho[\mathsf{Gg}](\xi_x^\alpha) \right\}, \end{aligned}$$

so that

$$V^*[\mathsf{Fr} \wedge \mathsf{Gq}](x) \leq \min \left\{ V^*[\mathsf{Fr}](x), V^*[\mathsf{Gq}](x) \right\}. \tag{4}$$

By contrast with the equality in 3, the inequality in 4 may indeed by strict. For example, suppose that I begin in a state x for which I can either (a) stay still indefinitely in my current state or (b) get across the river while necessarily hitting a rock on the way. In this case $V^*[\mathsf{Fr}](x) \geq 0$ and $V^*[\mathsf{Gq}](x) \geq 0$, but $V^*[\mathsf{Fr} \wedge \mathsf{Gq}](x) < 0$.

To summarize, even when an algebraic relation holds for the quantitative semantics of some predicate (as in 3), the corresponding algebraic relation for the optimal value functions may not hold. Instead, the two expressions may at best be related by an inequality (as in 4). This observation motivates our work on algebraically rules for decomposing optimal value functions.

D. Agreeable Algebra

In this section, we certify the algebraic properties of Bellman Value functions that match those of logic, corresponding to Lem. 1 from the main text, restated here for clarity. These will prove fundamental to the later derivations.

Lemma 1. *Let $\mathsf{v_p}$ be the predicate for $V[\mathsf{p}]$, i.e. $(\xi_x, t) \models \mathsf{v_p} \iff V[\mathsf{p}](\xi_x(t)) \geq 0$. Recall that*

$$\rho[\mathsf{v_p}](\xi_x, t) := V[\mathsf{p}](\xi_x(t)). \tag{2}$$

The following properties hold:

1) $V[\mathsf{a} \vee \mathsf{b}](x) = V[\mathsf{v_a} \vee \mathsf{v_b}](x)$
2) $V[\mathsf{a}\,\mathsf{U}\,\mathsf{b}](x) = V[\mathsf{a}\,\mathsf{U}\,\mathsf{v_b}](x)$

Proof. We give a direct algebraic derivation of each property. Recall that we write $\rho(\xi_x^\alpha) := \rho(\xi_x^\alpha, 0)$ for brevity. We begin with the first property,

$$\begin{aligned} V^*[\mathsf{a} \vee \mathsf{b}](x) &= \max_\alpha \max\{\rho[\mathsf{a}](\xi_x^\alpha),\ \rho[\mathsf{b}](\xi_x^\alpha)\} \\ &= \max \left\{ \max_\alpha \rho[\mathsf{a}](\xi_x^\alpha),\ \max_\alpha \rho[\mathsf{b}](\xi_x^\alpha) \right\} \\ &= \max \left\{ V^*[\mathsf{a}](x),\ V^*[\mathsf{b}](x) \right\} \\ &= \max_\alpha \max \left\{ V^*[\mathsf{a}](\xi_x^\alpha(0)),\ V^*[\mathsf{b}](\xi_x^\alpha(0)) \right\} \\ &= V^*[\mathsf{v_a^*} \vee \mathsf{v_b^*}](x). \end{aligned}$$

Next, we prove the second property using Lem. 3.

$$\begin{aligned} &V^*[\mathsf{a}\,\mathsf{U}\,\mathsf{b}](x) \\ &= \max_\alpha \max_t \min\{\rho[\mathsf{b}](\xi_x^\alpha, t),\ \min_{\kappa \in [0,t]} \rho[\mathsf{a}](\xi_x^\alpha, \kappa)\} \\ &= \max_t \max_{\alpha_{t-}} \min\{\max_{\alpha_{t+}} \rho[\mathsf{b}](\xi_{\xi_x^{\alpha_{t-}}(t)}^{\alpha_{t+}}, 0),\ \min_{\kappa \in [0,t]} \rho[\mathsf{a}](\xi_x^{\alpha_{t-}}, \kappa)\} \\ &= \max_t \max_{\alpha_{t-}} \min\{V^*[\mathsf{b}](\xi_x^{\alpha_{t-}}(t)),\ \min_{\kappa \in [0,t]} \rho[\mathsf{a}](\xi_x^{\alpha_{t-}}, \kappa)\} \\ &= \max_t \max_\alpha \min\{V^*[\mathsf{b}](\xi_x^\alpha(t)),\ \min_{\kappa \in [0,t]} \rho[\mathsf{a}](\xi_x^\alpha, \kappa)\} \\ &= V^*[\mathsf{a}\,\mathsf{U}\,\mathsf{v_b^*}](x) \end{aligned}$$

□

Intuitively, these properties illustrate when the algebra of Bellman Value functions is equivalent to that of logic vis a vis the logical operators that "align" with the optimum over actions. Namely, these are the $\vee$ and right-side U which are quantitatively represented by maxima, and hence, commute with the maxima over action sequences (in the appropriate settings, e.g. finite state spaces).

With these equivalences, relevant classes of predicates are immediately decomposable, given by the following corollaries.

Corollary 2. *Let a predicate p_N be defined by the chain of N-Untils over predicates a_i, s.t.*

$$\mathsf{p}_N = (\mathsf{a}_N \,\mathsf{U}\, \mathsf{p}_{N-1}), \qquad \mathsf{p}_1 = \mathsf{a}_1.$$

Then then following property holds,

$$V^*[\mathsf{p}](x) = V^*[\mathsf{a}_N \wedge \mathsf{v}^*_{\mathsf{p}_{N-1}}](x).$$

This result, which is proved by simple iterative application of the first property of Lem. 1, shows that the Bellman value for a chain of Untils is equivalent to a chain of $\mathcal{RA}$ Bellman Values. Notably, another special case of this property is the eventually-always predicate FGr, which corresponds to the reach-stay Value.

Corollary 3. *For the eventually-always predicate FGr, and corresponding reach-stabilize Value,*

$$V^*[\mathsf{FGr}](x) = V^*[\mathsf{Fv}^*_{\mathsf{Gr}}](x),$$

where V_{Gr} is the $\mathcal{A}$-Value for the region defined by $\neg\mathsf{r}$.

Ultimately, the equivalences given in Lem. 1 are vital tools to the following proofs. After a reorganization of the logic into an "agreeable" form, the application of these results yields the decomposed form, when combined with a few auxiliary algebraic results for manipulation. These are given here, the first of which concerns the next operator X.

Lemma 9. *For any predicate p,*

$$V^*[\mathsf{Xp}](x) = V^*[\mathsf{Xv}^*_{\mathsf{p}}](x).$$

Proof. By definition,

$$\begin{aligned}
V^*[\mathsf{Xp}](x) &= \max_{\alpha} \rho[\mathsf{p}](\xi_x^{\alpha}, 1) \\
&= \max_{a_1 \in \mathcal{A}} \max_{\alpha'} \rho[\mathsf{p}](\xi_{f(x,a_1)}^{\alpha'}, 0) \\
&= \max_{a \in \mathcal{A}} V^*[\mathsf{p}](f(x,a)) \\
&= \max_{\alpha} V^*[\mathsf{p}](\xi_x^{\alpha}(1)) \\
&= \max_{\alpha} \rho[\mathsf{v}^*_{\mathsf{p}}](\xi_x^{\alpha}, 1)
\end{aligned}$$

□

Finally, we have a result for a special case of conjunction $\wedge$ operator, corresponding to predicates which are unaffected by control actions.

Lemma 10. *Let a predicate c satisfy*

$$\rho[\mathsf{c}](\xi_x^{\alpha}, t) = \rho[\mathsf{c}](\xi_x^{\beta}, t), \qquad \forall \alpha, \beta \in \mathcal{A}^{\mathbb{N}}.$$

Then then following property holds,

$$V^*[\mathsf{c} \wedge \mathsf{p}](x) = V^*[\mathsf{c} \wedge \mathsf{v}^*_{\mathsf{p}}].$$

Proof.

$$\begin{aligned} V^*[\mathsf{c} \wedge \mathsf{p}](x) &= \max_{\alpha} \min\{\rho[\mathsf{c}](\xi_x^\alpha),\, \rho[\mathsf{p}](\xi_x^\alpha)\} \\ &= \min\left\{\rho[\mathsf{c}](\xi_x^\beta),\, \max_{\alpha} \rho[\mathsf{p}](\xi_x^\alpha)\right\}, \quad \beta \in \mathcal{A}^{\mathbb{N}} \\ &= \min\left\{\rho[\mathsf{c}](\xi_x^\beta),\, V^*[\mathsf{p}](x)\right\} \\ &= \max_{\alpha} \min\left\{\rho[\mathsf{c}](\xi_x^\alpha),\, V^*[\mathsf{p}](\xi_x^\alpha(0))\right\} \\ &= V^*[\mathsf{c} \wedge \mathsf{v}_\mathsf{p}^*](x). \end{aligned}$$

□

This result captures that when a predicate is unaffected by the control actions – and so we migth say "uncontrollable" – then trivially, the maxima over control actions may pass over the minimum defined by the $\wedge$ operator. With these rules, we are now able to simplify the decomposition of the Bellman Value for complex logic.

E. N-$\mathcal{RA}$ Results

In this section, we offer proof for the first main result in the work decomposing the N-$\mathcal{RA}$ Value, corresponding to Thm. 1 from the main text, restated here for clarity.

Theorem 1. *For the predicate* $\mathsf{p} := \bigwedge_{i \in \mathcal{I}} (\mathsf{q}_i \,\mathsf{U}\, \mathsf{r}_i)$*, the corresponding Bellman Value satisfies*

$$V^*\Big[\bigwedge_i (\mathsf{q}_i \,\mathsf{U}\, \mathsf{r}_i)\Big](x) = V^*\big[\tilde{\mathsf{q}} \,\mathsf{U}\, \tilde{\mathsf{r}}\big](x)$$

where,

$$\tilde{\mathsf{r}} := \bigvee_i \left(\mathsf{r}_i \wedge \mathsf{v}^*_{\mathsf{p}^{-i}}\right), \qquad \tilde{\mathsf{q}} := \bigwedge_i \mathsf{q}_i,$$

and $\mathsf{p}^{-i} := \bigwedge_{j \in \mathcal{I} \setminus \{i\}} \mathsf{q}_j \,\mathsf{U}\, \mathsf{r}_j$.

Proof. The strategy for the proof is to first rearrange the logic into a certain form for which application of the algebraic results in Sec. D is straightforward. Ultimately, this process yields the decomposition of the Bellman Value we desire.

Beginning with the logic, Lem. 5 reorganizes the N-Until conjunction, giving

$$\mathsf{p} := \bigwedge_{i=1}^N (\mathsf{q}_i \,\mathsf{U}\, \mathsf{r}_i) \equiv \Big(\bigwedge_{i=1}^N \mathsf{q}_i\Big) \,\mathsf{U}\, \Big(\bigvee_{i=1}^N \big(\mathsf{r}_i \wedge \mathsf{p}^{-i}\big)\Big) =: \tilde{\mathsf{q}} \,\mathsf{U}\, \mathsf{s}.$$

Hence,

$$V^*[\mathsf{p}](x) = V^*\Big[\tilde{\mathsf{q}} \,\mathsf{U}\, \mathsf{s}\Big](x).$$

Now, by applying the second property of Lem. 1, we have

$$V^*[\mathsf{p}](x) = V^*\Big[\tilde{\mathsf{q}} \,\mathsf{U}\, \mathsf{v}_\mathsf{s}^*\Big](x).$$

Given $\mathsf{w}_i := \mathsf{r}_i \wedge \mathsf{p}^{-i}$, we may apply the first property of Lem. 1,

$$V^*[\mathsf{s}](x) = V^*\Big[\bigvee_{i=1}^N \mathsf{v}^*_{\mathsf{w}_i}\Big](x) \implies \mathsf{v}^*_\mathsf{s} = \bigvee_{i=1}^N \mathsf{v}^*_{\mathsf{w}_i}.$$

Lastly, since r_i is immediate and thus uncontrollable, we may apply Lem. 10 to yield

$$V^*[\mathsf{w}_i](x) = V^*[\mathsf{r}_i \wedge \mathsf{v}^*_{\mathsf{p}^{-i}}](x) \implies \mathsf{v}^*_{\mathsf{w}_i} = \mathsf{r}_i \wedge \mathsf{v}^*_{\mathsf{p}^{-i}}.$$

In summary, we have

$$V^*[\mathsf{p}] = V^*\Big[\tilde{\mathsf{q}} \,\mathsf{U}\, \mathsf{v}^*_\mathsf{s}\Big] = V^*\Big[\tilde{\mathsf{q}} \,\mathsf{U}\, \Big(\bigvee_{i=1}^N \mathsf{v}^*_{\mathsf{w}_i}\Big)\Big] = V^*\Big[\tilde{\mathsf{q}} \,\mathsf{U}\, \tilde{\mathsf{r}}\Big],$$

where $\tilde{\mathsf{r}} := \bigvee_{i=1}^N \big(\mathsf{r}_i \wedge \mathsf{v}^*_{\mathsf{p}^{-i}}\big)$, as desired. □

The logic in this result, when combined with the $\mathcal{RAA}$ theorem in is equivalently applicable to the extended case involving Gq, given by the following corollary.

Corollary 4. *For the predicate*

$$\mathsf{p} := \bigwedge_{i\in\mathcal{I}} (\mathsf{q}_i \;\mathsf{U}\; \mathsf{r}_i) \wedge \mathsf{Gq},$$

the corresponding Bellman Value satisfies

$$V^*\big[\bigwedge_i (\mathsf{q}_i \;\mathsf{U}\; \mathsf{r}_i) \wedge \mathsf{Gq}\big](x) = V^*\big[\tilde{\mathsf{q}} \,\mathsf{U}\, \tilde{\mathsf{r}}\big](x)$$

where,

$$\tilde{\mathsf{r}} := \bigvee_i \left(\mathsf{r}_i \wedge \mathsf{v}^*_{\mathsf{p}^{-i}}\right), \qquad \tilde{\mathsf{q}} := \bigwedge_i \mathsf{q}_i \wedge \mathsf{q},$$

and $\mathsf{p}^{-i} := \bigwedge_{j\in\mathcal{I}\setminus\{j\}} (\mathsf{q}_j \;\mathsf{U}\; \mathsf{r}_j) \wedge \mathsf{Gq}$, *and*

$$V[(\mathsf{q}_j \;\mathsf{U}\; \mathsf{r}_j) \wedge \mathsf{Gq}](x) = V^*[\mathsf{q}_j \;\mathsf{U}\; \left(\mathsf{r}_j \wedge \mathsf{v}^*_{\mathsf{Gq}}\right)](x).$$

Proof. The proof follows identical to the previous theorem with the altered definition of p and p^{-i}. □

F. N-$\mathcal{RA}_\ell$ Results

In this section, we give several properties surrounding the GF operation, including the RA_ℓ Bellman equation that may be used in this context and the extension to G of multi-eventually and Until predicates.

Note, by definition we have the following property.

$$\rho[\mathsf{GFr}](\xi_x, t) = \inf_{t'\geq t} \sup_{t''\geq t'} \rho[\mathsf{r}](\xi_x, t'') = \limsup_{s\to\infty} \rho[\mathsf{r}](\xi_x, s).$$

This is, ofcourse, a special case of the $\mathsf{G}(\mathsf{q}\,\mathsf{U}\,\mathsf{r})$ Bellman equation, which itself satisfies

$$\begin{aligned}
&\rho[\mathsf{G}(\mathsf{q}\,\mathsf{U}\,\mathsf{r})](\xi_x, t) \\
&\qquad = \inf_{t'\geq t} \sup_{t''\geq t'} \min\{\rho[\mathsf{r}](\xi_x, t''), \min_{\kappa\leq t''} \rho[\mathsf{q}](\xi_x, \kappa)\} \\
&\qquad = \limsup_{s\to\infty} \min\{\rho[\mathsf{r}](\xi_x, s), \min_{\kappa\leq s} \rho[\mathsf{q}](\xi_x, \kappa)\} \\
&\qquad = \min\{\limsup_{s\to\infty} \rho[\mathsf{r}](\xi_x, s), \min_{\kappa\geq t} \rho[\mathsf{q}](\xi_x, \kappa)\} \\
&\qquad = \rho[\mathsf{GFr} \wedge \mathsf{Gq}](\xi_x, t).
\end{aligned}$$

In either case, the infinite-horizon nature of the G composition immediately xyields several qualities regarding the temporal-independence of the G compositions.

Lemma 11. *The following properties hold:*

- $\rho[\mathsf{G}(\mathsf{q}\,\mathsf{U}\,\mathsf{r})](\xi_x, t) = \rho[\mathsf{G}(\mathsf{q}\,\mathsf{U}\,\mathsf{r})](\xi_x, s), \quad \forall s \geq t.$
- $\mathsf{G}(\mathsf{q}\,\mathsf{U}\,\mathsf{r}) = \mathsf{X}^n\mathsf{G}(\mathsf{q}\,\mathsf{U}\,\mathsf{r}), \quad \forall n \in \mathbb{N}$
- $V^*[\mathsf{G}(\mathsf{q}\,\mathsf{U}\,\mathsf{r})](x) = V^*[\mathsf{G}(\mathsf{q}\,\mathsf{U}\,\mathsf{r})](\xi_x^\alpha(s)), \quad \forall s \geq 0.$

By logical rearrangement and application of the algebraic results, we may immediately have Thm. 2 restated here for clarity.

Theorem 2. *For the predicate* $\mathsf{p} := \mathsf{G}\,(\mathsf{q}\;\mathsf{U}\;\mathsf{r})$ *the corresponding Bellman Value satisfies*

$$V^*[\mathsf{G}\,(\mathsf{q}\;\mathsf{U}\;\mathsf{r})](x) = V^*\big[\mathsf{q}\,\mathsf{U}\,(\mathsf{r} \wedge \mathsf{X}\mathsf{v}^*_{\mathsf{p}})\big](x).$$

Proof. As with the proof of Thm. 1, we begin by rearranging the logic using Lem. 6,

$$\mathsf{G}(\mathsf{q}\,\mathsf{U}\,\mathsf{r}) = \mathsf{q}\,\mathsf{U}\,(\mathsf{r} \wedge \mathsf{XG}(\mathsf{q}\,\mathsf{U}\,\mathsf{r})) =: \mathsf{q}\,\mathsf{U}\,\mathsf{s}.$$

Hence, by applying the second property of Lem. 1, Lem. 10 and Lem. 9, we have

$$V^*[\mathsf{G}(\mathsf{q}\,\mathsf{U}\,\mathsf{r})] = V^*\Big[\tilde{\mathsf{q}}\,\mathsf{U}\,\mathsf{v}^*_{\mathsf{s}}\Big] = V^*\Big[\tilde{\mathsf{q}}\,\mathsf{U}\,(\mathsf{r} \wedge \mathsf{X}\mathsf{v}^*_{\mathsf{G}(\mathsf{q}\,\mathsf{U}\,\mathsf{r})})\Big].$$

□

Notably, we may generalize this result to handle a composition of G with multiple eventually and Until predicates, by considering a loop of Bellman Values of the previous form. This corresponds to Thm. 3 from the main text, restated as follows.

Theorem 3. *Given the set of coupled Bellman Values of length $J = |\mathcal{J}|$,*

$$V_j^*(x) := V^* \left[\tilde{\mathsf{q}}_j \,\mathsf{U}\, (\tilde{\mathsf{r}}_j \wedge \mathsf{X}\mathsf{v}_{j+1}^*)\right](x)$$

where $J+1 := 1$, $\tilde{\mathsf{q}}_j := \mathsf{q}_j \wedge (\mathsf{q}_{j+1} \vee \mathsf{r}_{j+1})$, and $\tilde{\mathsf{r}}_j := \mathsf{r}_j \wedge (\mathsf{q}_{j+1} \vee \mathsf{r}_{j+1})$, then $\forall j$, defined by

$$V^* \left[\mathsf{G}\left(\bigwedge_{j\in\mathcal{J}} (\mathsf{q}_j \,\mathsf{U}\, \mathsf{r}_j)\right)\right](x) = V_j^*(x).$$

Proof. Without loss of generality, we consider the case $N = 2$ for clarity, with the general case following similarly. Recall, by Lem. 7, for $\mathsf{p} = \mathsf{G}\big((\mathsf{q}_1 \,\mathsf{U}\, \mathsf{r}_1) \wedge (\mathsf{q}_2 \,\mathsf{U}\, \mathsf{r}_2)\big)$ we have

$$\begin{aligned} \mathsf{p} &\equiv \tilde{\mathsf{q}}_1 \,\mathsf{U}\, \Big(\tilde{\mathsf{r}}_1 \wedge \big(\tilde{\mathsf{q}}_2 \,\mathsf{U}\, (\tilde{\mathsf{r}}_2 \wedge \mathsf{p})\big)\Big) \\ &\equiv \tilde{\mathsf{q}}_2 \,\mathsf{U}\, \Big(\tilde{\mathsf{r}}_2 \wedge \big(\tilde{\mathsf{q}}_1 \,\mathsf{U}\, (\tilde{\mathsf{r}}_1 \wedge \mathsf{p})\big)\Big). \end{aligned}$$

For $j \in [1,2]$, $J+1 := 1$, let

$$\mathsf{p}_j := \tilde{\mathsf{q}}_j \,\mathsf{U}\, \Big(\tilde{\mathsf{r}}_j \wedge \mathsf{p}_i\Big)$$

Then by definition,

$$\mathsf{p}_j = \tilde{\mathsf{q}}_j \,\mathsf{U}\, \Big(\tilde{\mathsf{r}}_j \wedge \big(\tilde{\mathsf{q}}_i \,\mathsf{U}\, (\tilde{\mathsf{r}}_i \wedge \mathsf{p})\big)\Big) \equiv \mathsf{p}.$$

Thus, it follows that

$$V^*[\mathsf{G}(\mathsf{q} \,\mathsf{U}\, \mathsf{r})] = V^*[\mathsf{p}_j], \quad \forall j \in \mathcal{J}.$$

Now, by applying the second property of Lem. 1, Lem. 10 and Lem. 9, we arrive at the desired result. ☐

Although, these results appear like the previous decompositions, it is important to note that they are fundamentally different due to the implicit definition of the Value. Moreover, they do not guarantee the uniqueness or existence of the solution. To certify these properties, we may consider the G composition as the limit of the finite iterations. This is given in Sec. G.

With the Value iteration results, we may know conditions under which this Value exists (e.g. finite state spaces), and proceed to solve this Value. While the Value iteration is a nice theoretical procedure, it may not be practical for large state spaces and certain specifications. To address these challenges, we propose the $\mathcal{RA}_\ell$ Bellman Equation in the main text, given here for clarity,

Lemma 2. *For the set of J Values defined in Thm. 3, let the $\mathcal{RA}_\ell$-BE be defined as*

$$\begin{aligned} \mathcal{B}^{\gamma}_{\mathcal{RA}_\ell}[V_j] := (1-\gamma)\min\{\tilde{r}_j, \tilde{q}_j\} + \\ \gamma \min\Big\{\max\Big\{\min\left\{\tilde{r}_j, V_{j+1}^+\right\}, V_j^+\Big\}, \tilde{q}_j\Big\}. \end{aligned}$$

This is contractive such that $V_j^\gamma = \mathcal{B}^{\gamma}_{\mathcal{RA}_\ell}[V_j^\gamma]$ has a unique fixed point, satisfying $\lim_{\gamma\to 1} V_j^\gamma = V^*[\mathsf{G}(\bigwedge_{j\in\mathcal{J}}(\mathsf{q}_j \,\mathsf{U}\, \mathsf{r}_j))]$.

Proof. We first prove the existence of the fixed point by showing that the operator is contractive and then show that in the limit of discounting, the fixed point achieves the desired solution. Note, in this context, $V \in \mathbb{R}^{|\mathcal{J}|}$ is a vector of Values.

1. Contraction:

Consider two vectors $V, W \in \mathbb{R}^{|\mathcal{J}|}$, and let $\|\cdot\|_\infty$ be the infinity norm. Here, we write $r = \tilde{r}_j$ and $q = \tilde{q}_j$ for brevity. Note for each component j we have,

$$\begin{aligned} &\|\mathcal{B}^{\gamma}_{\mathcal{RA}_\ell}[V_j] - \mathcal{B}^{\gamma}_{\mathcal{RA}_\ell}[W_j]\| \\ &= \gamma \|\min\{\max\{\min\{r_j, V_{j+1}^+\}, V_j^+\}, q\} - \\ &\qquad\qquad \min\{\max\{\min\{r, W_{j+1}^+\}, W_j^+\}, q\}\| \\ &\le \gamma \left\|\max\{\min\{r, V_{j+1}^+\}, V_j^+\} - \max\{\min\{r, W_{j+1}^+\}, W_j^+\}\right\| \\ &\le \gamma \max\{\left\|\min\{r, V_{j+1}^+\} - \min\{r, W_{j+1}^+\}\right\|, \left\|V_j^+ - W_j^+\right\|\} \\ &\le \gamma \max\{\left\|V_{j+1}^+ - W_{j+1}^+\right\|, \left\|V_j^+ - W_j^+\right\|\} \\ &\le \gamma L \max\{\|V_{j+1} - W_{j+1}\|, \|V_j - W_j\|\}, \end{aligned}$$

where the last line follows from the lipschitz continuity of $V(x)$, $W(x)$ and $f(x,a)$, given the definition $V_j^+(x) := \max_{a\in\mathcal{A}} V_j(f(x,a))$. Taking the maximum over all components j, we have then

$$\begin{aligned}\|\mathcal{B}^{\gamma}_{\mathcal{RA}_\ell}[V] - \mathcal{B}^{\gamma}_{\mathcal{RA}_\ell}[W]\|_\infty &\le \gamma L \max_j\{\|V_j - W_j\|\} \\ &= \gamma L \left\|V - W\right\|_\infty ,\end{aligned}$$

demonstrating that the operator $\mathcal{B}^{\gamma}_{\mathcal{RA}_\ell}$ is a contraction mapping.

2. Convergence in the limit of $\gamma \to 1$:
Let V^γ be the vector-valued fixed point defined by $V^\gamma = \mathcal{B}^{\gamma}_{\mathcal{RA}_\ell}[V^\gamma]$, s.t. for each component j we have

$$\begin{aligned}V_j^\gamma(x) = (1-\gamma)\min\{\tilde{r}_j, \tilde{q}_j\} + \\ \gamma \min\{\max\{\min\{\tilde{r}_j, V_{j+1}^{\gamma +}\}, V_j^{\gamma +}\}, \tilde{q}_j\}.\end{aligned}$$

Note, each component is just a special case of the proof of Proposition 3 in [56], hence we may know,

$$\begin{aligned}&\lim_{\gamma\to 1} V_j^\gamma(x) \\ &= \max_\alpha \max_t \min\{\min\{\tilde{r}_j(x), V_{j+1}^{*;+}(x)\}, \max_{\kappa\in[0,t]} \tilde{q}_j(x)\} \\ &= V_j^*[\tilde{\mathsf{q}}_j \,\mathsf{U}\, (\tilde{\mathsf{r}}_j \wedge \mathsf{X}\mathsf{v}_{j+1}^*)](x) \\ &= V^*\left[\mathsf{G}\left(\bigwedge_{j\in\mathcal{J}} (\mathsf{q}_j \;\mathsf{U}\; \mathsf{r}_j)\right)\right](x),\end{aligned}$$

where the last line follows from Thm. 3. □

G. $\mathsf{G}(\dots)$ Fixed Point Iteration

In this section, we present an alternate perspective on the Bellman Value corresponding to the $\mathsf{G}(\dots)$ compositions based on finite iterations of recursion. Indeed, one may use this approach to solve the Value, however, for large state spaces or complicated specifications, this may be expensive. We principally employ this approach to guarantee the uniqueness and existence of the corresponding Bellman Values (which in general may be ill defined) in order to accompany the $\mathcal{RA}_\ell$-BE.

A. Single-Predicate Recurrence

For clarity, we begin by considering the case involving the recurrence of a single predicate (target to reach), given by $\mathsf{p} := \mathsf{GFr}$ and Value

$$V[\mathsf{GFr}](x) = \max_\alpha \max_{t\ge 0} \min\left\{r(\xi_x^\alpha(t)), V[\mathsf{GFr}](\xi_x^\alpha(t+1))\right\}$$

per Thm 2.

We now consider the following other value function:

$$\begin{aligned}V_{k+1}(x) &:= V^*[\mathsf{F}(\mathsf{r} \wedge \mathsf{X}\mathsf{v}_k)](x) \\ &= \max_\alpha \max_{t\ge 0} \min\Big(r(\xi_x^\alpha(t)),\ V_k(\xi_x^\alpha(t+1))\Big),\end{aligned}$$

where $V_0(x) := \infty$ for all x i.e. $\mathsf{v}_0 := \top$.

Lemma 12. *The sequence V^k converges to $V[\mathsf{GFr}]$ pointwise, i.e., for all x,*

$$\lim_{k\to\infty} V_k(x) = V[\mathsf{GFr}](x).$$

Proof. First, for an arbitrary threshold λ, construct the superlevel sets R, W^* and W_k as

$$\begin{aligned}R &:= \{x : r(x) \ge \lambda\}, \\ W^* &:= \{x : V^*[\mathsf{GFr}](x) \ge \lambda\}, \\ W_k &:= \{x : V_k(x) \ge \lambda\}.\end{aligned}$$

Note that W_k is exactly the set of states from which it is possible to reach R at least k times.

Since $V_0(x) = \infty$ for all x, we have $W_0 = \mathcal{X}$. Let $\mathcal{T}$ denote the operator that maps V_k to V_{k+1}, i.e., $V_{k+1} = \mathcal{T}V_k$. By Lem. 13, $\mathcal{T}$ is monotone, i.e., $U(x) \leq V(x) \implies \mathcal{T}U(x) \leq \mathcal{T}V(x)$ for all x. Moreover, since $V_1 \leq V_0$, we have $V_{k+1} \leq V_k$ for all k by induction, and thus $W_{k+1} \subseteq W_k$ for all k.

Since W_k is a decreasing sequence of sets, the limit $W_\infty = \bigcap_{k=0}^{\infty} W_k$ exists, and also that $\lim_{k\to\infty} V_k(x) = V^\infty(x)$ exists for all x.

a) 1. ($W^* \subseteq W_\infty$): Let $x \in W^*$. Then, by definition of $V^*[\mathsf{GFr}]$, there exists an action sequence α such that the system visits R infinitely often. In particular, for any $k \in \mathbb{N}$, the system can reach R at least k times under α. Hence, $x \in W_k$ for all k, and thus $x \in W_\infty$.

b) 2. ($W^* \supseteq W_\infty$): We apply either Lem. 14, 15, or 16 depending on the assumptions on the state and action spaces to conclude that $W_\infty \subseteq W^*$.

Since we have shown both inclusions, we conclude that $W^* = W_\infty$. Since this holds for any threshold λ, we have $\lim_{k\to\infty} V_k(x) = V^*[\mathsf{GFr}](x)$ for all x, i.e., V_k converges pointwise to $V^*[\mathsf{GFr}]$. □

Lemma 13. *The operator $\mathcal{T}$ defined as*

$$\mathcal{T}V(x) = \max_{\alpha} \max_{t\geq 0} \min\Big(r(\xi_x^\alpha(t)),\ V(\xi_x^\alpha(t+1))\Big)$$

is monotone, i.e., for any two functions U and V such that $U(x) \leq V(x)$ for all x, we have $\mathcal{T}U(x) \leq \mathcal{T}V(x)$ for all x.

Proof. Let U and V be two functions such that $U(x) \leq V(x)$ for all x. Then, for any action sequence α and any time t,

$$\min\Big(r(\xi_x^\alpha(t)),\ U(\xi_x^\alpha(t+1))\Big) \leq \min\Big(r(\xi_x^\alpha(t)),\ V(\xi_x^\alpha(t+1))\Big).$$

Taking $\max$ over t and α on both sides yields

$$\mathcal{T}U(x) \leq \mathcal{T}V(x).$$

□

Lemma 14. *Suppose the set of states $\mathcal{X}$ is finite. Then, $W_\infty \subseteq W^*$.*

Proof. First, since $\mathcal{X}$ is finite, $W_k \subseteq \mathcal{X}$ is finite for all k. Moreover, since $W_{k+1} \subseteq W_k$ for all k, the sequence W_k must stabilize at some finite K, i.e., $W_K = W_\infty$ for some K. Hence, W_∞ is a fixed point of the operator that maps W_k to W_{k+1}.

Now, let $x \in W_\infty$. Since W_∞ is a fixed point, there exists some action sequence α and time t such that $\xi_x^\alpha(t) \in R$, and $\xi_x^\alpha(t) \in W_\infty$. We can repeat this argument to construct an infinite action sequence α under which the system visits R infinitely often. Thus, $x \in W^*$, and $W_\infty \subseteq W^*$. □

Lemma 15. *Suppose the set of actions $\mathcal{A}$ is finite. Then, $W_\infty \subseteq W^*$.*

Proof. Let $x \in W_\infty$. Then, for any $k \in \mathbb{N}$, there exists an action sequence α^k such that the system can reach R at least k times under α^k. We now construct a "success tree" where, from every node, we create a branch for each action in $\mathcal{A}$, and we remove all nodes that are not in W_∞. Since $\mathcal{A}$ is finite, this tree has a finite branching factor. Moreover, since $x \in W_\infty$, for any depth k, there exists a path from the root to a node at depth k. By König's lemma [77], there exists an infinite path from the root. Since all nodes in the tree are in W_∞, this infinite path corresponds to an action sequence under which the system visits R infinitely often. Thus, $x \in W^*$, and $W_\infty \subseteq W^*$. □

Lemma 16. *Suppose the set of actions $\mathcal{A}$ is a compact space, and the dynamics f is continuous in a. Then, $W_\infty \subseteq W^*$.*

Proof. Let $x \in W_\infty$. Then, for any $k \in \mathbb{N}$, there exists an action sequence α^k such that the system can reach R at least k times under α^k. We now construct a sequence of non-empty compact sets C_n as follows. Let $C_0 = \mathcal{A}$. For each $n \geq 1$, let

$$C_n = \{a \in C_{n-1} : \exists a_{1:\infty} \text{ s.t. the system reaches } R \text{ at least } n \text{ times under } (a, a_{1:\infty})\}.$$

Note that C_n is non-empty since $x \in W_\infty$. Moreover, C_n is closed since the dynamics f is continuous in a, and thus C_n is compact as a closed subset of the compact set C_{n-1}. Since $C_{n+1} \subseteq C_n$ for all n, by Cantor's intersection theorem [78], the intersection $\bigcap_{n=0}^{\infty} C_n$ is non-empty. Let a_0 be an element in this intersection. By construction of C_n, there exists an action sequence $a_{1:\infty}$ such that the system reaches R at least n times under $(a_0, a_{1:\infty})$ for all n. Hence, the system visits R infinitely often under the action sequence $(a_0, a_{1:\infty})$, and thus $x \in W^*$. Therefore, $W_\infty \subseteq W^*$. □

B. Multi-Predicate Recurrence

Here we give a generalization of the previous finite recurrence approach to compositions of G with multi-Until predicates. We give the proofs for the case with $N = 2$ but the generalization to $N > 2$ follows similarly.

Let the globally-(until and until) value function be defined as

$$\begin{aligned}
&V^*[\mathsf{G}(\wedge_j \mathsf{q}_j \,\mathsf{U}\, \mathsf{r}_j)](x_0) \\
&:= \max_\alpha \rho\big[\mathsf{G}(\mathsf{q}_1 \,\mathsf{U}\, \mathsf{r}_1 \wedge \mathsf{q}_2 \,\mathsf{U}\, \mathsf{r}_2)\big](x_0, 0) \\
&= \max_\alpha \min_{t\geq 0} \min\Big\{ \max_{s\geq t} \min\big\{ r_1\big(\xi^\alpha_{x_0}(s)\big), \\
&\qquad \min_{0\leq \ell < s} q_1\big(\xi^\alpha_{x_0}(\ell)\big)\big\}, \\
&\qquad \max_{u\geq t} \min\big\{ r_2\big(\xi^\alpha_{x_0}(u)\big), \\
&\qquad \min_{0\leq \ell < u} q_2\big(\xi^\alpha_{x_0}(\ell)\big)\big\}\Big\}.
\end{aligned}$$

Let $w_1 := q_1 \vee r_1$ and $w_2 := q_2 \vee r_2$, and define the "until" objective function U_i as

$$\begin{aligned}
U_i\big(\xi^{\alpha_{t:\infty}}_x\big) := \sup_{s\geq t} \min\big\{ &r_i\big(\xi^{\alpha_{t:\infty}}_x(s)\big), \\
&\min_{t\leq \ell < s} q_i\big(\xi^\alpha_x(\ell)\big)\big\}.
\end{aligned}$$

We now consider the following coupled system of value functions:

$$\begin{aligned}
&V_{1,k+1}(x_0) \\
&:= \max_\alpha \rho\big[(\mathsf{q}_1 \wedge \mathsf{w}_2) \,\mathsf{U}\, (\mathsf{r}_1 \wedge \mathsf{w}_2 \wedge \mathsf{X}\, V_{2,k})\big](x_0) \\
&= \max_\alpha \max_{t\geq 0} \min\Big\{ \min\big(r_1\big(\xi^\alpha_{x_0}(t)\big), \\
&\qquad w_2\big(\xi^\alpha_{x_0}(t)\big), V_{2,k}\big(\xi^\alpha_{x_0}(t+1)\big)\big), \\
&\qquad \min_{0\leq \ell < t} \min\Big(q_1\big(\xi^\alpha_{x_0}(\ell)\big), \\
&\qquad\quad w_2\big(\xi^\alpha_{x_0}(\ell)\big)\Big)\Big\}, \\
&V_{2,k+1}(x_0) \\
&:= \max_\alpha \rho\big[(\mathsf{q}_2 \wedge \mathsf{w}_1) \,\mathsf{U}\, (\mathsf{r}_2 \wedge \mathsf{w}_1 \wedge \mathsf{X}\, V_{1,k})\big](x_0) \\
&= \max_\alpha \max_{t\geq 0} \min\Big\{ \min\big(r_2\big(\xi^\alpha_{x_0}(t)\big), \\
&\qquad w_1\big(\xi^\alpha_{x_0}(t)\big), V_{1,k}\big(\xi^\alpha_{x_0}(t+1)\big)\big), \\
&\qquad \min_{0\leq \ell < t} \min\Big(q_2\big(\xi^\alpha_{x_0}(\ell)\big), \\
&\qquad\quad w_1\big(\xi^\alpha_{x_0}(\ell)\big)\Big)\Big\},
\end{aligned}$$

where $V_{1,0}(x) := \infty$ and $V_{2,0}(x) := \infty$ for all x.

Lemma 17. *For any $k > 0$, let $\xi^\alpha_{x_0}$ be the trajectory generated by the policy achieving the supremum in $V_{i,k}(x_0)$. Then,*

$$V_{i,k}(x_0) \leq U_i\big(\xi^{\alpha_{0:\infty}}_{x_0}\big) \tag{5}$$

Proof.

$$
\begin{aligned}
&V_{i,k}(x_0)\\
&= \max_{t\geq 0} \min\Big\{\min\big(r_i\big(\xi^{\alpha}_{x_0}(t)\big),\\
&\quad w_{\neg i}\big(\xi^{\alpha}_{x_0}(t)\big), V_{\neg i,k-1}\big(\xi^{\alpha}_{x_0}(t{+}1)\big)\big),\\
&\quad \min_{0\leq \ell<t} \min\Big(q_i\big(\xi^{\alpha}_{x_0}(\ell)\big),\\
&\qquad w_{\neg i}\big(\xi^{\alpha}_{x_0}(\ell)\big)\Big)\Big\}\\
&\leq \max_{t\geq 0} \min\Big\{r_i\big(\xi^{\alpha}_{x_0}(t)\big),\\
&\quad \min_{0\leq \ell<t} q_i\big(\xi^{\alpha}_{x_0}(\ell)\big)\Big\}\\
&= U_i\big(\xi^{\alpha_{0:\infty}}_{x_0}\big).
\end{aligned}
$$

□

Lemma 18. *Both sequences $V_{1,k}$ and $V_{2,k}$ converge to $V^*[\mathsf{G}(\wedge_j \mathsf{q}_j \,\mathsf{U}\, \mathsf{r}_j)]$ pointwise, i.e., for all x,*

$$
\begin{aligned}
\lim_{k\to\infty} V_{1,k}(x) &= \lim_{k\to\infty} V_{2,k}(x)\\
&= V^*[\mathsf{G}(\wedge_j \mathsf{q}_j \,\mathsf{U}\, \mathsf{r}_j)](x).
\end{aligned}
$$

Before we prove Lem. 18, we set up a few useful definitions and lemmas.
Define the operator $\mathcal{T}$ mapping (J_1, J_2) to (J_1', J_2') as

$$
\begin{aligned}
&J_1'(x_0)\\
&:= \sup_{\alpha} \sup_{t\geq 0} \min\Big\{\min\big(r_1\big(\xi^{\alpha}_{x_0}(t)\big),\\
&\quad w_2\big(\xi^{\alpha}_{x_0}(t)\big), J_2\big(\xi^{\alpha}_{x_0}(t{+}1)\big)\big),\\
&\quad \min_{0\leq \ell<t} \min\Big(q_1\big(\xi^{\alpha}_{x_0}(\ell)\big),\\
&\qquad w_2\big(\xi^{\alpha}_{x_0}(\ell)\big)\Big)\Big\},
\end{aligned}
$$

$$
\begin{aligned}
&J_2'(x_0)\\
&:= \sup_{\alpha} \sup_{t\geq 0} \min\Big\{\min\big(r_2\big(\xi^{\alpha}_{x_0}(t)\big),\\
&\quad w_1\big(\xi^{\alpha}_{x_0}(t)\big), J_1\big(\xi^{\alpha}_{x_0}(t{+}1)\big)\big),\\
&\quad \min_{0\leq \ell<t} \min\Big(q_2\big(\xi^{\alpha}_{x_0}(\ell)\big),\\
&\qquad w_1\big(\xi^{\alpha}_{x_0}(\ell)\big)\Big)\Big\}.
\end{aligned}
$$

Lemma 19. *The operator $\mathcal{T}$ is monotone.*

Proof. It follows immediately from the monotonicity of the $\sup$ and $\min$ operators. □

Lemma 20. *Both sequences converge pointwise, i.e., $V_{1,\infty}$ and $V_{2,\infty}$ exist.*

Proof. Since $V_{1,0}(x) = \infty$ and $V_{1,1}(x)$ is finite, $V_{1,1}(x) \leq V_{1,0}(x)$ for all x. By monotonicity of $\mathcal{T}$, the sequence $V_{1,k}$ is non-increasing. Moreover, $V_{1,0}(x)$ is bounded below by $\min(\inf_x r_1(x), \inf_x r_2(x))$. Thus, by the monotone convergence theorem, $V_{1,\infty}(x) = \lim_{k\to\infty} V_{1,k}(x)$ exists for all x. The same reasoning applies to $V_{2,k}$ to show that $V_{2,\infty}(x) = \lim_{k\to\infty} V_{2,k}(x)$ exists for all x. □

We now show that $V_{1,\infty}$ and $V_{2,\infty}$ both equal $V^*[\mathsf{G}(\wedge_j \mathsf{q}_j \,\mathsf{U}\, \mathsf{r}_j)]$ via double inequality.

Lemma 21.

$$V^*[\mathsf{G}(\wedge_j \mathsf{q}_j \,\mathsf{U}\, \mathsf{r}_j)](x) \leq V_{i,\infty}(x) \quad \textit{for } i = 1,2. \tag{6}$$

Proof. Let $V^*[\mathsf{G}(\wedge_j \mathsf{q}_j \,\mathsf{U}\, \mathsf{r}_j)](x_0) = \lambda$. By definition of the sup in $V^*[\mathsf{G}(\wedge_j \mathsf{q}_j \,\mathsf{U}\, \mathsf{r}_j)]$, for any $\epsilon > 0$, there exists a policy α such that for all $t \geq 0$,

$$\begin{aligned} U_1\big(\xi_{x_0}^{\alpha_{t:\infty}}\big) &\geq \lambda - \epsilon, \\ U_2\big(\xi_{x_0}^{\alpha_{t:\infty}}\big) &\geq \lambda - \epsilon. \end{aligned} \tag{7}$$

Using the recursive relation of U_i,

$$\begin{aligned} &U_i\big(\xi_{x_0}^{\alpha_{t:\infty}}\big) \\ &= \max\big\{r_i\big(\xi_{x_0}^{\alpha}(t)\big), \\ &\qquad \min\big(q_i\big(\xi_{x_0}^{\alpha}(t)\big), U_i\big(\xi_{x_0}^{\alpha_{t+1:\infty}}\big)\big)\big\} \\ &\leq \max\big\{r_i\big(\xi_{x_0}^{\alpha}(t)\big), q_i\big(\xi_{x_0}^{\alpha}(t)\big)\big\} \\ &= w_i\big(\xi_{x_0}^{\alpha}(t)\big). \end{aligned}$$

Hence, (7) implies that under α, $w_i\big(\xi_{x_0}^{\alpha}(t)\big) \geq \lambda - \epsilon$ for all $t \geq 0$.

We now show via induction on k that $V_{1,k}\big(\xi_{x_0}^{\alpha}(t)\big) \geq \lambda - \epsilon$ and $V_{2,k}\big(\xi_{x_0}^{\alpha}(t)\big) \geq \lambda - \epsilon$ for all states visited by α.

a) Base Case ($k = 0$):: By definition, $V_{1,0}(x) = V_{2,0}(x) = \infty \geq \lambda - \epsilon$.

b) Inductive Step:: Assume the statement holds for some k, i.e., for all visited states,

$$V_{2,k}\big(\xi_{x_0}^{\alpha}(t)\big) \geq \lambda - \epsilon. \tag{8}$$

Consider $V_{1,k+1}(x_0)$. Under α, since $U_1\big(\xi_{x_0}^{\alpha_{0:\infty}}\big) \geq \lambda - \epsilon$, there exists some time t where $r_1\big(\xi_{x_0}^{\alpha}(t)\big) \geq \lambda - \epsilon$ and for all $0 \leq \ell < t$, $q_1\big(\xi_{x_0}^{\alpha}(\ell)\big) \geq \lambda - \epsilon$. By the inductive hypothesis, $V_{2,k}\big(\xi_{x_0}^{\alpha}(t+1)\big) \geq \lambda - \epsilon$. Thus,

$$\begin{aligned} &V_{1,k+1}(x_0) \\ &\geq \min\Big\{r_1\big(\xi_{x_0}^{\alpha}(t)\big), w_2\big(\xi_{x_0}^{\alpha}(t)\big), \\ &\qquad V_{2,k}\big(\xi_{x_0}^{\alpha}(t+1)\big), \\ &\qquad \min_{0\leq\ell<t} \min\big\{q_1\big(\xi_{x_0}^{\alpha}(\ell)\big), \\ &\qquad\quad w_2\big(\xi_{x_0}^{\alpha}(\ell)\big)\big\}\Big\} \\ &\geq \lambda - \epsilon. \end{aligned}$$

By symmetry, the same reasoning applies to $V_{2,k+1}(x_0)$. Since $\epsilon > 0$ was arbitrary, we have shown (6). □

Lemma 22.

$$V_{i,\infty}(x) \leq V^*[\mathsf{G}(\wedge_j \mathsf{q}_j \,\mathsf{U}\, \mathsf{r}_j)](x) \quad \textit{for } i = 1,2. \tag{9}$$

Proof. We construct a policy α that achieves a value arbitrarily close to $V_{1,\infty}(x_0)$.

Let $\lambda = V_{1,\infty}(x_0)$, and fix $\epsilon > 0$. Define "slack" variables $\delta_j = \epsilon/2^{j+1}$ for $j = 0, 1, \ldots$, so that $\sum_{j=0}^{\infty} \delta_j = \epsilon$ and $\sum_{j=0}^{N} \delta_j < \epsilon$ for all finite N.

We iteratively construct α by stitching together finite segments. Let $m = j \bmod 2 + 1$ denote the "mode" at switch j. We show that after j switches, the state x_{sw} satisfies

$$V_{m,\infty}(x_{\text{sw}}) \geq \lambda - \textstyle\sum_{i=0}^{j-1} \delta_i,$$

and for all times t between switches,

$$\begin{aligned} U_1\big(\xi_{x_0}^{\alpha_{t:\infty}}\big) &\geq \lambda - \epsilon, \\ U_2\big(\xi_{x_0}^{\alpha_{t:\infty}}\big) &\geq \lambda - \epsilon. \end{aligned}$$

c) Base Case.: At $j = 0$, we begin at x_0 with $V_{1,\infty}(x_0) = \lambda$.

d) Inductive Step.: Suppose after j switches we are at state $\xi^\alpha_{x_0}(t)$ with $m = 1$ (the case $m = 2$ follows by symmetry). Suppose $V_{1,\infty}\big(\xi^\alpha_{x_0}(t)\big) \geq \lambda - \sum_{i=0}^{2j-1} \delta_i$. By definition of $V_{1,\infty}$, there exists a finite time t_1 and policy segment $\alpha_{t:t_1-1}$ such that

- $r_1\big(\xi^\alpha_{x_0}(t_1)\big) \geq \lambda - \sum_{i=1}^{2j} \delta_i$
- $w_2\big(\xi^\alpha_{x_0}(t_1)\big) \geq \lambda - \sum_{i=1}^{2j} \delta_i$
- $V_{2,\infty}\big(\xi^\alpha_{x_0}(t_1+1)\big) \geq \lambda - \sum_{i=1}^{2j} \delta_i$
- $q_1\big(\xi^\alpha_{x_0}(s)\big) \geq \lambda - \sum_{i=1}^{2j} \delta_i$ for all $t \leq s < t_1$
- $w_2\big(\xi^\alpha_{x_0}(s)\big) \geq \lambda - \sum_{i=1}^{2j} \delta_i$ for all $t \leq s < t_1$

Hence, for all τ with $t \leq \tau < t_1$,

$$\begin{aligned}
&U_1\big(\xi^{\alpha_{\tau:\infty}}_{x_0}\big)\\
&\geq \min\Big(r_1\big(\xi^\alpha_{x_0}(t_1)\big),\\
&\qquad \min_{\tau \leq s < t_1} q_1\big(\xi^\alpha_{x_0}(s)\big)\Big)\\
&\geq \lambda - \epsilon.
\end{aligned}$$

For U_2, let $\tau \in [t, t_1 - 1]$. We consider two cases.

1) *There exists t' with $\tau \leq t' < t_1$ and $r_2\big(\xi^\alpha_{x_0}(t')\big) \geq \lambda - \epsilon$.* Let t' be the smallest such time. Since $w_2\big(\xi^\alpha_{x_0}(s)\big) \geq \lambda - \epsilon$ and t' is minimal, we have $q_2\big(\xi^\alpha_{x_0}(s)\big) \geq \lambda - \epsilon$ for all $\tau \leq s < t'$. Hence,

$$\begin{aligned}
&U_2\big(\xi^{\alpha_{\tau:\infty}}_{x_0}\big)\\
&\geq \min\Big(r_2\big(\xi^\alpha_{x_0}(t')\big),\\
&\qquad \min_{\tau \leq s < t'} q_2\big(\xi^\alpha_{x_0}(s)\big)\Big)\\
&\geq \lambda - \epsilon.
\end{aligned}$$

2) *No such t' exists.* Since $U_2\big(\xi^{\alpha_{t+1:\infty}}_{x_0}\big) \geq V_{2,\infty}\big(\xi^\alpha_{x_0}(t+1)\big) \geq \lambda - \epsilon$, there exists $t'' \geq t_1$ with $r_2\big(\xi^\alpha_{x_0}(t'')\big) \geq \lambda - \epsilon$ and $q_2\big(\xi^\alpha_{x_0}(s)\big) \geq \lambda - \epsilon$ for all $t + 1 \leq s < t''$. Since no t' exists, $q_2\big(\xi^\alpha_{x_0}(s)\big) \geq \lambda - \epsilon$ for all $\tau \leq s \leq t_1$. Thus,

$$\begin{aligned}
&U_2\big(\xi^{\alpha_{\tau:\infty}}_{x_0}\big)\\
&= \sup_{s \geq \tau} \min\big\{r_2\big(\xi^\alpha_{x_0}(s)\big),\\
&\qquad \min_{\tau \leq \ell < s} q_2\big(\xi^\alpha_{x_0}(\ell)\big)\big\}\\
&\geq \min\big\{r_2\big(\xi^\alpha_{x_0}(t'')\big),\\
&\qquad \min_{\tau \leq \ell < t''} q_2\big(\xi^\alpha_{x_0}(\ell)\big)\big\}\\
&\geq \lambda - \epsilon.
\end{aligned}$$

Hence, for all τ with $t \leq \tau < t_1$, both U_1 and U_2 are at least $\lambda - \epsilon$. We extend α with the segment $\alpha_{t:t_1-1}$ and transition to $\xi^\alpha_{x_0}(t_1+1)$, where

$$V_{2,\infty}\big(\xi^\alpha_{x_0}(t_1+1)\big) \geq \lambda - \textstyle\sum_{i=1}^{2j} \delta_i.$$

By symmetry, the same holds when $m = 2$. Thus, the inductive step holds.

By induction, at all times t,

$$\begin{aligned}
U_1\big(\xi^{\alpha_{t:\infty}}_{x_0}\big) &\geq \lambda - \epsilon,\\
U_2\big(\xi^{\alpha_{t:\infty}}_{x_0}\big) &\geq \lambda - \epsilon.
\end{aligned}$$

Hence,

$$\begin{aligned}
&V^*[\mathsf{G}(\wedge_j \mathsf{q}_j \,\mathsf{U}\, \mathsf{r}_j)](x_0)\\
&= \sup_\alpha \min\big\{U_1\big(\xi^{\alpha_{0:\infty}}_{x_0}\big),\\
&\qquad U_2\big(\xi^{\alpha_{0:\infty}}_{x_0}\big)\big\}\\
&\geq \lambda - \epsilon = V_{1,\infty}(x_0) - \epsilon.
\end{aligned}$$

Since $\epsilon > 0$ was arbitrary, $V^*[\mathsf{G}(\wedge_j \mathsf{q}_j \,\mathsf{U}\, \mathsf{r}_j)](x_0) \geq V_{1,\infty}(x_0)$. By symmetry, $V^*[\mathsf{G}(\wedge_j \mathsf{q}_j \,\mathsf{U}\, \mathsf{r}_j)](x_0) \geq V_{2,\infty}(x_0)$. This shows (9). □

We are now ready to prove Lem. 18.

Proof. The proof follows directly from (6) and (9). □

H. General Result

Here, we give a proof of the general result given in the main text, restated here.

Theorem 4. *For the predicate*

$$\mathsf{p} := \left(\bigwedge_{i\in\mathcal{I}} (\mathsf{q}_i \,\mathsf{U}\, \mathsf{r}_i)\right) \wedge \mathsf{G}\left(\bigwedge_{j\in\mathcal{J}} (\mathsf{q}_j \,\mathsf{U}\, \mathsf{r}_j)\right) \wedge \mathsf{Gq}$$

the corresponding optimal Value satisfies $V^*[\mathsf{p}](x) = V^*\left[\tilde{\mathsf{q}} \,\mathsf{U}\, \tilde{\mathsf{r}}\right](x)$ *where*

$$\tilde{\mathsf{r}} := \bigvee_i \left(\mathsf{r}_i \wedge \mathsf{v}^*_{\mathsf{p}^{-i}}\right), \qquad \tilde{\mathsf{q}} := \bigwedge_{k\in\mathcal{I}\times\mathcal{J}} \tilde{\mathsf{q}}_k \wedge \mathsf{q},$$

$$\mathsf{p}^{-i} := \bigwedge_{k\in\mathcal{I}\setminus\{i\}} (\mathsf{q}_k \,\mathsf{U}\, \mathsf{r}_k) \wedge \mathsf{G}\left(\bigwedge_{j\in\mathcal{J}} (\mathsf{q}_j \,\mathsf{U}\, \mathsf{r}_j)\right) \wedge \mathsf{Gq}.$$

Proof. The proof simply follows from the same reasoning as in the previous sections, utilizing the established relationships between the various Value functions and their decompositions. Namely, this result follows from a combination of logical rearrangement and then a usage of the algebraic properties of the Bellman equations.

First, we may have by Lem. 8 that p may be rewritten in one of two ways, depending on the remaining index set of Until predicates $\mathcal{J}$. Hence, the proof follows from either case.

a) Non-empty $\mathcal{J}$: In this case we have by Lem. 8, $\mathsf{p} = \tilde{\mathsf{q}} \,\mathsf{U}\, \tilde{\mathsf{r}}$, where $\tilde{\mathsf{r}}$ is given by,

$$\tilde{\mathsf{r}}_{\mathcal{I},\mathcal{J}} := \bigvee_{j\in\mathcal{J}} \mathsf{r}_j \wedge \Phi_{\mathcal{I},\mathcal{J}\setminus\{j\}}.$$

Notably, this case is algebraically equivalent to the previous proofs (e.g. Thm. 1), and hence, by Lemmas 1 and 10, we have the given result.

b) $\mathcal{J} = \emptyset$: In this case we have by Lem. 8,

$$\mathsf{p} \equiv \mathsf{G}\Big(\bigwedge_{i\in\mathcal{I}} (\mathsf{q}_i \wedge \mathsf{q}) \,\mathsf{U}\, (\mathsf{r}_i \wedge \mathsf{q})\Big)$$

On the other hand, this is a special case of the N-$\mathcal{RA}_\ell$ problem, and thus by Thm. 3, we may decompose this into N coupled Until-decompositions. □

I. Policy Results

In this section, we extend the previous results involving the optimal action sequence α to a state-feedback policy $\pi : \mathcal{X} \to \mathcal{A}$. For general TL predicates, the synthesis of a policy that matches open-loop action sequence performance requires state-augmentation [5, 68]. The nature of temporal logic is to score satisfaction over the entire trajectory Hence, to play optimally, the running performance is required. In [5], the authors show that for a reduced set of dual-predicates, the optimal policy may be derived as a function of the augmented-state and each decomposed Value. Here, we generalize these results to the decomposed Value graph that arises in the decomposition of the general predicates considered in this work.

To do so, we introduce the Q function, which defines the value of taking a particular action a at state x, then following the optimal policy thereafter. However, since the optimal policy for temporal logic is history-dependent, we will extend the Q to consider not just the current action, but also the next n actions.

As shown in Thm. 4, the TL can be transformed into a single Until but with a "reach" predicate that involves the value function of a subproblem. Hence, for conciseness, we will first define the Q function and its extensions for the Until case, then show how it can be applied to the general case.

Definition 5. *Consider the formula* $\mathsf{f} := \mathsf{q} \,\mathsf{U}\, \mathsf{r}$ *with atomic predicates* q *and* r. *Define the Q function* $Q[\mathsf{f}]$ *as*

$$Q[\mathsf{f}](x_0, a_0) = \min\Big\{q(x_0), \max\big(r(x_0), V[\mathsf{f}](x_1)\big)\Big\}, \quad \textit{where } x_1 = f(x_0, a_0). \tag{10}$$

Standard properties of the Q function hold, such as

$$V[\mathsf{f}](x) = \max_a Q[\mathsf{f}](x,a). \tag{11}$$

The Q function here has been introduced before in the literature [56]. However, we now introduce an extension of the Q function to consider the next n actions.

Definition 6. *We recursively define the n-step Q function as*

$$Q^{(n)}[\mathsf{f}](x_0, a_0, \dots, a_{n-1}) = \min\Big\{q(x_0), \max\big(r(x_0), Q^{(n-1)}[\mathsf{f}](x_1, a_1, \dots, a_{n-1})\big)\Big\}, \quad \text{where } x_1 = f(x_0, a_0). \tag{12}$$

where $Q^{(0)}[\mathsf{f}](x) := V[\mathsf{f}](x)$.

Note that the n-step Q function is a generalization of the standard Q function, and includes the standard Q function as a special case when $n = 1$ and the Value function as a special case when $n = 0$. We now prove a generalization of (11) to the n-step Q function.

Lemma 23. *For all $n \geq 0$,*

$$Q^{(n)}[\mathsf{f}](x_0, a_0, \dots, a_{n-1}) = \max_{a_n} Q^{(n+1)}[\mathsf{f}](x_0, a_0, \dots, a_{n-1}, a_n). \tag{13}$$

Proof. The proof follows from induction on n.

Base Case ($n = 0$): By definition, $Q^{(0)}[\mathsf{f}](x_0) = V[\mathsf{f}](x_0)$, and by (11), we have

$$V[\mathsf{f}](x_0) = \max_a Q^{(1)}[\mathsf{f}](x_0, a). \tag{14}$$

Inductive Step: Assume the statement holds for some n, i.e.,

$$Q^{(n)}[\mathsf{f}](x_0, a_0, \dots, a_{n-1}) = \max_{a_n} Q^{(n+1)}[\mathsf{f}](x_0, a_0, \dots, a_{n-1}, a_n). \tag{15}$$

Consider $Q^{(n+1)}[\mathsf{f}](x_0, a_0, \dots, a_n)$. By definition,

$$\begin{aligned} &Q^{(n+1)}[\mathsf{f}](x_0, a_0, \dots, a_n) \\ &= \min\Big\{q(x_0), \max\big(r(x_0), Q^{(n)}[\mathsf{f}](x_1, a_1, \dots, a_n)\big)\Big\}, \quad x_1 = f(x_0, a_0). \end{aligned}$$

By the inductive hypothesis,

$$\begin{aligned} &Q^{(n)}[\mathsf{f}](x_1, a_1, \dots, a_n) \\ &= \max_{a_{n+1}} Q^{(n+1)}[\mathsf{f}](x_1, a_1, \dots, a_n, a_{n+1}). \end{aligned}$$

Hence,

$$\begin{aligned} &Q^{(n+1)}[\mathsf{f}](x_0, a_0, \dots, a_n) \\ &= \max_{a_{n+1}} \min\Big\{q(x_0), \max\big(r(x_0), Q^{(n+1)}[\mathsf{f}](x_1, a_1, \dots, a_n, a_{n+1})\big)\Big\} \\ &= \max_{a_{n+1}} Q^{(n+2)}[\mathsf{f}](x_0, a_0, \dots, a_{n+1}). \end{aligned}$$

This completes the inductive step and thus the proof. □

By telescoping the above result, we have the following corollary which relates the n-step Q function to the Value function.

Corollary 5. *For all $n \geq 0$,*

$$V[\mathsf{f}](x_0) = \max_{a_0} \max_{a_1} \dots \max_{a_n} Q^{(n)}[\mathsf{f}](x_0, a_0, a_1, \dots, a_n). \tag{16}$$

Proof. The proof follows from telescoping the previous lemma. □

We can then compute the optimal policy as follows. Suppose, starting at state x_0, we have taken optimal actions $a_0^*, \dots, a_{k-1}^*$ to arrive at state x_k. Then, by (16),

$$V(x_0) = \max_{a_0} \dots \max_{a_k} Q^{(k+1)}[\mathsf{f}](x_0, a_0, \dots, a_{k-1}, a_k). \tag{17}$$

Hence the optimal action a_k^* can be obtained as

$$a_k^* = \arg\max_{a_k} Q^{(k+1)}[\mathsf{f}](x_0, a_0^*, \dots, a_{k-1}^*, a_k). \tag{18}$$

Beyond atomic predicates. The above results are stated for the case of a single Until operator with atomic predicates. However, by the results of the previous sections, we can decompose a general predicate into a graph of coupled Until operators with atomic predicates and Value functions as reach predicates. Without loss of generality, we now consider the formula f_1 defined as

$$\mathsf{f}_1 = \mathsf{q}_1 \,\mathsf{U}\, \Big(\mathsf{r}_1 \wedge \mathsf{f}_0\Big). \tag{19}$$

To define the Q function *correctly*, we start from the relation (11), but for f_1 instead of f, which gives

$$\begin{aligned} V[\mathsf{f}_1](x_0) &= \min\Big\{ q_1(x_0),\, \max\big(r_1(x_0) \wedge V[\mathsf{f}_0](x_0), \max_{a_0} V[\mathsf{f}_1](x_1)\big)\Big\} && (20)\\ &= \min\Big\{ q_1(x_0),\, \max\big(r_1(x_0) \wedge \big(\max_{a_0} Q[\mathsf{f}_0](x_0, a_0)\big), \max_{a_0} V[\mathsf{f}_1](x_1)\big)\Big\} && (21)\\ &= \min\Big\{ q_1(x_0),\, \max_{a_0} \max\big(r_1(x_0) \wedge Q[\mathsf{f}_0](x_0, a_0),\, \max_{a_0} V[\mathsf{f}_1](x_1)\big)\Big\} && (22)\\ &= \max_{a_0} \underbrace{\min\Big\{ q_1(x_0),\, \max\big(r_1(x_0) \wedge Q[\mathsf{f}_0](x_0, a_0),\, V[\mathsf{f}_1](x_1)\big)\Big\}}_{:= Q[\mathsf{f}_1](x_0, a_0)}. && (23) \end{aligned}$$

Note that the first argument of the $\max$ is a function of a_0 since $Q[\mathsf{f}_0]$ is a function of a_0. This is **different** from the previous case with atomic predicates, where the first argument of the $\max$ was only dependent on x_0.

We can now recursively define the n-step Q function by using (16).

Definition 7. *For the formula f_1 defined above, we define the n-step Q function as*

$$Q^{(n)}[\mathsf{f}_1](x_0, a_0, \dots, a_{n-1}) = \min\Big\{ q_1(x_0),\, \max\big(r_1(x_0) \wedge Q^{(n)}[\mathsf{f}_0](x_0, a_0, \dots, a_{n-1}),\, Q^{(n-1)}[\mathsf{f}_1](x_1, a_1, \dots, a_{n-1})\big)\Big\}, \tag{24}$$

where $x_1 = f(x_0, a_0)$ and $Q^{(n)}[\mathsf{f}_0]$ is defined as in the previous section.

We now prove that this definition of the n-step Q function satisfies Lemma 23.

Lemma 24. *For all $n \geq 0$,*

$$Q^{(n)}[\mathsf{f}_1](x_0, a_0, \dots, a_{n-1}) = \max_{a_n} Q^{(n+1)}[\mathsf{f}_1](x_0, a_0, \dots, a_{n-1}, a_n). \tag{25}$$

Proof. The proof follows from induction on n and is similar to the proof of Lemma 23 for the case of atomic predicates, but with the additional consideration of the $Q^{(n)}[\mathsf{f}_0]$ term.

Base Case ($n = 0$): By definition, $Q^{(0)}[\mathsf{f}_1](x_0) = V[\mathsf{f}_1](x_0)$ and $Q^{(1)}[\mathsf{f}_1](x_0, a_0) = Q[\mathsf{f}_1](x_0, a_0)$, so this holds by definition of $Q[\mathsf{f}_1]$ from (23).

Inductive Step: Assume the statement holds for some n, i.e.,

$$Q^{(n)}[\mathsf{f}_1](x_0, a_0, \dots, a_{n-1}) = \max_{a_n} Q^{(n+1)}[\mathsf{f}_1](x_0, a_0, \dots, a_{n-1}, a_n). \tag{26}$$

Consider $Q^{(n+1)}[\mathsf{f}_1](x_0, a_0, \dots, a_n)$. By the inductive hypothesis,

$$Q^{(n)}[\mathsf{f}_1](x_1, a_1, \dots, a_n) = \max_{a_{n+1}} Q^{(n+1)}[\mathsf{f}_1](x_1, a_1, \dots, a_n, a_{n+1}). \tag{27}$$

Hence, by definition of $Q^{(n+1)}[\mathsf{f}_1]$ and using Lemma 23 for $Q^{(n)}[\mathsf{f}_0]$,

$$
\begin{aligned}
&Q^{(n+1)}[\mathsf{f}_1](x_0, a_0, \ldots, a_n) && (28)\\
&= \min\Big\{q_1(x_0),\, \max\big(r_1(x_0) \wedge Q^{(n+1)}[\mathsf{f}_0](x_0, a_0, \ldots, a_n),\, Q^{(n)}[\mathsf{f}_1](x_1, a_1, \ldots, a_n)\big)\Big\}, && (29)\\
&= \min\Big\{q_1(x_0),\, \max\big(r_1(x_0) \wedge Q^{(n+1)}[\mathsf{f}_0](x_0, a_0, \ldots, a_n),\, \max_{a_{n+1}} Q^{(n+1)}[\mathsf{f}_1](x_1, a_1, \ldots, a_{n+1})\big)\Big\}, && (30)\\
&= \min\Big\{q_1(x_0),\, \max\big(r_1(x_0) \wedge \max_{a_{n+1}} Q^{(n+2)}[\mathsf{f}_0](x_0, a_0, \ldots, a_n, a_{n+1}),\, \max_{a_{n+1}} Q^{(n+1)}[\mathsf{f}_1](x_1, a_1, \ldots, a_{n+1})\big)\Big\}, && (31)\\
&= \min\Big\{q_1(x_0),\, \max_{a_{n+1}} \max\big(r_1(x_0) \wedge Q^{(n+2)}[\mathsf{f}_0](x_0, a_0, \ldots, a_n, a_{n+1}),\, Q^{(n+1)}[\mathsf{f}_1](x_1, a_1, \ldots, a_{n+1})\big)\Big\}, && (32)\\
&= \max_{a_{n+1}} \min\Big\{q_1(x_0),\, \max\big(r_1(x_0) \wedge Q^{(n+2)}[\mathsf{f}_0](x_0, a_0, \ldots, a_n, a_{n+1}),\, Q^{(n+1)}[\mathsf{f}_1](x_1, a_1, \ldots, a_{n+1})\big)\Big\}, && (33)\\
&= \max_{a_{n+1}} Q^{(n+2)}[\mathsf{f}_1](x_0, a_0, \ldots, a_n, a_{n+1}). && (34)
\end{aligned}
$$

This completes the inductive step and thus the proof. □

Similar to before, we can use Lemma 24 to relate the n-step Q function to the Value function as follows.

Corollary 6. *For all* $n \geq 0$*,*

$$V[\mathsf{f}_1](x_0) = \max_{a_0} \max_{a_1} \ldots \max_{a_n} Q^{(n)}[\mathsf{f}_1](x_0, a_0, a_1, \ldots, a_n). \tag{35}$$

Proof. The proof follows from telescoping the previous lemma. □

Thus, we can compute the optimal policy for f_1 by using the n-step Q function as follows. Suppose, starting at state x_0, we have taken optimal actions $a_0^*, \ldots, a_{k-1}^*$ to arrive at state x_k. Then, by the previous corollary,

$$V[\mathsf{f}_1](x_0) = \max_{a_0} \ldots \max_{a_k} Q^{(k+1)}[\mathsf{f}_1](x_0, a_0, \ldots, a_k). \tag{36}$$

Hence the optimal action a_k^* can be obtained as

$$a_k^* = \arg\max_{a_k} Q^{(k+1)}[\mathsf{f}_1](x_0, a_0^*, \ldots, a_{k-1}^*, a_k). \tag{37}$$

The optimal action a_k^* can be expressed in terms of the original Q function $Q[\mathsf{f}_1]$ in a recursive manner, as we now show in the following result.

Lemma 25. *For all* $k \geq 0$*, let* $a_0^*, \ldots, a_{k-1}^*$ *be the optimal actions taken from state* x_0 *to arrive at state* x_k*. Now consider the action* $\hat{a}_k$ *computed as*

$$\hat{a}_k \in \begin{cases} \arg\max_{a_k} Q^{(k+1)}[\mathsf{f}_0](x_0, a_0^*, \ldots, a_{k-1}^*, a_k), & r_1(x_0) \wedge V[\mathsf{f}_0](x_0) \geq V[\mathsf{f}_1](x_1) \\ \arg\max_{a_k} Q^{(k)}[\mathsf{f}_1](x_1, a_1^*, \ldots, a_{k-1}^*, a_k), & \textit{otherwise} \end{cases} \tag{38}$$

Then, $\hat{a}_k \in \arg\max_{a_k} Q^{(k+1)}[\mathsf{f}_1](x_0, a_0^*, \ldots, a_{k-1}^*, a_k)$*.*

Proof. From the definition of the n-step Q function and using properties of the $\arg\max$ operator,

$$
\begin{aligned}
&\arg\max_{a_k} Q^{(k+1)}[\mathsf{f}_1](x_0, a_0^*, \ldots, a_{k-1}^*, a_k) && (39)\\
&= \arg\max_{a_k} \min\Big\{q_1(x_0),\, \max\big(r_1(x_0) \wedge Q^{(k+1)}[\mathsf{f}_0](x_0, a_0^*, \ldots, a_k),\; Q^{(k)}[\mathsf{f}_1](x_1, a_1^*, \ldots, a_k)\big)\Big\} && (40)\\
&\supseteq \arg\max_{a_k} \max\Big(r_1(x_0) \wedge Q^{(k+1)}[\mathsf{f}_0](x_0, a_0^*, \ldots, a_k),\; Q^{(k)}[\mathsf{f}_1](x_1, a_1^*, \ldots, a_k)\Big) && (41)\\
&\supseteq \begin{cases} \arg\max_{a_k} r_1(x_0) \wedge Q^{(k+1)}[\mathsf{f}_0](x_0, a_0^*, \ldots, a_k), & \max_{a_k} r_1(x_0) \wedge Q^{(k+1)}[\mathsf{f}_0](x_0, a_0^*, \ldots, a_k) \geq \max_{a_k} Q^{(k)}[\mathsf{f}_1](x_1, a_1^*, \ldots, a_k) \\ \arg\max_{a_k} Q^{(k+1)}[\mathsf{f}_1](x_1, a_1^*, \ldots, a_k), & \text{otherwise} \end{cases} && (42)
\end{aligned}
$$

Note that

$$\max_{a_k} r_1(x_0) \wedge Q^{(k+1)}[\mathsf{f}_0](x_0, a_0^*, \dots, a_k) = r_1(x_0) \wedge \max_{a_k} Q^{(k+1)}[\mathsf{f}_0](x_0, a_0^*, \dots, a_k), \tag{43}$$

$$= r_1(x_0) \wedge V[\mathsf{f}_0](x_0). \tag{44}$$

and

$$\max_{a_k} Q^{(k)}[\mathsf{f}_1](x_1, a_1^*, \dots, a_k) = V[\mathsf{f}_1](x_1). \tag{45}$$

Hence,

$$\arg\max_{a_k} Q^{(k+1)}[\mathsf{f}_1](x_0, a_0^*, \dots, a_{k-1}^*, a_k) \tag{46}$$

$$\supseteq \begin{cases} \arg\max_{a_k} r_1(x_0) \wedge Q^{(k+1)}[\mathsf{f}_0](x_0, a_0^*, \dots, a_k), & r_1(x_0) \wedge V[\mathsf{f}_0](x_0) \geq V[\mathsf{f}_1](x_1) \\ \arg\max_{a_k} Q^{(k)}[\mathsf{f}_1](x_1, a_1^*, \dots, a_k), & \text{otherwise} \end{cases} \tag{47}$$

$$\supseteq \begin{cases} \arg\max_{a_k} Q^{(k+1)}[\mathsf{f}_0](x_0, a_0^*, \dots, a_k), & r_1(x_0) \wedge V[\mathsf{f}_0](x_0) \geq V[\mathsf{f}_1](x_1) \\ \arg\max_{a_k} Q^{(k)}[\mathsf{f}_1](x_1, a_1^*, \dots, a_k), & \text{otherwise} \end{cases} \tag{48}$$

Thus, for any $\hat{a}_k$ taken from the set on the right-hand side, this implies that $\hat{a}_k$ is also in the set on the left-hand side, which completes the proof. □

Lemma 25 enables us to compute the optimal action at time k using the $\arg\max$ of a $k+1$-step Q function by either taking the $\arg\max$ of the $k+1$-step Q function for f_0, a simpler subproblem, or the n-step Q function for f_1 the original problem with one fewer step depending on the comparison of the two terms. The base case is reached when either we reach the $\arg\max$ of the 1-step Q function for either f_0 or f_1, which can be computed directly without recursion.

Solving the general problem. The above results show how to compute the optimal policy for a single Until formula with a nested Until formula as the reach predicate. Note, however, that nowhere in the previous section did we rely on the fact that f_0 was an Until formula with atomic predicates, and the results hold for any formula f_0 for which we can define a n-step Q function. We have shown how to define the n-step Q function for a single Until formula with atomic predicates. The same can be done for Globally formulas, as well as for disjunctions of Untils.

Hence, by the results of the previous sections, we can apply Lemma 25 recursively to compute the optimal policy for any formula that can be decomposed into a graph of coupled Until formulas with atomic predicates and Value functions as reach predicates, which includes all formulas in our logic by Thm. 4.

Minimizing the required information. Note that, using Lemma 25 to compute the optimal action at time k requires comparing the sign of two terms at all previous time steps, which may require keeping track of the entire state trajectory history up to time k. However, we can minimize the amount of information that needs to be tracked by noting that the *same* comparison is made at all previous time steps. For example, for any value of k, the first comparison is always between $r_1(x_0) \wedge V[\mathsf{f}_0](x_0)$ and $V[\mathsf{f}_1](x_1)$. The result of this comparison does not change since the states x_0 and x_1 will have been in the past for $k \geq 1$. Similarly, if the result of this comparison then next asks for $\arg\max_{a_k} Q^{(k)}[\mathsf{f}_1](x_1, a_1^*, \dots, a_k)$, then the next comparison will always be between $r_1(x_1) \wedge V[\mathsf{f}_0](x_1)$ and $V[\mathsf{f}_1](x_2)$, and the result of this comparison will also not change for all $k \geq 2$.

This thus defines a tree of comparisons that can be pre-computed at the beginning of the episode, and the optimal action at time k can be computed by traversing this tree of comparisons to find the correct Q function to use for computing the optimal action, without needing to keep track of the entire state trajectory history.

J. VALTR DETAILS

In this section, we describe our tool `valtr`, that (1.) converts temporal logic predicates into a suitable form for decomposition, and (2.) applies the main results recursively to generate the decomposed Value graph.

To decompose the Value for a user-input predicate, the predicate must first be organized into the form given in Thm. 4. This is accomplished by lexing the temporal logic string into relevant tokens, such as atomic propositions and temporal operators, which may then be parsed to generate an abstract syntax tree (AST), which is thus a type of TL Tree (TLT). Over this AST, several passes are made to rearrange the tree into an intermediate representation. This rearrangement is accomplished by first applying well-known logical equivalences and then followed by cleaning (e.g. aggregating redundancies). The ultimate product is a TLT with structure that is amenable to the decompositional results.

To apply the main results recursively and generate the decomposed Value graph, we traverse the TLT and for each node, we apply the decomposition procedure outlined in Thm. 4. This involves identifying the relevant substructures, including constants (atomic predicates), negations, minima, maxima, and nodes which represent Value functions. After final cleaning passes, the resulting decomposed Value graph (DVG) is outputted, defining a topological order of nodes, which may be queried to assess a trajectory as well as identify dependencies, and thus suffices for dynamic programming and VDPPO.

K. VDPPO DETAILS

In this section we further describe our algorithm, VDPPO. VDPPO is a specialized form of PPO [79], designed to leverage the decomposed Value graph (DVG). We outline the two augmentations that distinguish it from standard PPO here.

1. The advantage and targets are solved with $\mathcal{A}$, $\mathcal{RA}$, and $\mathcal{RA}_\ell$ Bellman eqns. and bootstrapped Values. As given by the main results, the Bellman Value for a complex TL predicate may be decomposed into a graph of Bellman Values, connected by these atomic BEs. Hence, the Value at each node in the DVG may be approximated in the limit of discounting by the appropriate BE as a function of its dependencies: its decomposed sub-Values and the relevant predicates. To avoid topographically sequential approximation, we use the current Value approximations of the critic to solve these updates. This is denoted by the feedback loop in Fig. 4.

2. Nodes are embedded, allowing for a unified representation for each actor and critic We hypothesize that different Values in the DVG may share some similarity, implying the policies do as well, and thus may be jointly approximated by a single representation. Namely, we augment the states with a current Value node and - with a one-hot encoding - condition the MLP for each actor and critic on mixed-node batches. We validate this hypothesis and design choice in the ablations in Sec. N, demonstrating this yields equivalent performance while vastly improving the scaling ability compared to previous approaches [5].

Additionally, for live roll-outs and evaluation, we define the policy such that upon satisfying the trigger condition given in Sec.I, the current Value node switches to the triggered node in the current augmented state.

L. ENVIRONMENTS

We give here additional details on the environments tested in this work. The reader may refer to the main text for graphics and specs. We will publish all code after the anonymous stage of review is complete.

`DoubleInt`: The `DoubleInt` env is defined by up to N agents with 2-dimensional double integrator dynamics and velocity-tracking control. Namely, for each agent, the discrete action sets a desired velocity which is then tracked by a proportional controller in the acceleration (with $k_p = 1$). The possible discrete actions correspond to ± 1 per dimension, multiplied by the max acceleration. Velocity and acceleration limits are set per-agent. In the three sub-envs, `Breadth`, `Depth`, `Agents (dim.)`, we vary the number of targets to reach (any order), the number of targets to reach sequentially, and the number of agents and number of targets to reach (any order) respectively. In all cases, we define a set of obstacles for which all specifications involve avoid predicates.

`Herding`: The `Herding` env is an augmentation of the `DoubleInt` env, where we have a team of two agents (the herders) and multiple sheep agents (the herd). The sheep agents are defined by their own fixed policy which samples an action which maximizes the weighted soft-min of their distance to the herders, the walls and each another. The herders are defined such that one is twice as fast as the other, while the herders move at a maximum speed equivalent to the slow herder. A narrow gap divides the herders from the herd initially, as well as the target location of the herd and their initial position. The goal of the task is defined by moving the herd through the narrow passage toward the target region on the otherside and contain them there, while avoiding obstacles and collision. This additionally two intermediate goals to have the herd before the passage, and then to have the herd after the passage, which must be achieved sequentially. The full specification is given in the main text.

`Delivery`: The `Delivery` env is an augmentation of the `DoubleInt` env, where we have a team of three agents – two small, fast agents (the delivery robots) and one, big slow agent (the resupply truck) – and randomly spawning targets (delivery locations). The goal of the task is for the agents to recurrently reach the target locations and then recurrently visit the resupply truck. After a delivery target is reached by the corresponding agent, the location jumps to a new random location. Additionally, the domain is defined with the same obstacles used in the `DoubleInt` env, and the team must avoid collision with the obstacles and one another, despite both needing to resupply at the mobile agent. All agents are mobile and hence the truck agent may dynamically adjust its location to suit the current positions. Note, this simulated env differs from the hardware version, which includes a different obstacle layout as well as an additional aerial obstacle (no fly zone).

`Manipulator`: The `Manipulator` env is taken from [80], and involves a manipulator which must grasp and interact with objects in the environment. The specification for this task is to place the cube inside the drawer and eventually always have the drawer closed. Additional objects exist in the environment but have no relevance to task completion.

M. Baselines

In this section we discuss the baselines employed in this work.

LCRL: This baseline [69] is a deep RL method that augments the MDP with an automata for learning TL solutions. Specifically, an actor-critic variation of PPO is designed such that they are conditioned on the automaton and the current state of an augmented trajectory. As this is just another variation of PPO, we employ the same parameter set as used in VDPPO for a fair comparison.

TL-MPPI: This baseline is an extension of Model Predictive Path Integral (MPPI) [70] to tackle TL problems [71], which we denote TL-MPPI. Namely, this method plans a trajectory based on MPPI sample-based optimization of the TL robustness metric. The method in the work does not function adaptively as the controller has no memory without state-augmentation or automaton, however, we employ it as a trajectory optimization method which the agent then tracks. The parameters that worked best in the given environments included: 1000 samples per step, a horizon of 100 steps, 20 iterations per step, an initial standard deviation of 50, $\lambda = 1$, and an iteration temperature (shrink) parameter of 0.6.

N. Ablations

Here, we provide additional ablation experiments to analyze the design of our algorithm, VDPPO. In [5], authors similarly derived decompositional Value results, although for a greatly reduced set of predicates, and then faced the practical question of how to employ these results to learn the critics (Value estimates) effectively, deciding to use a different actor and critic for each decomposition. While this performed well for the dual-specifications that were considered, this approach scales poorly to tasks with complex logic, as the number of required actors and critics can grow combinatorially (see Thm.1).

Moreover, while Values can vary significantly for different rewards and specifications, in many practical cases, tasks often involve different sub-tasks which themselves differ only by translation (e.g. identical configuration goals in different locations), order (e.g. iteratively unlock doors with keys) or other simple transformation or symmetry. Under certain variations, the resulting Bellman Value may indeed differ only by the same transformation. In such cases, a partial consolidation of the representations may accomplish sufficient approximation while greatly reducing the learning challenge.

In VDPPO, we employed this idea, by embedding all Values into a shared space with the one-hot encoding to allow the actor and critic to to each use a shared MLP trunk (see Section K for details). To analyze the importance of this design choice, we compare against a version of VDPPO where each critic and actor has its own separate MLP trunk (i.e. no shared parameters). Moreover, we scan this comparison over an increasing range in the number of layers in the shared trunk (or each independent body, when not shared), to analyze the importance of the depth of the shared representation. The results are plotted in Fig. 9.

From a performance-only perspective, we find that sharing parameters for the value function alone erodes success rate while sharing parameters for the actor boosts success rate, and when combined, we observe performance that is nearly identical to performance without sharing. This result is inspiring as the shared architectures train nearly N-times faster than the standard approach employed in [5], where N is the quantity of decompositions.

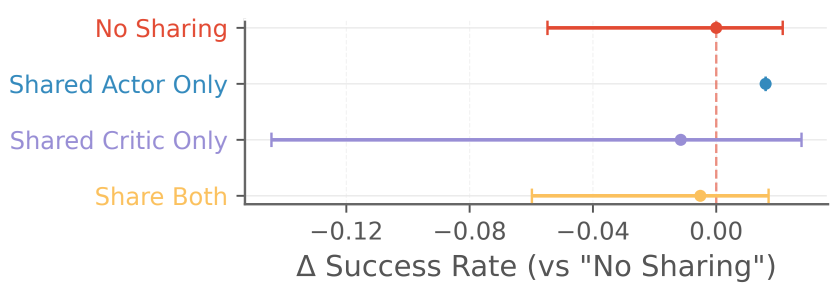

Fig. 9: **Effect of parameter sharing.** Sharing parameters for the actor only improves performance while reduce the variance.

O. Hardware

In the hardware experiments, we evaluate VDPPO performance in the Herding and Delivery tasks. In both tasks, the state position is reported by HTC Vive base stations in communication with the an attached Lighthouse deck to each Crazyflie. The Go2 quadruped's location is integrated into the same framework by attaching a propeller-less Crazyflie to its chassis, which transmits its position data to a single computer. The state of each agent is concatenated to form the full state used by the VDPPO policy, which is inferred on the local CPU of the coordinating laptop. The output action velocity commands are broadcasted to each agent's onboard controller, which tracks the transmitted velocity setpoint.

REFERENCES

[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: A Bradford Book, 2018.

[2] A. Camacho, R. Toro Icarte, T. Q. Klassen, R. Valenzano, and S. A. McIlraith, "LTL and beyond: Formal languages for reward function specification in reinforcement learning," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*. California: International Joint Conferences on Artificial Intelligence Organization, 1 Aug. 2019, pp. 6065–6073.

[3] I. M. Mitchell, A. M. Bayen, and C. J. Tomlin, "A time-dependent hamilton-jacobi formulation of reachable sets for continuous dynamic games," *IEEE Transactions on automatic control*, vol. 50, no. 7, pp. 947–957, 2005.

[4] J. F. Fisac, M. Chen, C. J. Tomlin, and S. S. Sastry, "Reach-avoid problems with time-varying dynamics, targets and constraints," in *Hybrid Systems: Computation and Control*. ACM, 2015.

[5] W. Sharpless, D. Hirsch, S. Tonkens, N. Shinde, and S. Herbert, "Dual-objective reinforcement learning with novel hamilton-jacobi-bellman formulations," *arXiv preprint arXiv:2506.16016*, 2025.

[6] M. H. Cohen, Z. Serlin, K. Leahy, and C. Belta, "Temporal logic guided safe model-based reinforcement learning: A hybrid systems approach," *Nonlinear Anal. Hybrid Syst.*, vol. 47, no. 101295, p. 101295, Feb. 2023.

[7] W. Qiu, W. Mao, and H. Zhu, "Instructing goal-conditioned reinforcement learning agents with temporal logic objectives," *Neural Inf Process Syst*, vol. 36, pp. 39 147–39 175, 2023.

[8] T. Brázdil, K. Chatterjee, M. Chmelík, V. Forejt, J. Křetínský, M. Kwiatkowska, D. Parker, and M. Ujma, "Verification of Markov decision processes using learning algorithms," *arXiv [cs.LO]*, 10 Feb. 2014.

[9] N. Hamilton, P. K. Robinette, and T. T. Johnson, "Training agents to satisfy timed and untimed signal temporal logic specifications with reinforcement learning," in *Software Engineering and Formal Methods*, ser. Lecture notes in computer science. Cham: Springer International Publishing, 2022, pp. 190–206.

[10] D. Sadigh, E. S. Kim, S. Coogan, S. S. Sastry, and S. A. Seshia, "A learning based approach to control synthesis of Markov decision processes for linear temporal logic specifications," in *53rd IEEE Conference on Decision and Control*. IEEE, Dec. 2014, pp. 1091–1096.

[11] A. K. Bozkurt, Y. Wang, M. M. Zavlanos, and M. Pajic, "Control synthesis from linear temporal logic specifications using model-free reinforcement learning," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2020, p. 10349–10355.

[12] F. Bacchus, C. Boutilier, and A. J. Grove, "Rewarding behaviors," in *Proceedings of the National Conference on Artificial Intelligence.* cs.toronto.edu, 4 Aug. 1996, pp. 1160–1167.

[13] S. Thiebaux, C. Gretton, J. Slaney, D. Price, and F. Kabanza, "Decision-theoretic planning with non-Markovian rewards," *J. Artif. Intell. Res.*, vol. 25, pp. 17–74, 29 Jan. 2006.

[14] A. Camacho, O. Chen, S. Sanner, and S. McIlraith, "Non-Markovian rewards expressed in LTL: Guiding search via reward shaping," *Proceedings of the International Symposium on Combinatorial Search*, vol. 8, no. 1, pp. 159–160, 1 Sep. 2021.

[15] R. T. Icarte, T. Q. Klassen, R. Valenzano, and S. A. McIlraith, "Using reward machines for high-level task specification and decomposition in reinforcement learning," *ICML*, vol. 80, pp. 2112–2121, 3 Jul. 2018.

[16] D. Aksaray, A. Jones, Z. Kong, M. Schwager, and C. Belta, "Q-learning for robust satisfaction of signal temporal logic specifications," in *2016 IEEE 55th Conference on Decision and Control (CDC)*. IEEE, Dec. 2016, pp. 6565–6570.

[17] X. Li, C.-I. Vasile, and C. Belta, "Reinforcement learning with temporal logic rewards," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Sep. 2017, pp. 3834–3839.

[18] M. Cai, M. Hasanbeig, S. Xiao, A. Abate, and Z. Kan, "Modular deep reinforcement learning for continuous motion planning with temporal logic," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, p. 7973–7980, Oct. 2021. [Online]. Available: http://dx.doi.org/10.1109/LRA.2021.3101544

[19] R. Wang, P. Zhong, S. S. Du, R. R. Salakhutdinov, and L. Yang, "Planning with general objective functions: Going beyond total rewards," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 14 486–14 497.

[20] W. Cui and W. Yu, "Reinforcement learning with non-cumulative objective," *IEEE Transactions on Machine Learning in Communications and Networking*, vol. 1, pp. 124–137, 2023.

[21] Y. Tang, Y. Zhang, J. Ackermann, Y.-J. Zhang, S. Nishimori, and M. Sugiyama, "Recursive reward aggregation," in *Reinforcement Learning Conference*, 2025.

[22] H. van Seijen, M. Fatemi, J. Romoff, R. Laroche, T. Barnes, and J. Tsang, "Hybrid reward architecture for reinforcement learning," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., 2017, p. 5398–5408.

[23] S. Pitis, "Consistent aggregation of objectives with diverse time preferences requires non-markovian rewards," in *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[24] Z. Lin, D. Yang, L. Zhao, T. Qin, G. Yang, and T.-Y. Liu, "Rdˆ2: Reward decomposition with representation decomposition,"

in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 11 298–11 308.
[25] E. Altman, *Constrained Markov decision processes: Stochastic modeling*. Boca Raton: Routledge, 13 Dec. 2021.
[26] J. Achiam, D. Held, A. Tamar, and P. Abbeel, “Constrained policy optimization,” *ICML*, vol. abs/1705.10528, pp. 22–31, 30 May 2017.
[27] A. Wachi and Y. Sui, “Safe reinforcement learning in constrained Markov decision processes,” *ICML*, vol. 119, pp. 9797–9806, 12 Jul. 2020.
[28] A. Stooke, J. Achiam, and P. Abbeel, “Responsive safety in reinforcement learning by PID lagrangian methods,” *ICML*, vol. 119, pp. 9133–9143, 8 Jul. 2020.
[29] T. Li, Z. Guan, S. Zou, T. Xu, Y. Liang, and G. Lan, “Faster algorithm and sharper analysis for constrained Markov decision process,” *Oper. Res. Lett.*, vol. 54, no. 107107, p. 107107, May 2024.
[30] Y. Chen, J. Dong, and Z. Wang, “A primal-dual approach to constrained Markov decision processes,” *arXiv [math.OC]*, 26 Jan. 2021.
[31] S. Miryoosefi and C. Jin, “A simple reward-free approach to constrained reinforcement learning,” *ICML*, vol. abs/2107.05216, pp. 15 666–15 698, 12 Jul. 2021.
[32] T.-Y. Yang, J. Rosca, K. Narasimhan, and P. J. Ramadge, “Projection-based constrained policy optimization,” *arXiv [cs.LG]*, 7 Oct. 2020.
[33] D. Ding, K. Zhang, T. Başar, and M. Jovanović, “Natural policy gradient primal-dual method for constrained Markov decision processes,” *Neural Inf Process Syst*, vol. 33, pp. 8378–8390, 2020.
[34] C. Tessler, D. J. Mankowitz, and S. Mannor, “Reward constrained policy optimization,” *arXiv [cs.LG]*, 28 May 2018.
[35] A. Gattami, Q. Bai, and V. Aggarwal, “Reinforcement learning for constrained Markov decision processes,” *AISTATS*, vol. 130, pp. 2656–2664, 2021.
[36] H. Satija, P. Amortila, and J. Pineau, “Constrained Markov decision processes via backward value functions,” *ICML*, vol. 119, pp. 8502–8511, 12 Jul. 2020.
[37] A. Castellano, H. Min, E. Mallada, and J. A. Bazerque, “Reinforcement learning with almost sure constraints,” in *Proceedings of The 4th Annual Learning for Dynamics and Control Conference*, ser. Proceedings of Machine Learning Research, vol. 168. PMLR, 2022, pp. 559–570.
[38] J. McMahan and X. Zhu, “Anytime-constrained reinforcement learning,” in *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, S. Dasgupta, S. Mandt, and Y. Li, Eds., vol. 238. PMLR, 02–04 May 2024, pp. 4321–4329.
[39] M. A. Wiering, M. Withagen, and M. M. Drugan, “Model-based multi-objective reinforcement learning,” in *2014 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*. IEEE, Dec. 2014, pp. 1–6.
[40] M. K. Van and A. Nowé, “Multi-objective reinforcement learning using sets of Pareto dominating policies,” *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 3483–3512, 2014.
[41] X.-Q. Cai, P. Zhang, L. Zhao, J. Bian, M. Sugiyama, and A. Llorens, “Distributional Pareto-optimal multi-objective reinforcement learning,” *Neural Inf Process Syst*, vol. 36, pp. 15 593–15 613, 2023.
[42] H. Mossalam, Y. M. Assael, D. M. Roijers, and S. Whiteson, “Multi-objective deep reinforcement learning,” *arXiv [cs.AI]*, 9 Oct. 2016.
[43] A. Abels, D. Roijers, T. Lenaerts, A. Nowé, and D. Steckelmacher, “Dynamic weights in multi-objective deep reinforcement learning,” in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 2019, pp. 11–20.
[44] R. Yang, X. Sun, and K. Narasimhan, “A generalized algorithm for multi-objective reinforcement learning and policy adaptation,” in *Advances in Neural Information Processing Systems*. proceedings.neurips.cc, 2019.
[45] E. Liu, Y.-C. Wu, X. Huang, C. Gao, R.-J. Wang, K. Xue, and C. Qian, “Pareto set learning for multi-objective reinforcement learning,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, no. 18, 2025.
[46] M. Liu, M. Zhu, and W. Zhang, “Goal-conditioned reinforcement learning: Problems and solutions,” *arXiv [cs.AI]*, 20 Jan. 2022.
[47] M. Plappert, M. Andrychowicz, A. Ray, B. McGrew, B. Baker, G. Powell, J. Schneider, J. Tobin, M. Chociej, P. Welinder, V. Kumar, and W. Zaremba, “Multi-goal reinforcement learning: Challenging robotics environments and request for research,” *arXiv [cs.LG]*, 26 Feb. 2018.
[48] Z. Ren, K. Dong, Y. Zhou, Q. Liu, and J. Peng, “Exploration via hindsight goal generation,” *Neural Inf Process Syst*, vol. 32, pp. 13 464–13 474, 1 Jun. 2019.
[49] J. Y. Ma, J. Yan, D. Jayaraman, and O. Bastani, “Offline goal-conditioned reinforcement learning via f-advantage regression,” in *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., vol. 35. Curran Associates, Inc., 2022, pp. 310–323.
[50] A. Campero, R. Raileanu, H. Küttler, J. B. Tenenbaum, T. Rocktäschel, and E. Grefenstette, “Learning with AMIGo:

Adversarially motivated intrinsic goals," *arXiv [cs.LG]*, 22 Jun. 2020.
[51] A. R. Trott, S. Zheng, C. Xiong, and R. Socher, "Keeping your distance: Solving sparse reward tasks using self-balancing shaped rewards," *Neural Inf Process Syst*, vol. abs/1911.01417, 4 Nov. 2019.
[52] B. Eysenbach, T. Zhang, R. Salakhutdinov, and S. Levine, "Contrastive learning as goal-conditioned reinforcement learning," *Neural Inf Process Syst*, vol. abs/2206.07568, pp. 35 603–35 620, 15 Jun. 2022.
[53] E. Chane-Sane, C. Schmid, and I. Laptev, "Goal-conditioned reinforcement learning with imagined subgoals," *ICML*, vol. abs/2107.00541, pp. 1430–1440, 1 Jul. 2021.
[54] M. Chen, Q. Tam, S. C. Livingston, and M. Pavone, "Signal temporal logic meets reachability: Connections and applications," in *International Workshop on the Algorithmic Foundations of Robotics*. Springer, 2018, pp. 581–601.
[55] O. So, C. Ge, and C. Fan, "Solving minimum-cost reach avoid using reinforcement learning," in *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. [Online]. Available: https://openreview.net/forum?id=jzngdJQ2lY
[56] K.-C. Hsu, V. Rubies-Royo, C. J. Tomlin, and J. F. Fisac, "Safety and liveness guarantees through reach-avoid reinforcement learning," in *Proceedings of Robotics: Science and Systems*, Held Virtually, July 2021.
[57] J. F. Fisac, N. F. Lugovoy, V. Rubies-Royo, S. Ghosh, and C. J. Tomlin, "Bridging hamilton-jacobi safety analysis and reinforcement learning," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 8550–8556.
[58] M. Ganai, C. Hirayama, Y.-C. Chang, and S. Gao, "Learning stabilization control from observations by learning lyapunov-like proxy models," *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023.
[59] D. Yu, H. Ma, S. Li, and J. Chen, "Reachability constrained reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2022, pp. 25 636–25 655.
[60] K. Zhu, F. Lan, W. Zhao, and T. Zhang, "Safe multi-agent reinforcement learning via approximate hamilton-jacobi reachability," *J. Intell. Robot. Syst.*, vol. 111, no. 1, 30 Dec. 2024.
[61] O. Maler and D. Nickovic, "Monitoring temporal properties of continuous signals," in *International symposium on formal techniques in real-time and fault-tolerant systems*. Springer, 2004, pp. 152–166.
[62] A. Donzé and O. Maler, "Robust satisfaction of temporal logic over real-valued signals," in *International conference on formal modeling and analysis of timed systems*. Springer, 2010, pp. 92–106.
[63] S. Bansal, M. Chen, S. Herbert, and C. J. Tomlin, "Hamilton-jacobi reachability: A brief overview and recent advances," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE, 2017, pp. 2242–2253.
[64] M. Ganai, Z. Gong, C. Yu, S. Herbert, and S. Gao, "Iterative reachability estimation for safe reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 36, pp. 69 764–69 797, 2023.
[65] C. Baier and J.-P. Katoen, *Principles of model checking*. MIT press, 2008.
[66] O. Grumberg, E. Clarke, and D. Peled, "Model checking," in *International Conference on Foundations of Software Technology and Theoretical Computer Science; Springer: Berlin/Heidelberg, Germany*, 1999.
[67] G. Bombara, C.-I. Vasile, F. Penedo, H. Yasuoka, and C. Belta, "A decision tree approach to data classification using signal temporal logic," in *Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control*, 2016, pp. 1–10.
[68] Y. Meng, F. Chen, and C. Fan, "Tgpo: Temporal grounded policy optimization for signal temporal logic tasks," *arXiv preprint arXiv:2510.00225*, 2025.
[69] M. Hasanbeig, D. Kroening, and A. Abate, "Lcrl: Certified policy synthesis via logically-constrained reinforcement learning," in *International Conference on Quantitative Evaluation of SysTems*. Springer, 2022, pp. 217–231.
[70] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Aggressive driving with model predictive path integral control," in *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2016, pp. 1433–1440.
[71] P. Halder, H. Homburger, L. Kiltz, J. Reuter, and M. Althoff, "Trajectory planning with signal temporal logic costs using deterministic path integral optimization," *arXiv preprint arXiv:2503.01476*, 2025.
[72] L. Yifru and A. Baheri, "Concurrent learning of control policy and unknown safety specifications in reinforcement learning," *IEEE Open Journal of Control Systems*, vol. 3, pp. 266–281, 2024.
[73] D. Kasenberg and M. Scheutz, "Interpretable apprenticeship learning with temporal logic specifications," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, 2017, pp. 4914–4921.
[74] M. Gaon and R. Brafman, "Reinforcement learning with non-markovian rewards," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, pp. 3980–3987, Apr. 2020.
[75] K. Jothimurugan, S. Bansal, O. Bastani, and R. Alur, "Compositional reinforcement learning from logical specifications," in *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., vol. 34. Curran Associates, Inc., 2021, pp. 10 026–10 039.
[76] A. Duret-Lutz, E. Renault, M. Colange, F. Renkin, A. Gbaguidi Aisse, P. Schlehuber-Caissier, T. Medioni, A. Martin, J. Dubois, C. Gillard *et al.*, "From spot 2.0 to spot 2.10: What's new?" in *International Conference on Computer Aided Verification*. Springer, 2022, pp. 174–187.

[77] R. Diestel, *Graph theory*. Springer Nature, 2025.
[78] W. Rudin, "Principles of mathematical analysis," *3rd ed.*, 1976.
[79] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint arXiv:1506.02438*, 2015.
[80] S. Park, K. Frans, B. Eysenbach, and S. Levine, "Ogbench: Benchmarking offline goal-conditioned rl," in *International Conference on Learning Representations (ICLR)*, 2025.