2023 62nd IEEE Conference on Decision and Control (CDC)
December 13-15, 2023. Marina Bay Sands, Singapore

Pointwise-in-Time Explanation for Linear Temporal Logic Rules

Noel Brindise and Cédric Langbort

Abstract— **The new field of Explainable Planning (XAIP) has produced a variety of approaches to explain and describe the behavior of autonomous agents to human observers. Many summarize agent behavior in terms of the constraints, or "rules," which the agent adheres to during its trajectories. In this work, we narrow the focus from summary to specific moments in individual trajectories, offering a "pointwise-in-time" view. Our novel framework, which we define on Linear Temporal Logic (LTL) rules, assigns an intuitive *status* to any rule in order to describe the trajectory progress at individual time steps; here, a rule is classified as *active*, *satisfied*, *inactive*, or *violated*. Given a trajectory, a user may query for status of specific LTL rules at individual trajectory time steps.**

In this paper, we present this novel framework, named Rule Status Assessment (RSA), and provide an example of its implementation. We find that pointwise-in-time status assessment is useful as a post-hoc diagnostic, enabling a user to systematically track the agent's behavior with respect to a set of rules.

I. Introduction

Planning and performing on their own, autonomous agents are useful in a wide variety of roles, from self-driving vehicles [1] and task-executing robots [2] to decision-making software [3]. However, it can be challenging to understand or characterize agent behavior. The emerging field of Explainable Planning (XAIP), a subfield of Explainable AI (xAI), takes on this explanatory challenge with mixed results [4]–[6]. Some approaches allow the user to query hypotheticals ("What would [agent] do in [state]?", etc.) [7]; others generate domain alterations that would cause hypothetical route changes ("What needs to change to make [plan] optimal instead?") [8] or use visualizations to help reconcile a human's mental model with that of the agent ("Where does the agent's behavior differ from my expectations?") [9].

A large body of work explains agent plans as a whole using *rule inference*, an approach originating in formal methods and software model checking [10]. Here, an agent is observed, and the explainer infers which "rules" or constraints the agent tends to follow ("What does the agent usually, or never, do?"). Rules may be expressed in signal temporal logic (STL) [11], [12] or, more commonly, linear temporal logic (LTL) [13], [14]. Approaches have learned contrasts between acceptable and unsatisfactory plans [15], inferred rules from purely positive behavior [16] and noisy data [17]. In general, rule inference provides global summaries of agent behavior by providing a set of satisfied (or violated) rules.

This research was funded in part by a National Defense Science and Engineering Graduate Fellowship.

N. Brindise and C. Langbort are with the Department of Aerospace Engineering, Grainger College of Engineering, University of Illinois Urbana-Champaign, 104 S Wright St, Urbana, IL 61801, USA nbrindi2@illinois.edu, langbort@illinois.edu

For live assessment, the related field of LTL *monitoring* predicts the future satisfaction of rules at a current point in a(n unfinished) trajectory [18]; similar methods have been used for STL rules in dynamic planning [19].

While global summaries produced by inference and monitoring are certainly valuable, they do not address local behavior of the agent, which evokes other questions ("Which rules is the agent following at time t? Which rules are not relevant at t?"). We may ask, for example, whether a rule is still in progress at t or already completed. Moreover, implication-type ($x \rightarrow y$) rules require a consequence y only after a triggering condition x occurs, making the rule trivially satisfied when x does not occur; we may want to know which rules are thus "triggered" at a given t.

In our work, we pursue an aspect of local, time step dependent diagnostics which, to the best of the authors' knowledge, remains unaddressed. We consider trajectories of an arbitrary, black-box agent and define a notion of rule *status*, which we use to assess when and how a system trajectory progresses through each behavior (LTL rule) of interest. This is accomplished by defining notions of *active*, *inactive*, *satisfied*, and *violated* rule status, such that rules are uniquely assigned a status dependent on trajectory and time step. We then introduce an algorithm to perform rule status assessment. We categorize this novel time step-dependent analysis method as"pointwise-in-time" explanation and demonstrate its potential applications on illustrative examples.

II. Background

We consider discrete-time, discrete-state systems, a common setting for modeling autonomous agents. Such systems include Markov decision processes (MDPs) and Kripke structures; we define the latter below. We then describe Linear Temporal Logic (LTL), a temporal-logical language which can express constraints ("rules") on such systems.

Kripke Structures. We define a Kripke structure over a set of atomic propositions, named *labels*, as in Definition 1.

Definition 1 (Kripke structure): Given a set P of labels, the Kripke structure K over P is a tuple

$$K = (S, I, \mathcal{T}, \mathcal{L}). \tag{1}$$

where S is a finite set of states, $I \subseteq S$ is the set of initial states, and $\mathcal{T} \subseteq S \times S$ is a transition relation with $(\sigma, \sigma') \in \mathcal{T}$, $\sigma' \in S$ for all $\sigma \in S$. Finally, $\mathcal{L} : S \rightarrow 2^P$ is a labeling function assigning a set $L \subseteq P$ to each state in S.

For clarity, we establish separate notation for a state variable s_t and a constant state $\sigma \in S$, i.e., the expression $s_2 = \sigma_1$ signifies *the state at time* $t = 2$ *is* σ_1. Each unique constant label is denoted $\alpha \in P$.

We consider a finite discrete time interval $\{t \mid T_0 \leq t \leq T_f\}$, where $T_0, t, T_f \in \mathbb{N}$. A *run* $\kappa = s_{T_0}, ..., s_{T_f}$ of the system must satisfy $s_{T_0} \in I$ and $(s_t, s_{t+1}) \in R$ for all $t \in \{T_0, ..., T_f\}$. Importantly, each κ induces a *trace* ρ:

Definition 2 (Trace): For any state $s_t \in S$, $T_0 \leq t \leq T_f$, let $\mathcal{L}(s_t) = L_t$, where L_t is a set of labels. A trace is the sequence $\rho = (L_{T_0}, ..., L_{T_f})$ produced by system trajectory $(s_{T_0}, ..., s_{T_f})$ from time step T_0 to T_f.

A trace *suffix* $\rho^{t_0 \cdots} = L_{t_0}, ..., L_{T_f}$ is the part of the trace beginning at $t_0 \geq T_0$ and ending at T_f.

Linear Temporal Logic. Linear temporal logic (LTL) is a human-intuitive way to express time- and order-dependent specifications [20], [21]. An LTL formula φ may be evaluated on a trace ρ provided φ is defined on labels in P.

Definition 3 (Linear Temporal Logic Formula): For the set of LTL formulas over a finite set P of labels,

- if $\alpha \in \mathcal{T}$, then α is itself an LTL formula.
- if φ_1 and φ_2 are LTL formulas, then $\neg\varphi_1$, $\varphi_1 \vee \varphi_2$, $\mathbf{X}\varphi_1$, and $\varphi_1 \mathbf{U} \varphi_2$ are LTL formulas.

Definition 4 (Formula Evaluation on Traces): LTL formula φ is true on trace ρ, denoted $\rho \models \varphi$, in the following cases:

- $\rho \models \alpha$ where $\alpha \in P$ iff $\alpha \in L_0$
- $\rho \models \neg\varphi$ iff $\rho \not\models \varphi$
- $\rho \models \varphi_1 \vee \varphi_2$ iff $\rho \models \varphi_1$ or $\rho \models \varphi_2$

and for the temporal operators Next $\mathbf{X}$ and Until $\mathbf{U}$,

- $\rho \models \mathbf{X}\varphi$ iff $\rho^{1 \cdots} \models \varphi$
- $\rho \models \varphi_1 \mathbf{U} \varphi_2$ iff $\exists i \geq 0$ s.t. $\rho^{i \cdots} \models \varphi_2$ and $\rho^{k \cdots} \models \varphi_1$ for all $0 \leq k < i$

Table I intuitively describes LTL operators. Note that all of the operators listed can be constructed from $\mathbf{X}, \mathbf{U}, \vee$, and $\neg$ [22]. Lastly, we define formula *arguments*, so that we may decompose formulas into their constituent parts.

Definition 5 (Arguments of an LTL formula): Consider the LTL order of operations: (1) grouping symbols; (2) $\neg, \mathbf{X}$, and other unary operators; (3) $\mathbf{U}$ and other temporal binary operators; and (4) $\vee, \wedge, \rightarrow$. For LTL formula φ, all φ_j bound by the weakest operator are the *arguments* of φ. For example, $\varphi = \mathbf{G}\alpha_1 \vee \alpha_2$ has arguments $\varphi_1 = \mathbf{G}\alpha_1$, $\varphi_2 = \alpha_2$. For labels, the argument is the formula itself (α).

III. Problem Statement

We consider a scenario in which an agent with unknown policy (a "black box" agent) is observed by a human. We suppose the human wishes to characterize the agent's behavior in terms of a set of rules; these rules may represent tasks or safety constraints, for example. *A priori*, an agent trajectory may or may not follow any of the rules.

We suppose the human would like to know (1) which rules a trajectory follows, as well as (2) when and how the trajectory progresses through each rule. For example, (2) is relevant when a rule has subtasks ("do this, then that") or multiple fulfillment conditions ("do this or that"). The second question requires diagnostics which check for specific conditions at each time step in the trace; here, rule inference, which examines entire traces at once, is insufficient.

Operator	Meaning
$\mathbf{G}\varphi_1$	**Global**: φ_1 is always true.
$\mathbf{F}\varphi_1$	**Eventual**: φ_1 eventually occurs.
$\mathbf{X}\varphi_1$	**Next**: φ_1 must occur at the next time step.
$\varphi_1 \mathbf{U} \varphi_2$	**Until**: φ_1 remains true until φ_2 occurs (and φ_2 must occur)
$\varphi_1 \mathbf{W} \varphi_2$	**Weak until**: φ_1 must remain true (1) always, or (2) until φ_2 becomes true.
$\varphi_1 \mathbf{R} \varphi_2$	**Release**: φ_1 must remain true (1) always, or (2) until (and including) the time step when φ_2 becomes true.
$\varphi_1 \mathbf{M} \varphi_2$	**Strong release**: True only if Condition (2) for Release is met.
$\varphi_1 \wedge \varphi_2$	**Conjunction**: both φ_1 and φ_2 must be true
$\varphi_1 \vee \varphi_2$	**Disjunction**: φ_1 or φ_2 or both must be true
$\varphi_1 \rightarrow \varphi_2$	**Implication**: if φ_1 is true, φ_2 must also be true

TABLE I
SELECTED LTL OPERATORS.

Formal Problem Setting. We consider agents whose dynamics are expressible as a Kripke structure, as well as a set of LTL rules over P which describe behaviors of interest. Given an observed trace ρ, our goal is to assess the progress of rules at any time step; we propose the assignment of a *status* to each rule and its arguments at query points $t^* \in \{T_0, ..., T_f\}$ for all trace suffixes $\rho^{t_0^* \cdots}$ where $t_0^* \leq t^*$.

Moving towards a notion of rule status, we begin with two observations. Firstly, consider a trace ρ and rule φ. Suppose some t_0 exists such that φ is true on $\rho^{t_0 \cdots}$ no matter which labels are contained in any $L_{t \geq t_0}$. In this case, the trace beyond t_0 appears to be arbitrary with respect to φ:

Definition 6 (Arbitrariness of suffix): For LTL formula φ, its argument(s) φ_j, trace $\rho^{T_0 \cdots}$, and associated suffix $\rho^{t_0 \cdots}$ where $T_0 \leq t_0 \leq T_f$, we say that $\rho^{t \cdots}$ is arbitrary with respect to φ if $\rho^{t_0 \cdots} \models \varphi$ regardless of the truth of $\rho^{t \cdots} \models \varphi_j$ for all t where $t_0 \leq t \leq T_f$.

Secondly, we consider implication rules $\varphi = \varphi_1 \rightarrow \varphi_2$. When φ is true on ρ, there are two very distinct possibilities: either φ_1 has occurred in ρ, in which case φ_2 must be made true; or φ_1 did not occur, in which case φ_2 plays no role. We call such a φ_1 a *precondition*:

Definition 7 (Precondition): The precondition of an LTL formula φ is defined as

1) φ_1 if φ has the form $\varphi_1 \rightarrow \varphi_2$
2) $\top$ otherwise

Motivated by these observations, we characterize the roles of rules at time steps by defining a novel LTL rule status.

Definition 8 (Status of LTL rule): For LTL formula φ, trace $\rho^{T_0 \cdots}$ and times $t_0, t \in \{T_0, ..., T_f\}$, $t_0 \leq t$:

- φ is active (a) at t iff (1) $\rho^{t_0 \cdots} \models \varphi$, (2) $\rho^{t_0 \cdots} \models \psi$ for precondition ψ of φ, and (3) $\rho^{t \cdots}$ is not arbitrary.
- φ is satisfied (s) at t iff (1) φ active at t and (2) $t = T_f$ or φ not active at $t + 1$.
- φ is inactive (i) at t iff (1) $\rho^{t_0 \cdots} \models \varphi$ and (2) φ is

neither active nor satisfied at t.

- φ is violated (v) at t iff $\rho^{t_0\cdots} \not\models \varphi$.

We briefly illustrate the status notions in simple examples, expressing rules in words for clarity.

Example 1 (Status): Consider trace $\rho = (L_{T_0}, ..., L_{T_f})$ produced by a sample collection robot. Suppose we select a suffix $\rho^{t_0\cdots}$ to analyze, where $T_0 \leq t_0 \leq T_f$. Then we have the following status examples:

Active: Let $\varphi =$ "Headlight always on at night" and suppose "night", "light on" $\in L_t$ for all $t \geq t_0$. Then φ is active at all $t \geq t_0$.

Satisfied: Let $\varphi =$ "Eventually deposit sample". Suppose "sample deposited" $\in L_t$, "sample deposited" $\notin L_{t'}$ for all $t' < t$. Then φ is satisfied at t.

Inactive (done): Let $\varphi =$ "Eventually deposit sample" and suppose "sample deposited" $\in L_{t'}$ for some $t' < t$. Then φ is inactive at t, since φ was already satisfied.

Inactive (not triggered): Let $\varphi =$ "If raining, stop" and suppose "rain" $\notin L_t$. Then φ is inactive at t, since precondition "rain" is not occurring and stopping is not necessary.

Violated: Let $\varphi =$ "Battery always above 10%" and suppose "battery empty" $\in L_t$. Then φ is violated at all t.

Finally, we can express the set of all times in $\{t_0, ..., T_f\}$ for which φ has status q on $\rho^{t_0\cdots}$ as a *timeset*

$$\tau^q = \{t \in \{t_0, ..., T_f\} \mid t \text{ has status } q\}. \tag{2}$$

If we recover such time sets for a rule φ, it is possible to check its status at any moment in an agent trajectory; moreover, time sets for each argument of φ may help to track progress "through" the rule, such as completion of subtasks.

Formal Problem Statement. For an LTL formula φ and a plan trace $\rho^{T_0\cdots T_f}$, an explainer must be able to provide:

- **Status** of φ at t on $\rho^{t_0\cdots}$ for all $\{t_0, t \in \{T_0, ..., T_f\} \mid t_0 \leq t\}$
- **Status timesets** $\tau^a, \tau^s, \tau^i, \tau^v$ of φ for each $t_0 \in \{T_0, ..., T_f\}$
- **Both of the above** for any argument φ_j of φ.

In this paper, we propose an algorithm that returns this information on demand and begin to explore its applications.

IV. Rule Status Assessment

We wish to evaluate rule status for all arguments of a formula. To accomplish this in a structured manner, formulas are first decomposed into a tree structure as follows.

Tree Structure. LTL formulas φ are commonly structured as trees with φ as the base node [13] [15]. Here, the children of a parent node are the arguments of the parent formula, until each branch terminates in a leaf containing only a label. We additionally store the type of the weakest operator in φ_j for each node, denoted θ_j; in the case where $\varphi_j = \alpha$, we assign type $\theta_j = AP$ ("atomic proposition") to denote that the node contains a label. An example is given in Figure 1.

Modules and Algorithm. Each node of a tree contains a formula consisting of one or more arguments bound by an operator. The operator type determines how the argument truth values map to a status; for instance, consider a trace ρ where a formula φ_j is true at a single time step t'. Here, a node with $\varphi = \mathbf{G}\varphi_j$ (type $\mathbf{G}$) is violated at all t; a node $\varphi = \mathbf{F}\varphi_j$ (type $\mathbf{F}$) is active until satisfaction at t'. Thus, we establish separate modules for every type, each with inputs φ, t_0, $\rho^{t_0\cdots}$ and outputs $\tau^a, \tau^s, \tau^i, \tau^v$ to satisfy Definition 8.

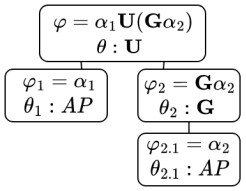

Fig. 1. Example formula tree for $\varphi = \alpha_1 \mathbf{U}(\mathbf{G}\alpha_2)$.

Our proposed algorithm works node-by-node, beginning with all leaf nodes and working up each branch until τ has been calculated for the entire tree (child nodes are always instantiated before their parents). The algorithm then accepts queries and returns status at the desired time steps:

Rule Status Assessment Algorithm

1) Construct tree for LTL rule φ.
2) Find $\tau^a, \tau^s, \tau^i, \tau^v$ of each node for all $\rho^{t_0\cdots}$, $t_0 \in \{T_0, ..., T_f\}$:
   a) For each distinct label α appearing in any leaf, pass $\rho^{t_0\cdots}$, $\varphi = \alpha$ into the AP module for all $t_0 \in \{T_0, ..., T_f\}$. Store outputs τ for all leaves containing this α.
   b) For leaf $\varphi_{...x.y}$, check parent $\varphi_{...x}$. If τ is already stored for all children of $\varphi_{...x}$, proceed to (c); otherwise, move to next leaf and repeat (b).
   c) Identify θ of $\varphi_{...x}$. Pass $\varphi_{...x}$, $\rho^{t_0\cdots}$ into θ module for all $t_0 \in \{T_0, ..., T_f\}$; store output τ for $\varphi_{...x}$.
   d) Check parent of $\varphi_{...x}$. If it exists and all its children are instantiated, repeat step c) for this parent. Otherwise, move to next leaf and begin again from step (b).
3) Answer status queries:
   a) Accept t^*, t_0^* and a node (e.g., $\varphi_{x.y}$) as input.
   b) Return status q if $t^* \in \tau^q$ for $\rho^{t_0^*\cdots}$.

The temporal and logical operator modules are defined in Tables III and II, respectively. For proof that the modules satisfy Definition 8, see Brindise and Langbort [23].

Algorithm Complexity. An upper bound on complexity may be established from the modules. For each node $\varphi_{...x}$ in a tree, the corresponding module is called once; this module produces τ for every t_0 in $\{T_0, ..., T_f\}$, resulting in $\leq |\rho|$ iterations. At each t_0, the module performs up to $T_f - t_0 \leq |\rho|$ evaluations per argument $\varphi_{...x.y}$. Thus, any node with a binary operator (two arguments) performs $n \leq 2|\rho|^2$ computations.

Now, for a tree with depth L, with base node at level $l = 0$ and maximum 2 children per node, the number of nodes at level L, n_L, satisfies $n_L \leq 2^L$. Maximum complexity is thus order $2^l(2|\rho|^2)$, or

$$\mathcal{O}(2^{l+1}|\rho|^2). \tag{3}$$

Logical operators with more than 2 arguments (e.g. $\varphi_1 \vee$

Not (neg)	LTL: $\varphi = \neg\varphi_1$
Initialize: If $\tau_1^v = \emptyset$: Else:	Load all τ_1^q of child node φ_1 at t_0. Set $\tau^v = \{t_0, ..., T_f\}$. Set $\tau^a, \tau^s = \{t_0\}, \tau^i = \{t_0, ..., T_f\} \setminus \tau^a$. Set $\tau^v = \emptyset$.
Or (or)	LTL: $\varphi = \varphi_1 \vee \varphi_2$
Initialize: *If $\tau_1^v, \tau_2^v \neq \emptyset$: Else:	Load all τ_1^q, τ_2^q of child nodes φ_1, φ_2 at t_0. Set $\tau^v = \{t_0, ..., T_f\}$; $\tau^a, \tau^s, \tau^i = \emptyset$. Set $\tau^a, \tau^s = \{t_0\}, \tau^i = \{t_0, ..., T_f\} \setminus \tau^a$. Set $\tau^v = \emptyset$.
And (and)	LTL: $\varphi = \varphi_1 \wedge \varphi_2$
	Same as $\vee$ with * replaced by: If τ_1^v or $\tau_2^v \neq \emptyset$: (...)
Implication $(\rightarrow)$	LTL: $\varphi = \varphi_1 \rightarrow \varphi_2$
Initialize: If $\tau_1^v \neq \emptyset$: Else: If $\tau_2^v \neq \emptyset$: Else:	Load all τ_1^q of child node φ_1 at t_0. Set $\tau^a, \tau^s, \tau^v = \emptyset$; $\tau^i = \{t_0, ..., T\} \setminus \tau^a$. **Exit the current module.** Load all τ_2^q of child node φ_2 at t_0. Set $\tau^v = \{t_0, ..., T_f\}$; $\tau^a, \tau^s, \tau^i = \emptyset$. Set $\tau^a, \tau^s = \{t_0\}; \tau^i = \{t_0, ..., T_f\} \setminus \tau^a$. Set $\tau^v = \emptyset$.

TABLE II

LOGICAL OPERATOR MODULE DEFINITIONS.

Atomic (AP)	LTL: $\varphi = \alpha$
If $\alpha \in L_{t_0}$: Else:	Set $\tau^a, \tau^s = \{t_0\}, \tau^i = \{t_0, ..., T_f\} \setminus \tau^a$. Set $\tau^v = \emptyset$. Set $\tau^v = \{t_0, ..., T_f\}$; $\tau^a, \tau^s, \tau^i = \emptyset$.
Next (X)	LTL: $\varphi = \mathbf{X}\varphi_1$
Initialize: If $\tau_1^v = \emptyset$: Else:	Load all τ_1^q of child node φ_1 at $t_0^1 = t_1$. Set $\tau^a = \{t_0, t_1\}$; $\tau^s = \{t_1\}$, $\tau^i = \{t_0, ..., T_f\} \setminus \tau^a$. Set $\tau^v = \emptyset$. Set $\tau^v = \{t_0, ..., T_f\}$; $\tau^a, \tau^s, \tau^i = \emptyset$.
Eventual (F)	LTL: $\varphi = \mathbf{F}\varphi_1$
For $t_0^1 = t_0 ... T_f$: If $\tau_1^v = \emptyset$: Else:	Load all τ_1^q of child node φ_1 at t_0^1. Continue until $t_0^1 = T_f$ or $\tau_1^v = \emptyset$. Set $\tau^a = \{t_0, ..., t_0^1\}$; $\tau^s = \{t_0^1\}$; $\tau^i = \{t_0, ..., T_f\} \setminus \tau^a$. Set $\tau^v = \emptyset$. **Exit the current module.** Set $\tau^v = \{t_0, ..., T_f\}$; $\tau^a, \tau^s, \tau^i = \emptyset$.
Global (G)	LTL: $\varphi = \mathbf{G}\varphi_1$
For $t_0^1 = t_0 ... T_f$: If $\tau_1^v \neq \emptyset$: Else:	Load all τ_1^q of child node φ_1 at t_0^1. Continue until $t_0^1 = T_f$ or $\tau_1^v \neq \emptyset$. Set $\tau^v = \{t_0, ..., T_f\}$; $\tau^a, \tau^s, \tau^i = \emptyset$. **Exit the current module.** Set $\tau^a = \{t_0, ..., T_f\}; \tau^s = \{T_f\}; \tau^i, \tau^v = \emptyset$.
Until (U)	LTL: $\varphi = \varphi_1 \mathbf{U} \varphi_2$
For $t_0' = t_0 ... T_f$: †If $\tau_2^v = \emptyset$: Else if $\tau_1^v \neq \emptyset$: *Else:	Load all τ_1^q, τ_2^q of child nodes φ_1 and φ_2 at t_0'. Continue until $\tau_1^v \neq \emptyset$ or $\tau_2^v = \emptyset$ or $t_0^2 = T_f$. Set $\tau^a = \{t_0, ..., t_0'\}; \tau^s = \{t_0'\}; \tau^i = \{t_0, ..., T_f\} \setminus \tau^a$. Set $\tau^v = \emptyset$. Set $\tau^v = \{t_0, ..., T_f\}$; $\tau^a, \tau^s, \tau^i = \emptyset$. Set $\tau^v = \{t_0, ..., T_f\}$; $\tau^a, \tau^s, \tau^i = \emptyset$.
W. until (W)	LTL: $\varphi = \varphi_1 \mathbf{W} \varphi_2$
	Same as $\mathbf{U}$ with * replaced by: Else: Set $\tau^a = \{t_0, ..., T_f\}; \tau^s = \{T_f\}, \tau^i = \emptyset$. Set $\tau^v = \emptyset$.
S. release (M)	LTL: $\varphi = \varphi_1 \mathbf{M} \varphi_2$
	Same as $\mathbf{U}$ with † replaced by: If $\tau_2^v = \emptyset$ and $\tau_1^v = \emptyset$: (...)
Release (R)	LTL: $\varphi = \varphi_1 \mathbf{R} \varphi_2$
	Same as $\mathbf{U}$ plus changes in $\mathbf{W}$ & $\mathbf{M}$.

TABLE III

TEMPORAL OPERATOR MODULE DEFINITIONS.

$\varphi_2 \vee \varphi_3$) can be rewritten as a tree of their groupings (e.g., $\varphi_1 \vee (\varphi_2 \vee \varphi_3)$), with L adjusted appropriately.

V. APPLICATION

In this section, we apply status assessment on hypothetical runs of two agents using the status assessment algorithm located at `https://github.com/n-brindise/live_expl`. We also suggest a basic heuristic to identify times t^*, t_0^* of potential interest in a trace.

Query Heuristics. Status assessment allows for a large number of queries ($\mathcal{O}(2^{l+1}|\rho|^2)$); thus, identification of informative queries from this large set is desirable. A first-order approach simply monitors the τ values as the algorithm iterates over $t_0 \in \{T_0, ..., T_f\}$ and stores any t_0 where ρ changes status to *active* or *violated* from the preceding t_0. We denote the set of all t_0^* of interest by τ^*.

Muddy Yard Example. The real world-inspired "muddy yard" scenario (Fig. 2) features an "outdoor" area with two "muddy" sections and an outdoor "mat," as well as an "indoor" area with a "sink." The two areas are separated by a "wall" with a "door." We formulate the domain as a Kripke structure with set S of states and set P of labels:

$$S = \{\sigma_1 = \text{grass} \quad \sigma_2 = \text{mud} \quad \sigma_3 = \text{mat} \quad \sigma_4 = \text{floor} \quad \sigma_5 = \text{sink} \quad \sigma_6 = \text{wall} \quad \sigma_7 = \text{door}\}$$
$$P = \{\alpha_1 = \text{outside} \quad \alpha_2 = \text{inside} \quad \alpha_3 = \text{muddy} \quad \alpha_4 = \text{wiped} \quad \alpha_5 = \text{washed} \quad \alpha_6 = \text{impassable}\}$$

where $I = S$. The transition relation $\mathcal{T}$ exclusively allows the agent to transition between adjacent states (e.g., $(\sigma_1, \sigma_2) \in \mathcal{T}$, but $(\sigma_1, \sigma_5) \notin \mathcal{T}$). Finally, the labeling function $\mathcal{L}$ is

$$\mathcal{L}(\sigma_1) = \{\alpha_1\} \quad \mathcal{L}(\sigma_2) = \{\alpha_1, \alpha_3\} \quad \mathcal{L}(\sigma_3) = \{\alpha_1, \alpha_4\}$$
$$\mathcal{L}(\sigma_4) = \{\alpha_2\} \quad \mathcal{L}(\sigma_5) = \{\alpha_2, \alpha_5\} \quad \mathcal{L}(\sigma_6) = \{\alpha_6\}$$
$$\mathcal{L}(\sigma_7) = \{\alpha_1, \alpha_2\}.$$

We wish to analyze the agent run in Figure 3 subject to the following LTL rules:

1) $\varphi = \mathbf{F}\alpha_1$ "Eventually be outside"
2) $\varphi = \mathbf{F}\alpha_2 \rightarrow \mathbf{F}\alpha_5$ "If eventually inside, eventually wash up"
3) $\varphi = \mathbf{G}(\alpha_3 \rightarrow (\neg\alpha_2 \mathbf{W} \alpha_4))$ "Always, if muddy, do not go inside until wiping off"
4) $\varphi = \mathbf{G}(\neg\alpha_6)$ "Never enter an impassable place"

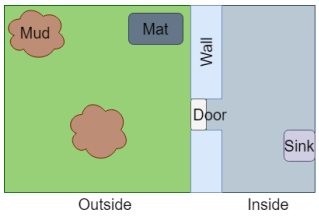

Fig. 2. "Muddy yard" example domain, representing a yard ("outside") with two muddy patches and a mat, as well as a home interior ("inside") with a sink. Outside and Inside are separated by an untraversable wall.

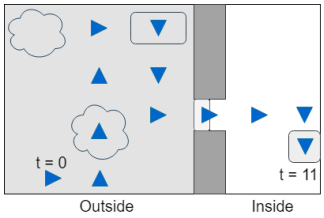

Fig. 3. Example path over the muddy yard domain. Each triangle represents the agent position and direction of travel at a single time step.

From Figure 3, the system run is

$$\kappa = \sigma_1, \sigma_1, \sigma_2, \sigma_1, \sigma_1, \sigma_3, \sigma_1, \sigma_1, \sigma_7, \sigma_4, \sigma_4, \sigma_5,$$

which produces the trace

$$\rho = \{\alpha_1\}_0, \{\alpha_1\}_1, \{\alpha_1, \alpha_3\}_2, \{\alpha_1\}_3, \{\alpha_1\}_4, \{\alpha_1, \alpha_4\}_5, \{\alpha_1\}_6, \{\alpha_1\}_7, \{\alpha_1, \alpha_2\}_8, \{\alpha_2\}_9, \{\alpha_2\}_{10}, \{\alpha_2, \alpha_5\}_{11}.$$

We now use status assessment to answer a series of questions. (When listing timeset outputs, we omit any $\tau = \emptyset$ for clarity.)

Question: "Which rules are active at time t^*?"

For $t_0 = T_0$, Rules 1-4 have (nonempty) timesets

1) $\tau^a = \{0\}, \tau^s = \{0\}, \tau^i = \{1, ..., 11\}$
2) $\tau^a = \{0\}, \tau^s = \{0\}, \tau^i = \{1, ..., 11\}$
3) $\tau^a = \{0, ..., 11\}, \tau^s = \{11\}$
4) $\tau^a = \{0, ..., 11\}, \tau^s = \{11\}$

At $t^* = 3$, output is:

- *"Rules 1 and 2 are inactive"*
- *"Rules 3 and 4 are active"*

Intuitively, Rule 1 requires that the agent goes outside; this occurred already at $t = 0$, hence the inactivity by $t = 3$. Rule 2 requires that, if the agent will eventually go inside, it must also eventually wash up; both of these conditions are already true on $\rho^{0\cdots}$, and thus inactive for $t^* > 0$. This output is less intuitive, as it is not clear *when* the agent will go inside or wash up. Finally, Rules 3 and 4 are global and must be active at every time step. For 2, 3, and 4, elaboration is likely necessary, prompting another question.

Question: "Which arguments of rules are active at t^*?"

We first query Rule 3 at node φ_1. From heuristics, $\tau^* = \{2\}$, so we query φ_1 at $t^*, t_0^* = 2$, which yields

- $\tau_1^a = \{2\}, \tau_1^s = \{2\}, \tau_1^i = \{0, 1, 3, ..., 11\}$
- *"Rule 3.1 is active and satisfied (at $t^* = 2$)"*

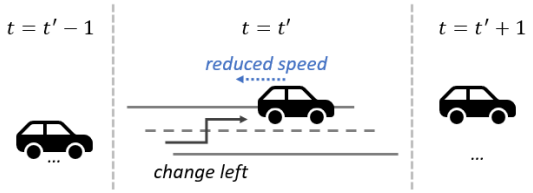

Fig. 4. Vehicle observed changing lanes and reducing speed at $t = t'$.

This suggests that $\alpha_3 \rightarrow (\neg\alpha_2 \mathbf{W} \alpha_4)$ is active on $\rho^{2\cdots}$. Indeed, α_3 ("muddy") occurs at $t = 2$, triggering the implication. Querying $\varphi_{1.2}$ at $t_0^* = 2, t^* = 5$,

- $\tau_{1.2}^a = \{2, ..., 5\}, \tau_{1.2}^s = \{5\}, \tau_{1.2}^i = \{6, ..., 11\}$
- *"Rule 3.1.2 is active and satisfied (at $t^* = 5$)"*

Intuitively, though Rule 3 always holds, it only takes direct effect starting at 2 (when the agent becomes muddy) and releases this effect at 5 (when the agent wipes itself off).

Moving on to Rule 2, we perform queries for $\varphi_1 = \mathbf{F}\alpha_2$ and $\varphi_2 = \mathbf{F}\alpha_5$, and we find that both are *active* at $t_0^* = 0, t^* = 3$. This makes sense, as the agent has neither gone inside nor washed up by step 3. At $t_0^*, t^* = 9$, φ_1 is now *inactive*, since the agent has been inside for several steps, but φ_2 remains *active*, since it has yet to wash up. Finally, the agent washes up at 11, and $t^* = 11$ gives φ_2 *satisfied*.

Autonomous Vehicle Example. Autonomous vehicles must comply to safety guidelines, operational limits, and traffic laws, all of which are typically expressible via temporal logics [24]. This example considers a hypothetical setting in which an autonomous vehicle completes three distinct trips, producing 3 traces over a large vocabulary ($|\mathcal{P}| = 54$) and 90 time steps. 21 LTL formulas describe various rules for safety, legal, and trip requirements, such as "reduce speed in construction zones" and "activate wipers if rain hazard present." The traces and list of rules are available at [23].

Notably, the raw traces are visually difficult to parse. Single time steps contain lengthy sets of labels, for example {leftmost, doors-closed, driver-awake, clear-ahead, want-turn-left, clear-right, lane-to-right, gas-low, intersection-ahead, green-ahead, reduced-speed, safe-stop} at $\rho_1, t = 36$.

To motivate our example, suppose we observe the vehicle **changes lanes to the left** and **reduces speed** at time t' in all three trips, as shown in Figure 4. We wish to investigate what has occurred in each trace that may explain this behavior.

As the traces are dense and the list of rules is long, we use status assessment. We place queries at $t_0^*, t^* = t'$ for the first argument of all 20 global rules $\mathbf{G}(...)$.

For ρ_1, we find (1) **satisfied** gas-low $\rightarrow \neg(\text{goal})\mathbf{U}$gas-station, (2) **satisfied** want-left-turn $\wedge$ near-intersection $\wedge$ clear-left $\wedge$ lane-to-left $\rightarrow$ change-left and (3) **satisfied** want-turn $\wedge$ near-intersection $\rightarrow$ reduced-speed. In words, the vehicle is heading to a gas station, approaching an intersection, and planning to turn left.

For ρ_2, we find (1) **satisfied** merge-left $\wedge$ clear-left $\wedge$ lane-to-left $\rightarrow$ change-left and (2) **satisfied** construction-zone $\rightarrow$ reduced-speed$\mathbf{U}$leave-zone. Here, the vehicle has entered a construction zone and must merge.

Finally, for ρ_3, we have (1) **satisfied** short-follow-distance $\rightarrow$ reduced-speed. We place an additional query at $t_0^*, t^* = t' - 1$, which returns (2) **satisfied** short-follow-distance $\wedge$ (clear-left $\wedge$ lane-to-left) $\rightarrow$ **X**change-left. Thus, the vehicle encountered another vehicle which had dangerously reduced the following distance.

In all, status assessment processed a large set of rules and dense traces to produce brief diagnostic statements which highlighted the specific circumstances of each left-lane change, a promising aid to understand the situation.

VI. Discussion and Conclusion

In this work, we introduce rule status assessment (RSA), a method for explanatory diagnostics for generic autonomous agent trajectories. RSA is applicable to any system expressible via Kripke structure, providing a novel framework to classify LTL rules of interest as active, satisfied, inactive, or violated at individual trajectory time steps. This makes it possible to track an agent's progress through subtasks and more; it also provides insight into the applicability of individual rules and reduces the total number of rules a human must consider at once.

Multiple directions exist for future work. Firstly, the broader application of RSA is a critical next step to demonstrate effectiveness for practical purposes, particularly in common settings such as Machine Learning (ML). Further application also calls for additional query heuristics; it is a challenge to identify useful queries to achieve insightful RSA diagnostics. Finally, the existing framework is entirely post-hoc; for certain families of systems, such as Reinforcement Learning agents, use of agent internals (i.e., Q function) could incorporate agent intentionality.

References

[1] Y. Xing, C. Lv, D. Cao, and P. Hang, "Toward human-vehicle collaboration: Review and perspectives on human-centered collaborative automated driving," *Transportation research part C: emerging technologies*, vol. 128, p. 103199, 2021.

[2] I. Seeber, L. Waizenegger, S. Seidel, S. Morana, I. Benbasat, and P. B. Lowry, "Collaborating with technology-based autonomous agents: Issues and research opportunities," *Internet Research*, 2020.

[3] S. Franklin and A. Graesser, "Is it an agent, or just a program?: A taxonomy for autonomous agents," in *Intelligent Agents III Agent Theories, Architectures, and Languages: ECAI'96 Workshop (ATAL) Budapest, Hungary, August 12–13, 1996 Proceedings 3*. Springer, 1997, pp. 21–35.

[4] S. Sreedharan, A. Kulkarni, and S. Kambhampati, "Explainable human–ai interaction: A planning perspective," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 16, no. 1, pp. 1–184, 2022.

[5] A. Lindsay, "Using generic subproblems for understanding and answering queries in xaip," *Proceedings of KEPS*, 2020.

[6] T. Chakraborti, S. Sreedharan, and S. Kambhampati, "The emerging landscape of explainable automated planning and decision making," in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, C. Bessiere, Ed. International Joint Conferences on Artificial Intelligence Organization, 7 2020, pp. 4803–4811, survey track. [Online]. Available: https://doi.org/10.24963/ijcai.2020/669

[7] K. Boggess, S. Kraus, and L. Feng, "Toward policy explanations for multi-agent reinforcement learning," *arXiv preprint arXiv:2204.12568*, 2022.

[8] M. Brandão and Y. Setiawan, "'why not this mapf plan instead?'contrastive map-based explanations for optimal mapf," in *ICAPS 2022 Workshop on Explainable AI Planning*, 2022.

[9] A. Kumar, S. L. Vasileiou, M. Bancilhon, A. Ottley, and W. Yeoh, "Vizxp: A visualization framework for conveying explanations to users in model reconciliation problems," in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 32, 2022, pp. 701–709.

[10] D. Neider and I. Gavran, "Learning linear temporal properties," in *2018 Formal Methods in Computer Aided Design (FMCAD)*. IEEE, 2018, pp. 1–10.

[11] 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.

[12] G. Bombara and C. Belta, "Offline and online learning of signal temporal logic formulae using decision trees," *ACM Trans. Cyber-Phys. Syst.*, vol. 5, no. 3, mar 2021. [Online]. Available: https://doi.org/10.1145/3433994

[13] C. Lemieux, D. Park, and I. Beschastnikh, "General ltl specification mining (t)," in *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2015, pp. 81–92.

[14] G. Chou, N. Ozay, and D. Berenson, "Explaining multi-stage tasks by learning temporal logic formulas from suboptimal demonstrations," *arXiv preprint arXiv:2006.02411*, 2020.

[15] J. Kim, C. Muise, A. J. Shah, S. Agarwal, and J. A. Shah, "Bayesian inference of linear temporal logic specifications for contrastive explanations," *IJCAI International Joint Conference on Artificial Intelligence*, 2019.

[16] R. Roy, J.-R. Gaglione, N. Baharisangari, D. Neider, Z. Xu, and U. Topcu, "Learning interpretable temporal properties from positive examples only," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 5, 2023, pp. 6507–6515.

[17] J.-R. Gaglione, D. Neider, R. Roy, U. Topcu, and Z. Xu, "Learning linear temporal properties from noisy data: A maxsat-based approach," in *Automated Technology for Verification and Analysis: 19th International Symposium, ATVA 2021, Gold Coast, QLD, Australia, October 18–22, 2021, Proceedings 19*. Springer, 2021, pp. 74–90.

[18] E. Bartocci, R. Bloem, D. Nickovic, and F. Röck, "A counting semantics for monitoring ltl specifications over finite traces," in *International Conference on Computer Aided Verification*. Springer, 2018, pp. 547–564.

[19] E. Aasi, C. I. Vasile, M. Bahreinian, and C. Belta, "Inferring temporal logic properties from data using boosted decision trees," *arXiv preprint arXiv:2105.11508*, 2021.

[20] X. Ding, S. L. Smith, C. Belta, and D. Rus, "Optimal control of markov decision processes with linear temporal logic constraints," *IEEE Transactions on Automatic Control*, vol. 59, no. 5, pp. 1244–1257, 2014.

[21] A. Camacho and S. A. McIlraith, "Learning interpretable models expressed in linear temporal logic," in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 29, 2019, pp. 621–630.

[22] A. Pnueli, "The temporal logic of programs," in *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*. ieee, 1977, pp. 46–57.

[23] N. Brindise and C. Langbort, "Pointwise-in-time explanation for linear temporal logic rules," *arXiv preprint arXiv:2306.13956*, 2023.

[24] S. Maierhofer, P. Moosbrugger, and M. Althoff, "Formalization of intersection traffic rules in temporal logic," in *2022 IEEE Intelligent Vehicles Symposium (IV)*, 2022, pp. 1135–1144.