Generalized Temporal Inference via Planning

Diego Aineto , Sergio Jiménez , Eva Onaindia
Valencian Research Institute for Artificial Intelligence
Universitat Politècnica de València
{dieaigar, serjice, onaindia}@upv.es

Abstract

This paper introduces the Temporal Inference Problem (TIP), a general formulation for a family of inference problems that reason about the past, present or future state of some observed agent. A TIP builds on the models of an actor and of an observer. Observations of the actor are gathered at arbitrary times and a TIP encodes hypothesis on unobserved segments of the actor's trajectory. Regarding the last observation as the present time, a TIP enables to hypothesize about the past trajectory, future trajectory or current state of the actor. We use LTL as a language for expressing hypotheses and reduce a TIP to a planning problem which is solved with an off-the-shelf classical planner. The output of the TIP is the most likely hypothesis, the minimal cost trajectory under the assumption that the actor is rational. Our proposal is evaluated on a wide range of TIP instances defined over different planning domains.

1 Introduction

Inference is the process of reaching a conclusion on the basis of evidence and reasoning. Many planning-related tasks are assimilated to inference tasks. A diagnosis problem defines a set of observations and a hypothesis space of faulty components with the purpose of finding an explanation of the observed behaviour through exploration of the hypothesis space (Grastien et al. 2011). Goal recognition is the problem of inferring the objective of an actor among a set of hypothesized goals by observing its behaviour (Ramírez and Geffner 2009; Sukthankar et al. 2014). Some plan recognition approaches additionally stress the role of unreliable and noisy observations in the inference task (Sohrabi, Riabov, and Udrea 2016; Kim et al. 2018).

Given some observed behaviour of an agent and a set of hypotheses, several different approaches have been proposed to tackle the problem of finding explanations (reasoning about past) or generating plans (reasoning about future). Finding preferred explanations for observed behavior has been formulated with LTL (Sohrabi, Baier, and McIlraith 2011). Situation calculus and event-based language (Grastien et al. 2011; Haslum and Grastien 2011) are commonly used to capture the evolution of the system. Recognition tasks have also been addressed from different perspectives: with a probabilistic interpretation of the recognition problem (Ramírez and Geffner 2010; Sohrabi, Riabov, and Udrea 2016; E-Martín, R.-Moreno, and Smith 2015), with planning-based formulations (Pereira, Oren, and Meneguzzi 2020) or adopting changes in the agent model to enhance the recognition task (Keren, Gal, and Karpas 2019; Keren, Gal, and Karpas 2020). Assumption-based planning has been proposed to address both past and future inference tasks under incomplete knowledge (Davis-Mendelow, Baier, and McIlraith 2013).

In general, inference tasks are differentiated between *explicability* and *predictability* . This distinction, which has been discussed in recent works (Chakraborti et al. 2019), conceives explicability as finding completions of a plan prefix via plans that satisfy the emitted observations, and predictability as minimizing the set of possible completions so as to unambiguously predict the goal of the agent. This view reveals that inference oriented towards explaining some past behaviour or to predicting some future behaviour are two sides of the same coin and as such, they can be formulated within a single unified framework.

Another relevant aspect is that inference tasks typically account for hypotheses on the unobserved behaviour at a single point of the actor's trajectory. Assumptions regarding the initial state are included in approaches to the diagnosis problem (Sohrabi, Baier, and McIlraith 2010), estimations about the actual state are considered in plan monitoring (Fritz and McIlraith 2007), and the set of hypotheses in plan recognition contains goal states. The ability to hypothesize that two variables are achieved in some particular order or conjecturing about whether a variable is achieved before or after some observation is a powerful inference ability.

In this paper, we present a general formulation of a temporal inference problem (TIP) and its relationship to planning from both a theoretical and practical perspective. Our contributions are:

- a decoupled planning-sensing model that allows representing observations captured with different types of sensors,
- a formalization of a temporal hypothesis on finite traces expressed in Linear Temporal Logic (LTL$_f$) that represents an intertwined combination of observations and conjectures,
- a TIP formulation in terms of a planning model and a set of hypotheses to infer the unobserved behaviour of the actor,

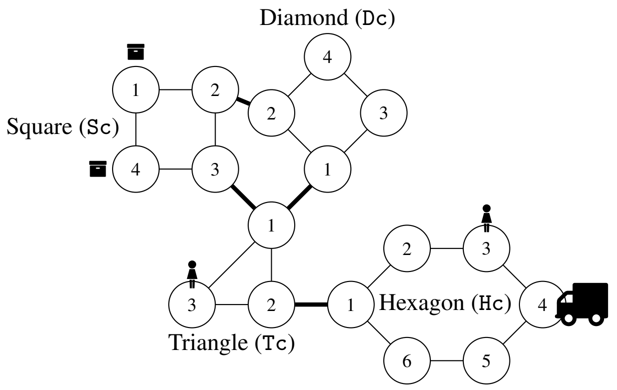

Figure 1: Initial state corresponding to a classical planning problem from the *driverlog* domain: Square city (Sc), Diamond city (Dc), Triangle city (Tc) and Hexagon city (Hc).

- a formal characterization of a TIP solution as planning via translation of a temporal formula to a deterministic finite automata (DFA).

The paper is structured as follows. The next section introduces a motivating example that we will use throughout the paper to support explanations and illustrate the application of theoretical concepts. Section 3 introduces the planning model and the sensor model. Section 4 formalizes the concept of hypothesis using LTL$_f$. Section 5 formulates the temporal inference problem (TIP) and identifies three types of TIP of special interest. In section 6 we show how to formally characterize a TIP solution as a planning problem by translating temporal formulas to a DFA. Section 7 presents the empirical evaluation of the three types of TIP. In section 8 we summarize related work and last section concludes.

2 Motivating Scenario

To exemplify the wide scope of temporal inference, Figure 1 shows an initial state of a classical planning problem from the *driverlog* domain introduced at the 3^{rd}*International Planning Competition* (IPC) (Fox and Long 2003). In this domain, packages are transported by trucks and trucks navigate between two connected locations provided that a driver is onto the truck, and drivers ride trucks. Drivers can also walk between two locations iff a walking path connects these locations (represented in Figure 1 by a thin line). Figure 1 features locations grouped in four cities, namely *Square city* (Sc), *Diamond city* (Dc), *Triangular city* (Tc) and *Hexagon city* (Hc). Two packages are found at locations 1 and 4 of the Square city, one driver at location 3 of the Triangular city, a second driver at location 3 of the Hexagon city, and a single truck at location 4 of this same city. There are no drivers or trucks in the Diamond city. Examples of TIPs are: tracking the driver who is currently driving the truck; uncovering the origin location of the truck; predicting the delivering order of the packages after observing only the cities that the truck traverses or the number of packages and drivers in each city.

3 Preliminaries

In this section we introduce the basic notions needed for the formalization of the temporal inference problem as well as a working example on the domain illustrated in Figure 1.

3.1 The Augmented Model

Our approach builds on a decoupled planning-sensing model. The model of the actor (planning agent) and the model of the observer (sensing agent) are defined independently of each other. We call *augmented model* the pair $\mathcal{M} = \langle \mathcal{M}_p, \mathcal{M}_s \rangle$ that contains a planning model $\mathcal{M}_p$, which implicitly represents the set of all possible plans of the actor, and a sensor model $\mathcal{M}_s$, which defines the type of observations that are captured by the observer.

The planning model A planning model is a tuple $\mathcal{M}_p = \langle X, A, \theta, cost \rangle$, where X is a finite set of state variables, A is a finite set of deterministic actions, $\theta : X \times A \rightarrow X$ is a successor function, and $cost : A \rightarrow \mathbb{N}$ is a cost function. Each state variable $X_i \in X$ takes on a single value from its corresponding domain D_{X_i}. A state $s = \langle X_1 = v_1, \ldots, X_{|X|} = v_{|X|} \rangle$ is a full assignment over X. The set of all possible states derivable from $\mathcal{M}_p$ is denoted by S.

A planning problem is a tuple $P = \langle \mathcal{M}_p, I, G \rangle$ where $\mathcal{M}_p$ is a planning model, $I \in S$ is the initial state of the problem and G is the goal condition over the state variables. G implicitly defines the subset $S_G \subseteq S$ of goal states.

A plan is an action sequence $\pi = [a_1, \ldots, a_n]$, and $|\pi| = n$ denotes the plan length. The execution of a plan π in an initial state I induces a *trajectory* $\tau = [s_0, \ldots, s_n]$, such that $s_0 = I$ and for each $1 \leq i \leq n$ it holds that $s_i = \theta(s_{i-1}, a_i)$. The cost of a trajectory $\tau = [s_0, \ldots, s_n]$ induced by a plan $\pi = [a_1, \ldots, a_n]$ is defined as $cost(\tau) = \sum_{i=1}^{n} cost(a_i)$. A solution plan π is a solution to a planning problem P iff the goal condition holds in the last state reached by the induced trajectory; i.e. $s_n \in S_G$. A plan is optimal iff it induces a minimal cost trajectory. Hereafter, and without loss of generality, we will assume that states are full assignments over $X \cup A$; i.e., the action executed by the actor are encoded as state variables and so the actor's plan is embedded in its corresponding trajectory.

The sensor model A sensor model is a tuple $\mathcal{M}_s = \langle X, Y, \Phi \rangle$. X are the state variables of the actor, whose values are actually hidden from the observer; Y are the observable variables (each $Y_i \in Y$ has a domain D_{Y_i}); and $\Phi = \{f_1, \ldots, f_{|Y|}\}$ is a set of sensing functions such that each $f_i \in \Phi$ models the sensor of one variable $Y_i \in Y$.

A sensing function $f_i : C_i \rightarrow \wp(D_{Y_i})$ associated to Y_i defines the possible emissions (elements of the powerset $\wp(D_{Y_i})$) that may be emitted when a condition $c \in C_i$ holds in a state. A condition is a partial assignment over X that entails the set of states S_c where c holds. The mapping defined by f_i must be *exhaustive* ($\bigcup_{c \in C_i} S_c = S$) and *exclusive* ($S_c \cap S_{c'} = \emptyset, \forall c, c' \in C_i$) so that a state emits exactly one observed value for each variable Y_i.

An *observation* $\omega = \langle Y_1 = w_1, \ldots, Y_{|Y|} = w_{|Y|} \rangle$, $w_i \in D_{Y_i}$, is a full assignment over the observable variables Y. An observation ω is valid in state s if $\forall i, w_i \in f_i(c_i)$ and $c_i \in s$; i.e., $w_1, \ldots, w_{|Y|}$ are the values observed through the $|Y|$ sensors in a state s which satisfies conditions $c_1, \ldots, c_{|Y|}$.

Now we exemplify a sensor model through the following variables:

- $X = \{loc\_tk_1\}$: one **state variable** of the planning model that represents the location of truck1. For instance, the fact $loc\_tk_1 = \mathtt{h4}$ denotes that truck1 is at location 4 of the Hexagon city; and the fact $loc\_tk_1 = \mathtt{t1}$ represents that truck1 is at location 1 of the Triangular city.
- $Y = \{city\_tk_1\}$: one **observable variable** that represents the city where truck1 is. $D_{Y_1} = \{\mathtt{Hc}, \mathtt{Dc}, \mathtt{Sc}, \mathtt{Tc}\}$ contains the four cities in Figure 1.

The sensor of Y_1 relies on a low resolution GPS located in each truck that reports the city where the truck is, but the exact location within a city is unknown. The sensing function associated to Y_1, $f_1 : C_1 \to \wp(D_{Y_1})$, establishes a relationship between the state variable and the observable variable. When a condition (a fact) of C_1 holds in a state, a value among $\wp(D_{Y_1}) = 2^{D_{Y_1}}$ is a possible emission (observation) of the variable Y_1. For instance, for a condition $c \in \{loc\_tk_1 = \mathtt{h1}, loc\_tk_1 = \mathtt{h2}, \ldots, loc\_tk_1 = \mathtt{h6}\}$, a value that can be emitted for Y_1 (assuming no malfunctioning sensors) is $f_1(c) = \{\mathtt{Hc}\}$; that is, $city\_tk_1 = \mathtt{Hc}$ is an observation broadcast by the GPS that denotes that truck1 is found in the Hexagon city on the condition that the truck is at one location of the Hexagon city. As another example, when $c \in \{loc\_tk_1 = \mathtt{t1}, \ldots, loc\_tk_1 = \mathtt{t3}\}$, a possible emission is $f_1(c) = \{\mathtt{Tc}\}$, which amounts to the observation $city\_tk_1 = \mathtt{Tc}$ being captured through the GPS. Thus, $city\_tk_1 = \mathtt{Tc}$ is **valid observation** in a state s as long as s satisfies one of the conditions of C_1 that may produce the observed value.

The set C_1 includes all possible locations of truck1, which guarantees **exhaustiveness** of the function f_1; that is, since one condition of C_1 will always hold in a state, one emission of the observable variable Y_1 is guaranteed. And the values of C_1 are mutually **exclusive** so as to avoid more than one observation of Y_1. Generally speaking, an observed value maps to a set of exclusive conditions, i.e., to a disjunctive set of values of one or more state variables.

The above examples present a deterministic sensor model because the set of possible emissions, $\wp(D_{Y_1})$, is a singleton. Although out of the scope of this paper, our definition of sensing function supports noisy and non-deterministic sensors. This is the case, for instance, of a weak GPS signal that returns either the correct city or a neighbouring one. We follow a model-based approach to noise where spurious observations that cannot be explained by the sensor model are not supported.

3.2 Working Example

This section presents the whole set of state variables X and observable variables Y on the scenario in Figure 1 that will be used throughout the paper.

State variables. Regarding the example of Figure 1, the set X of the planning model includes:

- For each *package*, a variable loc_pk_p represents the location of package p ; e.g., $loc\_pk_1 = \mathtt{s1} \wedge loc\_pk_2 = \mathtt{s4}$ means that package1 is at location 1 of Square city and that package2 is at location 4 of this same city. These variables also indicate whether a package is loaded onto a truck, in which case the variable is assigned to the corresponding truck, e.g. $loc\_pk_1 = \mathtt{truck1}$.
- For each *driver*, a variable loc_dr_d indicates its location; e.g., $loc\_dr_1 = \mathtt{t3} \wedge \mathtt{loc\_dr}_2 = \mathtt{h3}$ represents that driver1 is at location 3 of the Triangle city and driver2 is at location 3 of the Hexagon city. This variable can also indicate that a driver is on board of a truck, e.g. $loc\_dr_1 = \mathtt{truck1}$.
- For each *truck*, a variable loc_tk_t represents the location of the truck as explained previously.

Observable variables. As we mentioned before, each truck is equipped with a low resolution GPS that reports the city at which the truck is located. In addition, the competent authorities of the cities keep track of the number of available drivers (the ones who are not driving a truck) as well as of the number of undelivered packages (the delivery company has an office in each city that records this information). The exact location of a truck, package, or driver is however unknown. The set Y of observable variables is:

- Variables $city\_tk_t$ indicate the city where a truck t is located as explained in the example above.
- Variables num_pk_c and $free\_dr_c$ indicate the number of undelivered packages and available drivers in city c. E.g., $num\_pk_s = 2$ means there are two packages at Square city; $free\_dr_t = 1$ means that there is one available driver at the Triangular city. Let's call $T_{loc} = \{\mathtt{t1}, \mathtt{t2}, \mathtt{t3}\}$, the set of locations of the Triangle city. In Figure 1, the sensing function that returns the observation $free\_dr_t = 1$ is applicable to all the states that satisfy $loc\_dr_i = l_i \wedge loc\_dr_j = l_j / l_i \in T_{loc} \wedge l_j \notin T_{loc}$. Note that here there is a mapping from the observable variable $free\_dr_t$ to two state variables because the scenario features two drivers.

This particular set of observable variables report an abstraction of the world state, reason why there may exist multiple states that emit the same observation. That is, $\mathcal{M}_s$ emits the same ω from different states. For instance, following the modeled sensors, the observation of the state of Figure 1 is $\omega = \langle num\_pk_d = 0, num\_pk_h = 0, num\_pk_s = 2, num\_pk_t = 0, free\_dr_d = 0, free\_dr_h = 1, free\_dr_s = 0, free\_dr_t = 1, city\_tk_1 = \mathtt{Hc}\rangle$. Different world states can also emit this same observation.

Finally, the set of *actions* A comprises the load/unload actions for packages and trucks, board/debark actions for drivers and trucks, the walk action for drivers and locations, and the drive action for drivers, trucks and locations. For simplicity, we assume all actions have unitary cost.

4 Formalizing Hypotheses

A Temporal Inference Problem (TIP) is an inference task about finding the hypothesis that best explains some observations (our evidence). The starting point of our approach is then a sequence of observations $\{\omega_i\}_{i=1}^{k}$ of the acting agent. Considering that the last observation ω_k is part of the current state of the agent, we are interested in uncovering unobserved segments of the trajectory followed by the agent before ω_k, discerning the current state of the agent, or predicting the trajectory the agent will follow in the future.

We use **conjectures** to talk about the unobserved parts of the trajectory. A conjecture is a constraint over a single state that expresses our assumptions regarding the acting agent at some time point. A hypothesis is an expression that intertwines conjectures with an observation sequence and are used to indistinctly pose queries about the past (e.g., for finding explanations), present (e.g., for diagnosis and monitoring) or future (e.g., for prediction and goal recognition).

4.1 LTL$_\mathbf{f}$ Specifications of Hypotheses

We formulate intertwined sequences of observations and hypotheses in Linear Temporal Logic (LTL). Specifically, we use LTL$_f$ (De Giacomo and Vardi 2013), a variant of LTL interpreted over finite traces which uses the same syntax as that of LTL, and suffices for our purposes.

A formula of LTL$_f$ is written as φ. An atomic formula is an element of $\mathcal{P}$, a set of propositional symbols. A non-atomic formulae is built by applying negation ($\neg$), boolean connective ($\wedge$), and temporal modalities X (next), U (until), G (always) and F (eventually). We also assume the standard boolean abbreviations $true$, $false$, $\vee$ and $\rightarrow$ (implies) as well as the abbreviation $Last$, which stands for $\neg\mathsf{X}true$ and denotes the last instant of the finite trace (De Giacomo and Vardi 2013).

The semantics of a LTL$_f$ formula φ is given in terms of interpretations over *finite trajectories* τ denoting a finite sequence of states at consecutive instants of time $[s_0, s_1, \ldots, s_n]$. Essentially, the definition of a formula φ being true in τ is circumscribed to an instant i ($0 \leq i \leq n$) on the trajectory τ. A formula φ is true in τ if $\tau, i \models \varphi$, $i = 0$. An interpretation is represented as a finite trajectory τ over the alphabet of $2^{\mathcal{P}}$.

For the purposes of this work, we focus our attention in LTL$_f$ formulas that represent an *ordered occurrence* $\mathsf{XF}(\varphi_1 \wedge \mathsf{XF}(\varphi_2 \wedge \ldots \wedge \mathsf{XF}\varphi_n))$. As the name suggests, an ordered occurrence expresses the property that formulas $\varphi_1 \ldots \varphi_n$ must be satisfied in order. We define a syntactic constructor that builds an ordered occurrence from a list of formulas $\Phi = (\varphi_1, \ldots, \varphi_m)$:

$$\begin{aligned} \mathsf{ord}(\Phi) &:= \mathsf{XF}(\Phi[0] \wedge \mathsf{ord}(\Phi[1:])), \text{ when } |\Phi| > 1 \\ &:= \mathsf{XF}\Phi[0], \text{when } |\Phi| = 1 \end{aligned}$$

Note that we use the notation $\Phi[0]$ to denote the first element of the list Φ and $\Phi[1:]$ to denote the tail of the list.

We now explain how we employ LTL$_f$ to define the elements of our hypothesis language. We set $\mathcal{P}$ as the set of assignments to state variables of the form $state\_variable = value$. We first formally introduce a conjecture η and remark that an observation ω can be expressed as a non-atomic formula of LTL$_f$:

- A conjecture η is a Boolean expression over $\mathcal{P}$ and therefore a LTL$_f$ formula.
- Each variable of an observation $\omega = \langle Y_1 = w_1, \ldots, Y_{|Y|} = w_{|Y|}\rangle$ maps to a set of exclusive conditions, and a condition is a partial assignment over X. The LTL$_f$ formula that expresses an observation is $\bigwedge_{i=1}^{|Y|} \bigvee_{j=1}^{|C_i|} c_{ij}$, where $c_{ij} \in 2^{\mathcal{P}}$.

Definition 1 (**Hypothesis**). *A hypothesis is a LTL$_f$ formula* $\mathsf{ord}(\varphi_1, \ldots, \varphi_m)$ *such that* $\varphi_i : \omega_i \wedge \eta_i, 1 \leq i \leq m$.

For a formula $\varphi : \omega \wedge \eta$, φ may denote only an observation if η is $true$, only a conjecture if ω is $true$ or otherwise a joint occurrence of an observation along with a conjecture.

This definition is sufficiently expressive to write hypotheses that intertwine observations and conjectures. For example, the formula $\mathsf{ord}(\omega_1 \wedge true, true \wedge \eta_2, \omega_3 \wedge true, \omega_4 \wedge \eta_4)$ is used to find a trajectory whose states satisfy observation ω_1, later constraint η_2, later observation ω_3 and finally the trajectory contains a state in which $\omega_4 \wedge \eta_4$ holds.

4.2 Examples of Hypotheses

Once presented the specification of hypotheses as LTL$_f$ formulas, we show here some illustrative examples.

Example 1. `truck1` is first observed at some location of the Hexagon city, and it is observed later at some location of the Diamond city. Table 1 shows the observations and conjectures for this example[1].

Time	Observation	Conjecture
1	$\omega_1 : city\_tk_1 = \mathtt{Hc}$	$\eta_1 : true$
2	$\omega_2 : city\_tk_1 = \mathtt{Dc}$	$\eta_2 : true$

Table 1: Hypothesis of Example 1

The LTL$_f$ formula that expresses the hypothesis of Table 1 is $\mathsf{XF}(\varphi_1 \wedge \mathsf{XF}\varphi_2)$ where $\varphi_1 : \omega_1 \wedge \eta_1$ and $\varphi_2 : \omega_2 \wedge \eta_2$. For instance, the LTL$_f$ formula of ω_2 is $loc\_tk_1 = \mathtt{d1} \vee loc\_tk_1 = \mathtt{d2} \vee loc\_tk_1 = \mathtt{d3} \vee loc\_tk_1 = \mathtt{d4}$.

The formula $\mathsf{XF}(\varphi_1 \wedge \mathsf{XF}\varphi_2)$ is true in some trajectory $\tau = [s_0, s_1, \ldots, s_n]$ where `truck1` is at one location of the Hexagon city in some state at instant i such that $0 < i < n$, and for some j such that $i < j \leq Last$ there is a state s_j where we have observed that `truck1` is at one location of the Diamond city.

Example 2. We know that `truck1` has visited the Hexagon city and the Diamond city afterwards. We wish to find out whether `truck1` was driven by `driver2` and it traversed first location `t2` and then `t1` after visiting the Hexagon city and before the Diamond city. Table 2 shows the observations and conjectures for this example.

The formula $\mathsf{XF}(\varphi_1 \wedge \mathsf{XF}(\varphi_2 \wedge \mathsf{XF}(\varphi_3 \wedge \mathsf{XF}\varphi_4)))$ of Example 2 is satisfiable if it is true in some trajectory $\tau = [s_0, s_1, \ldots, s_n]$ such that τ contains four states s_i, s_j, s_k and s_l, $0 < i < j < k < l \leq n$ where ω_1 is true in s_i, η_2 is true in s_j, η_3 is true in s_k and ω_4 is true in s_l.

[1]For simplicity, observable variables whose value is zero in ω are not shown in the examples.

Time	Observation	Conjecture
1	$\omega_1 : city\_tk_1 = \texttt{Hc}$	$\eta_1 : true$
2	$\omega_2 : true$	$\eta_2 : loc\_tk_1 = \texttt{t2} \wedge loc\_dr_2 = \texttt{truck1}$
3	$\omega_3 : true$	$\eta_3 : loc\_tk_1 = \texttt{t1} \wedge loc\_dr_2 = \texttt{truck1}$
4	$\omega_4 : city\_tk_1 = \texttt{Dc}$	$\eta_4 : true$

Table 2: Hypothesis of Example 2

Example 3. We have observed `truck1` at some location of the Triangle city and later in the Diamond city. We hypothesize that `package1` was unloaded in location 4 of the Diamond city. The hypothesis $\mathsf{ord}(\varphi_1, \varphi_2, \varphi_3)$ of this example is shown in Table 3.

Time	Observation	Conjecture
1	$\omega_1 : city\_tk_1 = \texttt{Tc}$	$\eta_1 : true$
2	$\omega_2 : city\_tk_1 = \texttt{Dc}$	$\eta_2 : true$
3	$\omega_3 : true$	$\eta_3 : loc\_pk_1 = \texttt{d4}$

Table 3: Hypothesis of Example 3

5 The Temporal Inference Problem

In this section we formally introduce the *temporal inference problem* (TIP) and describe three relevant families of inference problems.

We formally define a TIP as a tuple $\langle \mathcal{M}, H \rangle$ where:

- $\mathcal{M} = \langle \mathcal{M}_p, \mathcal{M}_s \rangle$ is an augmented model.
- H is a finite and non-empty set of hypotheses.

The *solution* to a TIP is *the most likely hypothesis*, $h^* \in H$. We build upon the assumption that the planning agent is rational (i.e. lower cost trajectories are more likely). The most likely hypothesis is defined as follows:

$$h^* = \underset{h \in H}{\arg\min}\ cost(\tau), \tau \in \mathcal{T}(h) \qquad (1)$$

where $\mathcal{T}(h)$ is the set of trajectories (i.e., interpretations over finite traces in LTL$_f$ semantics) that satisfy the hypothesis h.

A wide palette of inference problems can be expressed as particular instances of a TIP. We focus on three types of TIP of special interest:

- **Monitoring**. Monitoring is a TIP about the current state of the actor. In a monitoring problem the conjectures are exclusively about the world state associated to the last observation. Formally, every hypothesis $h \in H$ of the form $h : \mathsf{ord}(\varphi_1, \ldots, \varphi_m)$ represents a Monitoring problem if:
  - $\forall i, 1 \leq i < m, \varphi_i : \omega_i \wedge true$, and
  - $\varphi_m : \omega_m \wedge \eta_m$
- **Hindsight**. Hindsight is the TIP concerned with uncovering the unobserved behaviour of the planning agent up to the last instant observed. A hypothesis that denotes a hindsight problem describes our conjectures about the past. Formally, every hypothesis $h \in H$ of the form $h : \mathsf{ord}(\varphi_1, \ldots, \varphi_m)$ represents a Hindsight problem if:
  - $\exists k < m$ such that $\varphi_k : true \wedge \eta_k$, and
  - $\forall i, k < i \leq m, \varphi_i : \omega_i \wedge true$
- **Prediction**. The prediction TIP formulates conjectures about the future of the agent. Formally, every hypothesis $h \in H$ of the form $h : \mathsf{ord}(\varphi_1, \ldots, \varphi_m)$ represents a Prediction problem if:
  - $\exists k < m$ such that $\forall i, 1 \leq i \leq k, \varphi_i : \omega_i \wedge true$, and
  - $\forall j, k < j \leq m, \varphi_j : true \wedge \eta_j$

5.1 Examples of TIPs

We exemplify the three types of TIP using the initial state illustrated in Figure 1. We will assume the two-observation sequence of Table 4 as the evidence for all the examples in this section. The observations portray that we first observed two packages at the Square city, one free driver at the Triangle city and that `truck1` is in Hexagon city; and afterwards we observed one package less in the Square city and `truck1` in the Diamond city.

Time	Observation
1	$\omega_1 : num\_pk_s{=}2 \wedge free\_dr_t{=}1 \wedge loc\_tk_1{=}\texttt{Hc}$
2	$\omega_2 : num\_pk_s{=}1 \wedge free\_dr_t{=}1 \wedge loc\_tk_1{=}\texttt{Dc}$

Table 4: Observation sequence for the TIP examples

Monitoring. An example of *monitoring* is identifying the driver who is currently on board of `truck1`, for which we generate two alternative hypotheses $H = \{h, h'\}$, one where `truck1` is driven by `driver1` and another where it is driven by `driver2`. The first two rows of Table 5 represent the hypothesis h and the other two rows the hypothesis h'.

Time	Observation	Conjecture
$1(h)$	ω_1	$\eta_1 : true$
$2(h)$	ω_2	$\eta_2 : loc\_dr_1 = \texttt{truck1}$
$1(h')$	ω_1	$\eta_1 : true$
$2(h')$	ω_2	$\eta_2 : loc\_dr_2 = \texttt{truck1}$

Table 5: Hypotheses of the monitoring example

Hindsight. An example of *hindsight* is determining whether the package picked up between the two observations is `package1` or `package2`. We formulate the two hypotheses shown in Table 6. Note that intertwining conjecture η_2 moves the second observation to time index 3.

Prediction. Let us assume the delivery point of the two packages is location `d1` of the Diamond city. An example of a *prediction* TIP is worded as follows: ”after having observed ω_1 and ω_2, will `package1` be delivered before `package2` or vice versa?”

This task is formulated with the hypotheses of Table 7 where h conjectures that `package1` is delivered first and h' conjectures the reverse order.

Time	Observation	Conjecture
$1(h)$	ω_1	$\eta_1 : true$
$2(h)$	$\omega_2 : true$	$\eta_2 : loc\_pk_1 = \texttt{truck1}$
$3(h)$	ω_3	$\eta_3 : true$
$1(h')$	ω_1	$\eta_1 : true$
$2(h')$	$\omega_2 : true$	$\eta_2 : loc\_pk_2 = \texttt{truck1}$
$3(h')$	ω_3	$\eta_2 : true$

Table 6: Hypotheses of the hindsight example

Time	Observation	Conjecture
$1(h)$	ω_1	$\eta_1 : true$
$2(h)$	ω_2	$\eta_2 : true$
$3(h)$	$\omega_3 : true$	$\eta_3 : loc\_pk_1\texttt{=d1} \wedge \neg loc\_pk_2\texttt{=d1}$
$4(h)$	$\omega_4 : true$	$\eta_4 : loc\_pk_2\texttt{=d1}$
$1(h')$	ω_1	$\eta_1 : true$
$2(h')$	ω_2	$\eta_2 : true$
$3(h')$	$\omega_3 : true$	$\eta_3 : loc\_pk_2\texttt{=d1} \wedge \neg loc\_pk_1\texttt{=d1}$
$4(h')$	$\omega_4 : true$	$\eta_4 : loc\_pk_1\texttt{=d1}$

Table 7: Hypotheses of the prediction example

The two hypotheses of Table 7 show a sequence of conjectures rather than an instant-wise conjecture. The same can be applied to the Hindsight TIP to query about the specific order of two past occurrences.

6 Temporal Inference as Planning

In this section we present the reduction of a TIP $\langle \mathcal{M}, H \rangle$ to $|H|$ classical planning instances, one for each hypothesis $h \in H$. Intuitively, each planning instance implements the product of the non-deterministic automaton that corresponds to the planning model M_p and a deterministic finite automaton (DFA) that accepts trajectories that satisfy the hypothesis h, thus restricting the possible solutions of the planning instance to $\mathcal{T}(h)$. *The most likely hypothesis* is the hypothesis that corresponds to the planning problem whose optimal plan presents the minimum cost.

6.1 Building a DFA for a Hypothesis

Our compilation to planning exploits the fact that every LTL$_f$ formula can be translated into an equivalent DFA. The details of this translation can be found in (De Giacomo and Vardi 2013). A DFA is specified as a tuple $A = \langle Q, \Sigma, \delta, q_0, F \rangle$, where:

- Q is a finite set of states,
- Σ is the input alphabet,
- $\delta : Q \times \Sigma \to Q$ is the state transition function,
- q_0 is the initial state, and
- $F \subseteq Q$ is the set of accepting states.

When a DFA is in a state $q \in Q$, and receives a symbol $\sigma \in \Sigma$, it transitions to state $q' = \delta(q, \sigma)$. This process starts with the initial state $q = q_0$, and it is repeated for $q = q'$, until the input sequence of symbols is exhausted. At this point, iff the automaton ends at a state that belongs to F, the input sequence of symbols is accepted.

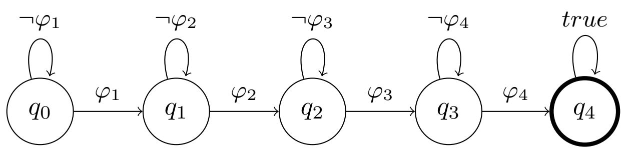

Figure 2: DFA for the hypothesis of Example 2 in section 4.2.

The ordered occurrence property denoted by a hypothesis in a TIP translates to a DFA with a forward topology and self-loops as illustrated in Figure 2. This DFA can be computed in linear time in the number of subformulas φ_i of $\mathsf{ord}(\varphi_1, \ldots, \varphi_m)$. Given a hypothesis $h : \mathsf{ord}(\varphi_1, \ldots, \varphi_m)$, we define $A_h = \langle Q, \Sigma, \delta, q_0, F\}\rangle$ as follows:

- $Q = \{q_i \mid 0 \leq i \leq m\}$ contains one state for each subformula φ_i in $h : \mathsf{ord}(\varphi_1, \ldots, \varphi_m)$ plus an additional initial state.
- $\Sigma = 2^{\mathcal{P}}$ where $\mathcal{P}$ is the set of assignments to state variables of the form $state\_variable = value$.
- δ defines transitions $q_i = \delta(q_{i-1}, \varphi_i)$, $1 \leq i \leq m$ to advance the state of the automaton, and defines loop transitions of the form $q_{i-1} = \delta(q_{i-1}, \neg\varphi_i)$ to remain in the same state.
- $F = \{q_m\}$.

The states of A_h represent the current progress in the validation of the satisfiability of a hypothesis $h : \mathsf{ord}(\varphi_1, \ldots, \varphi_m)$ by a trajectory τ. Being at state q_i means thereby that τ satisfies h up to φ_i. When a state s of a trajectory τ is processed in the automaton state q_{i-1}, the transition $q_i = \delta(q_{i-1}, \varphi_i)$ is triggered iff $s \models \varphi_i$, progressing the automaton to state q_i; otherwise the automaton remains at q_{i-1} through transition $q_{i-1} = \delta(q_{i-1}, \neg\varphi_i)$. Consequently, reaching state q_m means that τ satisfies h.

6.2 The Classical Planning Problem

Given a TIP $\langle \mathcal{M}, H \rangle$, this section shows how to build the classical planning instance $P_h = \langle X_h, A_h, I_h, G_h \rangle$ that corresponds to the hypothesis $h \in H$. Building P_h mainly involves extending the state variables and actions of the actor's planning model $\mathcal{M}_p \in \mathcal{M}$ with new variables and actions to represent the states and transitions of the DFA A_h.

Initial state and goals. Similarly to the compilation scheme for *planning with extended goals* (Baier and McIlraith 2006), X_h extends the set of original state variables with extra variables to keep track of the current state of the DFA A_h. Accordingly, the initial state I_h is also extended with the initial state of the A_h. The goal G_h of the classical planning instance comprise now only the single accepting state q_m of the automaton. Additionally, we introduce a Boolean variable $invalid$ which must be false in the goal state and is used to ensure that solutions satisfy the hypotheses.

Actions. The set A_h of the classical planning instance P_h contains two types of actions: *transition actions*, which implement the state transition function of the acting agent and *validation actions*, which implement the state transition function of A_h:

- **Transition actions**. These are the same actions as in the original planning model $\mathcal{M}_p$.
- **Validation actions**. A new $\mathsf{validate_i}$ action is introduced for every transition $q_i = \delta(q_{i-1}, \varphi_i)$, $1 \leq i \leq m$, in the DFA A_h. The applicability of the $\mathsf{validate_i}$ action requires that the automaton is at state q_{i-1}, and its execution updates the automaton state to q_i. The conjecture part of φ_i is encoded straightforward by inserting η_i in the preconditions of the $\mathsf{validate_i}$ action. Following the same approach for the observation part would introduce disjunctions in the precondition so we decided against. Instead, the observation is encoded as conditional effects with condition $c \in \{c \in C_i \mid w_i \notin f_i(c)\}$, $1 \leq i \leq |Y|$ and effect $invalid = true$. This produces a dead-end (through variable $invalid$) if a validation action associated to some observation ω is executed in a state for which ω is not a valid observation.

Example. Revisiting once again the hypothesis of Example 2, we have that `truck1` is observed first at Hexagon city and later at Diamond city and we conjecture that `truck1` traversed locations `t2` and `t1` of the Triangle city. In every solution plan that satisfies this hypothesis, `truck1` must initially be in one of the locations among `h1`, ..., `h6`; then `truck1` needs to be at `t2`, then at `t1`, and finally it must be in one of the locations of the Diamond City. Coding the DFA for this hypothesis requires four validating actions: $\mathsf{validate_1}$, executable in a state where `truck1` is at any location of the Hexagon city; $\mathsf{validate_2}$, executable after $\mathsf{validate_1}$ only in a state in which `truck1` is at location `t2` and driven by `driver2`, $\mathsf{validate_3}$, executable after $\mathsf{validate_2}$ in a state in which `truck1` is at location `t1` and driven by `driver2`; and $\mathsf{validate_4}$, executable after $\mathsf{validate_3}$ from any state that features `truck1` in a location of the Diamond city.

Theorem 1. *Completeness. For every trajectory τ that satisfies a hypothesis h there exists a solution plan π for P_h such that π induces τ.*

Theorem 2. *Soundness. If π is a solution plan for P_h then the trajectory τ induced by π satisfies the hypothesis h.*

Proof. The planning problem P_h extends M_p with only the state variables and sensing actions needed to model A_h. No other constraint is added that affects the applicability of the actions of the original planning model or their execution cost. Therefore, solutions to P_h are only restricted by A_h which strictly defines the language $\mathcal{T}(h)$. Additionally, the goal condition G_h is to reach automaton state q_m of A_f which can only be reached after processing a trajectory that satisfies h. □

The size of the resulting classical planning problem is determined by the number of conditional effects of the *validating actions*, which is given by (i) the sensor model and (ii), the size of the hypothesis.

7 Evaluation

This section evaluates the ability of an off-the-shelf planner to solve the palette of TIPs presented in this work.

7.1 Experimental Setup

All experiments were run on an Intel Core i5 3.10GHz x 4 16GB of RAM. The planning problems were optimally solved with the FAST-DOWNWARD planning system, following the configuration for the *merge and shrink* heuristic (Helmert et al. 2014) that uses bisimulation based shrinking, the merge strategy SCC-DFP, and the appropriate label reduction setting (Sievers, Wehrle, and Helmert 2016), as recommended at the http://www.fast-downward.org/ website. Planning problems are solved with a timeout value of 120 secs of CPU-time, and 8GB of memory. The source code and test benchmark for the evaluation is available at https://github.com/anonsubs/planning_inference. Our implementation uses the LTL$_f$2DFA tool[2] to transform general LTL$_f$ formulas into minimal DFAs as well as our own transformation optimized for ordered occurrence formulas.

We tested our approach on benchmark TIPs of **monitoring**, **hindsight**, and **prediction**, in five well-known classical planning domains from the IPC: *grid*, *miconic*, *driverlog*, *floortile*, and *openstacks*. A TIP $\langle \mathcal{M}, H \rangle$ is constructed as follows:

- For the augmented model $\mathcal{M} = \langle \mathcal{M}_p, \mathcal{M}_s \rangle$ we use the planning model of the original domain[3], and define a sensor model in the same way as the one used in our working example. This means that the values of the observable variables Y are high-level observations of the world states that convey how many objects meet some particular property, but not the actual identity of the objects. For instance, in the *miconic* domain, an observation is "the number of passengers inside a lift"; in the *openstacks* domain, an observation is "the number of waiting/started/shipped orders". In both cases, the actual identity of the passengers/orders remains unknown to the observer.
- H is a set of hypotheses regarding the current state in monitoring TIPs, a sequence of past states in hindsight TIPs, and a sequence of future states in prediction TIPs. Hypotheses allow us to answer queries such as "which are the currently opened locks?" (*grid*), "what was the order of the first four painted tiles?" (*floortile*), and "what will be the order in which the last three packages will be delivered?" (*driverlog*). For each TIP, we created a set H in which only one of the hypothesis is correct with respect to the trajectory τ. Observations were collected by observing some trajectory τ of the agent with the sensor model. We also define an observability parameter that determines the percentage of states of τ that emit an observation.

7.2 Empirical Results

Table 8 summarizes the results of evaluating our approach on monitoring, hindsight and prediction TIPs. Column "O%" is the observability parameter (30%, 50%, or 70%).

[2]https://github.com/whitemech/LTLf2DFA
[3]http://api.planning.domains

		Monitoring				**Hindsight**				**Prediction**			
Domain	O%	$\lvert H\rvert$	$\lvert H^*\rvert$	Q	T	$\lvert H\rvert$	$\lvert H^*\rvert$	Q	T	$\lvert H\rvert$	$\lvert H^*\rvert$	Q	T
	30		1.00	1.00	7.97		1.00	1.00	135.52		1.00	1.00	32.17
Grid	50	5.40	1.00	1.00	27.85	2.00	1.00	1.00	146.05	2.00	1.00	1.00	13.73
	70		1.00	1.00	65.78		1.00	1.00	138.33		1.00	1.00	17.79
	30		1.10	1.00	53.22		1.90	0.90	149.04		3.00	1.00	140.57
Miconic	50	6.00	1.10	1.00	180.29	5.00	2.00	1.00	140.24	5.00	2.60	1.00	255.56
	70		1.20	1.00	120.13		1.80	1.00	121.02		2.60	1.00	320.65
	30		1.40	0.90	162.61		1.00	1.00	238.16		1.00	1.00	169.24
Driverlog	50	5.40	1.40	0.90	284.90	5.30	1.00	1.00	399.23	5.30	1.00	1.00	250.70
	70		1.40	0.90	390.91		1.00	1.00	560.77		1.00	1.00	349.16
	30		1.30	0.40	37.31		4.70	0.80	443.10		3.00	1.00	676.76
Floortile	50	5.90	1.30	0.50	56.97	6.00	4.00	0.80	141.56	6.00	3.80	0.90	713.32
	70		1.50	0.60	44.53		3.50	0.80	128.48		4.70	1.00	734.02
	30		2.30	0.80	5.93		2.00	1.00	11.19		2.50	1.00	15.70
Openstacks	50	5.60	2.30	0.80	6.81	5.80	1.70	1.00	11.32	5.90	2.50	1.00	20.30
	70		2.30	0.80	9.60		1.60	1.00	14.28		2.50	1.00	18.77

Table 8: Results for temporal inference problems of *monitoring*, *hindsight*, and *prediction* in five different domains and for three different levels of observability.

For each type of TIP, we report the size of the hypotheses set ($|H|$), the number of most likely hypotheses as computed by our approach ($|H^*|$), the ratio of TIPs whose correct hypothesis is among H^* (Q), and the accumulated time (T) in secs to solve the TIP. We note that solving a TIP implies solving $|H|$ planning instances P_h, one instance for each h in H.

The best score is obtained when $|H^*| = 1$ and $Q = 1$, meaning that only one hypothesis is recognized as the most likely and that such hypothesis is the correct one. For each combination of TIP (monitoring, hindsight and prediction), domain and observability percentage, we report in Table 8 the average score across 10 problems. Thus, our approach was evaluated over $5 \times 3 \times 3 \times 10 = 450$ TIPs.

As expected, the best results, i.e. larger values for Q and/or lower values for $|H^*|$, are obtained for high degrees of observability since more informative evidence is available. Yet, we also find quite a few cases of 30% observability that achieve the top performance. Increasing observability comes however at the cost of longer planning solutions (more validating actions), which require longer computation times. The prediction results for the *floortile* in Table 8 show that increasing the observability may actually hinder the performance of the system as computation times are pushed over the set timeout of 120s.

Interestingly, we can observe in Table 8 that the worst quality results are found in monitoring TIPs and to a lesser extent in the hindsight tasks. This is particularly remarkable in the *floortile* domain. The reason behind these results is that the observations only convey a small subset of the final goals that the actor pursues in the trajectory τ. More specifically, the hypothesis only reveals the goals up to the last observation. This happens, for example, when the goal of the actor is to have the whole board painted and the last observation only shows a handful of the painted tiles. This mismatch between the information conveyed by the observations and the goal pursued by the agent in τ causes a drop of the quality, as the prefix of an optimal plan is not necessarily optimal with respect to the subset of achieved goals. This limitation can easily be overcome if the agent's goals are known, in which case they will be added as the last conjecture of the hypothesis.

In summary, the results show the versatility of our approach for solving temporal inference problems. We can also conclude an overall good-quality results even in the case of problems with a low level of informedness of the observations.

8 Related Work

Several of the inference problems found in the planning literature can be formulated as instances of our temporal inference problem. *Goal recognition* (Ramírez and Geffner 2010; Vered, Kaminka, and Biham 2016), for example, is a particular instance of our prediction TIP where hypotheses are limited to conjectures at a single point (the goal state) that must necessarily occur after the given sequence of observations. This limitation of the hypothesis language is also present in related work that builds on top of the goal-recognition as planning scheme; e.g., *goal recognition design* (GRD) (Keren, Gal, and Karpas 2020) or *counter-planning* (Pozanco et al. 2018). Likewise, a particular case of hindsight problem is the assumption-based diagnosis with assumptions regarding the initial state and observations within the diagnosis trajectory (Sohrabi, Baier, and McIlraith 2010).

More recently, we can find some works in the line of recognizing temporally extended goals, which amounts to infer the exact order that a set of facts of a goal must be achieved in a plan (Fraga Pereira, Fuggitti, and De Giacomo 2021). In this work, goals are formalized in LTL$_f$ and Pure Past LTL (PLTL$_f$) (De Giacomo et al. 2020). The use of PLTL$_f$ allows expressing past temporal goals, similarly to the hypotheses in our hindsight problem. Our approach however is restricted to a subset of LTL$_f$ that is sufficiently expressive to reason about past, present and future in linear trajectories, thus avoiding the worst-case double-exponential cost in the size of the formula of the DFA transformation.

Most of the works in the literature adopt a fairly constrained definition of observation usually limited to the actions executed by the agent although some exceptions exist. The work in (Keren, Gal, and Karpas 2020) deals with noisy (non-deterministic) sensor models that emit one among several observable tokens but these are likewise only over actions. Exceptionally, extending observations to handle noise over state fluents is proposed in (Sohrabi, Riabov, and Udrea 2016) but this formulation is constrained to a single fluent per observation and noisy observations are handled by skipping them. The idea of skipping observations is better suited for atomic observations (such as actions) that are either right or wrong. In our formulation we have several observable variables, and a single observation may contain a noisy reading for one variable and a correct one for another. In this work, we follow a model-based approach to noise that checks whether that noisy reading is possible with the given sensor model. Additionally, our definition of sensor model allows a broader range of observations to fit the needs of the problem at hand which include features as the ones used in this paper as well as noisy observations.

Our approach can be straightforwardly extended to a probabilistic formulation of a TIP by exploiting the notion of log-probability that allows an optimal planner to maximize the accumulated transition and emissions probabilities defined by a probabilistic specification of $\mathcal{M}_p$ and $\mathcal{M}_s$. However, given that current off-the-shelf planners are ill-suited to optimize state-dependent costs, the evaluation of a probabilistic setting would be limited to a handful of problems with small state spaces (Aineto, Jimenez, and Onaindia 2020).

Temporal inference is also a common task in probabilistic graphical models, a type of state-space models represented by a probabilistic state-transition function and a probabilistic observation function. In graphical models, the tasks of inferring the past, current and future state of the system given all evidence up to present are called smoothing, filtering and prediction, respectively (Koller and Friedman 2009). Probabilistic models handle rich sensor models but they are not particularly well suited for decoding gapped sequences observed at intermittent times (e.g. when sensing the environment regularly is too costly). Although some recent approaches have been proposed to address missing observations associated to state transitions (Yu 2016), these models are specifically designed to deal with bounded observation sequences.

9 Conclusions

In this paper, we have presented a formalism that given some evidence about the behaviour of an acting agent allows for posing hypothesis about the past, present or future trajectory of the agent. The hypotheses are expressed as LTL$_f$ formulas and unlike other inference schemes ours account for explaining and predicting the order in which two or more variables are achieved. The paper shows that the wide landscape of diverse inference tasks in this setting can be addressed with a unified method that leverages classical planning.

In the future, we plan to investigate the application of this inference framework to planning models that use a different notion of plan optimality such as, for instance, the plan makespan used in temporal planning (Cushing, Kambhampati, and Weld 2007).

Acknowledgments

This work is supported by the Spanish MINECO project TIN2017-88476-C2-1-R and partially supported by the EU ICT-48 2020 project TAILOR (No. 952215). Diego Aineto is partially supported by the *FPU16/03184* and Sergio Jiménez by the *RYC15/18009*.

References

Aineto, D.; Jimenez, S.; and Onaindia, E. 2020. Observation decoding with sensor models: Recognition tasks via classical planning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 30, 11–19.

Baier, J. A., and McIlraith, S. A. 2006. Planning with first-order temporally extended goals using heuristic search. In *National Conference on Artificial Intelligence, (AAAI-06)*.

Chakraborti, T.; Kulkarni, A.; Sreedharan, S.; Smith, D. E.; and Kambhampati, S. 2019. Explicability? legibility? predictability? transparency? privacy? security? the emerging landscape of interpretable agent behavior. In *29th International Conference on Automated Planning and Scheduling, ICAPS 2019*, 86–96.

Cushing, W.; Kambhampati, S.; and Weld, D. S. 2007. When is temporal planning really temporal? In *Proceedings of the 20th international joint conference on Artifical intelligence*, 1852–1859.

Davis-Mendelow, S.; Baier, J. A.; and McIlraith, S. A. 2013. Assumption-based planning: Generating plans and explanations under incomplete knowledge. In *27th AAAI Conference on Artificial Intelligence*.

De Giacomo, G., and Vardi, M. Y. 2013. Linear temporal logic and linear dynamic logic on finite traces. In Rossi, F., ed., *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*, 854–860. IJCAI/AAAI.

De Giacomo, G.; Stasio, A. D.; Fuggitti, F.; and Rubin, S. 2020. Pure-past linear temporal and dynamic logic on finite traces. In Bessiere, C., ed., *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, 4959–4965.

E-Martín, Y.; R.-Moreno, M. D.; and Smith, D. E. 2015. A fast goal recognition technique based on interaction estimates. In *Proc. 24th International Joint Conference on Artificial Intelligence, IJCAI 2015*, 761–768.

Fox, M., and Long, D. 2003. PDDL2.1: An extension to PDDL for expressing temporal planning domains. *Journal of Artificial Intelligence Research* 20:61–124.

Fraga Pereira, R.; Fuggitti, F.; and De Giacomo, G. 2021. Recognizing LTLf/PLTLf Goals in Fully Observable Non-Deterministic Domain Models. *CoRR* abs/2103.11692.

Fritz, C., and McIlraith, S. A. 2007. Monitoring plan optimality during execution. In *7th International Conference* *on Automated Planning and Scheduling, ICAPS-2007 2007*, 144–151. AAAI.

Grastien, A.; Haslum, P.; Thiébaux, S.; et al. 2011. Exhaustive diagnosis of discrete event systems through exploration of the hypothesis space. In *22nd International Workshop on Principles of Diagnosis (DX-11)*, 60–67.

Haslum, P., and Grastien, A. 2011. Diagnosis as planning: Two case studies. In *Proc. of the International Scheduling and Planning Applications workshop (SPARK)*.

Helmert, M.; Haslum, P.; Hoffmann, J.; and Nissim, R. 2014. Merge-and-shrink abstraction: A method for generating lower bounds in factored state spaces. *Journal of the ACM (JACM)* 61(3):1–63.

Keren, S.; Gal, A.; and Karpas, E. 2019. Goal Recognition Design in Deterministic Environments. *Journal of Artificial Intelligence Research* 65:209–269.

Keren, S.; Gal, A.; and Karpas, E. 2020. Goal recognition design - survey. In *29th International Joint Conference on Artificial Intelligence, IJCAI-2020*, 4847–4853.

Kim, J.; Woicik, M. E.; Gombolay, M. C.; Son, S.; and Shah, J. A. 2018. Learning to infer final plans in human team planning. In Lang, J., ed., *27th International Joint Conference on Artificial Intelligence, IJCA-2018I 2018*, 4771–4779.

Koller, D., and Friedman, N. 2009. *Probabilistic Graphical Models - Principles and Techniques*. MIT Press.

Pereira, R. F.; Oren, N.; and Meneguzzi, F. 2020. Landmark-based approaches for goal recognition as planning. *Artif. Intell.* 279.

Pozanco, A.; E.-Martín, Y.; Fernández, S.; and Borrajo, D. 2018. Counterplanning using goal recognition and landmarks. In *International Joint Conference on Artificial Intelligence, (IJCAI-18)*, 4808–4814.

Ramírez, M., and Geffner, H. 2009. Plan Recognition as Planning. In *International Joint conference on Artifical Intelligence, (IJCAI-09)*, 1778–1783. AAAI Press.

Ramírez, M., and Geffner, H. 2010. Probabilistic Plan Recognition Using Off-the-Shelf Classical Planners. In *24th AAAI National Conference on Artificial Intelligence, (AAAI-10)*.

Sievers, S.; Wehrle, M.; and Helmert, M. 2016. An analysis of merge strategies for merge-and-shrink heuristics. In *Proceedings of the 26th International Conference on Automated Planning and Scheduling*.

Sohrabi, S.; Baier, J. A.; and McIlraith, S. A. 2010. Diagnosis as planning revisited. In *12th International Conference on the Principles of Knowledge Representation and Reasoning*.

Sohrabi, S.; Baier, J. A.; and McIlraith, S. A. 2011. Preferred explanations: Theory and generation via planning. In *Proc. of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2011*. AAAI Press.

Sohrabi, S.; Riabov, A. V.; and Udrea, O. 2016. Plan recognition as planning revisited. In *International Joint Conference on Artificial Intelligence, (IJCAI-16)*, 3258–3264.

Sukthankar, G.; Goldman, R. P.; Geib, C.; Pynadath, D. V.; and Bui, H. 2014. *Plan, Activity, and Intent Recognition: Theory and Practice*. Morgan Kaufmann.

Vered, M.; Kaminka, G. A.; and Biham, S. 2016. Online goal recognition through mirroring: Humans and agents. In *The Fourth Annual Conference on Advances in Cognitive Systems*.

Yu, S. 2016. *Hidden Semi-Markov Models: Theory, Algorithms and Applications*. Boston: Elsevier.