Beyond Static Assumptions: the Predictive Justified Perspective Model for Epistemic Planning

Guang Hu[1], **Weijia Li**[1], **Yangmengfei Xu**[1]

[1]The Faculty of Engineering and Information Technology, The University of Melbourne, Australia
{ghu1, weijia3, yangmengfeix}@student.unimelb.edu.au

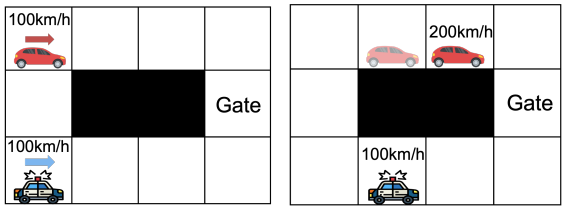
Abstract

Epistemic Planning (EP) creates agents capable of reasoning about the nested beliefs of others. However, existing frameworks fundamentally rely on the "static-environment" assumption inherited from classical planning. This constraint limits their applicability in dynamic settings where variables evolve independently of agent actions (e.g., moving targets or falling objects). To relax this assumption, we introduce the Predictive Justified Perspective (PJP) model. Unlike previous models that assume unobserved variables remain the same (unless observed evidence suggests otherwise), PJP allows agents to use their history of observations to predict how variables change over time. We formally define this model and prove that it retains the computational efficiency of state-based planning while ensuring logically sound belief reasoning (satisfying the **KD45** axioms). Experimental results on the Grapevine benchmark show that PJP successfully extends EP to dynamic environments, allowing agents to reason about nested beliefs regarding changing values—a capability absent in prior work.

1 Introduction

Epistemic planning (EP) is a planning paradigm that extends classical planning with explicit reasoning about agents' (nested) knowledge and beliefs. Such reasoning is valuable in multi-agent systems and human–computer interaction scenarios, where agents must make decisions that depend on the knowledge and beliefs of others.

A critical limitation of existing work in EP is the assumption that the environment remains static unless explicitly altered by an agent's actions—a constraint inherited from classical planning. While this "static" assumption holds in classical formulations where the world state is fully controlled by the agent, it breaks down in dynamic, physical environments. For example, treating the trajectory of a falling object or the movement of pedestrians as static is unrealistic. Although approaches related to planning frameworks such as PDDL+ (Fox and Long 2006), F-STRIPS (Francès and Geffner 2015), or Planning Against Nature (Chrpa and Karpas 2025) can model such dynamics, current EP frameworks lack such mechanisms. This discrepancy creates a gap between current EP formalisms and real-world deployment.



(a) Initial (b) Next timestamp

Figure 1: Motivative Example: Thief and Police

Bridging this gap requires an epistemic reasoning model capable of accounting for dynamic environmental changes.

The scenario in Figure 1 illustrates the necessity of integrating dynamic prediction models into epistemic reasoning. Consider a pursuit-evasion context where a police unit attempts to intercept rather than chase a thief. Initially (Figure 1a), both agents move at steady speeds to minimize detection risk before entering an area with obstructed visibility.

Upon moving behind the obstruction (Figure 1b), mutual line-of-sight is lost. Since the thief is no longer observable, a robust epistemic model should not assume that the thief remains at the last observed position; instead, it should predict the thief's trajectory from the police's observation history. Meanwhile, the thief can reason at a nested-belief level about the police's belief regarding where the thief will be. By accelerating to 200 km/h, the thief reaches an unanticipated location (the solid car), while the police falsely believes the thief to be at the predicted interception point (the faded car). Thus, the example involves not only belief about a dynamic variable, but also nested belief and false belief: the thief acts based on its belief about the police's predicted belief, and deliberately makes that belief false. Modeling such deceptive strategies requires an EP framework capable of deriving agents' beliefs through prediction models over partially observed dynamic variables.

Currently, EP is primarily addressed through three main approaches. The foundational approach, grounded in *Dynamic Epistemic Logic* (DEL) (Bolander and Andersen 2011), models knowledge via Kripke structures (Fagin et al. 1995) updated through explicit event models. Conversely, *compilation-based* strategies seek to transform epistemic problems into classical planning tasks, managing belief up-

dates within standard state variables (Muise et al. 2015, 2022; Cooper et al. 2019). However, both paradigms face significant scalability hurdles as the depth of nested beliefs increases.

To address this challenge, recent research has advanced a *state-based* paradigm exemplified by *Planning with Perspectives* (PWP) (Hu, Miller, and Lipovetzky 2022) and its belief-centric extension, the *Justified Perspectives* (JP) model (Hu, Miller, and Lipovetzky 2023). This framework utilizes lazy evaluation and external functions to offload complex reasoning. Crucially, the JP model defines a semantics where agent beliefs are derived constantly from past observations rather than requiring explicit action effects for every epistemic logical update.

We identify the state-based JP model as the most promising candidate for relaxing the "static" assumption. Its reliance on history-based derivation allows it to accommodate exogenous environmental changes naturally, without the rigid event-model specifications required by DEL.

In this paper, we formally define the Predictive Justified Perspective (PJP) model, which extends the JP framework by introducing processual variables and predictive retrieval functions to capture and reason about dynamic environmental dynamics. Theoretically, we prove that the PJP model preserves the desirable properties of the JP model: it maintains polynomial-time evaluation with respect to nesting depth and, provided the prediction functions satisfy specific consistency constraints, satisfies the **KD45** axiomatic system, ensuring that agents' predictive beliefs remain logically well-behaved—closed under implication, internally consistent, and introspective. Finally, we implement the PJP model and its semantics using PDDL+ and evaluate the approach on a dynamic variant of the *Grapevine* domain. The results demonstrate that our approach effectively reasons about nested beliefs in dynamic environments—a capability currently absent in other epistemic planners.

2 Preliminaries of the JP Model

The JP model (Hu, Miller, and Lipovetzky 2023) follows the intuition of justified belief (Goldman 1979): unless agents see evidence to the contrary, they believe that previously observed values persist. Specifically, when agents reason about unobservable variables, if there is no evidence suggesting that these variables have changed, they form justified beliefs by retrieving values from their past observations. "Justified perspectives" is the term used to represent what agents justifiably believe the sequence of states to be.

A JP **Signature** is defined as a tuple: $\Sigma = (Agt, V, \mathbb{D}, \mathcal{R})$ where Agt represents a finite set of agent identifiers containing all agents. The set V is a finite set of variables such that $Agt \subseteq V$, which is essential for allowing arbitrary nesting in beliefs. For each variable $v \in V$, D_v denotes a potentially infinite domain of constant symbols (containing a special value $\perp$ to represent a none value). The domains can be either discrete or continuous, and the overall set of values is given by $\mathbb{D} = \bigcup_{v \in V} \{D_v\}$. Additionally, $\mathcal{R}$ is a finite set of predicate symbols r.

With the above signature, the state s is defined as a set of assignments that match the variable $v \in V$ and any value $e_v \in D_v$ in the format of $v = e_v$. The value of the variable v in a state s can be represented by $s(v)$. The state space is denoted as $\mathcal{S}$ and the complete-state (complete assignments) space is denoted as $\mathcal{S}_c$. A global state is a complete state, while a local state might be a partial state. Besides, the state sequence is denoted as $\vec{s}$ with an index from 0 to n, where $n \in \mathbb{N}$, and the state with an index of t from this sequence can be represented by $\vec{s}[t]$, and its value for the variable v can be represented as $\vec{s}[t](v)$. Similarly, the sequence space is denoted as $\vec{\mathcal{S}}$ and the complete-sequence space is $\vec{\mathcal{S}}_c$. Besides, the following override function is provided to update states.

Definition 1 (State Override Function)**.** Given a state s and s', the state override function $\langle\rangle : \mathcal{S} \times \mathcal{S} \to \mathcal{S}$ to override s' with s is defined as follows:

$$s'\langle s \rangle = s \cup \{v = s'(v) \mid v \in s' \text{ and } v \notin s\} \quad \blacksquare$$

The above function overwrites state s' with state s, preserving assignments from s and maintaining those variables found only in s' but not in s. In addition, the state override function can also be applied to a sequence of states, where $s'\langle \vec{s} \rangle = [s'\langle \vec{s}[0] \rangle, \ldots, s'\langle \vec{s}[n] \rangle]$.

Definition 2 (Language)**.** The language of Knowledge and Belief, $L_{KB}(\Sigma)$, of the JP model is defined by the grammar:

$$\varphi ::= r(V_r) \mid \neg\varphi \mid \varphi \wedge \varphi \mid S_i v \mid S_i\varphi \mid K_i\varphi \mid B_i\varphi,$$

where $r \in \mathcal{R}$, $V_r \subseteq V$ are the terms of r, and $r(V_r)$ forms a predicate ; $v \in V$, and $i \in Agt$. $\blacksquare$

$S_i v$ means agent i sees variable v. Similarly, $S_i\varphi$, $K_i\varphi$, and $B_i\varphi$ mean that agent i sees, knows, and believes formula φ, respectively. The difference between knowledge and belief is that the knowledge of φ is consistent with the actual world while the belief may not be the truth. For illustration, in this paper, we denote the set of all predicates as $\mathcal{P}$.

Definition 3 (JP Model)**.** With the above signature and language, the JP **model** M is defined as:

$$M = (Agt, V, \mathbb{D}, \pi, O_1, \ldots, O_{|Agt|}),$$

where: Agt, V and $\mathcal{D}$ are from the signature Σ; π is an interpretation function, $\pi : \mathcal{S} \times \mathcal{P} \to \{true, false\}$, that determines whether the predicate $r(V_r)$ is true in s. Finally, $O_1, \ldots, O_{|Agt|}$ are observation functions defined below. $\blacksquare$

Hence, a special state $s_\perp$ can be used to represent a state in which all variables are $\perp$ ($s_\perp = \{v = \perp \mid v \in V\}$).

Definition 4 (Observation Function)**.** An observation function for agent i, $O_i : \mathcal{S} \to \mathcal{S}$, is a function that takes a state and returns a subset of that state, representing the part of the state visible to agent i. The following properties must hold for an observation function O_i for all $i \in Agt$ and $s \in S$:

1. $O_i(s) \subseteq s$, (Contraction)
2. $O_i(s) = O_i(O_i(s))$, (Idempotence) $\blacksquare$
3. If $s \subseteq s'$, then $O_i(s) \subseteq O_i(s')$, (Monotonicity)

The idea is to reason about the agents' epistemic formulae based on what they can observe from their own local states. This is accomplished by defining an observation function for

each agent i. Although the JP model reasons with sequences, its original authors did not specify the observation function to handle sequences. In this paper, we denote $O_i(\vec{s}) = [O_i(\vec{s}[0]), \ldots, O_i(\vec{s}[n])]$ (*i.e.* $O_i(\vec{s}[t]) \equiv O_i(\vec{s})[t]$).

Then, the **Retrieval function** is introduced to retrieve the most recent value of the variable v.

Definition 5 (Retrieval Function)**.** Given a sequence of states $\vec{s}$, a timestamp m, and a variable v, the retrieval function, $R : \vec{\mathcal{S}} \times \mathbb{Z} \times V \to \mathbb{D}$, is defined as:

$$R(\vec{s}, m, v) = \begin{cases} \vec{s}[\max(\text{LT})](v) & \text{if LT} \neq \{\} \\ \vec{s}[\min(\text{RT})](v) & \text{else if RT} \neq \{\} \\ \bot & \text{otherwise} \end{cases}$$

where $\text{LT} = \{j \mid v \in \vec{s}[j] \ \wedge \ \vec{s}[j](v) \neq \bot \ \wedge j \leq m\}$ and $\text{RT} = \{j \mid v \in \vec{s}[j] \ \wedge \ \vec{s}[j](v) \neq \bot \ \wedge 0 \leq m < j \leq |\vec{s}|\}$. ■

If the variable v appears in any previous state (or the current state) within the given justified perspective, we assume that v has remained unchanged since its most recent timestamp, $\max(\text{LT})$, which is the latest time v was observed in the given perspective (at or before the given timestamp m). If v has never been observed in the given justified perspective before, we assume that v has remained unchanged since the earliest timestamp $\min(\text{RT})$, which is the first time v's value appeared in the given perspective after the given timestamp m. Otherwise, v is considered $\bot$, as the given perspective does not contain v at all.

Then the **justified perspective function** can construct the agents' perspectives under the input state sequence.

Definition 6 (Justified Perspective Function)**.** Given the input state sequence $\vec{s}$ as $[s_0, \ldots, s_n]$, a *Justified Perspective* (JP) function for agent i, $f_i : \vec{\mathcal{S}}_c \to \vec{\mathcal{S}}_c$, is defined as follows:

$$f_i([s_0, \ldots, s_n]) = [s'_0, \ldots, s'_n]$$

where for $t \in [0, n]$ and $v \in V$:

$$lt_v = \max(\{j \mid v \in O_i(s_j) \wedge j \leq t\} \cup \{-1\}), \quad (1)$$
$$e_v = R([s_0, \ldots, s_t], lt_v, v), \quad (2)$$
$$s''_t = \{v{=}e_v \mid s_t(v) = e_v \vee v \notin O_i(s_t\langle\{v{=}e_v\}\rangle)\}, \quad (3)$$
$$s'_t = s_\bot\langle s''_t\rangle. \quad (4)$$

■

Function f_i takes a sequence of states (either the global state sequence or a justified perspective from another agent) and returns the sequence of local states that agent i believes. Line (4) overrides state $s_\bot$ with state s''_t, which ensures that the output is a complete state sequence, where any missing variables are filled with a placeholder value $\bot$. lt_v represents the last timestamp at which agent i observed the variable v, where "-1" is added to ensure it returns an integer. The value of v that agent i observed (or should have observed) at the relevant timestamp is retrieved in Line (2). Subsequently, Line (3) constructs a justified state at timestamp t, while the condition $v \notin O_i(s_t\langle\{v{=}e_v\}\rangle)$ ensures that agent i's "memory" remains consistent with its current observations at timestamp t.

Then, a ternary semantics is proposed for JP, which employs three truth values: 0 (false), 1 (true), and $\frac{1}{2}$ (unknown).

Definition 7 (Ternary semantics)**.** For any state sequence $\vec{s}$ and any epistemic formula in Language $\varphi \in L_{KB}(\Sigma)$, the ternary semantics function $T(\vec{s}, \varphi)$ is defined as follows (with the model M suppressed for readability):

(a) $T[\vec{s}, r(V_r)] =$ 1 if $\pi(\vec{s}[n], r(V_r)) = \mathit{true}$;
0 else if $\pi(\vec{s}[n], r(V_r)) = \mathit{false}$;
$\frac{1}{2}$ otherwise

(b) $T[\vec{s}, \varphi \wedge \psi] = \min(T[\vec{s}, \varphi], T[\vec{s}, \psi])$

(c) $T[\vec{s}, \neg\varphi] = 1 - T[\vec{s}, \varphi]$

(d) $T[\vec{s}, S_i v] =$ $\frac{1}{2}$ if $v \notin \vec{s}[n]$ or $i \notin \vec{s}[n]$
0 else if $v \notin O_i(\vec{s}[n])$
1 otherwise

(e) $T[\vec{s}, S_i\varphi] =$ $\frac{1}{2}$ if $T[\vec{s}, \varphi] = \frac{1}{2}$ or $i \notin \vec{s}[n]$;
0 else if $T[O_i(\vec{s}), \varphi] = \frac{1}{2}$;
1 otherwise

(f) $T[\vec{s}, K_i\varphi] = T[\vec{s}, \varphi \wedge S_i\varphi]$

(g) $T[\vec{s}, B_i\varphi] = T[f_i(s_\bot\langle\vec{s}\rangle), \varphi]$

where n is the last (current) timestamp of $\vec{s}$. ■

That is, the semantics of the belief are handled by reasoning with agents' generated justified perspectives. This ternary function T returns 1 or 0 when the formula is evaluated as *true* or *false*, respectively. When there is not enough valid value in the given local state $\vec{s}[n]$ to evaluate a predicate $r(V_r)$ ($\exists v \in V_r, \vec{s}[n](v) = \bot$), or the relevant variables (the nesting agent's identifier i or the targeting variable v) are not in the given local state ($i \notin \vec{s}[n]$ or $v \notin \vec{s}[n]$), the ternary function returns $\frac{1}{2}$, indicating that the truth value of the formula cannot be determined.

In addition, Hu, Miller, and Lipovetzky (2023) proved that their ternary semantics are in **the KD45** axiomatic system and that its evaluation is in polynomial time (with the sacrifice of completeness for the non-logically separable formulae, e.g., $p \wedge \neg p$). At last, they provided experiments to demonstrate its efficiency on EP benchmarks with comparisons against the current state-of-the-art planner, the PDKB planner (Muise et al. 2022).

3 Predictive Justified Perspective Model

As mentioned in Section 1, the "static" assumption in the existing EP works (including the JP model) is not practical in many applications. For example, the "static" variable cannot capture the position change of a falling ball, as its position change is not an effect of an agent's action but due to the environmental effect (gravity). To differentiate from the concept of the "static" variable in AI planning, the variables in our model are named as *processual*[1] variable which are state variables whose values can evolve as a result of external environmental processes, independently of the agent's actions. To describe a set of processual variables, the processual variable model is denoted as Ω and defined below.

Definition 8 (Processual Variable Model)**.** Given $\mathcal{T}$ is a set that includes all processual variable types, Ω is defined as:

$$\Omega = (V, \mathcal{T}, \mathit{type}, \eta),$$

where: $\mathit{type} : V \to \mathcal{T}$, $\eta : V \to \mathbb{R}^*, * \in \mathbb{N}$. ■

[1]The idea of the process is from PDDL+ (Fox and Long 2006).

To indicate the changing rules, for each processual variable $v \in V$, the type and coefficients (or parameters) are defined as $type(v)$ and $\eta(v)$. A "static" variable is modeled by the new framework as a processual variable with a special type *static*. It is considered as a base case ($*{=}0$) in which $\eta(v) = \{\}$. Unlike classical planning variables, which are modified exclusively by the agent's action effects, processual variables are updated according to not only the agent's action effects but also exogenous transition rules or continuous dynamics reflecting changes in the environment. In the remainder of this paper, the term "variable" refers to a processual variable unless stated otherwise.

Thus, with the introduction of processual variables, the PJP model can now be defined, following the same signature and language as the JP model introduced in Section 2.

Definition 9 (Model)**.** The PJP model M is defined as:

$$M = (Agt, \Omega, \mathbb{D}, \pi, O_1, \ldots, O_{|Agt|}, PR),$$

where PR is a set of predictive retrieval functions $pr_{type(v)}$ (Definition 10); Ω is the model defined in Definition 8; and the rest are adopted from the JP model (Definition 3). ■

Incorporating processual variables, the PJP model generates predictive justified perspectives through the *Predictive Justified Perspective Function* (Definition 11) relying on the *Predictive Retrieval Function*. An abstract definition of the predictive retrieval function, $pr_{type(v)} \in PR$, where $type(v) \in \mathcal{T}$ is provided in Definition 10. This provides flexibility and expressiveness for the modeler to model any variable with a known type.

Definition 10 (Predictive Retrieval Function)**.** A predictive retrieval function $pr_{type(v)} : \vec{S} \times \mathbb{N} \times V \to \mathbb{D}$ takes the input of a state sequence $\vec{s}$, a timestamp t and a variable v, and outputs the predicted value of v at t based on $\vec{s}$, where $type(v) \in \mathcal{T}$ is the processual variable v's type. The predictive retrieval function $pr_{type(v)}$ should satisfy the following properties. Given the input state sequence $\vec{s} = [s_0, \ldots, s_n]$, its prediction $\vec{p}$ is defined as $\vec{p} = [p_0, \ldots, p_n]$, where for $t' \in [0, n]$, $p_{t'} = \{v{=}pr_{type(v)}(\vec{s}, t', v) | v \in V\}$.

- **Preserving Consistency** [*Compulsory*]:

$$\forall v{\in}V, \forall t{<}|\vec{s}|, v{\in}\vec{s}[t] \Rightarrow \vec{p}[t](v) = \vec{s}[t](v)$$

- **Recursive Consistency** [*Compulsory*]: Let $\vec{W}$ be a subset of the sequence space $\vec{\mathcal{S}}$, such that for all $\vec{w} \in \vec{W}$: $|\vec{w}|{=}|\vec{s}|$ and $\forall t{<}|\vec{s}| \Rightarrow \vec{s}[t] \subseteq \vec{w}[t] \subseteq \vec{p}[t]$.

$$\forall v{\in}V, \forall t{<}|\vec{s}|, \forall \vec{w}{\in}\vec{W} \Rightarrow pr_{type(v)}(\vec{w}, t, v) = \vec{p}[t](v)$$

- **Reconstructive Consistency** [*Optional*]:

$$\forall i \in Agt \Rightarrow [\exists \vec{w} \in \vec{\mathcal{S}}, O_i(\vec{w}){=}\vec{s} \Rightarrow \vec{s} = O_i(\vec{p})] \quad ■$$

The prediction function $pr_{type(v)}$ estimates the value of variable v at timestamp t by deriving v's changing pattern based on the given state sequence $\vec{s}$. A valid predictive retrieval function must be both preserving consistent and recursive consistent to ensure that its semantics satisfy the **KD45** axioms (as proved later in this paper). **Preserving Consistency** ensures that the predicted value of the variable is consistent with its input. This gives $\forall t < |\vec{s}| \Rightarrow \vec{s}[t] \subseteq \vec{p}[t]$, where $\vec{p}$ is constructed in the above definition. **Recursive Consistency** ensures that the predictive retrieval function yields stable results when the input state sequence is the original input state sequence ($\vec{s}$) with extra values from its prediction ($\vec{p}$). This consistency condition requires that for all sequences in the set of sequences $\vec{W}$ (which is the original input sequence with extra assignments from its prediction), applying the predictive retrieval function to $\vec{w}$ yields the same predicted value as in $\vec{p}$. A simple example is for a sequence that contains only one variable v' with values of $[1, \perp, 3, \perp]$. By applying $pr_{type(v')}([1, \perp, 3, \perp], t', v')$ for all timestamps $t' < 3$, we have predicted values of v as $[1, 2, 3, 4]$. Then, for an input sequence that is the same as the original sequence but includes extra values from its prediction, such as $[1, 2, 3, \perp]$, the predicted value $pr_{type(v')}([1, 2, 3, \perp], t', v')$ should remain the same for all timestamps $t' < 3$ ($[1, 2, 3, 4]$). The optional **Reconstructive Consistency** ensures that for all agents, the predictive function is consistent with their own observation function. That is, when using the agent's observation as the input, the predicted values should not change the agent's observation. This property ensures that agents' beliefs are justifiable ($B_i K_i \varphi \Rightarrow K_i \varphi$) following the JP model.

Then, following Definition 10, we provide the definition of the base case of the predictive retrieval function, pr_{static}. The function pr_{static} is domain-independent due to the "static" assumption of classical planning. While for other $type(v) \in \mathcal{T}$, $pr_{type(v)}$ is domain-dependent, and they need to be defined by the modeler.

Definition 10.1 (Static Predictive Retrieval Function)**.** Given a sequence of states $\vec{s}$, a timestamp t, and a variable v, the retrieval function, the static predictive retrieval function $pr_{static} \in PR$, is defined as:

$pr_{static}(\vec{s}, t, v) = R(\vec{s}, t, v)$, where R in Definition 5 ■

The behavior of pr_{static} is semantically equivalent to the retrieval function R defined in the JP model (following the static assumption). Hence, the result of $pr_{static}(\vec{s}, t, v)$ is the same as $R(\vec{s}, t, v)$, ensuring compatibility between the predictive reasoning in the PJP model and the belief reasoning in the JP model for static variables. According to the theorem ($f_i(s) = f_i(f_i(s))$) and its proof of the JP model (Hu, Miller, and Lipovetzky 2023), pr_{static} meets all properties in Definition 10.

Then, it is up to the modeler to design PR functions for other processual variable types. With all PR functions that follow Definition 10, we can now provide our definition to generate justified perspectives with prediction.

Definition 11 (Predictive Justified Perspective Function)**.** Given the input state sequence $[s_0, \ldots, s_n]$, a *Predictive Justified Perspective* (PJP) function for agent i, $f_i : \vec{\mathcal{S}} \to \vec{\mathcal{S}}_c$, is defined as follows:

$$f_i([s_0, \ldots, s_n]) = [s'_0, \ldots, s'_n]$$

where for $t \in [0, n]$ and $v \in V$:

$$e_v = pr_{type(v)}(O_i([s_0, \ldots, s_n]), t, v), \quad (1)$$
$$s''_t = \{v{=}e_v \mid v \in O_i(s_t) \vee v \notin O_i(s_t\langle\{v{=}e_v\}\rangle)\}, \quad (2)$$
$$s'_t = s_\perp\langle s''_t\rangle. \quad (3)$$

■

The PJP Function f_i generates a local predictive justified perspective for agent i. A value e_v of the variable v is retrieved by its corresponding predictive retrieval $pr_{type(v)}$ in Line (1). If the type is *static*, the value of v is the same as that returned by the original retrieval function in the JP model (Definition 5). $v \in O_i(s_t)$ in Line (2) indicates that agent i sees v at t, and thus, the observed v and its value e_v will be part of the output perspective. When the agent i cannot see v at t, assigning e_v to v at t (represented as $s_t\langle\{v{=}e_v\}\rangle$) will not enable agent i to see v (represented as $v{\in}s_t\langle\{v{=}e_v\}\rangle$); the predicted value e_v will be assigned to v. This means that when the agent i cannot see v at t, the predicted value e_v does not create a conflict with the fact that agent i cannot see v at t. Line (3) ensures that the generated perspectives are complete-state sequences.

We can now return to Figure 1a. The agents' positions are processual variables, and line-of-sight determines their observations. When the thief is hidden, the PJP function reconstructs, within the thief's perspective, the police's predictive justified perspective by applying a predictive retrieval function to the police's observed position history. The thief can then reason about this nested belief and move elsewhere, making the police's predicted belief false. The following subsection gives a more comprehensive instantiation of this mechanism using *Grapevine* (Muise et al. 2015), a well-known epistemic-planning benchmark.

3.1 An Example Using First-Order Polynomials

We extend *Grapevine* by modeling agents' secrets as first-order-polynomial processual variables, illustrating how predictive retrieval supports belief and nested-belief evaluation.

Example 1 (Grapevine)**.** There are 3 agents, a, b, and c, each with their own secret: sa, sb, and sc. There are two rooms, rm_1 and rm_2. The actions include **move**, **share**, **lie** and **stop**[2]. In the origin domain, each agent's secret is a boolean value; however, here, we set them as first-order polynomials (processual variables). The shared value of the secret, using agent a's secret sa as an example, is represented as ssa, which equals tsa when a is sharing and lsa when a is lying. When an agent is sharing or lying, all the agents in the same room will hear the shared value of the secret.

Initially, all agents are in rm_1, $tsa = t + 3$, and $lsa = x + 1$. The following action sequence is performed:
[**share**(a), **stop**, **share**(a), **stop**, **move**(c, rm_2), **lie**(a), **stop**]

Here, we provide our definition[3] of pr_{1st_poly} for Example 1. Since the shared value of the secret (ssa) could be deceptive (by lying), the values in an agent's observation could come from different polynomials (tsa or lsa). Therefore, we use segmented prediction in this example. That is, agents use the closest two observations to interpolate the unobserved value. Noted that the number of observations needed depends on the type of processual variable, e.g., one observation for $linear$ and two observations for *1st_poly* (even if the *1st_poly* variable is actually linear, the agent would not know that beforehand).

Timestamp	0	1	2	3	4	5	6	7
Action		**share**(a)	**stop**	**share**(a)	**stop**	**move**(c,rm_2)	**lie**(a)	**stop**
tsa	3	4	5	6	7	8	9	10
lsa	1	2	3	4	5	6	7	8
ssa	-	4	-	6	-	-	7	-
$O_b(\vec{s})(ssa)$	-	4	-	6	-	-	7	-
$O_c(\vec{s})(ssa)$	-	4	-	6	-	-	-	-
$f_b(\vec{s})(ssa)$	3	4	5	6	6.33	6.67	7	7.33
$f_c(\vec{s})(ssa)$	3	4	5	6	7	8	9	10
$f_b(f_a(\vec{s}))(ssa)$	3	4	5	6	6.33	6.67	7	7.33
$f_b(f_c(\vec{s}))(ssa)$	3	4	5	6	7	8	9	10
$f_c(f_a(\vec{s}))(ssa)$	3	4	5	6	7	8	9	10
$f_c(f_b(\vec{s}))(ssa)$	3	4	5	6	7	8	9	10

Table 1: Observation and reasoning of ssa in Example 1.

Definition 10.2 (First-order Polynomial Predictive Retrieval Function)**.** Given AT= $\{j \mid v \in \vec{s}[j] \wedge \vec{s}[j](v) \neq \perp\}$, LT $= \{j \mid j \in \text{AT} \wedge j \leq t\}$, and RT= $\{j \mid j \in \text{AT} \wedge j > t\}$, the predictive retrieval function for the first-order polynomial, $pr_{1st\_poly}(\vec{s}, t, v)$, can be defined as:

$$pr_{1st\_poly}(\vec{s}, t, v) = \begin{cases} \perp & \text{if } |\text{AT}| = 0 \\ \vec{s}[\max(\text{AT})](v) & \text{if } |\text{AT}| = 1 \\ \frac{t-t_1}{t_2-t_1}\left(\vec{s}[t_2](v) - \vec{s}[t_1](v)\right) + \vec{s}[t_1](v) & \text{if } |\text{AT}| \geq 2 \end{cases}$$

where

$$\begin{cases} \text{if RT} = \emptyset, & t_2 = \max(\text{LT}),\ t_1 = \max(\text{LT} \setminus \{t_2\}) \\ \text{if LT} = \emptyset, & t_1 = \min(\text{RT}),\ t_2 = \min(\text{RT} \setminus \{t_1\}) \\ \text{otherwise}, & t_1 = \max(\text{LT}),\ t_2 = \min(\text{RT}) \end{cases}$$ ■

The proposed function above identifies the most recent two timestamps in relation to the given timestamp t that agent i observes v, denoted as t_1 and t_2 ($t_1 \neq t_2$, $\{t_j \mid j \in \text{AT}\}$), such that: $t_1 < t_2 \leq t$, if timestamp t is at or after the agent's latest observation of v; $t < t_1 < t_2$, if timestamp t is before the agent's first observation of v; $t_1 \leq t < t_2$, otherwise. For example, if the last observation is before or at the timestamp t, t_2 will be the timestamp of the last observation and t_1 will be the timestamp of the second-last observation where $t_1 = \max(\text{LT} \setminus \{t_2\})$.

Given pr_{1st_poly} in Definition 10.2, the values of ssa in each agent's observation (what they saw) and predictive justified perspective (what they believe), which are generated by iteratively applying observation functions and PJP functions, are shown in Table 1. Before timestamp 5, all agents are in rm_1, which results in them having the same (nested) observations and (nested) beliefs. However, after action **move**(c, rm_2), c is no longer in rm_1, and no longer "sees" ssa. This leads to different beliefs between agent b ($f_b(\vec{s})$) and agent c ($f_c(\vec{s})$) about the value of ssa, and agent b is able to reason about agent c's beliefs ($f_b(f_c(\vec{s}))$). For example, when generating s''_4 (timestamp 4 in Definition 11) in $f_b(\vec{s})$, $s''_4(ssa) = pr_{1st\_poly}(O_b(\vec{s}), 4, ssa) = 6.33$. This is calculated (Definition 10.2) by identifying t_1 and t_2 (3 and 6), and using ssa's values (6 and 7) to get its

[2] As the effects of **share** and **lie** are transient, we enforce a **stop** action to delimit their duration.

[3] Note: this is just one valid (reasonable) definition of the predictive retrieval function for the first-order polynomial.

value for timestamp 4. But for s_4'' in $f_c(\vec{s})$, $s_4''(ssa) = pr_{1st\_poly}(O_c(\vec{s}), 4, ssa)$ would be 7 by identifying t_1 and t_2 as 1 and 3, and using ssa's values (4 and 6) to predict its value at timestamp 4. This is because of the difference between $O_b(\vec{s})$ and $O_c(\vec{s})$. More challengingly, agent b's belief about agent c's belief is generated by $O_c(f_b(\vec{s}))$ (the same as $O_c(\vec{s})$ in this example), resulting in agent b holding these beliefs (for s_4): $B_b ssa = 6.33$; and $B_b B_c ssa = 7$.

3.2 Semantics and Axiomatic Validity

By constructing predictive justified perspectives, the evaluation of epistemic formulae in $L_{K!B}$ follows the ternary semantics defined in Definition 7, except that the JP function is replaced by the PJP function (Definition 11). Accordingly, as in the JP model (Hu, Miller, and Lipovetzky 2023), the expressive scope and limitations of ternary evaluation remain unchanged in the present setting.

The complexity of ternary semantics is polynomial if O and PR are polynomial. Specifically, given a ternary function instance $T[M, \vec{s}, \varphi]$, the worst-case scenario is that φ is a belief formula with a depth of d. Then, according to its definition, the complexity class is $\Theta(|V| \cdot d \cdot |\vec{s}|) \cdot \Theta(o) \cdot \Theta(pr)$, where o and pr are the most complex observation function ($o \in \{O_1, \ldots, O_{|Agt|}\}$) and predictive retrieval function ($pr \in PR$), respectively. That is, the depth of the epistemic formula, which usually causes exponential blowup in DEL or Knowledge-Base approaches, would not cause exponential blowup in the PJP model (similar to the JP model).

To examine the soundness of the PJP model, we show that our ternary semantics follows the **KD45** axiomatic system for all epistemic logics related to belief. At the end of this section, we also discuss a KB axiom that is affected by the property of the predictive retrieval function.

Firstly, we show that agents' predictive justified perspectives are consistent with their own observations.

Theorem 1. Given a state sequence $\vec{s}$, let i be an agent from Agt, for any timestamp t, we have:

$$O_i(\vec{s}[t]) \subseteq f_i(\vec{s})[t]$$

Proof. For any v in V such that $v \in O_i(\vec{s}[t])$, according to Line (1) in Definition 11 ($e = pr_{type(v)}(O_i(\vec{s}), t, v)$) and Preserving Consistency in Definition 10, the retrieved predictive value of v equals $O_i(\vec{s}[t])(v)$. Then, following $v \in O_i(\vec{s}[t])$, the generated s_t'' contains the assignment $v = O_i(\vec{s}[t])(v)$. In addition, s_t'' overriding the none state $s_\perp$ has no effect on existing assignments in s_t'', which means assignment $v = O_i(\vec{s}[t])(v)$ is in $f_i(\vec{s})[t]$. Hence, $O_i(\vec{s}[t])(v) = f_i(\vec{s})[t](v)$, which concludes $\forall v \in V,\ v \in O_i(\vec{s}[t]) \Rightarrow O_i(\vec{s}[t])(v) = f_i(\vec{s})[t](v)$. Therefore, Theorem 1 holds. □

Intuitively, the agent's predictive justified perspective $f_i(\vec{s})$ of the given state sequence $\vec{s}$ should be the same as their prediction $f_i(f_i(\vec{s}))$ of what they predicted ($f_i(\vec{s})$). Thus, we propose the following theorem.

Theorem 2 (Idempotence of Predictive Retrieval Function)**.** A predictive justified perspective function f_i is idempotent:

$$f_i(f_i(\vec{s})) = f_i(\vec{s})$$

Proof. Let the input sequence be $\vec{s} = [s_0, \ldots, s_n]$. First, we can construct a prediction $\vec{p} = [p_0, \ldots, p_n]$ based on agent i's observation $O_i(\vec{s}) = [O_i(\vec{s}[0]), \ldots, O_i(\vec{s}[n])]$ of the given input state sequence $\vec{s}$, following the same process in Definition 10, where for $t' \in [0, n]$, $p_t = \{v' = e' \mid e' = pr_{type(v')}(O_i(\vec{s}), t', v')\}$.

According to the Preserving Consistency (Definition 10) of all predictive retrieval functions, we have for all $t < |\vec{s}|$, $O_i(\vec{s}[t]) \subseteq \vec{p}[t]$. Following Line (1) and Line (2) in Definition 11 during the construction of $f_i(\vec{s})$, for all $t < |\vec{s}|$, since $s_t'' = \{v = e \mid v \in O_i(s_t) \vee v \notin O_i(s_t\langle\{v = e\}\rangle)\}$ ($e = pr_{type(v)}(O_i([s_0, \ldots, s_n]), t, v)$), we have $s_t'' \subseteq \vec{p}[t]$. Then, Line (3) of the same definition fills the missing variables (of s_t'') with none value assignments. It is noted that those added missing variables by Line (3) are not in agent i's observation, which means $O_i(s_t') = O_i(s_t'')$. According to the Contraction property of the observation function (in Definition 4), we have $O_i(s_t') \subseteq s_t'' \subseteq \vec{p}[t]$. From Theorem 1, we have $O_i(s_t) \subseteq s_t'$. According to the Idempotence and Monotonicity of the observation function, $O_i(s_t) \subseteq O_i(s_t')$.

Since $f_i(\vec{s})$ is formed by $[s_0', \ldots, s_n']$, the input for each variable's predication is $O_i(f_i(\vec{s}))$ when generating $f_i(f_i(\vec{s}))$. In addition, $O_i(f_i(\vec{s}))$ is effectively $[O_i(s_0'), \ldots, O_i(s_n')]$. According to the paragraph above, for all timestamps $t' < |\vec{s}|$, $O_i(s_t) \subseteq O_i(s_t') \subseteq \vec{p}[t]$. Due to the Recursive Consistency of all predictive retrieval functions, we have $\forall t < |\vec{s}|, \forall v \in V \Rightarrow pr_{type(v)}(O_i(f_i(\vec{s})), t, v) = pr_{type(v)}(\vec{p}, t, v) = pr_{type(v)}(O_i(\vec{s}), t, v)$. Comparing the construction of $f_i(\vec{s})$ and $f_i(f_i(\vec{s}))$ from Definition 11, since Line (1) retrieves the same value, Lines (2) and (3) will have the same effects. Therefore, Theorem 2 holds. □

With Theorem 2, we can now show that the ternary semantics T (Definition 7) in the PJP model satisfies the axioms of the **KD45** system.

Theorem 3. The ternary semantics for the belief operator B follows the axiomatic system **KD45** for any agent $i \in Agt$:

K (Distribution): $B_i\varphi \wedge B_i(\varphi \Rightarrow \psi) \Rightarrow B_i\psi$

D (Consistency): $B_i\varphi \Rightarrow \neg B_i \neg \varphi$

4 (Positive Introspection): $B_i\varphi \Rightarrow B_i B_i \varphi$

5 (Negative Introspection): $\neg B_i\varphi \Rightarrow B_i \neg B_i \varphi$

Proof. For Axiom **K**, $T[\vec{s}, B_i\varphi \wedge B_i(\varphi \Rightarrow \psi)] = 1$ is equivalent to $\min(T[\vec{s}, B_i\varphi], T[\vec{s}, B_i(\varphi \Rightarrow \psi)]) = 1$, which indicates both $T[f_i(\vec{s}), \varphi]$ and $T[f_i(\vec{s}), \varphi \Rightarrow \psi]$ are 1. Therefore, we have $T[f_i(\vec{s}), \psi] = 1$, which makes $T[\vec{s}, B_i\psi] = 1$.

For Axiom **D**, $T[\vec{s}, B_i\varphi] = 1$ means $T[f_i(\vec{s}), \varphi] = 1$. $T[\vec{s}, \neg B_i \neg \varphi] = 1 - T[\vec{s}, B_i \neg \varphi]$. Since $T[\vec{s}, B_i \neg \varphi] = T[f_i(\vec{s}), \neg \varphi] = 1 - T[f_i(\vec{s}), \varphi]$, we have $T[\vec{s}, B_i \neg \varphi] = 0$, which means $T[\vec{s}, \neg B_i \neg \varphi] = 1$.

For Axiom **4**, the premise $T[\vec{s}, B_i\varphi] = 1$ means $T[f_i(\vec{s}), \varphi] = 1$. By Theorem 2, $f_i(f_i(\vec{s})) = f_i(\vec{s})$ Therefore, $T[f_i(\vec{s}), \varphi] = T[f_i(f_i(\vec{s})), \varphi] = 1$, which means that $T[f_i(\vec{s}), B_i\varphi] = T[\vec{s}, B_i B_i \varphi] = 1$. Thus, Axiom **4** holds.

Axiom **5** can be proved similarly as **D** and **4**. □

In addition to **KD45**, the PJP model straightforwardly satisfies the KB axiom **KB1** ($K_i\varphi \Rightarrow B_i\varphi$) and **KB2** ($B_i\varphi \Rightarrow K_i B_i \varphi$), based on Theorem 1. In many epistemic

logic systems, Axiom **D**, Axiom **5**, and Axiom **KB1** cannot be true at the same time due to the "unwanted" axiom ($B_iK_i\varphi \Rightarrow K_i\varphi$) (Gochet and Gribomont 2006); however, as Hu (2025) claimed, in **justified** belief systems (e.g., the JP model), the "unwanted" axiom is valid because of their strict definition of belief. Similarly, we discuss whether the "unwanted" axiom holds in our PJP model.

Given $M = (Agt, \Omega, \mathbb{D}, \pi, O_1, \ldots, O_{|Agt|}, PR)$, a PJP model instance, where all $pr_{type(\cdot)} \in PR$ satisfy the preserving consistency and recursive consistency, the "unwanted" axiom is not necessarily guaranteed to hold. This is because what agent i sees in i's predictive justified perspective ($f_i(\vec{s})$) could be more than what agent i sees in the given state sequences $\vec{s}$. The original observation of agent i ($O_i(\vec{s})$) can be seen by i from the generated i's predictive justified perspective. That is, $\forall t < |\vec{s}| \Rightarrow O_i(\vec{s}[t]) \subseteq O_i(f_i(\vec{s})[t])$. However, the predicted values of those unobserved variables might allow agent i to see more in i's own perspective. In other words, $\forall t < |\vec{s}| \not\Rightarrow O_i(f_i(\vec{s})[t]) \subseteq O_i(\vec{s}[t])$. This results in what agent i believes that i knows could be inconsistent with what i actually knows.

However, if all $pr_{type(\cdot)} \in PR$ also satisfy the optional reconstructive consistency, the "unwanted" axiom can be guaranteed to hold.

Theorem 4. Given M=$(Agt, \Omega, \mathbb{D}, \pi, O_1, \ldots, O_{|Agt|}, PR)$, a PJP Model instance, where all $pr_{type(\cdot)} \in PR$ satisfy preserving consistency, recursive consistency, and reconstructive consistency, the following theorem holds:

$$\forall \varphi, B_iK_i\varphi \Rightarrow K_i\varphi$$

Proof. The premise of the above implication $T[\vec{s}, B_iK_i\varphi]$= 1 means $T[f_i(\vec{s}), K_i\varphi] = 1$, which is $T[f_i(\vec{s}), \varphi] = 1$ and $T[f_i(\vec{s}), S_i\varphi] = 1$. Meanwhile, the conclusion of the theorem implication $T[\vec{s}, K_i\varphi]$ is effectively the same as $\min(T[\vec{s}, \varphi], T[\vec{s}, S_i\varphi])$ according to our semantics (Definition 7).

For the latter part ($T[\vec{s}, S_i\varphi]$), when constructing $f_i(\vec{s})$, the input sequence for all predictive retrieval calls is $O_i(\vec{s})$ according to Line (1) of Definition 11. Since all $pr_{type(\cdot)} \in PR$ satisfy the reconstructive consistency (in Definition 10), we have $O_i(\vec{s}) = O_i(f_i(\vec{s}))$. Since $T[f_i(\vec{s}), S_i\varphi] = 1$ means $T[O_i(f_i(\vec{s})), \varphi]$ is evaluated as either 0 or 1, we have $T[O_i(\vec{s}), \varphi]$ evaluated as either 0 or 1, which concludes $T[\vec{s}, S_i\varphi] = 1$.

Then, according to $T[O_i(\vec{s}), \varphi] \neq \frac{1}{2}$ (item a in Definition 7), a partial-state sequence $O_i(\vec{s})$ is sufficient to evaluate φ. It means the predicted values of unobserved variables are irrelevant to φ. In addition, since $O_i(\vec{s}) = O_i(f_i(\vec{s}))$, the partial-state sequences that determine the value of φ from $f_i(\vec{s})$ and $\vec{s}$ are the same. Thus, $T[f_i(\vec{s}), \varphi] = 1$ gives $T[\vec{s}, \varphi] = 1$.

Therefore, Theorem 4 holds. □

4 Implementation

The PJP model, like the JP model, is implemented in the planning language PDDL+ (Fox and Long 2006), by incorporating the idea of external functions from F-STRIPS (Geffner 2000). It is worth noting that the epistemic logic reasoning – the PJP model and its semantics – are implemented in the external functions. The detailed explanations of the PDDL encoding, along with examples, are documented in the supplementary material.

5 Experiment

We evaluate the PJP model on the Grapevine domain, one of the most challenging benchmark domains in EP, by adding dynamically changing variables. Our experiments aim to validate three key hypotheses:

- **Expressiveness:** The PJP model can solve problems involving dynamic, unobserved variables that are unsolvable by standard "static" epistemic planners.
- **Computational Feasibility:** The overhead introduced by the predictive retrieval functions (pr) is negligible compared to the search effort, maintaining the tractability of the state-based approach.
- **Modularity:** The framework is solver-agnostic regarding the prediction method, allowing for the integration of both analytical and statistical models (e.g., linear regression) without modifying the planner's core logic.

The grapevine domain used in experiments is the same as in Example 1, except that the agents are allowed to **share** others' secrets to make it more challenging. It is noted that the other agents do not know whether the value they shared about another's secret (e.g., ssa) is true or false. Therefore, when they share others' secrets, they can only share the value they believe in, which can be equal to tsa, lsa, or some other values (e.g., from a false prediction). Secrets here are processual variables (e.g., polynomials) defined in Definition 8. To isolate belief (inferred from past observations) from knowledge (derived from direct, current observation), we append a $\neg sharing$ constraint to all goals. This precludes the agent from achieving the goal while hearing ssa, thereby forcing the planner to rely solely on perspectives (PJPs) derived from history. Experiments were conducted on Ubuntu running under WSL2 on a 13th Gen Intel Core i5-13490F CPU with a 10 GB memory limit and a 600-second timeout.

We utilize the tree-search version of Breadth-First Search (BFS) to isolate the performance contribution of the PJP model from the search algorithms and heuristics. As mentioned earlier, there is no existing planner tool that can reason about agents' nested beliefs with predictions, so we have no other approach to compare against.

5.1 Results and Analysis

Table 2 summarizes the results across nine instances.

Validation of Predictive Reasoning ($G0, G1, G5$): $G0$ serves as a baseline where observation data is insufficient for prediction[4]. The model correctly falls back to the static assumption (pr_{static}), validating the system's backward compatibility. In contrast, $G1$ requires predicting a first-order polynomial ($tsa = 7$). The PJP model successfully solves

[4]There are still 4 predictions due to the search frontier (reached) to a depth of 3.

	$\|gen\|$	$\|seg\|$	τ_p(ms)	τ_t(s)	Function of x	$\eta(tsa)$	$\eta(lsa)$	$pr_{type(x)}$	$\|plan\|$	Goal
G0	81	4	0.07	0.03	$x = a \cdot t + b$	$[1, 3]$	$[1, 1]$	pr_{1st_poly}	2	$tsa = 5 \wedge B_b ssa = 4$
G1	1211	251	0.10	0.53				pr_{1st_poly}	4	$tsa = 7 \wedge B_b ssa = 7$
G2	1211	587	0.25	0.93				pr_{1st_poly}	4	$tsa = 7 \wedge B_c B_a B_b ssa = 7$
G3	1211	955	0.40	1.33				pr_{1st_poly}	4	$tsa = 7 \wedge B_a B_b B_c B_a B_b ssa = 7$
G4	1211	223	0.14	0.60				pr_{linear_reg}	4	$tsa = 7 \wedge B_b ssa = 7$
G5	421	109	0.08	0.17				pr_{1st_poly}	4	$tsa = 7 \wedge B_b ssa = 8$
G6	58708	72020	0.39	65.26				pr_{1st_poly}	7	$B_c ssa = 10 \wedge B_a B_c ssa = \perp$
G7	64140	86432	0.38	72.66				pr_{1st_poly}	7	$B_c ssa = 10 \wedge B_a B_c (ssa \neq 10 \wedge ssa \neq \perp)$
G8	17277	1221	0.16	10.29	$x = a \cdot t^2 + b \cdot t + c$	$[1, 0, 2]$	$[1, 0, 0]$	pr_{2st_poly}	6	$tsa = 38 \wedge B_b ssa = 38$
G9	81	29	0.07	0.03	$x = a^t$	$[3]$	$[1]$	pr_{power}	2	$tsa = 9 \wedge B_b ssa = 9$
G10	17277	889	0.20	10.90	$x = a \cdot sin(b \cdot t + c)$	$[8, 5, 4]$	$[1, 1, 1]$	pr_{sin}	6	$tsa = 4.23 \wedge B_b ssa = 4.23$

Table 2: Experimental results for Grapevine, where: $|gen|$ is the number of generated nodes during the search; $|seg|$ is the number of segmented predictions (which is one per pr_x function called, except for pr_{1st_poly} from Definition 10.2); τ_p is the average prediction time per epistemic formula evaluation; τ_t is the total execution time; the ground truth models for tsa and lsa are defined by parametrizing the "Function of x" with the coefficient set $\eta(tsa)$ and $\eta(lsa)$ respectively; $|plan|$ is the length of the found plan; and "Goal" shows the goal epistemic formulae (omitting $\neg sharing$ for readability). The definitions of the other pr functions can be found in the supplementary material.

this by invoking pr_{1st_poly}, identifying the correct coefficients from history. Moreover, Instance $G5$ demonstrates the model's predictive capability by generating a belief ($ssa = 8$) that exists neither in the observation history nor in the ground truth domain ($max \leq 7$). This confirms that PJP can extend epistemic planning to dynamic environments without requiring explicit transition models in the action schema.

Impact of Prediction Overhead ($G1$, $G4$): To assess the cost of the predictor, we compared an analytical approach ($G1$, using pr_{1st_poly}) against a statistical approach ($G4$, using pr_{linear_reg}). Both instances generate an identical search tree (1211 nodes). However, the regression-based predictor increases the average prediction time (τ_p) from $0.10ms$ to $0.14ms$. Despite this increase, the overhead remains a small fraction of the total planning time, supporting the feasibility of integrating complex, learning-based predictors.

Scalability in Beliefs Depth ($G1 - G3$): $G1$, $G2$, and $G3$ have the same search tree; the only varying factor is the depth of the epistemic goal formula. The resulting increase in epistemic evaluation cost is roughly linear, not exponential, which matches our claim on complexity in Section 3.2.

Nested Beliefs with Misinformation ($G6, G7$): Instances $G6$ and $G7$ introduce higher-order nesting ($B_a B_c ssa$). In these instances, agent c believes in the correct ssa value, while agent a is either unaware of it or has false beliefs about c regarding ssa. Those false-belief instances are considered hard problems. As expected, the search space grows exponentially with these problems (caused by longer plan lengths), resulting in roughly $60,000$ generated nodes. Crucially, the PJP model successfully finds solutions (ssa is shared by b without being noticed by a) involving second order nested dynamic beliefs.

Flexibility of Variable Types ($G8$-$G10$): Finally, $G8, G9$, and $G10$ demonstrate the model's capacity to handle non-linear dynamics, including quadratic (t^2) and sinusoidal functions. The planner successfully identifies plans requiring agents to predict oscillating values ($G10$).The node counts $(17,277)$ reflect the complexity of the plan length required to gather sufficient observations (three) for these complex functions, rather than an inefficiency in the reasoning engine itself.

Robustness to Adversarial Noise (Lies): Beyond standard dynamics, the PJP framework's modularity allows for robust reasoning in adversarial settings. We analyzed a modified scenario (detailed in Supplementary Material) in which agent c injects a false value ("lie") of ssa to deceive b. Standard predictors (pr_{1st_poly} or pr_{linear_reg}) incorporate these outliers, leading b to form incorrect beliefs. However, by simply swapping in a robust prediction function that utilizes a majority-voting scheme, the planner successfully filters out the lie, maintaining the correct belief. This demonstrates that the PJP model can seamlessly integrate outlier-rejection algorithms, such as RANSAC (Fischler and Bolles 1987), to handle noisy or malicious environments.

6 Conclusion & Future Work

We introduced the PJP model to extend epistemic planning to dynamic environments. By replacing the static assumption with a predictive mechanism, PJP enables agents to reason about nested beliefs when variables evolve independently of their actions. We showed that this extension preserves the desirable computational and logical properties of the JP model under appropriate conditions. Experimentally, PJP was effective on dynamic Grapevine instances involving polynomial and sinusoidal changes, which are beyond the scope of prior epistemic planners based on static assumptions. In addition, the framework is flexible enough to accommodate different predictive mechanisms, ranging from simple mathematical functions to machine learning models, without modifying the core planning architecture.

Future work will focus on making this framework more autonomous by using learning methods to automatically detect changing patterns in the environment. Another direction would be to apply PJP in a real-world multi-agent or human-agent interaction setting.

References

Bolander, T.; and Andersen, M. B. 2011. Epistemic planning for single- and multi-agent systems. *Journal of Applied Non-Classical Logics*, 21(1): 9–34.

Chrpa, L.; and Karpas, E. 2025. On generating robust plans and linear execution strategies in planning against nature. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 35, 169–177.

Cooper, M. C.; Herzig, A.; Maffre, F.; Maris, F.; and Régnier, P. 2019. The epistemic gossip problem. *Discrete Mathematics*, 342(3): 654–663.

Fagin, R.; Halpern, J. Y.; Moses, Y.; and Vardi, M. Y. 1995. *Reasoning About Knowledge*. MIT Press. ISBN 9780262562003.

Fischler, M. A.; and Bolles, R. C. 1987. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. In Fischler, M. A.; and Firschein, O., eds., *Readings in Computer Vision*, 726–740. San Francisco (CA): Morgan Kaufmann. ISBN 978-0-08-051581-6.

Fox, M.; and Long, D. 2006. Modelling Mixed Discrete-Continuous Domains for Planning. *J. Artif. Intell. Res.*, 27: 235–297.

Francès, G.; and Geffner, H. 2015. Modeling and Computation in Planning: Better Heuristics from More Expressive Languages. In *International Conference on Automated Planning and Scheduling*.

Geffner, H. 2000. Functional STRIPS: a more flexible language for planning and problem solving. *Logic-based artificial intelligence*, 187–209.

Gochet, P.; and Gribomont, E. P. 2006. Epistemic logic. In Gabbay, D. M.; and Woods, J., eds., *Logic and the Modalities in the Twentieth Century*, volume 7 of *Handbook of the History of Logic*, 99–195. Elsevier.

Goldman, A. I. 1979. What is justified belief? In *Justification and knowledge*, 1–23. Springer.

Hu, G. 2025. *'Seen Is Believing': Modeling and Solving Epistemic Planning Problems using Justified Perspectives*. Ph.D. thesis, The University OF Melbourne.

Hu, G.; Miller, T.; and Lipovetzky, N. 2022. Planning with Perspectives – Decomposing Epistemic Planning using Functional STRIPS. *J. Artif. Int. Res.*, 75.

Hu, G.; Miller, T.; and Lipovetzky, N. 2023. Planning with multi-agent belief using justified perspectives. In *Proceedings of the Thirty-Third International Conference on Automated Planning and Scheduling*, ICAPS '23. AAAI Press. ISBN 1-57735-881-3.

Muise, C.; Belle, V.; Felli, P.; McIlraith, S.; Miller, T.; Pearce, A. R.; and Sonenberg, L. 2022. Efficient multi-agent epistemic planning: Teaching planners about nested belief. *Artificial Intelligence*, 302: 103605.

Muise, C.; Belle, V.; Felli, P.; McIlraith, S. A.; Miller, T.; Pearce, A. R.; and Sonenberg, L. 2015. Planning Over Multi-Agent Epistemic States: A Classical Planning Approach. In *AAAI Conference on Artificial Intelligence*.