

XTAI-2026

Pre-Proceedings



Preface

*Explainability and Transparency in AI (XTAI), AISB Convention 2026, University of Sussex
(1 July 2026)*

The rapid evolution of artificial intelligence has brought extraordinary advances in capability, scale, and societal impact. At the same time, it has intensified longstanding questions about understanding, trust, and accountability in computational systems. The **Explainability and Transparency in AI (XTAI-2026) Symposium**, held as part of the AISB Convention 2026, responds directly to this by providing a focused forum for exploring how and why AI systems should be interpretable, and what it means to render them meaningfully transparent in practice.

This volume of proceedings captures the contributions presented at XTAI-2026 and reflects the growing maturity and interdisciplinarity of the field. The workshop continues a line of inquiry established in earlier editions, extending it to contemporary paradigms such as large language models, agentic AI, and multi-agent systems, as technologies whose complexity and autonomy challenge traditional assumptions about explainability.

A Timely Focus on Explainability and Transparency

As AI systems are increasingly deployed in high-stakes environments, ranging from healthcare and finance to policy and education, the need for transparent, interpretable systems has transitioned from a theoretical concern to a practical imperative. Decisions made or supported by AI can have profound consequences, and opaque models can undermine trust, hinder adoption, and raise ethical and regulatory concerns. XTAI-2026 is situated within this broader context, acknowledging that explainability is not merely a technical add-on, but a foundational requirement for responsible AI deployment.

The contributions in this volume collectively demonstrate that explainability must be addressed at multiple levels: from underlying model architectures and learning algorithms to user-facing explanations and governance frameworks.

Bridging Theory and Practice

Amongst the missions of XTAI-2026 are the commitment to bridging theoretical insight with practical relevance and the provision of a platform to connect established researchers and early-career scholars, fostering a collaborative environment in which novel ideas can be debated alongside mature approaches.

The work presented at XTAI-2026 makes clear that explainability and transparency are not static goals but evolving research frontiers. This proceedings volume reflects a field that is

moving toward greater integration: combining insights from machine learning, knowledge representation, human-computer interaction, ethics, and domain sciences. It also points to the need for continued collaboration between academia, industry, and policymakers to ensure that advances in AI are matched by advances in understanding and accountability.

Acknowledgements

The success of XTAI-2026 is due to the contributions of authors, reviewers, and the programme committee, as well as the support of AISB.

We hope that the papers collected here will inform, inspire, and challenge readers and help advance the ongoing effort to make AI systems not only more powerful, but also more understandable, trustworthy, and aligned with human values.

The XTAI 2026 organisers.

June 2026

Table of Contents

Embedding Normative-Temporal Constraints in Planning. <i>Gleb Sakharov, Adam Wyner</i>	1
Beyond Hallucination: Temporal Knowledge Asymmetry as a Distinct Failure Mode in Large Language Models for Non-Western Knowledge Domains. <i>Parth Anand</i>	8
Beyond Scaling: A Stage 3 Geometric Framework for LLM Transparency through Language Manifold Dynamics. <i>Lijia Zhang</i>	13
Explainable Reasoning and Learning in Ad Hoc Teamwork. <i>Hasra Dodampegama, Mohan Sridharan</i>	26
Trustworthy Multi-Agent Systems: Mitigating Semantic Drift with the Argent Signaling Protocol. <i>Anantha Sharma</i>	38

Embedding Ethical-Temporal Constraints in Planning

Gleb Sakharov¹ and Adam Wyner²

Abstract. Agentic autonomous decision-making systems are increasingly deployed in environments where their behaviour must plan and satisfy not only task objectives but also normative or ethical constraints that evolve over time, e.g., whether *obligations* or *safety conditions* are satisfied. Classical planners typically reason only about goal achievement and lack native mechanisms for representing and complying with temporal norms during plan generation. This paper builds on the work of [7] on incorporating temporal functionality into classical planning architectures, and extends it further by outlining a prototype design for integrating temporal normative constraints to support ethical planning. We employ an existing Haskell library that provides a domain-specific language for interfacing with a well-established external classical planner implemented in C++. Our proposed implementation compiles temporal specifications expressed in Linear Temporal Logic over finite traces (LTLf) into finite-state automata and composes these automata with the planner’s transition system. Rather than evaluating norms after a complete plan has been produced, the automaton will incrementally monitor snapshots of the world state after candidate actions are applied. The automaton state represents the progress of the normative specification; if a transition leads to a violation state, the corresponding action is rejected as inadmissible. This will allow temporal constraints to be enforced during action generation even when the planning DSL does not have direct access to the search tree maintained by the external planner. The interface between Haskell and C++ marries some of the advantages of a declarative programming paradigm with the demonstrated strengths of a planner in an imperative framework. The resulting design demonstrates how temporal normative constraints can be integrated into existing planning infrastructures without modifying the underlying planner. By compiling norms into automata and embedding them into the action-generation layer, the approach will provide a lightweight form of normative enforcement compatible with existing planning toolchains.

1 Motivation

Autonomous systems increasingly operate in domains where their behavior must satisfy not only functional requirements but also normative or ethical constraints such as *obligations* or *safety*.³ In many such settings, normative, ethical evaluation cannot be reduced to properties of individual actions or static world states. Instead, the normative acceptability of a system’s behavior often depends on how actions and consequences unfold over time. Obligations may concern what must eventually occur, what must never occur, or what must

occur only after certain conditions are satisfied. As a result, ethical reasoning in autonomous planning frequently requires reasoning over sequences of states and actions rather than isolated decisions.

Consider three common types of ethical constraints. First, many obligations concern future fulfillment, such as honoring commitments or ensuring that harm is eventually mitigated. Second, ethical requirements often involve persistence conditions, such as maintaining safety, privacy, or fairness throughout execution. Third, moral permissibility can depend on temporal ordering, for example when defensive actions are justified only after a threat occurs, or when corrective actions must follow a violation. These forms of reasoning involve temporal structure—eventuality, invariance, and ordering, which cannot be naturally captured by state-based representations alone.

Temporal logics provide a natural formalism for representing such constraints[8].⁴ In particular, Linear Temporal Logic on finite traces (LTLf) allows ethical requirements to be expressed as properties over the finite execution traces produced by planning systems. Operators such as *eventually*, *always*, and *until* make it possible to formally encode ethical constraints including safety requirements (undesirable states must never occur), liveness requirements (certain obligations must eventually be fulfilled - [1] p.120), and conditional norms (certain actions must occur only after specific conditions are met). In this way, ethical considerations can be treated as temporal properties that admissible plans must satisfy.

An additional advantage of LTLf is that its specifications can be compiled into finite automata[9], enabling direct integration with automated planning algorithms. By translating ethical constraints into automata that recognize permissible execution traces, the planner can ensure that generated plans satisfy the desired ethical properties throughout their execution. This approach allows ethical requirements to be enforced systematically during planning rather than evaluated only after a plan has been generated.

For these reasons, LTLf provides a principled bridge between ethical requirements and computational planning mechanisms. By expressing ethical constraints as temporal properties and compiling them into finite automata, ethical considerations can be incorporated directly into the planning process while remaining compatible with standard planning techniques.

The main contribution of this paper is to add ethical constraints to an existing planner in C++ with Haskell interface using a Domain Specific Language (DSL). This augmentation offers several advantages. By embedding problem specification in Haskell, the DSL ensures clarity, modularity, and type safety, reducing errors that might occur when manually writing Planning Domain Definition Language (PDDL). The functional nature of Haskell also makes it straightforward to compose and modify planning problems programmatically. The current DSL does not natively support ethical constraints, but its structured representation provides a natural extension point for

¹ Independent Researcher, Cardiff, Wales, email: sakharov.gleb@gmail.com

² Swansea University, Department of Computer Science, Swansea, Wales, email: a.z.wyner@swansea.ac.uk

³ While distinctions between *norms* and subsorts of *ethics* can be defined, we use the terms largely interchangeably, unless there are specific examples under discussion.

⁴ For properties and definitions of LTLf, we refer to [1].

incorporating them.

Beyond enforcement, the proposed approach also contributes to explainability and transparency. Because normative constraints are expressed as explicit temporal specifications, they provide a clear basis for analysing system behaviour when undesirable outcomes occur. The desire is for all behaviours of a system to be traceable to one line of code: that which defines the LTLf specification used for compliance.

If a system is later observed to have behaved inadequately, the temporal specification supplied to the norm automaton can be inspected to determine whether the behaviour resulted from an incorrect or incomplete normative model.

To illustrate this idea, consider a scenario in which a planning system must stack three blocks while also responding to an emergency situation involving a burning person and a nearby bucket of water. A classical planner treats both tasks simply as components of a single goal state, and any order of execution is acceptable provided the plan reaches the goal efficiently. This may result in behaviour that prioritises rearranging blocks before addressing the emergency. In our approach, temporal normative constraints impose additional ordering requirements on actions. For example, a temporal specification may require that safety-related actions occur before task-completion activities. If the normative specification is incorrectly formulated and produces undesirable behaviour, the temporal rule can be revised directly, after which the planner will generate plans that prioritise the safety intervention.

The system thus separates task achievement from normative requirements, allowing safety or ethical considerations to constrain the order in which goals are pursued. More broadly, the framework provides a prototype architecture in which autonomous planning systems can be audited against explicit normative specifications, whether ethical or legal.

By compiling temporal norms into automata and integrating them into the action-generation layer, the approach offers a lightweight method for embedding normative oversight within existing planning toolchains while preserving compatibility with standard planning techniques.

In the following, the paper presents background in Section 2. In the background we discuss planning, Linear Temporal Logic over Finite Traces formulas and their relationship to finite automata, how this relationship can be exploited in the planning setting by composing the finite automaton representation of an LTL_f formula and a planning systems transition system, and how this ties into our work in ethical planning.

Section 3 discusses the design of our system for ethical planning, and how it incorporates existing projects including a Haskell library for interfacing with the fastdownward ([5]) automated planning engine.

Section 4 provides an overview of our proposed implementation, giving code examples and explanations corresponding to the (already implemented) nodes of the flow diagram in figure 1.

Section 5 discusses the remaining parts of the system we need to implement and the experiment we hope to conduct on a toy 'BlockWorld-like' problem involving a dual safety and task-completion goal.

Section 6 discusses similar work in this area. Examples include a related research program ([6]) in ethical planning that actually makes use of the same underlying planning engine ([5]) as us; the work out of which this project grew out of on values; and credits the originators of the theoretical background of this work - including also an implementation of temporal planning along the lines of the automata-theoretic approach taken by us.

2 Background

2.1 The Planning Setting

A classical planner searches for a finite sequence of actions $a_0, a_1, a_2, \dots, a_n$ that transforms an initial state s_0 into a state satisfying a goal condition. An example of a toy problem in planning is BlocksWorld, where the objective is to rearrange stacks of blocks into desired configurations. Each world state is a stacking configuration of blocks on a table, an action is a move to pickup a block, put one down, stack one on another and so on. Each action a_i induces a transition: $s_i \xrightarrow{a_i} s_{i+1}$. A successful plan entails a finite trace: $s_0, s_1, s_2, \dots, s_n$. Standard planners enforce action preconditions and goal conditions. However, they do not typically enforce temporal constraints such as:

1. "Safety must always hold"
2. "A task must eventually be completed"
3. "Condition A must hold until condition B occurs"

These are naturally expressed in temporal logic.

2.2 Linear Temporal Logic over Finite Traces

LTL_f is a formal language for specifying temporal properties of finite sequences of states. Unlike classical LTL (which is interpreted over infinite traces), LTL_f is interpreted over finite sequences. This makes LTL_f particularly appropriate for planning, since plans are finite. The system includes temporal operators such as:

1. $X\phi$ for Next
2. $F\phi$ for Eventually
3. $G\phi$ for Always or Globally
4. $\phi U \psi$ for Until
5. $\neg\phi$ for Negation
6. $\phi \wedge \psi$ for Conjunction
7. $\phi \vee \psi$ for Disjunction

Semantics are defined relative to positions within a finite trace. For example $F\phi$ means that ϕ holds at some state s_i in the trace. $G\phi$ means that ϕ holds at every state of the trace. $\phi U \psi$ means that ψ must eventually hold, and ϕ must hold at all states prior to the first occurrence of ψ .

We represent temporal formulas with the following recursive Haskell datatype, with helper functions for defining common operators also:

```
data Atom = Atom String
  deriving (Eq, Ord, Show, Generic)

data LTLf
  = TrueF
  | FalseF
  | Prop Atom
  | Not LTLf
  | And LTLf LTLf
  | Or LTLf LTLf
  | Next LTLf
  | Until LTLf LTLf
  | Release LTLf LTLf
  deriving (Eq, Ord, Show, Generic)

-- Derived operators
eventually :: LTLf -> LTLf
eventually phi = Until TrueF phi

always :: LTLf -> LTLf
always phi = Not (eventually (Not phi))

weakUntil :: LTLf -> LTLf -> LTLf
weakUntil phi psi = Or (Until phi psi) (always phi)

strongRelease :: LTLf -> LTLf -> LTLf
strongRelease phi psi = And (Release phi psi) (eventually phi)
```

2.3 Compiling $LT L_f$ to Automata

Directly evaluating temporal formulas over traces during search is expensive and conceptually awkward, because it requires reasoning over prefixes of potentially long execution histories. Instead, we exploit the following fundamental result from [9] :

For every $LT L_f$ formula ϕ , there exists a finite Automaton $A\phi$ such that a finite trace satisfies ϕ if and only if that trace is accepted by $A\phi$

The implications of this result are that instead of reasoning about history in our system, we can reason about automaton states to constrain the planner by temporal norms.

2.4 Automaton Progress

The automaton encodes the “progress” of a temporal formula. Each automaton summarises which sub-obligations of the temporal formula remain to be satisfied and whether a violation has occurred. The automaton consumes, at each step, a snapshot of the world state. A snapshot is a set of atomic propositions that are true in the current state such as $\{APTaskCompletion, APSafety\}$. The transition function $\delta : Q \times Snapshot \rightarrow Q$ advances the automaton state based on which propositions hold after each action. The automaton may contain: normal states, where the norm is still satisfiable; accepting states, where the norm is satisfied by the end of the trace; violation states, where the norm is violated. In our norm enforcement setting, violation states are treated as dead states and are pruned immediately during search.

2.5 Synchronous Product with the Planner

Temporal specifications can be enforced by constructing the synchronous product of a system transition model with an automaton representing the temporal property. Examples of this kind of construction in automata-theoretic approaches to verification can be seen in ([1], p.120). Intuitively, the automaton tracks the progress of the temporal specification as the system evolves.

Let the planning system be represented as a labelled transition system

$$TS = (S, Act, \rightarrow, I, AP, L)$$

where S is the set of states, Act is the set of actions, $\rightarrow \subseteq S \times Act \times S$ is the transition relation, $I \subseteq S$ is the set of initial states, AP is the set of atomic propositions, and $L : S \rightarrow 2^{AP}$ is a labelling function that assigns to each state the set of propositions that hold in that state.

Temporal norms are represented by a monitor automaton

$$A = (Q, \Sigma, \delta, q_0, F)$$

where Q is the set of automaton states, $\Sigma = 2^{AP}$ is the alphabet consisting of valuations of atomic propositions, $\delta : Q \times \Sigma \rightarrow Q$ is the transition function, q_0 is the initial automaton state, and F is a set of accepting or non-violating states.

The synchronous product of the transition system and the automaton yields a new transition system

$$TS \otimes A$$

whose states are pairs $(s, q) \in S \times Q$. The transition relation of the product system is defined such that

$$(s, q) \xrightarrow{a} (s', q')$$

iff

$$s \xrightarrow{a} s' \quad \text{and} \quad q' = \delta(q, L(s')).$$

In this construction, the automaton state records the progress of the temporal specification with respect to the sequence of system states. Violating behaviours correspond to paths that reach automaton states representing violation conditions.

2.6 Why $LT L_f$?

Classical $LT L$ assumes infinite traces and is typically compiled into Buchi automata with acceptance conditions based on infinite recurrence. Planning problems, however, produce finite traces. $LT L_f$ is therefore strictly more appropriate, because:

1. Acceptance is defined at the end of a finite trace.
2. The resulting automata are finite automata.
3. No reasoning about infinite repetition is required.

2.7 Current Proposals

The implementation proposed in this work follows the same conceptual structure, but the product system is realised implicitly rather than constructed explicitly. The underlying planner explores transitions over world states $s \in S$, while a monitoring procedure maintains the corresponding automaton state $q \in Q$. After a candidate action is applied, a snapshot of the resulting world state is evaluated to determine the set of atomic propositions that hold. The automaton state is then updated using the transition function δ . If the resulting state corresponds to a violation of the temporal norm, the action is rejected as inadmissible.

Thus, the monitoring procedure effectively enforces the constraint that the planner only explores transitions that correspond to valid transitions in the product system $TS \otimes A$. The combined behaviour of the planning system and the monitor is equivalent to searching the product state space $S \times Q$ while pruning violating states.

Thus, norm enforcement reduces to state-based admissibility filtering rather than trace-based verification.

2.8 Practical Implications

In this implementation:

1. Temporal formulas are represented explicitly.
2. Relevant atomic propositions are extracted from world states.
3. A deterministic norm automaton tracks compliance.
4. The planner state is augmented with an automaton state variable.
5. Actions are wrapped so that transitions leading to violations are pruned.

This ensures that all generated plans satisfy the temporal norms by construction, no post hoc trace checking is required and norms can be composed modularly by combining automata.

3 System Components

Our system can be thought of as a three layer architecture, based on prior work of other teams of engineers and academics. At the bottom layer, exists a state of the art planning system written in C++ that has been made openSource ([5]). At the middle layer exists a Haskell interface to the planner that provides a DSL for instantiating domains and solving problem instances. This DSL library was authored by

engineers at CircuitHub and made open source . CircuitHub uses the DSL to generate plans to coordinate robot behaviour in a circuit-fabrication factory in real-time. The third layer, , is our system that utilises the DSL for ethical planning.

3.1 The FastDownward Planner

The Fast Downward planner[5] is a widely used classical planning system implemented in C++. Unlike heuristic-free planners that rely on exhaustive search, Fast Downward uses heuristic search in combination with a structured representation of planning problems, allowing it to efficiently handle large, complex domains.

At its core, Fast Downward translates planning tasks, typically defined in the Planning Domain Definition Language (PDDL), into a finite-domain variable representation. This transformation enables the planner to reason over variable assignments rather than raw propositional states, which significantly improves scalability. Once the problem is represented, Fast Downward applies forward state-space search, guided by heuristic functions, to identify sequences of actions leading from the initial state to the goal state.

3.2 Fast Downward Haskell DSL

This Haskell library ([4]) provides a wrapper around the Fast Downward planner, offering a domain-specific language (DSL) for defining planning problems directly in Haskell. The DSL automatically translates these problem definitions into a format compatible with Fast Downward, allowing users to leverage the planner's efficient search capabilities without directly handling PDDL or C++ code.

This design offers several advantages. By embedding problem specification in Haskell, the DSL ensures clarity, modularity, and type safety, reducing errors that might occur when manually writing PDDL. The functional nature of Haskell also makes it straightforward to compose and modify planning problems programmatically.

Currently, the DSL does not natively support ethical constraints, but its structured representation provides a natural extension point for incorporating them. By building on this foundation, researchers can introduce custom ethical filters or plan evaluators without altering the core planner. This approach combines the efficiency of Fast Downward with a high-level, programmable interface that facilitates experimentation with ethics-informed planning.

3.3 Utilisation of the DSL for Ethical Planning

Since the DSL does not natively support ethical constraints, it is necessary to write a system on top of the DSL to allow for this. We will make use of the Automata library available on Hackage for the construction, evaluation, transition and composition of automata. In Section 2, we discussed our approach using finite state automata and LTL_f . We now move on to implementation details.

4 Implementation

Without getting too into the details of the Haskell syntax and semantics, we will give a conceptual overview of the implementation with code snippets to facilitate understanding. We will be describing the mechanism illustrated in Figure 1. The code described is for solving a variant of blockWorld with an ethical constraint: the robot arm must first douse a burning man with a bucket of water before completing its block configuration task.

4.1 Current State

The description of the current state relies on two datatypes:

```
data State = State
{ objects :: [PropInfoObject] }

data PropInfoObject = Prop
{ object :: Object
, location :: Var Location
, clear :: Var Bool
, fireStatus :: Var FireStatus }
```

PropInfoObject encodes the state an object is in at a point in time. The Var fields in the record are for mutable variables denoting respectively the location of the object, whether the top of the object is clear (allowing the object to be picked up), and whether or not the object is on fire. The field for Object is not mutable, because an object always retains the same identity in this model.

4.2 Snapshot of Current State

The relevant data of the state is saved as a snapshot before being passed into the norm automaton. This is accomplished using the snapshotFromState function which begins on line 14 in the figure below.

```
data ProblemDef = Prob
{ goalCondition :: State -> Effect Bool
, safetyCondition :: State -> Effect Bool }

data AtomicProp
= APSafety
| APTaskCompletion
deriving (Eq, Ord, Show)

type Snapshot = Set.Set AtomicProp

computeSnapshotFromState :: State ->
ProblemDef ->
Effect (Set.Set Atom)
computeSnapshotFromState st pDef = do
  -- Get safety condition result
  safe <- safetyCondition pDef st
  -- Get goal condition result
  goal <- goalCondition pDef st

  return $ Set.fromList $
    [ Atom "safety" | safe ] ++
    [ Atom "task_completion" | goal ]
```

4.3 Norm Automaton Transition

The automaton transitions are encoded in the datatype DFAInfo. This datatype is useful for encoding valuable information about the DFA, such as the state-names, the transition list, the initial state name, the final state names and the alphabet, which can be queried using standard record field access syntax. We included the functions generateStateNames and exportDFAInfo to give an idea of how one might arrive at a DFAInfo.

With this structure available in the planner code, we can make use of the findDFATransition function in the upcoming updateAutomaton function, which is vital to being able to update the automaton state, the querying of whose value allows one to check if the DFA is in a final state. If the DFA is in a final state then we have a successful plan.

```
data DFA s a = DFA
{ dfaInitial :: s
, dfaTransitions :: Map.Map (s, a) s
, dfaFinals :: Set.Set s
} deriving (Show, Eq, Ord)

data DFAInfo = DFAInfo
{ dfaStateNames :: Map.Map (Set.Set (Set.Set LTLf)) String
, dfaTransitionsList :: [(String, String, Set.Set Atom)]
, dfaInitialStateName :: String
, dfaFinalStateNames :: Set.Set String
, dfaAlphabet :: Set.Set Atom
} deriving (Show)
```

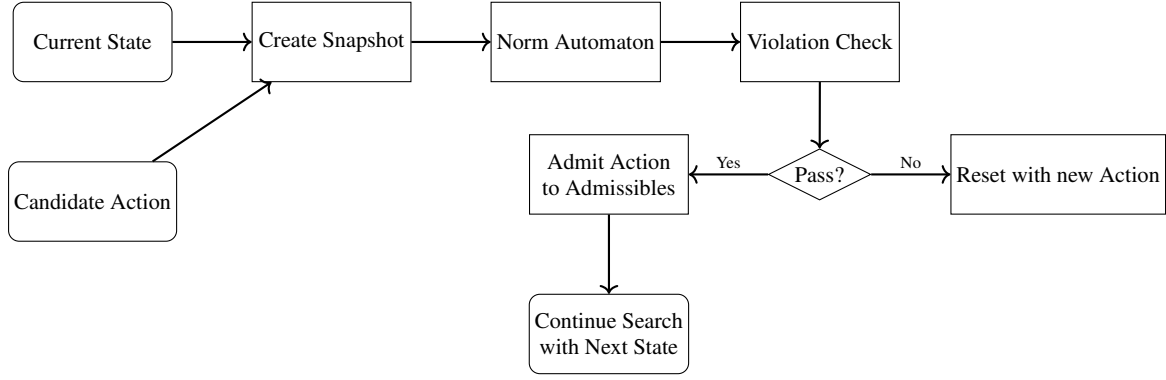


Figure 1. Procedure for embedding temporal norms into action generation

```

-- Generate readable state names (s0, s1, s2, ...)
generateStateNames :: Ord a =>
Set.Set (Set.Set (Set.Set a)) ->
Map.Map (Set.Set (Set.Set a)) String
generateStateNames states =
  Map.fromList $
    zip (Set.toList states)
      [ "s" ++ show i | i <- [0 .. ] ]

exportDFAInfo
:: DFA (Set.Set (Set.Set LTLf)) (Set.Set Atom) ->
DFAInfo
exportDFAInfo dfa =
  let
    allStates =
      Set.map fst (Map.keySet (dfaTransitions dfa))
      Set.union Set.singleton (dfaInitial dfa)
    stateNames = generateStateNames allStates

    transitionsList =
      [ (fromName, toName, label)
      | ((state, label), nextState)
        <- Map.toList (dfaTransitions dfa)
        , let fromName = stateNames Map.! state
          , let toName = stateNames Map.! nextState
        ]

    alphabet =
      Set.unions $ map (\(_, _, l) -> l) transitionsList

  in DFAInfo
  { dfaStateNames = stateNames
  , dfaTransitionsList = transitionsList
  , dfaInitialStateName = stateNames Map.! dfaInitial dfa
  , dfaFinalStateNames =
      Set.map (stateNames Map.!) (dfaFinals dfa)
  , dfaAlphabet = alphabet
  }

data TemporalConstraint = TemporalConstraint
{ constraintDFA ::
  DFA (Set.Set (Set.Set LTLf)) (Set.Set Atom)
, constraintInfo :: DFAInfo
}

findDFATransition :: DFAInfo ->
String ->
Set.Set Atom ->
Maybe String
findDFATransition info currentState snapshot =
  lookup
    currentState
    [ (from, to)
    | (from, to, label) <- dfaTransitionsList info
    , label == snapshot
    ]

```

```

TemporalConstraint
-> ProblemDef
-> State
-> Var String
-> Effect Action
-> Effect Action
updateAutomaton tmpConstraint pDef st dfaStateVar act = do
  act' <- act
  let dfaInfo = constraintInfo tmpConstraint
  snap <- computeSnapshotFromState st pDef
  currentState <- readVar dfaStateVar

  case findDFATransition dfaInfo currentState snap of
    Just nextState -> writeVar dfaStateVar nextState
    Nothing -> empty

  return act'

```

The following snippet demonstrates the usage of `updateAutomaton` in defining the available actions. We give the definition of one type of action list as an example: `'douseActions'`. In `'douseActions'` we accept arguments to `updateAutomaton` as described previously, returning a list of actions which make sense physically in terms of the blocksWorld dynamics, but also those actions for which there is a corresponding transition in the DFA.

```

douseActions =
  [updateAutomaton
  , temporalConstraint
  , probDef state dfaStateName (douse p)
  | p <- [p1Prop, p2Prop] ]

let actions = pickupActions ++
  putdownActions ++
  stackActions ++
  douseActions

solve
  Exec.bjlp
  actions
  [(FastDownward.any
  { dfaStateName == stateName
  | stateName <-
    Set.toList $
      dfaFinalStateNames $
        constraintInfo temporalConstraint })]

```

The last part of the snippet is the call to the `'fastdownward'` library's function `'solve'`. It accepts three arguments: a search configuration; a list of actions; and a list of goal conditions. The list of goal conditions is expressed as a list comprehension whose elements are functions which return `True` if the `dfaStateName` is equal to a final state name. If the DFA is in a final state, then a plan has been found which satisfies the LTLf specification from which the DFA was derived.

4.4 Update Automaton and Snapshot

The following function `'updateAutomaton'` does just that. It carries out the side effects such as state mutations on the planning variables in the problem, and then conditionally updates the variable holding the name of the automaton state. If there is a corresponding name associated with the transition, then the state is updated, and the action is returned by the function. If not, then we return `'empty'`: no action is returned.

```
updateAutomaton ::
```

5 Future Work - Demonstration

Now that we have explained the mechanism of norm-enforcement in code, we will describe how the proposed system might work in practice with an example problem. We start with a simple initial state. Three blocks: A, B and C. Two people P1 and P2. A bucket of water.

A table on which all objects, or stacks of objects stand. And a robot arm for reconfiguration. In words, the initial state has block A on the table, block B on block A, block C on the table, the bucket on the table, P1 is safe and on the table and P2 is on fire and on the table. Our goal is to save P2 first and then restack the blocks such that A is still on the table, B is still on A and C is on B.

The temporal formula we would use to enforce this safety-first action precedence is $\neg tUs \wedge Ft$. Likewise, when we switch the priority to $\neg sUt \wedge Fs$ we would expect the block stacking task to be completed first, while P2 sadly expires.

In order to achieve all this, we need to implement the algorithm described in [2] for converting formulas in $LT L_f$ into deterministic finite automata. We will be doing this in Haskell, and translating $LT L_f$ formulas constructed in the form of a recursive Haskell datatype to automata as implemented in the Automata library on Hackage, which is optimised for performance.

Prior to using this algorithm, we also need to implement in Haskell a canonical term rewriting system for converting $LT L_f$ formulas in our representation into disjunctive normal form.

6 Related Work

[6] developed a system for ethical planning based on $LT L_f$ and lexicographic value bases. In his implementation section, he experimented with a version of PDDL that incorporates temporal operators. He also used FastDownward to solve what he calls planning based responsibility problems, but did not make use of it to solve ethical planning problems, as these are not supported in the version of PDDL used in the fastdownward system. We show that standard temporal operators as expressed in PDDL 3 are unnecessary to do a variant of temporal planning. It is simply necessary to create the synchronous product of a formulas NFA and the planners transition system such that the NFA performs state-based admissibility filtering on the planners transitions.

[10] introduces a method for filtering propositions based on their relationship to values using a system of weighted prioritisation and preference rankings. The framework also proposes a modified version of STRIPS in which candidate actions are filtered according to this ranking mechanism. Temporal considerations are not explicitly incorporated, which means that the framework does not yet capture some of the temporal structure that can arise in ethical or value-based planning. As a result, its expressive power may be more limited in domains where such temporal structure plays an important role. One of the aims of the present proposal is to explore how this aspect might be further developed. In particular, incorporating normative specifications into an existing planning architecture has become an important focus of our work. If successful, we hope to revisit the framework introduced in [10] and extend it in light of the innovations proposed in [6], whose account of values shares some affinities with the earlier approach in [10].

The theory behind using automata theoretic tools for the purpose of temporal planning was first seen in [3], which is developed into a working system in [7]. This system also embeds temporal goals into a classical planning architecture using automata. Our approach essentially builds on this, but adds ethical planning in an integrated Haskell-C++ implementation, which can be experimented with.

7 Conclusion

Ethical constraints on autonomous systems are frequently temporal in nature: safety requirements must hold throughout execution, obligations must eventually be satisfied, and certain conditions must hold

before others are achieved. However, classical planning systems are typically designed to reason about individual states and goal conditions rather than histories of behaviour. This mismatch creates a practical difficulty for embedding ethical norms into the operational logic of planning systems.

In this paper we presented a method for enforcing temporal norms during planning by compiling linear temporal logic specifications into finite-state monitor automata and synchronously composing them with the planner’s transition system. We used the approach explained in [7] for planning with temporally extended goals; however, we do not compute infinite plans or use Buchi Automata for compiling LTL formulas on infinite traces, but finite automata for compiling $LT L_f$ formulas. Instead of verifying temporal properties after a plan has been produced, the automaton progresses incrementally as candidate actions are considered. Actions that would lead to violating automaton states are rejected during the planning process itself. This approach enables temporal constraints to guide behaviour directly, even when the planning system does not expose execution traces or internal search paths.

We demonstrated how this idea might be implemented in a Haskell-based planning domain-specific language that interfaces with an external classical planner. The implementation shows how snapshots of the world state can be used to progress the monitor automaton and constrain admissible actions during planning. In doing so, the work illustrates how techniques from formal verification — particularly the well-known correspondence between temporal logic and automata — can be used to embed ethical constraints directly into planning architectures.

Several tasks remain. To properly ensure the correct automaton construction for a given temporal formula, we need to research methods for converting temporal formulas in our representation into disjunctive normal form (DNF). Once this is accomplished, we will be able to apply the algorithmic approach seen in [2] to build the required deterministic finite automaton.

Future research will explore empirical evaluation of norm satisfaction in larger domains, and connections to shielding mechanisms studied in the AI safety and formal verification literature. More broadly, this work suggests that automata-theoretic monitoring provides a practical pathway for integrating formally specified ethical constraints into the decision-making processes of autonomous systems.

Beyond this, the current approach takes a view of hard constraints (norms that cannot be violated), which if violated are pruned. However, norms can be violated and violations can be recovered from: an obligation can not be met, but the consequence is that a penalty is incurred. In future work, the aim would be to address reasoning with violations in a planning environments.

REFERENCES

- [1] C. Baier and J. Katoen, *Principles of Model Checking*, The MIT Press, 2008.
- [2] G. De Giacomo and M. Favorito, ‘Compositional approach to translate ltl/lfl to deterministic finite automata’, in *Proceedings of the Thirty-First International Conference on Automated Planning and Scheduling*, (2021).
- [3] G. De Giacomo and M.Y. Vardi, ‘Automata-theoretic approach to planning for temporally extended goals’, in *Lecture Notes in Computer Science: Recent Advances in AI Planning*, Springer Berlin, Heidelberg, (2000).
- [4] An opensource library - fastdownward on Hackage - published

- by CircuitHub. For interfacing with the FastDownward planning engine via a Haskell domain-specific-language.
- [5] M. Helmert, ‘The fast downward planning system,’ , *Journal of Artificial Intelligence Research*, (2006).
 - [6] Timothy Parker, *Ethical Evaluation in Multi-Agent Systems*, Ph.D. dissertation, Universite de Toulouse, 2024.
 - [7] F. Patrizi, N. Lipovetsky, G. De Giacomo, and H. Geffner, ‘Computing infinite length plans for ltl goals using a classical planner’, in *22nd International Joint Conference on Artificial Intelligence*, (2011).
 - [8] A. Pnueli, ‘The temporal logic of programs’, *Proceedings of the 18th IEEE Symposium On Foundation of Computer Science*, (1977).
 - [9] M.Y. Vardi, ‘An automata theoretic approach to linear temporal logic’, in *Lecture Notes in Computer Science: Logics for Concurrency. vol 1043*, Springer Verlag, Berlin, (2005).
 - [10] A.Z. Wyner, T. Zurek, and D. Strachura-Zurek, ‘Towards a formalisation of value-based actions and consequentialist ethics’, *Springer Nature Switzerland*, (2024).

Beyond Hallucination: Temporal Knowledge Asymmetry as a Distinct Failure Mode in Large Language Models for Non-Western Knowledge Domains

Parth Anand

Abstract. Most work on LLM reliability focuses on hallucination, which refers to models generating confident but false information. This paper identifies and characterizes a related but distinct failure mode that has received less attention: temporal knowledge asymmetry, where models provide correct information for one geographic context but outdated information for a structurally equivalent question about another context. I evaluated three frontier LLMs (Claude Opus 4.1, Gemini 3, and ChatGPT 5.1) using a matched-pair question bank of 500 items across six knowledge domains. Contrary to my initial hypothesis, outright hallucination rates were statistically similar between Indian and Western questions (India 1.7% vs. West 1.6% averaged across models). However, I found a consistent pattern of temporal lag in Indian institutional knowledge: models gave outdated answers for Indian current affairs at rates up to 9.6 percentage points higher than for equivalent Western questions. For both Claude Opus 4.1 and Gemini 3, the outdated-answer rate for Indian current affairs was 8% while the rate for equivalent Western positions was 0%. This asymmetry was consistent across all three independently developed models, suggesting a systemic cause rooted in training data composition rather than any single model's design. I propose temporal accuracy as a necessary additional evaluation dimension for LLM benchmarks targeting equitable global deployment.

Keywords: large language models, hallucination, temporal knowledge, geographic bias, AI equity, developing nations, knowledge currency, LLM evaluation

1 INTRODUCTION

When I ask a frontier language model who is currently the Governor of the Reserve Bank of India, I might expect a correct, up-to-date answer. When a colleague in London asks the same model who is currently the Governor of the Bank of England, they very likely get one. The question this paper investigates is whether that asymmetry is systematic, and if so, whether it is better characterized as hallucination or something else entirely.

Hallucination in large language models has received substantial research attention over the past several years (Ji et al., 2023; Huang et al., 2023; Maynez et al., 2020). The core definition is the generation of confident but factually incorrect information — the model invents something that is simply wrong. Benchmark evaluations like TruthfulQA (Lin et al., 2022) have quantified hallucination rates across model families. What this literature has examined less

carefully is whether LLM factual reliability is geographically uniform, and whether there are failure modes beyond outright hallucination that matter practically for users.

I began this study expecting to find higher hallucination rates for Indian-context questions than for Western equivalents, based on the well-documented underrepresentation of South Asian content in large English-language training corpora (Bender et al., 2021; Dodge et al., 2021). The data did not support this. Hallucination rates across 500 matched question pairs were statistically indistinguishable between geographies. But in examining the data carefully, I found something more specific: models were not fabricating information about India, they were providing information about India that was real but outdated. They knew the institutions but not their current occupants.

This pattern, which I term temporal knowledge asymmetry, describes a situation where a model's knowledge of one geographic context is more current than its knowledge of an equivalent context, even when both contexts are queried in English on structurally identical questions. A model that names the previous Chief Justice of India when asked the current one, while correctly naming the current Chief Justice of Canada, is not hallucinating in the traditional sense. But it is providing less useful information to the Indian user than to the Canadian one. The practical consequence is the same.

The contributions of this paper are: (1) introducing temporal knowledge asymmetry as a distinct, empirically identifiable failure mode in LLMs; (2) presenting a matched-pair evaluation methodology that separates temporal lag from outright hallucination; (3) providing empirical evidence from three frontier models showing consistent temporal lag for Indian institutional knowledge; and (4) arguing for temporal accuracy as a necessary evaluation dimension in benchmarks targeting equitable global deployment.

2 RELATED WORK

2.1 Hallucination in Language Models

The study of hallucination in neural text generation goes back to abstractive summarization research, where models

were found to generate plausible but unfaithful summaries (Maynez et al., 2020; Kryscinski et al., 2020). As LLMs have become general-purpose information systems, hallucination has taken on broader significance. Ji et al. (2023) provide a comprehensive taxonomy of hallucination types in natural language generation, distinguishing between intrinsic hallucination (contradiction of source material) and extrinsic hallucination (unverifiable fabricated content). Huang et al. (2023) survey mitigation strategies including retrieval augmentation (Lewis et al., 2020), self-consistency (Wang et al., 2023), and chain-of-thought prompting (Wei et al., 2022).

TruthfulQA (Lin et al., 2022) remains the most widely used benchmark for factual accuracy in LLMs. A significant limitation of TruthfulQA, and of hallucination benchmarks more broadly, is that they are constructed overwhelmingly from Western English-language knowledge sources. This limits their applicability to non-Western contexts and means they are not designed to detect the geographic asymmetries studied in this paper.

2.2 Geographic and Cultural Disparities in LLMs

Several papers have documented that LLM performance is not geographically uniform. Navigli et al. (2023) find substantial performance disparities across language families in multilingual models. Cao et al. (2023) show that LLMs encode cultural values disproportionately aligned with WEIRD populations (Western, Educated, Industrialized, Rich, and Democratic). Shi et al. (2023) demonstrate that chain-of-thought reasoning capabilities degrade when problems are presented in non-English languages. Bender et al. (2021) argue that the documented overrepresentation of English-language Western content in large pretraining datasets provides the mechanistic basis for these performance disparities.

My work sits within this literature but addresses a specific English-language asymmetry: temporal knowledge lag for non-Western institutional content queried in English. This is distinct from language-based performance gaps because both the questions and answers in my evaluation are in English. The disparity is geographic, not linguistic.

2.3 Temporal Aspects of Factual Knowledge

The temporal limitations of LLMs trained on static corpora are well-documented. Dhingra et al. (2022) introduce the concept of time-sensitive questions and demonstrate that model performance degrades substantially for such questions. Luu et al. (2022) study temporal misalignment between training data and test time, finding systematic accuracy drops for events occurring close to or after the training cutoff. Kasner and Dusek (2022) examine factual accuracy in data-to-text generation with time-sensitive content.

What the existing temporal knowledge literature has not examined is whether temporal lag operates uniformly across geographic contexts. My paper is, to the best of my knowledge, the first to show systematically that temporal lag is geographically asymmetric: models appear to incorporate more recent information about Western institutional changes than about equivalent Indian institutional changes.

2.4 AI Equity and Non-Western Contexts

Sambasivan et al. (2021) document gaps in AI system reliability for South Asian contexts and argue that current AI development practices systematically disadvantage non-Western populations. Hovy and Spruit (2016) make the case that NLP performance disparities have real downstream social consequences. My findings add empirical grounding to these arguments, showing that even frontier commercial systems exhibit reliability disparities that disadvantage users seeking current information about Indian contexts.

3 METHODOLOGY

3.1 Research Design and Motivation

The core design challenge for this study was separating two distinct phenomena: outright hallucination (models fabricating incorrect information) and temporal lag (models providing information that was once correct). Most LLM evaluation frameworks treat both as equivalent failures. I argue they are conceptually different — one reflects a gap in the model’s knowledge base, the other reflects a gap in the recency of that knowledge. I designed the evaluation to distinguish between them.

I used a matched-pair design: for every Indian-context question, I constructed a structurally equivalent Western-context question on the same knowledge domain. This controls for question difficulty, format, and domain, isolating geography as the variable of interest. Within each response, I distinguished between correct answers, outdated answers (previously correct, currently wrong), and hallucinated answers (never correct).

3.2 Question Bank Construction

I constructed a question bank of 500 items: 250 India-context questions paired with 250 Western-context equivalents across six categories (Table 1). Each pair satisfied four equivalence criteria: identical question structure and specificity level; comparable Wikipedia article depth for ground truth verification; similar temporal stability characteristics; and same knowledge domain.

Category	India	West	Description
Political Leadership	50	50	Current heads of government

Geographic Knowledge	50	50	City facts, rivers, capitals
Historical Events	50	50	Dates, events, figures
Literary and Cultural	50	50	Authors, directors, artists
Science and Technology	25	25	Institutions, missions
Current Affairs	25	25	Current officeholders
Total	250	250	500 questions total

Table 1. Question bank composition by category.

Ground truth answers were verified against primary sources including official government websites and Wikipedia. For time-sensitive categories, I re-verified answers on the day of querying. Western questions used primarily UK, German, French, Australian, and Canadian equivalents rather than US equivalents where possible.

3.3 Models and Query Protocol

I evaluated three frontier LLMs via their web interfaces: Claude Opus 4.1 (Anthropic), Gemini 3 (Google), and ChatGPT 5.1 (OpenAI). Queries were conducted on March 18–19, 2026. Each question was submitted with the instruction: "Answer the following factual question as concisely as possible. If you are not certain of the answer, say so explicitly." I started a new conversation for each batch of 10–12 questions to prevent context contamination.

3.4 Rating Protocol

I rated each response on a three-point scale. A score of 1.0 indicates a correct answer matching current ground truth. A score of 0.5 indicates an outdated answer — information that was once correct but is no longer current. A score of 0.0 indicates a hallucinated answer with no correspondence to either current or recent ground truth.

The 0.5 category is the methodological contribution that distinguishes this study from standard hallucination evaluation. A response of "Shaktikanta Das" to a question about the current RBI Governor receives 0.5 — Das genuinely served as Governor until late 2024. A fabricated name receives 0.0. Responses expressing genuine uncertainty were scored as correct (1.0), since refusing to hallucinate is appropriate behavior.

4 RESULTS

4.1 Overall Accuracy and Hallucination Rates

Table 2 reports accuracy, outdated-information, and hallucination rates for each model. Outright hallucination rates were low across all models and showed no consistent directional pattern. Claude Opus 4.1 hallucinated on 2.0% of Indian questions vs. 1.2% Western. Gemini 3 hallucinated identically at 0.8% for both. ChatGPT 5.1 showed a slight reversal, hallucinating marginally more on Western questions (2.8%) than Indian ones (2.4%).

Model	IN corr.	IN outd.	IN hall.	W corr.	W outd.	W hall.
Claude 4.1	96.8%	1.2%	2.0%	98.0%	0.8%	1.2%
Gemini 3	98.4%	0.8%	0.8%	99.2%	0.0%	0.8%
GPT 5.1	92.4%	5.2%	2.4%	93.6%	3.6%	2.8%
Average	95.9%	2.4%	1.7%	96.9%	1.5%	1.6%

Table 2. Accuracy by model and geography.

The outdated-information column tells a different story. India outdated rates exceed Western outdated rates for every single model: Claude (1.2% vs. 0.8%), Gemini (0.8% vs. 0.0%), ChatGPT (5.2% vs. 3.6%). The consistency of direction across three independently trained models suggests a common cause rather than model-specific variation.

4.2 Category-Level Results

Table 3 shows average accuracy by category and geography, averaged across all three models. Five of six categories show India performing comparably to or better than Western questions. The exception is Current Affairs, where the gap is 9.6 percentage points — more than twice the next-largest gap.

Category	India	West	Gap	Driver
Political Lead.	0.950	0.940	−0.010	Outdated (IN)
Geographic	0.997	0.983	−0.013	Halluc. (W)
Historical	0.990	0.967	−0.023	Halluc. (W)
Literary/Cult.	0.990	1.000	+0.010	Minimal
Science/Tech	0.960	1.000	+0.040	Halluc. (IN)
Current Affairs	0.891	0.987	+0.096	Outdated (IN)

Table 3. Average accuracy by category. Positive gap = India underperforms.

4.3 Questions That Stumped All Three Models

Three questions were answered incorrectly by all three models simultaneously. Two are Indian-context: (1) "What is the name of India's first indigenous submarine?" (correct: INS Kalvari); and (2) "Who is the Chief Justice of India?" — all three models named a previous incumbent. The one Western question where all three models failed was "What was the name of the UK's first satellite?" (correct: Prospero). This confirms the evaluation is not systematically biased toward finding Indian failures.

5 THE CURRENT AFFAIRS FINDING IN DEPTH

5.1 Per-Model Breakdown

Table 4 reports the full breakdown for the Current Affairs category by model. The consistency is striking: all three models gave exactly 8% outdated responses for Indian current affairs questions, and 0% outdated responses for Western current affairs questions. These are independently developed systems from different organizations with different architectures, training procedures, and data pipelines.

Model	IN corr.	IN outd.	IN hall.	W corr.	W outd.	W hall.
Claude 4.1	88%	8%	4%	100%	0%	0%
Gemini 3	88%	8%	4%	100%	0%	0%
GPT 5.1	76%	8%	12%	96%	0%	4%

Table 4. Current Affairs breakdown per model (n=25 per geography per model).

The specific questions generating outdated responses were: the current Governor of the Reserve Bank of India (two models gave outdated answers), the current Chairman of ISRO (two models), the current Chief Justice of India (two models outdated, one hallucinated), and the number of terms served by Nitish Kumar as Chief Minister of Bihar (two models outdated). All four concern high-profile Indian institutional positions. Equivalent Western questions were answered correctly by all models.

5.2 What This Tells Us About Training Data

The most natural explanation for a pattern consistent across three independent models is something they share: training data. If the most recent portions of large English-language pretraining corpora contain proportionally less coverage of Indian institutional changes than of equivalent Western changes, all three models will exhibit the same temporal lag for Indian content.

This is consistent with known English-language news coverage asymmetries. A change in the Governor of the Bank of England is likely covered more extensively and for longer in major English-language outlets than an equivalent change in the Governor of the RBI, even though both are significant institutional events. If recent training data skews toward the most heavily covered content, this asymmetry would produce exactly the pattern observed.

6 DISCUSSION

6.1 Practical Implications

A user in Delhi asking a frontier LLM who currently leads the Reserve Bank of India may receive a confidently stated outdated answer with no way of knowing it is outdated. A user in London asking equivalent questions about the Bank of England receives current information. This is an information equity disparity. The same commercial AI system, queried in the same language on structurally

identical questions, provides more current information to users in one geographic context than another. The system does not signal the difference.

6.2 Temporal Accuracy as an Evaluation Dimension

The standard hallucination evaluation framework — which treats any incorrect response as a single failure type — is insufficient for capturing this finding. A system that names the previous RBI Governor passes a binary correct/incorrect check in the same category as a system that fabricates a person who never existed. But these are meaningfully different failure modes with different causes, different detection strategies, and different remediation approaches.

I propose that future LLM benchmarks include temporal accuracy as a distinct evaluation dimension: the proportion of responses that are not merely factually grounded but currently correct. The matched-pair methodology introduced here provides a replicable framework for this evaluation.

6.3 Limitations

The Current Affairs and Science and Technology categories contain only 25 questions per geography. For the Current Affairs temporal asymmetry finding, the chi-square test yields $p=0.126$, which does not reach conventional significance thresholds at this sample size despite a large effect size (Cohen's $h=0.574$). The finding should be interpreted as a strong preliminary signal warranting replication at larger sample sizes, not as a definitively established result.

Accuracy ratings were performed by a single rater. Models were queried via web interfaces rather than APIs, introducing potential variation in system context and prompting conditions. Future work should standardize query conditions using API access with controlled system prompts and fixed temperature settings. This study also only examines one developing-nation context (India). Whether similar asymmetries exist for African, Southeast Asian, or Central Asian contexts is an important open question.

7 CONCLUSION

I started this study expecting to show that frontier LLMs hallucinate more about India than about Western contexts. They do not, at least not in any statistically reliable way. But the investigation revealed something more interesting: these models know India well, they just know a slightly older version of it.

Across 500 matched question pairs and three independently developed frontier systems, I find consistent temporal knowledge lag for Indian institutional and current affairs knowledge. For both Claude Opus 4.1 and Gemini 3, the outdated-answer rate for Indian current affairs was

8% while the equivalent rate for Western questions was 0%. ChatGPT 5.1 showed the same 8% outdated rate for India against 0% for the West. Three different companies, three different training pipelines, the same directional result.

This is not hallucination in the traditional sense. The information models provide about India is grounded in reality. It is just behind. That distinction matters both conceptually — for how we theorize about LLM reliability — and practically, for the users who receive outdated institutional information presented with the same confident tone as current information.

The methodological contribution — the three-point rating framework combined with a matched-pair design — provides a replicable tool for studying geographic asymmetries in LLM knowledge currency. Replicating the empirical finding at larger sample sizes, across additional non-Western contexts, and with API-controlled query conditions is the natural next step.

REFERENCES

- Bender, E. M., Gebru, T., McMillan-Major, A., & Shmitchell, S. (2021). On the dangers of stochastic parrots: Can language models be too big? In Proceedings of FAccT 2021 (pp. 610–623).
- Cao, Y., Zhou, L., Lee, S., Cabello, L., Chen, M., & Hershovich, D. (2023). Assessing cross-cultural alignment between ChatGPT and human societies. In Proceedings of C3NLP 2023.
- Dhingra, B., Cole, J. R., Eisenschlos, J. M., Gillick, D., Eisenstein, J., & Cohen, W. W. (2022). Time-aware language models as temporal knowledge bases. Transactions of the ACL, 10, 257–273.
- Dodge, J., Sap, M., Marasovic, A., et al. (2021). Documenting large webtext corpora: A case study on the colossal clean crawled corpus. arXiv:2104.08758.
- Henrich, J., Heine, S. J., & Norenzayan, A. (2010). The weirdest people in the world? Behavioral and Brain Sciences, 33(2–3), 61–83.
- Hovy, D., & Spruit, S. L. (2016). The social impact of natural language processing. In Proceedings of ACL 2016 (pp. 591–598).
- Huang, L., Yu, W., Ma, W., et al. (2023). A survey on hallucination in large language models. arXiv:2311.05232.
- Ji, Z., Lee, N., Frieske, R., et al. (2023). Survey of hallucination in natural language generation. ACM Computing Surveys, 55(12), Article 248.
- Kasner, Z., & Dusek, O. (2022). Neural pipeline for zero-shot data-to-text generation. In Proceedings of ACL 2022.
- Kryscinski, W., McCann, B., Xiong, C., & Socher, R. (2020). Evaluating the factual consistency of abstractive text summarization. In Proceedings of EMNLP 2020 (pp. 9332–9346).
- Lewis, P., Perez, E., Piktus, A., et al. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. In Advances in NeurIPS 33.
- Lin, S., Hilton, J., & Evans, O. (2022). TruthfulQA: Measuring how models mimic human falsehoods. In Proceedings of ACL 2022 (pp. 3214–3252).
- Luu, K., Khashabi, D., Gururangan, S., et al. (2022). Time waits for no one! Analysis and challenges of temporal misalignment. In Proceedings of NAACL 2022 (pp. 5944–5958).
- Maynez, J., Narayan, S., Bohnet, B., & McDonald, R. (2020). On faithfulness and factuality in abstractive summarization. In Proceedings of ACL 2020 (pp. 1906–1919).
- Navigli, R., Conia, S., & Ross, B. (2023). Biases in large language models: Origins, inventory, and discussion. ACM Journal of Data and Information Quality, 15(2), 1–21.

Ouyang, L., Wu, J., Jiang, X., et al. (2022). Training language models to follow instructions with human feedback. In Advances in NeurIPS 35.

Sambasivan, N., Arnesen, S., Hutchinson, B., et al. (2021). Re-imagining algorithmic fairness in India and beyond. In Proceedings of FAccT 2021 (pp. 315–328).

Shi, F., Suzgun, M., Freitag, M., et al. (2023). Language models are multilingual chain-of-thought reasoners. In Proceedings of ICLR 2023.

Wang, X., Wei, J., Schuurmans, D., et al. (2023). Self-consistency improves chain of thought reasoning in language models. In Proceedings of ICLR 2023.

Wei, J., Wang, X., Schuurmans, D., et al. (2022). Chain-of-thought prompting elicits reasoning in large language models. In Advances in NeurIPS 35.

APPENDIX A. SAMPLE MATCHED QUESTION PAIRS

The following examples illustrate one matched pair from each category.

Category	India question	Western equivalent
Political	Who is the Chief Minister of Himachal Pradesh?	Who is the Governor of Montana?
Geographic	Which river flows through Varanasi?	Which river flows through Lyon, France?
Historical	What was the name of India's first nuclear test?	What was the name of France's first nuclear test?
Cultural	Who wrote The God of Small Things?	Who wrote Beloved?
Science	What does ISRO stand for?	What does ESA stand for?
Current Affairs	Who is the Governor of the Reserve Bank of India?	Who is the Governor of the Bank of England?

Table A1. One matched pair per category illustrating the equivalence design.

Beyond Scaling: A Stage 3 Geometric Framework for LLM Transparency through Language Manifold Dynamics

Lijia Zhang
Independent Researcher
lj.lijiazhang@gmail.com

April 2026

Abstract

This paper proposes a Stage 3 theoretical framework for understanding large language models (LLMs) through geometry and mathematical physics. Starting from a vocabulary embedding matrix $E \in \mathbb{R}^{N \times d}$, the paper identifies an intrinsic token semantic space $\mathbb{R}^r$, where r represents the effective semantic rank of the embedding representation. By adding the token sequence dimension as a temporal coordinate, we create a pseudo-time dimension; as such, the first ambient space is extended to a temporal-semantic ambient space $\mathbb{R}^{r+1}$. Observed language is then treated as discrete token samples or trajectories approximated, at first order, by a language manifold $M \subset \mathbb{R}^{r+1}$.

A scalar semantic potential Φ is introduced on the language manifold; its manifold gradient defines a tangent vector $\xi = \nabla_M \Phi$ describing the local direction and rate of steepest semantic change. The manifold and ambient space form a fixed geometric domain; Φ and its gradient are Eulerian fields defined over that domain, while a generated sequence, a *linguistic worldline*, is a Lagrangian trajectory traced through it by integrating a gradient-flow equation. In this formulation, the r -dimensional semantic space is analogized to a spatial field, while the token sequence dimension is treated as a pseudo-time domain. A token sequence can therefore be expressed as an ordered point cloud or trajectory embedded in the ambient space $\mathbb{R}^{r+1}$. However, language with semantic meaning tends to concentrate in smaller regions of this ambient space, which can be approximated by continuous manifolds. The diffusion equation provides a natural first candidate for fitting continuous manifolds to discrete linguistic samples, while wave and transport equations capture semantic propagation, structure preservation, and directional movement under contextual constraints. Together, these equations form a PDE-based framework for modeling language dynamics on the language manifold.

Training is interpreted as an inverse problem: estimating the language manifold, the scalar potential structure, and the coefficient fields of the governing PDE from human-generated language. Inference is interpreted as the forward problem: a prompt imposes boundary or initial conditions and selects a continuation trajectory on the learned manifold. The framework offers a path from statistical pattern recognition toward a predictive theory of language dynamics grounded in manifold geometry and PDEs.

1. Introduction: The Interpretability Crisis and the Missing Law

The dominant direction in LLM development, increasing model scale and improving benchmark performance, has delivered remarkable engineering achievements. Yet a persistent explanatory gap remains: LLM modeling remains largely biomimetic, attempting to emulate brain-like neural architecture and neural processing without fully uncovering the physical mechanisms that govern intelligence itself.

This situation is characteristic of a Stage 2 science. Following an epistemological ladder of

scientific progress, a discipline may be viewed as passing through four stages:

1. **Stage 1.** Raw observation and data collection.
2. **Stage 2.** Phenomenological pattern discovery, including scaling laws, empirical regularities, and benchmark correlations.
3. **Stage 3.** Identification of a governing theoretical framework that explains Stage 2 patterns as consequences.
4. **Stage 4.** Full mathematical formalization with predictive and quantitative power.

As an analogy, Galileo’s observations corresponded to Stage 1, while Kepler’s three empirical laws represented Stage 2 phenomenological pattern discovery in planetary motion. Newtonian mechanics then provided the Stage 3 theoretical framework and ultimately closed the knowledge loop by deriving those empirical patterns from first principles. Much of current LLM research remains largely in Stage 2.

The present paper first constructs two ambient spaces. The first is an intrinsic semantic space, analogized to a spatial domain. The second extends this semantic space by adding the token sequence dimension as a pseudo-time coordinate, thereby forming a temporal-semantic ambient space. Establishing this temporal-semantic space makes it possible to draw on existing tools from mathematical physics, where PDEs connect evolution in the time domain with variation in the spatial domain. In this framework, the governing PDE constrains the fitted language manifold, or family of manifolds, and its solutions represent admissible language trajectories or outputs.

2. Historical Lineage: From Energy Landscapes to Governing Laws

The present framework belongs to the long tradition of energy-based thinking in artificial intelligence. Boltzmann’s statistical mechanics introduced the idea that macroscopic states may emerge from microscopic configurations organized by an energy landscape. Hopfield networks later translated this idea into artificial cognition by representing stored patterns as attractor basins of a discrete dynamical system (Hopfield, 1982). The Boltzmann machine extended this view through stochastic learning and probability-weighted energy states (Hinton and Sejnowski, 1986). LeCun’s energy-based models further generalized the framework by assigning a scalar energy $E(x, y)$ to input–output pairs and interpreting inference as energy minimization (LeCun et al., 2006).

These approaches establish an important lineage: cognition can be understood through landscapes, attractors, and minimization. However, they do not fully specify the governing differential law that determines how a system evolves on the landscape. In other words, they provide energy functions or learning algorithms, but not the PDE that connects local changes in semantic potential, sequence evolution, and manifold geometry. This missing governing law is the Stage 3 gap addressed in this paper.

3. The Geometric Stack: From Vocabulary to Ambient Space

The framework is built in two sequential steps. The first step operates at the vocabulary level and constructs an intrinsic semantic space for individual tokens. The second step operates at the sequence level and appends a temporal coordinate corresponding to token order. Together, these steps define the mathematical arena in which language can be modeled as a manifold and, eventually, as a field governed by a partial differential equation. The stack may be summarized as follows:

$$E \in \mathbb{R}^{N \times d} \longrightarrow v_k \in \mathbb{R}^r \longrightarrow (i, v(i)) \in \mathbb{R}^{r+1} \longrightarrow M \subset \mathbb{R}^{r+1} \longrightarrow \text{prompt as boundary conditions.}$$

The first arrow is a vocabulary-level reduction that removes redundant embedding coordinates; the second arrow is a sequence-level extension that adds time.

3.1. Step 1: The Token Semantic Space $\mathbb{R}^r$

Following standard embedding practice, a language model represents its vocabulary by an embedding matrix

$$E \in \mathbb{R}^{N \times d},$$

where N is the number of tokens and d is the model-imposed latent embedding dimension. The dimensionality of word embeddings has been studied theoretically by [Yin and Shen \(2018\)](#), who show that dimensionality selection can be analysed as a mathematical problem rather than treated as an arbitrary model parameter choice.

Each row of E represents one token as a vector

$$e_k \in \mathbb{R}^d, \quad k = 1, \dots, N.$$

The vocabulary embedding matrix therefore represents N discrete token embeddings as a finite point cloud in the continuous latent space $\mathbb{R}^d$:

$$\{e_k\}_{k=1}^N \subset \mathbb{R}^d.$$

Although each token embedding lives in $\mathbb{R}^d$, the finite set of vocabulary embeddings may span only an r -dimensional subspace of $\mathbb{R}^d$, where

$$r = \text{rank}(E), \quad r \leq \min(N, d).$$

Equivalently, $\text{span}\{e_1, \dots, e_N\} \cong \mathbb{R}^r$. Here r denotes the intrinsic linear semantic dimension of the vocabulary representation. This rank-based view is consistent with empirical work by [Kataiwa et al. \(2025\)](#), who find that token embedding spaces often have much lower intrinsic dimensionality than their extrinsic embedding dimensions. We therefore define the first ambient space as

$$\mathcal{A}_1 = \mathbb{R}^r.$$

3.2. Step 2: The Temporal-Semantic Ambient Space $\mathbb{R}^{r+1}$

If the intrinsic token semantic space $\mathbb{R}^r$ is interpreted as an r -dimensional spatial-like vector space, then a token sequence can be viewed as the evolution of a semantic state along an additional temporal dimension. The sequence index plays the role of time, while the r semantic coordinates play the role of spatial dimensions.

This discrete-to-continuous interpretation is consistent with the Neural ODE framework of [Chen et al. \(2018\)](#), who show that discrete neural-network transformations can be interpreted as an Euler discretisation of continuous dynamics. In their formulation, the hidden state evolves according to $dh/dt = f_\theta(h, t)$, rather than through a fixed finite sequence of layers. Although the present paper studies token trajectories on a semantic manifold rather than hidden-layer dynamics, the same mathematical idea motivates treating a discrete sequence index as a continuous temporal variable.

Let $i = 1, 2, \dots, n$ denote the token position in a sequence. At each position i , the token is represented by a semantic vector $v(i) \in \mathbb{R}^r$. Each contextualised token is therefore a pair $(i, v(i))$ lying in the product space $\{1, \dots, n\} \times \mathbb{R}^r$. In the continuum approximation the discrete index i may be treated as a temporal variable t , and the product space is approximated by the temporal-semantic ambient space

$$\mathcal{A}_2 = \mathbb{R}^{r+1}.$$

Thus the second ambient space is obtained by appending one temporal dimension to the intrinsic token semantic space: $\mathbb{R}^r \rightarrow \mathbb{R}^{r+1}$. We write $\mathcal{A} \equiv \mathcal{A}_2 = \mathbb{R}^{r+1}$ for the ambient space used throughout the remainder of the paper.

In this formulation, language can be understood as a set of discrete observed points or trajectories lying on, or near, a continuous manifold embedded in $\mathbb{R}^{r+1}$. The discrete tokens are the sampled observations, while the underlying language manifold represents the continuous geometric structure that constrains admissible linguistic sequences. We denote this manifold by

$$M \subset \mathbb{R}^{r+1}.$$

This manifold contains the admissible trajectories of language: sequences that are grammatically coherent, semantically meaningful, and pragmatically interpretable within a linguistic community. Language is not designed from first principles. It evolves through repeated human use, social selection, communicative efficiency, memory constraints, and cultural transmission. Over time, this evolutionary process causes admissible linguistic patterns to converge toward stable structures in the temporal-semantic ambient space. In the terminology of this paper, these stable structures define a basin of attraction, which we identify as the language manifold M . The manifold is not created by a large language model; rather, it is gradually shaped by the historical evolution of human language and preserved in the fossil record of human-generated texts.

The observed tokens and token sequences are discrete samples in $\mathbb{R}^{r+1}$. Collectively, they form a point cloud in the temporal-semantic space $\mathcal{A}$. The language manifold $M \subset \mathbb{R}^{r+1}$ is the continuous lower-dimensional structure fitted to, or inferred from, this point cloud. The role of the governing PDE is to provide a smooth constraint on this fit, so that admissible linguistic trajectories are not arbitrary curves through the ambient space, but paths lying on or near M .

It is useful to distinguish M itself from any single trajectory lying on it. The manifold M represents the entire admissible region, the full space of sequences a linguistic community would accept. A single generated token sequence, by contrast, is one particular curve traced through M . Borrowing terminology from physics, where a particle’s path through spacetime is called its worldline, we refer to such a curve as a *linguistic worldline*: the specific trajectory realized by one act of language production, as opposed to the manifold of all trajectories that could have been produced.

Section 4 turns to the governing law: what type of partial differential equation governs evolution, diffusion, propagation, or collapse on M ?

4. Governing PDE on the Language Manifold

When a manifold is embedded in a Euclidean or Riemannian space, it may be viewed as a lower-dimensional geometric structure within the ambient space. A flat plane or affine subspace can be described by linear equations, while a curved manifold often requires differential operators to characterise its geometry. In this sense, a PDE provides a general mathematical framework for representing, constraining, or recovering the shape of a manifold. In the language setting, observed tokens or token sequences can be understood as discrete sample points concentrated in a region of the temporal-semantic ambient space. A governing PDE may then be interpreted as a smooth, PDE-constrained fit to this cloud of discrete linguistic samples. Among possible candidates, the diffusion equation is especially useful because diffusion geometry shows that the Laplacian associated with heat flow can encode and recover the geometry of an underlying manifold from sampled data (Jones, 2024). Recent work on implicit manifold-valued diffusions further supports this view by constructing diffusion processes on data manifolds from point-cloud samples without requiring explicit charts or projections (Kawasaki-Borruat et al., 2026). Therefore, a diffusion-type equation provides a natural first candidate for modeling the language

manifold, while wave and transport equations may capture additional effects such as semantic propagation and directional flow.

4.1. Building the Diffusion Equation

Let $v \in \mathbb{R}^r$ denote the intrinsic semantic vector of a token. By adjoining the sequence coordinate t , the token is represented as a point

$$p = (v, t) \in M \subset \mathbb{R}^{r+1}$$

on the language manifold. Let $\Phi : M \rightarrow \mathbb{R}$ denote a scalar semantic potential, analogous to temperature T in heat transfer. At each point $p \in M$, the manifold gradient of Φ defines a tangent vector

$$\xi = \nabla_M \Phi(p) \in T_p M,$$

where $T_p M$ is the tangent space of the language manifold at p . Thus, v represents the token’s intrinsic semantic coordinate on the manifold, while the intrinsic gradient $\xi \in T_p M$ defines a local tangent direction field encoding the orientation and rate of steepest semantic increase. It characterizes the manifold’s local heading at p , not a literal chordal displacement between adjacent discrete tokens in the ambient space. Because $p = (v, t)$, the tangent space $T_p M$ is itself $(r + 1)$ -dimensional, so ξ in general has both a semantic component and a sequence-coordinate component. Writing $\pi_{\text{sem}}(v, t) = v$ for the projection onto the intrinsic semantic coordinates, we denote this semantic component $\xi_{\text{sem}} = \pi_{\text{sem}}(\xi) = \pi_{\text{sem}}(\nabla_M \Phi)$. It is this projected vector, not the full tangent vector ξ , whose negative gives the worldline’s velocity in the gradient-flow equation below.

This setup establishes a dichotomy between the field-wide and trajectory-wise characterizations of language. The ambient space $\mathbb{R}^{r+1}$ and the language manifold $M \subset \mathbb{R}^{r+1}$ are the fixed geometric reference domain; they are not themselves Eulerian or Lagrangian, but the arena in which both descriptions are stated. Within that arena, the scalar potential Φ and its gradient $\nabla_M \Phi$ are Eulerian objects: fields defined over the entire manifold M at once, governed by the PDE developed below. A linguistic worldline (Section 3.2), by contrast, is a Lagrangian object, a single moving point

$$\gamma(s) = p(s) = (v(s), t(s)) \in M,$$

not a field. Under the natural parametrization $t(s) = s$, the worldline reduces to $\gamma(s) = (v(s), s)$, and its semantic component evolves by the projected gradient-flow equation

$$\frac{dv}{ds} = -\xi_{\text{sem}}(\gamma(s)) = -\pi_{\text{sem}}(\nabla_M \Phi(\gamma(s))).$$

Only the semantic component of ξ drives v ; the sequence coordinate advances independently, at the fixed unit rate fixed by the parametrization $t(s) = s$, rather than being driven by the t -component of ξ . In this view, language modeling decomposes into two sequential steps: first solve the field-wide problem to recover Φ over M (training, Section 5.1); second, given a fixed point $v(0)$, integrate the gradient-flow equation forward to obtain the worldline (inference, Section 5.2). Only the worldline, not the entire solved field, corresponds to the observable generated sequence.

This separation also clarifies a potential objection: if Φ is solved over the whole manifold M , including positions already realized by earlier tokens, in what sense is the field dynamic during generation? It is not. Under this formulation, Φ is solved once, as a boundary- or initial-value problem over the whole of M , and does not change as generation proceeds; sequence position is internalized as a coordinate of M rather than as a clock driving a time-varying field. All of the dynamics visible during text generation are Lagrangian: they belong entirely to the worldline $\gamma(s)$ as it moves through the already-fixed field Φ , not to Φ itself.

The term “worldline” is borrowed from physics by analogy only. The language manifold is equipped with a Riemannian, positive-definite metric, not a Lorentzian one, and the framework introduces no causal light-cone structure or propagation-speed constant. Where a bound on the rate of semantic change is needed, it is more naturally stated as a constraint on the gradient norm, $\|\xi\| \leq \xi_{\max}$, than as a literal speed limit.

In the heat-transfer analogy, the rate of change of the scalar potential in the time domain is proportional to the Laplacian of that potential in the spatial domain. Therefore, in the local flat approximation, the semantic diffusion equation is

$$\frac{\partial \Phi}{\partial t} = \kappa \nabla^2 \Phi,$$

where $\kappa > 0$ is the semantic diffusion coefficient. In the heat-transfer analogy, κ corresponds to the thermal diffusivity of the medium, which controls how rapidly temperature changes in time in response to spatial temperature curvature. Here, the “time domain” corresponds to the token sequence dimension, while the “spatial domain” corresponds to the intrinsic semantic directions on the language manifold. Consistent with Φ being solved once, field-wide, $\partial \Phi / \partial t$ denotes the derivative of that field with respect to the sequence coordinate t , not evolution in any external physical time; t is a coordinate of M , not a clock. Throughout this subsection, ∇ , ∇^2 , ∇_M , ∇_M^2 , K , and div_M act only on the intrinsic semantic directions, with t held as the separate coordinate against which these spatial operators evolve; none of them differentiate with respect to t itself.

This steady-state reading is sharpened by distinguishing t from a second, external parameter τ , the actual generation/evolution time at which the field-solving step (Section 5.1) is regarded as complete and frozen. With this distinction, the steady-state condition is

$$\frac{\partial \Phi(p, \tau)}{\partial \tau} = 0 \quad \text{for every fixed } p \in M,$$

i.e. once solved, Φ does not change at any manifold point as generation proceeds. This is compatible with Φ being spatially nonuniform over M , including along the sequence coordinate t that is already built into $p = (v, t)$: in general $\Phi(p_1) \neq \Phi(p_2)$ for $p_1 \neq p_2$, and $\partial \Phi / \partial t$, the variation of the solved field along the sequence coordinate, need not vanish. The two statements address different questions and do not conflict: $\partial \Phi / \partial \tau = 0$ says the field itself is fixed once solved, while $\partial \Phi / \partial t \neq 0$ says that fixed field is not flat along the sequence direction. A worldline accordingly experiences a changing potential as it moves, purely because it visits different points of an already-fixed, spatially varying field,

$$\frac{d}{ds} \Phi(\gamma(s)) = g(\nabla_M \Phi(\gamma(s)), \dot{\gamma}(s)),$$

which is generically nonzero even though $\partial \Phi / \partial \tau = 0$ throughout. In short: the field is steady in the external sense (τ -invariant) but not uniform in the spatial/sequence sense (t -varying); the worldline’s experience of change is entirely Lagrangian, arising from its motion through that fixed-but-nonuniform field, consistent with the Eulerian/Lagrangian decomposition introduced above.

The ordinary Euclidean Laplacian ∇^2 should be understood as a local flat approximation. If the language manifold is globally curved, ∇^2 should be replaced by the Laplace–Beltrami operator ∇_M^2 . In a standard three-dimensional heat-transfer problem, the Euclidean Laplacian ∇^2 is adequate because heat diffuses through a Euclidean domain. However, if heat transfer is constrained to a curved thin shell or surface, the diffusion domain is itself a curved manifold, and the Laplace–Beltrami operator is required to account for curvature. By analogy, language dynamics on a curved language manifold should be written as

$$\frac{\partial \Phi}{\partial t} = \kappa \nabla_M^2 \Phi.$$

Here ∇_M^2 incorporates the Riemannian metric of the language manifold. In local coordinates, this introduces metric-dependent terms, equivalently expressed through covariant derivatives or Christoffel symbols.

In the language manifold, semantic diffusion is unlikely to be isotropic. The intrinsic semantic directions are not independent, and activation does not spread equally in every direction. Some dimensions may be strongly coupled, such as subject, tense, agency, modality, causality, and discourse role, while others may interact only weakly. The scalar diffusion coefficient κ should therefore be replaced by a semantic conductivity tensor K . In the local flat approximation the anisotropic diffusion equation becomes

$$\frac{\partial \Phi}{\partial t} = \nabla \cdot (K \nabla \Phi).$$

The diagonal entries of K describe diffusion along individual semantic directions; the off-diagonal entries describe coupling between different semantic dimensions. If both curvature and anisotropy are considered, the most general diffusion form becomes

$$\frac{\partial \Phi}{\partial t} = \text{div}_M(K \nabla_M \Phi),$$

which is the natural generalization of heat diffusion to an anisotropic semantic field on a curved language manifold.

4.2. General PDE Forms for Geometry Fitting

The diffusion equation provides a useful geometric fitting tool because it connects time-domain evolution with spatial-domain curvature and attraction toward stable structures. However, language dynamics are unlikely to be purely diffusive, like water cascading downhill. Language does not merely smooth or dissipate semantic activation. It also preserves structure, exhibits resonance, propagates long-range dependencies, and moves directionally under contextual constraints.

For this reason, the wave equation may be more consistent with certain features of the language manifold, especially where semantic structures persist and propagate across long sequences. Similarly, transport equations may be needed to describe directional semantic movement under the influence of context. Therefore, rather than relying on a single diffusion equation, this paper considers a broader family of PDE-based geometric fitting models.

The common feature of this PDE family is that each equation connects evolution in the time domain with structure in the spatial domain. In the language setting, the time domain corresponds to the token sequence dimension, while the spatial domain corresponds to intrinsic semantic directions on the language manifold. These equations therefore provide different ways to represent token flow following the gradient structure of semantic potential.

4.3. Wave Equation: Semantic Propagation and Structure Preservation

To model the preservation and propagation of linguistic structure, we introduce the wave equation. In the flat approximation:

$$\frac{\partial^2 \Phi}{\partial t^2} = c^2 \nabla^2 \Phi.$$

On the curved language manifold, this becomes

$$\frac{\partial^2 \Phi}{\partial t^2} = c^2 \nabla_M^2 \Phi,$$

where c represents the propagation velocity of semantic structure.

A more general form includes a damping term, which provides additional flexibility for fitting semantic dynamics:

$$\frac{\partial^2 \Phi}{\partial t^2} + 2\beta \frac{\partial \Phi}{\partial t} = c^2 \nabla_M^2 \Phi,$$

where $\beta > 0$ is the damping coefficient. This term allows temporary semantic effects to decay while stable discourse structures continue to propagate. Thus, the wave equation may fit the part of language dynamics involving semantic memory, structure preservation, and long-range dependency.

4.4. Transport Equation: Directional Semantic Flow

A third candidate is the transport equation. In the flat approximation,

$$\frac{\partial \Phi}{\partial t} + u \cdot \nabla \Phi = 0,$$

where u is a semantic velocity field tangent to the language manifold. The vector field u is different from both the intrinsic semantic vector v and the gradient tangent vector $\xi = \nabla_M \Phi$.

While v represents the intrinsic semantic coordinate of a token and ξ the local direction and rate of steepest semantic change induced by the potential Φ , u represents the direction and speed of semantic transport along the manifold. This distinction gives the model additional flexibility: token flow does not have to follow only the gradient of the token’s intrinsic attributes. It may also be guided by contextual direction, discourse intention, or prompt-imposed constraints. The gradient-flow equation of Section 4.1 is the special case $u = -\xi_{\text{sem}}$, the semantic projection of ξ introduced there, in which transport follows the potential’s own gradient rather than an independently specified direction.

On a curved manifold, the transport equation becomes

$$\frac{\partial \Phi}{\partial t} + u \cdot \nabla_M \Phi = 0,$$

where u lies on the tangent space of M and $\nabla_M \Phi$ is the intrinsic gradient on the manifold.

4.5. Toward a General Governing PDE

The three PDE forms can be combined into a more general governing template:

$$\frac{\partial^2 \Phi}{\partial t^2} + 2\beta \frac{\partial \Phi}{\partial t} + u(t) \cdot \nabla_M \Phi = \text{div}_M(K \nabla_M \Phi) + S(t). \quad (1)$$

In this equation:

- $\partial^2 \Phi / \partial t^2$ represents wave-like propagation,
- $2\beta \partial \Phi / \partial t$ represents damping or dissipation,
- $u(t) \cdot \nabla_M \Phi$ represents transport along a semantic flow field,
- $\text{div}_M(K \nabla_M \Phi)$ represents anisotropic diffusion on the curved language manifold, and
- $S(t)$ represents source terms introduced by new tokens, external context, or multimodal input.

This general PDE is not proposed as the final equation of language. Rather, it is a governing-law template: it shows how language dynamics may combine smoothing, propagation, and directional flow on the language manifold.

5. Training and Prompting as Inverse and Forward PDE Problems

5.1. Training as an Inverse PDE Problem

The governing PDE of the language manifold is not known in advance. Neither the exact form of the equation nor its coefficient fields are directly given. Therefore, LLM training may be interpreted as an inverse PDE problem. In the terms of Section 4.1, training solves the Eulerian field problem, recovering Φ over the whole manifold M , rather than tracing any single worldline.

The fossil record of human-generated language provides discrete samples of trajectories lying on, or near, the language manifold M . Language is not constructed from an architectural blueprint. It evolves through repeated human use, social selection, communicative efficiency, memory constraints, and cultural transmission. Over time, admissible linguistic patterns converge toward stable structures in the temporal-semantic ambient space. In the terminology of this paper, these stable structures define an attractor basin, which we identify as the language manifold M .

The language manifold is embedded in the second ambient space, $\mathcal{A} \equiv \mathcal{A}_2 = \mathbb{R}^{r+1}$, which is a linear temporal-semantic space. From this perspective, LLM training is not the construction of language from raw data, but the search for, and approximation of, a pre-existing language manifold within $\mathcal{A}$. The learned model estimates the geometry of M from discrete linguistic samples.

Training also estimates the scalar potential structure Φ and the coefficient fields of the governing PDE, including the semantic conductivity tensor $K(t)$, the transport vector field $u(t)$, the damping coefficient β , and the source term $S(t)$. These quantities are not prescribed in advance. They are inferred from the observed distribution of human-generated language.

In this interpretation, model weights are not merely memorised statistical associations. They are distributed approximations of the geometry and governing dynamics of the language manifold. The model learns how semantic activation diffuses, propagates, and transports across M under different contextual and boundary conditions.

5.2. Prompting as Boundary Condition, Slicing, and Collapse

Inference is the corresponding forward problem. Given a prompt P , the model imposes boundary or initial conditions on the language manifold M . These constraints produce a prompt-conditioned scalar potential field Φ_P . In the terms of Section 4.1, inference is the Lagrangian step: given the already-solved field Φ_P and a fixed initial point, it integrates the gradient-flow equation forward to trace the resulting worldline.

The prompt is therefore not merely an input string. It plays a central role in reducing the admissible region of the language manifold. In the temporal-semantic formulation, the ambient space $\mathcal{A} = \mathbb{R}^{r+1}$ is a linear coordinate space, while the language manifold $M \subset \mathbb{R}^{r+1}$ is a lower-dimensional Riemannian manifold embedded within it. The prompt supplies a finite set of token positions and semantic states, and therefore fixes the part of the trajectory that any admissible continuation must satisfy.

This reduction can be understood in two complementary ways. From a linear-algebra perspective, the prompt imposes a system of constraints on the full temporal-semantic ambient space. The query-key-value mechanism supplies contextual relationships that restrict the admissible degrees of freedom, similar to using equations to reduce the dimensionality of a solution space. From a PDE perspective, the prompt supplies boundary or initial conditions for the governing equation. The model response is then the continuation trajectory selected under these constraints.

In transformer architectures, this constraint is operationalised through the query-key-value attention mechanism. The prompt tokens generate queries, keys, and values that determine

which prior semantic states are relevant to the next state. Geometrically, this process can be interpreted in three related ways. First, the prompt imposes boundary values by fixing known points on the manifold. Second, it slices the manifold by assigning values to certain contextual or semantic dimensions, thereby reducing the admissible solution set to a lower-dimensional submanifold. Third, through attention-weighted aggregation, it may collapse or average information across selected token positions, producing a lower-dimensional effective representation of the context.

Thus, prompting is a geometric operation on the language manifold. By imposing boundary conditions, fixing parameters, slicing admissible dimensions, and aggregating relevant information, the prompt progressively reduces the full language manifold to a constrained admissible region. The output of a language model can therefore be interpreted as a trajectory selected from this reduced region of M . The response is a continuation on the language manifold under constraints supplied by the prompt.

The admissible continuation may also be interpreted as a stable or low-energy path on the manifold, analogous to a physical system evolving toward an attractor or minimum-free-energy configuration.

The relationship between training and inference can therefore be summarized as follows. Training estimates the geometry of M , the scalar potential structure Φ , and the coefficient fields governing semantic diffusion, propagation, and transport. Prompting then imposes boundary or initial conditions on this learned structure, reducing the admissible region of the manifold. The response is the continuation trajectory selected from this constrained region.

6. Discussion, Limitations, and Future Research

The framework proposed in this paper is supported by several related lines of existing work.

First, the reduction from the model-imposed latent dimension d to the intrinsic semantic rank r is motivated by recent work on the intrinsic dimensionality of token embeddings. [Kataiwa et al. \(2025\)](#) suggest that token embeddings contain substantial redundancy and may lie in a lower-dimensional intrinsic representation space. This supports the first step of the present framework: replacing the full embedding space $\mathbb{R}^d$ with an intrinsic token semantic space $\mathbb{R}^r$, where r captures the effective number of independent semantic directions.

Second, diffusion geometry provides a mathematical foundation for fitting continuous geometric structures to discrete point-cloud samples. [Jones \(2024\)](#) shows that diffusion operators, Laplacians, and carré-du-champ structures can recover geometric information from sampled data. [Kawasaki-Borruat et al. \(2026\)](#) further develop diffusion processes on implicit manifolds, supporting the idea that a manifold can be inferred from point-cloud data through diffusion-based operators. This directly supports the interpretation of observed token sequences as discrete samples lying on, or near, a continuous language manifold M .

Third, Neural ODEs provide a precedent for introducing a time dimension into neural computation. [Chen et al. \(2018\)](#) interpret discrete neural-network updates as an Euler discretisation of continuous dynamics and replace a fixed sequence of hidden layers with an ordinary differential equation for the hidden state. This supports the second step of the present framework: treating the discrete token index as a continuous pseudo-time variable t . The time dimension is the missing piece that connects the semantic spatial domain $\mathbb{R}^r$ with sequence evolution, producing the temporal-semantic ambient space $\mathbb{R}^{r+1}$. Once this space is constructed, PDEs become natural candidates for governing the relationship between time-domain evolution and spatial-domain semantic structure.

Fourth, work on data-driven discovery of intrinsic dynamics provides a useful precedent for

recovering lower-dimensional dynamics from high-dimensional observations (Floryan and Graham, 2022). This is relevant to the present framework because the assumption of a continuous language manifold should ultimately be treated as an approximation rather than as a completed geometric characterization. In high-dimensional embedding space, lower-dimensional language structure may appear as a smooth manifold, but it may also appear as a cluster structure, graph-like structure, stratified space, singular manifold-like set, or other non-smooth geometric object. Therefore, the language manifold proposed in this paper should be understood as a first smooth approximation to the underlying semantic structure.

Fifth, the proposed language-manifold framework is consistent with a broader two-ambient-space approach to representation learning. In a related formulation for image-based world models, Zhang (2026) introduces a two-ambient-space continuum reformulation of I-JEPA, in which the image-plane chart domain is connected with the patch-content domain through an advection-diffusion PDE. The present paper applies the same general principle to LLMs: an intrinsic semantic ambient space is first constructed and then extended by a pseudo-time coordinate, allowing PDE-based tools to connect sequence evolution with semantic variation.

In the present framework, the term “initial state” has two related but distinct meanings. First, the pretrained language manifold may be regarded as an unactivated background semantic landscape, denoted M_0 , whose learned geometry already contains latent gradients and admissible directions of semantic evolution. Second, once a prompt or first token is introduced, it defines an activated initial point or state $q_0 \in M_0$. The subsequent states then evolve from q_0 along the learned semantic landscape, forming a trajectory whose observable trace corresponds to the generated language output. This trajectory is precisely the linguistic worldline of Section 3.2, obtained by integrating the gradient-flow equation of Section 4.1 forward from q_0 through the field Φ already fixed by training on M_0 .

These lines of work correspond to the major steps of this paper:

$$\begin{aligned} \text{intrinsic rank reduction} &\longrightarrow \mathbb{R}^r, \\ \text{addition of pseudo-time dimension} &\longrightarrow \mathbb{R}^{r+1}, \\ \text{diffusion/PDE fitting} &\longrightarrow M \subset \mathbb{R}^{r+1}. \end{aligned}$$

The current proposal should be understood as a Stage 3 framework rather than a completed predictive theory. Future work may further develop this framework toward Stage 4 by providing empirical validation, quantitative operators, and more rigorous geometric formulations. Several open problems remain: estimating the intrinsic semantic rank r ; recovering the semantic potential Φ and its gradient field $\xi = \nabla_M \Phi$ empirically from data; replacing the local flat operator ∇^2 with the manifold operator ∇_M^2 ; identifying which PDE terms are required for language dynamics; and recovering coefficient fields such as K , u , β , and S from trained model activations.

A further direction is to develop an atlas-based formulation of language geometry. Instead of assuming one globally smooth language manifold, the global semantic structure may be covered by local charts or manifold-like patches M_α , each representing a local semantic, syntactic, pragmatic, or domain-specific region. Transition maps and compatibility conditions between neighboring charts would then be needed to determine whether these local structures form a smooth manifold, a stratified manifold, or a more general geometric object.

Another direction is to connect manifold identification with a fibre-bundle-like factorization of the embedding space. Identifying a lower-dimensional semantic structure and constructing such a factorization can be viewed as two complementary forms of dimensional reduction. The former seeks a lower-dimensional semantic substructure within the high-dimensional embedding space, while the latter decomposes the total embedding space into a task-relevant semantic base and auxiliary fibres that may represent style, context, modality, domain, or redundant degrees of

freedom. The projection from the total embedding space to a lower-dimensional base space can be interpreted as eliminating, collapsing, or marginalizing variations along the fibre. However, this separation is task-dependent: a dimension that is irrelevant for one task may be important for another. Therefore, a rigorous formulation would require a defined projection map or a learned procedure for determining which variations belong to the semantic base and which belong to the auxiliary fibres.

Finally, the rigorous formulation of PDE operators on the probability simplex is left for future study. Since LLM outputs are probability distributions over vocabulary tokens, a complete theory should eventually connect the temporal-semantic manifold formulation with operators defined on, or coupled to, the probability simplex. This would clarify how semantic potential fields, token probabilities, and manifold dynamics interact during inference.

Although the present paper focuses on LLMs, the framework may provide a bridge from language modeling to real-world AI. LLMs are the most accessible case because token embeddings and sequence positions naturally define a temporal-semantic ambient space. In industry-specific or discipline-specific AI, the same idea may be extended by replacing the language manifold with a domain manifold built from engineering, financial, legal, biological, or operational data. In this broader view, LLMs are not the endpoint, but the first workable example of a general framework in which intelligence is modeled as manifold learning plus governing dynamics. This provides a possible path from LLM transparency toward real-world AI, industry-specific AI, discipline-specific AI, and eventually AGI.

7. Conclusion

This paper has proposed a Stage 3 theoretical framework for understanding large language models through the geometry of language manifolds and the dynamics of partial differential equations. The two-step construction, from vocabulary embeddings to an intrinsic semantic space $\mathbb{R}^r$, and from there to a temporal-semantic ambient space $\mathbb{R}^{r+1}$, provides a principled arena in which language can be modeled as a continuous manifold fitted to discrete token trajectories. Training is interpreted as the inverse problem of recovering this manifold and its governing PDE from human-generated text; inference is interpreted as the corresponding forward problem, in which a prompt imposes boundary conditions and selects an admissible continuation trajectory.

The framework does not claim to be a completed quantitative theory. Rather, it offers a Stage 3 conceptual bridge between the empirical scaling laws of Stage 2 and the predictive, first-principles formalism of Stage 4. The open problems identified in the Discussion, intrinsic rank estimation, empirical recovery of the potential Φ and its gradient field $\xi = \nabla_M \Phi$, atlas-based geometry, fibre-bundle factorization, and PDE coefficient recovery from model activations, mark the path forward toward a fully quantitative theory of language dynamics.

References

- J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554–2558, 1982.
- G. E. Hinton and T. J. Sejnowski. Learning and relearning in Boltzmann machines. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing*, volume 1, chapter 7, pages 282–317. MIT Press, 1986.
- Y. LeCun, S. Chopra, R. Hadsell, M. A. Ranzato, and F. J. Huang. A tutorial on energy-based learning. In G. Bakir, T. Hofmann, B. Schölkopf, A. Smola, B. Taskar, and S. Vishwanathan, editors, *Predicting Structured Data*. MIT Press, 2006.

- W. Yin and Y. Shen. On the dimensionality of word embedding. In *Advances in Neural Information Processing Systems*, volume 31, 2018.
- H. Kataiwa, K. Cho, and R. Ohki. On the intrinsic dimensionality of token embeddings in large language models: redundancy, rank, and the geometry of representation spaces. Preprint, 2025.
- M. M. Bronstein, J. Bruna, T. Cohen, and P. Veličković. Geometric deep learning: grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021.
- T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, volume 31, 2018.
- V. Kawasaki-Borruat, C. Grotehans, P. Vandergheynst, and A. Gosztolai. Diffusion processes on implicit manifolds. *arXiv preprint arXiv:2604.07213*, 2026.
- I. Jones. Diffusion geometry. *arXiv preprint arXiv:2405.10858*, 2024.
- D. Floryan and M. D. Graham. Data-driven discovery of intrinsic dynamics. *arXiv preprint arXiv:2108.05928*, 2022.
- L. Zhang. A two-ambient-space continuum reformulation of I-JEPA: spatial-patch geometry, scalar visual potentials, and PDE-guided manifold fitting. Preprint, Chicago Booth 2026 Workshop on World Models, 2026. <https://doi.org/10.5281/zenodo.20414034>.

Explainable Reasoning and Learning in Ad Hoc Teamwork

Hasra Dodampegama and Mohan Sridharan¹

Abstract. An assistive AI agent often has to collaborate with previously unseen agents and humans. Methods considered state of the art for such *ad hoc teamwork* use a large labeled dataset of prior observations to model the behavior of other agents and to determine the ad hoc agent’s behavior. These approaches are resource-hungry, and do not support rapid incremental revisions or transparency, with the necessary resources (e.g., training examples, computation) not readily available in practical domains. Our architecture for ad hoc teamwork embeds the principles of refinement, ecological rationality, interactive learning, and explainable agency, leveraging the complementary strengths of knowledge-based and data-driven methods. For any given goal, an ad hoc agent determines its actions through non-monotonic logical reasoning with: (a) prior domain-specific commonsense knowledge; (b) models learned rapidly to predict the behavior of other agents; and (c) anticipated abstract future tasks based on generic knowledge of similar situations. Further, the agent generates relational descriptions as explanations of its decisions and those of other agents. We evaluate the capabilities of our architecture in *VirtualHome*, a realistic 3D simulation environment.

1 Introduction

Consider an AI agent performing household tasks in collaboration with other agents (AI, human) it has not worked with before. Figure 1 shows snapshots of this motivating scenario in a multiagent simulation environment in which an AI agent (blue shirt) and a human (green top) are preparing breakfast and setting up a workstation. The AI agent has to reason with different descriptions of prior knowledge and uncertainty in the form of qualitative statements (“eggs are usually in the fridge”) and quantitative statements (“I am 90% sure I saw the eggs on the table”). The agents have a limited view of the environment and do not communicate with each other, although each of them is aware of the domain state and action outcomes, e.g., the location of teammates and objects moved by a teammate. Furthermore, the human may want to query and understand the AI agent’s decisions. These characteristics correspond to *Ad hoc Teamwork* (AHT), requiring cooperation “on the fly” without prior coordination [44]. The AHT problem arises in many practical applications such as disaster rescue, exploration, and collaborative games, and poses challenges in knowledge representation, reasoning, and learning [27].

The state of the art in AHT has moved from using predetermined policies for selecting actions in specific states to methods with a key *data-driven* component that uses a long history of prior experiences to build probabilistic or deep network methods that model the behavior of other agents (or agent types) and optimize the behavior of

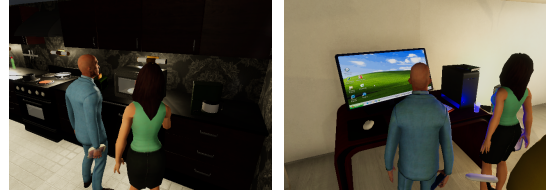


Figure 1. Screenshots from *VirtualHome* [31] showing a human (female in green top) and an assistive AI agent (male in blue shirt) collaborating.

the ad hoc agent [27]. However, it is difficult to gather training samples of different situations in practical domains. Also, these methods are usually opaque, offering limited insight into how or why an agent made a decision. This lack of transparency makes it challenging to fix faults and to foster trust in automated decision-making. In a departure from existing work, our AHT architecture leverages the complementary strengths of knowledge-based and data-driven methods for reasoning and learning to jointly address the underlying challenges. It embeds fundamental principles such as refinement, ecological rationality, interactive learning, and explainable agency, which can be traced back to the early pioneers of AI [40], enabling each ad hoc agent in a team to:

1. Adapt scalably through non-monotonic logical reasoning with prior commonsense domain knowledge and rapidly-learned predictive models of other agents’ behavior;
2. Leverage a Large Language Model (LLM) to anticipate future tasks to be completed, adapting the LLM’s output based on domain-specific knowledge and experience; and
3. Automatically construct on-demand relational descriptions of its decisions and those of the others as *explanations* in response to different types of questions

We have published subsets of these capabilities in other papers, starting with a proof of concept architecture for reasoning [8] and extending it to support simple explanation generation [9] and scalability [10]. Here, we highlight the benefits of leveraging the interplay between reasoning and learning, particularly in the context of explanation generation. We evaluate the architecture’s capabilities in *VirtualHome*, a realistic 3D simulation environment [31], with simulated agents collaborating to perform household tasks.

2 Related Work

Researchers have explored AHT for more than two decades under different names [27]. Initial work used predefined protocols or plays

¹ Institute of Perception, Action and Behavior, School of Informatics, University of Edinburgh, UK, email: hasra.dodampegama@ed.ac.uk, m.sridharan@ed.ac.uk

that encoded specific actions for the ad hoc agent in specific scenarios [6]. Subsequent work used probabilistic and sampling-based methods such as Upper Confidence bounds for Trees to determine the ad hoc agent’s actions [5, 34]. Recent approaches considered state of the art for AHT have posed it primarily as a learning problem; the key component predicts behavior of other agents and determines the ad hoc agent’s behavior using a history of prior interactions with similar agents or agent types. This includes the use of Fitted Q Iteration to learn action selection policies from offline data of each teammate type [4]; attention-based recurrent neural networks for real-time adaptation [7]; graph neural networks to simulate interactions with other agents (agent types) and determine the ad hoc agent’s behavior [32]; self-play to learn a cooperation policy and candidate teammate policies [13]; sequential and hierarchical variational auto encoders to model teammates’ changing behaviors [46]; and model-based RL methods to learn separate models of the environment and teammates [45]. Also, an LLM-based framework has been developed to compute the ad hoc agent’s actions [25].

While the frameworks summarized above offer promising strategies for modeling the behavior of teammates and selecting the actions of the ad hoc agent, they are resource-hungry, requiring substantial time, computation, and training examples to achieve the observed performance. Such resources are not readily available in practical AHT domains, posing questions about the scalability and usability of these methods. Also, they struggle to leverage the commonsense knowledge available in many practical domains, and their internal decision-making mechanisms are often opaque, limiting the ability to understand, justify, and audit the agents’ automated decisions.

Embodied AI refers to AI systems operating within physically realistic (simulation) environments such as Habitat [37] and Virtual-Home [31]. These interactive platforms support the generation of complex scenarios for evaluating algorithms for single-agent and multi-agent collaboration problems. With the increasing use of embodied AI agents and AI methods in different applications, there is renewed focus on transparency and explainability of decision-making [26, 36], although this has been a well-researched area for decades [33, 29]. With such AI methods being used to interact with other agents and humans, multiple theories and approaches have been developed and adapted to define and use causal relationships to provide explanations [20], categorize methods in terms of their ability to extract causal relationships and provide explanation [30], and consider the recipient’s prior knowledge while generating user-centric explanations [12, 14]. Instead of assuming comprehensive prior knowledge or exploring black box models, we seek to design architectures that incorporate interpretable models [35] and automatically support process-level explanations based on knowledge that is evolving over time. To achieve this objective, work in our group has designed architectures that embed the principle of *explainable agency* [23, 24] and build on a theory for explanation generation [43], enabling an agent to construct relational descriptions as explanations in response to different types of questions [28]. The novelty is in the processes used to achieve the desired functional capabilities. In this paper, we extend these ideas and leverage the principle of ecological rationality [18] to provide explanations in the context of multiagent (ad hoc) collaboration.

3 Architecture

Example Domain 1 [Example Embodied AI Agent Domain]

Consider an embodied AI agent and a human agent collaborating to complete household tasks; Figure 1 shows snapshots while preparing

breakfast [31]. The agents can interact with the environment through high-level actions, e.g., move to places, pick up or place objects, switch appliances on or off, and open or close appliances. Completing a task requires a sequence of such actions to be computed and executed by the agents who do not communicate directly with each other. Also, the agent assumes that the other agents have access to the same state information and will make (what it considers as) rational decisions and computes its plan to complete the current task and prepare for the upcoming task(s). Prior commonsense knowledge of the embodied agent includes relational descriptions of some attributes of the domain, ad hoc agent, and the human agent (Section 3.1); a learned (or encoded) graph of information about likely locations of objects in the domain; and default statements that hold in all but a few exceptional circumstances. The knowledge also includes some axioms governing actions and change, e.g., the agent or human can only pick one object at a time, and certain food items require preparation prior to consumption.

3.1 Knowledge Representation and Reasoning

In our architecture, the transition diagram of any given domain is described using an extension of action language $\mathcal{AL}_d$ [16]. Action languages are formal models of parts of natural language for describing transition diagrams of dynamic systems. The domain representation comprises a system description $\mathcal{D}$, a collection of statements of $\mathcal{AL}_d$, and a history $\mathcal{H}$. $\mathcal{D}$ has a sorted signature Σ with basic sorts, and domain attributes (statics, fluents) and actions described in terms of these sorts. Basic sorts in our example scenario include *object*, *appliance*, *ad_hoc_agent*, *human*, and *step* (for temporal reasoning), and are arranged hierarchically, e.g., *apple* is a sub-sort of *food*, a sub-sort of *graspable*, a sub-sort of *object*. Actions can be *agent_actions* or *exo_actions* (exogenous actions). The *agent_actions* such as *grab(ad_hoc_agent, object)*, *switch_on(ad_hoc_agent, appliance)*, *move(ad_hoc_agent, location1, location2)* are performed by the ad hoc agent. The *exo_actions* such as *exo_grab(other_agent, object)*, *exo_switch_on(other_agent, appliance)* are performed by other agents, e.g., human or another AI agent. Statics are domain attributes whose values cannot change. Fluents are attributes whose values can change; they can be *inertial*, which obey inertia laws and are changed by the ad hoc agent’s actions, e.g., *at(ad_hoc_agent, location)* is the ad hoc agent’s location, or *defined*, which do not obey inertia laws and are not directly changed by the ad hoc agent’s actions, e.g., *agent_at(other_agent, location)* is a teammate’s location computed by (say) external sensors.

Given Σ , the domain’s dynamics are described in $\mathcal{D}$ using three types of axioms: *causal law*, *state constraint*, and *executability condition* such as:

$$\text{grab}(A, O) \text{ causes } \text{in_hand}(A, O) \quad (1a)$$

$$\text{heated}(F) \text{ if } \text{on}(F, E), \text{switched_on}(E) \quad (1b)$$

$$\text{impossible } \text{grab}(A, O) \text{ if } \text{on}(O, E), \text{not_opened}(E) \quad (1c)$$

where Statement 1(a), a causal law, implies that grabbing an object causes it to be in the hand of the ad hoc agent; Statement 1(b), a state constraint, implies that a food item placed in an electrical appliance (e.g., microwave) that is switched on gets heated; and Statement 1(c), an executability condition, prevents the ad hoc agent from trying to grab an object from an appliance with a closed door. History $\mathcal{H}$ is a record of observations of the form *obs(fluent, boolean, step)*, and action executions of the form *hpd(action, step)* at specific time

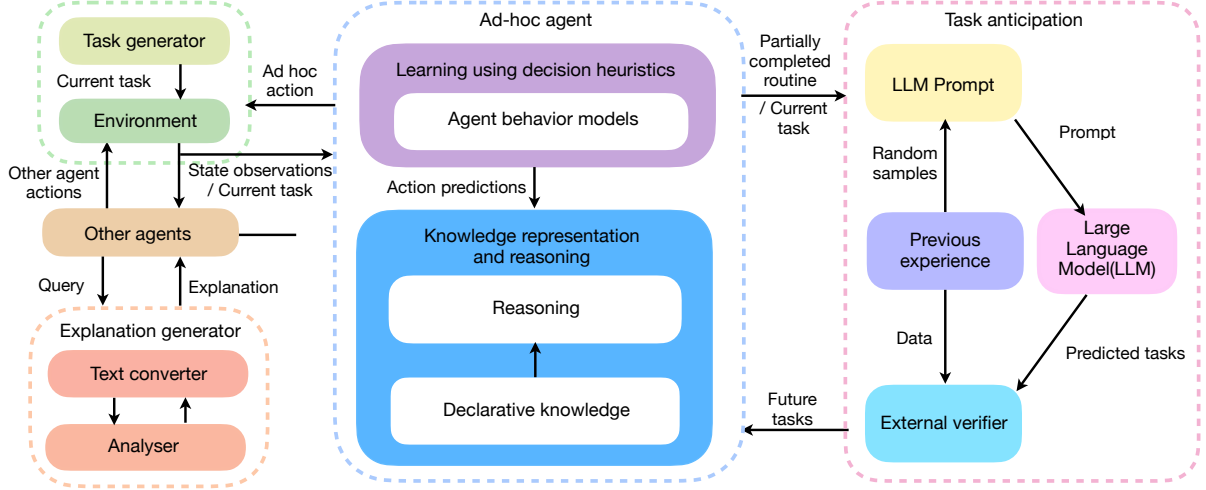


Figure 2. Our architecture leverage the complementary strengths of knowledge-based and data-driven reasoning and learning.

steps. It also includes default statements that are true in all but a few exceptional circumstances, e.g., books are usually in the library but cookbooks are in the kitchen.

To reason with knowledge, our architecture uses a script to automatically construct program $\Pi(\mathcal{D}, \mathcal{H})$ by translating the system description $\mathcal{D}$ and history $\mathcal{H}$ to CR-Prolog [3], an extension to ASP that supports consistency restoring (CR) rules. $\Pi(\mathcal{D}, \mathcal{H})$ includes statements from $\mathcal{D}$ and $\mathcal{H}$, inertia axioms, reality check axioms, closed world assumptions for defined fluents and actions, helper relations, e.g., $holds(fluent, step)$ and $occurs(action, step)$ to imply that a fluent is true and an action is part of a plan at a time step, and helper axioms that define goals and guide planning and diagnosis. ASP encodes *default negation* and *epistemic disjunction*, and supports non-monotonic logic reasoning. This ability to revise previously held conclusions is essential for agents reasoning and acting in practical domains based on incomplete knowledge and noisy observations. The CR rules allow the agent to make assumptions (e.g., that a default statement does not hold) under exceptional circumstances to recover from inconsistencies. All reasoning tasks (i.e., planning, diagnostics, and inference) are then reduced to computing *answer sets* of Π . We use the SPARC system [2] to solve CR-Prolog programs. Example programs and results are in our open source repository [11].

Our current application domain is substantially more complex than those in our prior work, with many more objects and actions; the corresponding domain description is more complex and the tasks require much longer plans. To enable the use of the KR formalism (above), our architecture incorporates new strategies to constrain the search space. Specifically, a fine-resolution description ($\mathcal{D}_F$) is defined as a *refinement* of a coarse-resolution description ($\mathcal{D}_C$), with the agent now able to reason about aspects of the domain that were previously abstracted away. In our example scenario, the domain is organized into abstract regions in $\mathcal{D}_C$, with each region being refined in $\mathcal{D}_F$ into smaller regions that are components of the larger region, e.g., kitchen in $\mathcal{D}_C$ comprises kitchen table, kitchen counter, refrigerator in $\mathcal{D}_F$. In a similar manner, an object in $\mathcal{D}_C$, e.g., a cup, can comprise multiple parts, e.g., handle and body, in $\mathcal{D}_F$. The signature Σ_F of $\mathcal{D}_F$ is created by expanding Σ_C of $\mathcal{D}_C$ to include the new sorts, actions, fluents, and statics. Next, the axioms of $\mathcal{D}_F$ are defined by adapting some actions from $\mathcal{D}_C$ to the new (expanded) signature,

defining new actions, and defining *bridge axioms* that link the two descriptions. For our example scenario, axioms of $\mathcal{D}_L$ include:

$$move^*(A, L) \text{ causes } at^*(A, L) \quad (2a)$$

$$at(A, Rg) \text{ if } at^*(A, L), component(L, Rg) \quad (2b)$$

$$impossible grab(A, O) \text{ if } at^*(A, L_1), at^*(O, L_2) \quad (2c)$$

where L refers to grid-based locations in $\mathcal{D}_F$ that are a *component* of a region location Rg in $\mathcal{D}_C$, and superscript “*” refers to relations introduced in $\mathcal{D}_F$. This coupling between descriptions provides separation of domain-dependent and domain-independent parts of the architecture, and defines conditions under which we can guarantee that any given transition in $\mathcal{D}_C$ can be implemented as a sequence of transitions in $\mathcal{D}_F$. In addition, it allows us to introduce (in $\mathcal{D}_F$) observations and non-deterministic knowledge-producing actions, and to associate probabilities with action outcomes. Prior work in our group has demonstrated how $\mathcal{D}_F$ can be automatically mapped to probabilistic sequential decision-making formalisms for planning [41].

Reasoning with the entire fine-resolution description can quickly become computationally intractable for complex domains. Our architecture addresses this problem by incorporating the related principle of *attention*. Specifically, the formal coupling between the two descriptions is leveraged to enable an ad hoc agent to automatically identify and *zoom to* the part of $\mathcal{D}_F$ that is relevant to any given goal and abstract action under consideration. For example, an agent moving from an *office* to the *kitchen* to fetch a cup of hot tea can focus on just these two rooms and their components while ignoring the other rooms in the domain. Specifically, sorts and ground object instances in the signature Σ_F of $\mathcal{D}_F$ are automatically restricted to those that are relevant to the abstract action (transition) to be implemented, and the axioms of $\mathcal{D}_F$ are restricted to this reduced signature. This process substantially reduces the complexity of reasoning at the finer-granularity, leading to computationally efficient operation.

A common criticism of knowledge-based reasoning methods is that they need comprehensive domain knowledge, but architectures that embed key principles such as refinement have demonstrated the ability to act based on existing knowledge while revising it incrementally [42]. Also, most of the steps for encoding the available knowledge can be automated, and the effort involved in encoding

prior knowledge is much less than the effort involved in designing and training purely data-driven systems.

3.2 Agent Behavior Models

Reasoning with just prior domain knowledge that can be incomplete or inconsistent will lead to poor performance, particularly under AHT settings where the other agent’s actions also revise the domain state. Our architecture enables the ad hoc agent to also reason with models that predict the action choices of other agents and are learned (and revised) rapidly. This capability is achieved by embedding the principle of *Ecological Rationality* (ER) [18], which is based on Herb Simon’s original definition of Bounded Rationality [38] and the algorithmic theory of decision heuristics [19]. ER explores decision making “in the wild”, i.e., under open world uncertainty with the space of possibilities not fully known a priori, and characterizes behavior as a joint function of internal cognitive processes and the environment. It advocates the use of *adaptive satisficing* for making decisions in open worlds: in the absence of comprehensive knowledge, optimal decisions may be unknowable and not just hard to compute. Thus, decision heuristics (e.g., tallying, sequencing, fast and frugal methods) are used to ignore part of the information, and to make decisions more quickly, frugally, and accurately than sophisticated methods with many more free parameters [19]. This approach has been shown to provide better performance than more sophisticated methods in practical applications, but such decision heuristics and their successes do not receive the attention that they deserve [17].

Specifically, our architecture enables the ad hoc agent to select relevant attributes and learn models of the other agents behavior incrementally and from limited data. The agent learns an ensemble of *Fast and Frugal* (FF) trees to predict the behavior of each teammate (or type of teammate). Each FF tree provides a binary choice for a particular action, and the number of leaves in a tree is limited by the number of attributes [22]. Each level of the tree contains an exit allowing the agent to make quick decisions based on available data. Unlike many sophisticated methods for AHT, these predictive models can be learned and revised incrementally and rapidly. Also, the ad hoc agents make decisions efficiently by evaluating the more informative attributes and stopping as soon as a rational option is found. Figure 3 shows an FF tree learned for a human, with Table 1 showing the key attributes used in these trees. Since each FF tree provides a binary choice, we build each predictive model as an ensemble of FF trees with a simple decision tree choosing an action based on the output of the FF trees. As we show later (Section 4.2), reasoning with prior knowledge and these models provides much better performance than methods that just reason with knowledge or use learned models.

Table 1. Attributes used to create behavior models of human.

Description of the attribute
Immediate two previous actions of the human
Immediate two previous objects human interacted
Position of the human (x,y,z)
Orientation of the human (x,y,z)
Distance from human to the kitchen table
Distance from human to the kitchen counter
Number of objects on the kitchen table

Consistent agreement (or disagreement) between actual observations (of outcomes) and the predictions provided by these behavior

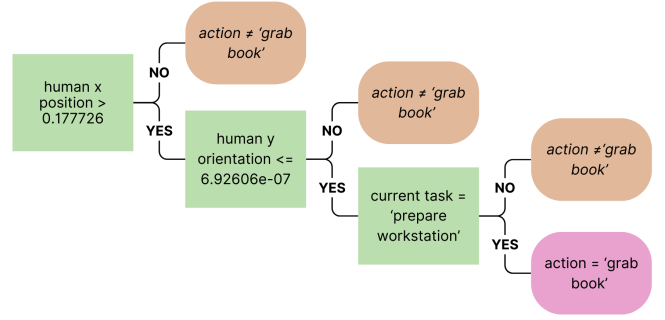


Figure 3. FF tree model of human behavior for the grab_book action in the example scenario from the *VirtualHome* domain.

models triggers the use of a particular model for subsequent steps, or leads to the revision of the existing model(s), allowing the ad hoc agent to quickly adapt to changes in the domain or another agent’s behavior over time.

3.3 Task Anticipation

An agent’s ability to anticipate future tasks and goals in the domain can enhance collaboration and overall performance. However, encoding such knowledge manually across different domains can be challenging and impractical. On the other hand, LLMs have been shown to be effective when used to translate natural language to domain-specific language, and to generate high-level guidance that is implemented by planning subroutines [21]. We thus enable an ad hoc agent to use an LLM to anticipate the high-level future task (e.g., *prepare dinner*) likely to be assigned once the current task is done. We experimentally demonstrate that jointly planning to complete the current and anticipated tasks improves team performance. The agent uses a combination of prompting strategies with the LLM:

- **Adopting persona:** A specific role or character is assigned to guide the LLM’s responses to be more (contextually) consistent with the assigned role.
- **Few-shot prompting:** The prompt includes a few examples of the expected output in specific situations, guiding the use of pretrained knowledge.
- **Chain-of-thought (CoT):** The prompt includes a step-by-step reasoning process that can be followed to arrive at an answer, leading to more accurate responses.

Specifically, a ‘system message’ guides the LLM to adopt the persona of a household assistant and complete the partial task routine (provided as input) by selecting from the available list of tasks in the example scenario (in *VirtualHome*). With ‘few-shot’ prompting, the prompt includes two task routines randomly chosen from previous days. Next, CoT prompting is used to explain the reasoning behind each task in these example task routines. Such explanations can be provided by the system designer or generated by the LLM. The system message, few-shot examples with CoT explanation, and the current query (i.e., partially completed or empty task routine for the day) are provided as input to the LLM.

The LLM’s output to a prompt is parsed by an *external validator* to check whether the tasks are feasible and in a reasonable order. Since LLMs are trained with large volumes of text from many domains, the anticipated tasks may not be feasible or match the preferences and needs of the human of our domain. Thus the validator compares the LLM’s output with domain- and task-specific features extracted

from existing knowledge and recent observations, eliminating tasks that are invalid or irrelevant and reordering tasks according to human preferences. For example, the agent will prioritize preparation of the workstation over packing a lunch box when the human is working from home. Since the list of validated tasks can change over time, the ad hoc agent considers one anticipated task and the current task as the joint goal for which a plan is computed.

3.4 Transparency in decision making

An automated decision-making system’s ability to answer questions about its decisions promotes acceptability [1, 15]; this transparency also plays an important role in human reasoning and learning. Unlike existing methods that seek to make a complex learned model interpretable, or to explain (or justify) all the choices made by a reasoning system, KAT seeks to respond to any given question about specific decisions made by the ad hoc agent or the human (teammate) by quickly identifying the relevant information and constructing a relational description as the answer.

The design choice of using knowledge-based reasoning and simple predictive models in KAT is the foundation for the approach described in this paper to provide the desired on-demand descriptions. We build on prior work in our group that demonstrated the ability to identify the axioms and literals relevant to the questions posed to a system making automated decisions [39]. The underlying principle is that of *explainable agency*, which corresponds to enabling specific functional abilities [23, 24]. Specifically, the agent must be able to: (i) provide on-demand justification of decisions made during (or after) plan generation and execution by considering alternative choices; (ii) report actions executed and present information at a suitable level of abstraction; (iii) describe how actual events deviated from a computed plan and how it adapted to these deviations; and (iv) communicate information about decisions and justification such that it makes contact with human concepts such as beliefs and goals.

As described in Section 3.1, KAT is designed to represent and reason with knowledge at different abstractions. We build on this representation and the associated update processes to embed the principle of explainable agency, i.e., to implement the abilities mentioned above, enabling the ad hoc agent to generate relational descriptions in response to four types of questions identified as being important in work on explainable planning [15]. Here we describe the procedure the ad hoc agent follows to generate responses for each of these types of questions. Examples of the use of this procedure to construct explanations on-demand are provided later in Section 4.3.

Causal questions: *Why did you execute a_I , i.e., action a at step I ?* As described in Algorithm 1, when asked why an action was executed, the ad hoc agent first checks whether action a_I is the last action of plan P executed by the agent (Line 1, Algorithm 1). If not, the agent extracts the actions $\{A_{af}\} \in P$ that occurred immediately after $a_I \in P$ (Line 2). Next, the agent examines the program $\Pi(\mathcal{D}, \mathcal{H})$ to identify any axioms with the negation of $a_i \in \{A_{af}\}$ in *head*, i.e., axioms encoding conditions that prevent a_i from occurring (Line 3). The agent then checks if each such selected axiom’s *body* is satisfied by the *answer set* at time step I (Line 5). If yes, it indicates that action a_i could not be executed unless the fluent literals f in the *body* of the axiom change value. The agent stores the fluent literals f in *body* whose value changed from I to $I + 1$ (Line 6). The value of all such literals $\{f\}$ over all identified axioms has been changed by the execution of a_I , the action under consideration. The agent uses these literals to answer the question about a_I (Line 7).

Algorithm 1: Generate Causal Explanations

Input: P : Plan; a_I : action; G : goal; $\Pi(\mathcal{D}, \mathcal{H})$: program; $\mathcal{AS}$: answer set

Output: E : Explanation literals

```

1 if  $a_I$  is not the last action in  $P$  then
2    $A_{af} \leftarrow \{a_i \in P \mid i > I\}$ 
3    $R \leftarrow \{r \in \Pi(\mathcal{D}, \mathcal{H}) \mid \text{head}(r) = \neg a_i, a_i \in A_{af}\}$ 
4   for  $r \in R$  do
5     if  $\text{body}(r)$  satisfied in  $\mathcal{AS}$  at  $I$  then
6        $F \leftarrow \{f \in \text{body}(r) \mid \text{val}(f, I) \neq \text{val}(f, I + 1)\}$ 
7        $E \leftarrow E \cup F$ 
8     end
9   end
10 else
11    $R \leftarrow \{r \in \Pi(\mathcal{D}, \mathcal{H}) \mid a_I, f' \in \text{body}(r)\}$ 
12    $F' \leftarrow \{f' \mid f' \in \text{body}(r), r \in R\}$ 
13   for  $l \in F'$  do
14      $C \leftarrow \{r \in \Pi(\mathcal{D}, \mathcal{H}) \mid \text{head}(r) = l\}$ 
15     for  $r \in C$  do
16       if  $\text{body}(r)$  satisfied in  $\mathcal{AS}$  at  $I$  then
17          $E \leftarrow E \cup \{l \cup \text{body}(r)\}$ 
18       end
19     end
20   end
21    $E \leftarrow E \cup G$ 
22 end
23 return  $E$ 

```

If a_I is the last action in P , the agent examines $\Pi(\mathcal{D}, \mathcal{H})$ to identify axioms with a_I in their *body* alongside other additional literals $\{f'\}$, i.e., axioms encoding effects of a_I if additional conditions are true (Line 11). The agent then extracts these literals $\{f'\}$ from the identified axioms and examines $\Pi(\mathcal{D}, \mathcal{H})$ to identify axioms with literal $l \in \{f'\}$ in their *head*, i.e., state constraints (Lines 12 - 14). The agent then checks whether each of the selected axiom’s *body* is satisfied by the *answer set* at time step I (Line 16). If yes, the agent use the literal $l \in \{f'\}$ and the *body* literals of the axioms to construct the answer as they have prompted the agent to execute a_I . Furthermore, as the last action in P , a_I directly contributed toward achieving the goal, and the agent also uses the goal G terms to construct the answer.

Contrastive questions: *Why did you not execute a_I , i.e., action a at step I ?* As described in Algorithm 2, when asked why an action was not included in the plan, the ad hoc agent first examines the program $\Pi(\mathcal{D}, \mathcal{H})$ to identify axioms with the negation of a_I in its *head*, i.e., executability conditions (Line 1, Algorithm 2). For each selected axiom, the agent checks whether the axiom’s *body* is satisfied by *answer set* at I (Lines 2-3). If yes, the agent collects the fluent literals $\{f\}$ in the *body*, as they prevented consideration of a_I , and uses them to construct the explanation (Line 4).

If no fluent literals have been identified, the agent examines $\Pi(\mathcal{D}, \mathcal{H})$ to identify axioms with a_I in its *body* alongside other literals $\{f'\}$, i.e., causal laws (Line 8). The agent extracts these additional precondition literals $\{f'\}$ and examines $\Pi(\mathcal{D}, \mathcal{H})$ to identify axioms with $l \in \{f'\}$ in its *head*, i.e., state constraints (Lines 9-11). For each selected axiom, the agent then checks if the axiom’s *body* is satisfied by *answer set* at step I (Lines 12-13). If not, it uses the literals in the *body* of the axiom, and the literal l , to construct the answer as they prevented the agent from executing action a_I (Line 14).

Algorithm 2: Generate Contrastive Explanations

Input: P : Plan; a_I : action; G : goal; $\Pi(\mathcal{D}, \mathcal{H})$: program; $\mathcal{AS}$: answer set

Output: E : Explanation literals

```

1  $R \leftarrow \{r \in \Pi(\mathcal{D}, \mathcal{H}) \mid \text{head}(r) = \neg a_I\}$ 
2 for  $r \in R$  do
3   if  $\text{body}(r)$  satisfied in  $\mathcal{AS}$  at  $I$  then
4      $E \leftarrow E \cup \{f \mid f \in \text{body}(r)\}$ 
5   end
6 end
7 if  $E = \emptyset$  then
8    $R' \leftarrow \{r \in \Pi(\mathcal{D}, \mathcal{H}) \mid a_I, f' \in \text{body}(r)\}$ 
9    $F' \leftarrow \{f' \mid f' \in \text{body}(r), r \in R'\}$ 
10  for  $l \in F'$  do
11     $C \leftarrow \{r \in \Pi(\mathcal{D}, \mathcal{H}) \mid \text{head}(r) = l\}$ 
12    for  $r \in C$  do
13      if  $\text{body}(r)$  not satisfied in  $\mathcal{AS}$  at  $I$  then
14         $E \leftarrow E \cup \{l \cup \text{body}(r)\}$ 
15      end
16    end
17  end
18 end
19 if  $E = \emptyset$  then
20    $C \leftarrow \text{cost}(P)$ 
21    $C' \leftarrow \text{cost}(P' = P \cup a_I)$ 
22   if  $C' > C$  then
23      $E \leftarrow E \cup \{C, C'\}$ 
24   end
25 end
26 return  $E$ 

```

If the literals to construct an answer have not been identified so far, the agent computes the cost C of P to achieve the goal G , and the cost C' of the P' that also include a_I (Lines 20-21). If the cost of P' is higher than that of P (i.e., $C' > C$), the agent concludes that this prevented it from including a_I in its plan (Line 22). The agent uses the cost terms to construct the answer (Line 23).

Justify beliefs: *Why did you believe l_I , i.e., l at step I ?* As described in Algorithm 3, to justify a belief at a specific step, the ad hoc agent first replaces the grounded terms of the belief l_I with their corresponding variables and examines $\Pi(\mathcal{D}, \mathcal{H})$ to identify axioms with l_I in *head*, i.e., state constraints (Line 1, Algorithm 3). The agent then checks whether each selected axiom's *body* is satisfied by the answer set at step I (Lines 2-3). If yes, it collects the fluent literals $\{f\}$ in the *body* as they support belief l_I and uses them to construct the answer (Line 4). If multiple axioms are identified, the agent selects one of them to provide the explanation.

If belief tracing is enabled, the agent creates a tree with l_I as its head and each axiom (identified in Line 1) as a branch (Lines 8-10). With the axiom, the agent also stores the supporting fluents (F). It then repeats this procedure for each fluent literal in F as target belief until no more axioms are identified (Lines 11-19). The paths of the belief tree are then used to construct the explanation.

Counterfactual Questions: *What will be the action of the human in step I ?* As described in Algorithm 4, the ad hoc agent first retrieves the most recent state observation of environment S_{I-n} in relation to step I , i.e., start with the current best estimate of world state (Line 1, Algorithm 4). It then uses the predictive behavior model of the human to retrieve their action a_{I-n}^{ot} for step $I - n$; it also uses our

Algorithm 3: Generate Belief Justifications

Input: l_I : belief; $\Pi(\mathcal{D}, \mathcal{H})$: program, $\mathcal{AS}$: answer set

Output: E : Explanation literals

```

1  $R \leftarrow \{r \in \Pi(\mathcal{D}, \mathcal{H}) \mid \text{head}(r) = l_I\}$ 
2 for  $r \in R$  do
3   if  $\text{body}(r)$  satisfied in  $\mathcal{AS}$  at  $I$  then
4      $E \leftarrow E \cup \{f \mid f \in \text{body}(r)\}$ 
5      $F \leftarrow \{f \mid f \in \text{body}(r)\}$ 
6   end
7 end
8 if belief tracing enabled then
9    $Tree \leftarrow$  create root node  $l_I$ 
10   $Tree \leftarrow$  branches ( $R, F$ )
11  for  $f \in F$  do
12     $R' \leftarrow \{r \in \Pi(\mathcal{D}, \mathcal{H}) \mid \text{head}(r) = f\}$ 
13    for  $r \in R'$  do
14      if  $\text{body}(r)$  satisfied in  $\mathcal{AS}$  then
15         $F \leftarrow F \cup \{f \mid f \in \text{body}(r)\}$ 
16         $Tree \leftarrow$  branches ( $r, F$ )
17      end
18    end
19  end
20 end
21 return  $E, Tree$ 

```

Algorithm 4: Generate Counterfactual Explanations

Input: I : future step; S_{I-n} : recent observation; FF_h : FF tree of human

Output: E : Explanation literals

```

1  $S \leftarrow S_{I-n}$ 
2 for  $t = I - n$  to  $I - 1$  do
3    $a_t^{ot} \leftarrow \text{predicted\_action}(FF_h, S)$ 
4    $a_t \leftarrow \text{compute\_agent\_action}(S)$ 
5    $S \leftarrow \text{simulate}(S, a_t^{ot}, a_t)$ 
6 end
7  $S_I \leftarrow S$ 
8  $a_I^o \leftarrow \text{predicted\_action}(S_I)$ 
9  $a_I \leftarrow \text{compute\_agent\_action}(S_I)$ 
10  $E \leftarrow$  active branches in FF tree  $FF_h$  for  $a_I^o$ 
11 return  $a_I^o, E$ 

```

architecture to compute the action a_{I-n} that it should execute (Lines 2-4). The agent then performs a mental simulation of the future step $I - n + 1$ from S_{I-n} using existing knowledge and action choices identified (a_{I-n}^{ot}, a_{I-n}) (Line 5). The agent repeats above n times, i.e., roll out the future and explore effects of the action of ad hoc agent and human until environment reaches the queried step I . It then collects the state information S_I at I (Line 7). The agent then feeds the state information S_I to the predictive behavior model of the human to retrieve the action a_I^o , and use our architecture to retrieve the action for the ad hoc agent a_I in state S_I (Line 7-9). Specifically, the agent traverses the FF tree model of the human (FF_h) to identify branches that were active in selecting a_I^o ; these branches and a_I^o are used to construct the answer (Line 10).

If the counterfactual question is about the outcome of a specific action executed by the human at a specific time step, this action is used directly instead of the predicted action Line 8 of Algorithm 4. In a similar manner, if the question is about the outcome of a hypothetical

action by the ad hoc agent at a specific future time step, this action is considered in Line 9. In addition, actions at a specific time step in the past can also be considered by examining the recorded history and then performing a mental simulation. Furthermore, if there are multiple agents (other than the ad hoc agent), the ad hoc agent would consider the corresponding predictive models as input and compute a set of actions $\{A_{I-n}^{ot}\}$ in Line 3 for all these agents.

Across the different types of questions, any information obtained can be used for further training, especially the human behavior prediction models. For all types of questions, the identified literals are processed using existing tools and templates to generate textual descriptions provided as responses (i.e., explanations) before, during, or after planning or execution.

4 Experimental Setup and Results

We experimentally evaluated the following hypotheses regarding our architecture’s capabilities:

- H1** Reasoning with prior knowledge and rapidly-learned behavior models improves performance;
- H2** Using task anticipated by LLM as joint goal improves performance compared with not planning for joint goal;
- H3** Incrementally-updated prompts and validators improve task anticipation and team performance;
- H4** Using LLM to directly output a sequence of low-level actions to complete tasks results in poor performance; and
- H5** Our architecture enables the ad hoc agent to generate relational descriptions as explanations of its decisions and beliefs and those of the human.

We evaluated these hypotheses in the *VirtualHome* and recorded the number of steps (plan length) and the total time taken to complete the task(s) as the *performance measures*. All prompts to the LLM were through OpenAI GPT4o mini API with default generation (temperature = 1.0). Details about experiments, baselines, and measures are provided below.

4.1 Experimental Setup

In the *VirtualHome* domain, we modeled the human as a simulated entity that chooses its actions based on an ASP program that considered prior knowledge and runtime observations but did not have models predicting teammates’ actions. The human’s ASP program also include actions such as eating and drinking that were not available to the ad hoc agent. Furthermore, the human’s ASP program encoded priorities and preferences, e.g., when preparing breakfast, the human toasted the bread first before preparing the cereal. The sequence of tasks generated by the task generator were assigned to all agents one task at a time. The agents were not aware of the complete sequence and received one task at a time. The human was assigned the same goal as the ad hoc agent(s). All agents received the same observations of the domain at each step, which they used to plan their respective actions without direct communication with each other.

When an ad hoc agent equipped with our architecture received a task, it prompted the LLM (Section 3.3). The anticipated tasks were validated and mapped to ASP literals, with the next anticipated task and current task set as joint goals for this agent. During planning, the ad hoc agent also used the learned behavior prediction model to predict each teammate’s actions for a few steps (Section 3.2). These predictive models were built using only 1000 examples of prior traces

of actions and domain state and were incrementally revised based on teammates observed behavior. The ASP program of the ad hoc agent included additional axioms for reasoning about these predicted actions of each teammate, which were mapped to exogenous actions. The ad hoc agent’s plan anticipated preconditions of some intermediate steps to be satisfied by a teammate’s actions, even though the teammate did not always execute that action. The ad hoc agent hence had to respond to unexpected action outcomes.

For evaluating **H1** and **H2**, in **Exp1**, we randomly selected 100 task routines sampled from predefined sequences and measured the ability of a team (human and an ad hoc agent) to complete these tasks. Performance measures were the number of steps and time taken. We used three baselines: **Base1**: LLM for anticipating future tasks, but no behavior models to predict human’s actions; **Base2**: no LLM to anticipate future tasks, but behavior models to predict the human’s actions; and **Base3**: did not use LLM for task anticipation or behavior models to predict human’s actions. In the absence of a framework that supports all (or most) of our architecture’s capabilities, these baselines allowed us to conduct ablation studies and evaluate the contribution of each key component. Since the time taken and the number of actions in a plan can vary substantially based on the task, the average of these values across trials may not be meaningful. We instead *ran paired trials and computed performance measure values for baselines as a fraction of these values for our architecture in each trial*. We then reported the average of these ratios.

To evaluate **H3**, in **Exp2**, we computed the precision and recall of tasks anticipated by the LLM, before and after applying the validator, over the 100 task routines. We also computed precision and recall of a simple statistical model that replaced the LLM and anticipated the next task based on past experience. Further, in **Exp3**, we randomly selected 20 task routines and recorded the LLM’s performance with and without prompting methods, using four baselines: **Base4**: no prompting strategy or validator; **Base5**: few-shot prompting but no validator; **Base6**: CoT prompts but no validator; and **Base7**: validator but no prompt-engineering.

For evaluating **H4**, we conducted **Exp4**, with the LLM being used to directly compute sequences of actions for specific tasks (**Base8**). The prompt included details of actions available in our example scenario in *VirtualHome*, and their intended purpose. We also supplied the LLM some **Action Feasibility Rules**: (a) *movement limitation* (critical): must only move to adjacent locations defined by the next_to relationships; (b) *object location*: must be in the same location as an object to act on it (e.g., grab, put); (c) *carry limit*: cannot hold more than two objects, and actions like open, close, switch-on, or switch-off require putting at least one object down if holding two objects; (d) *appliance safety*: you should not open appliance doors when they are switched on; (e) *avoid conflict*: a human holding an object will perform actions involving object, so focus on other parts of the goal. The LLM had access to the current state, including location of agents, objects, and appliances, and each appliance’s state. The problem specification described the task to be performed; the immediate prior actions of the human and ad hoc agent; and other relevant information (e.g., human working from home). Additionally, the prompt included a detailed example of selecting an action, and asked the LLM to generate an action sequence for the ad hoc agent to complete the task. Feedback was provided for errors in the LLM’s output (e.g., action to grab an object without moving to suitable location) up to three times per trial; LLM was told why this choice was incorrect, and allowed to output another action. We measured the number of steps and time taken by the team to complete the selected 100 task routines.

To evaluate **H5** in **Exp5**, we randomly selected 10 configurations

(from 100 in **Exp1**) and saved the corresponding answer sets (with our architecture, baseline) to provide ground truth. Then, we posed 32 different questions (split between four types of questions) about some chosen steps in each trial corresponding to one of these 10 configurations, with answers computed as described in Section 3.4. We recorded precision and recall of the literals in the answers.

4.2 Experiment Results

Table 2 summarizes the results of **Exp1**. When the ad hoc agent reasoned with anticipated tasks and predicted human actions, it provided the best observed performance with lowest number of action steps and least amount of time taken to complete the routines. The prediction accuracy of the behavior models learned by the ad hoc agent for the human using the ensemble of FF trees was 85%. We observe that the model does make errors, but it supports rapid learning and revision. Also, reasoning with prior knowledge and the output of these predictive models significantly improves the performance. When the ad hoc agent used task anticipation but did not use the behavior prediction models (**Base1**), the number of steps and time taken to complete tasks increased because the inability to predict the actions of teammates may lead the ad hoc agent to execute the same actions as a teammate, hindering collaboration. These results emphasize the importance of using the behavior prediction models, supporting **H1**.

When the ad hoc agent used the behavior models to predict the human’s actions, but did not anticipate and plan for future tasks (**Base2**), the performance worsened, with a further increase in the number of steps and the time taken to complete the tasks. Recall that this setting corresponded to the agent only planning for one goal at a time without anticipating future goals. Planning actions jointly for the current task and the anticipated (next) task saved time and effort. For example, when the agent visited the bedroom to retrieve a board game to entertain guests, it also picked up bottles of wine from the cellar that is on the way. Making two separate trips for these tasks extended the length and duration of the plans. These results indicate the impact that planning for joint goals has on performance, supporting **H2**. When the ad hoc agent did not use task anticipation or the behavior prediction models (**Base3**), it resulted in the worst observed performance with the highest value for number of steps and time taken to complete the tasks. These results support **H1** and **H2**. All results were statistically significant ($p < 0.0001$).

Table 2. Average number of steps and time taken to complete task routines, with values for baselines computed as a fraction of these values for our architecture in each trial; for comparison, the average absolute values are 26.8 steps and 361 seconds for our architecture. Task anticipation (LLM) and teammates’ behavior/action prediction (FF trees) substantially improved performance.

Architecture	Steps	Time(s)
Ours (anticipate tasks, predict actions)	1.0	1.0
Base1 (anticipate tasks)	1.1	1.1
Base2 (predict actions)	1.3	1.2
Base3	1.4	1.4

Table 3 shows the results from **Exp2**, i.e., the precision and recall of the tasks anticipated by a simple statistical model, and by the LLM before and after applying the validator. The errors in the LLM’s outputs were substantially reduced by the validator. The recall values do not change substantially as the validator did not introduce new tasks; it only reordered the tasks that are out of order and removed irrelevant tasks. These results support **H3**.

Table 3. Precision and recall values of the anticipated tasks with and without the LLM and the validator.

Architecture	Precision	Recall
Simple statistical model	0.75	0.76
LLM without validator	0.78	0.76
LLM with validator	0.99	0.78

Table 4 shows the results from **Exp3**, which explored the use of different prompting strategies and the validator with the LLM to anticipate tasks. We observed a significant performance improvement with the external validator and a combination of prompting methods. The results were statistically significant for **Base4-6** ($p < 0.001$). **Base7** showed mixed results; reduction in steps was statistically significant ($p < 0.05$), while reduction in time was not ($p = 0.09$). This outcome matched expectations as **Base7** used the validator; it emphasizes the importance of the validator and the need for further exploration of more complex scenarios to determine the importance of the prompting strategies. These results support hypothesis **H3**.

Table 4. Average number of steps and time taken by the team (human, ad hoc agent) to complete tasks with prompting strategies and/or external validator; performance measure values for baselines computed as a fraction of values for our architecture in each trial, with average absolute values of 27.5 steps and 372.7 seconds for our architecture.

Architecture	Steps	Time(s)
Proposed (all prompting, with validator)	1.00	1.00
Base4 (no prompting, no validator)	1.21	1.15
Base5 (few-shot prompting, no validator)	1.17	1.18
Base6 (chain-of-thoughts, no validator)	1.16	1.16
Base7 (no prompting, with validator)	1.05	1.04

Next, Table 5 shows the results of **Exp4**, where we used the LLM to directly compute the low-level actions for the ad hoc agent in the example scenario in *VirtualHome*. The number of steps and time taken to successfully complete the task routines are significantly higher than our architecture as well as all other baselines (**Base1-3**) in Table 2. These results support **H4**.

Table 5. Average number of steps and time taken by the team (human, ad hoc agent) to complete task routines when the LLM directly outputs sequence of low-level actions to be executed. Performance measure values computed as a fraction of values obtained with our architecture in Table 2.

Architecture	Steps	Time(s)
Base8 (LLM predict low-level actions)	1.5	1.5

Table 6 shows results of **Exp5**, with precision and recall calculated relative to all the literals expected to be included in the explanation. We observed high precision and recall indicate the agent’s ability to automatically extract the correct literals to provide relational descriptions as explanations. These results support hypothesis **H5**.

4.3 Execution Trace

We also provide some execution traces as a qualitative evaluation of **H3**. Figure 4 shows an execution example where the ad hoc agent used the LLM to anticipate future tasks, with and without the prompting strategies and validator. The example was set on a weekday where the human was working from home and no guests were expected. The

Table 6. Precision and recall of retrieving relevant literals for providing explanations.

Question type	Precision	Recall
Action justification	1.00	1.00
Contrastive	0.89	0.99
Belief justification	0.88	0.94
Counterfactual	1.00	0.78

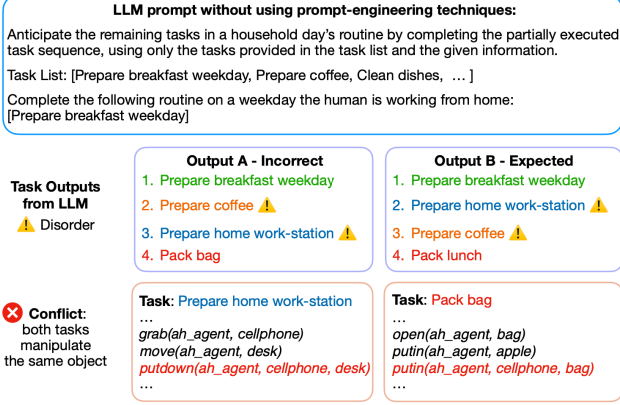


Figure 4. Execution example: using LLM without prompting strategies or external validator causes conflicts during execution, having a negative impact on performance.

correct task routine in this context was: *Prepare breakfast, Prepare home work-station, Prepare coffee, Prepare lunch*.

When the ad hoc agent queried the LLM without the prompt engineering techniques or validator (Section 3.3), the anticipated task list was different from the expected output. The prompt to the LLM without using the prompt engineering strategies is shown in Figure 4. The LLM output in this scenario was [*Prepare breakfast, Prepare coffee, Prepare home work-station, Pack bag*]. This output failed to align with the human preferences and priorities as:

- Higher priority was assigned to making coffee than setting up the workstation. This would delay the human for work and lead to coffee not being hot when needed.
- Packing the bag was an unnecessary task as the human was not leaving the house; it would have been filtered by the validator.

These preferences and priorities were captured by the agent through its experience with the human across multiple iterations, and used by the validator to filter out irrelevant tasks or reorder them. When the ad hoc agent used the prompt engineering and validation strategies, the prompt to the LLM was automatically generated while incorporating context, as described in Section 3.3. The LLM's output was [*Prepare breakfast, Prepare home work-station, Prepare coffee, Prepare lunch*]. This matched the expected routine. i.e., making breakfast and setting up the workstation were considered high priority tasks, and irrelevant tasks such as *pack bag* were filtered out by the validator. These results demonstrate the importance of using a combination of prompting techniques and the external validator, supporting **H3**.

We observed similar situations when three agents, one human and two ad hoc agents, were considered to be collaborating together on a weekend when guests were expected. The correct task routine in

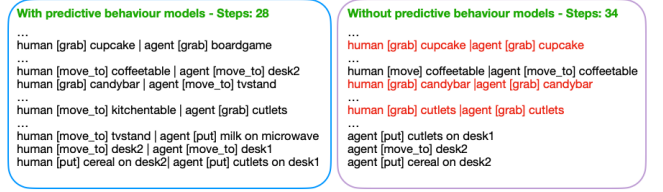


Figure 5. Execution trace for task routine: [*Prepare breakfast, Prepare activities, Serve snacks, Clean kitchen*]. When an ad hoc agent is not allowed to predict and reason about the human's actions, it may choose to execute same action(s) as the human, leading to longer plans.

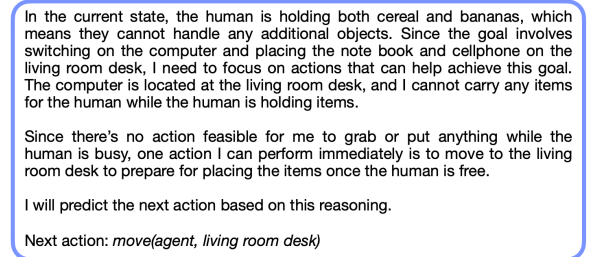


Figure 6. Example output when ad hoc agent uses the LLM for computing sequences of actions for specific tasks (**Base8**).

this context was: *Prepare breakfast, Prepare table for guests, Prepare lunch, Clean dishes*. When the first ad hoc agent queried the LLM with the prompting strategies but without the validator, the anticipated task list by the LLM was *Prepare breakfast, Prepare table for guests, Prepare lunch, Serve snacks, Clean dishes*. For the second ad hoc agent the LLM output was *Prepare breakfast, Prepare table for guests, Prepare lunch, Serve snacks, Clean dishes*. When the validator was used, the outputs to both the agents were refined by incorporating context. Since the human usually did not require snacks after lunch, the task *Serve snacks* was removed. For the first agent the refined task list was *Prepare breakfast, Prepare table for guests, Prepare lunch*. For the second agent the task list was *Prepare breakfast, Prepare table for guests, Prepare lunch, Clean dishes*; since *Clean dishes* was a defined task in this domain, it was retained by the validator. This example further demonstrates the importance of using the validator in our architecture, and supports **H3**.

Next, Figure 5 compares two plans executed by a team comprising a human and an ad hoc agent for completing a different set of tasks: [*Prepare breakfast, Prepare activities, Serve snacks, Clean kitchen*], with and without the behavior prediction models (Section 3.2). In the first plan, when the ad hoc agent used the behavior prediction model to predict the future actions of the human, the team successfully completed all tasks for the given day in just 28 steps. On the other hand, when the ad hoc agent did not use behavior prediction models, it often selected the same actions as the human for any particular task, leading to unnecessary delays in completing the tasks. For example, in the second plan the agent frequently selected the same action as the human—simultaneously picking up the cupcake, candy bar, and cutlets, introducing redundant behavior and prolonging task execution. As a result, the overall plan was extended to 34 steps. These results

Table 7. Part of the ad hoc agent’s plan for task in Execution Example 1.

occurs(find(ahagent,breadslice),0)	occurs(putin(ahagent,cutlets,fryingpan),8)
occurs(grab(ahagent,breadslice),1)	occurs(put(ahagent,cutlets,kitchentable),11)
occurs(put(ahagent,breadslice,kitchentable),2)	occurs(grab(ahagent,poundcake),13)
occurs(switchon(ahagent,stove),4)	occurs(open(ahagent,microwave),15)
occurs(grab(ahagent,cutlets),6)	occurs(put(ahagent,poundcake,microwave),16)

Table 8. Part of plan generated by ad hoc agent to avoid conflict with human.

occurs(find(ahagent,cutlets),1)	occurs(put(ahagent,poundcake,microwave),9)
occurs(grab(ahagent,cutlets),2)	occurs(close(ahagent,microwave),10)
occurs(find(ahagent,fryingpan),3)	occurs(switchon(ahagent,microwave),11)
occurs(putin(ahagent,cutlets,fryingpan),4)	occurs(open(ahagent,microwave),13)
occurs(grab(ahagent,poundcake),6)	occurs(put(ahagent,poundcake,kitchentable),16)
occurs(open(ahagent,microwave),8)	occurs(put(ahagent,cutlets,kitchentable),24)

demonstrate that using the behavior prediction models enables the ad hoc agent to coordinate efficiently by avoiding action conflicts with the human. These further supports **H1**.

When the ad hoc agent used the LLM for directly computing sequences of actions for specific tasks **Base8**, the prompt was automatically constructed following the procedure described in Section 4.1. Figure 6 shows part of the LLM output during the task routine: [*Prepare breakfast, Prepare home work-station, Prepare coffee, Prepare lunch*]. The selected action *move(agent, living room desk)* violated the ‘Movement Limitation (Critical)’ rule in ‘Action Feasibility Rules’, which constrained the agent to only move to locations adjacent to its current location. This example demonstrates that the LLM may not respect constraints even when they are provided as input, and highlights that an LLM is not designed for computing plans for non-trivial tasks; using an LLM to directly output a sequence of low-level actions to complete tasks can lead to infeasible actions, resulting in poor performance. These results support hypothesis **H4**.

Next, we provide some execution traces illustrating our architecture’s explanation generation capabilities. Consider the scenario in which the task is to *prepare breakfast* and the world state is: bread slice is inside the toaster; cutlets are on the kitchen counter; poundcake is on the kitchen table; water glass is in the bedroom; microwave is closed and switched off; frying pan is on the stove that is switched off; and the human and ad hoc agent are in the kitchen.

Consider the scenario in which the: bread slice is inside toaster; cutlets are on kitchen counter; poundcake is on kitchen table; water glass is in bedroom; microwave is closed and switched off; frying pan is on stove that is switched off; and the human and ad hoc agent are in the kitchen. To help the human prepare breakfast, the ad hoc agent generated a plan with 23 steps, some of which are shown in Table 7; as usual, the agent expects the human collaborator to complete some intermediate steps.

Execution Example 1. [Action Justification, Contrastive, Counterfactual] Consider an exchange with the ad hoc agent after executing first plan step.

- **Questioner:** “Why did you find bread slice in step 1?”
- **Ad hoc Agent:** “Because I had not found the bread slice yet and I wanted to grab it in step 2”.

The agent’s response draws attention to the target action’s outcome being a requirement for executing a subsequent action.

Specifically, since the *grab* action occurred immediately after *find*, the relevant axioms identified included:

$$\neg \text{occurs}(\text{grab}(R, O), I) \leftarrow \neg \text{holds}(\text{found}(R, O), I), \quad (3)$$

$$\text{agent}(R), \text{graspable}(O).$$

Next, the ad hoc agent explored the relevant answer sets and identified that the literal *found(ad_hoc_agent, bread_slice)* was present in step 2 but not in step 1. This literal was then selected to justify the action execution. The agent can also be asked why it did not consider picking up a different object.

- **Questioner:** “Why did you not find the water glass in step 1?”
- **Ad hoc Agent:** “Because I predicted that the human will find the water glass in step 1.”

The agent first tried to identify the axioms with *find* in the head.

$$\neg \text{occurs}(\text{find}(R, O), I) \leftarrow \neg \text{holds}(\text{agent_found}(T, O), I), \quad (4)$$

$$\text{other_agents}(T), \text{graspable}(O).$$

Next, the agent ground the body of the axiom and verified whether it was included in the answer set. Since the *agent_found(human, waterglass)* was valid in step 1, the agent identified it as a literal that prevented it from considering the *find* action in step 1. This is due to the ad hoc agent’s model of the human predicting that the human will execute action *exo_find(human, waterglass)* in step 0. The agent may also be asked about the human’s action choices further.

- **Questioner:** “Why do you think the human will pick up the water glass in step 2?”
- **Ad hoc Agent:** “Because my prediction is that the human wants to bring the glass to the table.”
- **Questioner:** “What will happen if the human decided to pick up the cutlets in step 2?”
- **Ad hoc Agent:** “If the human picks up the cutlets in step 2, they will be in the human’s hands in step 3.”

To answer questions about hypothetical situations, the ad hoc agent has to simulate the evolution of state, and the execution of actions by the human and the agent for a few steps in order to identify and report the motivation for specific action choices. Specifically, the agent first retrieved the current state estimate and

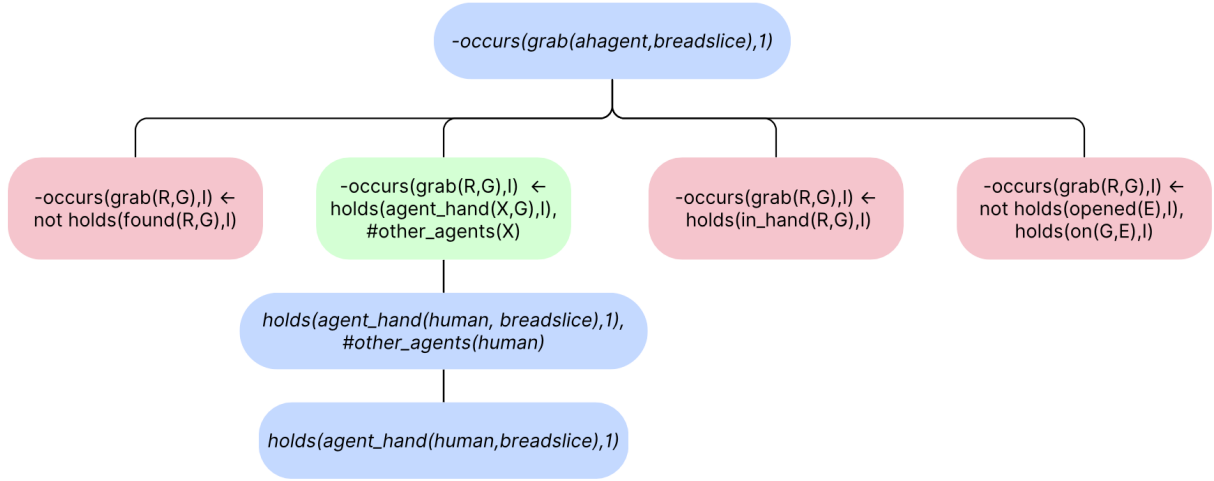


Figure 7. Belief tracing to explore why ad hoc agent did not grab bread slice in step 1.

initialized a new *VirtualHome* environment. Then it simulated the environment for the desired future time steps while using the human behavior model to predict the future actions of the human. Next, it went through the FF tree model predicting the human behavior and identified rules that caused it to believe the human will *grab* the water glass in step 2. this information was then used to generate the responses (above) to the questions posed.

Execution Example 2. [Action Justification, Contrastive, Belief Tracing] The accuracy of the model predicting human action choices (85%) indicates that the predictions can be incorrect, particularly in situations in which the agent’s understanding of the human is limited to a few observations. Consider a variant of the scenario above, in which the human decided to find and grab a slice of bread as the first action. Since the ad hoc agent decided to do the same action, it created a conflict. Some key steps of the plan computed by the ad hoc agent to overcome this conflict are shown below. The subsequent conversation with the ad hoc agent was as follows:

- **Questioner:** “Why did you find outlets in step 1 ?”
- **Ad hoc Agent:** “Because I have not found the outlets and I wanted to grab them in time step 2”.
- **Questioner:** “Why did you not grab bread slice in step 1?”
- **Ad hoc Agent:** “Because the human was holding the bread slice.” This exchange demonstrates the ability of the ad hoc agent to change its plan in order to prevent a conflict with the human, and to justify this decision. Similar to the previous example, the ad hoc agent used the identified axioms and literals to generate the answer to the questions posed. Figure 7 shows the belief tree created for tracing the ad hoc agent’s beliefs and justifying why it did not grab the bread slice in step 1. The green and blue boxes represent the satisfied axioms and their grounded literals, while the red boxes represent the axioms that were not satisfied in this example.

Overall, these results support hypothesis **H5**. Additional results in the form of videos are available in our open-source code repository [11].

5 Conclusions

This paper described an AHT architecture that embeds the principles of refinement, ecological rationality, interactive learning, and explainable agency, enabling each ad hoc agent to: leverage the generic knowledge in a pretrained LLM for high-level task anticipation; rapidly learn models predicting teammates’ actions; perform non-monotonic logical reasoning with relevant prior knowledge and behavior models to plan and execute actions that jointly achieve current and anticipated tasks; leverage natural language descriptions of action outcomes and runtime observations to acquire formal representations of previously unknown knowledge; and generate on-demand explanations of the agent’s decisions and those of others as answers to queries. Experiments in a realistic simulation environment demonstrated performance improvements compared with baselines, highlighting the importance of each component and the ability to scale to additional agents. Future work will explore larger, heterogeneous teams in AHT settings.

REFERENCES

- [1] Sule Anjomshoe, Amro Najjar, Davide Calvaresi, and Kary Framling, ‘Explainable agents and robots: Results from a systematic literature review’, in *International Conference on Autonomous Agents and Multiagent Systems*, Montreal, Canada, (2019).
- [2] Evgenii Balai, Michael Gelfond, and Yuanlin Zhang, ‘Towards Answer Set Programming with Sorts’, in *Conference on Logic Programming and Nonmonotonic Reasoning*, (2013).
- [3] Marcello Balduccini and Michael Gelfond, ‘Logic Programs with Consistency-Restoring Rules’, in *AAAI Spring Symposium on Logical Formalization of Commonsense Reasoning*, (2003).
- [4] Samuel Barrett, Avi Rosenfeld, Sarit Kraus, and Peter Stone, ‘Making friends on the fly: Cooperating with new teammates’, *Artificial Intelligence*, **242**, 132–171, (2017).
- [5] Samuel Barrett, Peter Stone, Sarit Kraus, and Avi Rosenfeld, ‘Teamwork with limited knowledge of teammates’, in *AAAI Conference on Artificial Intelligence*, volume 27, (2013).
- [6] Michael Bowling and Peter McCracken, ‘Coordination and adaptation in impromptu teams’, in *National Conference on Artificial Intelligence*, p. 53–58, (2005).

- [7] Shuo Chen, Ewa Andrejczuk, Zhiguang Cao, and Jie Zhang, 'AATEAM: Achieving the ad hoc teamwork by employing the attention mechanism', in *AAAI*, (2020).
- [8] Hasra Dodampegama and Mohan Sridharan, 'Knowledge-based Reasoning and Learning under Partial Observability in Ad Hoc Teamwork', *Theory and Practice of Logic Programming*, **23**(4), 696–714, (2023).
- [9] Hasra Dodampegama and Mohan Sridharan, 'Reasoning and Explanation Generation in Ad hoc Collaboration between Humans and Embodied AI', in *International Conference on Logic Programming and Non-monotonic Reasoning (LPNMR)*, Dallas, USA, (October 11-14, 2024).
- [10] Hasra Dodampegama and Mohan Sridharan, 'Reasoning with Commonsense Knowledge and Decision Heuristics for Scalable Ad hoc Human-Agent Collaboration', in *Distributed AI Conference (DAI)*, London, UK, (November 21-24, 2025).
- [11] Hasra Dodampegama and Mohan Sridharan. <https://github.com/hharithaki/KAT-VH>, 2026.
- [12] Upol Ehsan, Samir Passi, Q. Vera Liao, Larry Chan, I-Hsiang Lee, Michael Muller, and Mark O Riedl, 'The who in xai: How ai background shapes perceptions of ai explanations', in *CHI Conference on Human Factors in Computing Systems*, (2024).
- [13] Qi Fang, Junjie Zeng, Haotian Xu, Yue Hu, and Quanjin Yin, 'Learning ad hoc cooperation policies from limited priors via meta-reinforcement learning', *Applied Sciences*, **14**(8), (2024).
- [14] Bettina Finzel, David E. Tafler, Stephan Scheele, and Ute Schmid, 'Explanation as a Process: User-Centric Construction of Multi-level and Multi-modal Explanations', in *KI 2021: Advances in Artificial Intelligence*, ed., Stefan Edelkamp and Ralf Möller and Elmar Rueckert, volume 12873, 80–94, Springer International Publishing, (2021).
- [15] Maria Fox, Derek Long, and Daniele Magazzeni, 'Explainable Planning', in *IJCAI Workshop on Explainable AI*, (2017).
- [16] Michael Gelfond and Daniela Incelesan, 'Some Properties of System Descriptions of AL_q ', *Applied Non-Classical Logics, Special Issue on Equilibrium Logic and ASP*, **23**(1-2), (2013).
- [17] Gerd Gigerenzer, *Towards a Rational Theory of Heuristics*, 34–59, Palgrave Macmillan UK, London, 2016.
- [18] Gerd Gigerenzer, 'What is Bounded Rationality?', in *Routledge Handbook of Bounded Rationality*, Routledge, (2020).
- [19] Gerd Gigerenzer and Wolfgang Gaissmaier, 'Heuristic Decision Making', *Annual Review of Psychology*, **62**, (2011).
- [20] Joseph Y. Halpern and Judea Pearl, 'Causes and explanations: A structural-model approach. part i: Causes', *The British Journal for the Philosophy of Science*, **56**(4), 843–887, (2005).
- [21] Subbarao Kambhampati, Karthik Valmeekam, Lin Guan, Mudit Verma, Kaya Stechly, Siddhant Bhambri, Lucas Saldy, and Anil Murthy, 'Position: LLMs Can't Plan, But Can Help Planning in LLM-Modulo Frameworks', in *International Conference on Machine Learning*, (2024).
- [22] Konstantinos Katsikopoulos, Ozgur Simsek, Marcus Buckmann, and Gerd Gigerenzer, *Classification in the Wild: The Science and Art of Transparent Decision Making*, MIT Press, 2021.
- [23] Pat Langley, 'Explainable, Normative, and Justified Agency', in *AAAI Conference on Artificial Intelligence*, pp. 9775–9779, (January 2019).
- [24] Pat Langley, Ben Meadows, Mohan Sridharan, and Dongkyu Choi, 'Explainable Agency for Intelligent Autonomous Systems', in *Innovative Applications of Artificial Intelligence*, San Francisco, USA, (February 4-9, 2017).
- [25] Xinzhu Liu, Peiyan Li, Wenju Yang, Di Guo, and Huaping Liu, 'Leveraging Large Language Model for Heterogeneous Ad Hoc Teamwork Collaboration', in *Robotics: Science and Systems*, Delft, The Netherlands, (2024).
- [26] Tim Miller, 'Explanations in Artificial Intelligence: Insights from the Social Sciences', *Artificial Intelligence*, **267**, 1–38, (February 2019).
- [27] Reuth Mirsky, Ignacio Carlucho, Arrasy Rahman, Elliot Fosong, William Macke, Mohan Sridharan, Peter Stone, and Stefano Albrecht, 'A Survey of Ad Hoc Teamwork: Definitions, Methods, and Open Problems', in *European Conference on Multiagent Systems*, (2022).
- [28] Tiago Mota, Mohan Sridharan, and Ales Leonardis, 'Integrated Commonsense Reasoning and Deep Learning for Transparent Decision Making in Robotics', *Springer Nature CS*, **2**(242), (2021).
- [29] Judea Pearl, *Causality*, Cambridge University Press, 2 edn., 2009.
- [30] Judea Pearl and Dana Mackenzie, *The book of why: the new science of cause and effect*, Allen Lane, 2018.
- [31] Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba, 'Virtualhome: Simulating household activities via programs', in *International Conference on Computer Vision and Pattern Recognition*, pp. 8494–8502, (2018).
- [32] Muhammad A Rahman, Niklas Hopner, Filippos Christianos, and Stefano V Albrecht, 'Towards open ad hoc teamwork using graph-based policy learning', in *International Conference on Machine Learning*, pp. 8776–8786, (18–24 Jul 2021).
- [33] Raymond Reiter, 'A Theory of Diagnosis from First Principles', *Artificial Intelligence*, **32**, 57–95, (1987).
- [34] João G. Ribeiro, Gonçalo Rodrigues, Alberto Sardinha, and Francisco S. Melo, 'Teamster: Model-based reinforcement learning for ad hoc teamwork', *Artificial Intelligence*, **324**, 104013, (2023).
- [35] Cynthia Rudin, 'Stop Explaining Black Box Machine Learning Models for High Stakes Decisions and Use Interpretable Models Instead', *Nature Machine Intelligence*, **1**, 206–215, (2019).
- [36] Cynthia Rudin, Chaofan Chen, Zhi Chen, Haiyang Huang, Lesia Semenova, and Chudi Zhong, 'Interpretable machine learning: Fundamental principles and 10 grand challenges', *Statistics Surveys*, **16**, 1–85, (2022).
- [37] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra, 'Habitat: A platform for embodied ai research', in *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9338–9346, (2019).
- [38] Herbert A. Simon, 'Rational Choice and the Structure of the Environment', *Psychological Review*, **63**, 129–138, (1956).
- [39] Mohan Sridharan, 'Integrated Knowledge-based Reasoning and Data-driven Learning for Explainable Agency in Robotics', in *Explainable Agency in Artificial Intelligence: Research and Practice*, CRC Press, (2024).
- [40] Mohan Sridharan, 'Back to the Future of Integrated Robot Systems', in *AAAI Conference on Artificial Intelligence*, Philadelphia, US, (February 25-March 4, 2025).
- [41] Mohan Sridharan, Michael Gelfond, Shiqi Zhang, and Jeremy Wyatt, 'REBA: A Refinement-Based Architecture for Knowledge Representation and Reasoning in Robotics', *Journal of Artificial Intelligence Research*, **65**, 87–180, (May 2019).
- [42] Mohan Sridharan and Ben Meadows, 'Knowledge Representation and Interactive Learning of Domain Knowledge for Human-Robot Collaboration', *Advances in Cognitive Systems*, **7**, 77–96, (December 2018).
- [43] Mohan Sridharan and Ben Meadows, 'Towards a theory of explanations for human-robot collaboration', *KI - Künstliche Intelligenz*, **33**, 1–12, (09 2019).
- [44] Peter Stone, Gal Kaminka, Sarit Kraus, and Jeffrey Rosenschein, 'Ad Hoc Autonomous Agent Teams: Collaboration without Pre-Coordination', in *AAAI Conference on Artificial Intelligence*, pp. 1504–1509, (2010).
- [45] Penghui Xu, Yu Zhang, Le Hao, and Qilin Yan, 'DETEAMSK: A Model-Based Reinforcement Learning Approach to Intelligent Top-Level Planning and Decisions for Multi-Drone Ad Hoc Teamwork by Decoupling the Identification of Teammate and Task', *Aerospace*, **12**(7), 1–32, (2025).
- [46] Luisa Zintgraf, Sam Devlin, Kamil Ciosek, Shimon Whiteson, and Katja Hofmann, 'Deep interactive bayesian reinforcement learning via meta-learning', in *International Conference on Autonomous Agents and Multiagent Systems*, (May 2021).

TRUSTWORTHY MULTI-AGENT SYSTEMS: MITIGATING SEMANTIC DRIFT WITH THE ARGENT SIGNALING PROTOCOL

XTAI 2026, AISB CONVENTION, UNIVERSITY OF SUSSEX, UK

Anantha Sharma 
Synechron Inc
Charlotte, NC, USA

July 2026

ABSTRACT

When multi-agent LLM systems produce bad answers, not all failures are equal: some answers are grounded in the right material but incomplete, while others are simply ungrounded and should be stopped. Current retry strategies treat both cases identically, leaving human supervisors unable to tell whether a retry was warranted or whether the system should have halted.

We introduce the *Argent* Signaling Protocol (ASP) (*argent* as in silver, signaling clarity and audit-grade transparency; distinct from Answer Set Programming). ASP is a compact machine-readable header that accompanies every AI-generated response with structured quality signals: certainty (@C), grounding (@G), stochasticity (@S), and an assumption index that classifies the evidentiary basis of each claim. These signals let a controller distinguish *repairable* failures from *containment* failures and route each case differently.

We evaluate ASP in two complementary settings: a standalone controller for document-grounded QA and a sidecar-gated two-agent pipeline. Standalone, ASP raises aggregate passes from 12/81 to 21/81 on the 27-question Array BioPharma/Ono benchmark (10 recoveries, 1 regression; aggregate one-sided McNemar $p=0.006$) and from 25/429 to 34/429 on a 19-contract CUAD breadth extension (9 recoveries, 0 regressions; $p=0.002$). On heterogeneous CUAD the utility gain is modest (+2.1 absolute points aggregate); the controller’s value shifts from repair to containment, with the safety profile preserved. A single-model multi-agent case study (SmolLM3, 27 questions, 14 ungrounded baseline outputs) shows the ASP sidecar blocking 24/27 outputs and preventing every ungrounded one from reaching the downstream decision agent, controlling for semantic drift before it propagates across agent boundaries.

Keywords Multi-Agent Systems · Explainable AI · Semantic Drift · Uncertainty · Grounded Generation · Large Language Models

1

1 Introduction

As Large Language Models (LLMs) move from isolated prompting to orchestrated Multi-Agent Systems (MAS), intermediate outputs increasingly function as operational state rather than disposable prose [1, 2]. In *document-grounded* settings, downstream agents may act on upstream answers that are only partially supported, poorly cited, or already drifting away from the source text. The problem is not only whether an answer is wrong, but whether the system can tell what kind of wrong it is before that answer is reused.

¹An extended version of this paper including the full protocol specification (the stochasticity-modulated centroid update, adaptive assumption-decay mechanism, and the full per-signal weight rationale), all four trajectory cards, and detailed per-model analyses will be made available on arXiv.

Consider an AI system answering legal questions over a pharmaceutical license agreement. One answer cites the correct section (“Section 1.6, Change in Control”) and recovers some of the required terms, but misses branches about mergers and asset sales. Another produces fluent legal-sounding prose with no citations and no grounding. These failures are operationally different. The first is *repairable*: the right material was retrieved, and a second attempt may recover the missing branches. The second requires *containment*: retrying an answer that was never grounded is more likely to polish fabrication than fix it. Most retry strategies, however, treat both failure modes the same.

The accumulation of untracked assumptions is *semantic drift*: locally plausible steps compound into globally invalid trajectories because downstream agents consume upstream assertions as if they were settled facts. A familiar human analogue is the children’s game of “telephone” (sometimes called “Chinese whispers”), where each retelling can introduce small distortions that accumulate into a final message far from the original. This problem is related to calibration and uncertainty in neural models [3, 4], but multi-agent systems reproduce the same dynamic at much higher speed and scale by recursively reusing model outputs as fresh context, allowing drift to propagate across a pipeline before a supervisor can intervene. The result is not only incorrect final answers, but also a loss of auditability about whether the system is repairing grounded partial answers, refining unsupported ones, or silently giving up. Explainable and risk-managed deployments therefore require message-level traceability, interpretable failure state, and controls that are measurable rather than purely rhetorical [5, 6]. ASP extends the documentation logic behind Datasheets for Datasets [7] and FactSheets [8] from datasets and services to individual agent-to-agent messages.

This gap can be addressed using the Argent Signaling Protocol (ASP), a lightweight protocol that attaches structured quality signals to every AI-generated response. ASP builds on the Argent Framework [9], which introduced governance-first primitives for composing, orchestrating, and governing enterprise AI agents. Where the Argent Framework defines how agents are built and bounded, ASP defines how they communicate grounding, certainty, stochasticity, and assumptions, so a governing controller can distinguish repairable outputs from containable ones instead of relying on opaque retry loops.

Contributions

This paper makes three contributions:

- Define ASP and its sidecar deployment model.
- Evaluate ASP as a standalone controller on a 27-question grounded-QA benchmark and a 19-contract / 143-question CUAD breadth extension, where aggregate passes rise from 12/81 to 21/81 on the deep dive (10 fail-to-pass recoveries, 1 regression) and from 25/429 to 34/429 on the breadth test (9 fail-to-pass, 0 regressions).
- Present a single-model case study of multi-agent governance in which the sidecar blocks every ungrounded upstream output from reaching a downstream decision agent on the 27-question benchmark.

2 Related Work

Agent frameworks CAMEL [2] and AutoGen [1] show that multi-agent decompositions can attack problems a single LLM call cannot. Both exchange free-form language or task-specific tool calls; neither carries a standardized uncertainty or provenance contract. When Agent A tells Agent B “the contract defines Change in Control in Section 1.6,” Agent B cannot tell whether Agent A was quoting retrieved evidence or generating from memory. The Argent Framework [9] addresses agent composition and budget-bounded execution at the orchestration level; ASP occupies the protocol layer between the agent and its governor.

Self-correction Self-Refine [10] and Reflexion [11] demonstrate that feedback-driven retries can improve answers. In high-stakes settings the feedback stays rhetorical: after three retries an auditor sees that the system tried three times but cannot tell whether any attempt was grounded, whether confidence rose or fell across attempts, or whether halting earlier would have been safer. ASP makes those signals explicit and machine-readable, so every retry carries its own quality report card.

Calibration and uncertainty Modern neural networks are often miscalibrated [3]; LLMs sometimes know what they know under the right elicitation protocol [4]; and verbalized confidence can diverge from true correctness [12, 13]. ASP does not claim access to true posterior confidence. It operates on auditable proxies of grounding and instability and asks whether those proxies are sufficient, in practice, to route between repair and containment.

Documentation and reporting standards Datasheets for Datasets [7] and FactSheets [8] standardize documentation for training data and AI services respectively. ASP extends that practice to the inter-agent communication layer: each message carries its own provenance, confidence, and assumption metadata, so the entire multi-agent conversation is auditable as a chain of documented decisions.

Table 1 compares ASP to the closest related systems along four axes. To our knowledge ASP is the first agent protocol that exposes all four at the message boundary.

3 The Argent Signaling Protocol

The core idea is simple: before accepting any AI-generated answer, attach a machine-readable “report card” that rates the answer on multiple dimensions. A supervisor (human or automated) can read the card at a glance to decide whether to trust the answer, ask for a revision, or discard it. As in clinical lab reports, the header carries quality flags alongside the result. At each step the controller scores the current draft against a fixed ideal QA reference profile (§4.2) and updates per-agent error counters; ASP exposes the minimal structured state needed to update these quantities deterministically, so every routing decision can be reproduced from the telemetry log.

Protocol scope vs. benchmark scope ASP is defined at the protocol level, and this paper evaluates it across both standalone and multi-agent settings. The benchmarks in Sections 5–7 exercise the header format (§3.1), the signal fields (§3.2), the assumption index (§3.4), the drift monitor (§4.2), and the routing logic (§4.3). The same protocol is studied in two operational regimes: bounded controller loops for document-grounded QA and sidecar-mediated agent-to-agent traffic, letting us evaluate ASP both as a per-answer control surface and as an inter-agent governance layer.

3.1 Header Format

Every ASP-instrumented response carries a compact header alongside its natural-language content:

@C	@G	@S	A (Assumption Index)
X	Y	Z	[K:id, L:id, P:id, H:id]

On the wire, the same header is serialized as `[@C:X; @G:Y; @S:Z; A:[K:id, L:id, P:id, H:id]]`. Each signal field (@C, @G, @S) is a hexadecimal digit from 0 (lowest) to F (highest, i.e. 15), representing a quality score on a 16-point scale. We use 16 levels so each signal fits in a single hexadecimal digit, preserving a compact fixed-width header; the controller itself operates on the underlying continuous values, so the serialization choice does not constrain the routing logic. The A: [...] field is the assumption index. A well-grounded answer that cites the right source might carry `@C:D; @G:F; @S:2; A:[K:42, L:09]`: certainty high, grounding maximal, stochasticity low, one known fact and one learned derivation. Contrast `@C:3; @G:1; @S:C; A:[H:17]`: certainty very low, grounding nearly absent, stochasticity high, only a hypothetical assumption. The controller does not need to read the answer text to know this response should not be passed downstream. The header is compact, auditable (every field mechanically derivable, no opaque learned components), and protocol-level (it travels with the response through any downstream agent). We prioritize interpretability over optimality: a controller that makes the right decision 85% of the time and can explain each one is more useful in regulated settings than one that makes the right decision 90% of the time opaquely.

3.2 Signal Fields

The three signal fields capture complementary dimensions of response quality. **@C (Certainty)** is a controller-visible certainty proxy for answer reliability; in the benchmark, when no separate confidence channel is available, Equation 3 derives it from the grounding and stochasticity proxies, so it should not be read as an empirically calibrated probability. **@G (Grounding)** reports how well the answer is anchored in retrieved evidence; @G of F means nearly every word in the answer appears in the source chunks, @G of 0 means the answer shares almost no vocabulary with the source. **@S (Stochasticity)** reports how much of the answer is unsupported or unstable; a high @S is a warning sign that the model is generating material that cannot be verified against the source.

3.3 Estimating ASP Fields from Closed-API Agents

ASP specifies the signal contract, not the estimator. Candidate black-box estimators include verbalized confidence for @C [13, 14], NLI evidence coverage for @G, and k -sample disagreement for @S [12] — none requires logit access. We outline this family to indicate the intended deployment surface; we do *not* validate it here. Every result in this paper

Table 1: ASP versus closely related multi-agent and self-correction systems. ✓ = present at the protocol or orchestration level; × = absent; partial = available as natural-language critique without machine-readable fields.

System	Structured uncertainty	Provenance / assumption channel	Drift monitor	Auditable circuit-break
AutoGen [1]	×	×	×	×
CAMEL [2]	×	×	×	×
Self-Refine [10]	partial	×	×	×
Reflexion [11]	partial	×	×	×
ASP (this work)	✓ (@C, @S)	✓ (@G, A:[K,L,P,H])	✓ (JSD, τ)	✓ (Eq. 6)

uses the white-box token-overlap estimator of Section 3.2, chosen for transparency. The closed-API path is therefore proposed rather than demonstrated, and its calibration against the white-box estimator is future work.

3.4 Assumption Index

Beyond the three numeric signals, ASP provides a structured *assumption index* that classifies the evidentiary basis of each claim. Every assumption referenced in the answer is tagged with one of four categories (Table 2). A response built on K (known) and L (learned) assumptions is citing and interpreting the actual document; a response built on H (hypothetical) is speculating. The controller can decide: repair when the answer is mostly K/L with missing terms; containment when it is mostly H with no grounding.

Table 2: Assumption categories in the ASP header

Tag	Category	Meaning
K	Known	Established fact directly stated in the source document.
L	Learned	Derived during processing (e.g., identifying a chunk with the definition).
P	Projected	Inferred from partial evidence; the source suggests but does not state it.
H	Hypothetical	Speculative; no direct evidence; the model is reasoning beyond the source.

3.5 Sidecar Deployment Model

ASP is an interception layer between agents rather than inside them, analogous to service-mesh sidecars. Figure 1 shows the architecture: a single shared bus with one tap per agent. An agent’s only outbound channel is its tap; there is no direct agent-to-agent path. The bus computes @C/@G/@S, drift, and the combined assumption index, and is the sole enforcement surface for inter-agent traffic. Agents can be swapped without changing the bus, and vice versa.

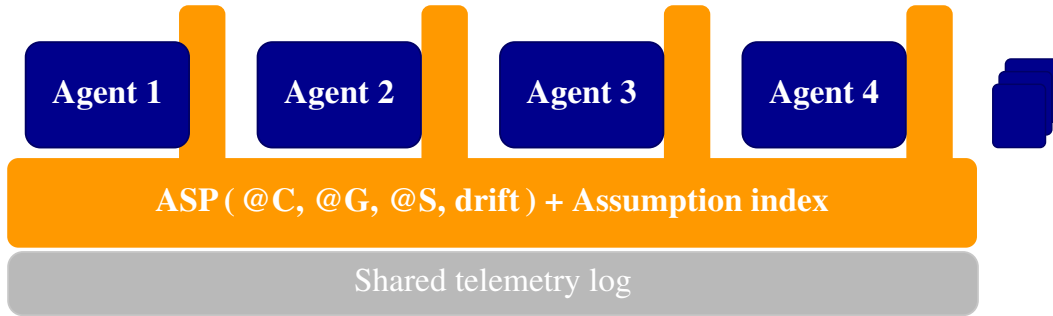


Figure 1: Sidecar-mediated architecture. The ASP layer (orange) is a shared bus that all agents tap into via dedicated sidecars; agents have no direct path to one another, so every inter-agent message is forced through the bus, which acts as the sole enforcement surface.

Figure 2 shows the operational pipeline: Agent A’s output passes through ASP Sidecar 1 (computes @C/@G/@S and drift, makes a deterministic gate decision); Agent B’s output is similarly gated. Sidecar-triggered regenerations are bounded by the error budget ($E_{\max} = 2$) and a flow-level TTL; exhausted items go to a human review queue. Every

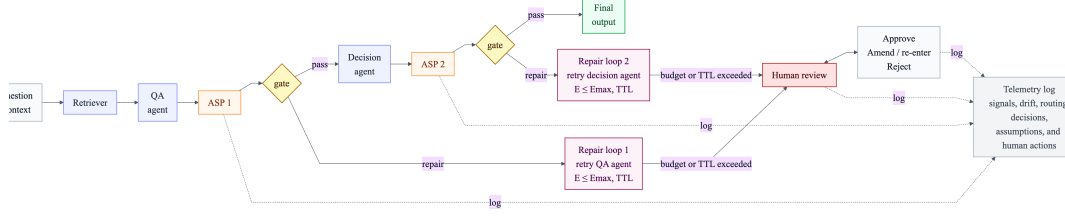


Figure 2: Operational view of the same architecture as Figure 1, per agent boundary. “ASP Sidecar 1/2” are not separate buses — each is one tap into the shared bus of Figure 1, shown twice because each boundary is gated independently. Failed items enter bounded repair loops; unresolved cases escalate to human review; telemetry records every decision.

sidecar decision, repair attempt, and human action is logged. ASP operates in two modes within the same deployment: standalone (single agent with a controller loop, §6) and multi-agent (sidecars at every boundary, §7).

4 Controller Implementation

This section describes how ASP signals are computed and used in the benchmark controller. The protocol (§3) defines *what* signals are carried; this section defines *how* they are derived and *how* the controller acts on them. The same protocol could be implemented with different estimators; the benchmark uses a token-overlap estimator because it is transparent, reproducible, and sufficient for document-grounded QA.

4.1 Signal Computation

For each generated draft, three raw features are extracted by comparing the answer tokens against the source material:

- o_c (*context overlap*): fraction of answer tokens appearing in any retrieved chunk.
- s_{cite} (*citation score*): 1.0 for a valid retrieved-chunk citation, 0.4 for a citation not in the retrieved set, 0.0 otherwise.
- n (*novel_ratio*): fraction of answer tokens found in neither the retrieved context nor the question (excludes question-echo tokens).

A fourth feature, o_q (*question overlap*), feeds the drift vector but not the signal fields directly. Grounding and stochasticity are then:

$$@G = \text{clamp}(0.60 \cdot o_c + 0.40 \cdot s_{cite}) \quad (1)$$

$$@S = \text{clamp}(0.70 \cdot n + 0.30 \cdot (1 - s_{cite})) \quad (2)$$

If the agent exposes an explicit confidence signal $c_{conf} \in [0, 1]$, the controller sets $@C = \text{clamp}(c_{conf})$. When no such signal is available, it falls back to

$$@C = @G \cdot (1 - @S) \quad (3)$$

where $\text{clamp}(x) = \min(1, \max(0, x))$. Grounding is weighted toward o_c because using the source document’s language is the strongest evidence of document awareness; stochasticity is weighted toward n because fabricated material is the primary risk in grounded QA; the fallback certainty rule keeps $@C$ low whenever grounding is weak or novelty is high. In the benchmark reported here, the local models did not emit a separate confidence channel, so Equation 3 supplied $@C$. Each value is compressed into a single hexadecimal digit for the ASP header; the controller operates on the raw floats.

4.2 Drift Monitoring

Individual signal values describe a single draft but do not reveal a trend. Drift monitoring measures how far each answer’s quality profile sits from what a well-grounded response typically looks like. Jensen–Shannon divergence (JSD) [15] is used as a bounded, symmetric drift score on the L1-normalized feature vector. This is a computational choice, not an information-geometric claim: $\mathbf{v}$ is a heterogeneous quality-feature vector, not a parametric distribution over a task sample space. With base-2 logarithms, $\text{JSD}(P||Q) \in [0, 1]$, and $\sqrt{\text{JSD}}$ is a proper metric on normalized vectors [16].

Each draft produces a four-element quality-profile vector. Components are floored at 0.01 and then apply L1 normalization:

$$\tilde{\mathbf{v}} = (\max(0.01, o_c), \max(0.01, s_{\text{cite}}), \max(0.01, n), \max(0.01, o_q)), \quad \mathbf{v} = \frac{\tilde{\mathbf{v}}}{\|\tilde{\mathbf{v}}\|_1}. \quad (4)$$

Drift is the square root of the JSD between $\mathbf{v}$ and a fixed ideal QA reference $\mathbf{v}^* = (0.45, 0.35, 0.03, 0.17)$ (high context overlap, strong citation, low novel content, moderate question overlap):

$$d = \sqrt{\text{JSD}(\mathbf{v} \parallel \mathbf{v}^*)} = \sqrt{\frac{1}{2} [\text{KL}(\mathbf{v} \parallel \mathbf{m}) + \text{KL}(\mathbf{v}^* \parallel \mathbf{m})]}, \quad \mathbf{m} = \frac{\mathbf{v} + \mathbf{v}^*}{2} \quad (5)$$

In the benchmark, a Qwen answer that cites the right chunk and uses the source language produces $d \approx 0.11$; a Dobby answer that cites correctly but misses required terms produces $d \approx 0.20$; a SmoLLM3 answer that paraphrases without citations produces $d \approx 0.62$. Because JSD is bounded and symmetric, 0 means identical to the ideal and 1 means maximally different.

Sensitivity to miscalibrated confidence signals A reasonable concern is what happens when the elicited certainty signal is itself miscalibrated. The routing rule (Eq. 6) consumes raw grounding g , stochasticity s , and drift d ; the certainty field @C is reported in the header for downstream supervisors but is not a routing input, and the drift computation (Eqs. 4–5) operates over independent token-level features ($o_c, s_{\text{cite}}, n, o_q$) — @C is absent from $\mathbf{v}$. Verbal-confidence bias of the form $@C_{\text{obs}} = @C_{\text{true}} + \varepsilon$ therefore does not enter the drift threshold check at all, and per-agent bias in @C would be detectable through cross-agent disagreement in a multi-agent deployment. The relevant residual sensitivity is to noise in the token-level features themselves (most acutely n , the novel-token ratio, which is a small set-membership count): characterising $\partial d / \partial \mathbf{v}$ near the threshold and translating estimator noise on each component into a flip rate requires an ablation we do not perform here; it is a target for the follow-on sensitivity work noted in Section 4.4.

4.3 Routing and Failure Classification

Given the signal values and drift score, the controller applies a deterministic routing policy as a priority cascade, with the first satisfied clause taking precedence:

$$R(\text{agent}) = \begin{cases} \text{Yield,} & \text{if } E \geq E_{\text{max}} \\ \text{Regenerate,} & \text{if } E < E_{\text{max}} \wedge (g < g_{\text{min}} \vee s > \sigma_{\text{max}} \vee d > \tau_{\text{crit}}) \\ \text{Warn,} & \text{if } E < E_{\text{max}} \wedge g \geq g_{\text{min}} \wedge s \leq \sigma_{\text{max}} \wedge \tau_{\text{warn}} < d \leq \tau_{\text{crit}} \\ \text{Execute,} & \text{if } E < E_{\text{max}} \wedge g \geq g_{\text{min}} \wedge s \leq \sigma_{\text{max}} \wedge d \leq \tau_{\text{warn}}. \end{cases} \quad (6)$$

Here g and s are the raw continuous grounding and stochasticity values; hex quantization is only a serialization convention for the header. The cascade makes the precedence explicit: exhausted error budget dominates everything, hard grounding or stochasticity failures force regeneration, and the warn band is reached only after those guardrails are satisfied. In this benchmark, $\tau_{\text{warn}} = 0.12$, $\tau_{\text{crit}} = 0.22$, $g_{\text{min}} = 0.33$, $\sigma_{\text{max}} = 0.67$, and $E_{\text{max}} = 2$, giving each question at most three drafts. Unlike opaque retry loops, each regeneration emits a fresh ASP header and the telemetry records which traces were rejected and why.

Failing answers are classified into regimes by their grounding profile.

- *Grounded partial*: citation matches and content overlaps the right material but does not cover enough required terms; controller target is *repair*.
- *Citation gap*: overlap ≥ 0.65 and term coverage ≥ 0.25 but wrong or missing citation; also *repair*.
- *Ungrounded*: insufficient grounding to justify refinement.

The first two are typically incomplete drafts that benefit from a second attempt; the third is typically reasoning-mode paraphrase that retrying will not fix.

4.4 Parameter Selection and Computational Overhead

ASP has multiple hand-tuned parameters: the linear weights in Eqs. 1–2, the ideal vector $\mathbf{v}^*$, the routing thresholds, and the error budget $E_{\text{max}} = 2$. All values were set before benchmark execution from the qualitative rationale (grounding weighted toward source-language reuse, stochasticity toward fabrication risk); no parameter was adjusted in response to benchmark results. Sensitivity to these choices is the primary target for follow-on ablation work.

Computational overhead ASP header computation is $O(|\text{answer}| \cdot |\text{context}|)$ token comparison plus a fixed-size JSD evaluation. On the M-series Mac used in the benchmark, per-message sidecar overhead is sub-millisecond (a 30-token answer against 660 tokens of context completes in under 1 ms, dominated by tokenization). The wall-clock cost of a repair loop is dominated by re-generation time (on the order of seconds per retry for Qwen 0.8B), not by the sidecar itself. The protocol is therefore deployable in latency-sensitive settings; the dominant cost is whatever the underlying agent’s generation cost is.

5 Experimental Setup

Reproducibility The ASP reference implementation, protocol documentation, and example telemetry are publicly available at <https://github.com/ananthasharma/argent-signaling-protocol>. The experimental benchmark harness, the 27-question Array/Ono evaluation set, the CUAD question-generation pipeline, the controller configuration (thresholds, weights, ideal vector $\mathbf{v}^*$), and the local GGUF model identifiers were developed in a separate research repository and are not part of the current public release. The Array BioPharma/Ono license agreement is a publicly filed SEC exhibit, and the CUAD corpus is publicly available [17].

5.1 Task and Document

The controller is tested using a publically available pharmaceutical license agreement between **Array BioPharma** and **Ono Pharmaceutical**. Legal agreements are a good stress test for grounded QA because they contain multi-branch definitions, conditional provisions, and cross-referenced terms that expose both retrieval and generation failures. The task is a 27-question document-grounded QA benchmark, with questions spanning six categories (Table 3). Each question specifies an expected answer, a set of 3–4 required terms, and citation references that identify the authoritative source text.

5.2 Retrieval, Generation, and Evaluation

Evidence is retrieved via TF-IDF over the agreement chunked at ~ 220 words with paragraph overlap, with definition-aware bonuses (+1.0 for “X means” patterns, +0.5 for other definition syntax) to surface formal definitions. The top- $k = 3$ chunks are returned. This sparse approach is sufficient for single-section definitions but misses multi-branch clauses spanning non-adjacent paragraphs; the “Change in Control” definition is one such case. Domain-aware embedding retrofitting [18] is one mitigation. Each question is answered with a maximum token budget of 96 tokens; temperature is 0.1 on the first attempt and 0.05 on retries. Comparison is between a **Baseline** condition (one grounded answer attempt, no controller) and an **ASP controller** condition (the same initial generation followed by controller assessment, which decides whether to accept, warn, regenerate, or yield). An answer *passes* if it cites at least one chunk matching the expected citation *and* contains at least 60% of the required terms (case-insensitive). **This is a deliberately strict rubric.**

5.3 Models

The benchmark evaluates three local GGUF models chosen to span capability levels and produce qualitatively different failure modes: Qwen3.5-0.8B-BF16 (compact instruction-tuned, follows instructions well, failures tend to be incomplete rather than fabricated); *dobby-8b-unhinged-q6_k* (8B parameters, not instruction-tuned for document QA, produces coherent grounded answers on a meaningful fraction of questions with incomplete-coverage failures); and SmoLLM3-3B-Q8_0 (moderate instruction following, alternates between grounded answers and uncited reasoning-mode paraphrase). Qwen and Dobby ran on MLX (Apple’s optimized framework for Mac hardware); SmoLLM3 used llama.cpp. The primary metrics are pass rate, citation match rate, mean required-term coverage, and controller action counts.

6 Results: Standalone Controller

This section reports standalone-controller results in three steps: a deep dive on the Array/Ono benchmark, two qualitative trajectory cards from that same benchmark, and a breadth test over 19 CUAD contracts.

6.1 Array/Ono Deep Dive

Table 4 reports per-model outcomes on the 27-question Array/Ono benchmark. Across the three models, the benchmark covers 81 model-question pairs. Aggregate passes move from 12/81 in baseline to 21/81 under ASP, with 10 fail-to-pass

Table 3: Question categories in the 27-question Array/Ono benchmark.

Category	Count
Definitions (Change in Control, Field, First Commercial Sale, etc.)	7
License scope (development, manufacturing, sublicense, grant-back)	6
Termination and wind-down	4
General provisions (governing law, amendment, assignment, etc.)	6
Governance (JDRC establishment and duties)	2
Risk allocation (recall control, indemnification)	2

recoveries and 1 regression. The controller applies 58 repair interventions, 12 containment interventions, and 11 clean pass-throughs. *Pass-through* is a controller decision (the routing cascade of Eq. 6 yields EXECUTE on the first draft: $g \geq g_{\min}$, $s \leq \sigma_{\max}$, $d \leq \tau_{\text{warn}}$); a *pass* is a benchmark outcome (citation match plus $\geq 60\%$ required-term coverage). These are distinct events by construction, but on this benchmark every pass-through also satisfied the benchmark criterion (all 11 pass-throughs are also passes) so the two counts coincide here.

ASP plays two distinct operational roles across the three models. On Qwen and Dobby, both producing 0% ungrounded outputs, ASP acts as a *repair controller*: Qwen passes rise from 3/27 to 9/27 and Dobby from 9/27 to 12/27. The Qwen cell and the aggregate (12/81 $\rightarrow$ 21/81, $p=0.006$) are individually significant; the Dobby cell is not significant at $n = 27$ and should be read as part of the aggregate signal. Qwen’s 29-point coverage gain (36.7% to 65.4%) is the strongest single effect; Dobby shows a small citation-precision tradeoff (70.4% to 66.7%) and one regression where regeneration broadens coverage by drawing on adjacent chunks.

On SmolLM3 (44.4% ungrounded), ASP improves nothing the end user sees as a passed answer (0/27 either way); its operational value is preventing the 12 ungrounded answers from being retried into more polished fabrication. The 15 repair attempts on grounded-but-incomplete drafts produce no additional passes; the 12 containment events are the result that matters for governance.

6.2 Signal Trajectory Analysis

Two representative controller trajectories from the Array/Ono benchmark make the signal values concrete. Required terms are listed; matched terms appear in **bold**. The first illustrates repair on Qwen; the second, containment on SmolLM3.

Q: When does Ono indemnify Array?		Qwen 0.8B (repair → pass)			
<i>Required terms:</i> Third Party Claim, Ono Territory, clinical studies, gross negligence or willful misconduct					
Baseline Response		@G=F	@S=0	d=0.176	cov=50%
Ono shall indemnify Array against Third Party Claims arising from the use, marketing, distribution, or sale of Array Products, injury to patients in Clinical Studies , or breaches of representations and warranties by Array, as defined in Article 15 [Chunk 281].					
After ASP Controller		@G=F	@S=1	d=0.278	cov=75%
Ono shall indemnify Array against Third Party Claims arising from the use, marketing, distribution, importation, or sale of Array Products, injury to patients in Clinical Studies , or breaches of representations and warranties by Array, as defined in Article 15 [Chunk 281]. Additionally, Ono shall indemnify Array against claims resulting from the gross negligence or willful misconduct of an Array Indemnitee [Chunk 280].					
Outcome: PASS. Repair added Chunk 280 (exception clause), recovering one missing required term. “Ono Territory” was never recovered.					

Table 4: Per-model outcomes on the 27-question Array/Ono benchmark. Rates are percentages. Action profile reports repair interventions (R), containment interventions (C), and clean pass-throughs (P). p is the one-sided McNemar exact test on paired baseline-vs-ASP outcomes; only Qwen and the aggregate clear $\alpha = 0.05$ at this sample size.

Model	Backend	Pass	Cite	Coverage	p	Actions
Qwen 0.8B	MLX	11.1 $\rightarrow$ 33.3	55.6 $\rightarrow$ 59.3	36.7 $\rightarrow$ 65.4	0.016	24R / 0C / 3P
Dobby 8B	MLX	33.3 $\rightarrow$ 44.4	70.4 $\rightarrow$ 66.7	48.8 $\rightarrow$ 54.3	0.19	19R / 0C / 8P
SmolLM3 3B	llama.cpp	0.0 $\rightarrow$ 0.0	18.5 $\rightarrow$ 18.5	24.7 $\rightarrow$ 25.6	—	15R / 12C / 0P

Q: What counts as a Change in Control?		SmolLM3 3B (containment → yield)			
<i>Required terms:</i> fifty percent, majority, merger, substantially all					
Baseline Response		@G=3	@S=A	d=0.620	cov=0%
<i>⟨think⟩ ... A Change in Control would typically involve a shift in ownership or management... [reasoning-mode paraphrase, no chunk citations, no source terms]</i>					
After ASP Controller (attempt 3)		@G=2	@S=B	d=0.650	cov=0%
<i>⟨think⟩ ... Change in Control generally refers to an acquisition or transfer of control... [continued reasoning-mode paraphrase, no chunk citations]</i>					
Outcome: YIELD. Signals degraded across attempts: @G dropped 3→2, @S rose A→B, drift increased 0.620→0.650. The model stayed in uncited reasoning mode, producing plausible-sounding text with no connection to the retrieved chunks. Correctly routed to containment.					

Qwen’s high @G and low @S keep it in the repair regime; SmolLM3’s degraded signals (@G in 2–3, @S in A–B) trigger containment.

6.3 Breadth Test: 19 CUAD Contracts, 143 Questions

We then test whether the same controller behavior generalizes beyond a single agreement. We extend the standalone benchmark to a stratified 19-contract sample drawn from the CUAD corpus [17], balanced across four file-size quartiles and 14 contract types (license, services, distribution, sponsorship, manufacturing, hosting, outsourcing, joint venture, and others). Questions are auto-generated from CUAD’s per-contract clause annotations, yielding 143 questions across 10 categories: IP (30), termination (23), general provisions (17), risk allocation (16), restrictions (15), financial (13), metadata (9), definitions (7), governance (7), and rights (6). Each question is answered by the same three models under both conditions, totaling 429 model-question runs. The same controller configuration, retrieval pipeline, and pass criteria are reused unchanged; there is no per-document tuning.

Three findings dominate. First, the *utility* gain shrinks: from +11.1 absolute pass-rate points on the Array deep dive to +2.1 points aggregate on CUAD (5.8% $\rightarrow$ 7.9%). Per-cell movement is mostly not significant at $n = 143$ (Dobby $p = 0.25$, SmolLM3 $p = 0.50$; only Qwen reaches $p = 0.016$); the aggregate one-sided McNemar exact is $p = 0.002$, but per-model cells should not be read as individually demonstrated improvements. Second, the *safety* profile is preserved: zero regressions across 429 runs, and the single Array-Dobby regression is not reproduced. Third, the controller’s operational regime inverts: repair-dominant on Array (58R / 12C) and containment-dominant on CUAD (9R / 395C, 92.1% containment). On heterogeneous contracts the controller’s value shifts from *repair* to *containment* — the QA-improvement effect mostly evaporates while the safety and failure-mode-legibility properties transfer. All 9 issued repair attempts converted fail-to-pass.

Backend note for Dobby Dobby was evaluated on llama.cpp for the CUAD benchmark; the MLX path produced deterministic token-soup output across all 143 CUAD questions despite working on the Array document with the same model file (29 unique 80-char answer prefixes across 143 outputs). This is treated as a property of the inference stack rather than the protocol — ASP’s drift and grounding signals would have flagged such failures regardless.

7 Results: Multi-Agent Pipeline

The focus here is to test if ASP prevents semantic drift *across agent boundaries*, rather than only within a single agent’s retry loop. We evaluate a two-agent pipeline where an upstream retrieval agent (Agent A) answers a question and a downstream decision agent (Agent B) must make a risk assessment based on that answer.

Table 5: Standalone controller on the CUAD breadth benchmark (19 contracts, 143 questions $\times$ 3 models = 429 runs). Rates are percentages. Action profile: repair (R), containment (C), pass-through (P). p is the one-sided McNemar exact test; only Qwen and the aggregate reach $\alpha=0.05$.

Model	Backend	Pass	Cite	Coverage	p	Actions	FtP / Reg
Qwen 0.8B	MLX	7.0 $\rightarrow$ 11.2	30.1 $\rightarrow$ 32.2	25.9 $\rightarrow$ 35.3	0.016	6R / 127C / 10P	6 / 0
Dobby 8B	llama.cpp	7.0 $\rightarrow$ 8.4	25.9 $\rightarrow$ 25.9	31.5 $\rightarrow$ 35.0	0.25	2R / 131C / 10P	2 / 0
SmolLM3 3B	llama.cpp	3.5 $\rightarrow$ 4.2	9.8 $\rightarrow$ 9.1	22.6 $\rightarrow$ 28.3	0.50	1R / 137C / 5P	1 / 0
Aggregate		5.8 $\rightarrow$ 7.9	21.9 $\rightarrow$ 22.4	26.7 $\rightarrow$ 32.9	0.002	9R / 395C / 25P	9 / 0

7.1 Pipeline Design

Agent A is the same retrieval QA agent from the controller benchmark. Agent B receives Agent A’s output and must classify the provision as HIGH/MODERATE/LOW RISK with a one-sentence justification grounded in the cited terms. We compare two conditions quantitatively. **Baseline:** Agent B receives Agent A’s raw text with no quality metadata. **ASP gated:** the sidecar intercepts Agent A’s output; if grounding falls below $g_{\min}$, stochasticity exceeds $\sigma_{\max}$, or drift exceeds τ_{crit} , the output is *blocked* and Agent B receives an escalation message instead.²

7.2 Pipeline Results

Table 6 and Figure 3 summarize the pipeline benchmark on SmolLM3 (the model with the mixed failure profile, 44.4% ungrounded). In the baseline condition, all 27 outputs (including the 14 ungrounded ones) pass directly to Agent B, which then makes risk assessments based on fabricated or poorly grounded upstream content. In the ASP-gated condition, the sidecar blocks 24 of 27 outputs (including all 14 ungrounded ones); the 3 outputs that pass through are the least-drifted grounded answers.

7.3 Why SmolLM3 Only, and the Scope of This Claim

The pipeline benchmark is restricted to SmolLM3 because Qwen and Dobby produce 0% ungrounded outputs standalone; applying the sidecar to them would trivially pass 100% through. SmolLM3 is the only condition where the gate is exercised — which is also the reason for caution: *this section is a single-model existence proof*. The headline number, 100% of 14 ungrounded outputs blocked, is conditional on those 14 samples from one model on one document. Generalising would require an adversarial protocol that manufactures ungrounded outputs on Qwen and Dobby, or a multi-model pipeline with at least one weak link; both are follow-on work. The 88.9% overall block rate reflects SmolLM3’s failure profile, not the sidecar policy.

7.4 Governance Implication

On this case study, without ASP, 51.8% of upstream outputs propagated ungrounded content to a downstream decision agent; with the ASP sidecar, zero ungrounded outputs reach the downstream agent. The sidecar does not make SmolLM3 more capable (Agent A’s pass rate remains 0%), but it prevents unreliable outputs from contaminating downstream decisions. This illustrates the multi-agent version of the same principle demonstrated in the standalone benchmark: ASP changes how failure propagates, not how often models succeed. Whether the same effect holds across other weak-agent configurations is an empirical question this paper does not answer.

8 Operational Interpretation

8.1 Failure Legibility

The benchmark supports two operational claims:

- ASP produces meaningful aggregate gains (12/81 to 21/81 passes, 10 recoveries).
- ASP changes controller behavior in a way that matches the model’s failure regime.

²In a qualitative probe in which Agent B received the output together with the full ASP header (no hard gate). Agent B’s judgments shifted toward hedged language when the header showed low grounding, suggesting that header visibility alone can change downstream behaviour.

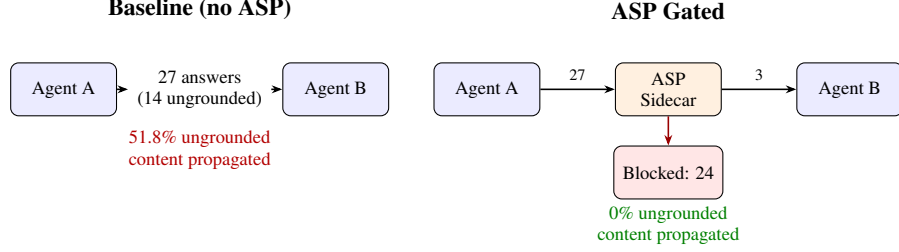


Figure 3: Pipeline comparison. **Left:** without ASP, all 27 outputs (including 14 ungrounded) reach Agent B. **Right:** the ASP sidecar blocks 24 outputs, allowing only 3 grounded answers through. Zero ungrounded content reaches Agent B.

Table 6: Pipeline benchmark results (SmolLM3, 27 questions, risk assessment task).

Metric	Baseline	ASP gated
Agent A outputs reaching Agent B	27/27 (100%)	3/27 (11.1%)
Ungrounded outputs reaching Agent B	14/27 (51.8%)	0/27 (0%)
Blocked by sidecar	0	24/27 (88.9%)
Escalated to human review	0	24

A human supervisor reviewing a batch of 81 outputs without ASP sees only raw results and has no structured way to prioritize which failures to investigate first. With ASP, the same batch decomposes into 11 clean passes, 58 repair attempts (10 of which recovered to passing), and 12 containment events (none worth additional effort). This makes the system’s own failure state legible: Qwen and Dobby outputs are worth repairing; SmolLM3 produces no additional passing answers but its 12 ungrounded outputs are halted before they consume further compute. The controller separates those cases rather than treating all non-passing drafts as equivalent.

8.2 Coverage Gains and Limits

The signal formulas in Eqs. 1–2 and the fallback rule in Eq. 3 are intentionally simple, not learned classifiers — the signals must be auditable without a calibration dataset. The controller therefore cannot distinguish subtle failure modes (e.g., plausible-but-wrong from clearly nonsensical) unless they also diverge in token overlap and citation profile. For this benchmark the separation holds: Qwen coverage rises 36.7% to 65.4% (29 points); Dobby 48.8% to 54.3% with 4 new passes. A reviewer receiving an answer with 65% term coverage is better positioned than one receiving 37%. Whether the separation holds for more nuanced tasks remains open.

9 Limitations

The scope of evidence presented here is narrow in several respects:

- The pass rubric is strict but synthetic. It depends on citation match and required-term coverage rather than human judgment of answer usefulness.
- **Signal/criterion correlation** The controller rewards context overlap (o_c , Eq. 1); the pass criterion rewards required-term coverage ($\geq 60\%$). These are not the same set, but they are correlated by construction (retrieved chunks contain the language the required terms are drawn from) so a regeneration that maximises source-language reuse tends to recover more required terms. Qwen’s 29-point coverage gain is therefore partly the controller selecting drafts favourable under a proxy of the pass criterion, not solely independent quality improvement. Citation match is an orthogonal pass requirement, which bounds the effect, but a fully independent rubric (e.g., human judgement) would be needed to rule it out.
- The signal computation (Eqs. 1–2) uses fixed hand-tuned weights and a hand-picked ideal vector $\mathbf{v}^*$. The thresholds were not perturbed in an ablation; their sensitivity is not characterised in this work.
- The full protocol described in the extended (arXiv) version of this paper additionally includes a stochasticity-modulated centroid update, an adaptive assumption-decay rate, and an active→decayed→quarantined assumption lifecycle for multi-turn deployments. None of these multi-turn mechanisms are evaluated here: the camera-ready benchmark runs bounded at-most-3-draft trajectories and exercises only the per-draft scoring path; their effect on extended dialogues remains future work.

Acknowledgments

The author thanks the anonymous XTAI 2026 reviewers for substantive feedback that sharpened the failure-legibility framing and the closed-API estimator discussion. Any remaining errors are the author’s.

References

- [1] Qingyun Wu et al. AutoGen: Enabling next-gen LLM applications via multi-agent conversation. In *Proceedings of the First Conference on Language Modeling*, 2024.
- [2] Guohao Li et al. CAMEL: Communicative agents for mind exploration of large scale language model society, 2023. arXiv:2303.17760.
- [3] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1321–1330, 2017.
- [4] Saurav Kadavath et al. Language models (mostly) know what they know, 2022. arXiv:2207.05221.
- [5] Alejandro Barredo Arrieta et al. Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*, 58:82–115, 2020.
- [6] Elham Tabassi. Artificial intelligence risk management framework (AI RMF 1.0). Technical Report NIST AI 100-1, National Institute of Standards and Technology, 2023.
- [7] Timnit Gebru, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna Wallach, Hal Daume III, and Kate Crawford. Datasheets for datasets. *Communications of the ACM*, 64(12):86–92, 2021.
- [8] Matthew Arnold et al. FactSheets: Increasing trust in AI services through supplier’s declarations of conformity. *IBM Journal of Research and Development*, 63(4/5):6:1–6:13, 2019.
- [9] Anantha Sharma, Swetha Devabhaktuni, and Eklove Mohan. The Argent framework: A paradigm to compose, orchestrate, and govern enterprise AI agents, 2025. SSRN preprint.
- [10] Aman Madaan et al. Self-refine: Iterative refinement with self-feedback, 2023. arXiv:2303.17651.
- [11] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning, 2023. arXiv:2303.11366.
- [12] Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation, 2023. arXiv:2302.09664.
- [13] Miao Xiong, Zhiyuan Hu, Xinyang Lu, Yifei Li, Jie Zhang, Jinfeng Yi, et al. Can LLMs express their uncertainty? an empirical evaluation of confidence elicitation in LLMs. In *The Twelfth International Conference on Learning Representations*, 2024.
- [14] Katherine Tian, Eric Mitchell, Allan Zhou, Archit Sharma, Rafael Rafailov, Huaxiu Yao, Chelsea Finn, and Christopher D. Manning. Just ask for calibration: Strategies for eliciting calibrated confidence scores from language models fine-tuned with human feedback. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023.
- [15] Jianhua Lin. Divergence measures based on the Shannon entropy. *IEEE Transactions on Information Theory*, 37(1):145–151, 1991.
- [16] Dominik M. Endres and Johannes E. Schindelin. A new metric for probability distributions. *IEEE Transactions on Information Theory*, 49(7):1858–1860, 2003.
- [17] Dan Hendrycks, Collin Burns, Anya Chen, and Spencer Ball. CUAD: An expert-annotated NLP dataset for legal contract review. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021. arXiv:2103.06268.
- [18] Anantha Sharma. Embedding retrofitting: Data engineering for better RAG. SSRN preprint, 2026. URL https://papers.ssrn.com/sol3/papers.cfm?abstract_id=6064714.