ECLIPSE: AN EXECUTABLE CONTEXTUAL INTERVENTION AND PLANNING AGENT FOR TIME-SERIES TASKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Real-world time series cannot be fully characterized by historical observations alone, because their dynamics also depend on sparse contextual clues, external events, and domain knowledge. Existing methods often flatten such context into a prompt and require a single model pass to perform both semantic grounding and numerical extrapolation, leading to brittle reasoning. We propose **ECLIPSE**, an agentic system that recasts context-rich time-series reasoning as executable scenario reconstruction. ECLIPSE assembles a task-relevant scenario state from textual clues, retrieved temporal precedents, and domain priors, then compiles it into a temporal program. A unified planner selects contextual constraints and temporal operations, while foundation models and executable operators perform numerical computation. An audit agent checks contextual and numerical consistency, triggering replanning or a fallback to the unedited forecast when necessary. Across context-rich forecasting and temporal QA benchmarks, ECLIPSE outperforms direct prompting with the same language model, and a controlled comparison under a shared toolchain attributes the gain to the orchestration of history-side and future-side interventions rather than to tool access alone.

1 INTRODUCTION

Historical numerical observations alone are often insufficient for reliable real-world time-series forecasting and reasoning. In practice, the observed history provides only a partial view of the underlying temporal scenario. Future dynamics may depend both on explicit external events, such as weather shifts, holidays, promotions, or outages, and on implicit domain-specific regularities and operational constraints (Williams et al., 2024; Jiang et al., 2025a). For example, traffic forecasting may be affected by accidents or road closures while also exhibiting recurring rush-hour dynamics, retail forecasting may depend on promotions and social trends, and solar-generation forecasting is governed by both weather conditions and physical daylight patterns. Consequently, the central challenge is not merely to treat contextual information as auxiliary input to a forecasting model. Instead, an intelligent system must dynamically infer *which* external events and domain priors are relevant, *whether* their effects are transient or persistent, and exactly *how* they should influence the underlying temporal computation.

This reconstruction view exposes three coupled challenges for context-rich time-series systems. First, external information has task-dependent *relevance*: a large portion of available signals may be irrelevant for a particular forecast or question, and indiscriminately injecting them can distract the model. Second, external information has varying *persistence*: some events are transient, some recurring patterns should be stored as empirical memory, and some conceptual rules remain stable across tasks. Third, external information must be *operationally grounded*: a textual event, retrieved analogy, or domain rule must be mapped to a target variable, temporal window, numerical regime, or answer format. LLMs can map natural language to structured decisions, VLMs can describe visual temporal patterns (Liu et al., 2025; Trabelsi et al., 2025), and retrieval can recover empirical precedents that ground abstract context in concrete numerical patterns (Jiang et al., 2024b; 2025a; Tire et al., 2024; Yang et al., 2024; Ning et al., 2025). Yet representation-level multimodal alignment alone does not directly resolve the execution problem in this setting. The core difficulty is not representational but dynamic: whether and when external information is relevant to a given forecast depends on the current series state, the task horizon, and the nature of the event—and this relevance cannot be resolved by a fixed alignment layer trained on average co-occurrence patterns.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

We therefore treat context-rich time-series reasoning as two coupled steps: scenario-state reconstruction and executable temporal programming. Rather than feeding textual context into a monolithic predictor, the agent assembles a task-relevant information state from sparse clues, retrieved empirical precedents, and stated constraints. It then synthesizes a machine-executable temporal program, so that the language model acts as the planner while quantitative tools carry out the numerical computation.

To instantiate this paradigm, we introduce **ECLIPSE**, a multi-agent system that operationalizes this decoupled architecture. Execution begins with an intent-aware routing agent that assesses the required output format, efficiently bypassing heavy semantic parsing for simple, self-contained queries. For knowledge-augmented tasks that rely on external knowledge for precision, a contextual prior agent constructs a compact guidance state that separates semantic grounding from raw numerical computation. An optional intervention advisor, trained on a small human-reviewed dataset, can further suggest whether a scenario calls for history reconditioning, future constraint enforcement, or both; we evaluate it only with a weaker planner (Qwen3-8B; Appendix C.3.4), and it is disabled in our main experiments. A unified planner agent then synthesizes the temporal program, orchestrating the numerical backbone and the required interventions. Finally, an audit agent checks the program and output for consistency with numerical constraints and contextual logic, triggering re-planning or a fallback when a check fails. Our main contributions are summarized as follows:

- **Context compiled into two-sided temporal operations.** ECLIPSE turns textual context into explicit operations on both sides of a time-series foundation model: *history reconditioning* repairs or enriches the observed history before forecasting, and *future scenario enforcement* applies context-derived events, bounds, and rules to the forecast horizon. Retrieved temporal archetypes serve as priors for history completion, and the same planner interface produces evidence-extraction plans for temporal QA.
- **Evidence that the gain comes from orchestration.** Under a shared toolchain, where all methods use the same language model, forecasting backbone, retrieved archetypes, and temporal tools, ECLIPSE lowers DTR to 0.187 from 0.209–0.236 for single-agent, fixed-program, and retrieval-plus-post-processing controls (Section 5). Paired, tail-robust comparisons show positive gains from both history reconditioning and future enforcement when each is applied.
- **Inspectable execution traces and a diagnosis of failure modes.** Every run records its retrieved references, program, and audit decisions. Using these traces, we characterize where the approach is limited: retrieval mainly reduces tail risk rather than improving every instance, the audit detects injected errors but repairs only some of them, and weakly grounded analogical context remains difficult.

2 Related Work

We summarize the most relevant lines of work here and provide a fuller discussion in Appendix B. Time-series reasoning extends temporal modeling beyond direct value prediction toward intermediate computation, natural-language questions, what-if scenarios, and decision support (Ye et al., 2024; Kong et al., 2025; Jin et al., 2024). Recent benchmarks, surveys, and position papers further argue that time-series systems should integrate numerical signals with text, metadata, covariates, visual patterns, and natural-language intent (Chen et al., 2025; Kong et al., 2025; Zhang et al., 2025b; Jiang et al., 2025a). A central challenge is multimodal grounding: textual context may describe delayed effects, shifted event windows, qualitative mechanisms, hard constraints, hidden regimes, or output-specific requirements rather than simple timestamp-level correspondences. In parallel, forecasting has progressed from classical and deep models to transformer architectures and time-series foundation models trained across broad temporal corpora (Benidis et al., 2022; Wen et al., 2023; Liang et al., 2024; Ye et al., 2025), while LLM-based forecasting explores serialization, prompting, adaptation, and multimodal prompting (Gruver et al., 2023; Chang et al., 2025; Jiang et al., 2024b; Zhang et al., 2024). These methods show the promise of language priors and contextual reasoning, but also reveal weaknesses in direct numerical reasoning, tokenization, temporal grounding, and robust zero-shot behavior (Merrill et al., 2024; Spathis & Kawsar, 2024; Zhang et al., 2025a; Tan et al., 2024). Like several recent agents, ECLIPSE does not treat the LLM as the primary numerical forecaster: language components plan and audit, while forecasting backbones and deterministic operators perform the numerical computation. This division of labor is not new in itself, and we do not claim it as a contribution.

Time-series agents. TimeCopilot (Garza & Rosillo, 2025) already uses an LLM to orchestrate time-series foundation models for feature analysis, model selection, and cross-validation. Two concurrent systems share the name TimeClaw. Li et al. (2026) equip a generalist agent with executable temporal tools, experience-driven capability evolution, and episodic multimodal memory; Liu et al. (2026) learn from exploratory executions through an explore–compare–distill–reinject loop. The TimeClaw baseline in our tables is the system of Li et al. (2026), run from its public repository with the same LLM backbones as ECLIPSE; the exploration-based TimeClaw of Liu et al. (2026) is not evaluated. Relative to these agents, ECLIPSE differs in three specific mechanisms: (i) context is compiled into *event-scoped operations on both sides* of the foundation model, with history reconditioning before forecasting and future scenario enforcement after it, rather than into model selection or post-hoc explanation; (ii) retrieval is used as a *prior for completing the history* (e.g., supplying a full daily cycle), not as a memory of past reasoning traces; and (iii) an audit loop checks each intervention against the context and tool contracts before the output is returned. Appendix B gives a mechanism-level comparison (Table 6), and Section 5 tests whether these mechanisms, rather than tool access, account for the gain.

ECLIPSE also builds on retrieval-augmented time-series modeling, agentic tool use, and editable temporal dynamics. Retrieval-augmented forecasting uses similar histories, reference sequences, or temporal patterns to improve adaptation and grounding (Tire et al., 2024; Yang et al., 2024; Ning et al., 2025), and agentic retrieval further suggests that retrieved temporal evidence can support explanation and tool use (Ravuru et al., 2024). Unlike methods that append retrieved records as extra conditioning input, ECLIPSE treats retrieved references as part of the belief state constructed by the contextual prior agent: they can guide reference-based history enrichment, local repair, event-shape transfer, future adjustment, or comparative evidence construction. LLM agents have been studied as systems that decompose tasks, call tools, plan intermediate steps, and verify outputs (Ke et al., 2025; Shen, 2024; Huang et al., 2024; Ferrag et al., 2025); context-rich time-series tasks impose additional constraints on such agents, including temporal-window validity, numerical plausibility, covariate handling, evidence support, and the distinction between historical reconditioning and future scenario enforcement. Related work on counterfactual reasoning and time-series editing studies how trajectories can be simulated or modified under constraints (Kiciman et al., 2023; Zhang et al., 2022; Cao et al., 2024; Ye et al., 2024). ECLIPSE focuses on scenario-consistent execution rather than causal identification or universal trajectory generation: a unified planner maps the reconstructed belief state into either a generative state-editing program for forecasting or an analytical evidence-extraction program for QA, and an audit module validates the resulting trace, evidence, and output before finalization.

3 METHODOLOGY

3.1 PROBLEM SETUP

Each task instance consists of a time-series record and a natural-language context. Let $\mathbf{X}_{1:T} \in \mathbb{R}^{D\times T}$ denote the observed target history, $\mathbf{U}^h_{1:T}$ optional historical covariates, and future covariates known a priori. Alongside the numerical data, we are given a textual context C, which may contain explicit events, logical rules, domain constraints, or scenario-based questions. Given an input instance $\mathcal{I} = (\mathbf{X}_{1:T}, \mathbf{U}^h_{1:T}, \mathbf{U}^f_{1:H}, C)$, the system produces either a numerical forecast $\hat{\mathbf{Y}}_{1:H} \in \mathbb{R}^{D\times H}$ or an analytical answer a for a QA task. Instead of treating context C merely as an auxiliary feature to estimate $P(\mathbf{Y}_{1:H} \mid \mathbf{X}_{1:T}, C)$, context-rich tasks require dynamic interventions.

We treat the task as program synthesis over temporal states: the agent extracts a structured guidance state from C and emits a *temporal program*, a sequence of explicit operations (history reconditioning, foundation forecasting, future scenario enforcement, and evidence extraction) that transforms the initial state into the output. This is an empirical design choice rather than a formal optimization claim; its value is tested experimentally in Section 5.

3.2 INTENT-AWARE ROUTING AGENT

The Intent-Aware Routing Agent ρ serves as the initial gatekeeper, resolving both the required output format $m \in \{\text{FORECAST}, \text{QA}\}$ and the information sufficiency $k \in \{\text{SELF-CONTAINED}, \text{KNOWLEDGE-AUGMENTED}\}$. Given an input instance $\mathcal{I}$, it evaluates

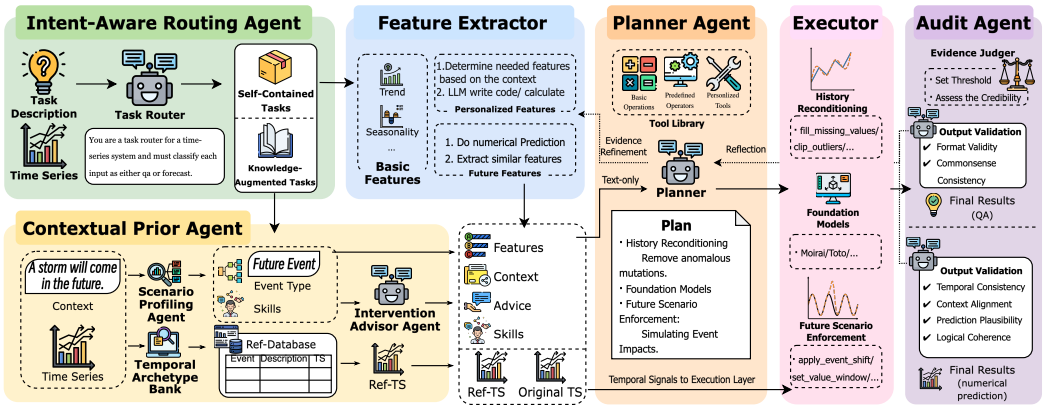


Figure 1: Overview of ECLIPSE. ECLIPSE compiles textual context into executable edits of the temporal state. The system routes the task by its output form, builds a guidance state from scenario profiling and Temporal Archetype Bank retrieval, synthesizes history reconditioning and future scenario enforcement plans around a time-series foundation model, and passes the result through an audit agent. The Intervention Advisor Agent is *optional* and is disabled in all main experiments (Section 3.3.3).

$(m, k, \mathbf{v}, c_\rho) = \rho_\theta(\mathcal{I})$, where $\mathbf{v}$ denotes the normalized task view that specifies the downstream execution contract and c_ρ is the routing confidence score. The sufficiency flag k determines the subsequent execution path.

For SELF-CONTAINED tasks, where the numerical history and prompt already provide sufficient evidence (e.g., identifying historical trends), the system bypasses heavy semantic grounding and directly extracts the required temporal evidence using registered statistical tools or lightweight executable logic. In contrast, KNOWLEDGE-AUGMENTED tasks require additional contextual grounding beyond the observed numerical history. Such grounding may arise from external events, domain-specific constraints, implicit operational priors, or retrieved temporal analogies, rendering the initial observation only partially informative. These tasks are therefore dispatched to the contextual prior agent to construct the guidance state $\mathcal{G}$ for downstream planning, which are detailed in Appendix E.1. A mis-routed task produces an output of the wrong form, which the audit's format-validity check (Section 3.5) rejects before the output is returned.

3.3 CONTEXTUAL PRIOR AGENT

The Contextual Prior Agent constructs a compact guidance state for downstream planning. Rather than exposing the LLM directly to high-dimensional raw time-series arrays $\mathbf{X}_{1:T}$, the framework enforces semantic-numerical decoupling: the agent performs contextual grounding and prior construction, while numerical computation is delegated to forecasting backbones and executable operators. To achieve this, the agent combines skill-based scenario profiling and temporal archetype retrieval, optionally complemented by the intervention advisor (Section 3.3.3).

3.3.1 SKILL-BASED SCENARIO PROFILING AND CONTEXT GROUNDING

The Scenario Profiling Agent first parses the textual context to extract target variables, timestamps, candidate rules, and event descriptions. We externalize domain knowledge through a structured skills registry, where each temporal skill s_z defines: *(i)* the class of scenarios it applies to; *(ii)* the typical temporal effects and persistence patterns associated with the scenario; and *(iii)* a scenario-conditioned guidance template for downstream planning. Given the parsed context, the agent classifies the task into a scenario type and retrieves the corresponding skill s_z. The retrieved skill is instantiated with parsed contextual fields to construct a semantic control state $\mathcal{S}_C$, containing the scenario type, relevant event or rule descriptions, candidate temporal scopes, salient covariates, and the skill-augmented planner prompt. The seven scenario categories form a planning-oriented ontology informed by common context types in contextual forecasting benchmarks, including the task families of CiK; we

discuss this dependence as a limitation in Section 6. Appendix E.2 further describes the construction of this semantic control state.

3.3.2 Temporal Archetype Bank and Fast Retrieval

A key assumption underlying contextual time-series reasoning is that many temporal behaviors recur across related scenarios. When the observed history is sparse or noisy, these recurring patterns can provide useful empirical priors for contextual grounding. To support this, we construct a *Temporal Archetype Bank* (Appendix E.3), an empirical memory designed to ground abstract contextual descriptions in concrete historical temporal patterns. The bank is built from data disjoint from all evaluation instances (non-overlapping years or training splits; Appendix E.3.2).

The retrieval query consists of three components: the observed numerical window $\mathbf{q} = \mathbf{X}_{T-L_q+1:T}$, domain metadata, and a context-derived semantic descriptor q_{desc}. To support efficient online retrieval, the database is organized offline through hierarchical DTW k-medoids clustering, compressing large collections of raw cycles into representative macro-archetypes. Each record is represented as

$$r_i = (\mathbf{p}_i, \ell_{a(i)}, \mu_i, \sigma_i, \mathbf{m}_i, a(i)),$$

where $\mathbf{p}_i$ denotes the normalized temporal pattern and $\ell_{a(i)}$ its semantic description.

During inference, the engine performs a bi-modal search over the archetype bank, computing a numerical and a semantic similarity:

$$d_{\text{ts}}(\mathbf{q}, \mathbf{p}_i, \varphi) = \|\tilde{\mathbf{q}} - \tilde{\mathbf{p}}_i[\varphi : \varphi + L_q - 1]\|_2, \qquad s_{\text{desc}}(i) = \cos\big(\text{enc}(q_{\text{desc}}), \text{enc}(\ell_{a(i)})\big). \quad (1)$$

The two signals are fused through Reciprocal Rank Fusion to obtain the top-K retrieved references R_K. Rather than directly appending retrieved records to the predictor input, ECLIPSE treats them as structured temporal priors for downstream planning, including sparse-history enrichment, event-shape transfer, and comparative evidence construction. Empirically, retrieval mainly helps when the history lacks full-cycle structure, and across the full benchmark it mainly reduces tail risk (Appendix C.3.3).

3.3.3 Optional Intervention Advisor Agent

This module is optional and disabled in our main experiments; it is intended for weaker planners. The same contextual cue can call for different interventions depending on the domain (e.g., a holiday may require future-side enforcement in one task and historical reconditioning in another). The Intervention Advisor Agent is a lightweight classifier that, given the condensed contextual state $\mathcal{C}$, suggests an intervention mode $\eta \in \{\texttt{hist}, \texttt{future}, \texttt{both}, \texttt{none}\}$, which is added to the guidance state $\mathcal{G} = (\mathcal{C}, \eta)$ as a soft prior; without the advisor, $\mathcal{G} = \mathcal{C}$. It is trained on 323 of 404 human-reviewed event-level records (81 are held out) by supervised fine-tuning (SFT) followed by a compatibility-aware preference refinement that treats `both` as partially compatible with single-sided labels (Appendix E.4). The evidence for this module is limited to a weak planner. With a fixed Qwen3-8B planner, the full advisor (SFT followed by preference refinement) lowers mean RCRPS from 0.280 to 0.214 relative to no advisor (Figure 2(b)), and the SFT-only advisor lowers median DTR from 0.158 to 0.142 (Table 15). The two checkpoints have not been compared end to end under one protocol, and as a classifier the refinement trades strict accuracy ($0.864 \rightarrow 0.691$) for compatibility-aware metrics (Table 23).

3.4 Unified Planner and Executor Agent

As the central decision-making hub of ECLIPSE, the Planner Agent functions as the execution policy π within our state-transition formulation. Rather than generating the final numerical trajectory or analytical answer in a single ungrounded forward pass, it maps the extracted guidance state $\mathcal{G}$ into an executable temporal program Π. The planner follows a unified program-synthesis interface, while the concrete execution plan depends on the routed output form m.

Planning for QA outputs. When $m = \text{QA}$, the planner synthesizes a *Feature Extraction Plan* Π_{Q} instead of directly producing an answer. This plan specifies *how* to measure the temporal features required by the question and orchestrates executable operators to compile a structured evidence bundle

$$\mathcal{E} = \mathcal{E}_{\text{hist}} \cup \mathcal{E}_{\text{ctx}} \cup \mathcal{E}_{\text{fore}} \cup \mathcal{E}_{\text{cmp}}.$$

For instance, the plan may call statistical tools to calculate historical trends or peak timing ($\mathcal{E}_{\text{hist}}$), extract explicit contextual constraints ($\mathcal{E}_{\text{ctx}}$), and execute comparisons between candidate options ($\mathcal{E}_{\text{cmp}}$). If the question depends on future dynamics, the planner invokes the FORECAST execution path to obtain forecast-mediated evidence ($\mathcal{E}_{\text{fore}}$). The final answer is then resolved from the evidence bundle rather than generated from the prompt alone.

Planning for FORECAST outputs. When $m = \text{FORECAST}$, the planner synthesizes a generative state-editing program, which is detailed in Appendix E.5.

$$\Pi_{\text{F}} = (\Pi_{\text{hist}}, \Pi_{\text{fm}}, \Pi_{\text{future}}).$$

This program constructs a context-aligned computational state for the numerical foundation model, preventing the forecast from being biased by historical anomalies or ignoring future contextual constraints. It consists of three stages. First, *History Reconditioning* (Π_{hist}) repairs or enriches the raw history $\mathbf{X}_{1:T}$ using retrieved priors R_K when needed. Second, *Foundation Forecasting* (Π_{fm}) invokes a dedicated time-series backbone for numerical extrapolation. Finally, *Future Scenario Enforcement* (Π_{future}) edits the base forecast to satisfy explicit contextual constraints such as bounds or event-induced shifts. This planned sequence allows semantic context to govern both the reconstructed input state and the constrained future trajectory. In the 25-sample probabilistic evaluation, one program is synthesized per instance and applied to every sampled trajectory of the foundation model, rather than to a single point forecast.

3.5 EXECUTION AUDIT AND CLOSED-LOOP REFLECTION

The audit agent serves as the final validation module for both execution branches. Rather than acting as a simple formatting checker, it verifies that the generated temporal operations and final outputs remain consistent with the constraints implied by the original context. Formally, it evaluates the generated program Π, guidance state $\mathcal{G}$, evidence bundle $\mathcal{E}$, and final output o:

$$b = \nu(\Pi, \mathcal{G}, \mathcal{E}, o) \in \{\text{ACCEPT}, \text{REVISE}, \text{FALLBACK}\}. \tag{2}$$

For knowledge-augmented forecasting tasks, the audit checks numerical and temporal consistency, including operation validity, constraint satisfaction, and temporal scope alignment. For self-contained QA tasks, it verifies logical consistency between the generated answer and the supporting evidence bundle $\mathcal{E}$. These are detailed in Appendix E.7.

A key feature of ECLIPSE is its closed-loop reflection mechanism. If the audit detects a violation, it returns a $b = \text{REVISE}$ signal and appends an explicit error trace to the context, prompting the Planner Agent to re-plan the flawed operation. There is no global retry budget across the pipeline. The bounded retry is at the post-adjustment step: if the LLM's revised series violates the output schema, one strict-schema retry is issued (at most two requests), and a still-invalid output triggers $b = \text{FALLBACK}$. For forecasting, the fallback discards all interventions and returns the unedited foundation-model forecast on the raw history; for QA, it returns an explicit unresolved-answer token (Appendix E.7.5). In a controlled test the audit flags every injected error but suggests a correct repair for only some error types (Appendix C.3.7); it is not a correctness guarantee.

Finally, ECLIPSE records retrieved references, programs, evidence objects, audit feedback, and outputs in an execution trace, which supports debugging and error analysis.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Benchmarks. We evaluate ECLIPSE on two benchmarks: **Context is Key (CiK)** Williams et al. (2024) for contextual time-series forecasting, and **TemporalBench** Weng et al. (2026) for a broader set of temporal reasoning tasks (including event-informed prediction and temporal QA/MCQ). Together, they cover both output forms of our framework: numerical trajectories and evidence-grounded answers conditioned on natural-language context.

Baselines. For CiK, we compare against a comprehensive taxonomy of baselines, categorized into: (1) **LLM-based paradigms** (Direct Prompting and LLM Processes) Williams et al. (2024) using diverse open-source families including Llama-3 Grattafiori et al. (2024), Qwen-3/2.5 Qwen et al. (2025); Yang et al. (2025), and Mixtral Jiang et al. (2024a); (2) **Time-Series Foundation Models** (e.g., Moirai Woo et al. (2024), Lag-Llama Rasul et al. (2023), Chronos-Large Ansari et al. (2024), TimeGPT Garza et al. (2023)); (3) **Multimodal Models** (UniTime Liu et al. (2024), Time-LLM Jin et al. (2023)); and (4) **Statistical Models** (ARIMA Box et al. (2015), ETS, Exp-Smoothing Hyndman & Khandakar (2008)); and (5) **Time-series agents**: TimeClaw (Li et al., 2026), run with the same DeepSeek-Chat and GPT-5.4-mini backbones as ECLIPSE, and TimeCopilot (Garza & Rosillo, 2025) with DeepSeek-Chat (single-sample setting, Table 11). For TemporalBench, we compare against direct LLM prompting, general-purpose agent frameworks such as AgentScope Gao et al. (2024), CAMEL Li et al. (2023), and MetaGPT Hong et al. (2023), and the time-series agent TimeSeriesScientist Zhao et al. (2025). ECLIPSE uses DeepSeek-Chat DeepSeek-AI (2026), GPT-5.4-mini OpenAI (2026), and Qwen3 as its base models, with Moirai-2.0-R-small as the default forecasting backbone. The Intervention Advisor is disabled in all main experiments. Note that the LLM baselines in Table 1 use different (open-weight) models from the ECLIPSE backbones; direct prompting with the *same* backbones is compared in Table 11, and a same-toolchain comparison is given in Table 3.

Metrics. For CiK's 25-sample probabilistic evaluation, we use the Region-of-Interest Continuous Ranked Probability Score (RCRPS), which emphasizes context-sensitive forecast regions and penalizes constraint violations. For single-sample deterministic settings, we use Deterministic Task Risk (DTR), a single-sample, ROI-weighted and constraint-penalized point risk (Appendix C.2). TemporalBench tasks are evaluated using standard accuracy for QA/MCQ. For RCRPS, each method is run once and $\pm$ is computed from the 25 sampled trajectories per instance. Appendix A lists, for every results table, the LLM, forecasting backbone, Advisor state, instance set, metric, and aggregation.

4.2 Overall Performance

Results on CiK. Among the methods we run under our protocol (Table 1), ECLIPSE (DeepSeek-Chat) attains the lowest average RCRPS (0.174), ahead of the strongest agentic baseline, TimeClaw with the same backbone (0.188). It is not the best entry overall: the largest direct-prompting model reported by CiK, Llama-3.1-405B-Inst, reaches 0.159, and it is also better on the intemporal, historical, and covariate subsets. ECLIPSE's clearest advantage is on Causal Information tasks (0.292, the lowest in the table, against 0.345 for TimeClaw and 0.398 for Llama-3.1-405B-Inst), where the context specifies mechanisms that must be applied as explicit operations.

On Future Information tasks, ECLIPSE (GPT-5.4-mini) obtains 0.067, second only to TimeClaw (DeepSeek-Chat, 0.060); in the single-sample setting, ECLIPSE (GPT-5.4-mini) reduces future-information DTR relative to direct prompting with the same backbone from 0.088 to 0.061 (Table 11). These results suggest that contextual information is most useful when it is turned into executable control over the forecast rather than supplied as auxiliary prompt input.

However, ECLIPSE does not achieve the best performance on the Historical Information subset. In this category, Moirai-2.0-R-small obtains the strongest result (RCRPS of 0.118), while ECLIPSE variants range between 0.153 and 0.159. Historical contexts often involve weakly grounded analogies or incomplete drivers, where enforcing ambiguous context can be less effective than the broad temporal priors of foundation models; context *availability* is not context *actionability* (failure cases in Appendix D.2).

Results on TemporalBench. On TemporalBench (Table 2), ECLIPSE achieves the best CR accuracy on all four datasets (FreshRetailNet, PSML, Causal Chambers, and MIMIC) and the best EIP accuracy on PSML, Causal Chambers, and MIMIC. It does not lead on PSML-HU, where direct prompting is stronger (74.00 vs. 67.00), or on FreshRetailNet-EIP, where TimeSeriesScientist is stronger (56.82 vs. 40.15). The decomposition into context translation, program synthesis, and evidence-grounded answer resolution is thus more effective on context-dependent tasks (CR/EIP) than on purely historical ones.

Table 1: CiK results under the 25-sample probabilistic evaluation setting. We report RCRPS (lower is better). Results are averaged over all tasks and stratified by context type; $\pm$ is computed from the 25 samples of a single run. †Reported by Williams et al. (2024) (Table 1, arXiv v4), not re-run by us; its $\pm$ is the standard error reported there. TimeClaw is the system of Li et al. (2026). The Advisor is disabled for all ECLIPSE rows.

Model	Average RCRPS	Intemporal Information	Historical Information	Future Information	Covariate Information	Causal Information
Direct Prompt						
Llama-3-70B-Inst	0.342 ± 0.073	0.347 ± 0.051	0.183 ± 0.016	0.353 ± 0.263	0.241 ± 0.042	0.664 ± 0.115
Qwen-3-8B-Inst	0.358 ± 0.041	0.371 ± 0.033	0.262 ± 0.022	0.336 ± 0.140	0.296 ± 0.027	0.544 ± 0.052
Qwen-3-0.6B-Inst	0.519 ± 0.062	0.562 ± 0.071	0.699 ± 0.073	0.280 ± 0.093	0.471 ± 0.076	1.000 ± 0.175
Qwen-2.5-7B-Inst	0.394 ± 0.040	0.472 ± 0.052	0.280 ± 0.021	0.219 ± 0.051	0.382 ± 0.051	0.442 ± 0.060
Qwen-2.5-0.5B-Inst	0.463 ± 0.012	0.609 ± 0.019	0.165 ± 0.004	0.218 ± 0.012	0.476 ± 0.015	0.429 ± 0.006
Llama-3.1-405B-Inst†	$\mathbf{0.159 \pm 0.008}$	$\mathbf{0.174 \pm 0.010}$	0.146 ± 0.001	0.075 ± 0.005	$\mathbf{0.164 \pm 0.010}$	0.398 ± 0.045
Mixtral-8x7B-Inst	0.523 ± 0.023	0.723 ± 0.037	0.236 ± 0.002	0.241 ± 0.001	0.359 ± 0.028	0.875 ± 0.128
LLMP						
Llama-3-70B-Inst	0.547 ± 0.015	0.436 ± 0.020	0.514 ± 0.029	0.842 ± 0.025	0.548 ± 0.019	0.395 ± 0.022
Llama-3-70B	0.236 ± 0.006	0.212 ± 0.005	0.121 ± 0.008	0.299 ± 0.017	0.193 ± 0.004	0.360 ± 0.011
Mixtral-8x7B-Inst	0.264 ± 0.004	0.242 ± 0.007	0.173 ± 0.004	0.324 ± 0.005	0.219 ± 0.005	0.437 ± 0.007
Mixtral-8x7B	0.262 ± 0.008	0.250 ± 0.008	0.119 ± 0.003	0.310 ± 0.019	0.229 ± 0.006	0.457 ± 0.011
Qwen-3-8B	0.313 ± 0.026	0.364 ± 0.034	0.214 ± 0.015	0.194 ± 0.032	0.319 ± 0.034	0.526 ± 0.056
Qwen-3-0.6B	0.708 ± 0.133	0.990 ± 0.188	0.192 ± 0.010	0.416 ± 0.152	0.772 ± 0.168	1.617 ± 0.563
TS Foundation Models						
Lag-Llama	0.335 ± 0.040	0.337 ± 0.028	0.154 ± 0.009	0.305 ± 0.144	0.301 ± 0.028	0.531 ± 0.060
Chronos-Large	0.326 ± 0.002	0.314 ± 0.002	0.179 ± 0.003	0.379 ± 0.003	0.255 ± 0.002	0.460 ± 0.004
TimeGEN	0.353 ± 0.000	0.332 ± 0.000	0.177 ± 0.000	0.405 ± 0.000	0.292 ± 0.000	0.474 ± 0.000
Moirai-2.0-R-small	0.320 ± 0.071	0.251 ± 0.025	$\mathbf{0.118 \pm 0.007}$	0.468 ± 0.277	0.250 ± 0.024	0.469 ± 0.061
Multimodal Models						
UniTime	0.370 ± 0.001	0.457 ± 0.002	0.155 ± 0.000	0.194 ± 0.003	0.395 ± 0.001	0.423 ± 0.001
Time-LLM (ETTh1)	0.476 ± 0.001	0.518 ± 0.002	0.183 ± 0.000	0.403 ± 0.002	0.441 ± 0.001	0.482 ± 0.001
Statistical Models						
ARIMA	0.561 ± 0.104	0.535 ± 0.047	0.375 ± 0.044	0.640 ± 0.402	0.425 ± 0.047	0.469 ± 0.039
ETS	0.878 ± 0.139	0.986 ± 0.111	0.513 ± 0.098	0.795 ± 0.471	0.567 ± 0.061	0.514 ± 0.037
Exp-Smoothing	0.809 ± 0.117	0.893 ± 0.071	0.331 ± 0.062	0.802 ± 0.433	0.584 ± 0.058	0.519 ± 0.050
Time-Series Agents						
TimeClaw (DeepSeek-Chat)	0.188 ± 0.025	0.218 ± 0.033	0.170 ± 0.015	$\mathbf{0.060 \pm 0.015}$	0.191 ± 0.032	0.345 ± 0.065
TimeClaw (GPT-5.4-mini)	0.390 ± 0.058	0.378 ± 0.030	0.203 ± 0.010	0.403 ± 0.218	0.311 ± 0.027	0.540 ± 0.056
Ours						
ECLIPSE (Qwen3-0.6B)	0.390 ± 0.096	0.286 ± 0.023	0.159 ± 0.008	0.599 ± 0.378	0.284 ± 0.024	0.464 ± 0.064
ECLIPSE (Qwen3-8B)	0.251 ± 0.029	0.286 ± 0.039	0.155 ± 0.010	0.118 ± 0.035	0.283 ± 0.037	0.439 ± 0.061
ECLIPSE (GPT-5.4-mini)	0.190 ± 0.017	0.214 ± 0.018	0.153 ± 0.008	0.067 ± 0.020	0.212 ± 0.021	0.395 ± 0.039
ECLIPSE (DeepSeek-Chat)	0.174 ± 0.019	0.200 ± 0.023	0.155 ± 0.010	0.075 ± 0.028	0.190 ± 0.024	$\mathbf{0.292 \pm 0.045}$

Table 2: Accuracy comparison on TemporalBench using DeepSeek-Chat as the base model. We report three task families: Historical Understanding (HU: interpreting trends, volatility, seasonality, and anomalies from historical observations), Contextual Reasoning (CR: grounding temporal behavior in textual context), and Event-Informed Prediction (EIP: predicting or assessing event-conditioned future behavior).

Agent	FreshRetailNet			PSML			Causal Chambers			MIMIC		
	HU	CR	EIP	HU	CR	EIP	HU	CR	EIP	HU	CR	EIP
Single LLM	68.18	16.85	26.52	**74.00**	34.80	36.00	15.33	44.40	29.33	33.00	39.14	22.67
TimeSeriesScientist	36.36	34.09	**56.82**	38.50	22.80	27.33	28.67	29.60	2.67	15.43	25.94	23.40
AgentScope	70.45	5.68	42.42	68.50	24.80	31.33	16.00	46.40	42.00	32.98	25.71	24.82
MetaGPT	69.89	5.11	37.12	71.50	28.00	30.67	15.33	46.40	40.00	32.98	25.79	30.50
CAMEL	67.05	19.89	29.55	65.00	25.20	30.67	12.67	46.80	48.00	31.38	25.61	31.21
ECLIPSE	**73.90**	**45.00**	40.15	67.00	**46.00**	**50.67**	**68.70**	**56.40**	**66.67**	**62.20**	**54.00**	**38.33**

Note: Context-free future prediction provides no textual context and is evaluated with forecasting metrics rather than QA-style accuracy, so it is omitted from this context-focused table.

5 Ablation Study

Orchestration versus tool access. Table 3 holds the language model, forecasting backbone, retrieved archetype, and tool library fixed and varies only how they are organized. A single agent that calls the same tools, a fixed program template, and retrieval-plus-post-processing reduce DTR from 0.246 (Moirai alone) to 0.209–0.236; ECLIPSE reaches 0.187. The largest differences are on intemporal, covariate, and causal tasks, where the context must be turned into a specific intervention. The controls remain competitive elsewhere: retrieval-plus-post-processing is best on future-information tasks, and all methods are similar on historical-information tasks. The controls share inputs, tools,

Table 3: Shared-toolchain comparison on CiK (DTR, lower is better). All methods use the same input, retrieved archetype, DeepSeek-Chat, tools, and Moirai; Advisor off. Best **bold**, second underlined.

METHOD	OVERALL	INTEMPORAL	HISTORICAL	FUTURE	COVARIATE	CAUSAL
Moirai baseline	0.246	0.566	0.147	0.147	0.402	0.907
Single Agent + Same Tools	0.223	0.532	**0.145**	0.076	0.381	0.898
Fixed Program Template	0.236	0.562	0.146	0.111	0.408	0.908
RAG + Post-processing	0.209	0.540	0.162	**0.025**	0.351	0.893
ECLIPSE	**0.187**	**0.266**	0.147	0.042	**0.196**	**0.388**

Table 4: Tail-robust evidence (DTR). Top: paired 10%-trimmed improvement. Bottom: degradation on removal, over pairs with both $\mathrm{DTR} \leq 1$. Positive favors the full system; bootstrap 95% CIs over instances (Appendix C.3.6).

COMPARISON	N	EFFECT	95% CI
ECLIPSE vs. paired TSFM	355	0.0170	[0.0090, 0.0270]
History reconditioning, when applied	102	0.0142	[0.0046, 0.0261]
Future-scenario enforcement, when applied	327	0.0104	[0.0026, 0.0203]
w/o Contextual Priors	344	0.0143	[0.0064, 0.0239]
w/o All Context	341	0.0140	[0.0060, 0.0245]
w/o Future Scenario Enforcement	316	0.0092	[0.0015, 0.0183]

and models but not the compute budget: ECLIPSE uses about 7 LLM calls per task versus one for the single-agent control (Table 19). We attribute the aggregate gain to the coordination of history-side and future-side operations rather than to tool or backbone access alone, without claiming that the full workflow dominates every context type.

Component and intervention ablations. Raw-mean ablations (Table 13, Appendix C.3.1) are dominated by rare catastrophic instances; for example, a single recursive rule synthesized without contextual priors reaches $\mathrm{DTR} \approx 10^7$ (Appendix C.3.2). We therefore report tail-robust statistics in Table 4. On ordinary instances, removing contextual priors or all context increases DTR by about 0.014, and removing future scenario enforcement by 0.009; all three intervals exclude zero. When applied, history reconditioning and future enforcement each yield positive paired gains (0.0142 and 0.0104, with intervals excluding zero). The audit's effect is concentrated in the tail: without it, a few covariate and causal instances explode, so it mainly prevents rare catastrophic failures. Removing all context raises the average DTR from 0.197 to 0.313, with degradation in every category.

6 CONCLUSION AND LIMITATIONS

We introduced ECLIPSE, an agentic framework that compiles textual context into executable operations on both sides of a time-series foundation model: history reconditioning before forecasting and future scenario enforcement after it, with retrieval as a prior for history completion and an audit loop around the program. Experiments on CiK and TemporalBench show that this is an effective and inspectable alternative to direct prompting, and a shared-toolchain comparison attributes the gain to orchestration rather than to tool access alone. Future work includes adaptive temporal memory and more conservative, uncertainty-aware intervention policies for ambiguous context.

Limitations. (1) *Backbone dependence and tail risk.* With less stable backbones or without contextual priors, a single recursive rule can make the mean error explode (Appendix C.1.2, C.3.2). (2) *Scenario ontology.* The scenario categories were designed with CiK-style context types in view; coverage of other domains is untested. (3) *Advisor.* The Advisor uses only 404 imbalanced records and has been tested end to end only with Qwen3-8B, without a direct SFT-vs.-preference comparison; its both-tolerant accuracy does not exceed that of a trivial always-`both` baseline (Appendix E.4.8). (4) *Audit.* The audit suggests correct repairs for only some error types (Appendix C.3.7). (5) *Reproducibility and cost.* Main results rely on closed API models; cost was measured on 25 tasks. Weakly grounded analogical context remains a failure mode (Appendix D.2).

AI USE STATEMENT

Large language models (LLMs) are components of the agent system studied in this work; their roles, configurations, and use in the experiments are described in the methodology and experimental sections. Separately, generative AI tools were used to improve the clarity and readability of the manuscript. All AI-assisted edits were reviewed and revised by the authors, who take full responsibility for the final content, claims, and conclusions of this work.

REFERENCES

Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, et al. Chronos: Learning the language of time series. *arXiv preprint arXiv:2403.07815*, 2024.

Konstantinos Benidis, Syama Sundar Rangapuram, Valentin Flunkert, Yuyang Wang, Danielle Maddix, Caner Turkmen, Jan Gasthaus, Michael Bohlke-Schneider, David Salinas, Lorenzo Stella, et al. Deep learning for time series forecasting: Tutorial and literature survey. *ACM Computing Surveys*, 55(6):1–36, 2022.

George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

Luis Candanedo. Appliances Energy Prediction. UCI Machine Learning Repository, 2017. DOI: https://doi.org/10.24432/C5VC8G.

Defu Cao, Wen Ye, Yizhou Zhang, and Yan Liu. Timedit: General-purpose diffusion transformers for time series foundation model. *arXiv preprint arXiv:2409.02322*, 2024.

Ching Chang, Wei-Yao Wang, Wen-Chih Peng, and Tien-Fu Chen. Llm4ts: Aligning pre-trained llms as data-efficient time-series forecasters. *ACM Trans. Intell. Syst. Technol.*, 16(3), April 2025. ISSN 2157-6904. doi: 10.1145/3719207. URL https://doi.org/10.1145/3719207.

Jialin Chen, Aosong Feng, Ziyu Zhao, Juan Garza, Gaukhar Nurbek, Cheng Qin, Ali Maatouk, Leandros Tassiulas, Yifeng Gao, and Rex Ying. Mtbench: A multimodal time series benchmark for temporal reasoning and question answering. 2025. URL https://arxiv.org/abs/2503.16858.

Song Chen. Beijing PM2.5. UCI Machine Learning Repository, 2015. DOI: https://doi.org/10.24432/C5JS49.

Song Chen. PM2.5 Data of Five Chinese Cities. UCI Machine Learning Repository, 2016. DOI: https://doi.org/10.24432/C52K58.

Ben Cohen, Emaad Khwaja, Kan Wang, Charles Masson, Elise Ramé, Youssef Doubli, and Othmane Abou-Amal. Toto: Time series optimized transformer for observability. *arXiv preprint arXiv:2407.07874*, 2024.

Gordon V. Cormack, Charles L A Clarke, and Stefan Buettcher. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '09, pp. 758–759, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605584836. doi: 10.1145/1571941.1572114. URL https://doi.org/10.1145/1571941.1572114.

DeepSeek-AI. DeepSeek API Documentation. https://api-docs.deepseek.com/, 2026. Model identifier: deepseek-chat. Accessed: 2026-05-07.

Hadi Fanaee-T. Bike Sharing. UCI Machine Learning Repository, 2013. DOI: https://doi.org/10.24432/C5W894.

Mohamed Amine Ferrag, Norbert Tihanyi, and Merouane Debbah. From llm reasoning to autonomous ai agents: A comprehensive review. *arXiv preprint arXiv:2504.19678*, 2025.

Juan L Gamella, Jonas Peters, and Peter Bühlmann. The causal chambers: Real physical systems as a testbed for ai methodology. *arXiv preprint arXiv:2404.11341*, 2024.

Dawei Gao, Zitao Li, Xuchen Pan, Weirui Kuang, Zhijian Ma, Bingchen Qian, Fei Wei, Wenhao Zhang, Yuexiang Xie, Daoyuan Chen, et al. Agentscope: A flexible yet robust multi-agent platform. *arXiv preprint arXiv:2402.14034*, 2024.

Azul Garza and Renée Rosillo. TimeCopilot. *arXiv preprint arXiv:2509.00616*, 2025. URL https://arxiv.org/abs/2509.00616.

Azul Garza, Cristian Challu, and Max Mergenthaler-Canseco. Timegpt-1. *arXiv preprint arXiv:2310.03589*, 2023.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Nate Gruver, Marc Anton Finzi, Shikai Qiu, and Andrew Gordon Wilson. Large language models are zero-shot time series forecasters. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=md68e8iZK1.

Georges Hebrail and Alice Berard. Individual Household Electric Power Consumption. UCI Machine Learning Repository, 2006. DOI: https://doi.org/10.24432/C58K54.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, et al. Metagpt: Meta programming for a multi-agent collaborative framework. In *The twelfth international conference on learning representations*, 2023.

Yifan Hu, Jie Yang, Xilin Dai, Wanxu Cai, Kuiye Ding, Yuante Li, Qinghua Liu, Enze Ma, Zhiyuan Qu, Yixin Wang, et al. The landscape of agentic time series systems: Architectures, reliability, and frontiers. 2026.

Xu Huang, Weiwen Liu, Xiaolong Chen, Xingmei Wang, Hao Wang, Defu Lian, Yasheng Wang, Ruiming Tang, and Enhong Chen. Understanding the planning of llm agents: A survey. *arXiv preprint arXiv:2402.02716*, 2024.

Rob J Hyndman and Yeasmin Khandakar. Automatic time series forecasting: the forecast package for r. *Journal of statistical software*, 27:1–22, 2008.

James M. Kilts Center for Marketing. Dominick's dataset. https://www.chicagobooth.edu/research/kilts/research-data/dominicks. Store-level scanner data from Dominick's Finer Foods. Accessed 2026-05-06.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024a.

Yushan Jiang, Zijie Pan, Xikun Zhang, Sahil Garg, Anderson Schneider, Yuriy Nevmyvaka, and Dongjin Song. Empowering time series analysis with large language models: A survey. *arXiv preprint arXiv:2402.03182*, 2024b.

Yushan Jiang, Kanghui Ning, Zijie Pan, Xuyang Shen, Jingchao Ni, Wenchao Yu, Anderson Schneider, Haifeng Chen, Yuriy Nevmyvaka, and Dongjin Song. Multi-modal time series analysis: A tutorial and survey. *arXiv preprint arXiv:2503.13709*, 2025a.

Yushan Jiang, Wenchao Yu, Geon Lee, Dongjin Song, Kijung Shin, Wei Cheng, Yanchi Liu, and Haifeng Chen. Explainable multi-modal time series prediction with llm-in-the-loop. 2025b. URL https://arxiv.org/abs/2503.01013.

Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Y Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, et al. Time-llm: Time series forecasting by reprogramming large language models. *arXiv preprint arXiv:2310.01728*, 2023.

Ming Jin, Yifan Zhang, Wei Chen, Kexin Zhang, Yuxuan Liang, Bin Yang, Jindong Wang, Shirui Pan, and Qingsong Wen. Position: what can large language models tell us about time series analysis. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Zixuan Ke, Fangkai Jiao, Yifei Ming, Xuan-Phi Nguyen, Austin Xu, Do Xuan Long, Minzhi Li, Chengwei Qin, PeiFeng Wang, silvio savarese, Caiming Xiong, and Shafiq Joty. A survey of frontiers in LLM reasoning: Inference scaling, learning to reason, and agentic systems. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=SlsZZ25InC. Survey Certification.

Emre Kiciman, Robert Ness, Amit Sharma, and Chenhao Tan. Causal reasoning and large language models: Opening a new frontier for causality. *Transactions on Machine Learning Research*, 2023.

Yaxuan Kong, Yiyuan Yang, Shiyu Wang, Chenghao Liu, Yuxuan Liang, Ming Jin, Stefan Zohren, Dan Pei, Yan Liu, and Qingsong Wen. Position: Empowering time series reasoning with multimodal llms. 2025. URL https://arxiv.org/abs/2502.01477.

Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. Camel: Communicative agents for" mind" exploration of large language model society. *Advances in neural information processing systems*, 36:51991–52008, 2023.

Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *International Conference on Learning Representations (ICLR '18)*, 2018.

Zihao Li, Kaifeng Jin, Yuanchen Bei, Jiaru Zou, Avaneesh Kumar, Xuying Ning, Yanjun Zhao, Mengting Ai, Baoyu Jing, Hanghang Tong, and Jingrui He. Harnessing generalist agents for contextualized time series. *arXiv preprint arXiv:2606.05404*, 2026. URL https://arxiv.org/abs/2606.05404.

Yuxuan Liang, Haomin Wen, Yuqi Nie, Yushan Jiang, Ming Jin, Dongjin Song, Shirui Pan, and Qingsong Wen. Foundation models for time series analysis: A tutorial and survey. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '24, pp. 6555–6565, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400704901. doi: 10.1145/3637528.3671451. URL https://doi.org/10.1145/3637528.3671451.

Hangchen Liu, Dongyuan Li, Renhe Jiang, Jiewen Deng, Weiwei Ye, and Yoshihide Sekimoto. TimeClaw: A time-series AI agent with exploratory execution learning. *arXiv preprint arXiv:2605.10038*, 2026. URL https://arxiv.org/abs/2605.10038.

Haoxin Liu, Chenghao Liu, and B. Aditya Prakash. A picture is worth a thousand numbers: Enabling LLMs reason about time series via visualization. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 7486–7518, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-189-6. doi: 10.18653/v1/2025.naacl-long.383. URL https://aclanthology.org/2025.naacl-long.383/.

Xu Liu, Junfeng Hu, Yuan Li, Shizhe Diao, Yuxuan Liang, Bryan Hooi, and Roger Zimmermann. Unitime: A language-empowered unified model for cross-domain time series forecasting. In *Proceedings of the ACM Web Conference 2024*, pp. 4095–4106, 2024.

Mike A Merrill, Mingtian Tan, Vinayak Gupta, Thomas Hartvigsen, and Tim Althoff. Language models still struggle to zero-shot reason about time series. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 3512–3533, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.201. URL https://aclanthology.org/2024.findings-emnlp.201/.

Abdullah Mueen, Sheng Zhong, Yan Zhu, Michael Yeh, Kaveh Kamgar, Krishnamurthy Viswanathan, Chetan Gupta, and Eamonn Keogh. The fastest similarity search algorithm for time series subsequences under euclidean distance, August 2022. http://www.cs.unm.edu/~mueen/FastestSimilaritySearch.html.

Kanghui Ning, Zijie Pan, Yu Liu, Yushan Jiang, James Y. Zhang, Kashif Rasul, Anderson Schneider, Lintao Ma, Yuriy Nevmyvaka, and Dongjin Song. TS-RAG: Retrieval-augmented generation based time series foundation models are stronger zero-shot forecaster. 2025. URL https://arxiv.org/abs/2503.07649.

OpenAI. GPT-5.4 mini and nano: Smaller, faster versions of GPT-5.4. https://openai.com/index/introducing-gpt-5-4-mini-and-nano/, March 2026. Model identifier: gpt-5.4-mini. Accessed: 2026-05-07.

Marco A. F. Pimentel, Alistair E. W. Johnson, Peter H. Charlton, Drew Birrenkott, Peter J. Watkinson, Lionel Tarassenko, and David A. Clifton. Toward a robust estimation of respiratory rate from pulse oximeters. *IEEE Transactions on Biomedical Engineering*, 64(8):1914–1923, 2017. doi: 10.1109/TBME.2016.2613124.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.

Kashif Rasul, Arjun Ashok, Andrew Robert Williams, Hena Ghonia, Rishika Bhagwatkar, Arian Khorasani, Mohammad Javad Darvishi Bayazi, George Adamopoulos, Roland Riachi, Nadhir Hassen, et al. Lag-llama: Towards foundation models for probabilistic time series forecasting. *arXiv preprint arXiv:2310.08278*, 2023.

Chidaksh Ravuru, Sagar Srinivas Sakhinana, and Venkataramana Runkana. Agentic retrieval-augmented generation for time series analysis. 2024. URL https://arxiv.org/abs/2408.14484.

Pablo Romeu-Guallart and Francisco Zamora-Martinez. SML2010. UCI Machine Learning Repository, 2014. DOI: https://doi.org/10.24432/C5RS3S.

Zhuocheng Shen. Llm with tools: A survey. *arXiv preprint arXiv:2409.18807*, 2024.

shenba. Time series datasets. https://www.kaggle.com/datasets/shenba/time-series-datasets, 2017. Kaggle dataset. Accessed 2026-05-06.

Dimitris Spathis and Fahim Kawsar. The first step is the hardest: Pitfalls of representing and tokenizing temporal data for large language models. *Journal of the American Medical Informatics Association*, 31(9):2151–2158, 2024.

Mingtian Tan, Mike A Merrill, Vinayak Gupta, Tim Althoff, and Thomas Hartvigsen. Are language models actually useful for time series forecasting? In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=DV15UbHCY1.

Kutay Tire, Ege Onur Taga, Muhammed Emrullah Ildiz, and Samet Oymak. Retrieval augmented time series forecasting. 2024. URL https://arxiv.org/abs/2411.08249.

Mohamed Trabelsi, Aidan Boyd, Jin Cao, and Huseyin Uzunalioglu. Time series language model for descriptive caption generation. 2025. URL https://arxiv.org/abs/2501.01832.

Artur Trindade. ElectricityLoadDiagrams20112014. UCI Machine Learning Repository, 2015. DOI: https://doi.org/10.24432/C58C86.

Saverio Vito. Air Quality. UCI Machine Learning Repository, 2008. DOI: https://doi.org/10.24432/C59K5F.

Xinyue Wang, Jiacheng Pang, Kun Zhou, Kexin Zhang, Defu Cao, Fan Feng, Faisal, Songyao Jin, Yan Liu, and Biwei Huang. Timebraid: Unifying time series and language for understanding and forecasting, 2026. URL https://arxiv.org/abs/2609.29792.

Qingsong Wen, Tian Zhou, Chaoli Zhang, Weiqi Chen, Ziqing Ma, Junchi Yan, and Liang Sun. Transformers in time series: a survey. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, IJCAI '23, 2023. ISBN 978-1-956792-03-4. doi: 10.24963/ijcai.2023/759. URL https://doi.org/10.24963/ijcai.2023/759.

Muyan Weng, Defu Cao, Wei Yang, Yashaswi Sharma, and Yan Liu. Temporalbench: A benchmark for evaluating llm-based agents on contextual and event-informed time series tasks. *arXiv preprint arXiv:2602.13272*, 2026.

Andrew Robert Williams, Arjun Ashok, Étienne Marcotte, Valentina Zantedeschi, Jithendaraa Subramanian, Roland Riachi, James Requeima, Alexandre Lacoste, Irina Rish, Nicolas Chapados, et al. Context is key: A benchmark for forecasting with essential textual information. *arXiv preprint arXiv:2410.18959*, 2024.

Gerald Woo, Chenghao Liu, Akshat Kumar, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. Unified training of universal time series forecasting transformers. In *Forty-first International Conference on Machine Learning*, 2024.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Silin Yang, Dong Wang, Haoqi Zheng, and Ruochun Jin. Timerag: Boosting llm time series forecasting via retrieval-augmented generation. 2024. URL https://arxiv.org/abs/2412.16643.

Jiexia Ye, Yongzi Yu, Weiqi Zhang, Le Wang, Jia Li, and Fugee Tsung. Empowering time series analysis with foundation models: A comprehensive survey. 2025. URL https://arxiv.org/abs/2405.02358.

Wen Ye, Yizhou Zhang, Wei Yang, Lumingyuan Tang, Defu Cao, Jie Cai, and Yan Liu. Beyond forecasting: Compositional time series reasoning for end-to-end task execution. *CoRR*, abs/2410.04047, 2024. URL https://doi.org/10.48550/arXiv.2410.04047.

Xinyu Zhang, Shanshan Feng, and Xutao Li. From text to time? rethinking the effectiveness of the large language model for time series forecasting. *arXiv preprint arXiv:2504.08818*, 2025a.

Xiyuan Zhang, Ranak Roy Chowdhury, Rajesh K Gupta, and Jingbo Shang. Large language models for time series: A survey. *arXiv preprint arXiv:2402.01801*, 2024.

Xiyuan Zhang, Boran Han, Haoyang Fang, Abdul Fatir Ansari, Shuai Zhang, Danielle C. Maddix, Cuixiong Hu, Andrew Gordon Wilson, Michael W. Mahoney, Hao Wang, Yan Liu, Huzefa Rangwala, George Karypis, and Bernie Wang. When does multimodality lead to better time series forecasting? 2025b. URL https://arxiv.org/abs/2506.21611.

Yizhou Zhang, Defu Cao, and Yan Liu. Counterfactual neural temporal point process for estimating causal influence of misinformation on social media. *Advances in Neural Information Processing Systems*, 35:10643–10655, 2022.

Haokun Zhao, Xiang Zhang, Jiaqi Wei, Yiwei Xu, Yuting He, Siqi Sun, and Chenyu You. Timeseries-scientist: A general-purpose ai agent for time series analysis. *arXiv preprint arXiv:2510.01538*, 2025.

Andrey Ziyatdinov and Jordi Fonollosa. Gas sensor array under flow modulation. UCI Machine Learning Repository, 2015. DOI: https://doi.org/10.24432/C5BG7G.

Appendix

A Detailed Experimental Setup **17**

B Full Related Work **19**

C More Analysis **20**

C.1 Sensitivity to Model Backbones 20

C.1.1 Effect of LLM Backbone 20

C.1.2 Effect of Forecasting Backbone 21

C.2 Deterministic Task Risk (DTR) Metric 22

C.2.1 DTR Evaluation Results 23

C.2.2 Standard Forecasting Metrics 24

C.3 More Ablation Study 24

C.3.1 Raw-Mean Ablations 24

C.3.2 Case Study on Catastrophic Failure under `w/o Contextual Priors` . 24

C.3.3 Ablation on the Temporal Archetype Bank. 25

C.3.4 Ablation on the Intervention Advisor Agent. 26

C.3.5 Controlled coordination baselines. 27

C.3.6 Paired and Tail-aware Analysis 27

C.3.7 Audit Diagnostics 28

C.4 Computational cost and latency 28

D Case Study **29**

D.1 Qualitative Analysis of Successful Execution 29

D.2 Error Analysis and Dataset Diagnostics 30

E Detailed Agent Workflow **32**

E.1 Intent-Aware Routing Agent 32

E.2 Temporal Scenario Profiler 34

E.3 Temporal Archetype Bank 37

E.3.1 Reference Database Construction 37

E.3.2 Reference Database Composition 38

E.3.3 Retrieval Pipeline 39

E.4 Intervention Advisor Agent 40

E.4.1 Prediction Space 40

E.4.2 Human-Reviewed Intervention Data 40

E.4.3 Supervised Initialization 42

E.4.4 Preference Data Construction 42

E.4.5 Static Reward for Pair Weighting 43
E.4.6 Offline Preference Optimization 43
E.4.7 Evaluation Metrics 43
E.4.8 Effect on Advisor Distribution 44
E.5 Planner & Executor Agent 44
E.5.1 Planner-Facing Report 45
E.5.2 Tool Library 46
E.5.3 Program Output Protocol 47
E.5.4 Prompt Structure 48
E.6 Feature Requirement Analysis 48
E.6.1 Feature Routing 48
E.6.2 Feature Planner 49
E.6.3 Evidence Families 49
E.6.4 Feature Requirement Prompt 50
E.6.5 Output Protocol 51
E.7 Audit Agent 51
E.7.1 Verification Targets 51
E.7.2 Verifier Input 52
E.7.3 Verifier Prompt 52
E.7.4 Verifier Output Protocol 53
E.7.5 Closed-Loop Revision and Fallback 53

A Detailed Experimental Setup

Benchmarks. We evaluate ECLIPSE on two context-rich time-series benchmarks: Context is Key (CiK) and TemporalBench. CiK focuses on contextual time-series forecasting, where each sample provides numerical history together with natural-language context that may contain background information, events, constraints, or scenario descriptions. This benchmark evaluates whether ECLIPSE can convert textual context into improved numerical forecasts. TemporalBench contains a broader set of temporal tasks, including contextual forecasting, event-informed prediction, and temporal QA/MCQ across multiple domains. It is used to evaluate whether ECLIPSE can handle both output forms supported by our framework: numerical trajectories through the Forecast Execution Chain and answers through the Evidence-Grounded QA Chain.

Baselines. For CiK, we follow the baseline protocols used in the original benchmark and categorize our extensive baselines into five distinct groups. First, we consider two LLM-based forecasting paradigms: **Direct Prompting (DP)** and **LLM Processes (LLMP)**. DP treats forecasting as a single-shot generation problem, whereas LLMP treats it as an iterative generation process where the model predicts future values autoregressively. We evaluate these two paradigms across a broad spectrum of open-source LLM families, including Llama-3 (70B), Qwen-3 (8B, 0.6B), Qwen-2.5 (7B, 0.5B), and Mixtral-8x7B. Second, we compare against **Time-Series Foundation Models**, including Lag-Llama, Chronos-Large, TimeGEN, and Moirai-2.0-R-small. Third, we include **Multimodal Models** designed for time-series, specifically UniTime and Time-LLM (ETTh1). Fourth, we benchmark against classical **Statistical Models**, including ARIMA, ETS, and Exponential Smoothing.

Following the CiK evaluation setting, we sample 25 trajectories for the aforementioned baselines when probabilistic forecasts are required. Finally, in our single-sample deterministic evaluations, we include stronger recent LLMs and agent-based systems, such as TimeCopilot, GPT-5.4-mini,

DeepSeek-Chat, and Qwen-3-8B. Due to API and computational costs, these are evaluated with a single sample.

For TemporalBench, we follow the baseline setting of the original benchmark. The compared methods include direct prompting with LLMs, TimeSeriesScientist, AgentScope, CAMEL, and MetaGPT. These baselines cover both direct LLM reasoning and agent-based temporal task solving.

Metrics. For CiK, we use different metrics depending on whether a method produces probabilistic or deterministic forecasts. For models evaluated with 25 forecast samples, we follow the original CiK protocol and report the Region-of-Interest Continuous Ranked Probability Score (RCRPS). RCRPS extends CRPS by emphasizing context-sensitive forecast regions and penalizing task-specific constraint violations. Given a predictive distribution $\tilde{\mathbf{X}}_{1:H}$, ground-truth future $\mathbf{x}_{1:H}$, region of interest $\mathcal{R} \subseteq \{1, \ldots, H\}$, and task-specific violation function $v_{\mathcal{C}}(\cdot)$, we write the score as:

$$\begin{aligned}\mathrm{RCRPS}(\tilde{\mathbf{X}}_{1:H}, \mathbf{x}_{1:H}) = \gamma \Bigg[& \frac{1}{2|\mathcal{R}|} \sum_{t \in \mathcal{R}} \mathrm{CRPS}(\tilde{X}_t, x_t) \\ & + \frac{1}{2|\bar{\mathcal{R}}|} \sum_{t \in \bar{\mathcal{R}}} \mathrm{CRPS}(\tilde{X}_t, x_t) \\ & + \beta\, \mathrm{CRPS}\Big(v_{\mathcal{C}}(\tilde{\mathbf{X}}_{1:H}), 0\Big) \Bigg].\end{aligned} \tag{3}$$

where $\bar{\mathcal{R}}$ denotes non-ROI positions, γ is the task-dependent scale-normalization factor, and β controls the strength of the constraint-violation penalty. Lower RCRPS indicates better probabilistic forecasting performance.

For methods evaluated with a single deterministic forecast (e.g., our single-sample API evaluations), we evaluate performance using the Deterministic Task Risk (DTR) metric. The formal definition and formulation of DTR are detailed in Appendix C.2.

For TemporalBench, we follow its original evaluation setting. QA/MCQ tasks are evaluated by answer accuracy.

Run configuration per table. Table 5 gives the configuration behind each reported result. Each analysis comes from a single evaluation run; Tables 7 and 9 reuse the ECLIPSE runs of Table 1 (their ECLIPSE rows are identical).

Table 5: Configuration of each result. LLM: planner backbone; Adv.: Advisor state (off / SFT / PO = SFT followed by preference optimization); Instances: CiK instances (355 = full set). Retries: one strict-schema retry at the post-adjustment step; there is no global retry budget. The remaining analyses (Tables 13, 14, 16, 17 and Figure 2(a)) use the default pipeline with the Moirai backbone and the Advisor disabled.

Result	LLM	TSFM	Adv.	Instances	Metric	Aggregation
Tab. 1, 7	Qwen3 / GPT-5.4-mini / DeepSeek	Moirai	off	355	RCRPS (25 samples)	mean; ± from 25 samples
Tab. 2	DeepSeek-Chat	Moirai	off	full benchmark	Accuracy	per dataset–task cell
Tab. 8	GPT-4o-mini	Moirai	off	full benchmark	Accuracy	per dataset–task cell
Tab. 9	DeepSeek-Chat	Moirai / Toto / Chronos	off	355	RCRPS (25 samples)	mean; ± from 25 samples
Tab. 10	DeepSeek-Chat	Moirai / Toto / Chronos	off	355 / 343 valid	DTR	mean, median, trimmed
Tab. 11, 12	DeepSeek / GPT-5.4-mini	Moirai	off	355	DTR, MAE, RMSE	mean
Fig. 2(b)	Qwen3-8B	Moirai	off vs. PO	CiK	Success, RCRPS, DTR	mean
Tab. 15	Qwen3-8B	Moirai	off / base / SFT / fixed	355	RCRPS, DTR	median
Tab. 3	DeepSeek-Chat	Moirai	off	355	DTR	mean
Tab. 18	DeepSeek-V4-Flash	–	off	83 corrupted traces	detection, localization, repair	per error type
Tab. 19	DeepSeek-Chat	Moirai	off	25 tasks	cost	mean per task

Why "Full ECLIPSE" differs across tables. Apart from the reused runs noted above, each analysis was produced by its own evaluation run, and the LLM planner is sampled anew in every run, so the full system does not have a single canonical score. Its average DTR is 0.187 in Table 3, 0.194 (DeepSeek) and 0.200 (GPT-5.4-mini) in Table 11, 0.197 in Table 13, and 0.190 in Table 14. In Table 10 the mean is 0.440 but the 10%-trimmed mean is 0.155, because that run contains one exploding instance. The same holds for baselines that were re-run with each analysis: the Moirai-only baseline scores 0.278 in Table 11 and 0.246 in Table 3, and the Advisor-off Qwen3-8B system scores

0.280 in Figure 2(b) and 0.251 in Table 1. We therefore draw every comparison *within* one table, where all methods share the same run, instance set, and aggregation, and never compare numbers across tables. Excluding the tail-dominated run and the GPT-5.4-mini run, the full system spans 0.187–0.197 across runs, a spread of about 0.01 and roughly half the gap between ECLIPSE and the strongest shared-toolchain control (0.187 vs. 0.209). Because each method in Table 3 is still a single run, the component-level conclusions rest on paired per-instance statistics (Tables 16 and 17) rather than on differences of means.

B Full Related Work

Mechanism-level comparison with time-series agents. Table 6 compares ECLIPSE with the closest time-series agents along the mechanisms that distinguish it. Entries for other systems follow their papers' descriptions.

Table 6: Mechanism comparison. ✓: present as a core mechanism; ○: partially or indirectly; ✗: not described.

System	LLM + TSFM/tools	Context-driven history editing	Context-driven future constraints	Retrieval use	Audit loop / QA+forecast
TimeCopilot (Garza & Rosillo, 2025)	✓	✗	✗	✗	✗ / ✗
TimeClaw (Li et al., 2026)	✓	○	○	reasoning-trace memory	○ / ✓
TimeClaw (Liu et al., 2026)	✓	○	○	distilled experience	✗ / ✓
TimeSeriesScientist (Zhao et al., 2025)	✓	○ (preprocessing)	✗	✗	○ / ✗
ECLIPSE	✓	✓	✓	history-completion prior	✓ / ✓

Time-series reasoning as system-level inference. Time-series reasoning extends temporal modeling beyond direct value prediction toward explicit intermediate computation, natural-language questions, what-if scenarios, and decision support (Ye et al., 2024; Kong et al., 2025; Jin et al., 2024). This framing is important for ECLIPSE because context-rich temporal tasks are not solved by producing a forecast or answer in a single forward pass. They also require deciding what information is missing, which output contract should be satisfied, whether the current observation is sufficient, and which computational state transitions should be executed. ECLIPSE follows this system-level view by distributing reasoning across intent-aware routing, belief-state construction, retrieval, intervention advising, temporal program synthesis, tool execution, forecasting, evidence construction, and audit. Unlike language-only rationalization, the intermediate states in ECLIPSE are executable: retrieved references, intervention decisions, history reconditioning/ future scenario enforcement programs, evidence bundles, and audit results can be inspected and checked against the underlying series and task context.

Time-series foundation models and LLM-based forecasting. Time-series forecasting has progressed from classical statistical methods and deep neural models to transformer-based architectures and foundation models trained across broad temporal corpora (Benidis et al., 2022; Wen et al., 2023; Liang et al., 2024; Ye et al., 2025). In parallel, LLM-based forecasting methods have explored serialization, prompting, adaptation, and multimodal prompting (Gruver et al., 2023; Chang et al., 2025; Jiang et al., 2024b; Zhang et al., 2024). These approaches show that language models can provide useful priors and contextual reasoning, but they also expose weaknesses in direct numerical reasoning, tokenization, temporal grounding, and robust zero-shot behavior (Merrill et al., 2024; Spathis & Kawsar, 2024; Zhang et al., 2025a; Tan et al., 2024). ECLIPSE adopts a different division of labor. The LLM is not used as the primary numerical forecaster or as an unconstrained answer generator. Instead, language components resolve the task contract, assess information sufficiency, reconstruct a scenario-specific belief state, synthesize executable temporal programs, and verify consistency. Numerical extrapolation is delegated to specialized time-series forecasting backbones, which are invoked only after the agent has constructed the appropriate computational state.

Multimodal time-series reasoning and grounding. Recent benchmarks, surveys, and position papers argue that time-series systems should handle text, metadata, images, covariates, and natural-language questions in addition to raw signals (Chen et al., 2025; Kong et al., 2025; Zhang et al., 2025b; Jiang et al., 2025a; Wang et al., 2026). A major challenge is multimodal grounding: textual context may describe an event window, a delayed effect, a qualitative mechanism, a hypothetical condition, a hard constraint, or a hidden regime, while the corresponding temporal evidence may be shifted, noisy, partial, or only indirectly observable. Existing multimodal methods often learn joint representations

or adapt LLMs to mixed textual–numerical inputs. TimeXL, for example, demonstrates that LLM modules can interact with an explainable temporal encoder to improve prediction and explanation (Jiang et al., 2025b). ECLIPSE is complementary but operationally different. Rather than relying on a learned alignment layer to implicitly bind every phrase to every timestamp, it externalizes grounding into structured decisions: the router resolves the output contract and information sufficiency, the contextual prior agent parses events and expands scenario skills, the temporal archetype bank grounds abstract context in empirical temporal precedents, and the planner turns these signals into explicit time-indexed operations.

Retrieval-augmented time-series systems. Retrieval-augmented methods have been used to improve adaptation and grounding in time-series forecasting (Tire et al., 2024; Yang et al., 2024; Ning et al., 2025). Agentic retrieval for time-series analysis further suggests that retrieved temporal evidence can support explanation and tool use (Ravuru et al., 2024). Existing retrieval-augmented forecasters mainly use retrieved sequences or patterns as additional conditioning input for the prediction model. ECLIPSE instead treats retrieval as part of belief-state construction under partial observability. The temporal archetype bank provides non-parametric temporal priors that help complete missing scenario information when the local history is too sparse, noisy, or contaminated to reveal the relevant dynamics. Retrieved references are passed to the unified planner and executor as planning evidence rather than blindly injected into the forecasting backbone. The planner then decides how to use them: for reference-based history enrichment, local repair, event-shape transfer, future-side adjustment, or comparative evidence construction for QA. Thus, retrieval in ECLIPSE is not passive prompt augmentation; it is a mechanism for completing the agent's computational state before executable temporal operations are selected.

Agentic tool use and executable planning. LLM agents are increasingly studied as systems that decompose tasks, call tools, plan intermediate steps, and verify outputs (Ke et al., 2025; Shen, 2024; Huang et al., 2024; Ferrag et al., 2025; Hu et al., 2026). Context-rich time-series reasoning imposes additional requirements on such agents: temporal-window validity, numerical plausibility, covariate handling, event-scope grounding, and the distinction between historical-side and future-side interventions. ECLIPSE encodes these requirements through a constrained temporal tool library and an executable program interface. For forecasting, the planner synthesizes a generative state-editing program consisting of historical reconditioning, foundation forecasting, and future scenario enforcement. For QA/MCQ, the same planner switches to an analytical mode and synthesizes an evidence-extraction plan that gathers historical, contextual, comparative, and optional forecast-mediated evidence. This unified planning interface reduces ambiguity compared with free-form chain-of-thought explanations: rationales may be logged for interpretability, but only validated tool names, arguments, evidence objects, and temporal programs affect execution.

What-if simulation and editable temporal dynamics. Counterfactual reasoning is central to decision support, causal inference, and temporal simulation (Kiciman et al., 2023; Zhang et al., 2022). Time-series generation and editing methods also study how to synthesize or modify temporal trajectories under constraints (Cao et al., 2024; Ye et al., 2024). ECLIPSE focuses on scenario-consistent temporal execution rather than causal identification or learning a universal generative model for all hypothetical interventions. In our setting, a what-if prompt, future event, domain rule, or textual constraint is treated as a control signal over the temporal computation. The agent may recondition the observed history when the past is incomplete or contaminated, invoke a forecasting foundation model on the reconditioned state, and then edit the predicted horizon when the future contains events, rules, or hard constraints. Each transformation remains explicit and testable, allowing the system to distinguish between soft priors, executable constraints, and unsupported causal claims.

C More Analysis

C.1 Sensitivity to Model Backbones

C.1.1 Effect of LLM Backbone

We study the sensitivity of ECLIPSE to the LLM backbone used by the agent components on CiK. As shown in Table 7, performance generally improves with stronger LLM backbones. Qwen3-0.6B

Table 7: CiK results under the 25-sample probabilistic evaluation setting with different LLM backbones inside ECLIPSE. We report RCRPS averaged over all tasks and stratified by context type. Lower is better.

LLM Backbone	Average RCRPS	Intemporal Information	Historical Information	Future Information	Covariate Information	Causal Information
Qwen3-0.6B	0.390 ± 0.096	0.286 ± 0.023	0.159 ± 0.008	0.599 ± 0.378	0.284 ± 0.024	0.464 ± 0.064
Qwen3-8B	0.251 ± 0.029	0.286 ± 0.039	0.155 ± 0.010	0.118 ± 0.035	0.283 ± 0.037	0.439 ± 0.061
GPT-5.4-mini	0.190 ± 0.017	0.214 ± 0.018	$\mathbf{0.153 \pm 0.008}$	$\mathbf{0.067 \pm 0.020}$	0.212 ± 0.021	0.395 ± 0.039
DeepSeek-Chat	$\mathbf{0.174 \pm 0.019}$	$\mathbf{0.200 \pm 0.023}$	0.155 ± 0.010	0.075 ± 0.028	$\mathbf{0.190 \pm 0.024}$	$\mathbf{0.292 \pm 0.045}$

Table 8: TemporalBench accuracy comparison under the GPT-4o-mini base model setting. We report Historical Understanding (HU), Contextual Reasoning (CR), and Event-Informed Prediction (EIP).

Agent	**FreshRetailNet**			**PSML**			**Causal Chambers**			**MIMIC**		
	HU	**CR**	**EIP**	**HU**	**CR**	**EIP**	**HU**	**CR**	**EIP**	**HU**	**CR**	**EIP**
Single LLM	63.64	2.89	13.64	67.50	34.80	36.00	13.33	35.20	26.00	46.81	36.61	29.79
TimeSeriesScientist	33.52	3.41	56.82	28.00	21.60	27.33	28.67	21.60	2.67	10.11	28.87	23.40
AgentScope	62.50	13.64	18.94	66.00	27.20	35.33	12.00	44.00	32.00	44.68	23.95	22.70
MetaGPT	62.50	5.11	14.39	67.50	22.00	31.33	10.67	45.20	16.00	45.74	28.97	25.53
CAMEL	64.20	6.25	31.06	68.50	18.40	30.67	10.00	42.00	26.67	46.81	30.14	23.40
ECLIPSE	73.86	30.11	35.61	57.00	29.60	29.33	91.33	70.00	57.33	62.23	31.60	43.26

performs the worst, Qwen3-8B brings a clear improvement, and GPT-5.4-mini and DeepSeek-Chat achieve the strongest overall results. This trend is consistent with our expectation: stronger LLMs provide better context interpretation, intervention planning, and execution validation, which are critical for converting textual context into executable temporal programs.

DeepSeek-Chat obtains the best average RCRPS, while GPT-5.4-mini achieves the best result on future-information tasks. This suggests that the framework benefits from stronger semantic planning ability, but the exact advantage can vary across context types. Overall, the results on CiK show that ECLIPSE scales positively with the capability of its LLM backbone, while maintaining the same execution framework and forecasting interface.

Furthermore, to isolate the structural advantages of our framework from the raw capabilities of the underlying LLM, we investigate performance on TemporalBench when the backbone is constrained to a fixed, lightweight model (GPT-4o-mini). As presented in Table 8, under GPT-4o-mini ECLIPSE leads on 7 of 12 dataset–task cells, with large gains on Causal Chambers and on MIMIC-HU/EIP, but it trails the baselines on all three PSML tasks, on FreshRetailNet-EIP, and on MIMIC-CR. The largest margins are on Causal Chambers, where ECLIPSE reaches 91.33% in HU and 70.00% in CR against 28.67% and 45.20% for the next best baselines. HU questions on this dataset ask about trends, levels, and peaks of the observed signals, which ECLIPSE answers with its generic statistical evidence tools (Appendix E.6) instead of reading values off the serialized series. The Advisor is disabled in these runs, so although its training seeds include Causal Chambers windows (Table 21), this cannot affect the reported accuracies. These results indicate that the structure of the workflow, not only the capability of the backbone, contributes to the gains, while the PSML results show that the benefit is dataset-dependent.

C.1.2 Effect of Forecasting Backbone

Table 9: Effect of different time-series forecasting backbones on CiK under the 25-sample probabilistic evaluation setting. We report RCRPS (lower is better), averaged over all tasks and stratified by context type. DeepSeek-Chat is used as the LLM backbone for all variants.

TSFM Backbone	Average RCRPS	Intemporal Information	Historical Information	Future Information	Covariate Information	Causal Information
Moirai-2.0-R-small	0.174 ± 0.019	0.200 ± 0.023	0.155 ± 0.010	0.075 ± 0.028	0.190 ± 0.024	0.292 ± 0.045
Toto-Open-Base-1.0	3.241 ± 2.740	0.398 ± 0.051	0.202 ± 0.014	0.083 ± 0.030	4.129 ± 3.596	17.960 ± 14.486
Chronos-T5-Large	108.778 ± 101.541	0.280 ± 0.046	0.345 ± 0.134	0.080 ± 0.034	140.993 ± 133.839	643.942 ± 530.120

We further investigate the effect of the numerical forecasting backbone within ECLIPSE by replacing the default forecasting module with three representative time-series foundation models: Moirai-2.0-R-small, Toto-Open-Base-1.0 Cohen et al. (2024), and Chronos-T5-Large. In all settings, the remaining components of ECLIPSE are kept unchanged, and DeepSeek-Chat is used as the LLM backbone. Results on the CiK benchmark are summarized in Table 9.

The results show that the choice of forecasting backbone substantially affects the overall stability of the system. Moirai achieves the strongest overall performance, obtaining the lowest average RCRPS (0.174) together with strong results across most contextual categories. In contrast, Toto and Chronos exhibit severe instability on several settings, particularly on covariate and causal information tasks, where errors occasionally explode to extremely large values. Interestingly, both models still maintain relatively reasonable performance on future-information tasks, suggesting that the contextual intervention mechanisms themselves remain functional, while the underlying numerical forecasting dynamics become unstable under certain execution regimes.

These findings suggest that ECLIPSE is not tied to a specific forecasting backbone, but its overall robustness still depends on the numerical stability and controllability of the underlying foundation model. In particular, forecasting backbones that are more robust to contextual perturbations and temporal state editing appear to interact more reliably with executable intervention pipelines.

Robust statistics across forecasting backbones. The mean DTR of Toto and Chronos is dominated by a single recursive-SVAR instance, making the raw mean a poor summary of their typical behavior. We therefore supplement the probabilistic RCRPS results above with the median and 10%-trimmed mean DTR in Table 10. Under these robust summaries, Moirai remains strongest, while the apparent gap to Toto and Chronos is substantially smaller than suggested by their raw means. The single recursive case contributes 91.1% of Toto's aggregate DTR and 99.7% of Chronos's; after removing it, their mean DTR values decrease to 0.254 and 0.253, respectively.

On the 343 instances valid for both Chronos and Moirai, Chronos records 157 wins, 23 ties, and 163 losses, with a median paired difference of zero. Toto's median paired difference relative to Moirai is 0.00345. Thus, the backbone study supports a modest and robust advantage for Moirai in the full pipeline, while also revealing that the very large raw-mean degradation of alternative backbones is driven by a rare recursive failure rather than broad underperformance.

Table 10: DTR statistics for alternative forecasting backbones. The median and 10%-trimmed mean reduce sensitivity to rare extreme failures. Lower is better.

BACKBONE	VALID N	MEAN DTR	MEDIAN DTR	10% TRIMMED MEAN
Moirai	355	**0.440**	**0.124**	**0.155**
Toto	355	2.857	0.168	0.194
Chronos	343	98.759	0.131	0.165

C.2 DETERMINISTIC TASK RISK (DTR) METRIC

The original Context is Key (CiK) benchmark evaluates probabilistic forecasting performance using the Region-of-Interest Continuous Ranked Probability Score (RCRPS). This metric requires modeling a predictive distribution through repeated sampling (e.g., 25 samples per instance). For closed API models, drawing 25 samples per instance is costly, so we additionally evaluate these models with a single deterministic forecast.

To bridge this evaluation gap, we introduce the **Deterministic Task Risk (DTR)** metric. DTR is designed as a deterministic counterpart to RCRPS that requires only a single generation sample. It is designed to mirror RCRPS's ROI weighting and constraint penalty. As a coarse check of agreement, over the six methods evaluated under both metrics (Moirai, Lag-Llama, Qwen-3-0.6B and Qwen-3-8B direct prompting, and ECLIPSE with DeepSeek-Chat and GPT-5.4-mini; Tables 1 and 11), the Spearman rank correlation between average RCRPS and average DTR is $\rho = 0.83$. With six methods this is only indicative, and the two metrics are computed on different forecasts (25 samples vs. one), so DTR should be read as a separate single-sample risk rather than as a validated surrogate for RCRPS.

Table 11: CiK results under the single-sample deterministic evaluation setting. We report Deterministic Task Risk (DTR), where lower indicates better deterministic task-level performance. Results are averaged over all tasks and stratified by context type. Direct-prompting and TimeCopilot rows use the same LLMs as the ECLIPSE rows.

Model	Average DTR	Intemporal Information	Historical Information	Future Information	Covariate Information	Causal Information
TS Foundation Models						
Moirai-2.0-R-small	0.278 ± 0.024	0.291 ± 0.027	0.157 ± 0.008	0.239 ± 0.060	0.284 ± 0.027	0.521 ± 0.064
Lag-Llama	0.333 ± 0.020	0.370 ± 0.020	0.201 ± 0.007	0.216 ± 0.038	0.342 ± 0.024	0.462 ± 0.051
Direct Prompt						
Qwen-3-0.6B	0.325 ± 0.022	0.371 ± 0.026	0.379 ± 0.039	0.203 ± 0.032	0.305 ± 0.026	0.565 ± 0.049
Qwen-3-8B	0.336 ± 0.030	0.389 ± 0.039	0.310 ± 0.040	0.230 ± 0.047	0.290 ± 0.026	0.543 ± 0.053
DeepSeek-Chat	0.231 ± 0.039	0.283 ± 0.055	0.172 ± 0.014	0.110 ± 0.037	0.230 ± 0.042	0.637 ± 0.117
GPT-5.4-mini	0.226 ± 0.032	0.281 ± 0.043	0.157 ± 0.012	0.088 ± 0.030	0.240 ± 0.040	0.631 ± 0.110
Agentic System						
TimeCopilot (DeepSeek-Chat)	0.229 ± 0.039	0.281 ± 0.054	0.173 ± 0.014	0.106 ± 0.037	0.230 ± 0.043	0.644 ± 0.121
Ours						
ECLIPSE (DeepSeek)	$\mathbf{0.194 \pm 0.020}$	$\mathbf{0.226 \pm 0.023}$	$\mathbf{0.155 \pm 0.010}$	0.077 ± 0.029	$\mathbf{0.215 \pm 0.026}$	$\mathbf{0.364 \pm 0.051}$
ECLIPSE (GPT-5.4-mini)	0.200 ± 0.021	0.230 ± 0.026	0.157 ± 0.009	$\mathbf{0.061 \pm 0.020}$	0.226 ± 0.027	0.500 ± 0.048

Formally, given the ground-truth sequence $y = (y_1, \ldots, y_H)$ and a single deterministic prediction $\hat{y} = (\hat{y}_1, \ldots, \hat{y}_H)$, DTR is defined as:

$$\mathrm{DTR}(y, \hat{y}) = s \cdot E_{\mathrm{base}}(y, \hat{y}) + P_{\mathrm{constraint}}(\hat{y}), \tag{4}$$

where s is a scale-normalization factor, E_{base} is an ROI-aware absolute-error term, and $P_{\mathrm{constraint}}$ penalizes violations of task-specific constraints. DTR measures ROI-weighted point error plus constraint compliance; it is not a replacement for MAE or RMSE.

When a valid region of interest $\mathcal{R}$ is provided, we compute the base error as:

$$E_{\mathrm{base}} = \alpha \cdot \frac{1}{|\mathcal{R}|} \sum_{t \in \mathcal{R}} |y_t - \hat{y}_t| + (1 - \alpha) \cdot \frac{1}{|\bar{\mathcal{R}}|} \sum_{t \in \bar{\mathcal{R}}} |y_t - \hat{y}_t|, \tag{5}$$

where $\bar{\mathcal{R}}$ denotes non-ROI positions and α controls the ROI weight. If no valid ROI is available, E_{base} reduces to the mean absolute error over the full horizon. The ROI and non-ROI errors are each averaged before weighting; we use $\alpha = 0.5$ unless the task metadata specifies another ROI weight. The scale factor is the task's `metric_scaling` from the CiK metadata when available, and otherwise $s = 1/(\max_t y_t - \min_t y_t)$, which makes errors comparable across tasks with different units. The constraint penalty is

$$P_{\mathrm{constraint}}(\hat{y}) = \lambda \sum_c V_c(\hat{y}), \qquad V_{\mathrm{upper}}(\hat{y}) = \frac{1}{H} \sum_{t=1}^{H} \max(s\hat{y}_t - sU,\, 0)\,,$$

with $\lambda = 1$, where each V_c is the mean scaled exceedance of one task constraint (upper bound U as shown; lower bounds and index-specific thresholds analogously), and penalties of multiple constraints are summed. This deterministic penalty differs from the probabilistic CRPS-based violation term of RCRPS (Eq. equation 3), so the two scores are not numerically interchangeable. A lower DTR indicates better task-level deterministic performance.

C.2.1 DTR Evaluation Results

To evaluate deterministic task performance under a strict single-sample budget, we benchmark our proposed ECLIPSE against a diverse set of baselines. As shown in Table 11, our evaluation includes Time-Series Foundation Models (Moirai-2.0-R-small, Lag-Llama), standard LLMs evaluated via Direct Prompting (Qwen-3-8B, DeepSeek-Chat, GPT-5.4-mini), and an existing Agentic System (TimeCopilot).

ECLIPSE obtains the lowest average DTR in this setting. ECLIPSE (DeepSeek) attains 0.194 ± 0.020, compared with 0.231 ± 0.039 for direct prompting and 0.229 ± 0.039 for TimeCopilot with the same base model; overlapping error bars do not by themselves establish significance, and paired comparisons are given in Appendix C.3.6. A similar trend is observed with GPT-5.4-mini, where ECLIPSE reduces the Average DTR from 0.226 ± 0.032 (Direct Prompt) to 0.200 ± 0.021.

Furthermore, our framework exhibits substantial improvements over dedicated time-series foundation models like Moirai and Lag-Llama across nearly all context types.

The largest gap to same-backbone direct prompting and TimeCopilot is on causal-information tasks (0.364 vs. 0.637 and 0.644), where the context specifies mechanisms that must be applied as explicit adjustments.

C.2.2 Standard Forecasting Metrics

Although RCRPS and DTR are the primary metrics because they normalize for task difficulty and probabilistic forecast quality, we additionally report MAE and RMSE in Table 12. ECLIPSE with DeepSeek obtains the lowest aggregate error among the evaluated methods, reducing MAE from 108.413 to 51.401 and RMSE from 241.488 to 104.168 relative to the Moirai backbone. These correspond to relative reductions of 52.6% and 56.9%, respectively. The large dispersion across tasks reflects the heterogeneous scales in CiK, so these unnormalized metrics should be interpreted as complementary evidence rather than as replacements for the normalized primary metrics.

Table 12: Standard forecasting errors on CiK. Lower is better. Because CiK tasks have heterogeneous numerical scales, MAE and RMSE complement rather than replace the normalized primary metrics.

Method	Average MAE	Average RMSE
ECLIPSE (DeepSeek-Chat)	**51.401 ± 101.971**	**104.168 ± 265.533**
Moirai	108.413 ± 180.670	241.488 ± 462.485
Qwen3-0.6B	121.485 ± 23.333	257.572 ± 56.494
Lag-Llama	138.214 ± 27.550	261.777 ± 57.320
Direct Prompt (Qwen3-8B)	140.108 ± 26.516	270.321 ± 56.385
TimeCopilot (DeepSeek-Chat)	248.234 ± 66.058	313.308 ± 81.339
Direct Prompt (DeepSeek-Chat)	248.369 ± 66.050	313.525 ± 81.327

C.3 More Ablation Study

C.3.1 Raw-Mean Ablations

Table 13 reports the full raw-mean ablation moved from the main text. Several ablated means are driven by a handful of exploding instances, so they indicate tail risk rather than typical degradation. Tail-robust statistics for the main components are given in Table 4 and Appendix C.3.6.

Table 13: Ablation results of ECLIPSE under different context and intervention configurations. The table includes component-level, context-level, and intervention-level ablations. Lower DTR is better. Means are dominated by rare catastrophic instances (Appendix C.3.2); see Table 4 for tail-robust statistics.

Model	Average DTR	Intemporal Information	Historical Information	Future Information	Covariate Information	Causal Information
Full ECLIPSE	0.197 ± 0.021	0.231 ± 0.025	0.153 ± 0.008	0.067 ± 0.021	0.227 ± 0.027	0.414 ± 0.043
Ablation on Component						
w/o Contextual Priors	47853.410 ± 36364.130	77301.539 ± 54121.684	0.202 ± 0.038	154602.627 ± 143134.069	61841.273 ± 47675.331	0.461 ± 0.133
w/o Audit Agent	105.198 ± 98.140	0.204 ± 0.019	0.187 ± 0.017	0.066 ± 0.029	135.908 ± 128.822	630.447 ± 519.059
Ablation on Context						
w/o Direct Context	48.530 ± 74.095	78.294 ± 110.909	434.969 ± 365.520	0.084 ± 0.029	0.249 ± 0.039	0.428 ± 0.107
w/o Any Context	0.313 ± 0.037	0.352 ± 0.047	0.181 ± 0.012	0.192 ± 0.040	0.300 ± 0.035	0.486 ± 0.105
Ablation on Intervention						
w/o History Reconditioning	105.645 ± 98.133	0.278 ± 0.069	0.414 ± 0.219	0.065 ± 0.029	136.456 ± 128.811	632.896 ± 518.844
w/o Future Scenario Enforcement	0.268 ± 0.024	0.276 ± 0.027	0.151 ± 0.009	0.239 ± 0.060	0.281 ± 0.026	0.512 ± 0.061

C.3.2 Case Study on Catastrophic Failure under `w/o Contextual Priors`

We further analyze an extreme failure case from the `w/o Contextual Priors` ablation to better understand the role of structured contextual grounding. The affected sample is `IncreasedWithdrawalScenario-4` from the CiK benchmark, whose original context describes an ATM withdrawal forecasting task with a localized carnival event lasting 11 days and causing withdrawals to reach "5 times the usual withdrawals during that period." Under the full

ECLIPSE configuration, this sample achieves stable performance with $\mathrm{DTR} = 0.0777$. However, under the `w/o Contextual Priors` setting, the same sample catastrophically explodes to $\mathrm{DTR} \approx 10^7$.

Inspection of the execution trace reveals that this failure is not caused by the underlying data itself, but by a structurally unstable intervention rule synthesized after removing contextual priors. In the full setting, the system correctly identifies the task as a localized future event and applies a bounded event-shift operation with smooth recovery. In contrast, the ablated system loses both retrieval-based temporal priors and stable task classification, causing the planner to fall back to a generic formula-based intervention:

$$y_t = y_{t-1}(1 + 4 \cdot \mathrm{carnival_indicator}),$$

which effectively becomes $y_t = 5y_{t-1}$ during the event window. Because the carnival lasts for 11 days, the resulting recursive multiplication produces exponential growth and numerical explosion.

Importantly, this failure is not purely random. Other seeds within the same task family remain stable, indicating that the benchmark itself is not inherently pathological. Instead, the ablation exposes a structural vulnerability: without contextual priors, the planner becomes more likely to misinterpret localized future events as generic recursive update rules. This case therefore illustrates why contextual priors are important not only for improving accuracy, but also for constraining the planner toward numerically stable and semantically appropriate intervention strategies.

C.3.3 Ablation on the Temporal Archetype Bank.

Figure 2(a) evaluates the effect of the contextual prior agent's temporal archetype bank on the subset of samples where retrieved references are used for reference-based history enrichment. Bars are normalized per metric (w/o bank $= 1$); raw values are annotated on top. Lower is better.

On this subset, where retrieval is actually used, the temporal archetype bank reduces the error on all four metrics; across the full benchmark its effect is mainly a reduction of tail risk (Table 14). Specifically, on this history-enrichment subset, RCRPS decreases from 0.281 to 0.237, MAE from 41.71 to 29.86, RMSE from 49.64 to 38.80, and DTR from 0.281 to 0.235. This corresponds to substantial relative reductions of approximately 15.7%, 28.4%, 21.8%, and 16.4%, respectively. The drop appears on both point-error metrics and task-aware risks, so it is not tied to a single evaluation criterion.

The strongest improvements arise when the observed history contains only partial-cycle information or when the target dynamics depend on domain-specific periodic structure, such as solar generation profiles and peak timing. In these cases, the temporal archetype Bank supplies a full-cycle prior that the forecasting backbone cannot reliably infer from the short history alone. Thus, retrieval is not used merely as an extra context string appended to the prompt; rather, it provides missing temporal structure that can be grounded into numerical history reconditioning. This supports our broader design principle that external temporal memory is most effective when transformed into executable control over the time-series computation.

Retrieval perturbation. We evaluate retrieval robustness on all 355 CiK instances by replacing the clean top-K archetypes with no retrieval, an irrelevant cross-domain archetype, or a low-quality same-domain archetype. Table 14 shows that median DTR alone does not reveal a consistent advantage for clean retrieval: the irrelevant cross-domain condition attains a slightly lower median. The difference is instead concentrated in the upper tail. Clean retrieval yields the lowest DTR capped at one and the fewest instances with DTR above one, whereas low-quality same-domain retrieval produces the largest capped mean and the most extreme failures. We therefore interpret the bank primarily as reducing tail risk under this perturbation, rather than claiming uniform instance-level improvement from retrieval.

This full-set stress test complements the targeted history-enrichment analysis above. On the subset for which the retrieved archetype supplies relevant historical structure, retrieval reduces RCRPS from 0.281 to 0.237, MAE from 41.71 to 29.86, and RMSE from 49.64 to 38.80. Together, the two analyses distinguish conditional utility from aggregate robustness.

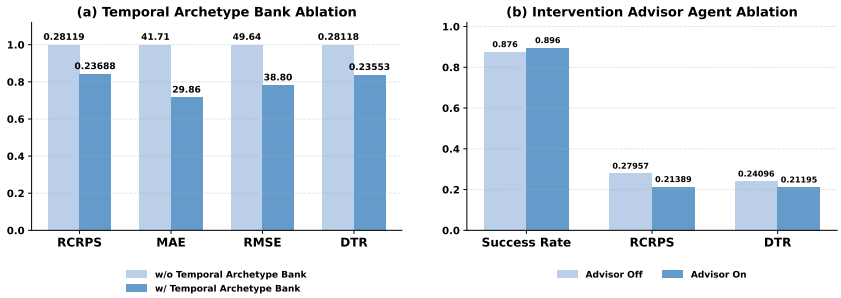


Figure 2: Ablation studies of two key ECLIPSE components. (a) evaluates the temporal archetype bank in the contextual prior agent on samples where retrieved temporal references are used for history enrichment. (b) evaluates the intervention advisor agent (SFT followed by preference optimization) on ECLIPSE with Qwen3-8B as the LLM backbone. For both subfigures, higher Success Rate is better, while lower RCRPS, MAE, RMSE, and DTR indicate better performance.

Table 14: Retrieval perturbation on 355 CiK instances. Lower is better. The capped mean replaces each DTR value by $\min(\mathrm{DTR}, 1)$ before averaging.

Retrieval Condition	Median DTR	Mean DTR	Mean DTR, cap@1	DTR > 1
Clean top-K	0.1222	**0.190**	**0.1829**	**5**
No retrieval	0.1264	1.432	0.1856	6
Irrelevant cross-domain	**0.1195**	0.224	0.1877	7
Low-quality same-domain	0.1266	0.225	0.1953	9

C.3.4 Ablation on the Intervention Advisor Agent.

Figure 2 (b) evaluates the effect of the intervention advisor agent, using the final advisor checkpoint, i.e., SFT followed by compatibility-aware preference optimization. As discussed above, this module is designed to inject human knowledge and domain knowledge into the planning process by providing explicit intervention suggestions to the planner. This design is particularly useful for relatively weaker lightweight open-source models, whose internal planning ability may be less reliable than that of stronger proprietary LLMs. Therefore, we conduct this ablation on ECLIPSE with Qwen3-8B as the LLM backbone.

The results show that enabling the intervention advisor agent improves the agent's performance. The success rate, measured as the proportion of cases that do not produce extreme errors with $\mathrm{MAE} > 150$, increases from 0.876 to 0.896. Meanwhile, RCRPS decreases from 0.280 to 0.214, and DTR decreases from 0.241 to 0.212. These improvements indicate that the advisor not only reduces catastrophic failures, but also improves the quality of successful forecasts under both probabilistic and deterministic evaluation metrics. This supports the motivation of the module: structured external advice can help a lightweight planner make more reliable intervention decisions, especially when the time-series behavior requires domain-specific adjustment rather than direct extrapolation from the observed history alone.

End-to-end effect of Advisor routing. We further evaluate the Advisor while holding the Qwen3-8B Planner, Moirai backbone, retrieval results, temporal tools, and decoding configuration fixed. Table 15 distinguishes the untuned Base Advisor from the supervised fine-tuned (SFT) Advisor. This distinction is consequential: the Base Advisor degrades both median RCRPS and median DTR relative to disabling the module, whereas the SFT Advisor improves both metrics, from 0.1290 to 0.1159 in RCRPS and from 0.1577 to 0.1424 in DTR. Median MAE similarly changes from 3.592 without the Advisor and 3.713 with the Base Advisor to 3.316 with the SFT Advisor. Thus, the result supports the learned routing policy, rather than the unconditional claim that adding an Advisor is inherently beneficial.

The fixed-guidance runs provide a more fine-grained reference. Historical guidance is the strongest individual fixed choice, but no single choice matches the SFT Advisor on both metrics. The hindsight

best-of-four reference selects, for each instance, the fixed guidance yielding the lowest RCRPS and is therefore not a deployable policy or a gold routing label. Its gap to the SFT Advisor quantifies remaining headroom under the tested guidance set. In Advisor-enabled traces, the Planner modifies or extends the Advisor recommendation in 48.3% of cases, indicating that the Advisor acts as a routing prior rather than as a hard controller. As stated in Sections 3.3.3 and 4, the main DeepSeek and GPT experiments disable this optional module. Table 15 covers the Base and SFT checkpoints, whereas Figure 2(b) evaluates the preference-optimized checkpoint against no advisor (and reports means rather than medians); a direct SFT-vs.-preference comparison under one protocol is not included.

Table 15: End-to-end Advisor analysis on 355 CiK instances with a fixed Qwen3-8B Planner and Moirai forecasting backbone. Lower is better. "Hindsight best-of-four" selects the lowest-RCRPS result among the four fixed guidance choices for each instance and is not deployable. The rows "Fixed: Moirai" and "Fixed: future" produced identical results (verified from the run logs).

ADVISOR SETTING	MEDIAN RCRPS	MEDIAN DTR
Base Advisor	0.1445	0.1748
No Advisor	0.1290	0.1577
Fixed: none	0.1285	0.1574
Fixed: Moirai	0.1246	0.1786
Fixed: future	0.1246	0.1786
Fixed: both	0.1251	0.1481
Fixed: historical	0.1196	0.1545
SFT Advisor	**0.1159**	**0.1424**
Hindsight best-of-four	0.0916	0.1162

C.3.5 CONTROLLED COORDINATION BASELINES.

To separate the effect of ECLIPSE's coordination mechanism from access to external resources, we construct three controls that use the same parsed input, retrieved temporal archetype, DeepSeek language model, Moirai forecasting backbone, and temporal tools as ECLIPSE. The Advisor is disabled for all methods in this comparison. The controls differ only in how these shared resources are organized: a single agent selects and invokes the tools, a fixed program applies a predefined sequence of operations, and a retrieval-augmented post-processing baseline adjusts the base forecast using retrieved context without the full Planner–Executor workflow.

The results are reported in Table 3 (Section 5). All methods share the task instances, parsed and retrieved inputs, tools, and models, paired by task ID. Their compute budgets are *not* matched: full ECLIPSE makes about 7 LLM calls per task, whereas the single-agent control makes one (Table 19). Part of ECLIPSE's advantage may therefore come from its additional LLM calls, and the comparison isolates the organization of shared resources, not their cost. Full ECLIPSE obtains the lowest overall DTR, reducing it from 0.209–0.236 for the controlled alternatives to 0.187. Its largest advantages occur on intemporal, covariate, and causal information, where successful prediction requires combining contextual evidence with an executable intervention. The controls remain competitive on simpler cases: the retrieval-plus-post-processing baseline is strongest on future-information tasks, while all methods are similar on historical-information tasks. These results therefore attribute the aggregate gain to structured coordination rather than to tool or backbone access alone, without implying that the full workflow dominates every context type.

C.3.6 PAIRED AND TAIL-AWARE ANALYSIS

Paired ordinary-case gains. Because aggregate means can be dominated by a small number of unstable ratios, we complement Table 13 with paired, tail-robust comparisons. For each comparison, we compute the per-instance DTR improvement favoring the full system and report its 10%-trimmed mean with a bootstrap 95% confidence interval, obtained by resampling paired instance-level differences ($\mathrm{DTR}_{\text{ablated}} - \mathrm{DTR}_{\text{full}}$) with replacement, 20,000 times. Resampling is over individual instances, not task families or seed clusters; because instances from the same CiK task are correlated, these intervals may be somewhat narrower than cluster-level intervals. Table 16 shows a positive full-system improvement over the paired TSFM baseline on all 355 instances. The two intervention

stages also provide positive conditional gains when they are applicable, with the clearest effect on future-context tasks. The win/tie/loss counts show that these improvements are distributed across many instances rather than arising solely from one extreme case.

Tail-aware component removal. We also compare full and ablated runs only when both outputs have DTR at most one. This joint filter is applied symmetrically and is reported with the retained sample size. Removing contextual priors increases DTR by 0.0143 on the 344 retained pairs, while removing future-scenario enforcement increases it by 0.0092 on 316 pairs (Table 17). These estimates support the direction of the aggregate ablations in Table 13 while making explicit how much of the evidence remains after excluding catastrophic ratio failures.

Table 16: Paired, tail-robust DTR improvements. Positive values favor ECLIPSE or the enabled stage. We report the 10%-trimmed mean paired improvement and a 95% bootstrap confidence interval.

COMPARISON	N	IMPROVEMENT	95% CI	IMPROVED/TIED/WORSENED
ECLIPSE vs. paired TSFM	355	0.0170	[0.0090, 0.0270]	205/30/120
History reconditioning, when applied	102	0.0142	[0.0046, 0.0261]	35/40/27
Future-scenario enforcement, when applied	327	0.0104	[0.0026, 0.0203]	172/40/115
Future-context tasks	70	0.0744	[0.0401, 0.1199]	54/10/6

Table 17: Conditional paired ablations after retaining instances for which both the full and ablated runs have $\mathrm{DTR} \leq 1$. Positive degradation indicates that removing the component hurts performance.

REMOVED COMPONENT	JOINT N	DTR DEGRADATION	95% CI
Contextual Priors	344	0.0143	[0.0064, 0.0239]
All Context	341	0.0140	[0.0060, 0.0245]
Future Scenario Enforcement	316	0.0092	[0.0015, 0.0183]

C.3.7 AUDIT DIAGNOSTICS

Injected-error diagnostics. To test whether the Audit stage detects specific execution defects rather than merely producing plausible critiques, we inject one of five error types into otherwise executable traces (83 corrupted traces built from 31 CiK instances) and separately measure detection, localization, and repair. Both the traces and the Audit use DeepSeek-V4-Flash in non-thinking mode. Detection and localization are scored automatically; repair quality is scored by a human annotator on a 0–2 scale, and we report the fraction of suggested repairs that name the wrong field and give a sound minimal fix (score 2). Table 18 shows that the Audit detects every injected error in the supplied evaluation and usually identifies the responsible operation. Successful repair is more difficult: it reaches 0.80–0.93 for temporal scope, direction, and magnitude errors, but falls to 0.35 for invalid tool arguments and 0.30 for unsupported interventions. Detection should therefore not be conflated with automatic recovery; in the latter cases, the appropriate behavior is often to reject or defer the intervention rather than invent a replacement.

We additionally inspect 34 naturally occurring traces that had been automatically selected as clean. A human annotator judged 13 of them to contain genuine defects, and the Audit flags all 13, including unsupported interventions, irrelevant retrieval, and incorrect intervention magnitude or direction. In the supplied under-specified-context evaluation, it flags insufficient evidence in every tested case and yields a 5% hallucinated-mechanism rate. These diagnostics establish a concrete capability boundary, but they do not constitute a formal guarantee of correctness or factual grounding.

C.4 COMPUTATIONAL COST AND LATENCY

Computational cost. Table 19 reports per-task resource use on a 25-task CiK subset. Full ECLIPSE uses 7.2 language-model calls and 21.8K tokens per task, with a mean latency of 21.97 seconds. Relative to the single-agent control using the same forecasting backbone, the structured workflow adds 6.33 seconds per task. The single-agent control reuses the full system's parsing and retrieval outputs; its LLM-call and token counts cover only its own call, while the measured parsing and retrieval time is added back into its reported latency. API-level retries are not separated from these

Table 18: Audit performance under injected execution errors (DeepSeek-V4-Flash, non-thinking). n: number of corrupted traces per error type. Detection: the error is flagged; localization: the responsible operation is identified; repair: fraction of suggested repairs scored 2 by a human annotator (correct field and a sound minimal fix). Higher is better.

INJECTED ERROR	n	DETECTION	LOCALIZATION	REPAIR
Invalid tool or argument	20	1.00	0.70	0.35
Wrong temporal scope	15	1.00	0.93	0.80
Wrong intervention direction	14	1.00	1.00	**0.93**
Wrong magnitude or bound	14	1.00	1.00	0.86
Unsupported intervention	20	1.00	1.00	0.30

Table 19: Average per-task computational cost on a 25-task CiK subset. K denotes thousand tokens. Latency is measured under the reported experimental configuration.

METHOD	LLM CALLS	TOKENS	FORECAST CALLS	LATENCY (S)
TSFM only	0.0	0.0K	1 TSFM	**0.42**
Direct LLM	1.0	2.3K	0	1.39
Single Agent + TSFM	1.0	0.9K	1 TSFM	15.64
ECLIPSE	7.2	21.8K	1 TSFM	21.97
TimeClaw	10.9	109.5K	2.14 classical forecasting tools	65.05

measurements. In this evaluation, only one of the 25 ECLIPSE tasks requires an Audit-induced adjustment. TimeClaw uses more calls, approximately five times as many tokens, and roughly three times the latency under the measured configurations. These measurements describe the tested implementations and hardware conditions; they should not be interpreted as architecture-independent complexity guarantees.

D CASE STUDY

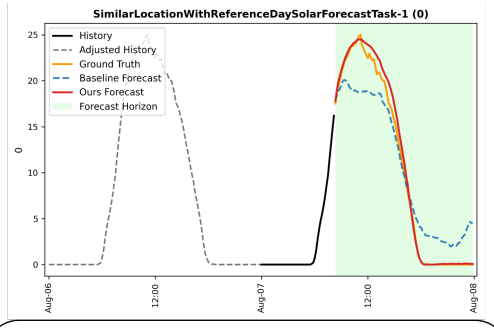
D.1 QUALITATIVE ANALYSIS OF SUCCESSFUL EXECUTION

We further present two representative successful cases from CiK to illustrate how ECLIPSE converts textual context into executable temporal control. The goal is not merely to show examples with lower prediction error, but to highlight two complementary mechanisms in the framework: pre-adjustment through the Temporal Archetype Bank, and post-adjustment through context-derived constraints.

Reference-based historical reconditioning. The left case, `CiK/SimilarLocationWithReferenceDaySolarForecastTask`, illustrates the role of the Temporal Archetype Bank. The provided history only contains limited information about the daily solar cycle, while the context gives a reference from similar locations. A direct baseline forecast underestimates the peak and fails to recover the full daytime decay pattern. ECLIPSE retrieves a reference daily profile and uses it to enrich the historical state before forecasting. This turns the textual reference into an executable historical prior rather than simply appending it to the prompt.

This behavior is consistent with our Temporal Archetype Bank ablation. On the `SimilarLocationWithReferenceDaySolarForecastTask` family, reference-based enrichment reduces RCRPS from 0.2356 to 0.1466, corresponding to a relative reduction of approximately 37.8%. This confirms that the improvement in this type of case comes from completing missing temporal structure, especially when the observed history does not fully reveal the relevant cycle.

Constraint-driven future enforcement. The right case, `CiK/BoundedPredConstraintsBasedOnPredQuantilesTask`, shows a different use of context. Here, the context specifies that the future values should be bounded above. The baseline forecast produces a sharp spike that violates the intended future behavior, even though the historical series contains large past peaks. ECLIPSE



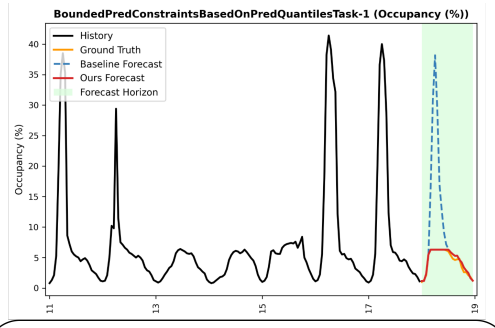
CiK/SimilarLocationWithReferenceDaySolarForecastTask

Context: This series estimates the power production for a given day of a new solar power plant located in the state of Georgia, which has a humid subtropical climate. As reference, the maximal power production in similar states on June 20th was of 25.83 at 11:10:00.

Mechanism: Temporal Pattern Memory + Historical State Reconditioning

TS-Agent retrieves a reference daily solar profile and enriches the short history with a full-cycle prior, allowing the forecast to recover the correct daytime peak and nighttime decay.

Takeaway: Retrieved reference patterns complete missing temporal structure and help the forecast match the true solar cycle.



CiK/BoundedPredConstraintsBasedOnPredQuantilesTask

Context: Suppose that in the forecast, the values are bounded above by 4.36.

Mechanism: Future Scenario Enforcement

The context specifies a future occupancy range based on predictive quantiles. TS-Agent suppresses the implausible baseline spike and enforces a bounded future trajectory.

Takeaway: Context-derived constraints prevent unrealistic extrapolation and keep the forecast aligned with the expected occupancy range.

Figure 3: Representative successful cases from CiK. Left: the Temporal Archetype Bank retrieves a reference solar profile and supports pre-adjustment, allowing ECLIPSE to recover the missing daily cycle. Right: post-adjustment applies a context-derived upper bound, suppressing an implausible baseline spike and producing a trajectory aligned with the specified constraint.

converts the textual bound into post-adjustment, suppressing the implausible spike and producing a trajectory that better follows the ground truth and the context-implied range.

This case highlights why context should not be treated only as auxiliary text. Some contextual information directly defines valid future behavior, but a forecasting backbone may not enforce it automatically. By representing the bound as an executable future-side operation, ECLIPSE can apply the constraint only where it is needed in the forecast horizon.

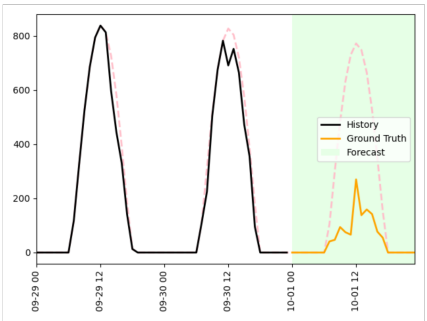
Overall, these cases show two ways in which textual context becomes useful only after being operationalized. In the solar case, context supplies missing historical structure through the Temporal Archetype Bank and pre-adjustment. In the bounded-occupancy case, context imposes a future constraint through post-adjustment. Together, they support the central claim of ECLIPSE: context-rich time-series tasks require executable planning over temporal states, not merely stronger sequence extrapolation or longer prompts.

D.2 Error Analysis and Dataset Diagnostics

We analyze representative failures from CiK to characterize the boundary conditions of context-driven temporal execution. The goal is not to enumerate all incorrect predictions, but to understand when textual context can or cannot be reliably converted into executable temporal control. We focus on three recurring patterns, with representative CiK samples shown in Figure 4.

Incomplete contextual drivers. Some samples provide context that is informative but incomplete. For example, in `CiK/GlobalHorizontalIrradianceFromClearsky`, the clear-sky profile provides a useful prior about daily shape and peak timing, but the realized future trajectory may still depend on weather-related deviations that are not specified in the prompt. In such cases, the Temporal Archetype Bank or commonsense priors can recover the expected periodic structure, but the task remains partially underidentified without the missing future driver. This suggests that some contextual information should be treated as a soft prior rather than a hard future constraint.

Incomplete Contextual Drivers



CiK/GlobalHorizontalIrradianceFromClearsky

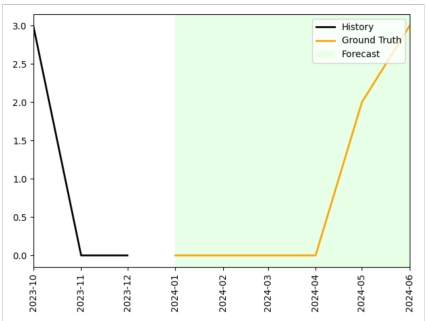
Context: This series contains Global Horizontal Irradiance for a location in Oaxaca, Mexico. Global Horizontal Irradiance is the total amount of sun energy arriving on a horizontal surface.

The context provides a cloudless-sky reference profile: [0,0,...,675,1090,857,...,0]

The full context continues with similar cloudless-sky values for 2022-05-15 and 2022-05-16.

Diagnostic: The context provides a clear-sky upper-envelope prior, but does not specify cloud/weather deviations. Thus, the future trajectory is underdetermined by the provided context.

Weakly Grounded Analogical Transfer



CiK/MontrealFireNauticalRescueAnalogy-FullLocalizationMaybeWaterTask

Context: Montreal recently inaugurated a new borough, Baie-des-Marins, adjacent to the Saint-Lawrence River. The Fire Chief seeks an estimate of nautical rescues for the next 6 months.

For reference, historical May--September incident counts in two boroughs are provided: · Ahuntsic-Cartierville, major bodies of water: [9, 25] · Le Plateau-Mont-Royal, no major bodies of water: [0, 1].

This context gives an analogy based on water access, but the new borough itself has no historical rescue records.

Diagnostic: The prompt gives reference boroughs with/without major water access, but Baie-des-Marins has no own history, making the forecast highly dependent on analogy transfer.

Figure 4: Representative failure cases from CiK. Left: the context provides a clear-sky irradiance profile, which gives useful daily-shape information but omits cloud/weather deviations that determine the realized trajectory. Right: the context provides a water-access analogy for a new borough, but the target location has no historical rescue records, making the forecast dependent on weakly grounded analogy transfer.

Weakly grounded analogical transfer. Another failure mode appears in analogy-heavy tasks. For example, in `CiK/MontrealFireNauticalRescueAnalogy`, the prompt provides geographic or domain-level analogy information, but the available numerical history is too short to determine how that semantic relationship should quantitatively affect the future values. Here the bottleneck is not sequence extrapolation alone, but grounding semantic similarity into a calibrated temporal intervention. These cases reveal the difficulty of converting weakly grounded analogies into executable control signals.

Overall, these cases highlight a distinction between *context availability* and *context actionability*. ECLIPSE is most effective when textual information can be grounded into pre-adjustment, post-adjustment, Temporal Archetype Bank retrieval, or structured evidence construction. When context is incomplete, weakly grounded, or overly ambiguous, the system requires stronger uncertainty estimation and more conservative intervention policies.

E Detailed Agent Workflow

E.1 Intent-Aware Routing Agent

ECLIPSE uses an Intent-Aware Routing Agent as the unified entry module before invoking any downstream execution module. Its purpose is to determine the required output contract of a task, rather than solving the task itself. This distinction is important because many temporal QA tasks may require forecasting-like computation internally, while their final deliverable is still an answer, label, or structured response. The router therefore dispatches tasks according to the final required deliverable: numerical-trajectory tasks are sent to the Planner & Executor Agent, while answer-format tasks are sent to the Feature Extractor.

Routing space. The resolver predicts one of two top-level task families:

$$\mathcal{F} \in \{\texttt{qa}, \texttt{forecast}\}.$$

The label `qa` is used when the final output is a choice, label, short answer, or structured answer field. The label `forecast` is used only when the final required output is a numerical future time-series sequence evaluated by forecasting metrics. Thus, the resolver is a deliverable-level dispatcher rather than an open-ended semantic task classifier.

Router prompt. We use a dedicated system prompt to enforce deliverable-level routing.

Prompt 1: Intent-Aware Routing Agent System Prompt

```
You are a task-family router for a time-series multi-agent system.
Classify each input into exactly one top-level family:

1. qa:
   The task requires selecting or outputting an answer, label, option,
   or structured response, even if solving it may require forecasting
    internally.

2. forecast:
   The task primarily requires generating a future time-series forecast
   and will be evaluated with forecasting metrics.

Decision rule:
Classify by the FINAL REQUIRED DELIVERABLE, not by background context.

If the prompt asks the model to choose an option, answer sub-questions,
return labels, fill answer fields, or output categorical/structured
    answers,
choose qa, even if the prompt also requires internal forecasting.

Choose forecast only when the final required output is a numeric future
time-series forecast array or sequence.
```

```
When the prompt is long, pay special attention to the tail section
    containing
"Output format", "Constraints", or final instructions. These override
    earlier
background text.

Return JSON only with keys: task_family, confidence, rationale.
```

User message template. Rather than passing the entire prompt verbatim to the resolver, ECLIPSE constructs a compact head–tail preview.

Input Template: Router User Message

```
{
  "sample_id": "<sample_id>",
  "tier": "<tier>",
  "prompt_head": "<prompt prefix>",
  "prompt_tail": "<prompt suffix>"
}
```

The `prompt_head` captures task background and high-level semantics, while the `prompt_tail` captures final output constraints and instructions. This design reduces token overhead while preserving the information most relevant to routing, since the final output contract is often specified near the end of long prompts.

Output schema. The resolver is required to output a parseable JSON object following the schema below.

Schema 1: Router Output Protocol

```
{
  "task_family": "qa" | "forecast",
  "confidence": <float>,
  "rationale": "<short explanation>"
}
```

The `task_family` field is the dispatch signal used by ECLIPSE. The `confidence` and `rationale` fields are retained for trace inspection and routing-error analysis. Requiring JSON output prevents downstream execution from depending on unconstrained natural-language explanations.

Output examples. Examples of resolver outputs are shown below.

Example 1: QA Route

```
{
  "task_family": "qa",
  "confidence": 0.92,
  "rationale": "The prompt requires an answer or option rather than a
  numerical forecast sequence."
}
```

Example 2: Forecast Route

```
{
  "task_family": "forecast",
  "confidence": 0.88,
```

```
  "rationale": "The final output format requires a numeric future
    forecast
  array."
}
```

Dispatch behavior. After the JSON decision is parsed, the dispatcher invokes the corresponding downstream chain:

`qa` → Feature Extractor/Contextual Prior Agent, `forecast` → Contextual Prior Agent.

The resolver itself does not produce the final answer or forecast. It produces a structured execution-path decision that determines which specialized pipeline should be used. For mixed-format tasks, ECLIPSE may also construct separate task views, such as a QA-preserving view and a forecast-only view, so that the same underlying temporal scenario can be evaluated under different output contracts without forcing a single ambiguous route.

E.2 Temporal Scenario Profiler

After the task has been routed to the Planner & Executor Agent, ECLIPSE applies a **Temporal Scenario Profiler** inside the Contextual Prior Agent. Its role is not to generate forecasts or intervention actions directly. Instead, it maps the textual context into a coarse temporal scenario category, which is then used as high-priority guidance for planning.

This module addresses an important ambiguity in context-rich forecasting tasks: different forms of textual context imply fundamentally different planning strategies. A historical artifact should bias the agent toward History Reconditioning, a future event should bias it toward Future Scenario Enforcement, a rule-based prompt should prioritize formula-driven execution, and indirect analogies may require Temporal Archetype Bank retrieval. The profiler therefore acts as an intermediate semantic abstraction layer between raw context and executable planning. The category set was designed by the authors with the context types of contextual forecasting benchmarks in view, including the task families of CiK; it is therefore aligned with the benchmark we evaluate on, and its coverage of other domains is untested (Section 6).

Scenario categories. The profiler predicts exactly one category from the following fixed set:

$$\mathcal{Z} = \{\textsc{AddFutureConstraint}, \textsc{CommonsenseInference}, \textsc{FutureTimedEvent}, \textsc{PastTimedEvent}, \textsc{PastAndFutureEvents}, \textsc{RuleBased}, \textsc{IndirectReasoning}\}.$$

These categories are not intended as a general-purpose taxonomy of all time-series tasks. They are a planning-oriented ontology designed to indicate which intervention family should be emphasized downstream.

Classifier prompt. The classifier prompt defines the category space, gives compact prototypes, and specifies priority rules for resolving overlapping signals.

Prompt 2: Temporal Scenario Profiler System Prompt

```
You are a task classifier for context-rich time-series forecasting
    tasks.
Read the provided context and classify it into exactly one category.

Categories:
1) AddFutureConstraint:
   The future must satisfy additional bounds, constraints, or
    restrictions.

2) CommonsenseInference:
   The context gives domain background, but the behavior must be
    inferred
```

```
   from commonsense or physical/domain knowledge.

3) FutureTimedEvent:
   An explicit event or intervention occurs only in the future horizon.

4) PastTimedEvent:
   An explicit event occurred in the historical window and should not
   be carried into the future.

5) PastAndFutureEvents:
   Related events appear in both historical and future windows.

6) RuleBased:
   Explicit formulas, equations, or rules are provided and should drive
   the forecast behavior.

7) IndirectReasoning:
   The context provides indirect clues, analogies, or similar-scenario
   references rather than direct event or rule specifications.

Priority rules:
- If explicit formulas or equations exist, choose RuleBased.
- If explicit future bounds exist without event timing, choose
    AddFutureConstraint.
- If explicit event timing exists, decide FutureTimedEvent,
    PastTimedEvent,
  or PastAndFutureEvents based on its time scope.
- If no explicit event or constraint is given but domain background
    implies
  behavior, choose CommonsenseInference.
- If the context relies on analogy or similar-scenario clues, choose
  IndirectReasoning.

Return JSON only with keys: category, confidence, reason.
```

Input template. The profiler receives a structured payload rather than an unconstrained prompt. This explicitly restricts the output space and reduces category drift.

Input Template: Scenario Profiler User Message

```
{
  "context": "<task context text>",
  "allowed_categories": [
    "AddFutureConstraint",
    "CommonsenseInference",
    "FutureTimedEvent",
    "PastTimedEvent",
    "PastAndFutureEvents",
    "RuleBased",
    "IndirectReasoning"
  ],
  "output_schema": {
    "category": "string",
    "confidence": "0.0-1.0",
    "reason": "short explanation"
  }
}
```

Output schema. The profiler returns a structured JSON object:

Schema 2: Scenario Profiler Output Protocol

```
{
  "category": "<one of the allowed scenario categories>",
  "confidence": <float>,
  "reason": "<short explanation>"
}
```

The `category` field is used as a planning signal. The `confidence` and `reason` fields are stored in the trace for debugging and error analysis.

Output examples. Two representative outputs are shown below.

Example 3: Commonsense Scenario

```
{
  "category": "CommonsenseInference",
  "confidence": 0.91,
  "reason": "The context describes solar irradiance and implies daytime
  periodicity and near-zero nighttime values, but gives no explicit
    event
  or formula."
}
```

Example 4: Historical Event Scenario

```
{
  "category": "PastTimedEvent",
  "confidence": 0.88,
  "reason": "The context indicates a historical event whose effect
    should
  not recur during the forecast horizon."
}
```

Auxiliary structural attributes. In addition to the scenario category, the profiler may attach lightweight structural metadata used by the planner, including forecast structure and covariate availability. For example, it can indicate whether the task is univariate or multivariate, identify the main target variable, and list available historical or future covariates. These attributes help the planner avoid unnecessary assumptions, such as inventing a future covariate schedule when one is already provided.

Planner guidance. The profiler output is not used only for logging. It is converted into a compact guidance block injected into the Planner & Executor Agent.

Guidance Block: Scenario-Specific Planning Signal

```
[Temporal Scenario Guidance]
Classifier output:
- category: <category>
- confidence: <confidence>
- reason: <reason>
- forecast_structure: <univariate | multivariate | unknown>
- covariate_mode: <with_covariates | no_covariates | unknown>
- future_covariate_keys: <...>

Planner instruction:
Use the scenario category as high-priority planning guidance.
```

```
Verify semantic consistency with the original context and physical
    plausibility.

Category guidance:
<category-specific guidance sentence>
```

The category-specific sentence biases the planner toward the appropriate intervention family. For example, PASTTIMEDEVENT encourages local historical cleanup, FUTURETIMEDEVENT encourages future-localized enforcement, COMMONSENSEINFERENCE and INDIRECTREASONING encourage reference-based history enrichment when appropriate, and RULEBASED prioritizes explicit formula or rule execution.

E.3 TEMPORAL ARCHETYPE BANK

The **Temporal Archetype Bank** externalizes prior knowledge of typical temporal shapes for ECLIPSE to consult at planning time. In many context-driven tasks, the textual prompt C asserts future shapes that a short observed history $X_{1:T}$ does not itself reveal (e.g., "rush hour traffic congestion expected"). Without historical references, the planner is forced to extrapolate from an under-determined history or to fabricate the missing structure. The Temporal Archetype Bank addresses this gap by retrieving real, previously observed segments that aid the agent's temporal state interventions.

The design of this memory module addresses two pivotal questions: (1) what the reference database should contain as typical time series shapes and (2) what retrieval signal ensures that both the numerical evidence in $X_{1:T}$ and the semantic evidence in C are jointly grounded. In planning time, given a query window and a textual descriptor, we rank candidates by shape distance and by description similarity, fuse the ranks, and return top-K medoids. An overview of the Temporal Archetype Bank design is shown in Figure 5.

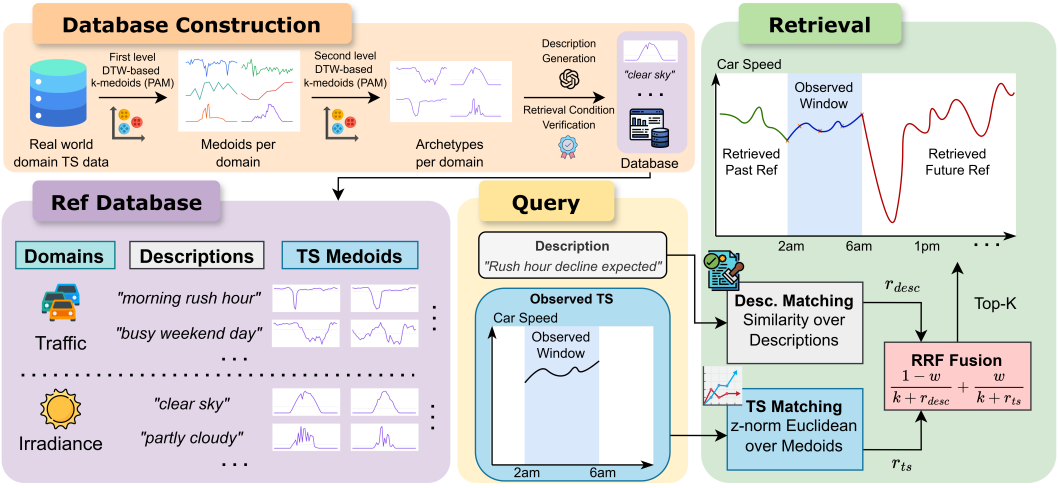


Figure 5: **Temporal Archetype Bank**. The offline construction stage (top) condenses raw multi-domain time series into a two-level hierarchy of medoids; descriptions are then generated for the coarser layer and validated by retrieval conditions. At query time (bottom), an observed partial cycle and a textual descriptor derived from C are ranked against database candidates, and their ranks are fused with Reciprocal Rank Fusion. The retrieved candidate provides historical and future references to the planner for context.

E.3.1 REFERENCE DATABASE CONSTRUCTION

The reference database is a non-parametric summary of the space of temporal shapes that occur within each domain. Its design balances both retrieval efficiency and candidate diversity. The database is built offline with a fixed procedure for each domain: two successive rounds of Dynamic Time

Warping (DTW) k-medoids clustering condense raw cycles first into shape regimes and then into macro-archetypes; a vision-language model labeler produces a textual description for each archetype; and a self-retrieval verification step admits a description only if it can recover its own archetype. We describe each step and note the design choices it embodies.

Two-level shape clustering. We first normalize raw time series cycles and run k-medoids with DTW in the Partitioning Around Medoids (PAM) form. DTW absorbs phase variation in raw time series data, so the medoid budget is spent on distinct shape regimes rather than on phase-shifted variants of the same regime, and medoids in place of centroids ensure retrieved patterns are real observed cycles. This step produces fine-grained representatives of the level 1 (L1) regimes, often on the order of 10^3 medoids, at which scale free-form labeling collapses, as a labeler asked to describe 10^3 profiles would produce uninformative near-duplicate labels. We therefore run a second pass of DTW k-medoids over the L1 medoids to recover a smaller set of level 2 (L2) macro-archetypes that capture coarse-grained temporal regimes. Each L1 medoid inherits its parent L2 archetype's identifier, so shape retrieval continues to discriminate within an archetype while description retrieval ranks at the archetype level.

Archetype description generation. We use a vision language labeler to generate concise descriptions for each archetype's temporal characteristics by passing in its plot and metadata, including domain, frequency, unit of measure, cycle lengths, etc. The prompt asks for a macro-shape semantic that ignores low-level numeric fluctuation (e.g., "early_morning_trough_evening_peak").

Retrieval condition verification. A generative labeler can place visually distinct archetypes near each other in the embedding space, silently degrading the description channel. We close the loop with a truncation-tolerant self-retrieval test: a label is admitted only if it can be used to retrieve its original archetype with itself and its 1- and 2-word-truncated variants, ensuring robust and distinctive archetype descriptions.

E.3.2 Reference Database Composition

Pattern record schema. Each L1 entry exposes the minimal information that the planner needs to use the pattern as evidence: an identifier, a textual description, the cycle metadata under which the series was sampled, the z-scored series itself, the pair (μ, σ) that maps the series back to its physical scale, and a pointer to the parent archetype. The pointer is what links the L1 shape index to the L2 description index at query time; it is what allows description-side ranking to be performed on the small archetype set even though shape-side ranking is performed on the full L1 set.

Schema: L1 pattern record

```
{
  "pattern_id":           "metr_la_speed_cluster17_v1",
  "domain":               "traffic_speed",
  "freq":                 "30min",
  "cycle_length":         48,
  "description":          "rush_hour_double_peak",
  "series":               [...],          // z-scored, length =
    cycle_length
  "normalization_params": {"mean": 58.7, "std": 6.4},
  "cluster_meta": {
    "cluster_id":   17,
    "cluster_size": 312,
    "archetype_id": "metr_la_speed_a3"
  }
}
```

Coverage. Table 20 reports the empirical composition of the current database. The 14 domain entries cover common real world time series domains. Note that for data-scarce domains with limited profiles, the L2 archetype layer is omitted, because it would not be meaningfully distinct from the L1 medoid set, and in this case, a textual description is generated for each L1 medoid.

Table 20: Reference-database composition. *Profiles* is the number of cycles fed into L1 clustering after filtering and stratified subsampling. k_1 is the L1 medoid count; k_2 is the L2 archetype count (– when the L2 stage is omitted).

Domain entry	Native freq	Cycle	Profiles	k_1	k_2
etth1_etth2	1h	24	847	80	10
electricity	30min	48	653	60	10
metr_la	30min	48	13,453	1,000	10
pems_bay	30min	48	15,000	1,000	10
solar_ghi	30min	48	14,793	1,000	10
solar_dhi	30min	48	14,793	1,000	10
solar_dni	30min	48	14,793	1,000	10
exchange_rate	1d	7	758	70	10
illness	1w	52	13	4	–
field_fire	1M	12	40	4	–
trash_fire	1M	12	40	4	–
nautical_rescue	1M	12	24	4	–
ice_rescue	1M	12	16	4	–
unemployment	1M	–	1	1	–

Evaluation contamination control. Where the database overlaps with our evaluation benchmarks, we are careful to construct it from non-overlapping data so that retrieval cannot leak test labels into the planner. For example, solar irradiance, fire incident, and rescue incident entries are built from years not present in the CiK benchmark, and for domains with native train/test splits, only the training split is used for clustering. This separation keeps test targets out of the retrieval bank.

E.3.3 Retrieval Pipeline

In planning time, a query is parameterized by a domain tag, an observed window $\mathbf{q} = X_{T-L_q:T}$ (where L_q is the length of the $\mathbf{q}$), an optional cycle phase ϕ indicating where $\mathbf{q}$ falls within a full cycle, and an optional textual descriptor q_{desc} produced by the upstream Contextual Prior Agent. We design the retrieval pipeline to combine two complementary retrieval signals: one operating on the numerical shape of $\mathbf{q}$, the other operating on the semantics of q_{desc}, and fuse them with customizable weights on their rank scales.

Shape signal r_{ts} The shape signal compares the query window against each candidate's slot at the corresponding cycle position under z-normalized Euclidean distance.

$$d_{\text{ts}}(\mathbf{q}, \mathbf{p}, \phi) = \left\| \tilde{\mathbf{q}} - \tilde{\mathbf{p}}[\phi : \phi + L_q] \right\|_2,$$

where the tilde denotes per-window z-normalization and $\mathbf{p}$ denotes a candidate series. When ϕ is unknown, we fall back to free search over the full candidate cycle using Mueen's Algorithm for Similarity Search (MASS) (Mueen et al., 2022); phase-anchored matching is preferred when available because it provides deterministic prefix and suffix context that the planner can reliably interpret. A query that exhausts the cycle before L_q slots have been compared is handled by tiling the candidate against itself, so that the suffix continues to cover at least one full cycle.

Description signal r_{desc}. The description signal scores each candidate by the cosine similarity between an embedding of q_{desc} and an embedding of the candidate's parent archetype label,

$$s_{\text{desc}}(i) = \cos\big(\, \text{enc}(q_{\text{desc}}),\ \text{enc}(\ell_{a(i)}) \,\big),$$

where $\ell_{a(i)}$ is the textual description of the archetype identifier of L1 medoid i. Crucially, all L1 medoids inheriting the same archetype share their parent's description rank. This means that the language descriptor q_{desc} serves as a high level semantic guidance instead of a fine-grained shape distinction that separates two mildly different medoids.

Rank fusion. The two signals are combined with Reciprocal Rank Fusion (RRF) (Cormack et al., 2009).

$$\text{score}(i) = \frac{1-w}{k + r_{\text{ts}}(i)} + \frac{w}{k + r_{\text{desc}}(i)},$$

where w, a hyperparameter, denotes the weight on the description path, and k denotes the RRF smoothing constant. We recommend $w = 0.3$ to keep textual description as a coarse archetype-level prior while letting shape dominate fine-grained ranking. We use $k = 10$ as the default for sharper ranking. The top-K retrieved medoids are returned to the planner for further operations.

E.4 Intervention Advisor Agent

ECLIPSE optionally uses an **Intervention Advisor Agent** as a learned soft guidance signal for the Planner Agent. The module is disabled in all main experiments (Section 3.3.3). Its purpose is not to execute interventions directly, but to inject human-reviewed intervention knowledge into the planner-facing report. Given a structured event/context description, the advisor predicts whether the forecasting task is likely to benefit from Historical State Reconditioning, Future Scenario Enforcement, both, or no explicit intervention. The final executable decision is still made by the Planner Agent, which jointly considers the original context, Temporal Scenario Profiler output, Temporal Archetype Bank, Tool Library constraints, and Audit Agent's feedback.

E.4.1 Prediction Space

The advisor predicts one of four intervention modes:

$$\mathcal{Y} = \{\texttt{hist}, \texttt{future}, \texttt{both}, \texttt{none}\},$$

where `hist` corresponds to Historical State Reconditioning, `future` corresponds to Future Scenario Enforcement, `both` indicates that both intervention directions may be useful, and `none` indicates that no explicit temporal intervention is recommended.

We treat these labels as structured intervention sets rather than unrelated classes:

$$\texttt{none} = \emptyset, \qquad \texttt{hist} = \{\text{hist}\}, \qquad \texttt{future} = \{\text{future}\}, \qquad \texttt{both} = \{\text{hist}, \text{future}\}.$$

This structure is important because `both` is partially compatible with either single-sided intervention. For example, if the desired behavior requires Future Scenario Enforcement, predicting `both` still preserves the future-side intervention, whereas predicting `none` removes all intervention capacity. Therefore, in addition to strict four-way classification metrics, we also evaluate compatibility-aware behavior.

E.4.2 Human-Reviewed Intervention Data

We construct the supervision data for the Intervention Advisor Agent through a semi-automatic, LLM-assisted event-hypothesis generation and human-review pipeline. The goal is to identify whether a temporal window contains an event, constraint, or state change that requires intervention, and to determine the appropriate adjustment strategy.

Rather than asking human annotators to directly label raw time-series windows from scratch, we shift the annotation unit to an *event-level intervention hypothesis*. The overall construction pipeline is defined as:

$$\begin{aligned}\text{raw time series} &\to \text{seed windows} \to \text{multi-model trigger proposals} \\ &\to \text{cross-model merge} \to \text{event-level review table} \\ &\to \text{human correction} \to \text{final reviewed dataset}\end{aligned}$$

This design has two key motivations. First, it reduces the human annotation cost by leveraging Large Language Models (LLMs) to propose and pre-label candidate triggers, shifting the human role from generation to verification. Second, it aligns the supervision signal with the advisor's intended role: learning to map contextual triggers to specific intervention modes.

Seed-Series Construction and Compression. To ensure the advisor learns generalizable intervention patterns rather than dataset-specific biases, we begin by sampling representative historical windows from a highly diverse catalog of public and synthetic time-series benchmarks. As detailed in Table 21, our seed pool covers 6 distinct domains, resulting in 94 representative seed samples.

Crucially, neither the proposing models nor the human reviewers operate on unstructured raw CSV files. Each raw window is compressed into a structured view containing `sample_id`,

Table 21: Source domains and representative datasets used for seed-series construction.

Domain	Source Datasets
Demand & Retail	Bike Sharing Fanaee-T (2013), Dominick's grocery sales James M. Kilts Center for Marketing, Kaggle classic time-series bundle shenba (2017)
Utility & Energy	ElectricityLoadDiagrams20112014 Trindade (2015), Household Electric Power Hebrail & Berard (2006), Appliances Energy Candanedo (2017), SML2010 Romeu-Guallart & Zamora-Martinez (2014)
Traffic	METR-LA, PEMS-BAY Li et al. (2018)
Air Quality & Env.	Beijing PM2.5 Chen (2015), PM2.5 Five Chinese Cities Chen (2016), Air Quality Vito (2008), Gas Sensor Array Ziyatdinov & Fonollosa (2015)
Physiology & Medical	BIDMC PPG/Respiration Pimentel et al. (2017)
Causal & Control	Causal Chambers Gamella et al. (2024), Synthetic Formula Intervention Bank

`domain`, a natural language `background` description, `target_name`, `frequency`, and available `covariate_names`. Furthermore, the raw data is aggregated into a `history_summary` that captures statistical features such as sequence length, start/end timestamps, min/max/mean, recent trend, recent percentage change, and head/tail previews. This compact representation forces the pipeline to focus on state changes and constraints rather than raw noise.

Multi-Model Trigger Proposals. For each structured seed sample, we query multiple LLMs to propose candidate events that might necessitate forecasting adjustments. We utilize a diverse model pool, `gpt-4o-mini`, `deepseek-chat`, and `grok-4.1-fast-non-reasoning`, to increase trigger recall and mitigate biases from any single annotator model.

Each model receives the compressed history summary and outputs a structured JSON object containing `event_candidates`, `primary_recommendation`, and `global_uncertainty`. Each candidate is generated based on a rich predefined adjustment ontology (e.g., event type, trigger scope, actionability basis, polarity, magnitude) and includes an initial `recommended_mode`. The models pre-label the candidates using a legacy action space of $\{\texttt{pre_adjust}, \texttt{post_adjust}, \texttt{both}, \texttt{none}\}$, which maps to our formal notation as $\texttt{pre_adjust} \mapsto \texttt{hist}$ and $\texttt{post_adjust} \mapsto \texttt{future}$.

Cross-Model Merging and Action Prelabeling. Because different models frequently propose variations of the same underlying event for a given seed sample, we cluster the proposals into canonical event-level hypotheses. The merging step groups events based on event type, trigger scope, polarity, time buckets, and semantic keyword overlap. Through this process, the 94 seed samples are expanded and deduplicated into approximately 652 distinct canonical event hypotheses.

To further streamline human verification, we apply a secondary model pass using `deepseek_chat` to assign a preliminary `final_action` recommendation (along with confidence and rationale) to each merged canonical event. Reviewers are then presented with an aggregated event-level review table containing the representative event summary, the supporting models' evidence, vote counts, and the pre-labeled action hint.

Human Correction and Final Dataset. Human reviewers validate, correct, or override the model's action recommendation to finalize the label. The review was performed by five MS and PhD students. Records were labeled independently by multiple reviewers, with an inter-annotator agreement of 92.4%. The final label space is defined as $\mathcal{Y} = \{\texttt{hist}, \texttt{future}, \texttt{both}, \texttt{none}\}$. A label of `hist` indicates Historical State Reconditioning (e.g., repairing corrupted history); `future` indicates Future Scenario Enforcement (e.g., applying a future hard constraint); `both` implies explicit interventions are needed on both sides; and `none` means the event is not actionable enough for intervention.

The final human-reviewed records are split into training and test sets, yielding 404 total event-level samples. The distribution of these human-verified records is detailed in Table 22.

Table 22: Distribution of intervention modes in the final human-reviewed dataset.

Split	`future`	`hist`	`none`	`both`	Total
Train	261	34	16	12	**323**
Test	65	9	4	3	**81**
Total	**326**	**43**	**20**	**15**	**404**

As shown in Table 22, the dataset distribution is highly imbalanced. Future-side interventions heavily dominate, while true two-sided (`both`) interventions are exceedingly rare. This reflects the structure of context-rich forecasting tasks, where external context most frequently dictates future constraints. With only 3 `both` and 4 `none` test records, per-class metrics for the rare classes are highly uncertain. The scarcity of rare labels motivates the compatibility-aware preference refinement described below.

E.4.3 Supervised Initialization

We first train the advisor with supervised fine-tuning. Each reviewed example is converted into an instruction-style input containing the structured event/context fields, and the target output is a minimal JSON object:

Schema 12: Intervention Advisor Agent Output

```
{
  "recommended_mode": "hist" | "future" | "both" | "none"
}
```

The supervised objective is:

$$\mathcal{L}_{\mathrm{SFT}} = -\log p_\theta(y^\star \mid x),$$

where x is the structured event/context input and $y^\star$ is the reviewed intervention label. This stage teaches the model to produce valid JSON and to learn the basic mapping from structured context to intervention mode. However, because the data is strongly skewed toward `future`, a strict one-hot objective can under-represent rare but important compatible modes such as `both`.

E.4.4 Preference Data Construction

To refine the advisor beyond strict supervised labels, we construct offline preference pairs over intervention modes. Each pair has the form (x, y^+, y^-), where y^+ is preferred over y^- for the same structured context x. The preference data is designed to inject intervention compatibility and external planning knowledge into the advisor.

We use several types of preference pairs:

- **Hard counterfactual pairs**: the reviewed label is preferred over incompatible alternatives.
- **SFT-error pairs**: when the supervised advisor predicts the wrong mode, the reviewed label is preferred over the SFT prediction.
- **Compatibility pairs**: for single-sided gold labels, `both` is preferred over `none`, because `both` preserves at least one useful intervention direction.
- **Both-anchor pairs**: for true `both` examples, `both` is preferred over `hist`, `future`, and `none`.
- **Format pairs**: valid JSON outputs are preferred over malformed outputs.

Preference optimization is not applied only to mistakes. Correctly classified examples are also retained as anchors to prevent the policy from shifting too aggressively toward rare modes. This is important because `both` is rare but semantically close to the single-sided labels.

E.4.5 STATIC REWARD FOR PAIR WEIGHTING

We use a static reward model to score candidate intervention modes when constructing and weighting preference pairs. This reward is not an online environment reward; it is an offline compatibility signal:

$$R(x, y) = 0.6\, R_{\text{struct}}(y, y^\star) + 0.2\, R_{\text{sem}}(x, y) + 0.2\, R_{\text{fmt}}(y).$$

The structural term R_{struct} measures compatibility between the predicted intervention set and the reviewed label. For example, `both` receives high compatibility with `hist` or `future` because it preserves the relevant intervention side. The semantic term R_{sem} checks whether the predicted mode is consistent with structured fields such as event type, trigger scope, actionability basis, polarity, magnitude, and time-to-event. The format term R_{fmt} rewards valid JSON output.

This reward formulation reflects the advisor's role: it should select intervention strategies that remain useful for downstream program synthesis, rather than merely reproducing strict labels.

E.4.6 OFFLINE PREFERENCE OPTIMIZATION

Starting from the supervised advisor, we apply reference-based offline preference optimization. Let π_θ be the trainable advisor policy and π_{ref} be the supervised reference model. For each preference pair (x, y^+, y^-), we define the reference-corrected preference margin:

$$m_\theta = \left[\log \pi_\theta(y^+|x) - \log \pi_\theta(y^-|x)\right] - \left[\log \pi_{\text{ref}}(y^+|x) - \log \pi_{\text{ref}}(y^-|x)\right].$$

We use a DPO-style preference lossRafailov et al. (2023):

$$\mathcal{L}_{\text{DPO}} = -\log \sigma(\beta m_\theta),$$

where β controls the sharpness of the preference margin.

To preserve the supervised decision boundary, we add a soft-label cross-entropy term:

$$\mathcal{L}_{\text{soft}} = -\sum_{c \in \mathcal{Y}} q(c \mid y^\star) \log \pi_\theta(c \mid x).$$

The target distribution $q(c \mid y^\star)$ assigns most mass to the reviewed label and smaller mass to compatible alternatives. For example, for a `future` gold label, the soft target assigns high mass to `future` and non-zero mass to `both`, reflecting that `both` still preserves Future Scenario Enforcement.

The final objective is:

$$\mathcal{L} = w_{\text{class}} w_{\text{hard}} w_{\text{role}} \left(\mathcal{L}_{\text{DPO}} + \lambda \mathcal{L}_{\text{soft}}\right),$$

where w_{class} compensates for label imbalance, w_{hard} emphasizes hard examples, w_{role} controls the contribution of different preference-pair types, and λ controls the strength of soft-label regularization. For compatibility pairs, we use a smaller role weight so that they guide the policy without dominating strict reviewed labels.

E.4.7 EVALUATION METRICS

We report four metrics for the Intervention Advisor Agent advisor. **Accuracy** and **Macro F1** use strict four-way labels over $\{\texttt{hist}, \texttt{future}, \texttt{both}, \texttt{none}\}$. Because the advisor is used as a soft planning prior rather than as the final executor, we also report compatibility-aware metrics. **Both-Tolerant Accuracy** and **Both-Tolerant Macro F1** count a prediction of `both` as correct whenever the reviewed label is `hist`, `future`, or `both`; all other predictions are scored as in the strict metrics, so predicting `both` for a `none` label remains an error. Under this definition, a policy that always outputs `both` is correct on every record except the `none` ones.

These both-tolerant metrics are important because the advisor's role is to preserve useful intervention directions for downstream planning. A strict mismatch is harmful when it removes the needed intervention side, such as predicting `none` for a future-event task. In contrast, predicting `both` for a single-sided case may still provide a useful prior, because the Planner Agent can reject unnecessary actions through tool contracts and verification.

Table 23: Intervention Advisor Agent advisor results. Accuracy and Macro F1 use strict four-way labels. Both-Tolerant metrics count a `both` prediction as correct when the reviewed label is `hist`, `future`, or `both` (Appendix E.4.7). Trivial constant policies are computed from the test label counts (65/9/4/3). Bold marks the best trained advisor per backbone; the trivial always-`both` policy exceeds all trained advisors on both both-tolerant columns.

Model	Stage	Accuracy ↑	Macro F1 ↑	Both-Tolerant Acc. ↑	Both-Tolerant Macro F1 ↑
Qwen3-8B	Base	0.4198	0.3155	0.4938	0.4127
Qwen3-8B	SFT	**0.8642**	**0.4194**	**0.8642**	0.4194
Qwen3-8B	Preference Opt.	0.6914	0.3845	0.8148	**0.5107**
Qwen3-0.6B	Base	0.0864	0.0521	0.7654	0.4242
Qwen3-0.6B	SFT	**0.8025**	0.3628	0.8025	0.3628
Qwen3-0.6B	Preference Opt.	0.6420	**0.3913**	**0.8889**	**0.5706**
Trivial: always `future`		0.8025	0.2226	0.8025	0.2226
Trivial: always `both`		0.0370	0.0179	0.9506	0.6500

E.4.8 Effect on Advisor Distribution

Table 23 summarizes the effect of supervised initialization and preference refinement. SFT substantially improves strict prediction accuracy, showing that the advisor learns the basic mapping from structured context to intervention modes. Preference optimization shifts the advisor distribution toward compatibility-aware behavior, especially preserving useful intervention directions through `both`-style predictions.

The results show a distinction between strict classification and compatibility-aware planning behavior. SFT achieves the strongest strict accuracy because it directly optimizes the reviewed labels. Preference optimization lowers strict accuracy but improves compatibility-aware behavior: Both-Tolerant Macro F1 improves from 0.4194 after SFT to 0.5107 for Qwen3-8B, and from 0.3628 to 0.5706 for Qwen3-0.6B. For Qwen3-0.6B, preference optimization also achieves the best Both-Tolerant Accuracy among the trained advisors.

These gains should be read with the trivial baselines in Table 23. Always predicting `future` already reaches 0.8025 strict accuracy, and always predicting `both` reaches 0.9506 both-tolerant accuracy, higher than any trained advisor. A higher both-tolerant score therefore does not by itself show better routing: it can be obtained by predicting `both` more often, which shifts the burden of rejecting the unnecessary side to the planner. Preference optimization thus trades strict accuracy for compatibility, and end-to-end evidence is limited: the preference-optimized advisor improves over no advisor with a Qwen3-8B planner (Figure 2(b)), but it has not been compared directly with the SFT advisor under the same protocol (Table 15 covers SFT only).

Calibration and selective routing. Classification accuracy alone does not establish that the Advisor's confidence is useful for routing. We therefore additionally report multiclass Brier score and adaptive expected calibration error (ECE) on the 81 held-out records. Supervised fine-tuning reduces the Brier score from 1.143 to 0.234 and adaptive ECE from 0.539 to 0.055. The SFT Advisor reaches 86.4% routing accuracy overall; accuracy rises to 91.8% on the most confident 75% of examples and 95.0% on the most confident 50%. This supports using confidence to abstain or defer uncertain routing decisions, although the small held-out set limits fine-grained calibration conclusions.

Table 24: Advisor calibration on the 81-record held-out set. Lower is better.

Advisor	Brier Score	Adaptive ECE
Base Advisor	1.143	0.539
SFT Advisor	**0.234**	**0.055**

E.5 Planner & Executor Agent

The **Planner & Executor Agent** is the central planning module for forecast-form tasks. Its role is to convert the structured information produced by upstream modules into an executable temporal

program:

$$\Pi = (\Pi_{\text{hist}}, \Pi_{\text{fm}}, \Pi_{\text{future}}),$$

where Π_{hist} specifies History Reconditioning, Π_{fm} specifies the numerical forecasting backbone, and Π_{future} specifies Future Scenario Enforcement.

The planner does not directly generate numerical forecasts. Instead, it decides how the time series should be processed before, during, and after numerical forecasting. This design keeps semantic decision-making separate from numerical prediction, and ensures that the final forecast is produced through inspectable operations rather than unconstrained natural-language reasoning.

E.5.1 PLANNER-FACING REPORT

Before invoking the planner, ECLIPSE constructs a compact planner-facing report. The report serves as the interface between context understanding and executable planning. It aggregates the original task information, context-derived control signals, retrieved temporal patterns, and available tool contracts into a single structured input.

The report is designed to answer three questions for the planner:

1. **What is the task?** The report summarizes the target variable, forecasting horizon, timestamps, and available covariates.
2. **What does the context imply?** The report includes extracted events, rules, constraints, commonsense profiles, and the scenario category from the Temporal Scenario Profiler.
3. **What actions are available?** The report lists the relevant operation families from the Tool Library, including historical-side and future-side interventions.

A simplified report schema is shown below.

Schema 3: Planner-Facing Report

```
{
  "task_summary": {
    "target": "<target variable>",
    "history_length": "<number of history steps>",
    "horizon": "<number of forecast steps>",
    "frequency": "<time granularity>",
    "available_covariates": ["<covariate keys>"]
  },

  "context_control": {
    "scenario_category": "<Temporal Scenario Profiler output>",
    "event_candidates": ["<event descriptions and time scopes>"],
    "rule_candidates": ["<rules or constraints>"],
    "commonsense_profile": "<domain-specific prior if available>"
  },

  "temporal_memory": {
    "retrieved_patterns": [
      {
        "pattern_id": "<reference pattern id>",
        "description": "<short pattern description>",
        "metadata": "<frequency, phase, or event metadata>"
      }
    ]
  },

  "intervention_prior": {
    "preferred_direction": "historical | future | both | none",
    "rationale": "<short planning hint>"
  },

  "tool_contracts": {
```

```
    "historical_operations": ["<allowed historical tools>"],
    "forecasting_backbones": ["<available forecasting models>"],
    "future_operations": ["<allowed future tools>"]
  }
}
```

This report intentionally separates raw context from context-derived control signals. The planner may inspect the original context when necessary, but the report provides a structured abstraction that highlights the parts most relevant to temporal execution. For example, a future outage is represented not only as text, but also as a candidate future event with a time scope; a solar-generation task may include a commonsense profile about daily periodicity; and retrieved reference patterns are exposed as explicit Temporal Archetype Bank evidence rather than hidden embeddings.

E.5.2 TOOL LIBRARY

The Tool Library defines the executable action space of ECLIPSE. It constrains the planner to registered temporal operations rather than arbitrary code, while still allowing bounded autonomy: the planner decides which tools to compose, which variables and time windows they affect, and how their parameters should be instantiated from context.

We organize the Tool Library into three groups aligned with the temporal program:

$$\mathcal{A} = \mathcal{A}_{\text{hist}} \cup \mathcal{A}_{\text{fm}} \cup \mathcal{A}_{\text{future}},$$

where $\mathcal{A}_{\text{hist}}$ contains History Reconditioning operations, $\mathcal{A}_{\text{fm}}$ contains forecasting backbones, and $\mathcal{A}_{\text{future}}$ contains Future Scenario Enforcement operations.

History Reconditioning. These tools modify the observed history before forecasting when the raw history is not the appropriate computational state for the current scenario. Representative operations include missing-value repair, outlier clipping, local history repair, event-artifact removal, fluctuation damping, level or slope shifting, and reference-based enrichment using the Temporal Archetype Bank.

Numerical Forecasting Backbone. This block invokes a dedicated time-series forecasting model on the reconditioned history. The planner may specify configuration such as horizon, frequency, sampling mode, and model name, but the numerical trajectory is produced by the forecasting backbone rather than by the LLM.

Future Scenario Enforcement. These tools modify the predicted future trajectory when the context specifies events, rules, or constraints that should hold in the forecast horizon. Representative operations include event-window shifts, set-value windows, non-negativity enforcement, bound enforcement, formula-rule application, tail anchoring, and noise-pattern transfer.

A compact view of the tool space is shown below.

Schema 4: Temporal Tool Library

```
{
  "historical_state_reconditioning": [
    "fill_missing_values",
    "clip_outliers",
    "repair_local_history_window",
    "remove_event_artifact",
    "dampen_fluctuations",
    "shift_level_or_slope",
    "enrich_history_with_reference"
  ],

  "numerical_forecasting_backbone": [
    "foundation_forecast"
  ],
```

```
  "future_scenario_enforcement": [
    "apply_event_shift",
    "set_value_window",
    "enforce_non_negative",
    "enforce_bounds",
    "apply_formula_rule",
    "apply_tail_anchor",
    "transfer_noise_pattern"
  ]
}
```

This design gives the planner flexibility without making execution unconstrained. The Planner & Executor Agent can choose tool combinations and instantiate arguments such as target variables, temporal scopes, reference patterns, formula parameters, or intervention magnitudes. However, every selected operation must belong to the Tool Library and all arguments are validated before execution. Thus, ECLIPSE preserves semantic flexibility while keeping numerical manipulation executable, auditable, and safe.

E.5.3 Program Output Protocol

The planner returns a structured program with three blocks. A simplified output schema is shown below.

Schema 5: Temporal Program Output

```
{
  "historical_state_reconditioning": {
    "use": true,
    "actions": [
      {
        "tool": "<historical operation name>",
        "arguments": { "<key>": "<value>" },
        "rationale": "<why this operation is needed>"
      }
    ]
  },

  "numerical_forecasting_backbone": {
    "tool": "foundation_forecast",
    "arguments": {
      "horizon": "<forecast horizon>",
      "frequency": "<time frequency>",
      "sampling": "<sampling mode>"
    }
  },

  "future_scenario_enforcement": {
    "use": true,
    "actions": [
      {
        "tool": "<future operation name>",
        "arguments": { "<key>": "<value>" },
        "rationale": "<why this operation is needed>"
      }
    ]
  }
}
```

The `rationale` fields are used for trace inspection rather than execution. Only the tool names and validated arguments are passed to the executor. This ensures that the program remains interpretable without allowing free-form explanations to affect numerical outputs.

E.5.4 PROMPT STRUCTURE

The planner prompt is organized around the same abstraction as the temporal program. Rather than asking the LLM to "predict the future," the prompt asks it to decide which operations should be applied to the temporal state.

Prompt 3: Planner & Executor Agent Prompt

```
You are a temporal program synthesizer for context-guided forecasting.

Your task is to produce an executable temporal program with three
    blocks:

1. historical_state_reconditioning:
   Decide whether the observed history should be modified before
    forecasting.
   Use this only when context indicates missing values, artifacts,
    abnormal
   historical events, insufficient history, or a need for reference-
    based
   enrichment.

2. numerical_forecasting_backbone:
   Select the forecasting backbone and required configuration.

3. future_scenario_enforcement:
   Decide whether the predicted future should be modified after
    forecasting.
   Use this when context specifies future events, explicit constraints,
   formula rules, or domain mechanisms that must hold in the forecast
    horizon.

Use the planner-facing report, Temporal Scenario Profiler output,
Temporal Archetype Bank records, and tool contracts.
Select only tools from the allowed Tool Library.
Return JSON only following the required program schema.
```

This prompt emphasizes the intended separation of roles: the LLM chooses an executable computation plan, while the forecasting backbone and deterministic tools carry out the numerical operations.

E.6 FEATURE REQUIREMENT ANALYSIS

ECLIPSE does not apply the same analysis procedure to every task instance. Instead, it first performs **Feature Requirement Analysis**, where the system determines which evidence, features, or downstream modules are required before execution. This design avoids both unnecessary invocation of the full planning pipeline for simple self-contained tasks and insufficient grounding for knowledge-augmented tasks.

At a high level, the execution process follows:

$$\text{Input} \rightarrow \text{Feature Requirements} \rightarrow \text{Execution Plan} \rightarrow \text{Feature Composition} \rightarrow \text{Final Resolution}.$$

This process keeps downstream reasoning grounded in structured temporal evidence rather than unconstrained prompt-level inference.

E.6.1 FEATURE ROUTING

The feature router determines which evidence families and execution modules are required for a given task. It considers whether the task requires historical statistics, contextual grounding, covariate

reasoning, future forecasting, or threshold resolution. The router then outputs a compact set of requirement flags used to construct the downstream execution plan.

Schema 6: Feature Requirement Flags

```
{
  "requires_history_analysis": true,
  "requires_context_grounding": true,
  "requires_covariate_reasoning": false,
  "requires_forecast": false,
  "requires_threshold_resolution": true
}
```

A key decision is whether `requires_forecast` should be enabled. For self-contained tasks, ECLIPSE may directly resolve the task using lightweight statistical extraction and structured evidence computation. For knowledge-augmented tasks that depend on future behavior or external contextual effects, the system invokes the Planner & Executor Agent to obtain forecast-mediated evidence.

E.6.2 Feature Planner

Given the requirement flags, ECLIPSE constructs a structured execution plan. The plan is not a free-form chain of thought, but a compact specification describing which evidence modules and operators should be executed.

Schema 7: Feature Execution Plan

```
{
  "evidence_families": [
    "generic_statistical_evidence",
    "operator_derived_structured_evidence",
    "forecast_mediated_future_evidence"
  ],
  "subgoals": [
    "extract_history_statistics",
    "select_evidence_operator",
    "estimate_future_trajectory",
    "compare_future_with_history",
    "resolve_answer_threshold"
  ],
  "output": "evidence_bundle"
}
```

This execution plan determines how evidence is collected and composed. Simple self-contained tasks may only require lightweight statistical features, whereas knowledge-augmented tasks may additionally require contextual grounding and forecast-mediated evidence.

E.6.3 Evidence Families

ECLIPSE organizes extracted evidence into three complementary families.

Generic Statistical Evidence. This family contains reusable low-level features extracted from the observed sequence, including level, trend, volatility, seasonality, peak timing, amplitude, and profile shape. These features are inexpensive and stable, making them suitable for many self-contained analytical tasks.

Operator-Derived Structured Evidence. Some tasks require evidence aligned with a particular temporal structure, such as conditioned comparisons, recovery analysis, transition comparisons, or secondary-peak detection. For these cases, ECLIPSE uses a **Temporal Evidence Operator Planner** to select an executable evidence operator. When predefined operators are insufficient, the system may generate a restricted custom-operator specification rather than unconstrained executable code.

Forecast-Mediated Future Evidence. Some tasks depend on future behavior and therefore cannot be resolved from historical observations alone. In such cases, ECLIPSE invokes the Planner & Executor Agent and converts the resulting future trajectory into structured comparison features, such as future-vs-history level changes, volatility ratios, amplitude changes, or peak timing shifts.

Schema 8: Forecast-Mediated Evidence Object

```
{
  "primary_key": "<target variable>",
  "future_series": ["<forecast values>"],
  "comparison_features": {
    "future_vs_history_level_change": "<value>",
    "future_vs_history_volatility_ratio": "<value>",
    "peak_timing_shift": "<value>",
    "profile_similarity": "<value>"
  }
}
```

This mechanism makes future-dependent reasoning explicit. Rather than forecasting for every task, ECLIPSE invokes future forecasting only when the required evidence cannot be resolved from the observed history and contextual information alone.

E.6.4 Feature Requirement Prompt

The feature requirement prompt determines which evidence families and execution modules should be activated.

Prompt 4: Feature Requirement Analysis Prompt

```
You are a feature planner for temporal reasoning tasks.

Given a task description, the available time-series history,
and the textual context, determine what evidence and computation
are required before producing the final output.

Determine whether the task requires:
1. historical statistics from the observed sequence;
2. contextual grounding over events, rules, or domain descriptions;
3. covariate reasoning;
4. future forecast evidence;
5. threshold or criterion resolution.

Do not solve the task directly.
Return a structured execution plan specifying which evidence families
should be collected and which downstream modules should be invoked.

Return JSON only with keys:
requires_history_analysis,
requires_context_grounding,
requires_covariate_reasoning,
requires_forecast,
requires_threshold_resolution,
evidence_families,
subgoals,
rationale.
```

E.6.5 OUTPUT PROTOCOL

Schema 9: Feature Requirement Output

```
{
  "requires_history_analysis": true,
  "requires_context_grounding": true,
  "requires_covariate_reasoning": false,
  "requires_forecast": true,
  "requires_threshold_resolution": true,
  "evidence_families": [
    "generic_statistical_evidence",
    "forecast_mediated_future_evidence"
  ],
  "subgoals": [
    "extract_history_statistics",
    "invoke_forecast_execution_chain",
    "compare_future_with_history",
    "resolve_answer_threshold"
  ],
  "rationale": "The task depends on future behavior and requires
  comparing the predicted future with historical statistics."
}
```

The resulting execution specification is stored as part of the execution trace and passed to the downstream evidence composition modules, making the reasoning process inspectable and traceable.

E.7 AUDIT AGENT

ECLIPSE uses an **Audit Agent** to validate both forecasting and analytical outputs. The auditor is not intended to solve the task again. Instead, it checks whether the generated temporal program, intermediate states, and final outputs remain consistent with the task context, executable tool contracts, and required output format.

This module serves two purposes. First, it prevents invalid intermediate decisions from silently propagating to final outputs. Second, it enables closed-loop reflection by determining whether the current execution should be accepted, revised, or terminated through fallback handling.

E.7.1 VERIFICATION TARGETS

The verifier checks different targets depending on the execution mode.

Forecast verification. For forecasting-oriented execution, the verifier examines the generated temporal program and the resulting forecast trajectory. The main checks include:

- **Format validity**: the output has the expected horizon, target dimension, and finite numerical values.
- **Tool-contract validity**: selected tools belong to the Tool Library and their arguments satisfy the expected schema.
- **Temporal-scope validity**: historical and future interventions are applied to valid time windows.
- **Context alignment**: History Reconditioning and Future Scenario Enforcement are supported by the textual context or retrieved Temporal Archetype Bank evidence.
- **Numerical plausibility**: the final trajectory satisfies basic domain constraints such as non-negativity or explicit bounds when applicable.

Analytical verification. For analytical execution, the verifier checks whether the final answer is supported by the constructed evidence bundle. The main checks include:

- **Answer-format validity**: the output follows the required answer format.

- **Evidence support**: the selected answer is consistent with historical, contextual, operator-derived, or forecast-mediated evidence.
- **Temporal consistency**: the answer does not contradict the relevant time windows or event ordering.
- **Commonsense consistency**: the answer does not violate obvious domain constraints extracted from context.

E.7.2 VERIFIER INPUT

The verifier receives a compact execution trace rather than only the final output. This allows it to audit not only *what* the system produced, but also *how* it was produced.

Schema 10: Verifier Input Trace

```
{
  "task_family": "forecast | analytical",
  "task_context": "<original or summarized context>",

  "forecast_trace": {
    "scenario_profile": "<Scenario Profiling output>",
    "retrieved_patterns": ["<Temporal Archetype Bank records>"],
    "temporal_program": "<generated execution program>",
    "reconditioned_history": "<summary statistics or reference>",
    "base_forecast": "<summary or trajectory>",
    "final_forecast": "<summary or trajectory>"
  },

  "analytical_trace": {
    "feature_requirements": "<Feature Requirement Analysis output>",
    "evidence_bundle": "<feature composition output>",
    "candidate_answer": "<answer before verification>"
  }
}
```

Only the fields relevant to the current execution mode are populated. For example, a pure forecasting task does not require an analytical evidence bundle, while an analytical task may include forecast traces when future evidence is required.

E.7.3 VERIFIER PROMPT

A simplified verifier prompt is shown below.

Prompt 5: Audit Agent Prompt

```
You are an execution consistency verifier for a context-guided
time-series agent.

Your role is not to solve the task again.
Your role is to check whether the produced output is consistent with:
1. the required output format;
2. the original task context;
3. the temporal program and tool contracts;
4. the available historical, contextual, and forecast evidence;
5. basic temporal and domain constraints.

For forecasting-oriented execution, check:
- output shape and numerical validity;
- whether interventions are supported by context;
- whether future constraints, rules, or event windows are respected;
- whether the trajectory is numerically plausible.

For analytical execution, check:
```

```
- answer format;
- whether the answer is supported by the evidence bundle;
- whether the answer respects temporal ordering and context;
- whether unresolved or weakly supported answers are flagged.

Return JSON only.
Do not introduce new assumptions beyond the provided trace.
```

E.7.4 VERIFIER OUTPUT PROTOCOL

The verifier returns a structured decision used by the closed-loop reflection mechanism.

Schema 11: Verifier Output Protocol

```
{
  "verdict": "accept | revise | fallback",
  "format_valid": true,
  "context_aligned": true,
  "temporally_consistent": true,
  "evidence_supported": true,
  "issues": [
    "<short description of detected issue>"
  ],
  "suggested_repair": "<optional high-level repair instruction>"
}
```

The `verdict` determines whether the current execution should be accepted, revised through iterative replanning, or terminated through fallback handling. The boolean fields provide a compact breakdown of the detected failure mode, while `issues` and `suggested_repair` are stored in the execution trace for debugging and error analysis.

E.7.5 CLOSED-LOOP REVISION AND FALLBACK

When verification fails, ECLIPSE performs bounded revision rather than unconstrained regeneration. If the verifier returns `revise`, the system appends the generated error trace to the execution context and triggers iterative replanning. For forecasting-oriented execution, revisions are restricted to executor-safe operations such as correcting invalid bounds, restoring valid temporal scopes, or removing unsupported interventions. For analytical execution, revisions are limited to evidence-consistent answer correction or unresolved-answer handling when support is insufficient.

The implementation has no global retry budget across the pipeline. For the post-adjustment step, an LLM-revised series that violates the output schema receives one strict-schema retry, so at most two requests are made; if the output is still invalid, the verifier returns `fallback`, and the system terminates with a fixed default. Infrastructure-level API retries are handled separately and are not included in these counts. For forecasting, all interventions are discarded and the output is the unedited foundation-model forecast on the raw history (i.e., Π_{hist} and Π_{future} become no-ops). For QA, the system returns an explicit unresolved-answer token, which is scored as incorrect. All revision and fallback decisions are recorded in the final execution trace.