PEARL: Self-Evolving Assistant for Time Management with RL

Anonymous ACL submission

Abstract

Overlapping calendar invitations force busy professionals to repeatedly decide which meetings to attend, reschedule, or decline. We refer to this preference-driven decision process as calendar conflict resolution. Automating such process is crucial yet challenging: Scheduling logistics drain hours, and human delegation often fails at scale, raising the question of whether large language model (LLM) or language agents can reliably learn and apply user preferences to manage time. To enable systematic study, we introduce CALCONFLICTBENCH, a benchmark for long-horizon calendar conflict resolution. Conflicts are presented sequentially and agents receive feedback after each round, requiring them to infer and adapt to user preferences progressively. Our experiments show that current LLM agents perform poorly with high error rates, e.g., Qwen-3-30B-Think with 35% average error rate. To address this gap, we propose **PEARL**, a reinforcement-learning framework that augments language agent with an external memory module and optimized round-wise reward design, enabling agent to progressively infer and adapt to user preferences on-the-fly. Experiments on CALCONFLICTBENCH shows that **PEARL** achieves 0.76 error reduction rate, and 55% improvement on average error rate compared to the strongest baseline.

1 Introduction

Overlapping meetings is a common problem in modern workplaces. Consider a CEO of a company or PI of a research lab: They need to coordinate a large amount of events with different stakeholders every day, but their daily working hours are limited. When multiple events conflicts with each other, they must decide which event to attend, which to postpone, and which to decline. We refer to this repeated, preference-driven decision problem as *calendar conflict resolution*.

Automating calendar conflict resolution is important because it quietly drains one's time and undermines productivity. Scheduling logistics associated with meetings, e.g., coordinating availability or rescheduling around last-minute conflicts, can easily amount up to hours each week; workplace statistic suggests that 43% of professionals spend at least three hours per week on scheduling meetings (Reclaim.ai, 2024; Calendly, 2024; Microsoft WorkLab, 2025). While in practice these decisions are often delegated to human assistants such as administrative staffs (U.S. Bureau of Labor Statistics, 2025), it can easily break down at scale. Not only do human assistants frequently confront high volume of tasks, but also coordinate multiple stakeholders' schedule in order to reliably resolve scheduling logistics. Furthermore, when a conflict occurs, human assistants have to rely on sparse, incomplete signals from past trajectories about what the delegator values to resolve the conflict. This leads to their internal preference model drifting over time, leading to judgments that are distant from the delegator's preference. This calls for a reliable agent that can resolve calendar conflicts. Concretely, a reliable calendar conflict resolution agent should: (i) model long-term individual preferences from past decisions, (ii) adapt when preferences evolve with new context and constraints, and (iii) resolve each conflict by explicitly grounding decisions in the inferred user priors.

The explosive growth of LLMs has enabled *language agents*. Their ability to perceive and reason over complex information show promise as intelligent assistants that automate real-world tasks across different domains such as software development, chart generation, and travel planning (Wang et al., 2024; Li et al., 2025; Qian et al., 2025). Yet it remains unclear whether their performance is *trustworthy* for *calendar conflict resolution*, where small mistakes compound and mis-modeled preferences directly translate into costly time allocation

errors. This motivates a central question:

Can we trust LLMs to manage time?

To enable a systematic investigation of this problem, we introduce CALCONFLICTBENCH, a benchmark for evaluating language agents on calendar event conflict resolution. CALCONFLICTBENCH features synthetic users with diverse organizational roles and year-long calendars populated with carefully designed conflict scenarios. Conflict events are presented sequentially over time, and the agent receives feedback after each decision. This interactive setup closely mirrors real-world calendar management, where agents must infer and adapt to user preferences progressively through repeated interaction, rather than relying on fixed or one-shot instructions. Our empirical results show that current LLMs struggle on this task with high error rates. These failures reveal a fundamental limitation: **LLM agents have *weak* ability to infer, retain, and refine preference-driven decision principles over long horizons.**

To address this gap, we propose **PEARL** (**P**reference **E**volving **A**gent with **R**einforcement **L**earning), a reinforcement learning framework that trains language agents to *infer* user preferences online and *apply* them consistently over long-horizon calendar conflicts. **PEARL** introduces a structured rollout with a persistent external memory, the *Strategy Hub*, which stores a set of interpretable decision strategies (preference states) and is iteratively retrieved and updated at each round to capture newly revealed user priorities. To make preference learning explicit and stable, we optimize the agent with a curriculum-based reward , gradually shifting emphasis from preference inference in early rounds to preference-consistent decision making in later rounds. Experiments shows that **PEARL** achieves 0.76 error reduction rate CALCONFLICTBENCH, and 55% improvement on average error rate compared to the strongest baseline.

In summary, our main contributions are:

- **Task.** We formulate *calendar conflict resolution* as a new challenging task for LLMs agents, requiring preference-sensitive decision-making for conflict events over long horizons.
- **Benchmark.** We construct CALCONFLICTBENCH, a evaluation suite with synthetic data generation engine and standardized evaluation protocols to systematically evaluate LLM agents on calendar conflict resolution, and we provide an in-depth analysis of their failure modes.

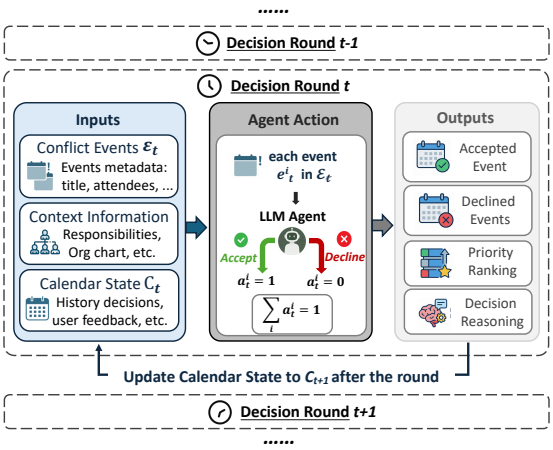

Figure 1: **Illustration of the *proposed* calendar conflict resolution task.** At decision round t, the agent observes (i) the conflicting events $\mathcal{E}_t$, (ii) contextual information, and (iii) the current calendar state $\mathcal{C}_t$. The agent selects exactly one event to accept ($a^i_t = 1$) and declines the rest ($a^i_t = 0$), producing the accepted event, declined events, a priority ranking, and rationale.

- **Method.** We propose **PEARL** (§5), a reinforcement learning framework that enables agents to progressively infer and adapt to user preferences on-the-fly with explicit memory module and carefully designed round-wise rewards, improving average error rate by 55% over the strongest baseline on CALCONFLICTBENCH.

2 Task Formulation

In this section, we formally define the proposed *calendar conflict resolution* task. Appendix C.4 illustrated an example data point.

Task Objective. The task is modeled as a sequential decision process with state transitions. As illustrated in Figure 1, the goal of *calendar conflict resolution* is to construct a valid calendar for a single user by resolving a sequence of event conflicts over time. At each step t, the agent is presented with current calendar state $\mathcal{C}_t$, and a set of temporally overlapping events $\mathcal{E}_t = \{e^1_t, \ldots, e^{N_t}_t\}$ and must accept exactly one event $e^i_t \in \mathcal{E}_t$, rejecting all others. The objective is to progressively model user preferences through interaction and contextual signals, producing a final calendar state $\mathcal{C}_T$ that aligns with the user's preferences and decision context.

Agent Action Space. At step t, the agent is tasked to assign a binary decision $a^i_t \in \{0, 1\}$ to each event $e^i_t \in \mathcal{E}_t$, where $a^i_t = 1$ denotes acceptance and $a^i_t = 0$ denotes rejection. The action must satisfy the constraint $\sum_i a^i_t = 1$.

Environment Observation Space. The observation space is designed to reflect real-world calendar

usage. At each step t, the agent observes contextual information (e.g. organization chart), the current calendar state $\mathcal{C}_t$, and the set of conflicting events $\mathcal{E}_t$. Each event $e_t^i \in \mathcal{E}_t$ is represented by structured metadata, including temporal attributes (e.g., start and end times), participant information, event descriptions (e.g. meeting topic or event summarization). The calendar state $\mathcal{C}_t$ summarizes previous calendar events and user decisions.

3 CalConflictBench

We introduce CALCONFLICTBENCH to support the evaluation of proposed task. In the benchmark, we present a synthetic data engine (Section 3.1) for generating realistic, role-specific calendars and a comprehensive evaluation protocol (Section 3.2).

3.1 Synthetic Data Engine

We construct the synthetic data engine to generate data for training and evaluation. We report the detail of data engine design in Appendix C, and we summarize key steps as following.

Organizational Schema Curation. We begin by crafting organizational schemas that capture real-world structures (e.g., research laboratories and technology companies). We conduct interviews with domain practitioners and analyze the collected real-world calendar data and organizational charts to extract role-specific information for each position (e.g. PI, postdoc, PhD student; CEO, SWE, HR). For each role, we curate schemas based on the extracted information, including: (1) regular meeting schemas, such as typical topics, frequencies, and attendees; (2) priority principles P that govern decision-making (e.g., leadership duties, deadline sensitivity, people management); and (3) common conflict reasons C (e.g., deadline clashes, hierarchical obligations, external commitments). These priority principles are not directly observable by the agent. We further perform human verification on all schema to ensure reliability.

Step 1: Synthetic Organization and User Profile Generation. Given an organizational schema, we instantiate user profiles for each role within the organization. Each user is associated with a fixed role, a regular meeting pattern, and a priority principle set. This step defines the ground-truth preference structure that governs all downstream calendar decisions.

Step 2: Regular Event Generation. For each user, we generate a year-long calendar consisting of regular events using python scripts. Events are sampled according to role-specific meeting schemas, resulting in 52 weeks of weekly schedules. At this stage, calendars contain no conflicts and reflect the user's normal workload and responsibilities.

Step 3: Conflict Event Generation. We then carefully and systematically inject conflict events by overlapping regular events within the same time window. Given the user's priority principles, conflict reasons, and predefined accept/decline ratios, we generate conflicting event sets together with a unique ground-truth resolution. These conflicts vary in difficulty, ranging from single-factor trade-offs to multi-factor conflicts that require balancing urgency, interpersonal relationships, and values.

Step 4: Human Annotator Verification. In the last step, we perform human verification to ensure the validity of the synthetic data and filter out implausible or inconsistent cases.

3.2 Evaluation Protocol

Our evaluation is designed to assess the *preference-evolving capability* of LLM agent, which is whether the agent can infer decision-making principles of users over time. Note that the evaluation designed as **single-turn** format, and each instance contains history context (past rounds information).

Parameters. We define three evaluation parameters: (i) the total number of decision rounds N, (ii) the context window size W, which specifies how many past rounds of information are provided to the agent, and (iii) the total number of events are conflicting with each other per round M.

Procedure. Each evaluation instance (one trajectory) simulates one year of calendar usage for a single synthetic user. Calendar conflicts are presented sequentially over time, mimicking realistic calendar dynamics. The agent does not have access to the ground-truth priority principles and must infer them solely from history and contextual information. The agent may update its internal beliefs or strategies across rounds, and performance is evaluated over the full trajectory of N rounds to capture long-horizon adaptation.

Per-Round Metrics. We design the following metrics to evaluate decision quality at each round:

- ***Decision Accuracy***. A binary indicator of whether the agent's accepted event matches the ground-truth accepted event. Note that invalid outputs are counted as incorrect.
- ***Optimal Rank Distance (ORD)***. For rounds with $M \geq 3$, we ask the agent to produce a rank-

	Average Error Rate of N rounds					Optimal Rank Distance of N rounds					Error Reduction Rate
	1	25	50	75	104	1	25	50	75	104	
Base Models											
Qwen3-4B	0.44	0.46	0.44	0.45	0.45	0.73	0.73	0.75	0.75	0.76	-0.029
Qwen3-8B	**0.30**	0.38	0.36	0.37	0.37	0.76	0.78	0.79	0.79	0.79	0.026
Qwen3-14B	0.38	0.42	0.41	0.40	0.41	0.82	0.75	0.75	0.74	0.75	-0.039
Qwen3-30B	0.34	0.39	0.39	0.39	0.38	0.79	0.79	0.79	0.78	0.78	0.069
Qwen3-30B-Think	0.36	0.38	**0.34**	**0.36**	**0.35**	0.80	0.79	0.81	0.81	0.82	**0.161**
LLaMA-3.1-8B	0.66	0.66	0.67	0.65	0.65	0.58	0.58	0.60	0.61	0.62	-0.027
OLMo3-7B-Instruct	0.98	1.00	1.00	1.00	1.00	0.01	0.00	0.00	0.00	0.00	-0.004
OLMo3-32B-Think	0.40	0.45	0.46	0.46	0.45	0.72	0.72	0.72	0.72	0.72	0.050
GPT-5-nano	**0.30**	0.42	0.41	0.43	0.41	**0.85**	0.77	0.78	0.77	0.78	0.122
GPT-5	0.42	0.39	0.36	**0.36**	**0.35**	0.83	**0.81**	**0.82**	**0.82**	**0.83**	0.092
Gemini-2.5-flash	**0.30**	0.40	0.39	0.40	0.38	0.84	0.79	0.79	0.79	0.81	0.088
Agentic Rollouts											
ReAct	0.34	0.40	0.39	0.39	0.39	0.78	0.78	0.79	0.79	0.80	0.007
Mem+ReAct	0.36	**0.37**	0.39	0.39	0.40	0.84	**0.81**	0.81	0.80	0.79	-0.162

Table 1: **Performance across different numbers of rounds N.** All results are evaluated with context window size $W = 20$ and $M = 5$ conflicting events per round. Results are averaged over ten independent instances. For each N, the best performance is shown in **bold**, and the second-best is underlined.

ing ρ_t over over the $M = |\mathcal{E}_t|$ candidate events. Let e_t^* be the ground-truth accepted event with 0-indexed position $\mathrm{pos}_t(e_t^*; \rho_t) \in \{0, \ldots, M\}$. We define the Optimal Rank Distance (ORD) as

$$ORD = 1 - \frac{\mathrm{pos}_t(e_t^*; \rho_t)}{M - 1}, \quad ORD \in [0, 1].$$

Per-Instance Metrics. To measure preference learning and adaptation over time, we define three instance-level metrics:

- ***Average Error Rate of N rounds***. The mean decision error across all N rounds in a trajectory, capturing overall long-horizon performance.
- ***Average ORD of N rounds***. The average ORD across all N rounds in a trajectory, measuring how close the predicted event priority is to the optimal ranking.
- ***Error Reduction Rate***. The relative decrease in average error rate in the first quarter of the instance to average error rate in the last quarter of the same instance, measuring the agent's ability to learn and improve its decisions over time.

4 Evaluation

4.1 Setup.

We follow protocol described in Section 3.2. We vary $M \in \{2, 3, 4, 5\}$ and $W \in \{1, 5, 10, 20\}$ to control the combinatorial difficulty and historical context available at each decision round. More details are reported in Appendix D.

Data. We evaluate agents on full-year calendars (52 weeks) constructed for ten synthetic users drawn from two synthetic organizations. To manage computational cost, we uniformly sample one decision round per week. Each evaluation trajectory therefore consists of 104 decisions (i.e. conflict events series), resulting in 1,040 total decisions.

Models. We evaluate a diverse set of strong LLMs as agent base models, spanning open-source, reasoning-oriented, and proprietary families. Our open-source models include Qwen3-8B/14B/30B/30B-Think (Yang et al., 2025), OLMo3-7B/OLMo3-32B-Think (Olmo et al., 2025), and LLaMA-3.1-8B (Grattafiori et al., 2024). We also include GPT5-nano, GPT5 (OpenAI, 2025), and Gemini-2.5-Flash (Comanici et al., 2025) for proprietary model families. On top of these base models, we further evaluate representative agentic rollout style prompting, including ReAct (Yao et al., 2023) and Memory-Augmented ReAct (Zhu et al., 2025).

4.2 Results and Analysis

Table 1 presents the evaluation results across different numbers of decision rounds N. We summarize key insights as follows.

Insight 1. **Current LLMs do not exhibit Preference-Evolving capability.** As indicated by the *Error Reduction Rate* in Table 1, no evaluated

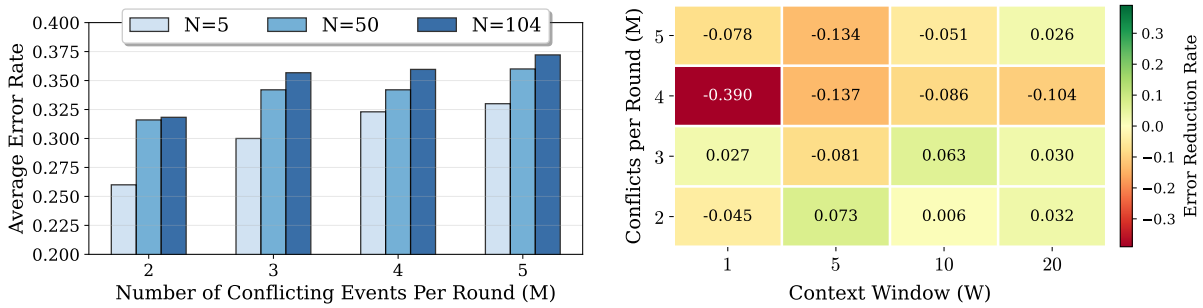

Figure 2: Average Error Rate of Qwen3-8b under different the number of conflicting events per round (M) (left), and Error Reduction Rate of Qwen3-8B under different evaluation parameters (right).

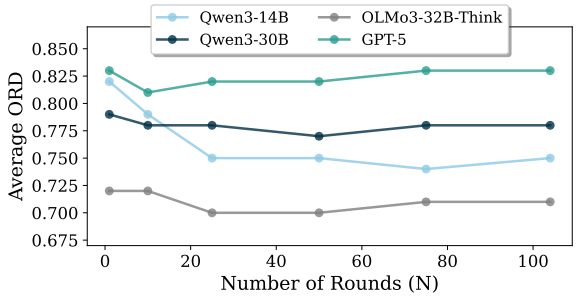

Figure 3: Average Optimal Rank Distance (ORD) over different numbers of decision rounds (N).

LLM shows consistent performance improvement when transitioning from single-round ($N = 1$) to multi-round settings. Error reduction rates are near zero or negative across models, including GPT-5 and Gemini-2.5-flash, suggesting that additional interaction rounds do not help refine decision principles. Figure 3 corroborates this finding, with error rates remaining flat or increasing as N grows.

***Insight 2*. Increasing local decision complexity degrades performance.** As shown in Figure 2 (left), the average error rate increases monotonically as the number of conflicting events per round M grows. This trend reflects a rapid escalation in local decision complexity caused by higher event overlap, which expands the combinatorial decision space and increases ambiguity among candidate choices. Notably, this degradation is also observed in the single-round setting, indicating that errors arise primarily from local reasoning difficulty rather than long-horizon dependencies. As M increases, these local errors accumulate across rounds, leading to compounded performance degradation in multi-round scenarios.

***Insight 3*. Larger context windows do not enable long-horizon reasoning.** As shown in Figure 2 (right), increasing the context window size W yields marginal and inconsistent changes in error reduction rate, with no clear monotonic improvement. In some cases, larger context windows even degrade performance, suggesting that additional context length does not translate into better preference-aligned decisions, and it is insufficient for preference-evolving behavior.

5 PEARL

We propose **PEARL**, a reinforcement learning framework for long-horizon, preference-evolving language agents. In this section, we introduce our rollout design (Section 5.1), reward modeling (Section 5.2), and the experiment results for **PEARL** evaluation (Section 5.3).

5.1 Rollout Design for Preference Inference

We design a rollout mechanism that centers decision-making on a persistent, compact preference representation, enabling incremental inference and reuse across rounds.

Strategy Hub. Long-horizon preference learning via pure in-context history is challenging: As interactions grow, agents must repeatedly rediscover the same preference cues from a lengthy, noisy transcript, and the resulting preference state remains implicit and hard to reuse or update. To address this, we introduce the *Strategy Hub* ($\mathcal{S}$) as an external memory module that maintains a *fixed-size* set of decision strategies. Each strategy encodes an user *preference state* in natural language (See Appendix E.1 for details). The design of $\mathcal{S}$ explicitly separates *preference inference*—identifying which strategy types matter and assigning their weights—from *preference execution*—applying these learned priorities to new conflict contexts. This decomposition compresses preference learning into a compact and interpretable state that can be persistently updated across rounds, avoiding brittle reliance on implicit long-context representations.

At each decision round, the agent observes the current context (i.e. previous decisions and contex-

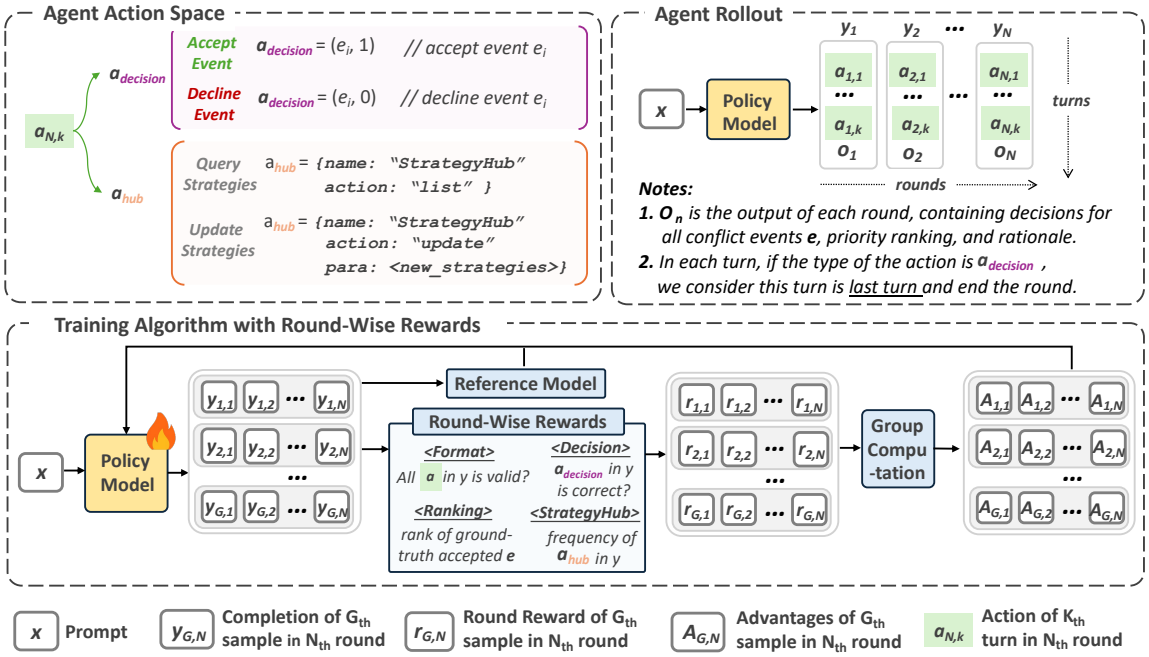

Figure 4: **Overview of PEARL. Top-left: Agent action space.** At each turn, the agent can take a *decision action* a_{decision} (accept/decline an event e_i) or a *hub action* a_{hub} that queries (`list`) or updates (`update`) the external *Strategy Hub*. **Top-right: Agent rollout.** The policy model generates a multi-turn trajectory; when a decision action is emitted, the round terminates and the next conflict is presented. **Bottom: Training with round-wise rewarding.** For each round, we sample multiple completions, score them with the curriculum-based reward model, and aggregate rewards into group-wise advantages by each round to update the policy.

tual information), and a set of conflicting events, and is granted access to $\mathcal{S}$, which is initialized as empty at the initial round. As shown in Algorithm 1, the agent interacts with the $\mathcal{S}$ for a bounded number of turns (k), to retrieve and update strategies as needed.

Agent Structured Rollout. As illustrated in Figure 4, the agent may take up to K turns within each round. At turn k, it emits an action $a_{t,k} \in \mathcal{A} = \mathcal{A}_{\text{hub}} \cup \mathcal{A}_{\text{decision}}$, where $\mathcal{A}_{\text{decision}}$ contains accept/decline decisions for events, and $\mathcal{A}_{\text{hub}}$ contains interactions with $\mathcal{S}$ (e.g., `list` current strategies or `update` current strategies). We denote the round output as $O_t = (d_t, \rho_t, \xi_t)$, where d_t is the accept/decline decision set over events in $\mathcal{E}_t$ (typically accepting exactly one and declining the rest), ρ_t is the priority ranking over $\mathcal{E}_t$, and ξ_t is the rationale. The round terminates at the first turn k_t such that $a_{t,k_t} \in \mathcal{A}_{\text{decision}}$. The rollout can be written as a sequence of round outputs

$$y = (O_1, \ldots, O_N), \qquad \tau(x, y) = \{(o_t, O_t)\}_{t=1}^{N},$$

Equivalently, the trajectory can also be represented by the per-turn action trace $\{a_{t,k}\}_{t=1..N,\, k=1..k_t}$, where $k_t \le K$ is the stopping turn when the decision action is emitted.

Algorithm 1: Agent Rollout Procedure

Input: StrategyHub $\mathcal{S}_0$; rounds $t = 1..N$; context $\mathcal{C}_t$; conflicts $\mathcal{E}_t$; max turns K
Output: $y = (O_1, \ldots, O_N)$, where $O_t = (d_t, \rho_t, \xi_t)$

```
S ← S_0; H_<1 ← ∅;
for t ← 1 to N do
    u_t ← 0; O_t ← ⊥;
    H_<t ← {C*_τ}_{τ<t};                    // history
    for k ← 1 to K do
        a_{t,k} ~ π_θ(· | C_t, H_<t, E_t, S);
        if a_{t,k} ∈ A_hub then
            if a_{t,k} = list then
                LIST(S);
            else if a_{t,k} = update(Δ) then
                S ← UPDATE(S, Δ);
            u_t ← 1;
        else if a_{t,k} ∈ A_decision then
            Parse a_{t,k} into (d_t, ρ_t, ξ_t);
            O_t ← (d_t, ρ_t, ξ_t); break;
return y
```

5.2 Reward Modeling for Preference-Evolving

To train agents that both *infer* user preferences and *act* on them over long horizons, we design a curriculum-based reward model that encourages *preference evolution* across rounds.

Round-Level Rewards. We assign rewards only

at the round level. Each round t consists of up to K turns and terminates when the agent commits to a decision action or reaches the maximum number of turns K. At each round t, we design four reward signals that target complementary aspects at different granularities:

- ***Format Reward.*** To prevent catastrophic "invalid action" failures that break environment execution and learning, we reward outputs that are syntactically valid (i.e., parseable and in the allowed action space): $r_t^{\mathrm{f}}(x, y) = \mathbb{I}[a_t \in \mathcal{A}_{\text{valid}}]$.
- ***Decision Reward.*** To directly optimize preference aligned correctness, we reward agent make correct decision: $r_t^{\mathrm{a}}(x, y) = \mathbb{I}[a_t = a_t^*]$, where a_t^* denote the ground-truth round decision (accept / decline for events in $\mathcal{E}_t$).
- ***Ranking Reward.*** To alleviate sparsity in r_t^{a}, we add a denser signal based on the predicted priority ranking. We reward placing the ground-truth accepted event e_t^* closer to the top of the agent-produced ranking ρ_t over the $M = |\mathcal{E}_t|$ candidate events: $r_t^{\mathrm{r}}(x, y) = 1 - \frac{\mathrm{pos}_t(e_t^*; \rho_t)}{M-1}$.
- ***Strategy Hub Interaction Reward.*** To encourage deliberate preference retrieval/refinement rather than purely reactive decisions, we reward rounds where the agent performs a valid StrategyHub interaction ($u_t \in \{0, 1\}$): $r_t^{\mathrm{s}}(x, y) = u_t$.

Trajectory-Level Curriculum. In long-horizon calendar decisions, the agent faces a *cold-start* problem: In early rounds, user preferences are poorly identified, so directly optimizing action correctness can be high-variance and brittle, while the most useful behavior is to *extract and consolidate* preference evidence into persistent memory (S). As interaction progresses, the preference state becomes more stable; at that point, the learning signal should shift toward *preference-consistent execution*, where fine-grained prioritization among many candidates matters. To encourage this staged learning, we treat the format reward and decision reward weights, λ^{f} and λ^{a}, as fixed hyperparameters, and schedule the ranking reward and strategy hub interaction reward, λ^{r} and λ^{i} weights, as a function of the round index. We define the normalized round index: $i_t = \frac{t}{N} \in [0, 1]$. Then, we set round-dependent weights by linear interpolation:

$$\lambda_t^{\mathrm{r}} = 0.5 * i_t, \qquad \lambda_t^{\mathrm{s}} = 0.5 * (1 - i_t).$$

The shaped per-round reward as

$$\tilde{r}_t(x, y) = \lambda^{\mathrm{f}}\, r_t^{\mathrm{f}} + \lambda^{\mathrm{a}}\, r_t^{\mathrm{a}} + \lambda_t^{\mathrm{r}}\, r_t^{\mathrm{r}} + \lambda_t^{\mathrm{i}}\, r_t^{\mathrm{i}}$$

and the trajectory return is computed as

$$R(x, y) = \sum_{t=1}^{N} \gamma^{t-1}\, \tilde{r}_t(x, y).$$

Round-Wise Advantage Estimation. The trajectory contains N decision rounds, and the curriculum makes the reward distribution *non-stationary across rounds*. If we normalize advantages using a single trajectory-level baseline, (i) later rounds can dominate the learning signal due to larger/more direct rewards, and (ii) early-round updates become noisy because their returns are intrinsically more uncertain (preferences are not yet identified). To stabilize training and improve credit assignment, we further group the roll-outs based on the round position, and compute advantages *separately for each round position*. Let $\tilde{r}_{t,i}$ be the shaped reward of rollout y_i at round t. We compute a round-position return-to-go:

$$G_{t,i}(x) = \sum_{\tau=t}^{N} \gamma^{\tau-t}\, \tilde{r}_{\tau,i}(x, y_i).$$

For each round position t, we normalize these returns across the group:

$$\mu_t(x) = \frac{1}{G} \sum_{i=1}^{G} G_{t,i}(x),$$

$$\sigma_t(x) = \sqrt{\frac{1}{G} \sum_{i=1}^{G} \big(G_{t,i}(x) - \mu_t(x)\big)^2 + \varepsilon}.$$

Then the round-wise advantages are

$$\hat{A}_{t,i}(x, y_i) = \frac{G_{t,i}(x) - \mu_t(x)}{\sigma_t(x)}.$$

Objective. We train the policy with the standard clipped GRPO objective, adapted with our computed round-wise advantages $\hat{A}_{t,i}(x, y_i)$.

5.3 Experiment

Setup. We adopt Qwen3-4B as the base language model. We compare **PEARL** against three baselines under the same evaluation protocol as Section 3: (i) **Zero-shot**, which directly prompts the base model to resolve conflicts; (ii) **Zero-shot + *StrategyHub***, which augments the prompt with access to the external Strategy Hub but without parameter updates; and (iii) **SFT**, which performs supervised fine-tuning on training data. Unless otherwise specified, all methods operate on the same observed context and interaction history at each

round, and are evaluated over the same set of evaluation data as Section 4. All training details are provided in Appendix E.4.

Results and Analysis. Figure 5 reveals a clear separation in *adaptation dynamics*. The zero-shot baseline stays nearly flat around a high error band across rounds, indicating that simply conditioning on growing history does not reliably improve preference alignment and can even slightly drift (negative ERR in Table 2). In contrast, **PEARL** exhibits a *monotonic* reduction in error as number of rounds N increases, suggesting that it is not merely exploiting longer context, but is learning to *update* its decision policy across decision rounds.

Table 2 further disentangles the sources of gains. Providing the memory module access alone (Zero-shot + StrategyHub) yields only modest improvement (***AER.*** decreases from 0.45 to 0.41; ***ERR.*** increases from -0.029 to 0.048), implying that *having* an external memory without learning is insufficient for robust preference-evolving. Supervised training (SFT) improves final-round accuracy (with ***AER.*** of 0.27) but still lags behind **PEARL** (with ***AER.*** of 0.12) and achieves substantially weaker adaptation (***ERR.*** 0.325 vs. 0.761). This gap suggests that imitation-style training learns better *static* decision heuristics, yet struggles with long-horizon credit assignment and compounding preference-dependent errors across decision rounds over long horizon. Notably, **PEARL** achieves 55% improvement on ***AER.*** compared to the strongest baseline.

Overall, these results highlight that preference-evolving behavior requires *long-horizon optimization* over multi-round trajectories: **PEARL** can translates the history of previous rounds into measurable error reduction, validating the necessity of reinforcement learning for preference adaptation rather than one-shot prompting or purely SFT.

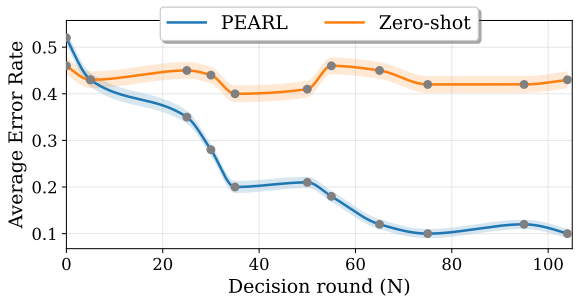

Figure 5: **Error vs. decision rounds** of **PEARL** and zero-shot baseline

Method	***AER.*** (N=104)	***ERR.***
Zero-shot	0.45	-0.029
SFT	0.27	0.325
Zero-shot + ***StrategyHub***	0.41	0.048
PEARL	**0.12** (↓)	0.761 (↑)

Table 2: **Final-round performance and adaptation.** Average Error Rate(***AER.***) at the last decision round and Error Reduction Rate (***ERR.***) across methods.

6 Related Works

LLM-based agents have been developed as intelligent assistants for tool-augmented QA, web browsing, and task-oriented dialog (Wang et al., 2024; Li et al., 2025; Qian et al., 2025), with frameworks such as ReAct and AutoGPT enabling autonomous behavior by interleaving reasoning and tool use (Yao et al., 2023; Yang et al., 2023). Yet personal time management remains less explored: earlier systems (e.g., Calendar.help) depended on predefined workflows with human-in-the-loop execution (Cranshaw et al., 2017), recent studies begin to investigate LLM-based scheduling agents (Shen et al., 2024; Wijerathne et al., 2025). Our work extends this line to long-horizon calendar conflict resolution where agents must adapt to user-specific preferences over many decisions. Preference alignment is commonly achieved via RLHF, which fine-tunes models using human feedback (Ziegler et al., 2019; Stiennon et al., 2020; Ouyang et al., 2022); to reduce labeling cost, methods leverage AI-generated principles (e.g., Constitutional AI) (Bai et al., 2022), and self-evaluation/self-correction (Wu et al., 2025). Distinctly, we target *sequential* preference alignment under long horizons. Since long-horizon learning is hindered by limited context and state retention, prior work explores curriculum learning (Narvekar et al., 2020) and external memory/state tracking (Yan et al., 2025); we design external memory module to accumulate past decisions for preference inference and reuse across rounds.

7 Conclusion

In this work, we study calendar conflict resolution, a long-horizon, preference-driven decision-making task. We introduce CALCONFLICTBENCH for systemic investigation, and evaluation results shows that current LLM agents degrade as horizons grow and conflicts become denser. To address this, We propose **PEARL**, a RL framework with explicit memory module and round-wise rewards, achieving strong gains on CALCONFLICTBENCH.

Limitations

Our study is an initial step toward systematically evaluating and training preference-evolving agents for calendar conflict resolution, and it leaves several limitations for future work. First, CALCONFLICTBENCH represents user preferences via structured, role-conditioned rules over event attributes, which makes evaluation reproducible but inevitably incomplete. In real-world settings, decisions can be driven by transient and hard-to-observe factors that are not reflected in calendar metadata—e.g., "I'm not in the mood for meetings today," fatigue, stress, interpersonal dynamics, or unexpected urgent tasks. Such affective and situational signals are difficult to simulate faithfully and may only be expressed through natural language messages or behavioral cues. Consequently, agents that perform well in our benchmark may still fail under implicit, rapidly shifting drivers of user choices. Second, while we conduct all the necessary experiments to support our main claims, computational and time constraints prevent an exhaustive sweep over all possible combinations of evaluation parameters. Third, because current LLMs have limited context windows, we only evaluate histories of up to 20 past events. We leave designing principled mechanisms for dynamically selecting and summarizing relevant context over long horizons as future work.

References

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, and 1 others. 2022. Constitutional AI: Harmlessness from AI feedback. *arXiv preprint arXiv:2212.08073*.

Calendly. 2024. What is automated scheduling? https://calendly.com/blog/automated-scheduling. Accessed: 2025-12-29.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, and 1 others. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*.

Justin Cranshaw, Emad Elwany, Todd Newman, Rafal Kocielnik, Bowen Yu, Sandeep Soni, Jaime Teevan, and Andrés Monroy-Hernández. 2017. Calendar.help: Designing a workflow-based scheduling agent with humans in the loop. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI)*, pages 2382–2393.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Bingxuan Li, Yiwei Wang, Jiuxiang Gu, Kai-Wei Chang, and Nanyun Peng. 2025. METAL: A multi-agent framework for chart generation with test-time scaling. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 30054–30069, Vienna, Austria. Association for Computational Linguistics.

Microsoft WorkLab. 2025. Breaking down the infinite workday. https://www.microsoft.com/en-us/worklab/work-trend-index/breaking-down-infinite-workday. Accessed: 2025-12-29.

Sanmit Narvekar, Bo Peng, Matteo Leonetti, Jivko Sinapov, Matthew E. Taylor, and Peter Stone. 2020. Curriculum learning for reinforcement learning domains: A framework and survey. *Journal of Machine Learning Research*, 21(181):1–50.

Team Olmo, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, and 1 others. 2025. Olmo 3. *arXiv preprint arXiv:2512.13961*.

OpenAI. 2025. Gpt-5 system card. https://cdn.openai.com/gpt-5-system-card.pdf. Accessed: 2026-01-03.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems 35 (NeurIPS)*.

Cheng Qian, Zuxin Liu, Akshara Prabhakar, Jielin Qiu, Zhiwei Liu, Haolin Chen, Shirley Kokane, Heng Ji, Weiran Yao, Shelby Heinecke, and 1 others. 2025. Userrl: Training interactive user-centric agent via reinforcement learning. *arXiv preprint arXiv:2509.19736*.

Reclaim.ai. 2024. Smart meetings trends report (145+ stats). https://reclaim.ai/blog/smart-meetings-report. Accessed: 2025-12-29.

Yuanhao Shen, Xiaodan Zhu, and Lei Chen. 2024. SMARTCAL: An approach to self-aware tool-use evaluation and calibration in LLMs. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (Industry Track)*.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel M. Ziegler, Ryan J. Lowe, Caleb Barnes, Alec Radford,

Dario Amodei, and Paul Christiano. 2020. Learning to summarize with human feedback. In *Advances in Neural Information Processing Systems 33 (NeurIPS)*, pages 3008–3021.

Sijun Tan, Michael Luo, Colin Cai, Tarun Venkat, Kyle Montgomery, Aaron Hao, Tianhao Wu, Arnav Balyan, Manan Roongta, Chenguang Wang, Li Erran Li, Raluca Ada Popa, and Ion Stoica. 2025. rllm: A framework for post-training language agents. Notion Blog.

U.S. Bureau of Labor Statistics. 2025. Secretaries and administrative assistants. https://www.bls.gov/ooh/office-and-administrative-support/secretaries-and-administrative-assistants.htm. Accessed: 2025-12-29.

Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. 2024. Executable code actions elicit better llm agents. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

Oshadha Wijerathne, Amandi Nimasha, Dushan Fernando, Nisansa de Silva, and Srinath Perera. 2025. Scheduleme: Multi-agent calendar assistant. *arXiv preprint arXiv:2509.25693*.

Yue Wu, Zhiqing Sun, Huizhuo Yuan, Kaixuan Ji, Yiming Yang, and Quanquan Gu. 2025. Self-play preference optimization for language model alignment. In *International Conference on Learning Representations (ICLR)*.

Sikuan Yan, Xiufeng Yang, Zuchao Huang, Ercong Nie, Zifeng Ding, Zonggen Li, Xiaowen Ma, Kristian Kersting, Jeff Z. Pan, Hinrich Schütze, Volker Tresp, and Yunpu Ma. 2025. Memory-R1: Enhancing large language model agents to manage and utilize memories via reinforcement learning. *arXiv preprint arXiv:2508.19828*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Hui Yang, Sifu Yue, and Yunzhong He. 2023. Autogpt for online decision making: Benchmarks and additional opinions. *arXiv preprint arXiv:2306.02224*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*. ArXiv:2210.03629.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, Bangkok, Thailand. Association for Computational Linguistics.

Kunlun Zhu, Zijia Liu, Bingxuan Li, Muxin Tian, Yingxuan Yang, Jiaxun Zhang, Pengrui Han, Qipeng Xie, Fuyang Cui, Weijia Zhang, and 1 others. 2025. Where llm agents fail and how they can learn from failures. *arXiv preprint arXiv:2509.25370*.

Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, Tom Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. 2019. Fine-tuning language models from human preferences. In *Proceedings of the 36th International Conference on Machine Learning (ICML) Workshop*. ArXiv:1909.08593.

Appendix

A Use of LLMs

In this work, LLMs are used strictly for research support rather than as sources of substantive content. Their use falls into: (i) serving as the tested and trained model, and (ii) assisting with language refinement during paper writing. For writing support, we used GPT-5 solely to polish text (improving coherence and grammar) while all ideas, logic, results, and technical contributions originate from the authors.

B Potential Risks

Calendar conflict resolution is a high-stakes setting: incorrect accept/decline decisions can cause missed deadlines, lost opportunities, and interpersonal or organizational harm, especially over long horizons where errors compound. Calendar data and org context are also sensitive and can encode confidential relationships and priorities. Additionally, such agents could be misused for surveillance or coercive scheduling, and benchmark success may be over-interpreted because our setting models preferences as structured, role-conditioned rules that omit transient, hard-to-observe factors (fatigue, stress, interpersonal context). To mitigate these risks, we position our work as a controlled abstraction for reproducible evaluation rather than a deployment-ready system.

C Synthetic Data Engine Details

C.1 Organizational Schema

Our synthetic data engine is grounded in *role-conditioned organizational schemas* that capture how different positions operate and make trade-offs in calendar decisions. We first conduct semi-structured interviews with domain practitioners (e.g., PIs and PhD students in academia; executives and engineers roles in tech company) and analyze

both (i) de-identified real-world calendar traces (event titles, recurrence patterns, attendee structures, meeting durations) and (ii) publicly available or provided organizational charts. From these sources, we extract role-specific attributes and encode them into a unified schema.

Schema fields. For each role r, we curate a schema $\mathcal{S}(r)$ consisting of three components:

1. **Regular meeting schemas** $\mathcal{M}(r)$: templates for commonly recurring events, including (i) canonical topics (e.g., "weekly group meeting", "1:1 mentoring", "sponsor sync"), (ii) typical cadence (weekly/biweekly/monthly), (iii) default duration distributions, (iv) attendee patterns (direct reports, cross-team stakeholders, external partners), and (v) common metadata realizations (location type, meeting modality, title variants).
2. **Priority principles** $P(r)$: a small set of explicit, interpretable principles governing decisions under conflict, such as leadership/oversight obligations, deadline sensitivity, people management duties, and external relationship maintenance.
3. **Conflict reasons** $C(r)$: common causes of decline/postpone for that role, such as deadline clashes, hierarchical obligations, travel constraints, task urgency spikes, teaching/committee constraints, or sponsor milestone collisions. Each conflict reason $c \in C(r)$ defines a transformation over event metadata (e.g., inserting a deadline marker, adding a senior attendee, changing modality to "in-person required").

Unified representation. Concretely, a regular meeting template $m \in \mathcal{M}(r)$ is represented as

$$m = \langle \texttt{topic},\ \texttt{freq},\ \texttt{dur},\ \texttt{attendees},\ \texttt{cts.} \rangle,$$

where `constraints (cts.)` includes optional hard constraints (e.g., "must be attended", "cannot be moved") and soft constraints (e.g., "prefer mornings", "avoid back-to-back"). Priority principles are encoded as a weighted set

$$P(r) = \{\langle p_k, w_k, g_k(\cdot) \rangle\}_{k=1}^{K_r},$$

where $g_k(\cdot)$ is an attribute-based trigger function that maps an event (and local context) to $\{0, 1\}$. Conflict reasons are encoded as operators

$$C(r) = \{\mathcal{T}_j\}_{j=1}^{J_r},$$

where each $\mathcal{T}_j$ mutates an event into a plausible competing event (e.g., "upgrade urgency", "attach deadline").

C.2 Conflict Event Generation

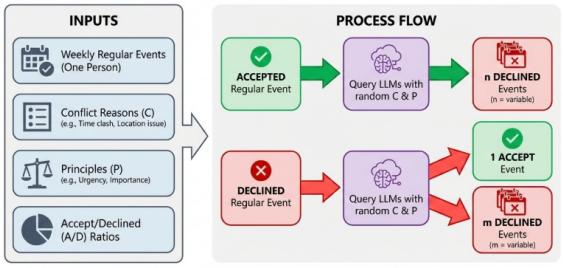

Figure 6: Conflict event generation process.

Figure 6 illustrated the conflict event generation process. Given a role-conditioned weekly calendar $\mathcal{C}$ sampled from $\mathcal{M}(r)$, we generate *conflict rounds* by constructing a candidate set of overlapping events $\mathcal{E}_t$ for each decision round t. Our generation procedure explicitly couples each synthetic conflict with (i) a *conflict reason* $c \in C(r)$ and (ii) a *priority principle* $p \in P(r)$ so that accepted/declined outcomes are explainable and consistent with role behavior.

Step 1: Sample anchor events. We first sample a set of *anchor* regular events from the weekly calendar and assign each anchor a decision label (accepted or declined) based on role-conditioned constraints and accept/reject ratio. Intuitively, accepted anchors reflect high-priority routine obligations (e.g., weekly lab meeting for a PI), while declined anchors reflect lower-priority or optional events. The accept/decline ratio injects controlled randomness into the process.

Step 2: Generate competing events via principle–reason pairing. For each anchor event e at round t, we sample a pairing (p, c) where $p \sim P(r)$ (proportional to w_p and triggers) and $c \sim C(r)$, then apply the corresponding transformation to create competing events that overlap in time. We denote the conflict generator as

$$\mathcal{G}(e; p, c) \to \{e'_1, \ldots, e'_q\},$$

where each e'_i inherits the timeslot of e but differs in attributes (attendees, urgency, topic, location) induced by (p, c).

Case A: accepted anchor → declined competitors. If the anchor e is labeled **accepted**, we generate n plausible **declined** competitors:

$$\mathcal{E}_t \ = \ \{e\} \cup \{e'_1, \ldots, e'_n\}.$$

Competitors are created to be *credible* yet dominated by e under the role's principles, e.g., a PI's weekly group meeting competing with ad-hoc low-stakes chats.

Case B: declined anchor → one accepted competitor + extra declined. If the anchor e is labeled **declined**, we generate (i) one **accepted** competitor $\hat{e}$ that is justified by a strong principle trigger (e.g., deadline-driven sponsor call), plus (ii) m additional **declined** competitors to increase local complexity:

$$\mathcal{E}_t = \{\hat{e}\} \cup \{e\} \cup \{e'_1, \ldots, e'_m\}.$$

This construction ensures each round contains a non-trivial trade-off and supports ranking-based supervision: the accepted event should be near the top even among multiple plausible alternatives.

Attribute realization and naturalization. To improve realism, we instantiate event surface forms using role-specific lexicons and title templates (e.g., "1:1", "sync", "deep dive", "reading group") and generate consistent metadata:

- **Attendees:** sampled from the organizational chart with correct reporting lines (direct reports, peers, external partners).
- **Duration:** sampled from template distributions (e.g., 30min 1:1, 60min weekly meeting) with mild noise.
- **Urgency/deadlines:** inserted via c (e.g., "milestone due 5pm", "release cutoff today").
- **Constraints:** hard constraints introduced for certain roles/events (e.g., committee meeting non-movable).

C.3 Human Verification

We incorporate a human verification stage to ensure (i) *plausibility* of event metadata, (ii) *organizational consistency* (attendee relations match the org chart), and (iii) *decision validity* (accepted/declined labels align with the stated principles). Annotators are provided with the role schema $\mathcal{S}(r)$, the organizational chart, and the conflict round $\mathcal{E}_t$, and are asked to verify both the surface form and the underlying rationale.

Verification checklist. Each datapoint is reviewed with the following criteria:

1. **Role realism:** Are the event topics and cadences plausible for this role?
2. **Org-chart consistency:** Do attendees reflect correct reporting lines and stakeholder relationships?
3. **Conflict coherence:** Do the competing events genuinely overlap and create a meaningful trade-off?
4. **Principle alignment:** Is the accepted event justified by $P(r)$ under the provided context signals?
5. **Metadata quality:** Are titles, locations, and constraints natural (no duplicates, no contradictions)?

Edits and rejection. Annotators can (i) edit event titles/attributes, (ii) swap the accepted label if inconsistent with principles, (iii) rewrite the conflict reason/context for coherence, or (iv) reject the datapoint if it cannot be repaired cheaply.

Annotation protocol. Each datapoint is reviewed by three annotators. The first two annotate independently, proposing edits and/or rejection decisions. A third annotator then adjudicates disagreements and produces the final verified version by consolidating the two reviews. Data annotators are recruited from third party crowd-sourcing platform.

C.4 Example Data

Here is an example data point from generated synthetic organization.

Example Datapoint: Input (Decision round t)

User. PhD student (**James Carter**) at the **BioInnovate Research Lab**.

Conflict events $\mathcal{E}_t$ (overlapping timeslot on 2025-01-03).

- e_1: *"Experiment planning and daily priorities sync"* 14:30–14:45 Attendees: J. Carter, E. White, M. Lee, S. Mitchell Type: internal coordination Location: BioInnovate Research Lab Conference Room
- e_2: *"Equipment calibration check — imaging suite"* 14:32–14:40 Attendees: J. Carter, E. White Type: operations/quality control Location: BioInnovate Research Lab Conference Room
- e_3: *"Data preprocessing script optimization"* 14:35–14:42 Attendees: J. Carter, M. Lee, E. White Type: technical unblock Location: BioInnovate Research Lab Conference Room
- e_4: *"Weekly lab reading group: methods paper"* 14:38–14:44 Attendees: J. Carter, S. Mitchell Type: reading/discussion Location: BioInnovate Research Lab Conference Room
- e_5: *"Career development info session: resume work-*

shop" 14:30–14:33 Attendees: J. Carter, A. Patel Type: professional development Location: BioInnovate Research Lab Conference Room

Context information.

- Lab mission/direction: advancing biomedical research through innovative methodologies (`bioinnovate.org`).
- User responsibilities: develop thesis research, run experiments, analyze data, write papers/present, contribute to mentoring junior students, take courses.
- **Organization chart of BioInnovate Research Lab.**
  - **Dr. Sarah Mitchell** — *Principal Investigator* (Management).
    Responsibilities: scientific vision/long-term strategy; secure funding; mentor/supervise all members; external representation.
  - **Dr. Emily White** — *Postdoctoral Researcher* (Research). Supervisor: PI (Sarah Mitchell).
    Responsibilities: lead projects; mentor students; proposals/reports; write/present manuscripts and talks.
  - **Dr. Michael Lee** — *Postdoctoral Researcher* (Research). Supervisor: PI (Sarah Mitchell).
    Responsibilities: lead projects; mentor students; proposals/reports; write/present manuscripts and talks.
  - **Aisha Patel** — *PhD Student* (Research). Supervisor: PI (Sarah Mitchell).
    Responsibilities: thesis research; analysis/papers/presentations; mentor juniors; coursework.
  - **James Carter** — *PhD Student* (Research). Supervisor: PI (Sarah Mitchell).
    Responsibilities: thesis research; analysis/papers/presentations; mentor juniors; coursework.
  - **Lila Nguyen** — *PhD Student* (Research). Supervisor: PI (Sarah Mitchell).
    Responsibilities: thesis research; analysis/papers/presentations; mentor juniors; coursework.
  - **Rajiv Sharma** — *PhD Student* (Research). Supervisor: PI (Sarah Mitchell).
    Responsibilities: thesis research; analysis/papers/presentations; mentor juniors; coursework.
  - **Nina Garcia** — *PhD Student* (Research). Supervisor: PI (Sarah Mitchell).
    Responsibilities: thesis research; analysis/papers/presentations; mentor juniors; coursework.
  - **Samuel Lee** — *Master's Student* (Research). Supervisor: PI (Sarah Mitchell).
    Responsibilities: focused research project; data collection/analysis/documentation; present results; coursework.
  - **Mia Thompson** — *Master's Student* (Research). Supervisor: PI (Sarah Mitchell).
    Responsibilities: focused research project; data collection/analysis/documentation; present results; coursework.
  - **Elena Martinez** — *Master's Student* (Research). Supervisor: PI (Sarah Mitchell).
    Responsibilities: focused research project; data collection/analysis/documentation; present results; coursework.
  - **Noah Kim** — *Master's Student* (Research). Supervisor: PI (Sarah Mitchell).
    Responsibilities: focused research project; data collection/analysis/documentation; present results; coursework.
  - **Olivia Rodriguez** — *Master's Student* (Research). Supervisor: PI (Sarah Mitchell).
    Responsibilities: focused research project; data collection/analysis/documentation; present results; coursework.
  - **Jordan Rivera** — *Undergraduate Research Assistant* (Research). Supervisor: Postdoc (Emily White).
    Responsibilities: support experiments; data entry/basic analyses; attend lab meetings/reading groups; coursework.
  - **Sophia Chen** — *Undergraduate Research Assistant* (Research). Supervisor: Postdoc (Emily White).
    Responsibilities: support experiments; data entry/basic analyses; attend lab meetings/reading groups; coursework.
  - **Lucas White** — *Undergraduate Research Assistant* (Research). Supervisor: Postdoc (Emily White).
    Responsibilities: support experiments; data entry/basic analyses; attend lab meetings/reading groups; coursework.
  - **Zoe Anderson** — *Undergraduate Research Assistant* (Research). Supervisor: PhD (Aisha Patel).
    Responsibilities: support experiments; data entry/basic analyses; attend lab meetings/reading groups; coursework.
  - **Ethan Park** — *Undergraduate Research Assistant* (Research). Supervisor: PhD (Aisha Patel).
    Responsibilities: support experiments; data entry/basic analyses; attend lab meetings/reading groups; coursework.

Calendar state $\mathcal{C}_t$.

- **Prior decisions.** (Note: The information is summarized here due to limited space, but we provided the full calendar information during evaluation and training) On 2025-01-01 (14:15–15:45), user accepted *"PhD qualifying exam planning session"* and declined: *"Coursework and professional development check-in"*, *"Weekly lab meeting – planning and status update"*, *"Internal project brainstorming session"*, and *"Bluesky reading discussion"*.

Example Datapoint: Output

Accepted event. e_2 (Equipment calibration check — imaging suite).

Declined events. e_1 (daily priorities sync), e_3 (preprocessing script optimization), e_4 (reading group), e_5 (resume workshop).

Priority ranking π_t (high $\rightarrow$ low).

$$\pi_t : e_2 > e_3 > e_1 > e_4 > e_5$$

Decision reasoning.

- **Time-critical quality control:** calibration directly protects near-term experiment validity and data quality; missing it risks invalid data / wasted instrument time.
- **Execution before coordination:** once instruments

are calibrated, follow-up coordination/unblocking can proceed with higher efficiency.

- **Unblocking is next:** preprocessing optimization resolves a pipeline blocker and likely accelerates the day's progress, but is slightly more flexible than a calibration window.
- **Sync is helpful but deferrable:** the broader stand-up includes PI/postdocs, yet can be replaced by an async update if needed.
- **Deferable learning/career items:** reading group and resume workshop have lower immediate cost to reschedule and are not tied to a hard operational dependency.
- **History-consistent:** prior choices favored milestone-/execution-critical sessions over routine meetings and exploratory discussions.
- **Reschedule suggestion:** move e_3 to 14:45–15:00 for rapid follow-up; send a brief written status update to e_1 attendees.

D Evaluation Details

We underscore again here that the evaluation in section 4 is conducted in single-turn manner.

D.1 Prompt Template

We attached the prompt template used for evaluation in section 4.

```
prompt_template: |
    You are tasked with resolving a
    calendar conflict by analyzing the
    situation and making a decision
    based on organizational context and
    historical patterns.

    # Task:
    1. Evaluate all conflict events
    considering:
       - The principles and reasoning
    provided for each event
       - The organizational hierarchy
    and relationships
       - The urgency and importance of
    each event
       - Historical patterns from
    similar past decisions
       - The impact on stakeholders and
    organizational goals
       - Time constraints and scheduling
     flexibility
    2. Rank all conflict events (
    including the regular event) in
    order of priority
    3. Select the single event that
    should be accepted
    4. Respone in the required format.

    # Inputs:

    ## History Conflict Calendar Events
    and User Decisions:
    {history_calendar_events}

    ## Organization Chart:
    {org_chart}

    ## Conflict Calendar Event to Solve:
    {conflict_calendar_event}

    #Output Format:
    Provide your response in the
    following structured format:
    ```json
    {{
        "priority_ranking (total {M}
    events)": ["ranked_event_id_1", ...
    , "ranked_event_id_{M}"],
        "reasoning": "Brief explanation
    of priority ranking and why the
    selected event was accepted",
        "selected_event_to_accept": "
    event_id"
    }}
    ```
```

D.2 More Evaluation Settings

Since our evaluation uses a single-turn interface, we implement agentic rollouts as a chain-of-thought-style output schema. For the ReAct baseline, we prepend a ReAct-style system prompt that instructs the model to produce an explicit <reasoning>...</reasoning> block followed by a <response>...</response> block. For ReAct + Memory, we additionally require a brief memory-aware analysis in an <observation> field: the model first emits <observation>...</observation> containing the provided past-round context, then generates <reasoning>...</reasoning>, and finally outputs <response>...</response>.

D.3 Case Study

Scenario. We analyze a representative conflict round where the agent must choose between a **doctor appointment** and an internal **SEV2 incident meeting**. Both events overlap in time, and the user context indicates this is a *personal healthcare* commitment (non-delegable, often hard to reschedule) versus a mid-severity operational sync (important, but potentially delegable and recoverable via async updates).

Model behaviors. Figure 7 contrasts two models. **GPT-5** correctly ranks the *doctor appointment* above the *SEV2 meeting*, emphasizing that healthcare appointments are typically time-sensitive, have higher personal risk, and are harder to reschedule than many internal meetings. In contrast, **Qwen3-32B** incorrectly prioritizes the *SEV2 meeting*, arguing that missing the meeting could slow

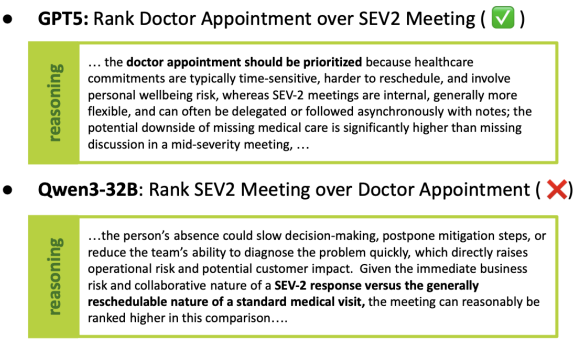

Figure 7: Case study: Responses from two models

mitigation and increase business risk.

Why this matters. This failure mode is not merely a "wrong preference"—it reflects a deeper modeling gap in *role- and person-conditioned* decision policies. In real workflows, users frequently treat certain personal commitments as **hard constraints**: *non-delegable*, *high cost to cancel*, and *limited reschedulability*. Meanwhile, even urgent workplace meetings often admit mitigations: sending a delegate, joining partially, or catching up asynchronously via notes and incident logs.

Error analysis. The incorrect choice is driven by two systematic biases:

- **Overweighting organizational risk signals.** The model over-generalizes from "incident response" to a near-hard obligation, treating SEV2 as always overriding other commitments, without calibrating severity or availability of substitutes.
- **Undermodeling non-delegability and rescheduling friction.** The model implicitly assumes a medical visit is easily movable ("generally reschedulable") and ignores hidden costs: lead times, clinician schedules, cancellation fees, and health risks from delay.

E PEARL Details

E.1 StrategyHub Details

StrategyHub Tool. We implement StrategyHub as an external tool exposed to the agent via function calling. At each round, the StrategyHub is reset to an empty list, and the agent may invoke the tool to *read* or *update* it, which is carried across decision rounds. Unless otherwise specified, the StrategyHub has a maximum capacity of 10 entries.

Provided Tool Schema. To support consistent tool use, we provide the agent with a fixed metadata specification describing the StrategyHub schema, available fields, and constraints:

```
description = "Manage a short list of
    concise strategies. Actions: `list`,
     `update`"

# metadata
_json = {
            "type": "function",
            "function": {
                "name": self.name,
                "description": self.
    description,
                "parameters": {
                    "type": "object",
                    "properties": {
                        "action": {
                            "type": "
    string",
                            "enum": ["
    list", "update"],
                            "description
    ": "Operation to run on the strategy
     list. If action is `list`, the
    response will be the current
    strategies. If action is `update`,
    the response will be the updated
    strategies.",
                        },
                        "strategies": {
                            "type": "
    array",
                            "items": {
                                "type":
    "string",
                                "
    description": "Strategy text to add
    or replace with (each strategy
    should be <=350 characters).",
                            },
                        },
                    },
                    "required": ["action
    "],
                },
            },
        }
```

System Prompt. To ensure a fair comparison, we keep the task prompt unchanged during evaluation. To make the agent aware of the available tool, we prepend an additional system prompt, shown below:

```
system_prompt: |
    You are a calendar conflict
    resolution agent.
    Think step-by-step, you can use the
    StrategyHub tool to help you and
    return the final answer strictly in
    the required JSON.

    StrategyHub tool:
      - You can list strategies with {"
    action": "list"}.
```

```
    - You can update strategies with {
"action": "update", "strategies": ["
strategy 1", "strategy 2", ...]}.
    - Keep strategies short (<=350
chars) and only a small set of the
most useful ones.
    - Decide yourself whether an
update is helpful (e.g., when no
strategies exist or when a better
summary is identified). If so, call
the tool in tool_call fashion before
 producing the final answer.

Before answering, you should first
call the StrategyHub tool to get the
 latest strategies.
Then you should analyze the history
calandar events and see if the
current strategies are helpful or
need to be updated.
    - If the current strategies are
not helpful or empty, you should
update the strategy and update it to
 the StrategyHub with StrategyHub
tool.
    - If the current strategies are
helpful, you should use them to help
 you answer the question.
```

E.2 Training and Validation Data Details

We construct four synthetic organizations, each containing 10 users. For every user, we synthesize a one-year calendar with realistic recurring meetings and injected conflict episodes, and then pool the calendars across all users and organizations to form the full dataset. We randomly split the resulting dataset into training and validation sets using an 80/20 ratio. For training efficiency, we set the environment parameters $W = M = 5$.

E.3 Baseline Details

Zero-shot. This is the first single-turn baseline. We use direct prompting under the same evaluation setting described in Section D.

SFT. This is the second single-turn baseline. We implement the SFT baseline using the LlamaFactory framework (Zheng et al., 2024). Due to limited computational budget, we fine-tune the base model in a single-turn setting. The SFT baseline is trained on the same training subset as **PEARL**. We format the training data as independent single-turn conversations, where each decision round is treated as a separate example. We keep the model's thinking mode enabled throughout training.

Zero-shot + StrategyHub. This is the multi-turn baseline. We add the same system prompt as **PEARL** and grant the agent access to the StrategyHub tool, but do not apply any training.

E.4 PEARL Training Details

We implement the training recipe based on rLLMs framework (Tan et al., 2025). Note that we didn't perform any cold-start SFT. We directly train with original checkpoint. To stabilize preference learning and avoid cross-user leakage within an episode, we ensure that each episode contains events from exactly one user. Since training on 104-step trajectories is both unstable and prohibitively long, we instead train the model on shorter-horizon instances by setting the number of decision rounds to $N = 20$ for the training subset, while keeping validation aligned with the full evaluation setting by using $N = 104$.

Training Logs. Figure 8 reports the learning dynamics of **PEARL** during RL training. We plot the mean reward (averaged over training rollouts) as a function of update steps.

Figure 9 visualizes the average response length over the same training trajectory.

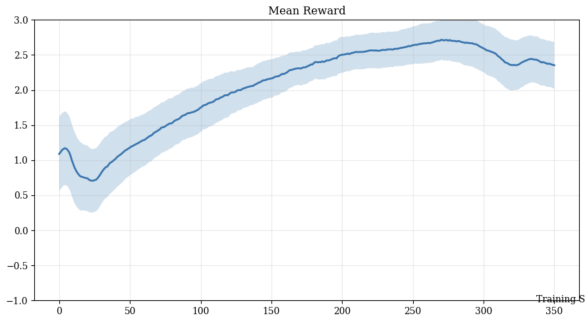

Figure 8: **Mean reward over RL training.**

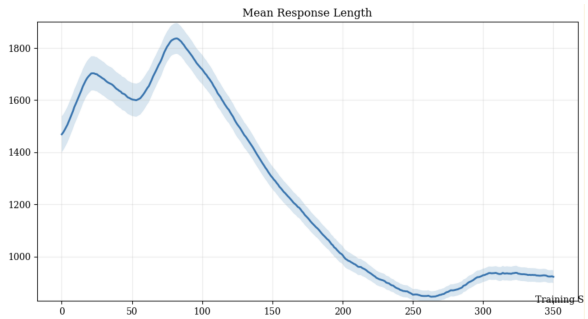

Figure 9: **Average response length over RL training.**

Computation Resource. All training is conducted on 8× NVIDIA H100 GPUs (80GB memory per GPU). The training is consumed around 40 GPU hours.

Training Hyperparameters. Training hyperparameters and system configurations are summarized in Table 3.

Group	Parameter	Value
Algorithm	Advantage estimator	`algorithm.adv_estimator=grpo`
	KL coefficient	`algorithm.kl_ctrl.kl_coef=0.001`
Model / PPO	Base model	`Qwen/Qwen3-4B`
	Learning rate	`actor_rollout_ref.actor.optim.lr=1e-6`
	PPO clip (high)	`actor_rollout_ref.actor.clip_ratio_high=0.28`
	Loss aggregation	`seq-mean-token-mean`
	Use KL loss term	`actor_rollout_ref.actor.use_kl_loss=False`
Batch / Length	Train batch size	`data.train_batch_size=16`
	Val batch size	`data.val_batch_size=10`
	Max prompt/response length	`16384 / 16384`
Rollout (train / val)	Rollout engine	`vllm` (`mode=async`)
	Samples per prompt (train)	`actor_rollout_ref.rollout.n=8`
	Temperature (train)	`0.7`
	Samples per prompt (val)	`actor_rollout_ref.rollout.val_kwargs.n=1`
	Temperature (val)	`0.6`
	Top-p (val)	`0.95`
Efficiency / Systems	GPUs $\times$ nodes	`trainer.n_gpus_per_node=8`, `trainer.nnodes=1`
	Max tokens per GPU (PPO)	`actor_rollout_ref.actor.ppo_max_token_len_per_gpu=32768`
	vLLM GPU mem util.	`actor_rollout_ref.rollout.gpu_memory_utilization=0.85`
	Grad checkpointing	`actor_rollout_ref.model.enable_gradient_checkpointing=True`
Stepwise advantage	Enable	`rllm.stepwise_advantage.enable=True`
	Mode	`rllm.stepwise_advantage.mode=per_step`

Table 3: **Key training and rollout hyperparameters for PEARL (Qwen3-4B).**