ENGAGEMENT PROCESS: MODELING AGENTS AND ENVIRONMENTS EVOLVING IN TIME

Anonymous authors
Paper under double-blind review

ABSTRACT

In real-world interaction, computation, action execution, and environmental events overlap across different timescales. Synchronous observation–action loops bind computation and interaction to shared decision boundaries, obscuring their distinct temporal structure. We introduce *Engagement Process* (EP), a new interaction paradigm that explicitly represents the agent's internal state and models the agent and environment as two processes evolving on a shared physical timeline. This makes computational progress and duration explicit while removing the one-to-one coupling between observations and actions. We formalize EP, analyze the utility of timely observations and access to computational progress, and present basic value-learning and policy-gradient methods. Guided by EP, we augment LLMs with *TimeRoPE* for temporal awareness and an EP interaction protocol that coordinates agent–environment interaction as both evolve in physical time. Experiments show that EP agents can learn when to compute, wait, and act, enabling more effective interaction under deadlines and asynchronous events.

1 INTRODUCTION

The world does not pause while an agent thinks or acts. The environment may change during a robot's ongoing motion; a digital assistant may receive an urgent request during its thinking process. Computation and actions take time, while observations arrive at different rates or irregular moments. Effective behavior therefore requires coordinating how long to compute, when to act, and how to revise ongoing work as the environment evolves.

Markov decision processes (MDPs) and partially observable MDPs (POMDPs) describe decision steps in which an agent selects an action using available information and receives environmental feedback (Puterman, 1994; Kaelbling et al., 1998). Common LLM-agent loops, including ReAct, follow this organization (Yao et al., 2023). Temporal extensions address complementary aspects: semi-Markov models and options represent extended actions (Howard, 1971; Sutton et al., 1999); metareasoning studies deliberation costs (Russell & Wefald, 1991; Zilberstein, 1996); full-duplex and asynchronous agents support concurrent, time-aligned interaction (Veluri et al., 2024; Cui et al., 2026; Orantqing et al., 2026). Our focus is on the agent's internal dynamics between observations and actions, including how its internal state evolves while the environment continues to change. Appendix A provides detailed comparisons.

We introduce *Engagement Process* (EP), a new interaction paradigm modeling agent and environment as distinct stateful processes on a shared physical timeline. Alongside environment state s_t, the agent's internal state m_t records retained information, intermediate results, computational progress, and pending work. Thus, m_t describes not only what the agent has observed, but also what it has computed. Either process can evolve without receiving an event: the agent can keep thinking while the environment keeps changing. Observations originate from s_t and actions from m_t, naturally decoupling their timing rather than pairing them into turns (Figure 1). This formulation naturally accommodates unexpected observations, asynchronous actions, and concurrent tasks, with observations and actions occurring at different or varying rates. Our analysis identifies two benefits while holding computational capability fixed. We formalize EP, define its optimization objective, and describe basic algorithms for policy evaluation and learning. We further analyze how observation timing and access to internal computational progress affect decision-making under specified execution constraints.

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.

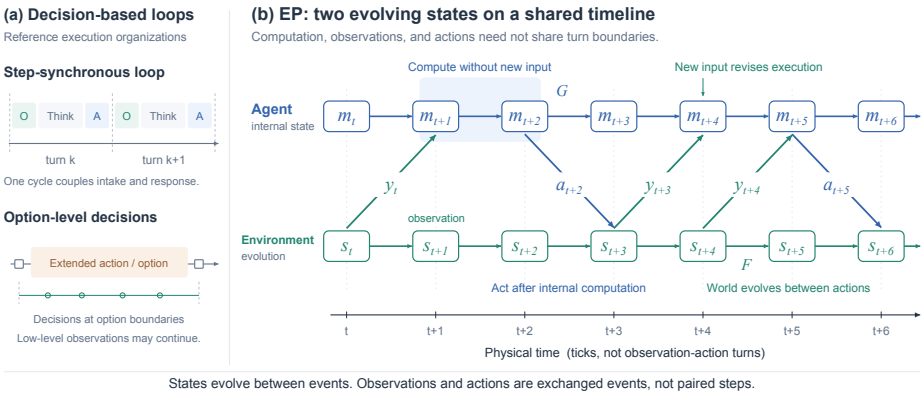


Figure 1: Agent–environment interaction in physical time. (a) Step/option-based interfaces organize interaction around decision boundaries. (b) EP models agent internal state m_t and environment state s_t on a shared timeline, enabling computation without new observations and decoupled observation–action timing.

To bring EP to LLM-based agents, we address two limitations of conventional LLM interfaces: token positions do not explicitly encode elapsed physical time, and turn-based interaction ties observation delivery and action submission to turn boundaries. Guided by EP, we introduce *TimeRoPE* for temporal awareness and a persistent *EP interaction protocol* that lets the agent interact with the environment as both evolve in physical time.

Our evaluation progresses from controlled illustrations to LLM temporal control and diverse task environments. Toy experiments illustrate how EP supports adaptive deliberation and dynamic responses to changing timing requirements. A rhythm task demonstrates that TimeRoPE enables LLMs to use elapsed-time information during decoding for precise action timing. A time-window-constrained mathematics task shows that the EP protocol supports autonomous coordination of reasoning, waiting, and answer submission. Finally, a suite of 14 heterogeneous environments demonstrates the practical applicability of EP, with trajectory-based training establishing learnability and paired interface comparisons showing the benefits of timely observation integration.

Our contributions are threefold:

- **A new interaction paradigm.** EP explicitly models agent internal state and environment dynamics on a shared physical timeline, decoupling observations and actions. We define its objective and basic learning methods, and analyze the value of timely observations and internal progress information.
- **A time-aware LLM instantiation.** We instantiate EP with TimeRoPE and a persistent interaction protocol. Together, they support temporal awareness, observation insertion, independent action submission, and autonomous wake-up.
- **Empirical validation.** Toy experiments, rhythm and timed mathematics tasks, and 14 heterogeneous environments test EP's capabilities. Results demonstrate adaptive computation, precise timing, autonomous temporal control, and learning under asynchronous events.

2 Engagement Process

EP models the agent and environment as distinct stateful processes on a shared physical timeline. For convenience, we discretize this timeline into ticks $t \in \{0, 1, \ldots\}$ of duration $\Delta > 0$. Ticks specify temporal resolution, not reasoning passes or observation–action turns.

2.1 States and Temporal Dynamics

The environment state $s_t \in \mathcal{S}$ describes the external world and ongoing processes. The agent's internal state $m_t \in \mathcal{M}$ can include retained information, predictions, plans, intermediate results, and computation progress. Even with the same observation history, further computation may lead to

better action choices. EP explicitly models the evolution of the agent's internal state and generates actions from that state, rather than specifying actions directly as a function of observation history.

Let $\mathcal{A}$ and $\mathcal{Y}$ denote the action and observation spaces, with $a_t \in \mathcal{A}$ and $y_t \in \mathcal{Y}$. Each value may represent one or multiple simultaneous events; both spaces include a no-event value $\perp$. The environment $\mathcal{E} = (\mathcal{S}, F, O, r, \rho_E)$ and agent $\mathcal{C} = (\mathcal{M}, G, \pi, \rho_C)$ are specified separately:

$$y_t \sim O(\cdot \mid s_t), \qquad a_t \sim \pi(\cdot \mid m_t), \tag{1}$$

$$s_{t+1} \sim F(\cdot \mid s_t, a_t), \qquad m_{t+1} \sim G(\cdot \mid m_t, y_t, a_t). \tag{2}$$

Here ρ_E, ρ_C specify initialization and r the task reward. Emissions are conditionally independent given (s_t, m_t); both sides then update using the exchanged events. Observations update the agent's internal state, which in turn guides subsequent action selection.

The environment may continue to change even when the agent takes no new action ($a_t = \perp$), and the agent may continue its internal computation even when no new observation arrives ($y_t = \perp$). How much computation the agent can perform within each interval Δ depends on its implementation; a reasoning process may span multiple ticks.

Implications of explicit agent state. Action execution and delayed effects evolve through F, while G incorporates incoming observations and updates the agent's internal state as computation proceeds. Separate emissions support asynchronous, multi-rate interaction without paired observation–action turns. The choice of $(\mathcal{M}, G)$ defines a design space for explicitly timed memory retrieval, predictive computation, and interacting internal modules, using only agent-accessible information. EP does not prescribe a particular architecture. For an autoregressive Transformer-based LLM, the current context and generated prefix provide an intuitive representation of m_t.

Relation to POMDPs. A POMDP can represent this interaction by augmenting the environment state with the agent's internal state. This construction combines world dynamics with the agent's computation and memory mechanisms: changing the agent then changes the augmented model, even when the task remains the same. EP instead specifies the environment and agent separately, making it explicit which dynamics belong to the world and which belong to the agent. This allows different architectures and computational mechanisms to be evaluated against the same environment and task.

2.2 Objective, Value, and Learning

For an evaluation horizon of T ticks and discount $\gamma \in (0, 1]$, define

$$J_T(\mathcal{C}; \mathcal{E}) = \mathbb{E}_{\mathcal{C},\mathcal{E}}\left[\sum_{t=0}^{T-1} \gamma^t r(s_t, a_t)\right], \qquad J^*_{T,\mathfrak{C}}(\mathcal{E}) = \sup_{\mathcal{C}\in\mathfrak{C}} J_T(\mathcal{C}; \mathcal{E}). \tag{3}$$

The admissible class $\mathfrak{C}$ specifies architectures and execution constraints. Learning can optimize (G, π) with $\mathcal{E}$ fixed; m_t evolves through these mechanisms rather than being freely chosen. Deliberation affects return through elapsed time, deadlines, and lost opportunities.

Value under partial observability. Let $b_t(s) = \Pr(s_t = s \mid \mathcal{H}_t)$, where $\mathcal{H}_t$ records past events and realized agent-state updates through m_t, before current emissions. For a given agent $\mathcal{C}$, $V^{\mathcal{C}}_t(b_t, m_t)$ denotes the expected cumulative reward from time t to the end of the evaluation horizon, conditioned on belief b_t and internal state m_t, with future rewards discounted by γ per tick. Under the assumptions in Appendix B.1,

$$V^{\mathcal{C}}_t(b_t, m_t) = \mathbb{E}_{\mathcal{C},\mathcal{E}}\left[\bar{r}_t + \gamma V^{\mathcal{C}}_{t+1}(b_{t+1}, m_{t+1}) \mid b_t, m_t\right], \qquad V^{\mathcal{C}}_T = 0, \tag{4}$$

where $\bar{r}_t = \mathbb{E}[r(s_t, a_t)|b_t, m_t]$ and $J_T = \mathbb{E}_{(b_0,m_0)}[V^{\mathcal{C}}_0(b_0, m_0)]$. Belief captures environmental uncertainty; m_t additionally captures execution progress, so equal beliefs need not imply equal values. The recursion advances even without new observations or actions.

Learning on the physical timeline. This recursion supports tick-wise TD evaluation; a parameterized agent can instead be optimized through trajectory-level policy gradients. Appendix C gives standard TD and REINFORCE instances (Sutton, 1988; Williams, 1992) with physical-time returns and their required assumptions.

2.3 WHY TEMPORAL FLEXIBILITY MATTERS

EP offers temporal flexibility by allowing observations to arrive without requiring computation to finish within the same cycle. We analyze when this decoupling improves utility over a fixed-cycle interface. Full assumptions and proofs are provided in Appendix B.

Separating observation timing from computation time. As a fixed-cycle abstraction of ReAct-style agent loops (Yao et al., 2023), consider an interface that receives observations every $\ell > 0$ time units and requires each response to be computed within its admission cycle. Short cycles reduce observation delay but limit computation; long cycles allow more computation but delay task receipt. EP instead delivers observations immediately without imposing this computation cap.

Under the task assumptions in Appendix B.2, let d be the response window and $\tau^* > 0$ the computation duration achieving the unique positive maximum utility, with $d \geq \tau^*$. With identical computational capabilities, immediate submission after computation, and uniformly random arrival phases, the synchronous interface matches EP's optimal expected utility if and only if

$$\tau^* \leq \ell \leq d - \tau^*. \tag{5}$$

The cycle must accommodate computation while limiting observation delay. Thus, when $\tau^* \leq d < 2\tau^*$, every fixed cycle incurs a strict expected utility loss.

3 INSTANTIATING EP WITH LLMS

Extended reasoning can improve LLM performance, but it also delays action, posing a challenge for embodied agents and other time-sensitive applications. EP provides a way for LLMs to adapt their reasoning and actions to the time available and to changes in the environment. Instantiating EP with LLMs requires temporal information and an interface for interaction during generation. Token order alone does not reveal elapsed time, while turn boundaries restrict observation delivery and action submission. We address these gaps with TimeRoPE and a persistent EP interaction protocol.

For an LLM, m_t can be represented by its current context, including received observations, generated reasoning, and actions. Decoding implements both G and π: generated tokens update m_t, and completed action blocks specify external actions.

TimeRoPE: representing physical time. TimeRoPE introduces additional, dedicated temporal rotary dimensions in queries and keys, separate from the model's original positional rotary dimensions. Existing positional dimensions, including those used for video, remain unchanged. Physical session time therefore provides an additional encoding axis rather than replacing an existing one. Starting from t_{start}, a token at physical time t_i receives the session-relative index

$$q_i = \left\lfloor \frac{t_i - t_{\text{start}}}{20\,\text{ms}} \right\rfloor. \tag{6}$$

Rotations based on q_i expose elapsed intervals while preserving positional encoding: tokens may share a time index or be separated by long waits. The 20 ms resolution specifies temporal quantization, not interaction frequency or guaranteed accuracy. Temporal rotation makes elapsed intervals available through attention; it does not return a numerical timestamp. A separate clock-reading operation provides session time for explicit numerical reasoning.

EP protocol: interacting on the physical timeline. Deliberation and action generation consume time while the environment evolves. For *observation insertion*, the runtime pauses decoding at a token boundary, inserts the observation, and resumes generation in the updated context. For *independent action submission*, closing an action block commits it immediately, allowing execution to proceed asynchronously while generation continues. Results arrive as subsequent observations. For *autonomous wake-up*, the agent can suspend inference until a specified time or a new observation arrives, whichever occurs first. During waiting, context is retained while the environment continues to evolve.

Figure 2 summarizes how TimeRoPE and the EP protocol coordinate time-aware generation and interaction. Implementation details appear in Appendix D.

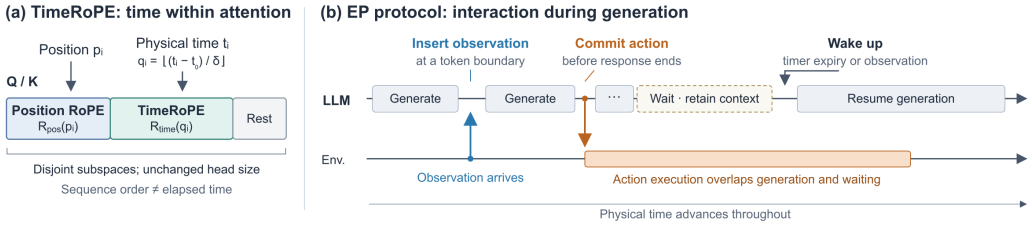


Figure 2: LLM implementation of EP. (a) Position and time rotate disjoint Q/K subspaces without enlarging the head; $t_0 = t_{\text{start}}$ and $\delta = 20\,\text{ms}$. (b) Observations enter during generation, actions commit independently, and waiting retains context until timer expiry or observation arrival.

4 Experiments

Our experiments examine how EP supports decisions about when to compute, wait, and act. We first use controlled toy tasks to study computation allocation under deadlines and interruption decisions during ongoing execution. We then evaluate the two components of our LLM instantiation: rhythm tasks test whether TimeRoPE supports precise action timing without explicit clock readings, while Timed MATH-500 tests whether the EP protocol enables models to coordinate reasoning, waiting, and answer submission using clock feedback. Finally, experiments across 14 heterogeneous environments assess the benefits of observation delivery during generation and the learnability of EP control behaviors through supervised fine-tuning.

4.1 Toy Experiments

We use two controlled toy experiments to study computation allocation under deadlines and interruption decisions during execution. The tabular Q-learning policies connect these experiments to the temporal-difference learning principles discussed above. Details are given in Appendix E.

4.1.1 Urgency-Aware Deliberation

Each task has a difficulty, a deadline, and rewards or penalties for its outcome. Before acting, the agent selects one of five deliberation modes: deeper modes are more accurate but take longer. We consider single-task episodes and chains of ten tasks with absolute deadlines, where computation on earlier tasks reduces the time available for later ones. Policies use tabular Q-learning with identical task distributions, mode qualities, and rewards. During training, deliberation advances the clock in **EP** but is treated as instantaneous in **Step**. Policies are evaluated with computation time charged.

Table 1: Deliberation results. Both trained policies are evaluated with computation time charged.

Setting	Training	Return ↑	Success (%) ↑	Timeout (%) ↓	Mode 5 (%)
Single-task	EP	**0.800**	**46.7**	**0.0**	21.2
	Step	−1.032	16.1	79.0	93.6
Sequential-task	EP	**10.07**	**50.1**	**3.7**	15.9
	Step	−14.98	8.4	88.9	84.0

Table 1 shows that treating deliberation as instantaneous during training leads Step to favor the slowest mode, resulting in a 79.0% timeout rate on single tasks when computation time is charged. EP reduces slowest-mode usage from 93.6% to 21.2%, with no observed timeouts. In task chains, where computation consumes downstream time budgets, EP improves return from −14.98 to 10.07. The mode distributions in Appendix E.1 show that EP reserves deeper deliberation for tasks with slack, supporting time-sensitive computation allocation rather than uniformly shorter computation.

4.1.2 Patrol with Interruptible Execution

An agent navigates among checkpoints and performs time-consuming routines while alarms arise elsewhere. Resolving alarms before their deadlines earns rewards; missed alarms incur penalties. We first test the value of decisions during execution. Checkpoint handling takes 20 ticks, comparable

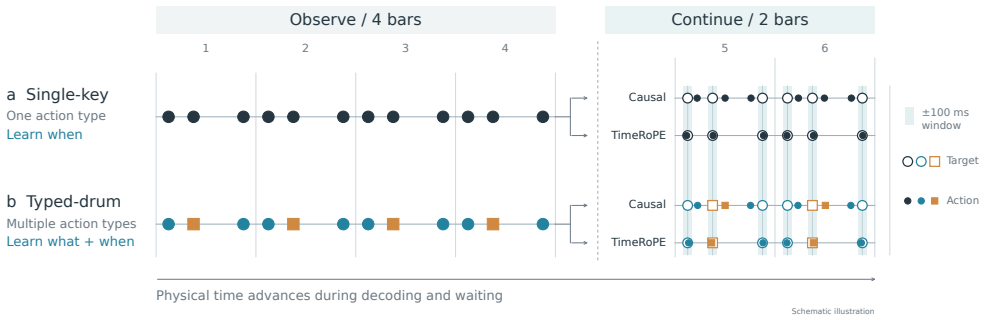


Figure 3: Rhythm-task setup and schematic timing comparison. The model observes a four-bar demonstration and continues the pattern for two bars while physical time advances.

to alarm deadlines of 14–22 ticks. Both agents share dynamics and action primitives, but **EP** can interrupt at any tick, whereas **Loop** re-decides only at module boundaries.

We then introduce phase-dependent interruption costs to study when interruption is worthwhile. Routines contain two or three phases, and the learned EP policy uses alarm urgency, current phase, remaining phase time, and distance to decide whether to continue or interrupt. We compare it with phase-boundary **Loop** and two interruptible rule-based controllers: **Patch**, which uses urgency and phase, and **PatchPro**, which additionally considers reachability, progress, and costs.

Table 2: Patrol results with module- and state-level interruption. Cost denotes interruption cost.

(a) Module-level interruption.

Method	Return ↑	Resolved (%) ↑	Expired (%) ↓	Ticks/alarm ↓
EP	**896.6**	**90.9**	**8.0**	**10.9**
Loop	−163.5	51.5	47.1	14.4

(b) State-level interruption.

	Two-phase routine			Three-phase routine		
Method	Return ↑	Resolved (%) ↑	Cost	Return ↑	Resolved (%) ↑	Cost
EP	**175.1**	**62.8**	61.7	**143.3**	**62.4**	80.6
PatchPro	25.1	53.9	28.0	68.8	57.2	48.4
Patch	−156.5	47.6	63.7	−101.0	48.4	27.1
Loop	−345.0	34.9	0.0	−298.7	37.5	0.0

Allowing decisions during execution raises alarm resolution from 51.5% to 90.9% in the module-level setting (Table 2a). With phase-dependent costs, EP achieves the highest return in both phase structures despite paying more interruption cost than PatchPro (Table 2b). These results illustrate the value of timely intervention and of learning to balance its benefits against switching costs. Detailed rules and execution traces appear in Appendix E.2.

4.2 Learning Precise Action Timing

We next ask whether an LLM operating on the EP timeline can use elapsed physical time to act at the right moment. We introduce two rhythm tasks that progressively separate action timing from action selection. In the *single-key* task, the action identity is fixed, so the only challenge is *when* to act. In the *typed-drum* variant, rhythmic patterns contain multiple action types, requiring the model to determine both *what* to play and *when* to play it.

Figure 3 illustrates the task setup. In both tasks, the model observes a four-bar demonstration and then performs for two bars while the environment continues to evolve. Evaluation instances include temporal variants that preserve the same visible token sequence up to the first decision point while changing the timing of those events and hence the correct action times. No numerical timestamps or explicit clock readings are exposed. We compare matched causal SFT and TimeRoPE SFT models; full task, training, and evaluation details are provided in Appendix F.

Table 3: Rhythm-task performance (%; ±100 ms timing tolerance). Zeroed and shuffled ablate TimeRoPE's time input.

	Single-key	Typed drums	
Model / condition	**Hit**	**Type**	**Type + timing**
Causal SFT	11.0	100.0	4.6
TimeRoPE SFT	**87.6**	99.9	**60.3**
Time-input interventions			
Zeroed	11.0	91.5	3.0
Shuffled	13.2	80.4	3.8

Table 4: Timed MATH-500 results (%). Joint requires both correct and on-time submission.

Model / interface	**Correct**	**On-time**	**Joint**
Qwen3.5-9B / Turn-based	51.2	48.8	47.8
Qwen3.5-9B / EP	67.8	67.2	64.0
EP-SFT 9B / EP	**89.4**	**94.4**	**89.0**

Table 3 highlights the role of temporal information in translating action selection into precisely timed behavior. On typed drums, both models achieve near-perfect type accuracy when emitting actions, yet joint type-and-timing accuracy rises from 4.6% with causal SFT to 60.3% with TimeRoPE. Zeroing or shuffling TimeRoPE's time input largely removes its timing advantage, while also causing some degradation in type accuracy, supporting that the learned policy relies on the supplied temporal information. These results establish TimeRoPE as an effective mechanism for fine-grained, interval-based timing in EP. When no explicit clock reading is available, temporal positions allow the model to condition its actions on elapsed physical time rather than token order alone.

4.3 Autonomous Temporal Control with the EP Protocol

The rhythm tasks test fine-grained timing without explicit clock readings. We next consider a complementary setting in which the model receives numerical clock feedback and must autonomously coordinate reasoning, waiting, and action.

To study this, we construct *Timed MATH-500* from MATH-500 (Lightman et al., 2024), a 500-problem subset of the MATH benchmark (Hendrycks et al., 2021). Standard MATH-500 evaluates whether the model derives the correct mathematical answer. Timed MATH-500 additionally assigns each problem a submission window, so correctness alone is insufficient: the model must both solve the problem and deliver its answer at the appropriate time. For reproducible timing across runs, a virtual clock advances with generated tokens and explicit waiting, and the model must autonomously query the clock, schedule waiting, and decide when to submit without further user prompts.

We evaluate Qwen3.5-9B with turn-based and EP interaction, together with its EP-SFT variant (EP-SFT 9B) obtained by EP-specific post-training on mathematical trajectories derived from OpenThoughts3 (Guha et al., 2026). While both interfaces expose the same temporal operations, EP allows actions to take effect during ongoing reasoning rather than only after a response completes.

Table 4 shows that EP adaptation raises joint success from 47.8% to 64.0% with unchanged weights. EP-SFT further achieves 94.4% on-time submission and 89.0% joint success, supporting the effectiveness of EP interaction and its associated training for autonomous temporal control.

The examples in Table 5 expose two complementary failure modes. In panel (a), the turn-based model repeatedly expresses an intention to query the clock but never executes it, while the same model under EP successfully waits and submits on time. In panel (b), both base-model settings continue deliberating despite having essentially solved the problem, whereas EP-SFT submits only 0.358 seconds after receiving post-wait feedback. Together, these cases separate the ability to execute actions during reasoning from the learned policy for deciding when to stop deliberating and commit. Expanded case analyses and timeout diagnostics appear in Appendix G.

4.4 Learning and Evaluation across EP Environments

We evaluate EP across 14 heterogeneous environments with shared temporal and action-lifecycle semantics. Using this suite, we examine whether EP control behaviors can be learned through supervised fine-tuning and whether receiving observations during generation improves task performance. Appendices H–I provide implementation details and disaggregated results.

Table 5: Two representative behavioral contrasts on Timed MATH-500.

(a) Turning an intention into an action

$21420N$ divisible by 6; answer $N = 6$.
Window: $[208, 261]$ s.

Observed behavior	Outcome
Qwen3.5-9B / Turn-based	
"Let's call `read_clock`*."* Repeats a two-line clock-query intention 274 times; executes no query.	**No submission**
Qwen3.5-9B / EP	
Reads at 10.595 s, waits 198 s, then submits 6 at 214.296 s.	**Correct, on time**
EP-SFT 9B / EP	
Reads at 5.865 s, waits 203 s, then submits 6 at 210.340 s.	**Correct, on time**

(b) Ending deliberation at the right time

$\min(x^4 + 4y^2 + 4z^4)$, $xyz = 2$, $x, y, z > 0$; answer 16.
Window: $[156.197, 197.842]$ s.

Observed behavior	Outcome
Qwen3.5-9B / Turn-based	
Receives 160.323 s inside the window, but generates 7,683 more tokens before submitting 16.	**14.277 s late**
Qwen3.5-9B / EP	
"So the minimum is indeed 16." Keeps checking; no clock query.	**No submission**
EP-SFT 9B / EP	
Receives 156.894 s; after only 54 tokens, submits 16 0.358 s later.	**Correct, on time**

Table 6: SFT performance on 560 episodes. All runs allow at most 100 turns; SFT models retain completed reasoning in context.

Model	Training	TimeRoPE	Success ↑	Reward ↑
Qwen3.5-9B Base	None	Disabled	0.000	0.000
Qwen3.5-9B Instruct	None	Disabled	0.000	0.000
Qwen3.5-9B Base	EP SFT	Disabled	0.577	70.387
Qwen3.5-9B Base	EP SFT	Enabled	0.591	72.143
Qwen3.5-9B Instruct	EP SFT	Disabled	0.598	72.649
Qwen3.5-9B Instruct	EP SFT	Enabled	**0.623**	**75.387**

Batch environment architecture. Temporal templates, domain scenarios, and embodiment and verification components are compiled, parameterized, and validated into a shared event-driven runtime for synthetic trajectory generation and frozen evaluation (Appendix Figure 8).

Benchmark and evaluation protocol. The 560-episode holdout has 40 episodes per environment and 140 per embodiment level: 140 single-task, 140 serial-feasible, and 280 async-required episodes (Appendix Table 24). Training and evaluation schedules and seeds are disjoint. Under matched timing, computation advances the environment clock. We evaluate all episodes with a 100-turn cap; full success requires successful termination with maximum reward.

Models and SFT setup. We synthesize trajectories on disjoint production schedules, recording reasoning, actions, waits, observations, rewards, and timing without hidden events, verifier state, or oracle plans. Reward-prioritized selection and lifecycle validation retain 2,832 trajectories balanced across single-task, serial, and asynchronous categories; the latter covers background execution, preemption, and cancel-and-replan behavior. Four Qwen3.5-9B models share data and optimization settings in a 2×2 design: Base versus Instruct initialization, with temporal positions enabled or disabled.

SFT results. SFT raises full success from zero to 57.7–62.3% (Table 6), demonstrating learnable EP control behavior. Using TimeRoPE adds 1.4 percentage points for Base and 2.5 for Instruct, also improving mean reward. Instruct consistently outperforms Base, indicating that initialization matters. Appendix I gives training details and level-wise results.

Interface ablation: EP versus Step. We compare EP with a synchronized step interface under matched environments, schedules, seeds, actions, model settings, timing draws, and 100-turn budgets. EP may interrupt generation when an observation arrives; Step queues the observation until generation finishes. To keep reasoning evidence homogeneous, we include only models with provider-native raw

Table 7: An EP coffee-shop trace: brewing and handoff must meet a deadline while restocking runs in parallel. Observations interrupt reasoning during overlapping actions; ellipses omit intermediate events.

EP stream	Execution state
`<OBS> drink_ticket open; ∼20 s remain. </OBS>`	**ticket open**
I should start the brew immediately... `<ACTION>{"arguments":{},"name":"brew_drink"}</ACTION>`	**brew: BG**
The `brew_drink` is running in background. Now I should start `restock_bar` in parallel... `<ACTION>{"arguments":{},"name":"restock_bar"}</ACTION>`	**brew: BG** **restock: FG**
We have `restock_bar` running in foreground... and `brew_drink` running in background... ← **observation interrupts decode**	**BG + FG** **3.0 s overlap**
`<OBS> bar_restock event; 18 s remain. </OBS>`	**event delivered**
Now I have... **[interrupted]** `<OBS> drink_brew_result_ready. </OBS>`	**brew complete** **result ready**

`<ACTION>{"arguments":{},"name":"inspect_brew_result"}</ACTION>`
...
`<ACTION>{"arguments":{},"name":"handoff_drink"}</ACTION>`
	both tasks complete
	reward = 100

Table 8: Paired EP–Step results on 560 episodes with a 100-turn cap and provider-native raw reasoning. Gains are EP minus Step.

Model	EP success	Step success	Success gain	EP reward	Step reward	Reward gain
DeepSeek-V4-Pro	**0.423**	0.323	+0.100	**57.887**	48.869	+9.018
GLM-5.2	**0.361**	0.213	+0.148	**53.750**	39.554	+14.196
Qwen3.8-Max	**0.286**	0.209	+0.077	**46.101**	36.071	+10.030

reasoning: DeepSeek-V4-Pro (DeepSeek-AI, 2026), GLM-5.2 (GLM-5-Team, 2026; Z.ai, 2026), and Qwen3.8-Max (Alibaba Cloud, 2026). Appendix H.3 gives the paired protocol and category-level breakdowns.

Table 7 illustrates successful asynchronous execution: foreground and background actions overlap while incoming observations interrupt ongoing reasoning before the next control is complete.

EP improves both success and reward across all three model families (Table 8): average success rises from 0.248 to 0.357 and reward from 41.498 to 52.579. GLM-5.2 gains most: 14.8 percentage points and 14.196 reward. Improvements span both complete task resolution and graded progress, suggesting benefits beyond one provider.

5 Conclusion and Discussion

Engagement Process (EP) places agent computation and environment dynamics on a shared physical timeline, exposing internal progress and decoupling observations from actions. Under stated constraints, our analyses establish benefits of timely observations and progress-aware decisions. TimeRoPE and the EP protocol enable physical-time encoding and asynchronous interaction in LLMs. Experiments demonstrate improvements in timing and asynchronous tasks, with trajectory-based training further improving performance. These findings highlight when to compute, wait, and act as integral parts of agent behavior.

Future work includes deeper theoretical analysis with explicit models of computation and learning algorithms for computation allocation and temporal control. Extending EP to multimodal and embodied agents can further explore coordination among perception, reasoning, and action.

AI USE STATEMENT

Generative AI assisted manuscript drafting and revision, literature search and synthesis, figures, research code and environments, theoretical discussions and preliminary analyses, and synthetic data and training-trajectory generation. The authors refined and verified the theoretical arguments and references and take full responsibility for the final manuscript, code, data, and results.

REPRODUCIBILITY STATEMENT

Experimental setups and parameters are detailed in the corresponding appendices.

REFERENCES

Alibaba Cloud. Qwen3.8-Max. GitHub repository, 2026. URL https://github.com/AlibabaCloud-Official/Qwen3.8-max. Accessed September 25, 2026.

Bastian Alt, Matthias Schultheis, and Heinz Koeppl. POMDPs in continuous time and discrete spaces. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 13151–13162. Curran Associates, Inc., 2020.

Ruzena Bajcsy, Yiannis Aloimonos, and John K Tsotsos. Revisiting active perception. *Autonomous Robots*, 42(2):177–196, 2018.

Junbo Cui, Bokai Xu, Chongyi Wang, Tianyu Yu, Weiyue Sun, Yingjing Xu, Tianran Wang, Zhihui He, Wenshuo Ma, Tianchi Cai, Jiancheng Gui, Luoyuan Zhang, Xian Sun, Fuwei Huang, Moye Chen, Zhuo Lin, Hanyu Liu, Qingxin Gui, Qingzhe Han, Yuyang Wen, Huiping Liu, Rongkang Wang, Yaqi Zhang, Hongliang Wei, Chi Chen, You Li, Kechen Fang, Jie Zhou, Yuxuan Li, Guoyang Zeng, Chaojun Xiao, Yankai Lin, Xu Han, Maosong Sun, Zhiyuan Liu, and Yuan Yao. MiniCPM-o 4.5: Towards real-time full-duplex omni-modal interaction. *arXiv preprint arXiv:2604.27393*, 2026.

DeepSeek-AI. DeepSeek-V4: Towards highly efficient million-token context intelligence, 2026.

Alexandre Défossez, Laurent Mazaré, Manu Orsini, Amélie Royer, Patrick Pérez, Hervé Jégou, Edouard Grave, and Neil Zeghidour. Moshi: A speech-text foundation model for real-time dialogue. *arXiv preprint arXiv:2410.00037*, 2024.

Antonio A Ginart, Naveen Kodali, Jason Lee, Caiming Xiong, Silvio Savarese, and John Emmons. Asynchronous tool usage for real-time agents. *arXiv preprint arXiv:2410.21620*, 2024.

GLM-5-Team. GLM-5: from vibe coding to agentic engineering, 2026.

Gonzalo Gonzalez-Pumariega, Leong Su Yean, Neha Sunkara, and Sanjiban Choudhury. Robotouille: An asynchronous planning benchmark for LLM agents. In *International Conference on Learning Representations*, 2025.

Etash Guha, Ryan Marten, Sedrick Keh, Negin Raoof, Georgios Smyrnis, Hritik Bansal, Marianna Nezhurina, Jean Mercat, Trung Vu, Zayne Sprague, et al. OpenThoughts: Data recipes for reasoning models. In *International Conference on Learning Representations*, 2026.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021.

Ronald A. Howard. *Dynamic Probabilistic Systems, Volume II: Semi-Markov and Decision Processes*. Wiley, New York, 1971.

Mehrdad Jazayeri and Michael N. Shadlen. Temporal context calibrates interval timing. *Nature Neuroscience*, 13(8):1020–1026, 2010. doi: 10.1038/nn.2590.

Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1–2):99–134, 1998. doi: 10.1016/S0004-3702(98)00023-X.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, 2024.

Atsumoto Ohashi, Neil Zeghidour, Alexandre Défossez, and Eugene Kharitonov. Multi-faceted interactivity alignment in full-duplex speech models. *arXiv preprint arXiv:2606.11167*, 2026. Accepted to EMNLP 2026, main conference.

Orantqing, Shengpeng Ji, Junlong Tong, Jialong Zuo, Dongjie Fu, Di Cao, Yangzhuo Li, Shangda Wu, Franz, Evan, Theron Veyra, Changhao Pan, Jingyu Lu, Dongchao Yang, Zhifei Xie, Yang Tan, Xiaoyu Shen, Xiaoda Yang, Wenfu Wang, Teddy Sun, Steve Yves, and Zhou Zhao. Multimodal duplex interaction agent. *arXiv preprint arXiv:2609.08977*, 2026.

Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley, New York, 1994. ISBN 9780471619772.

Stuart Russell and Eric Wefald. Principles of metareasoning. *Artificial intelligence*, 49(1-3):361–395, 1991.

A. Semjen, H.-H. Schulze, and D. Vorberg. Timing precision in continuation and synchronization tapping. *Psychological Research*, 63(2):137–147, 2000. doi: 10.1007/PL00008172.

Richard S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3: 9–44, 1988. doi: 10.1007/BF00115009.

Richard S. Sutton, Doina Precup, and Satinder Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1–2):181–211, 1999. doi: 10.1016/S0004-3702(99)00052-1.

Paulo Tabuada. Event-triggered real-time scheduling of stabilizing control tasks. *IEEE Transactions on Automatic Control*, 52(9):1680–1685, 2007. doi: 10.1109/TAC.2007.904277.

Bandhav Veluri, Benjamin N Peloquin, Bokai Yu, Hongyu Gong, and Shyamnath Gollakota. Beyond turn-based interfaces: Synchronous LLMs as full-duplex dialogue agents. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 21390–21402. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.emnlp-main.1192.

J. H. Wearden and Rebecca Farrar. Effects of feedback and calibration on the verbal estimation of the duration of tones. *Acta Psychologica*, 126(1):1–17, 2007. doi: 10.1016/j.actpsy.2006.10.003.

Yule Wen, Yixin Ye, Yanzhe Zhang, Diyi Yang, and Hao Zhu. Real-time reasoning agents in evolving environments. In *International Conference on Learning Representations*, 2026.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992. doi: 10.1007/BF00992696.

Beverly A. Wright, Dean V. Buonomano, Henry W. Mahncke, and Michael M. Merzenich. Learning and generalization of auditory temporal-interval discrimination in humans. *Journal of Neuroscience*, 17(10):3956–3963, 1997. doi: 10.1523/JNEUROSCI.17-10-03956.1997.

Jin Xu, Zhifang Guo, Jinzheng He, Hangrui Hu, Ting He, Shuai Bai, Keqin Chen, Jialin Wang, Yang Fan, Kai Dang, Bin Zhang, Xiong Wang, Yunfei Chu, and Junyang Lin. Qwen2.5-Omni technical report. *arXiv preprint arXiv:2503.20215*, 2025.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023.

Z.ai. GLM-5.2. Hugging Face model card, 2026. URL https://huggingface.co/zai-org/GLM-5.2. Accessed September 25, 2026.

Gengyuan Zhang, Tanveer Hannan, Hermine Kleiner, Beste Aydemir, Xinyu Xie, Jian Lan, Thomas Seidl, Volker Tresp, and Jindong Gu. AViLA: Asynchronous vision-language agent for streaming multimodal data interaction. *arXiv preprint arXiv:2506.18472*, 2025. doi: 10.48550/arXiv.2506.18472.

Yangqing Zheng, Shunqi Mao, Dingxin Zhang, and Weidong Cai. Reflex first, reflect later: Latency-aware embodied LLM agents for dynamic response. *arXiv preprint arXiv:2506.07223*, 2025. Accepted at the CVPR 2025 Embodied AI Workshop.

Shlomo Zilberstein. Using anytime algorithms in intelligent systems. *AI Magazine*, 17(3):73–83, 1996.