TIDE: A Benchmark for Temporal State Misalignment in Language Agents

Anonymous ACL submission

Abstract

Large language model agents are often evaluated in discrete observation-action-feedback loops that implicitly assume observations and prior feedback remain valid when needed. In dynamic and asynchronous environments, however, this assumption can fail as state information changes between observation, action, and feedback. To systematically study this failure mode, we introduce TIDE, a diagnostic benchmark for temporal state misalignment in LLM agents. TIDE contains 711 multiple-choice instances organized into three misalignment types: observation-to-action, action-to-feedback, and feedback-to-observation. Experiments across recent LLMs show that temporal state misalignment remains challenging: models often proceed when additional validation is needed, but may also become overly conservative when the state is already reliable. A temporal-state prompt improves performance on the main benchmark, but does not fully resolve the issue and can bias models toward conservative actions. These results suggest that reliable agent evaluation should explicitly account for state validity under dynamic interaction.

1 Introduction

Recent work has increasingly framed large language models (LLMs) as agents that combine language-based reasoning with tool use, web interaction, software control, and multi-step task execution (Yao et al., 2022; Schick et al., 2023; Zhou et al., 2024; Liu et al., 2024). In these settings, agents observe external states, issue actions, and adapt their behavior based on environmental feedback, enabling capabilities such as planning, tool use, web navigation, and task completion in interactive environments (Schick et al., 2023; Zhou et al., 2024; Koh et al., 2024; Yao et al., 2024).

Most existing evaluations instantiate this interaction pattern as a discrete loop: a model observes the environment state, selects an action based on that observation, and receives feedback after issuing the action. This abstraction has enabled controlled evaluations across tool use, web navigation, software automation, and multi-step decision making (Zhou et al., 2024; Liu et al., 2024; Koh et al., 2024; Xie et al., 2024). However, this abstraction also relies on an assumption of temporal alignment: the observed state is treated as valid when the action is issued or takes effect, and returned feedback is treated as a reliable basis for the next decision.

In dynamic and asynchronous environments, however, this temporal-alignment assumption often fails. The environment may change while a model is generating a response, a tool call is executing, or a user is away between interaction turns. For example, a spreadsheet row identified by position may be reordered before the agent edits it; a print job may have been submitted to a queue but not yet completed; and an inventory record returned yesterday may no longer be valid when the user resumes the task.

We refer to the resulting inconsistency between the state an agent observes, the state in which its action takes effect, and the state implied by feedback as *temporal state misalignment*. This form of misalignment is distinct from standard semantic understanding or tool-selection errors: a model may correctly infer the user intent and choose an action that is reasonable under the original observation, yet still fail because the state required by that action has become stale, incomplete, or insufficient for the current decision. To systematically study this failure mode, we organize temporal state misalignment by its dominant source in the interaction loop: observation-to-action (O2A), action-to-feedback (A2F), and feedback-to-observation (F2O). Figure 1 presents motivating cases of the three forms of temporal state misalignment.

Reliable LLM agents therefore require more

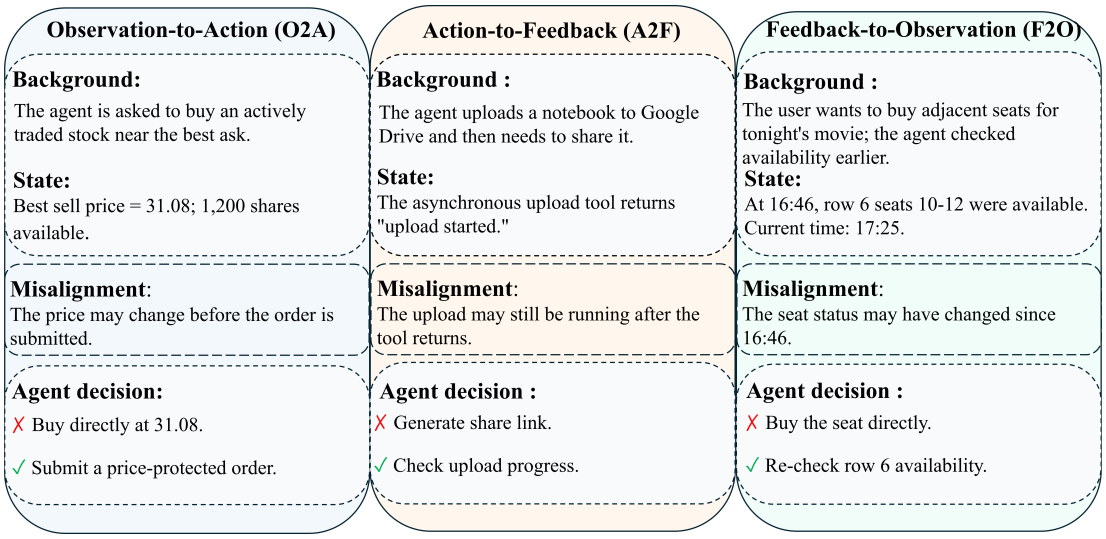

Figure 1: Motivating examples of temporal state misalignment. Across observation-to-action (O2A), action-to-feedback (A2F), and feedback-to-observation (F2O) misalignment, the state available to an agent may differ from the state that determines whether the next action is valid. Each case contrasts a decision that relies on temporally misaligned information with a more robust next action, such as constraining the action to tolerate state changes, checking execution progress, or re-observing stale feedback.

than next-action prediction from a static observation alone; they must also assess whether the available information remains valid for the current decision. We call this capability *temporal validity judgment*: determining whether observations, action outcomes, and prior feedback provide a sufficient basis for proceeding. Under this framing, actions such as waiting, polling, re-observing, verifying state, or recovering from inconsistency are not peripheral fallbacks, but valid decisions when the current state is not yet decision-ready.

To evaluate this capability, we construct TIDE, a controlled diagnostic benchmark for LLM agents. The benchmark contains 711 multiple-choice instances spanning the three misalignment categories introduced above. Each instance presents an interaction history, optional tool schemas and feedback, and candidate next actions; the model must choose the most appropriate next step under the current temporal state. By abstracting each scenario into a next-action decision, TIDE isolates whether the model can choose an action that is temporally justified by the interaction history.

We evaluate a range of recent proprietary and open-weight models on TIDE. The results show that current LLM agents still struggle to determine whether available state information remains valid for action, and their performance varies substantially across misalignment types. Our analyses further show that errors are not explained by a simple preference for either proceeding or being conservative: models often proceed when synchronization is needed, while prompting them to attend to temporal state can improve benchmark accuracy, but may also shift them toward overly conservative actions, such as unnecessary waiting or verification.

Our main contributions are as follows:

- We identify *temporal state misalignment* as a systematic failure mode for LLM agents in dynamic and asynchronous environments, and characterize it through observation-to-action, action-to-feedback, and feedback-to-observation misalignment.
- We introduce TIDE, a controlled diagnostic benchmark with 711 multiple-choice instances for evaluating whether LLM agents can select next actions that are temporally justified by interaction histories.
- We evaluate representative proprietary and open-weight LLMs and show that current agents struggle with temporal validity under interaction, with errors reflecting both premature task continuation and overly conservative synchronization behavior.

2 Related Work

LLM agent evaluation. Recent work has developed LLM agents that combine language-based

reasoning with tool use, web interaction, software control, and multi-step task execution (Yao et al., 2022; Schick et al., 2023). Benchmarks such as AgentBench, WebArena, VisualWebArena, OSWorld, GAIA, Gaia2, and τ-bench evaluate broad agentic competence across tool use, web navigation, computer control, multimodal grounding, dynamic environment handling, and user-tool interaction (Liu et al., 2024; Zhou et al., 2024; Koh et al., 2024; Xie et al., 2024; Mialon et al., 2024; Froger et al., 2026; Yao et al., 2024). While such benchmarks have substantially advanced the evaluation of interactive language agents, their primary focus is broad agentic competence and end-to-end task success. Our work focuses on a complementary dimension: whether the state represented by observations, actions, and feedback is still temporally valid for the next decision.

Temporal reasoning and temporal robustness. Temporal reasoning benchmarks evaluate whether LLMs can reason about event order, duration, temporal arithmetic, and related textual relations (Chu et al., 2024; Wang and Zhao, 2024). Other work studies time-sensitive or dynamic question answering, where answers depend on when the question is asked or require up-to-date information (Zhang et al., 2021; Kasai et al., 2022; Vu et al., 2023). Temporal change has also been studied as a robustness issue under calendar-time distribution shift (Luu et al., 2022). Beyond these conventional settings, recent work has begun to examine whether LLMs and LLM agents account for elapsed time in interactive contexts. Discrete Minds studies whether LLMs perceive the passage of wall-clock time and adapt their behavior under urgency or dynamic time pressure (Wang et al., 2025). Temporally Blind investigates how elapsed time between user-agent messages affects tool-use decisions, showing that agents often fail to account for stale context in multi-turn interactions (Cheng et al., 2026). Concurrent work on stale agent memory further studies whether agents can detect when previously valid memories are invalidated by later evidence (Chao et al., 2026). These works primarily evaluate temporal understanding, knowledge freshness, elapsed-time awareness, tool-use timing, or memory invalidation, whereas our benchmark evaluates whether temporal changes affect the validity of observations, action outcomes, and feedback for next-action selection.

Dynamic and asynchronous interaction. Several recent lines of work have begun to move agent evaluation beyond static and fully synchronous interaction. Asynchronous and embodied benchmarks evaluate agents in settings where actions may take time, overlap with future decisions, or interact with changing world states (González-Pumariega et al., 2025). Recent formal work further argues that fixed action–observation steps hide timing issues such as deliberation latency, delayed feedback, and persistent actions (Li et al., 2026). Related work on delayed decision making also studies how actions and observations can become misaligned under delay (Wang et al., 2024). Gaia2 (Froger et al., 2026) provides a closely related evaluation setting: its Agent Runtime Environment advances simulated time according to the duration of tool calls and model responses, allowing the environment to evolve even while the agent is generating an action. Our work differs from these efforts in evaluation granularity: rather than measuring end-to-end task completion in a live dynamic environment or proposing a new interaction formalism, we abstract each scenario into a controlled next-action decision and isolate temporal state misalignment across observation-to-action, action-to-feedback, and feedback-to-observation settings.

3 Dataset Construction

3.1 Task Formulation

We characterize each interaction turn using three event times corresponding to observation, action issuance, and feedback. Specifically, t_0 denotes the time at which the agent receives an observation or contextual information, t_1 denotes the time at which the agent completes reasoning and issues an action or recommendation, and t_2 denotes the time at which the agent receives the next observable feedback from a tool or the environment, which may indicate submission, progress, completion, or an updated state. In multi-turn tasks, feedback from one turn becomes part of the context for the next turn, forming a repeated interaction process.

Temporal state misalignment can arise between these stages. We therefore construct samples around three types of misalignment: *Observation-to-Action Misalignment* (O2A), where an observed state may change before the agent's action is issued or takes effect; *Action-to-Feedback Misalignment* (A2F), where an issued action may take

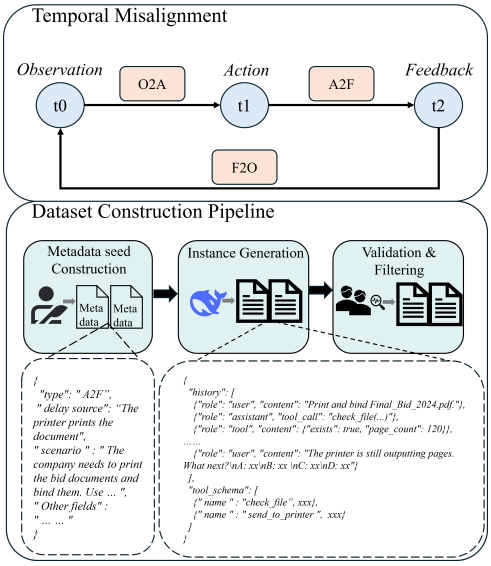

Figure 2: Overview of the TIDE construction pipeline. The upper panel illustrates the temporal structure of an agent interaction and the three corresponding misalignment types. The lower panel shows the construction pipeline from metadata seeds to LLM-generated instances and manual validation. Subtype-specific metadata fields and representative examples are provided in Appendix A.

time to execute and the available feedback may not yet reflect its resulting state; and *Feedback-to-Observation Misalignment* (F2O), where feedback from a previous turn may become stale or insufficient by the time it is used as context for a later decision.

Given a dialogue history, optional tool schemas, and a multiple-choice next-action question, the model is required to select the next action best supported by the current temporal state. The dialogue history follows an API-style interaction format and may include user requests, assistant tool calls and tool responses.

3.2 Controlled Instance Construction

Step 1: Metadata Seed Construction. We first manually construct metadata seeds according to the three misalignment categories defined above. The seeds are derived from manually curated scenarios involving representative digital services, human-facing tools, and physical processes in which state information may change or lose validity over time. For O2A misalignment, we focus on cases where the observed state may become unreliable before the action is issued or takes effect, such as changing resource availability, moving object references, or updated environment status. For A2F misalignment, we cover asynchronous action execution in both physical and digital settings, where an issued action may require additional time before its resulting state is established. For F2O misalignment, we focus on cross-turn interaction gaps, where feedback obtained in an earlier turn is reused after a time interval and may no longer support the later decision.

This seed construction process results in 84 O2A, 216 A2F, and 120 F2O metadata seeds. Each metadata seed serves as a lightweight specification for instance generation. It specifies the misalignment category, scenario background, source of temporal uncertainty, relevant state evidence when applicable, and the intended decision tension. For example, an A2F seed may specify a printing task in which a queued status should not be treated as task completion. These seeds constrain the temporal conflict and target decision pattern before LLM-based instance generation.

Step 2: Instance Generation. We use LLMs to instantiate each metadata seed into one or two complete multiple-choice samples under seed-level constraints. Each generated sample contains an API-style dialogue context, optional tool schemas and tool interactions, and a final decision question with candidate actions. The dialogue context is generated under the constraints specified by the seed, and may include user requests, assistant tool calls, tool responses, explicit or implicit time cues, status fields, and follow-up user queries. The seed constrains the intended temporal conflict and target temporally valid decision pattern, which is realized as one of the candidate actions in the final question. The remaining options are generated as plausible distractors that correspond to common temporal failure modes, such as relying on stale observations, prematurely advancing the task, mistaking submission for completion, ignoring cross-turn time gaps, or executing an action that no longer matches the current state.

Step 3: Validation and Filtering. After LLM-based generation and refinement, we manually review the retained candidate samples before forming the final dataset. The review is conducted by two reviewers with graduate-level computer science training. For each reviewed sample, we check whether the dialogue and tool interactions follow the expected format, whether tool calls are consistent with the provided schemas and responses, and whether the temporal misalignment

Misalignment Type	# Instances	Proportion
O2A	132	18.6%
A2F	378	53.2%
F2O	201	28.3%
Total	711	100.0%

Table 1: Dataset statistics by temporal misalignment type.

matches the intended category. We further verify that the correct answer is uniquely supported by the dialogue history, available state evidence, optional tool outputs, and explicitly provided task context, rather than relying primarily on unstated assumptions or the generation model's implicit judgment. We also check that distractors are plausible but do not make the answer ambiguous. Samples with obvious lexical cues that reveal the target behavior are removed, such as those directly mentioning that the agent should wait, poll, or recheck when such wording is not justified by the dialogue context. Samples with severe temporal inconsistency, ambiguous answers, invalid tool use, or mismatched misalignment types are discarded; samples with minor wording or formatting issues are manually revised.

Additional construction details are provided in Appendix A.

3.3 Dataset Statistics

The final benchmark contains 711 multiple-choice instances. As the benchmark is organized around temporal misalignment types, Table 1 reports the primary dataset statistics at the O2A/A2F/F2O level.

We use scenario families to instantiate the three misalignment types, but do not treat them as separate evaluation axes. Representative scenarios include resource availability, asynchronous job execution, cross-turn booking or purchasing, collaborative editing, and monitoring.

4 Experiments

4.1 Experimental Setup

In our core experiments, we evaluate representative recent language models from multiple model families (DeepSeek-AI, 2026; Moonshot AI, 2026; Z.AI, 2026; OpenAI, 2026; MiniMax, 2026; Qwen Team, 2026). We additionally include Qwen3-8B as a representative smaller open-weight reference model (Qwen Team, 2025). Each model is given the full dialogue history, available tool schemas when applicable, and the final multiple-choice decision question, and is required to output a single option. Unless otherwise constrained by a provider-specific API, we use temperature 0. Because models expose heterogeneous thinking controls, we keep the primary comparison under non-thinking or provider-default settings: for models with explicit thinking controls, we disable thinking unless otherwise noted, and for APIs without an equivalent control, we use the provider default. Reasoning-mode sensitivity results for supported models are reported in Appendix B.4. We report accuracy overall and by temporal misalignment type. Predictions that cannot be mapped to a unique option label are counted as incorrect.

4.2 Main Results

Table 2 reports model performance across the three temporal misalignment types. Several observations emerge.

Current models leave substantial room for improvement. Although GPT-5.5 achieves the strongest overall performance at 79.7%, most evaluated models fall within the 50–63% range, while the smaller Qwen3-8B reference model obtains 42.5%. These results indicate that temporal validity judgment remains challenging for current language agents.

F2O misalignment is consistently the most challenging category. All evaluated models obtain their lowest accuracy on F2O, with scores ranging from 29.9% to 63.7%. This suggests that models frequently struggle to determine whether feedback from a previous turn still provides valid evidence for a later decision. Such failures are particularly relevant for multi-turn agents, where stale prices, inventories, reservations, or execution results may be carried forward as if they were current.

Aggregate accuracy masks type-specific weaknesses. Performance varies substantially across misalignment types. For example, GLM-5.1 reaches 77.3% on O2A but drops to 49.3% on F2O, while MiniMax-M2.7 performs better on A2F than on F2O by more than 30 points. These gaps show that a model with reasonable aggregate performance can still fail on specific forms of temporal misalignment. This variation motivates reporting performance by misalignment type rather than only as an aggregate score.

Model	O2A	A2F	F2O	Overall
Qwen3-8B	50.8	46.3	29.9	42.5
DeepSeek V4 Flash	54.5	55.8	42.8	51.9
DeepSeek V4 Pro	59.8	63.8	36.3	55.3
GLM-5.1	77.3	62.7	49.3	61.6
Kimi-K2.6	65.9	61.4	47.8	58.4
MiniMax-M2.7	61.4	65.9	33.8	56.0
Qwen3.6-Max-Preview	68.9	66.7	51.7	62.9
GPT-5.5	**80.3**	**88.1**	**63.7**	**79.7**

Table 2: Accuracy on the main benchmark by temporal misalignment type. Accuracy is reported in percentages. O2A, A2F, and F2O denote Observation-to-Action, Action-to-Feedback, and Feedback-to-Observation misalignment, respectively. Best results are bolded and second-best results are underlined.

4.3 Analysis

Policy annotation. To analyze model behavior beyond accuracy, we use LLMs to annotate each candidate action with one of four policy types: *proceed*, *conservative*, *recover*, and *other*. A *proceed* action directly advances the task under the available state; a *conservative* action explicitly synchronizes, refreshes, waits for, polls, or validates state before proceeding; a *recover* action handles a failed, invalid, unsafe, unavailable, or not-ready state through repair, retry, rollback, or fallback compensation; and *other* denotes irrelevant, task-changing, or redundant actions. Importantly, *proceed* is not necessarily erroneous: as shown in Table 3, some gold answers directly advance the task when the available state already supports the decision. Detailed policy annotation is reported in Appendix B.1.

4.3.1 Policy Types of Incorrect Predictions

Table 4 shows the policy distribution among parsed incorrect predictions. Across models, errors are dominated by *proceed* actions: even for GPT-5.5, 80.8% of parsed errors correspond to directly advancing the task. This suggests that many failures are not arbitrary option confusions, but cases where models act on an observation, feedback, or intermediate result without performing the synchronization or validation required by the temporal state. The dominant failure mode is therefore under-synchronization.

4.3.2 Temporal Insensitivity Control

We further construct a temporal-insensitivity control set to diagnose whether models can avoid unnecessary conservative actions when the relevant state is already reliable. The set is derived from benchmark instances by neutralizing the temporal uncertainty while preserving plausible conservative distractors; detailed construction rules are provided in Appendix B.2. The resulting set contains 100 instances, with 20 O2A, 40 A2F, and 40 F2O controls. The gold policy distribution is intentionally different from the main benchmark: 92 gold options are labeled *proceed*, while 8 are labeled *conservative*. Thus, models that adopt a uniformly conservative strategy, for example by re-checking or waiting even when the state is already reliable, are expected to incur over-conservative errors on this set.

We report three metrics on this control set: accuracy, conservative action rate, and over-conservative error rate (OCER), defined as

$$\mathrm{OCER} = \frac{N_{\mathrm{wrong,cons}}}{N_{\mathrm{wrong}}} \times 100\%,$$

where $N_{\mathrm{wrong,cons}}$ is the number of incorrect predictions whose selected option is conservative and N_{wrong} is the total number of incorrect predictions mapped-to-option. Table 5 shows that models do not achieve perfect performance when temporal uncertainty is neutralized. More importantly, their errors are often over-conservative, with OCER ranging from 50.0% to 88.0%. This complements the main-benchmark analysis: on the original benchmark, models often fail by directly proceeding when synchronization is needed; on the control set, they often fail by continuing to synchronize or re-check even when the context already establishes that the state is reliable. Together, these results suggest that the challenge is not merely choosing a conservative template, but judging whether the available temporal state is valid enough to support direct progression.

4.3.3 Effect of Temporal-State Prompting

We further evaluate whether a direct temporal-state instruction can mitigate temporal-validity er-

Dataset	Proceed	Conservative	Recover	Other
Main benchmark (N=711)	121 (17.0%)	582 (81.9%)	8 (1.1%)	0 (0.0%)
Temporal-insensitivity control (N=100)	92 (92.0%)	8 (8.0%)	0 (0.0%)	0 (0.0%)

Table 3: Policy-type distribution of correct answers in the main benchmark and the temporal-insensitivity control set. Percentages are computed within each dataset. Detailed category-wise distributions are reported in Table 8.

Model	# Wrong	Proceed	Conservative	Recover	Other
Qwen3-8B	401	85.8	5.5	0.7	8.0
DeepSeek V4 Flash	324	79.6	7.4	1.5	11.4
DeepSeek V4 Pro	284	80.3	7.4	1.4	10.9
GLM-5.1	214	72.4	10.7	2.3	14.5
Kimi-K2.6	295	83.4	7.5	1.4	7.8
MiniMax-M2.7	265	80.8	8.7	0.4	10.2
Qwen3.6-Max-Preview	264	84.8	6.4	1.1	7.6
GPT-5.5	130	80.8	5.4	1.5	12.3

Table 4: Policy distribution of parsed incorrect predictions on the main benchmark. Unparsed responses are counted as incorrect for accuracy but excluded from this policy table because they cannot be mapped to a candidate action.

Model	Acc.	Conservative	OCER
Qwen3-8B	81.0	22.0	78.9
DeepSeek V4 Flash	73.0	21.0	62.5
DeepSeek V4 Pro	66.0	21.0	57.1
GLM-5.1	63.0	32.0	78.1
Kimi-K2.6	73.0	29.0	81.5
MiniMax-M2.7	65.0	24.0	50.0
Qwen3.6-Max-Preview	75.0	27.0	80.0
GPT-5.5	73.0	29.0	88.0

Table 5: Results on the temporal-insensitivity control set after blind output extraction. **Conservative** denotes the overall conservative-action selection rate.

rors and how it affects policy selection. The prompt asks the model to judge whether the current observation, prior action outcome, and feedback are temporally reliable before selecting an option. We prepend the temporal-state instruction to the system prompt before each query; the exact prompt is provided in Appendix B.3.

Table 6 shows a clear but incomplete benefit. Temporal-state prompting improves main-benchmark accuracy for every evaluated model, with gains ranging from 1.5 to 9.7 percentage points. This suggests that explicit instructions can help models attend to temporal validity cues that they otherwise miss.

However, the same prompt also exposes a calibration problem. On the temporal-insensitivity control set, where the context establishes that the relevant state is stable, completed, or still valid, accuracy decreases for seven of eight models and remains unchanged for DeepSeek V4 Pro. Table 7 shows that the prompt often reduces *proceed* choices or increases *conservative* choices on the control set, although the Qwen3-8B result shows that the shift is not uniform across all models. Thus, prompting does not simply make models better at temporal validity judgment; it changes policy selection in ways that can help when temporal uncertainty is real but can produce calibration errors when the state is already reliable.

These results suggest that the benchmark is useful for diagnosing temporal-validity behavior, and that prompt-based interventions alone do not fully resolve the issue.

5 Conclusion

We introduce TIDE, a benchmark for temporal state misalignment in LLM agents. Our experiments suggest that the central challenge is not simply choosing a conservative action, but judging whether the information currently available is still temporally valid for the next decision. This challenge is especially pronounced in F2O settings, where models often over-trust prior feedback, highlighting a broader weakness in maintaining the validity of past state claims across interaction turns. Our analysis further suggests that temporal-state prompting can partially improve performance, but may also shift models toward overly conservative behavior. These findings indicate that agent evaluation should go beyond tool use and end-to-end task completion under syn-

Model	Main	Main+P	Δ	Ctrl.	Ctrl.+P	Δ
Qwen3-8B	42.5	44.3	+1.8	81.0	79.0	-2.0
DeepSeek V4 Flash	51.9	55.4	+3.5	73.0	63.0	-10.0
DeepSeek V4 Pro	55.3	65.0	+9.7	66.0	66.0	+0.0
GLM-5.1	61.6	63.9	+2.3	63.0	48.0	-15.0
Kimi-K2.6	58.4	62.7	+4.3	73.0	68.0	-5.0
MiniMax-M2.7	56.0	64.1	+8.1	65.0	64.0	-1.0
Qwen3.6-Max-Preview	62.9	64.4	+1.5	75.0	72.0	-3.0
GPT-5.5	79.7	84.2	+4.5	73.0	67.0	-6.0

Table 6: Effect of temporal-state prompting. Main and Ctrl. report overall accuracy on the main benchmark and temporal-insensitivity control set. Main+P and Ctrl.+P use the temporal-state prompt. Δ denotes prompt minus base accuracy.

Model	Main ΔProceed	Main ΔCons.	Ctrl. ΔProceed	Ctrl. ΔCons.
Qwen3-8B	+0.1	+1.1	+3.0	-4.0
DeepSeek V4 Flash	-9.2	+1.7	-9.0	+5.0
DeepSeek V4 Pro	-8.6	+9.1	-3.0	+3.0
GLM-5.1	-1.0	-0.2	-9.0	+3.0
Kimi-K2.6	-5.7	+4.2	-2.0	-1.0
MiniMax-M2.7	-5.7	+6.8	-8.0	+2.0
Qwen3.6-Max-Preview	-3.1	+3.0	-6.0	+5.0
GPT-5.5	-10.4	+4.7	-4.0	+4.0

Table 7: Prompt-induced change in policy-selection rates. Values are percentage-point differences between the temporal-state prompt and the base prompt. Cons. denotes conservative-action selection.

chronized interaction, and should explicitly assess whether agents can judge temporal validity under asynchronous and dynamic conditions.

Limitations

TIDE is designed as a controlled diagnostic benchmark rather than a full end-to-end agent environment. Each instance is framed as a multiple-choice next-action decision, which allows us to isolate temporal state misalignment but does not capture all challenges of long-horizon exploration, low-level tool execution, or interactive recovery. In addition, although the benchmark covers three misalignment types and a range of representative scenarios, it does not exhaust the diversity of dynamic real-world environments. Finally, the dataset is constructed from metadata seeds with LLM-based instantiation and manual validation, so residual biases in scenario selection, wording, and candidate-action design may remain.

References

Hanxiang Chao, Yihan Bai, Rui Sheng, Tianle Li, and Yushi Sun. 2026. Stale: Can llm agents know when their memories are no longer valid? *arXiv preprint arXiv:2605.06527*.

Yize Cheng, Arshia Soltani Moakhar, Chenrui Fan, Parsa Hosseini, Kazem Faghih, Zahra Sodagar, Wenxiao Wang, and Soheil Feizi. 2026. Your LLM agents are temporally blind: The misalignment between tool use decisions and human time perception. *arXiv preprint arXiv:2510.23853*.

Zheng Chu, Jingchang Chen, Qianglong Chen, Weijiang Yu, Haotian Wang, Ming Liu, and Bing Qin. 2024. Timebench: A comprehensive evaluation of temporal reasoning abilities in large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1204–1228.

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

Romain Froger, Pierre Andrews, Matteo Bettini, Amar Budhiraja, Ricardo Silveira Cabral, Virginie Do, Emilien Garreau, Jean-Baptiste Gaya, Hugo Laurençon, Maxime Lecanu, and 1 others. 2026. GAIA2: Benchmarking LLM agents on dynamic and asynchronous environments. *arXiv preprint arXiv:2602.11964*.

Marcos González-Pumariega, Daniel Fried, and Yonatan Bisk. 2025. Robotouille: An asynchronous planning benchmark for LLM agents. *arXiv preprint arXiv:2502.05227*.

Jungo Kasai, Keisuke Sakaguchi, Ronan Le Bras, Akari Asai, Dragomir Radev, Noah A. Smith, and

Yejin Choi. 2022. RealTimeQA: What's the answer right now? *arXiv preprint arXiv:2207.13332*.

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Russ Salakhutdinov, and Daniel Fried. 2024. Visualwebarena: Evaluating multimodal agents on realistic visual web tasks. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 881–905.

Jialian Li, Yuchen Cao, Junhong Liu, Weiran Guo, Xutao Wang, Jiaming Song, Jiahao Zhang, and Jie Chen. 2026. Engagement process: Rethinking the temporal interface of action and observation. *Preprint*, arXiv:2605.11484.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, and 1 others. 2024. Agentbench: Evaluating llms as agents. In *International Conference on Learning Representations*, volume 2024, pages 52989–53046.

Kelvin Luu, Daniel Khashabi, Suchin Gururangan, Karishma Mandyam, and Noah A. Smith. 2022. Time waits for no one! analysis and challenges of temporal misalignment. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5944–5958.

Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, Yann LeCun, and Thomas Scialom. 2024. Gaia: a benchmark for general ai assistants. In *International Conference on Learning Representations*, volume 2024, pages 9025–9049.

MiniMax. 2026. MiniMax M2.7: Early echoes of self-evolution. https://www.minimax.io/news/minimax-m27-en.

Moonshot AI. 2026. Kimi K2.6: Advancing open-source coding. https://www.kimi.com/blog/kimi-k2-6.

OpenAI. 2026. GPT-5.5 Instant system card. https://openai.com/index/gpt-5-5-instant-system-card/.

Qwen Team. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Qwen Team. 2026. Qwen3.6-max-preview. https://qwen.ai/blog?id=qwen3.6-max-preview.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *Advances in neural information processing systems*, 36:68539–68551.

Tu Vu, Mohit Iyyer, Xuezhi Wang, Noah Constant, Jerry Wei, Jason Wei, Chris Tar, Yun-Hsuan Sung, Denny Zhou, Quoc Le, and 1 others. 2023. FreshLLMs: Refreshing large language models with search engine augmentation. *arXiv preprint arXiv:2310.03214*.

Bin Wang, Yuxuan Li, Yi Zhang, Xiang Li, Jianye Liu, and Yang Liu. 2024. Addressing signal delay in deep reinforcement learning. In *International Conference on Learning Representations*.

Minghan Wang, Ye Bai, Thuy-Trang Vu, Ehsan Shareghi, and Gholamreza Haffari. 2025. Discrete minds in a continuous world: Do language models know time passes? In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 18703–18729.

Yuqing Wang and Yun Zhao. 2024. Tram: Benchmarking temporal reasoning for large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 6389–6415.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh J Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, and 1 others. 2024. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *Advances in Neural Information Processing Systems*, 37:52040–52094.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. 2024. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv preprint arXiv:2406.12045*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

Z.AI. 2026. GLM-5.1. https://docs.z.ai/guides/llm/glm-5.1.

Xikun Zhang, Yixin Ma, Lin Wang, Jiawei Wang, Wei Lu, and Mohit Iyyer. 2021. SituatedQA: Incorporating extra-linguistic contexts into QA. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, and 1 others. 2024. Webarena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*, volume 2024, pages 15585–15606.

A Additional Details of Dataset Construction

A.1 Metadata Seeds Design

We construct TIDE from metadata seeds, but the seed schema is intentionally lightweight and

category-dependent. This reflects the fact that different misalignment types require different control information. For example, A2F seeds are organized around a delay source and a concrete delay scenario, while F2O seeds additionally specify the previous-turn result, the cross-turn gap, the new user request, the intended strategy, and common bad answer patterns. Additional fields are used for variants that require explicit control over temporal gaps, as described in Section A.2. Across categories, the common role of a seed is to constrain the scenario, the source of temporal uncertainty, and the decision tension before LLM-based instance generation.

A.2 Category-Specific Seed Design

For A2F, we first organize seeds by the source of action-execution delay. The seed set distinguishes physical and digital settings: physical settings involve non-instantaneous processes such as actuation, movement, device operation, or other physical execution, while digital settings involve delayed service execution, remote processing, scheduling, or queueing. One A2F subtype concerns motion-compensation tasks, where the agent must account for the time between issuing a physical command and the command taking effect. For example, an instance may show two schematic frames of an object moving on a conveyor belt and ask where a robotic arm should aim if it starts moving now. The temporally valid action is to target the predicted action-time position rather than the current position. These motion-compensation answers are treated as *proceed* actions in our policy annotation because they directly execute the task rather than synchronize or verify state.

For F2O, we construct seeds around cross-turn interaction gaps and use two surface realizations of the gap: a natural-language variant and a timestamp-based variant. In these cases, feedback returned in an earlier turn is carried into a later decision point and may no longer be sufficient for the current action. In the natural-language variant, the gap is expressed through the user's wording or interaction context. In the timestamp variant, the previous result time and current request time are explicit fields in the seed; the script checks that the current timestamp is later than the previous timestamp, computes the elapsed interval, and inserts formatted timestamps into the dialogue.

For O2A, we focus on cases where information observed by the agent may become unreliable before the action is issued or takes effect. Representative cases include resource availability, changing prices or inventory, moving targets, unstable screen or table references, delayed metrics, and situations where a stable identifier or lock is needed.

A.3 Instance generation details.

We use DeepSeek[1] to instantiate metadata seeds into complete multiple-choice instances. For A2F, the generation prompt provides the delay source, scenario description, and few-shot examples, and asks the model to produce one or two instances containing a dialogue history, tool schemas when needed, a final question, candidate choices, and an answer. The generated content is then normalized into the common instance fields: `history`, `tools`, and `ans`. For F2O and O2A, later generation scripts directly require the same common fields. Across all categories, the prompts require the dialogue to follow an API-style interaction format, include objective tool responses when tools are used, end with the final user decision question, and avoid directly revealing the target temporal behavior.

A.4 Automatic Checks and Manual Filtering

The generation scripts apply basic structural checks before retaining candidates. These checks include whether the output contains the expected fields, whether the answer is a single option letter, whether the dialogue ends with a user message containing candidate options, whether tool calls have corresponding tool responses and schemas, and whether the text contains obvious lexical leakage. For timestamp-based F2O data, the script additionally validates timestamp order, checks the computed elapsed interval, requires the previous and current timestamps to appear in the dialogue, and verifies that the final user message begins with the current timestamp. Some F2O generation scripts also include an LLM-based review stage, which checks answer inferability, cross-turn gap realization, dependence on the current state, uniqueness of the best answer, plausibility of distractors, naturalness, tool consistency, and agreement with the seed strategy.

After LLM-based generation and automatic checking, we manually review the retained candidate samples before forming the final benchmark. The manual review checks whether the intended

[1] https://platform.deepseek.com/

misalignment category is preserved, whether the correct option is inferable from the dialogue history, tool feedback, timestamps, or explicitly provided task context, and whether distractors are plausible without making the answer ambiguous. Samples with severe temporal inconsistency, ambiguous answer choices, invalid tool use, obvious leakage cues, or mismatched misalignment categories are removed. Samples with minor wording, formatting, or tool-schema issues are manually revised.

A.5 Representative Examples

We provide shortened examples from the final benchmark. For brevity, we abbreviate long dialogue text with ellipses and retain only the tool names and brief descriptions, omitting the full parameter schema.

O2A example. This example tests whether the model can recognize that an observed market state may become unreliable before the corresponding order takes effect.


```
{
  "category": "O2A",
  "history": [
    {
  "role": "user",
  "content": [
    {
      "type": "text",
      "text": "The current order-book snapshot shows a best ask
      ↪  price of 31.08 with 1,200 shares available at that
      ↪  level. The stock is actively traded. I want to buy
      ↪  near the best ask while avoiding substantial
      ↪  slippage. What should I do?\nA. Buy at 31.08\nB. Buy
      ↪  at the second-best ask price\nC. Submit a
      ↪  price-protected order\nD. Give up the purchase"
    }
  ]
}
  ],
  "tools": [],
  "ans": "C"
}
```


A2F example. This example tests whether the model can distinguish action submission from action completion before relying on the resulting state.


```
{
  "category": "A2F",
  "history": [
    {"role": "user", "content": "Upload \"Llama
    ↪  Finetune.ipynb\" to Google Drive and generate a share
    ↪  link for Xiaoming."},
    {"role": "assistant", "tool_calls": [{"name":
    ↪  "check_file_exists", "arguments": {"file_path": "Llama
    ↪  Finetune.ipynb"}}]},
    {"role": "tool", "content": {"exists": true}},
    {"role": "assistant", "tool_calls": [{"name":
    ↪  "async_upload_to_drive", "arguments": {"file_path":
    ↪  "Llama Finetune.ipynb", "destination": "Google
    ↪  Drive"}}]},
    {"role": "tool", "content": {"status": "upload_started",
    ↪  "message": "File upload initiated successfully."}},
    {"role": "assistant", "content": "The upload task has
    ↪  started."},
    {"role": "user", "content": "Great! If I want to send the
    ↪  link to Xiaoming, what is the best next step?\nA.
    ↪  Generate a shareable link\nB. Check whether the file
    ↪  still exists locally\nC. Restart the upload process\nD.
    ↪  Check the upload progress"}
  ],
  "tools": [
  {
    "name": "check_file_exists",
    "description": "Check whether a file exists at the
    ↪  specified path."
  },
  {
    "name": "async_upload_to_drive",
    "description": "Asynchronously upload a file to a cloud
    ↪  storage service."
  },
  {
    "name": "generate_shareable_link",
    "description": "Generate a shareable link for a file in
    ↪  cloud storage."
  }
],
  "ans": "D"
}
```


A2F motion-compensation example. For readability, we display the user message directly below; in the benchmark instance, the same content is stored as the user message in the `history` field. The example tests whether the agent accounts for the delay between issuing a physical action and the action taking effect. Although the tray is currently aligned with marker A, it is moving rightward; therefore, the temporally valid release point is B.

> The two frames below are shown in Markdown/ASCII format. `edge` denotes the conveyor boundary, and `upper/center/lower` denote the three rows inside the conveyor; x increases from left to right.
>
> Notation:
>
> - `R`: the robotic arm, positioned beside the conveyor, holding the item to be placed;
> - `T`: the tray;
> - `A/B/C/D`: candidate marker points in the second frame;
> - `A(T)` means that candidate point A overlaps with the current position of the tray;
> - Task objective: place the item held by the robotic arm into the moving tray;
> - Physical setting: the arm takes a short time to contact or affect the target after starting the action.

```
Frame 1: only the initial position of the target object is
↪  shown
robot :       R
edge  :
↪  ---------------------------------------------------------------
upper :
center:         T
lower :
edge  :
↪  ---------------------------------------------------------------

Frame 2: the current position of the target object and
↪  candidate marker points are shown
robot :       R
edge  :
↪  ---------------------------------------------------------------
upper :                       D
center:       C          A(T)       B
```

```
lower :
edge  :
↪ ------------------------------------------------------------
```

Question: From Frame 1 to Frame 2, the tray moves to the right. The task is to place the item held by the robotic arm into the moving tray. If the robotic arm starts the placement action now, which marker in Frame 2 should it target so that the item lands in the moving tray?

A. A B. B C. C D. D

Answer: B

F2O example. This example tests whether the model can recognize that seat availability returned in a previous turn may no longer be sufficient for a later booking decision.


```
{
  "category": "F2O",
  "history": [
    {
      "role": "user",
      "content": [
        {
          "type": "text",
          "text": "Could you check which exam seats are
          ↪  currently available at the test center?"
        }
      ]
    },
    {
      "role": "assistant",
      "tool_calls": [
        {
          "id": "call_seat_001",
          "type": "function",
          "function": {
            "name": "query_seat_availability",
            "arguments": "{\"center\":\"Test Center\"}"
          }
        }
      ]
    },
    {
      "role": "tool",
      "content": [
        {
          "type": "text",
          "text": "{\"center\":\"Test
          ↪  Center\",\"query_time\":\"2026-05-04
          ↪  10:30:00\",\"available_seats\":[\"A17\"],\"occu⌋
          ↪  pied_seats\":[\"A18\"]}"
        }
      ],
      "tool_call_id": "call_seat_001"
    },
    {
      "role": "assistant",
      "content": "A17 is currently available, and A18 is
      ↪  occupied."
    },
    {
      "role": "user",
      "content": [
        {
          "type": "text",
          "text": "Yesterday A17 was open. Can I select it
          ↪  directly when I register today?\nA. Select A17
          ↪  directly\nB. Select A18\nC. Check the exam
          ↪  session first\nD. Check the current availability
          ↪  of A17"
        }
      ]
    }
  ],
  "tools": [
    {
      "name": "query_seat_availability",
      "description": "Query the availability of exam seats at a
      ↪  specified test center."
    }
  ],
  "ans": "D"
}
```


F2O timestamp example. This example uses explicit timestamps to represent the cross-turn gap, requiring the model to judge whether the earlier feedback is still fresh enough for the current decision.


```
{
  "category": "F2O-timestamp",
  "history": [
    {
      "role": "user",
      "content": [
        {
          "type": "text",
          "text": "[2025-04-10 16:45:50 UTC] I want to watch
          ↪  Star Trek tonight. Could you check whether there
          ↪  are any good adjacent seats left?"
        }
      ]
    },
    {
      "role": "assistant",
      "tool_calls": [
        {
          "id": "call_seats_001",
          "type": "function",
          "function": {
            "name": "check_cinema_seats",
            "arguments": "{\"movie\":\"Star
            ↪  Trek\",\"date\":\"2025-04-10\"}"
          }
        }
      ]
    },
    {
      "role": "tool",
      "content": [
        {
          "type": "text",
          "text": "[2025-04-10 16:46:00 UTC] {\"movie\":\"Star
          ↪  Trek\",\"date\":\"2025-04-10\",\"screen\":\"Hall
          ↪  7 19:30\",\"seats\":[{\"row\":6,\"seat_numbers\⌋
          ↪  ":[10,11,12],\"status\":\"available\",\"is_cont⌋
          ↪  iguous\":true}]}"
        }
      ],
      "tool_call_id": "call_seats_001"
    },
    {
      "role": "assistant",
      "content": "[2025-04-10 16:46:08 UTC] Seats 10, 11, and
      ↪  12 in row 6 are adjacent and available."
    },
    {
      "role": "user",
      "content": [
        {
          "type": "text",
          "text": "[2025-04-10 17:25:00 UTC] I am ready to buy
          ↪  the middle seat in row 6. What should I do?\nA.
          ↪  Buy the seat directly\nB. Check the row 6 seat
          ↪  status\nC. Buy a first-row seat\nD. Check the
          ↪  cast list"
        }
      ]
    }
  ],
  "tools": [
    {
      "name": "check_cinema_seats",
      "description": "Query cinema-seat availability for a
      ↪  specified movie and date."
    }
  ],
  "ans": "B"
}
```


B Additional Experimental Details

B.1 Policy Annotation

We use the four policy types defined in Section 4.3: *proceed*, *conservative*, *recover*, and *other*. The

annotation is performed with LLM assistance under a predefined codebook that specifies label definitions, boundary cases, and tie-breaking rules. Each candidate option is labeled by its decision policy rather than by whether it is correct. To check annotation quality, we manually inspect a subset of labels, with particular attention to low-confidence or boundary cases.

B.2 Temporal-Insensitivity Control Set Construction

We construct the temporal-insensitivity control set to test whether models over-select conservative actions when temporal cues are present but synchronization is not actually required. The control set is derived from benchmark instances, but the context is rewritten so that the relevant state is sufficiently reliable for direct progression at the decision point. Conservative options such as re-observation, waiting, polling, or verification are still kept as plausible distractors.

B.2.1 Category-Specific Control Design

For *O2A* controls, we edit the context to establish that the observed state remains valid through the next action window. Typical edits include adding an active hold, a lock, a stable object identifier, a version check, or an explicit validity window. For *A2F* controls, we replace uncertain execution feedback such as submitted, queued, or in progress with feedback indicating that the action has completed and that the resulting output is available. For *F2O* controls, we ensure that earlier feedback is still usable at the current decision point, for example by keeping the current request within a stated validity window, preserving an active reservation or hold, or adding confirmation that the prior result has not expired.

B.2.2 Construction and Validation

The control instances preserve the original task structure and candidate-action style, but remove the need for explicit synchronization. We do not add surface instructions such as "wait," "recheck," or "proceed" to the final user query; instead, the model must infer state reliability from the dialogue history and tool feedback. We validate each control instance for structural consistency, tool-call consistency, temporal coherence, answer uniqueness, and absence of lexical leakage. Instances that violate the intended temporal relation, make multiple options valid, or expose the answer through surface wording are revised or discarded.

B.3 Temporal-State Prompt

The prompt used in the temporal-state prompting experiment is:

> You are a helpful AI assistant. Your task is to answer a multiple-choice question by replying with A, B, C, or D according to the options provided by the user.
>
> Before selecting the next action, assess whether the currently available information remains temporally reliable: whether the observed state is still valid, whether the previous action has truly taken effect or completed, and whether the existing feedback is fresh enough to support the current decision.

B.4 Reasoning-Mode Evaluation

For models that expose a controllable thinking mode, we additionally evaluate a subset of models with thinking enabled. This analysis tests whether additional inference-time reasoning mitigates temporal state misalignment while keeping the benchmark instances and answer extraction protocol unchanged. Remaining unparsed outputs are counted as incorrect; one unresolved timeout for Qwen3.6-Max-Preview is also counted as incorrect.

Thinking-enabled runs improve overall accuracy for all three evaluated models, indicating that additional inference-time reasoning can help with temporal-state decisions. However, the gains do not eliminate temporal state misalignment errors. F2O remains the weakest category for DeepSeek V4 Pro and Qwen3.6-Max-Preview, and Qwen3.6-Max-Preview decreases on F2O despite improving overall accuracy. We therefore treat reasoning mode as an inference-time sensitivity factor rather than as a replacement for the main evaluation setting.

B.5 Performance on A2F Motion-Compensation Instances

Motion-compensation instances form a subset of A2F cases designed to test whether the agent can account for execution delay. Unlike standard A2F cases where the robust behavior is often to wait, poll, or verify completion, these instances require the model to directly choose an action that compensates for the future action-time state, such as targeting the predicted position of a moving object. We evaluate the same non-reasoning main-benchmark runs following the main evaluation protocol.

Category	Proceed	Cons.	Recover	Other
O2A (132)	29 (22.0%)	103 (78.0%)	0 (0.0%)	0 (0.0%)
A2F (378)	92 (24.3%)	278 (73.5%)	8 (2.1%)	0 (0.0%)
F2O (201)	0 (0.0%)	201 (100.0%)	0 (0.0%)	0 (0.0%)
Overall (711)	121 (17.0%)	582 (81.9%)	8 (1.1%)	0 (0.0%)

Table 8: Gold-option policy distribution by misalignment category. Each cell reports count and percentage within the corresponding category.

Model	Setting	O2A	A2F	F2O	Overall	Δ
DeepSeek V4 Pro	Main	59.8	63.8	36.3	55.3	–
DeepSeek V4 Pro	Thinking	74.2	76.5	49.8	68.5	+13.2
Kimi-K2.6	Main	65.9	61.4	47.8	58.4	–
Kimi-K2.6	Thinking	73.5	63.5	56.2	66.0	+7.6
Qwen3.6-Max-Preview	Main	68.9	66.7	51.7	62.9	–
Qwen3.6-Max-Preview	Thinking	75.8	77.5	45.8	68.2	+5.3

Table 9: Reasoning-mode sensitivity results. Main denotes the setting used in Table 2; Thinking denotes runs with thinking enabled for models with controllable thinking modes. Δ reports the overall accuracy difference between Thinking and Main.

Model	Correct	Acc.
Qwen3-8B	4	5.3
DeepSeek V4 Flash	18	24.0
DeepSeek V4 Pro	30	40.0
GLM-5.1	53	70.7
GPT-5.5	71	94.7
Kimi-K2.6	28	37.3
MiniMax-M2.7	33	44.0
Qwen3.6-Max-Preview	39	52.0

Table 10: Performance on the A2F motion subset. The subset contains 75 instances from the main benchmark. Unparsed outputs are counted as incorrect.

The results show that motion-compensation behavior is highly uneven across models. GPT-5.5 solves most instances, while GLM-5.1 is the strongest non-GPT model on this subset. Several models perform substantially below their overall A2F accuracy, suggesting that this subset tests a different behavior from simply recognizing that an asynchronous action is unfinished. These instances usually have a simple underlying pattern: the model should not act on the currently observed position, but on the expected position when the physical action takes effect. The large performance gap therefore suggests that many models do not reliably translate execution delay into an action-time state, even when the required compensation pattern is visually or textually simple.

To further characterize the errors, we map each option in this subset to one of four semantic types: *motion-compensated* (the correct future action-time position), *current-position* (the object's current position in the second frame), *past-position* (a position in the opposite or previous motion direction), and *off-path* (a point away from the motion path). Table 11 reports the selected option types among parsed incorrect predictions.

The dominant failure pattern for several models is selecting the current position rather than the motion-compensated position. This pattern is especially pronounced for DeepSeek V4 Flash, DeepSeek V4 Pro, Qwen3.6-Max-Preview, and Qwen3-8B, suggesting that these models often identify the relevant object but fail to extrapolate its position at the time the physical action takes effect. By contrast, GLM-5.1 and Kimi-K2.6 make a larger fraction of past-position errors, indicating a different confusion pattern over the direction of compensation.

Model	#Wrong	Current Pos.	Past Pos.	Off-path
DeepSeek V4 Flash	57	84.2	14.0	1.8
DeepSeek V4 Pro	45	86.7	11.1	2.2
GLM-5.1	22	36.4	59.1	4.5
GPT-5.5	4	25.0	75.0	0.0
Kimi-K2.6	47	25.5	59.6	14.9
MiniMax-M2.7	16	37.5	50.0	12.5
Qwen3.6-Max-Preview	36	69.4	25.0	5.6
Qwen3-8B	69	91.3	7.2	1.4

Table 11: Error-type distribution among parsed incorrect predictions on the A2F motion-compensation subset. Values are percentages over parsed wrong predictions. Current Pos. denotes selecting the object's current position in the second frame; Past Pos. denotes selecting a position in the opposite or previous motion direction; Off-path denotes selecting a point away from the motion path.