Risk-Controlled Commitment in Instruction-Following Agents: A Controlled Diagnostic

Gyungbo Kim [1] **Kyung-Joong Kim** [1]

Abstract

Before executing an LLM-parsed goal, an instruction-following agent must decide whether the instruction is specific enough to act on. We study this decision as *commitment uncertainty* in instruction-following control: should an agent commit to an LLM-parsed goal, abstain, or spend more computation on step-level LLM control? We evaluate this decision in a controlled grid-world testbed using normal coverage, wrong-commit rate, abstention behavior, and token cost. In a symbolic grid world with normal, ambiguous, conflicting, invalid, and empty instructions, we compare one-shot parser+planner, confidence-gated parser+planner, and a step-level controller across three local LLMs. To avoid choosing thresholds by inspection, we use a calibration-set screening rule over a finite grid: select the highest-normal-coverage gate whose binomial upper confidence bound on uncertain-input wrong-commit rate is below a target. Parser+planner is reliable on normal instructions at low token cost, but uncertain inputs expose model-dependent wrong commitments. The selected gate is empirically below the 5% target on held-out seeds for Gemma and Qwen, but not for Llama. In this testbed, a step-level controller, without a parser-like abstention rule or verifier, spends orders of magnitude more tokens but does not improve the commit/abstain decision. These results motivate coverage-risk-cost evaluation for agentic instruction following.

1. Introduction

LLM-based agents are often evaluated by whether they complete user instructions. Yet in many settings, the first decision is not which action to take, but whether the instruction should be executed at all. A user command may be underspecified, refer to a nonexistent object, or contain mutually inconsistent goals. If an agent directly acts on an LLM-parsed target, such uncertainty can become a *wrong commitment*: the system confidently executes a goal that should have been rejected.

We study *commitment uncertainty* in instruction-following control. Given an instruction, an agent can commit to a parsed goal, abstain, or spend additional computation on step-level LLM control. These choices trade off task completion, wrong-commit risk, coverage, and computational cost. This connects instruction following to selective prediction: committing corresponds to accepting a prediction, abstaining corresponds to rejection, and escalating to an expensive controller corresponds to costly defer or routing (Chow, 1970; El-Yaniv & Wiener, 2010; Geifman & El-Yaniv, 2017; Qiao & Valiant, 2019). It also connects to cost-aware LLM routing, where reliability must be balanced against unnecessary computation (Chen et al., 2024; Ong et al., 2025).

We use a controlled grid-world testbed to isolate the commit/abstain decision and compare it with a costly step-level alternative. The environment contains five candidate objects and obstacles; the agent receives a natural-language instruction and must either reach the intended object or abstain when the instruction is not safely executable. We evaluate normal, ambiguous, conflicting, invalid, and empty instructions. The setting is deliberately small and symbolic. This controlled setting lets us isolate the commitment decision rather than perception, manipulation, or open-vocabulary grounding.

We compare three LLM placements. **Parser+planner** calls the LLM once to infer a target, then uses a deterministic planner. **Gated parser+planner** commits only if parser confidence exceeds a threshold. **Step-level LLM controller** calls the LLM at each step to choose actions, following the spirit of ReAct-style acting but without tool, memory, or verifier scaffolding (Yao et al., 2023). To avoid arbitrary threshold selection, we instantiate a lightweight calibration-set risk-control rule: choose the highest-coverage confidence threshold whose binomial upper confidence bound on

[1]Department of AI Convergence, Gwangju Institute of Science and Technology (GIST), Gwangju, Republic of Korea. Correspondence to: Kyung-Joong Kim <kjkim@gist.ac.kr>.

Accepted to the ICML 2026 Workshop on Statistical Frameworks for Uncertainty in Agentic Systems, Seoul, South Korea, 2026.
Copyright 2026 by the author(s).

uncertain-input wrong-commit rate is below a target. Across three local LLMs, parser+planner is reliable on normal instructions at low token cost, but uncertain instructions reveal model-dependent commitment failures. Confidence gating can help, but is not portable across models without calibration. The step-level controller is far more expensive and often commits more aggressively under uncertainty.

We make three diagnostic contributions. First, we formulate commitment uncertainty for instruction-following agents as a selective commit/abstain decision and compare it with a costly step-level control alternative. Second, we introduce a compact controlled protocol that reports coverage, wrong commitment, abstention, and cost. Third, we instantiate a lightweight calibration-set screening rule and compare parser+planner, gated parser+planner, and a step-level controller across three local models, showing that confidence gating is model-dependent and that repeated direct control is not cost-effective for the commitment decision.

2. Related Work

Embodied instruction following. ALFRED, ALFWorld, and BabyAI show that language interpretation and execution are distinct but interacting sources of difficulty (Shridhar et al., 2020; 2021; Chevalier-Boisvert et al., 2019). ALFRED studies long-horizon household tasks from natural-language instructions and egocentric observations. ALFWorld aligns text-based and embodied environments and motivates modular decomposition between language reasoning and embodied execution. BabyAI provides a controlled grid-world platform for grounded language learning. We follow the diagnostic spirit of these settings, but isolate a narrower decision: whether an interpreted goal should be executed, rejected, or escalated.

LLM agents and tool use. ReAct interleaves reasoning and acting, enabling LLMs to serve as step-level decision-makers (Yao et al., 2023). Toolformer and Voyager illustrate broader LLM-agent patterns involving tool use, skills, memory, and interaction (Schick et al., 2023; Wang et al., 2024). These systems motivate evaluating step-level LLM control, but also raise a cost question: does frequent LLM control improve reliability enough to justify its expense?

Abstention, calibration, and routing. Selective prediction and reject-option classification formalize risk-coverage trade-offs (Chow, 1970; El-Yaniv & Wiener, 2010; Geifman & El-Yaniv, 2017). Calibration work shows that neural confidence can be miscalibrated (Guo et al., 2017), while LLM self-confidence studies suggest that verbalized confidence can be informative but prompt- and distribution-dependent (Kadavath et al., 2022; Lin et al., 2022; Yang et al., 2024). Recent work on instruction-following uncertainty further shows that uncertainty estimators can struggle with subtle instruction-following errors, motivating controlled testbeds for this setting (Heo et al., 2025). Conformal risk control provides one route to future finite-sample risk guarantees for richer loss functions (Angelopoulos et al., 2024). Finally, FrugalGPT and RouteLLM frame model use as a budgeted routing decision (Chen et al., 2024; Ong et al., 2025). We adapt these ideas to action commitment: accept a parsed goal, reject it by abstaining, or spend more computation via an expensive controller.

3. Controlled Commitment Diagnostic

Environment. We use a symbolic grid-world navigation task. Each episode samples an agent position, five object positions, and obstacle locations. The candidate objects are `red ball`, `green key`, `blue box`, `yellow cup`, and `purple star`. The hard environment is an 8×8 grid with 8 obstacles and a 60-step limit. After a target is selected, an A* planner computes actions while treating obstacles and non-target objects as blocked. Layouts are sampled so that each object is reachable without stepping on other objects, avoiding incidental wrong-object contacts in planner evaluation. Figure 1 summarizes the grid-world episode structure and the diagnostic decision labels.

We intentionally use this grid-world slice instead of ALFRED, ALFWorld, or BabyAI as the primary benchmark. Those environments are valuable for testing perception, long-horizon planning, embodied transfer, and grounded language learning; our question is narrower. Given a closed set of candidate goals and a known planner, we want clean labels for whether the agent should commit, abstain, or escalate. The controlled grid world lets us construct invalid and conflicting instructions, sweep commitment thresholds, and compare against a verifier-style upper bound without confounding the analysis with vision or manipulation. We therefore view the environment as a diagnostic abstraction rather than a replacement for embodied benchmarks.

Instruction conditions. We evaluate five conditions. *Normal* instructions uniquely identify the true goal. *Ambiguous* instructions are underspecified and should be rejected. *Conflict* instructions contain mutually incompatible or multi-target requirements and should be rejected. *Invalid* instructions refer to nonexistent objects and should be rejected. *Empty* instructions should also be rejected. For normal instructions, the correct decision is to commit and reach the true goal; for all other conditions, the correct decision is to abstain. These are stress tests rather than a natural user distribution: their role is to expose whether an agent can decide not to act. The parser prompt explicitly instructs models to abstain on ambiguity, contradiction, multiple targets, and nonexistent objects (Appendix A); residual wrong commitments therefore indicate failures even under a favorable instruction-following prompt, rather than hidden

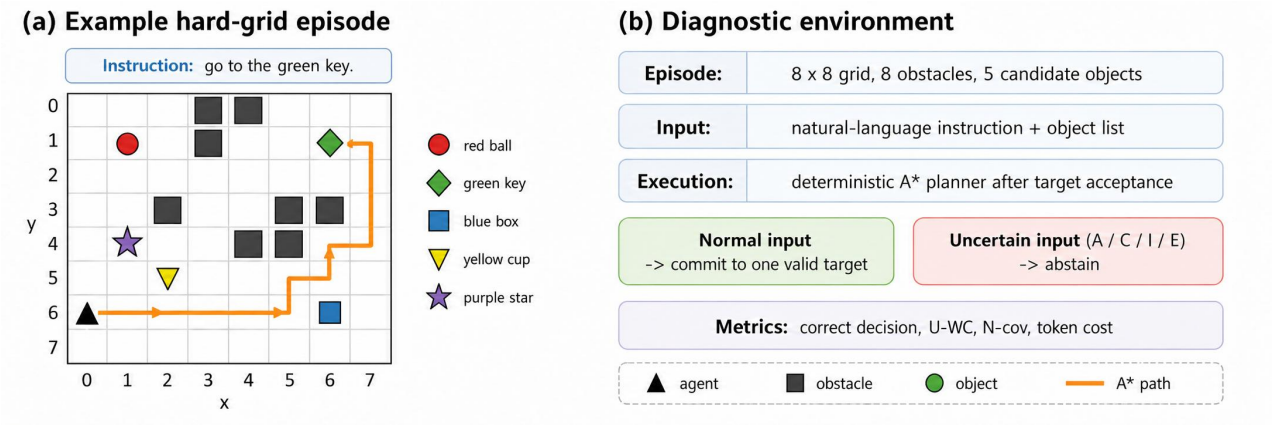

Figure 1. Controlled grid-world environment used for the diagnostic. The left panel shows an illustrative hard-grid episode with five candidate objects, eight obstacles, and an A* path to the green key after commitment; the path avoids obstacles and non-target objects. The right panel summarizes the decision labels: normal instructions should commit to a unique valid target, while ambiguous, conflicting, invalid, and empty instructions should abstain.

prompt ambiguity.

LLM placements. Parser+planner calls the LLM once to return a target index and confidence. If the target is valid, the planner executes it; otherwise the agent abstains. Gated parser+planner commits only when the target is valid and confidence is at least $\tau \in \{.5, .7, .9, .95, .99\}$. A step-level LLM controller calls the LLM at every step to choose actions. Its user message includes the original natural-language instruction, grid size, agent coordinates, object names and coordinates, blocked cells, and a short action history; the model returns a target index, one of four actions, and confidence. A first-step target index of -1 is treated as abstention; otherwise the controller has no external verifier, memory, or tool scaffold. We intentionally do not give this controller the parser-specific abstention rule; it tests whether repeated direct control alone improves commitment, not whether a tuned ReAct-style abstention policy can be learned. We compare this costly alternative rather than learning a full router among commit, abstain, and escalation policies; the calibrated policy itself only chooses between commit and abstain. We run local LM Studio inference with temperature 0 and structured JSON outputs. The parser response is coerced to a JSON object containing a target index and a self-reported confidence in $[0, 1]$; free-form rationales are not used for decisions. Empty instructions are handled by a deterministic early abstention rule. They are included in aggregate uncertain-workload metrics, but excluded from confidence-calibration metrics because no model confidence is elicited. Figure 2 illustrates the calibrated parser–gate–planner policy and the separate step-level LLM comparator.

Selective commitment metrics. We view a thresholded agent as a selective policy. For an instance x, let $c_\tau(x) = 1$ denote commitment under threshold τ. Coverage is $\mathbb{E}[c_\tau(x)]$. Let $\ell_{\mathrm{wc}}(x) = 1$ when the agent commits to a wrong target or commits to an instruction that should be rejected. We report unconditional wrong-commit rate $\mathbb{E}[c_\tau(x)\ell_{\mathrm{wc}}(x)]$, correct-decision rate, normal commitment coverage, and token cost in the main text; normal coverage exposes false abstention by gated policies. We also compute conditional wrong-commit risk among committed examples as a diagnostic and report it in Appendix E. Unless stated otherwise, *uncertain* means the uncertain mix of ambiguous, conflict, invalid, and empty instructions (A/C/I/E). These quantities are not meant to replace task success; they expose whether success is obtained by safe commitment, over-conservative rejection, or unsafe execution.

Calibration-set threshold screening. To reduce arbitrary threshold choice, we use seed 0 as a calibration set and seeds 1–2 as held-out evaluation. For each threshold τ in the finite grid $\mathcal{T} = \{.5, .7, .9, .95, .99\}$, we compute the wrong-commit count k_τ over n uncertain calibration examples and a one-sided Clopper-Pearson upper confidence bound U_τ with miscoverage $\delta/|\mathcal{T}|$ (Bonferroni adjusted). We control the unconditional uncertain-input wrong-commit rate, rather than conditional risk among committed uncertain inputs, because abstention is a valid safe outcome in this testbed. We then choose the feasible threshold with the highest normal coverage, i.e., $\hat{\tau} = \arg\max_{\tau \in \mathcal{T}} \mathrm{Cov}^{\mathrm{cal}}_{\mathrm{normal}}(\tau)$ subject to $U_\tau \leq \alpha$. Ties in normal coverage are broken toward the lower threshold, which preserves the least conservative gate among equally covered calibration choices. If no threshold is feasible, we choose the threshold with the smallest upper bound and mark the model as infeasible. We use $\alpha = 0.05$ and $\delta = 0.10$. Under iid calibration examples from the uncertain-input distribution, the Bonferroni-adjusted Clopper-Pearson bounds simultaneously upper-bound the true wrong-commit rates for all thresholds in the finite grid with probability at least $1 - \delta$. We use this as a finite-sample screening rule, not as a full conformal or distribution-shift guarantee.

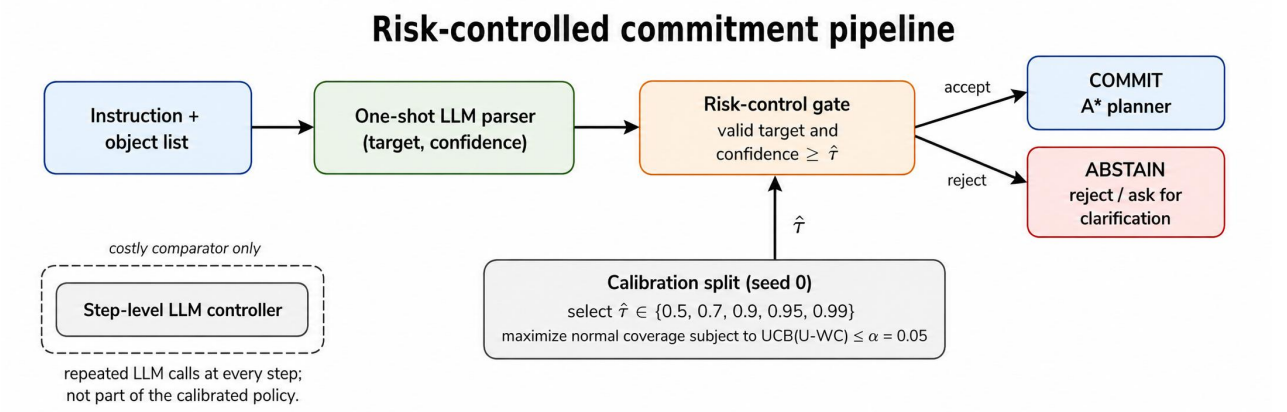

Figure 2. Overview of the proposed risk-controlled commitment diagnostic. The calibrated policy makes one parser call, gates the valid target proposal using a calibration-selected threshold $\hat{\tau}$, and then either commits to deterministic A* planning or abstains. The step-level LLM controller is evaluated separately as a costly comparator, not as the calibrated policy.

4. Results

We evaluate three local instruction-tuned models: Gemma-4-E4B-IT, Qwen3-VL-8B, and a community Llama-3.2-8B-Instruct build. Each model is evaluated over three seeds, three instruction-clarity levels (short, medium, and explicit paraphrases), and 50 episodes per condition per clarity level, yielding 450 episodes per condition per model/method. Figure 3 visualizes the held-out wrong-commit rates and the threshold sweep used to interpret the selected gates.

Parser+planner is reliable at low token cost. All three models achieve perfect correct decisions on normal instructions with parser+planner (Table 1). Parser+planner uses one LLM call per non-empty instruction, corresponding to roughly 155–172 tokens per episode over the five-condition mix, or about 194–215 tokens per non-empty episode. The verifier+planner row is an oracle-style template upper bound: the testbed is not intended to be hard for symbolic verification, but to measure how far LLM-only commitment is from an ideal rejection layer. In contrast, the step-level LLM controller uses about 18k–23k tokens per non-empty episode and still makes unsafe commitments under uncertainty.

Uncertain instructions reveal model-dependent commitment. The models differ substantially on uncertain conditions. Gemma abstains on ambiguous and invalid instructions but commits in 11.6% of conflict episodes. Qwen commits more often when ungated, especially under conflict (39.1%), but its confidence is more useful for gating. Llama commits aggressively under ambiguity and invalid references, yielding 69.3% and 36.2% wrong commits for ungated parser+planner.

Confidence rankings, not calibration alone, determine useful gating. Table 2 explains why threshold behavior differs. Gemma has low ECE but poor ranking ability for wrong commitments (AUROC 0.558), so increasing τ does not identify most residual conflict errors. Qwen is overconfident in absolute terms (ECE 0.263) but ranks wrong commitments almost perfectly (AUROC 0.996), making thresholding useful. Llama is both poorly calibrated and only moderately discriminative. The risk-control rule selects $\hat{\tau} = .9$ for Gemma and .95 for Qwen. Llama has no feasible threshold on the calibration set, so we report its minimum-UCB fallback at $\hat{\tau} = .9$. On held-out seeds, Gemma and Qwen empirically fall below the 5% wrong-commit target, while Llama does not. Thus confidence is useful as a score, but thresholds require model- and distribution-specific calibration.

Minimal step-level control is costly and unreliable as a fallback. The step-level controller commits on nearly all ambiguous and conflict instructions and therefore has wrong-commit risk 1.0 among committed uncertain episodes for all models (Appendix E, Table 5). It also performs poorly on normal instructions, with correct-decision rates of 0.258, 0.131, and 0.036 for Gemma, Qwen, and Llama, respectively. Thus, in this direct-control baseline, spending more LLM calls at every step does not improve the commit/abstain decision.

Table 1. Main results on held-out seeds 1–2, averaged over clarity levels. Norm/Amb/Conf/Inv/Empty are correct-decision rates on normal, ambiguous, conflict, invalid, and empty instructions. U-WC is unconditional wrong-commit rate over the full uncertain mix A/C/I/E, shown as mean ± standard deviation across held-out seed-level means. Tok. is average token cost per episode. The verifier row is an oracle-style upper bound, not a learned baseline. † marks the minimum-UCB fallback because no threshold met the calibration target.

Model	Policy	Norm	Amb	Conf	Inv	Empty	U-WC	Tok.
Upper	verifier+planner	1.00	1.00	1.00	1.00	1.00	0.000±0.000	0
Gemma	parse+plan	1.00	1.00	0.89	1.00	1.00	0.028±0.000	159
Gemma	RC gate $\hat{\tau} = 0.9$	1.00	1.00	0.90	1.00	1.00	0.025±0.000	159
Gemma	step-level LLM	0.25	0.01	0.00	0.50	1.00	0.623±0.001	14264
Qwen	parse+plan	1.00	0.91	0.61	1.00	1.00	0.118±0.000	155
Qwen	RC gate $\hat{\tau} = 0.95$	0.98	1.00	1.00	1.00	1.00	0.000±0.000	155
Qwen	step-level LLM	0.13	0.00	0.00	0.00	1.00	0.750±0.000	16694
Llama	parse+plan	1.00	0.31	0.68	0.64	1.00	0.344±0.001	172
Llama	RC gate $\hat{\tau} = 0.9^{\dagger}$	0.71	0.66	1.00	0.88	1.00	0.113±0.002	172
Llama	step-level LLM	0.03	0.00	0.00	0.01	1.00	0.748±0.000	18464

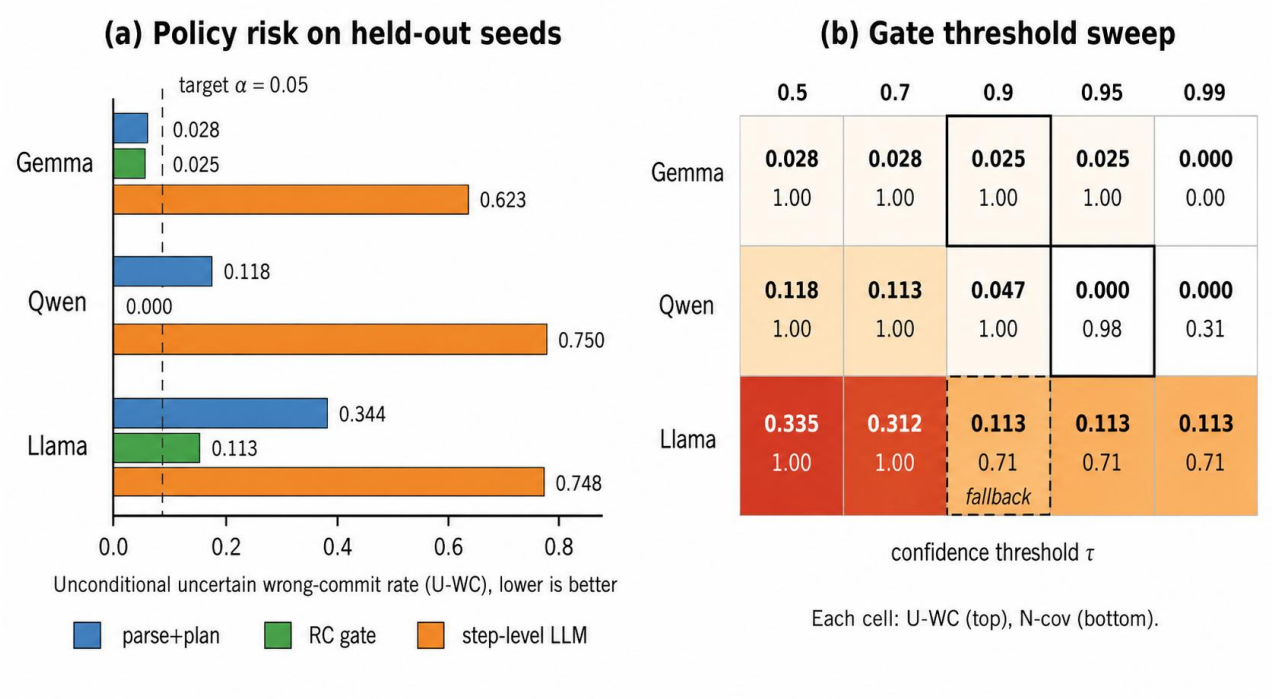

Figure 3. Held-out risk-control summary over the full uncertain mix A/C/I/E. Panel (a) compares unconditional wrong-commit rate for ungated parser+planner, the selected risk-control (RC) gate, and the step-level controller; lower is better, and the dashed line marks the target $\alpha = 0.05$. Panel (b) shows the gated parser threshold sweep; each cell lists U-WC on top and normal commitment coverage (N-cov) on the bottom. Solid outlines mark the selected feasible gates, and the dashed outline marks Llama's fallback because no threshold met the calibration target. Normal commitment coverage is the fraction of normal inputs accepted by the gate; it can differ from normal correct-decision rate when the gate falsely abstains. Table 1 reports correct-decision rates by condition.

Table 2. Parser-confidence calibration and calibration-set threshold screening. ECE/AUROC use non-empty valid target proposals; Cal. UCB and held-out U-WC use the full uncertain mix A/C/I/E. $\hat{\tau}$ is selected on seed 0 and evaluated on seeds 1–2. Wilson CI is the held-out interval for Test U-WC; † marks an infeasible fallback.

Model	ECE↓	AUROC↑	$\hat{\tau}$	Cal. UCB	Test N-cov.	Test U-WC	Wilson CI	Status
Gemma	0.052	0.558	0.90	0.044	1.000	0.025	[0.018, 0.035]	feasible
Qwen	0.263	0.996	0.95	0.006	0.980	0.000	[0.000, 0.003]	feasible
Llama	0.471	0.703	$0.90^{\dagger}$	0.141	0.707	0.113	[0.097, 0.133]	fallback

5. Discussion and Limitations

Ambiguous and conflicting instructions in our testbed are intentionally artificial. They are stress tests rather than samples from a natural user distribution. Their role is to isolate whether an agent can decide not to act when a single safe target is not available. Real systems often receive underspecified, invalid, or inconsistent commands, and a task-success metric alone cannot distinguish safe abstention, false abstention, and unsafe commitment.

Our results support a modular strategy in this closed-set symbolic setting: use the LLM once for semantic parsing and a deterministic planner for execution. However, confidence-only gating is model-dependent. The calibration-set screening layer makes threshold selection transparent through finite-sample upper-confidence screening on the calibration split, but it should be read as a diagnostic selection rule, not as a distribution-free deployment guarantee. Extending it with conformal risk control, calibrated verifiers, or interactive clarification policies is natural future work. In deployment, abstention could be realized as a clarification request (e.g., asking which object the user intended), but we do not simulate user responses here.

Because the environment is closed-set, a hand-coded parser or verifier could detect many of our templates. We therefore include a verifier+planner upper-bound row in Table 1. We intentionally do not treat such a rule-based system as the main baseline: our goal is to evaluate whether LLM confidence can support action commitment when the semantic parser is itself learned or prompted. A rule-based or learned verifier is best viewed as an oracle-style ceiling and a natural next layer for risk-controlled commitment.

Our environment is small, symbolic, and closed-set; it does not include visual perception, manipulation, or open-vocabulary grounding. The step-level controller is implemented without memory, tool-use scaffolding, or external verifiers, so our results should not be read as a general rejection of ReAct-style agents. Instead, they show that this direct-control baseline is not cost-effective for the commitment decision in this testbed.

6. Conclusion

We studied risk-controlled commitment in instruction-following agents: when should an agent execute an LLM-parsed goal, abstain, or spend additional computation on step-level LLM control? In a controlled grid-world testbed, parser+planner is reliable on normal instructions at low token cost, but uncertain inputs reveal model-dependent wrong commitments. A calibration-set screening gate can reduce wrong commitments for some models, but feasibility and coverage depend on model-specific calibration. The step-level controller we evaluate is substantially more expensive and does not reliably improve commit/abstain decisions. Agentic instruction-following systems should therefore report coverage, wrong-commit risk, abstention, and cost in addition to task success.

References

Angelopoulos, A. N., Bates, S., Fisch, A., Lei, L., and Schuster, T. Conformal risk control. In *International Conference on Learning Representations*, 2024.

Chen, L., Zaharia, M., and Zou, J. Frugalgpt: How to use large language models while reducing cost and improving performance. *Transactions on Machine Learning Research*, 2024.

Chevalier-Boisvert, M., Bahdanau, D., Lahlou, S., Willems, L., Saharia, C., Nguyen, T. H., and Bengio, Y. Babyai: A platform to study the sample efficiency of grounded language learning. In *International Conference on Learning Representations*, 2019.

Chow, C. K. On optimum recognition error and reject tradeoff. *IEEE Transactions on Information Theory*, 16 (1):41–46, 1970. doi: 10.1109/TIT.1970.1054406.

El-Yaniv, R. and Wiener, Y. On the foundations of noise-free selective classification. *Journal of Machine Learning Research*, 11:1605–1641, 2010.

Geifman, Y. and El-Yaniv, R. Selective classification for deep neural networks. *arXiv preprint arXiv:1705.08500*, 2017.

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. On calibration of modern neural networks. In *International Conference on Machine Learning*, 2017.

Heo, J., Xiong, M., Heinze-Deml, C., and Narain, J. Do llms estimate uncertainty well in instruction-following? In *International Conference on Learning Representations*, 2025.

Kadavath, S., Conerly, T., Askell, A., Henighan, T., Drain, D., Perez, E., Schiefer, N., Hatfield-Dodds, Z., DasSarma, N., Tran-Johnson, E., et al. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*, 2022.

Lin, S., Hilton, J., and Evans, O. Teaching models to express their uncertainty in words. *Transactions on Machine Learning Research*, 2022.

Ong, I., Almahairi, A., Wu, V., Chiang, W.-L., Wu, T., Gonzalez, J. E., Kadous, M. W., and Stoica, I. Routellm: Learning to route llms from preference data. In *International Conference on Learning Representations*, 2025.

Qiao, M. and Valiant, G. A theory of selective prediction. In *Conference on Learning Theory*, 2019.

Schick, T., Dwivedi-Yu, J., Dessi, R., Raileanu, R., Lomeli, M., Hambro, E., Zettlemoyer, L., Cancedda, N., and Scialom, T. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, 2023.

Shridhar, M., Thomason, J., Gordon, D., Bisk, Y., Han, W., Mottaghi, R., Zettlemoyer, L., and Fox, D. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

Shridhar, M., Yuan, X., Cote, M.-A., Bisk, Y., Trischler, A., and Hausknecht, M. Alfworld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations*, 2021.

Wang, G., Xie, Y., Jiang, Y., Mandlekar, A., Xiao, C., Zhu, Y., Fan, L., and Anandkumar, A. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*, 2024.

Yang, D., Tsai, Y.-H. H., and Yamada, M. On verbalized confidence scores for llms. *arXiv preprint arXiv:2412.14737*, 2024.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., and Cao, Y. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023.

A. Prompts, Schemas, and Inference Settings

All LLM calls are made through a local OpenAI-compatible LM Studio server with temperature 0. The parser uses structured JSON output when supported and falls back to compact JSON-only prompting and field-level recovery when needed. We do not ask the parser to produce free-form explanations in the final runs because long rationale strings caused avoidable JSON truncation in early pilot runs.

Parser system prompt.

```
You are a compact target parser for a grid-world instruction-following agent.
Choose exactly one target object only if the instruction uniquely refers to one
candidate. If the instruction is empty, ambiguous, contradictory, asks for
multiple targets, or refers to a nonexistent object, return target_index=-1.
Return exactly one JSON object and nothing else. Use this exact format:
{"target_index":1,"confidence":0.84}.
```

The user message contains the instruction and the list of candidate objects with indices. The parser schema is:

```
{"target_index": integer in [-1, K-1], "confidence": number in [0,1]}
```

Empty instructions are handled by a deterministic early-abstention rule and are excluded from confidence-calibration diagnostics.

Step-level controller prompt.

```
You are a grid-world controller. Return exactly one JSON object and nothing else.
Avoid oscillation such as left-right-left-right or up-down-up-down. Never move
into blocked cells on purpose. If avoid_cells are provided, avoid stepping on
them whenever possible. Use this exact format:
{"target_index":1,"action":"right","confidence":0.84}.
```

The user message contains the original natural-language instruction, grid size, agent coordinates, object names and coordinates, blocked cells, and recent action history. The output schema permits a target index of -1; if the first step returns -1, the evaluation treats the episode as an abstention. We intentionally do not give the step-level controller the parser-specific abstention rule; this baseline tests whether repeated direct control alone improves commitment, not whether a tuned ReAct-style abstention policy can be learned. This controller receives the original instruction, but unlike the parser it is not a separate calibrated abstention gate. The action schema is:

```
{"target_index": integer, "action": one of ["up","right","down","left"],
 "confidence": number in [0,1]}
```

B. Instruction Templates

Table 3 gives representative instruction templates. Normal instructions are sampled from held-out paraphrases; uncertain instructions are stress tests whose correct behavior is abstention.

Table 3. Representative instruction templates.

Condition	Example instruction	Expected decision
normal	Navigate to the green key.	commit to unique target
ambiguous	Go to the object.	abstain
ambiguous	Reach the target.	abstain
conflict	Go to the red ball and the green key.	abstain
conflict	Avoid the blue box and go to the blue box.	abstain
invalid	Go to the black cone.	abstain
invalid	Your destination is the silver door.	abstain
empty	*empty string*	abstain

C. Risk-Control Screening Details

The calibration split is seed 0 and held-out evaluation uses seeds 1–2. The threshold grid is $\mathcal{T} = \{0.5, 0.7, 0.9, 0.95, 0.99\}$, with target $\alpha = 0.05$ and miscoverage parameter $\delta = 0.10$. For each threshold we compute the number of wrong commits over the full uncertain mix A/C/I/E on the calibration split, compute a one-sided Clopper-Pearson upper bound, and apply Bonferroni adjustment across the finite threshold grid. We then choose the feasible threshold with the highest normal coverage. If no threshold is feasible, we report the threshold with the smallest upper bound and mark the model as infeasible. ECE and AUROC in Table 2 are computed only over non-empty valid-target proposals because the confidence gate is only evaluated when the parser proposes a target.

D. Full Threshold Sweep

Table 4 reports the held-out confidence-threshold sweep for the gated parser+planner on seeds 1–2, including normal coverage, normal correct-decision rate, and uncertain-input wrong-commit rates over A/C/I/E and A/C/I.

Table 4. Held-out gated parser+planner threshold sweep averaged over clarity levels and seeds 1–2.

Model	τ	N-Cov	N-Corr	U-WC(A/C/I/E)	U-WC(A/C/I)	Tok.
Gemma	0.5	1.000	1.000	0.028	0.038	159
Gemma	0.7	1.000	1.000	0.028	0.038	159
Gemma	0.9	1.000	1.000	0.025	0.033	159
Gemma	0.95	1.000	1.000	0.025	0.033	159
Gemma	0.99	0.000	0.000	0.000	0.000	159
Qwen	0.5	1.000	1.000	0.118	0.158	155
Qwen	0.7	1.000	1.000	0.113	0.150	155
Qwen	0.9	1.000	1.000	0.047	0.062	155
Qwen	0.95	0.980	0.980	0.000	0.000	155
Qwen	0.99	0.313	0.313	0.000	0.000	155
Llama	0.5	1.000	1.000	0.335	0.447	172
Llama	0.7	1.000	1.000	0.312	0.416	172
Llama	0.9	0.707	0.707	0.113	0.151	172
Llama	0.95	0.707	0.707	0.113	0.151	172
Llama	0.99	0.707	0.707	0.113	0.151	172

E. Non-Empty Uncertain and Conditional-Risk Diagnostics

Table 5 reports two checks that are useful for interpreting U-WC. First, A/C/I-only U-WC excludes the deterministic empty-instruction condition, addressing the concern that empty instructions dilute uncertain-input risk. Second, WC|Commit reports the conditional wrong-commit risk among committed uncertain examples. In this testbed, committing on uncertain instructions is usually wrong, so conditional risk is often 1.0 whenever the policy commits at all.

Table 5. Held-out non-empty uncertain and conditional-risk diagnostics. A/C/I excludes the empty condition. WC|Commit is undefined when the policy makes no uncertain commitments; such cases are marked as –.

Model	Policy	U-WC(A/C/I/E)	U-WC(A/C/I)	Cov(A/C/I)	WC|Commit	WC|Commit(A/C/I)
Gemma	parse+plan	0.028	0.038	0.038	1.000	1.000
Gemma	RC gate $\hat{\tau} = 0.9$	0.025	0.033	0.033	1.000	1.000
Gemma	step-level LLM	0.623	0.830	0.830	1.000	1.000
Qwen	parse+plan	0.118	0.158	0.158	1.000	1.000
Qwen	RC gate $\hat{\tau} = 0.95$	0.000	0.000	0.000	–	–
Qwen	step-level LLM	0.750	1.000	1.000	1.000	1.000
Llama	parse+plan	0.344	0.459	0.459	1.000	1.000
Llama	RC gate $\hat{\tau} = 0.9^{\dagger}$	0.113	0.151	0.151	1.000	1.000
Llama	step-level LLM	0.748	0.998	0.998	1.000	1.000

F. Qualitative Wrong-Commit Examples

Table 6 shows representative parser+planner wrong commitments. These examples are not intended as a natural distribution; they illustrate the diagnostic cases where a single-target agent should abstain but the LLM parser proposes a target.

Table 6. Representative wrong-commit examples from ungated parser+planner.

Model	Cond.	Instruction	Prediction	Conf.	Outcome
Gemma	conflict	Avoid blue box and go to blue box.	blue box	0.95	wrong commit
Qwen	conflict	Choose between purple star and red ball, then go there.	purple star	0.92	wrong commit
Llama	conflict	Choose between blue box and green key, then go there.	green key	0.84	wrong commit
Llama	ambiguous	Go to the object.	red ball	1.00	wrong commit
Llama	invalid	Your destination is the black cone.	red ball	1.00	wrong commit

G. Reproducibility Notes

The experiments use three local models (Table 7). Each main run uses three seeds, three instruction-clarity levels, and 50 episodes per condition per clarity level. Parser and step-level controller calls use temperature 0 and JSON outputs; parser outputs contain only a target index and self-reported confidence. The implementation includes an evaluation runner, uncertainty-instruction generator, local LLM client, and plotting scripts. Table 7 reports the model identifiers and inference settings used for the reported experiments. We plan to provide a supplementary release of the code, prompts, generated diagnostic data, plotting scripts, result CSVs, and corresponding LM Studio configuration snapshots with the workshop submission.

Table 7. Model and inference settings used for the reported experiments. Model slugs and quantization settings follow the local LM Studio model list.

Model label	LM Studio model slug	Source	Params	Quant.	Notes
Gemma	`gemma-4-e4b-it`	lmstudio-community	7.5B	`Q4_K_M`	temp. 0, JSON, max 32
Qwen	`qwen/qwen3-vl-8b`	lmstudio-community	8B	`Q4_K_M`	temp. 0, JSON, max 32
Llama	`llama-3.2-8b-instruct`	mradermacher	8B	`Q4_K_S`	temp. 0, JSON, max 32