ATC-BENCH: A Simulation-Grounded Benchmark for Evaluating LLM-Generated Air Traffic Control Instructions

Jinjun Cai[1], **Jiaxing Li**[1], **Xingyue Wang**[1]
Weihuang Zheng[1], **Youyong Kong**[1,*], **Hui Ding**[2,*]
[1] School of Computer Science and Engineering, Southeast University
[2] Jiangsu Province Key Laboratory of Smart Low-Altitude Flight Management and Service, Nanjing Les Information Technology Co., Ltd., Nanjing, Jiangsu Province, China
Correspondence:
kongyouyong@seu.edu.cn, dhlshy2006@163.com

Abstract

Air traffic control (ATC) requires structured, safety-critical decision making whose effects unfold through aircraft trajectories, so text-level correctness alone is insufficient for evaluating large language models (LLMs) in this domain. We introduce ATC-BENCH, a simulation-grounded benchmark for ATC instruction generation with Chinese-facing scenario descriptions and controller-facing instruction fields. Given a structured Chinese traffic situation, a model generates a JSON control plan that is translated into BlueSky commands and evaluated through closed-loop simulation against separation, no-fly-zone (NFZ) avoidance, and route-recovery constraints. The benchmark spans four task families that form a controlled complexity gradient: single-aircraft NFZ avoidance, pairwise conflict resolution, joint conflict–airspace constraints, and multi-aircraft conflict graphs. Experiments with 11 representative LLM endpoints show that while most models produce format-valid instructions, task success varies widely across models and task families: the highest-scoring endpoint under our fixed protocol achieves 63.5% macro-average task success rate (TSR), while the two lowest endpoints fall below 25%, with the largest gaps appearing under joint constraints and multi-aircraft coordination. These results expose a persistent execution gap between static validity and simulation-passing behavior, motivating simulation-in-the-loop evaluation for LLM-based decision making in safety-critical aviation settings.

1 Introduction

Large language models (LLMs) have shown broad capabilities across general language tasks, mathematical reasoning, code generation, and multimodal inputs (Brown et al., 2020; Wei et al., 2022; Chen et al., 2021; OpenAI, 2023; Liang et al., 2023), and domain-specific benchmarks increasingly study high-stakes settings such as law (Guha et al., 2023), medicine (Han et al., 2024), and embodied decision making (Ichter et al., 2023). ATC decision making, however, imposes stricter requirements than open-domain tasks. Civil aviation scenarios are dynamic, constrained by operational rules and procedures, and inherently safety-critical (Endsley, 1995; Erzberger et al., 2012). A model must understand an evolving traffic situation while coordinating multiple objectives: aircraft separation, airspace restrictions, procedural compliance, and recovery to operational goals. Thus, a fluent response is insufficient; evaluation must check whether instructions remain executable and satisfy trajectory-level constraints in simulation. An ATC instruction can be syntactically valid and operationally plausible, yet still lead to loss of separation, restricted-airspace intrusion, or failed route recovery after execution.

Unlike general text reasoning, ATC instruction generation is a structured decision problem over executable actions, spanning single-aircraft no-fly-zone (NFZ) avoidance, two-aircraft conflict resolution, and coupled conflict–NFZ and multi-aircraft settings. The central challenge is not only to produce well-formed outputs, but also to satisfy constraints after closed-loop execution, requiring spatiotemporal reasoning, multi-objective trade-offs, and decisions grounded in execution outcomes.

Existing benchmarks for general reasoning, rule-based reasoning, and high-stakes evaluation (Chang et al., 2024; Zhou et al., 2025) do not fully capture ATC instruction generation because they lack aviation procedures, airspace constraints, and continuously evolving interactions. Prior aviation and UAV work mainly studies navigation, trajectory prediction, scenario generation, isolated task success, or low-level step-wise control (Wu et al., 2026; Sui et al., 2023; Kuchar and Yang, 2000). These settings are valuable, but they differ from sparse, event-triggered, high-level ATC interventions. They therefore do not directly evaluate

Findings of the Association for Computational Linguistics: EMNLP 2026, pages 33032–33055
October 24-29, 2026 ©2026 Association for Computational Linguistics

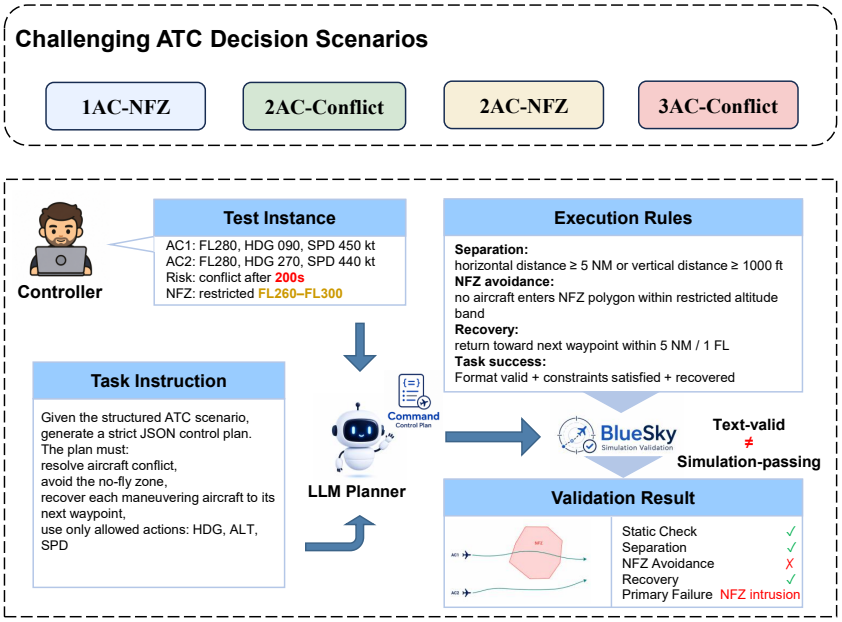


Figure 1: Overview of ATC-Bench. A structured Chinese ATC scenario is converted into a JSON control plan and evaluated with static checks plus BlueSky closed-loop simulation. The example highlights the core gap: static-valid instructions are not necessarily simulation-passing, and a text-valid plan may still fail trajectory-level validation through NFZ intrusion.

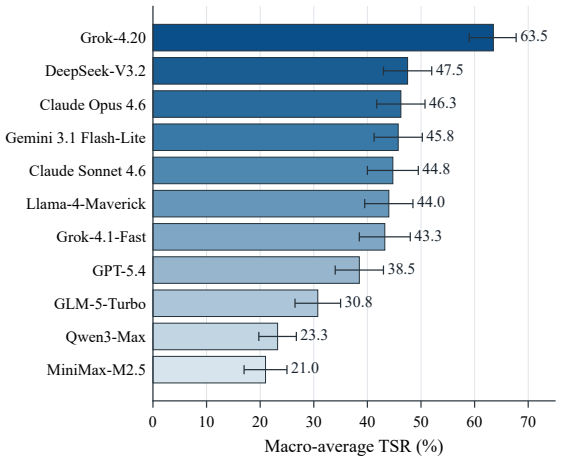


Figure 2: Macro-average task success rate (TSR, %) of the 11 evaluated LLMs on ATC-Bench. Error bars show parametric bootstrap 95% confidence intervals over the four fixed 100-sample task cells; detailed per-task results are reported in Table 2. Adjacent models with small overlapping intervals should be interpreted cautiously. Shortened display names are used in the plot for readability.

whether general-purpose LLMs can map structured ATC situations into machine-readable, controller-facing instructions whose downstream simulated trajectories jointly maintain separation, avoid restricted airspace, and recover to intended routes.

To address this gap, we introduce ATC-BENCH, a benchmark for evaluating LLM-based ATC instruction generation. ATC-Bench uses Chinese as the model-facing language to broaden the coverage of execution-grounded ATC evaluation while keeping the control interface machine-readable. This choice reflects the practical need to study ATC-oriented language understanding beyond English and complements the predominantly English-centric landscape of LLM evaluation. This design tests whether models can ground non-English structured operational descriptions into a language-independent executable control interface, while systematic bilingual comparison is left to future work. It formulates representative area-control problems as mappings from structured Chinese scenario descriptions to JSON control plans with Chinese controller-facing instruction and rationale fields. Rather than evaluating whether a model can produce a plausible textual response, ATC-BENCH evaluates whether a model can generate executable high-level plans that satisfy benchmark constraints after simulation. Model outputs are checked in a BlueSky-based closed-loop simulation (Hoekstra and Ellerbroek, 2016), which determines whether generated instructions maintain separation, avoid

restricted airspace, and recover to task goals. We emphasize that ATC-Bench is a synthetic simulation benchmark and does not capture the full complexity of real ATC operations (communication phraseology, sector workload, weather, or human-in-the-loop coordination).

Using this benchmark, we evaluate 11 representative general-purpose LLM endpoints. Figure 2 summarizes their macro-average task success rate (TSR) on ATC-Bench. The results show that many models can produce structurally valid control instructions with high consistency, yet still fail to complete the full loop of avoidance, constraint satisfaction, and route recovery in scenarios with coupled constraints and competing objectives. This finding suggests that a key limitation of current models lies not only in generating plausible instructions, but also in anticipating downstream trajectory consequences. The value of a civil aviation benchmark should therefore lie not only in comparing aggregate scores under a fixed protocol, but also in diagnosing capability boundaries and failure modes.

Our contributions are threefold:

- We introduce ATC-BENCH, a simulation-grounded benchmark for ATC instruction generation that unifies conflict resolution, restricted-airspace avoidance, and coupled multi-constraint planning within a single evaluation framework.
- We develop an evaluation pipeline coupling Chinese-facing scenario descriptions and controller-facing instruction fields with JSON action plans and BlueSky trajectory checking, moving assessment beyond text-level correctness toward simulation-grounded constraint satisfaction and task completion.
- We provide a systematic empirical study of 11 representative LLM endpoints, define a 300-sample official pool per task, and report main results on fixed stratified 100-sample evaluation subsets.

2 Related Work

LLM benchmarks increasingly target professional, rule-constrained, and high-stakes settings. General suites such as MMLU (Hendrycks et al., 2021), BIG-bench (BIG-bench authors, 2023), and HELM (Liang et al., 2023) established broad protocols, while LegalBench (Guha et al., 2023), VIVA+ (Hu et al., 2025), MedSafetyBench (Han et al., 2024), DeonticBench (Dou et al., 2026), and RuleArena (Zhou et al., 2025) emphasize domain rules and safety objectives. Most judgments remain textual or symbolic, whereas ATC responses can be schema-valid and rule-plausible while still causing loss of separation or restricted-airspace intrusion after execution.

Executable evaluation is closer to our setting: SWE-bench (Jimenez et al., 2024), AgentBench (Liu et al., 2024), WebArena (Zhou et al., 2024), and ToolLLM (Qin et al., 2024) evaluate outputs through downstream execution, while LLM-agent work studies reasoning-and-acting, tool use, and embodied planning (Yao et al., 2023; Schick et al., 2023; Ichter et al., 2023; Huang et al., 2023). Complementary safety benchmarks such as SafetyBench, TrustLLM, SAFEL, and α^3-Bench diagnose safety or robustness failures (Zhang et al., 2024; Huang et al., 2024; Son et al., 2025; Ferrag et al., 2026). However, these benchmarks do not evaluate high-level ATC control instructions whose correctness depends on future multi-aircraft trajectories, joint NFZ and separation constraints, and recovery behavior in an air-traffic simulator.

Aviation and ATC research provides mature conflict detection and resolution (CD&R) formulations and simulation tools, including conflict prediction, hybrid-system models, optimization, separation assurance, and deep-RL solvers (Kuchar and Yang, 2000; Paielli and Erzberger, 1997; Tomlin et al., 1998; Pallottino et al., 2002; Erzberger et al., 2012; Sui et al., 2023; Pham et al., 2022). Operational references such as ICAO Doc 4444 and FAA Order JO 7110.65 specify phraseology, separation services, vectoring, altitude assignment, and restricted-airspace procedures (International Civil Aviation Organization, 2016; Federal Aviation Administration, 2025), while corpora such as ATCOSIM and ATCO2 support empirical study of ATC communications (Hofbauer et al., 2008; Zuluaga-Gomez et al., 2022). ATC-Bench does not attempt to reproduce this full radiotelephony setting. Instead, it abstracts a small set of high-level control capabilities into executable JSON plans, complementing LLM-oriented aviation studies such as PilotBench (Wu et al., 2026), FlightGPT (Cai et al., 2025), and AirTrafficGen (Gould et al., 2025), which study trajectory prediction, UAV navigation, or scenario generation.

Task	Risk	Core capability
1AC-NFZ	NFZ	airspace avoidance
2AC-Conflict	LoS	pairwise separation
2AC-NFZ	LoS + NFZ	joint constraints
3AC-Conflict	multi-edge LoS	coordinated resolution

Table 1: ATC-Bench task families and the main capability tested by each family. All tasks additionally require recovery to the original navigation goal.

3 ATC-Bench

Given a structured Chinese traffic description, a model produces a high-level JSON control plan whose controller-facing instruction and rationale fields are also Chinese. The evaluator translates the plan into BlueSky commands (Hoekstra and Ellerbroek, 2016) and checks whether the resulting trajectory satisfies aircraft separation, NFZ avoidance, and recovery constraints, as illustrated in Figure 1.

3.1 Task Taxonomy

ATC-Bench contains four task families spanning single-aircraft spatial constraints to multi-aircraft coupled conflicts (Table 1). 1AC-NFZ tests restricted-airspace avoidance and recovery; 2AC-Conflict tests pairwise loss of separation (LoS; horizontal distance < 5 NM and vertical distance < 1000 ft); 2AC-NFZ combines LoS and NFZ-intrusion risks; and 3AC-Conflict uses connected multi-edge conflict graphs. These families are benchmark abstractions over control capabilities, not an official ICAO/FAA task taxonomy; Appendix Table 16 gives the scope mapping. We use flight level (FL), indicated airspeed (IAS), and closest point of approach (CPA) for altitude, speed, and conflict timing diagnostics. All tasks require executable control instructions rather than multiple-choice answers or free-text explanations.

3.2 Dataset Construction

Each task starts from 1,000 programmatically generated candidate scenarios. After static checks, BlueSky no-instruction risk validation, task-specific feasibility or consistency validation, and manual review, 300 official samples are retained and balanced across L1/L2/L3. Main experiments use the same stratified 100-sample subset for all models.

Scenario generation and validation. For conflict tasks, the generator samples relative positions, headings, speeds, and altitudes to create LoS risks, covering head-on, crossing, overtaking, and connected multi-edge geometries. For NFZ tasks, it samples polygon extents, centers, and altitude ranges so aircraft would enter restricted airspace without control. A BlueSky no-instruction trajectory τ_x^0 must contain the intended risk: LoS for 2AC-Conflict and 3AC-Conflict, NFZ entry within the restricted altitude band for 1AC-NFZ, and at least one of the two for 2AC-NFZ.

Feasibility filtering and difficulty. A deterministic reference and validation suite screens generated scenarios and constructs diagnostic baselines, but is not exposed to models, used as feedback, or included in the LLM comparison. Appendix Tables 13 and 14 report retained validation evidence and a bounded witness audit for the fixed evaluation subsets; the benchmark is a controlled simulation benchmark rather than a formal proof of real-world ATC solvability. Difficulty labels are diagnostic strata, assigned by task-specific scoring rules over urgency, constraint pressure, LoS–NFZ coupling, and recovery-related factors rather than by a single cross-task scalar. Appendix Table 15 gives the concrete scores and thresholds; the non-monotonic L1/L2 pattern for 2AC-Conflict (Table 6) suggests that geometry also affects empirical difficulty.

3.3 Quality Control

Automated checks cover field validity, no-instruction risk, task-specific feasibility or consistency validation, and re-simulation after fixes; manual review checks consistency among aircraft states, conflict pairs, NFZ geometry, Chinese descriptions, and BlueSky trajectories. Models see only task-facing information such as aircraft states, navigation goals, conflict pairs, predicted conflict times, NFZ polygons, and altitude ranges; hidden metadata is used for stratification, difficulty assignment, and failure attribution. The appendix provides a compact benchmark card and prompt/schema details.

3.4 Executable Evaluation Protocol

A model emits a task-specific JSON instruction plan in a single forward pass. The evaluator extracts the first parsable JSON block and normalizes task-specific legacy fields to the common execution schema in Table 19 before validation and execution. Under this normalized schema, AVOID actions use `HEADING`, `ALTITUDE`, `SPEED`, or `COMBINED`, while MAINTAIN and RECOVER carry no action block. Valid AVOID commands

become low-level BlueSky HDG/ALT/SPD commands, while RECOVER maps to direct/manual guidance toward the original waypoint. RECOVER is a deterministic execution primitive rather than an additional planner: the model must still decide which aircraft should recover and when, and recovery succeeds only if the preceding maneuver leaves the aircraft able to return within tolerance. Models receive no online execution feedback. Let f denote the model's instruction generator, so $y = f(x)$ is the JSON plan for scenario x. Let $\tau_{x,y}$ denote the BlueSky trajectory obtained by executing plan y on scenario x. Let $V(y)$, $S(\tau_{x,y})$, and $R(\tau_{x,y})$ denote instruction validity, trajectory safety, and task recovery. Sample success is

$$\mathrm{Succ}(x, y) = \mathbf{1}[V(y) \wedge S(\tau_{x,y}) \wedge R(\tau_{x,y})]. \quad (1)$$

Here, V requires parsable JSON and legal commands; S requires horizontal separation of at least 5 NM or vertical separation of at least 1000 ft for conflict tasks, and no entry into any NFZ polygon within the restricted altitude band for NFZ tasks; and R requires the relevant aircraft to end within 5 NM of the target waypoint and within 1 FL of the target altitude. This deliberately conjunctive definition reflects the benchmark goal: a plan is not successful if it is well-formed but fails safety or recovery after execution. Let $\mathcal{D}_{\mathrm{eval}}$ denote the fixed stratified 100-sample evaluation subset for the corresponding task. The task success rate is

$$\mathrm{TSR} = \frac{100}{|\mathcal{D}_{\mathrm{eval}}|} \sum_{x \in \mathcal{D}_{\mathrm{eval}}} \mathrm{Succ}(x, f(x)). \quad (2)$$

Beyond TSR, ATC-Bench reports format-valid rate, separation safety, NFZ avoidance, recovery, operational cost on successful samples, and a mutually exclusive primary failure type assigned by priority: format failure, separation failure, NFZ intrusion, and not recovered. Operational cost is computed only on successful samples and operationalized as extra flown distance relative to the evaluator's reference route. For a successful plan $y = f(x)$, we compute

$$C(x, y) = \sum_{i \in \mathcal{A}(x)} \left[L_i(\tau_{x,y}) - L_i^{\mathrm{ref}}(x) \right], \quad (3)$$

where $\mathcal{A}(x)$ is the evaluated aircraft set, $L_i(\tau_{x,y})$ is the flown distance after executing y, and $L_i^{\mathrm{ref}}(x)$ is the evaluator reference-route distance. We report the conditional mean

$$\bar{C}_f = \frac{\sum_{x \in \mathcal{D}_{\mathrm{eval}}} \mathbf{1}[\mathrm{Succ}(x, f(x))] C(x, f(x))}{\sum_{x \in \mathcal{D}_{\mathrm{eval}}} \mathbf{1}[\mathrm{Succ}(x, f(x))]}, \quad (4)$$

so cost compares the efficiency of valid solutions rather than penalizing already-failed trajectories twice. The primary-failure priority is an accounting convention rather than a severity ranking; non-exclusive staged pass rates are reported separately to expose co-occurring violations.

4 Experiments

The experiments are organized around four research questions. RQ1 asks how reliable current models are under the benchmark protocol; RQ2 measures the gap between static instruction validity and simulation-passing execution; RQ3 evaluates difficulty effects; and RQ4 examines model-specific execution profiles and operational trade-offs across safety, recovery, intervention timing, and operational cost.

4.1 Experimental Setup

Tasks and models. Each task uses a fixed stratified 100-sample evaluation subset drawn from the 300-sample official pool, preserving L1/L2/L3 coverage and major scenario structures. The full 300-sample pools are retained for broader follow-up evaluation. Across four tasks, the expected number of model–sample pairs is $11 \times 4 \times 100 = 4400$. We report TSR over these requested samples: missing generations, invalid instructions, and missing evaluation records are counted as failures. We evaluate 11 general-purpose LLMs: Claude Opus 4.6, Claude Sonnet 4.6, DeepSeek-V3.2, Gemini 3.1 Flash-Lite Preview, GLM-5-Turbo, GPT-5.4, Grok-4.1-Fast, Grok-4.20 Beta, Llama-4-Maverick, MiniMax-M2.5, and Qwen3-Max. We treat each hosted model as an endpoint snapshot rather than a stable model family. Within each task, all endpoints receive the same scenario inputs and task prompt, and all outputs are evaluated by the same deterministic evaluator after normalization to the common execution schema. The final runs use an OpenRouter-compatible API with temperature 0.2, top-p 0.8, and a 4000-token budget; exact endpoint IDs are in Appendix Table 12.

Baselines, metrics, and uncertainty. We also include deterministic diagnostic baselines, separate from the 11-model comparison. No instruction, fixed altitude change, and fixed speed change

Model	Task-family TSR (%)				Overall
	1AC-NFZ ↑	2AC-Conf. ↑	2AC-NFZ ↑	3AC-Conf. ↑	Avg. ↑
General-purpose LLMs, ordered by macro-average TSR					
Grok-4.20 Beta	83	82	45	44	63.5
DeepSeek-V3.2	78	39	31	42	47.5
Claude Opus 4.6	46	76	24	39	46.3
Gemini 3.1 Flash-Lite Preview	76	50	34	23	45.8
Claude Sonnet 4.6	56	58	41	24	44.8
Llama-4-Maverick	67	54	22	33	44.0
Grok-4.1-Fast	63	41	41	28	43.3
GPT-5.4	58	42	15	39	38.5
GLM-5-Turbo	55	25	10	33	30.8
Qwen3-Max	59	13	19	2	23.3
MiniMax-M2.5	30	20	21	13	21.0
Deterministic diagnostic baselines, not included in endpoint comparison					
Reference policy	57	35	87	30	52.3
No instruction	0	0	0	0	0.0
Fixed altitude	29	18	68	17	33.0
Fixed speed	0	2	0	1	0.8

Table 2: Main results on ATC-Bench. Each task-family column reports task success rate (TSR, %), and Avg. reports the macro-average across the four task families. LLMs are ordered by macro-average TSR, with deterministic diagnostic baselines reported separately; Figure 2 visualizes macro-TSR uncertainty, and small adjacent differences should be interpreted cautiously.

serve as sanity checks, while a stronger reference policy uses task-specific deterministic maneuvers and recovery timing to anchor solvability under the same BlueSky evaluator. We additionally run BlueSky's built-in Airborne Separation Assurance System with the Modified Voltage Potential method (ASAS-MVP) on the two conflict-only task families as a specialized CD&R diagnostic. These baselines do not generate Chinese instructions and should not be interpreted as language-model competitors; they instead check task risk, evaluator behavior, and the headroom left by fixed-prompt LLM outputs. We therefore interpret low LLM TSR as evidence of limited fixed-prompt instruction planning under the benchmark protocol, rather than as a claim about absolute ATC solvability. TSR requires format validity, trajectory-level constraint satisfaction, and recovery; secondary diagnostics include staged pass rates, primary failure types, operational cost on successful samples, and observed API latency (Liang et al., 2023; Zheng et al., 2023). The baselines are deliberately simple and deterministic so that their failures remain interpretable: they usually indicate that a single-axis maneuver or fixed recovery schedule is insufficient under the evaluator, rather than stochastic search variance. Conversely, when a baseline succeeds, we treat it as evidence that the scenario admits at least one simple executable policy within the benchmark action space, not as evidence of operational optimality. Because each model–task cell has 100 requested samples, differences of a few percentage points should be interpreted cautiously; adjacent models with small TSR gaps should not be strictly ranked. For Figure 2, we estimate uncertainty with a parametric bootstrap over the four fixed 100-sample task cells.

4.2 RQ1: Overall Task Success

Table 2 shows that even the highest-scoring endpoint remains far from reliable under our fixed protocol. Grok-4.20 Beta obtains the highest macro-average TSR, but still leaves a large fraction of scenarios unresolved after the full validity–constraint–recovery check. Given the uncertainty implied by the fixed sample size, the clearest cross-model result is the broad separation between the leading model and the lower-performing group, while many adjacent differences should be treated as suggestive rather than definitive. Task structure also matters: single-aircraft NFZ avoidance is generally easier, whereas joint NFZ–traffic constraints and multi-aircraft conflict graphs are substantially harder. Performance profiles are task-dependent as well: Claude Opus 4.6 is comparatively strong on pairwise conflict resolution but weaker on NFZ-related tasks, while DeepSeek-V3.2 shows the opposite tendency.

Interpreting endpoint comparisons. The main table is a fixed-protocol endpoint comparison, not a prompt-invariant or general LLM ordering. The gap is not mainly JSON validity: models usu-

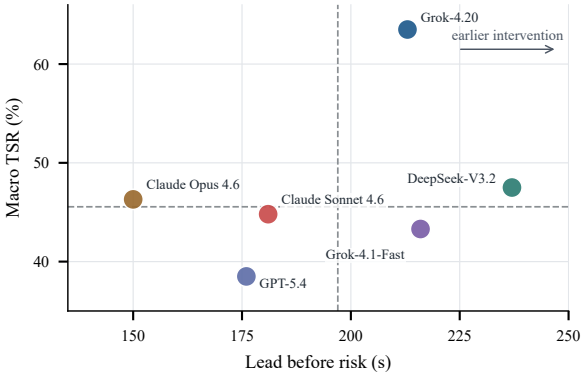


Figure 3: Control-policy diagnostics for representative models. The x-axis reports lead time, $t_{\text{risk}} - t_{\text{first}}$, where t_{first} is the first effective maneuver time and t_{risk} is the earliest risk event; the y-axis reports fixed-denominator macro TSR from Table 2. The plot is a representative diagnostic pattern, not a causal claim or standalone ranking; shortened display names are used for readability.

ally produce executable structured outputs, but differ after BlueSky execution in trigger timing, maneuver axis/magnitude, and recovery timing. Per-sample diagnostics suggest that Grok-4.20 Beta more often uses earlier, more assertive combined lateral/vertical maneuvers, while GPT-5.4 and Claude variants more often produce schema-valid but conservative or single-axis plans that fail NFZ avoidance, separation, or recovery. Figure 3 visualizes this representative diagnostic pattern: higher-scoring displayed models are associated with earlier interventions and, for Grok-4.20 Beta and DeepSeek-V3.2, more frequent combined maneuvers. This is an observational pattern rather than a causal claim about why a model succeeds.

Prompt-sensitivity check. To probe prompt dependence, we rerun fixed 50-scenario subsets from 2AC-NFZ and 3AC-Conflict for GPT-5.4, Claude Sonnet 4.6, and Qwen3-Max with an added directive encouraging earlier, larger, and multi-axis maneuvers, observing TSR gains of 4.0–20.0 points (Appendix Table 8). All endpoints in the main evaluation use one identical fixed prompt, so no endpoint receives bespoke or tuned instructions. We nevertheless cannot rule out that a single fixed prompt fits some models better than others, and we therefore leave prompt-averaged or prompt-invariant comparison to future work. Residual execution failures remain under the directive prompt, so the execution gap is not reducible to prompt wording or schema following.

We also report a repeated-decoding probe for the endpoint groups discussed during review (Appendix Table 9). Hosted APIs do not expose inference seeds, so this analysis measures repeated decoding under fixed serving parameters rather than controlled seed variance. We use it only as an auxiliary robustness check; the main conclusions do not rely on re-ordering endpoints by this smaller probe, and small within-group differences should not be interpreted as strict orderings.

Baseline	Task	Sep. ↑	Arr. ↑
ASAS-MVP	2AC-Conflict	100/100	90/100
ASAS-MVP	3AC-Conflict	97/100	29/100

Table 3: Conflict-only CD&R diagnostic baseline. MVP is evaluated as BlueSky's internal separation-assurance module, not as a Chinese instruction generator; Sep. reports LoS-free execution and Arr. reports arrival/recovery under the ATC-Bench monitor.

The deterministic baselines further clarify the missing capability. No instruction confirms that the subsets contain real execution risks; fixed-altitude is competitive on 2AC-NFZ, where many cases admit a simple vertical separation strategy, but is much less robust on single-aircraft NFZ avoidance and multi-aircraft conflict resolution. The reference policy improves the macro average to 52.3% and reaches 87% on 2AC-NFZ, showing that substantial headroom exists when task-specific maneuvers and recovery timing are hard-coded. It still falls short on conflict-only and three-aircraft cases, so it is a diagnostic anchor rather than an oracle or a solvability certificate.

Table 3 further anchors the conflict subproblem. ASAS-MVP nearly solves separation on both conflict-only families, but its arrival/recovery rate drops sharply on 3AC-Conflict. ATC-Bench is therefore not reducible to separation assurance alone. Current LLMs often fail not because no collision-avoidance action exists, but because full benchmark success also requires choosing executable maneuvers that preserve downstream route completion.

4.3 RQ2: Execution Gap and Failure Attribution

Figure 4 shows that format validity remains high after missing generations are counted as logic/format failures, while final task success is much lower. The largest gaps are constraint-specific: 2AC-NFZ is dominated by NFZ avoidance failures despite

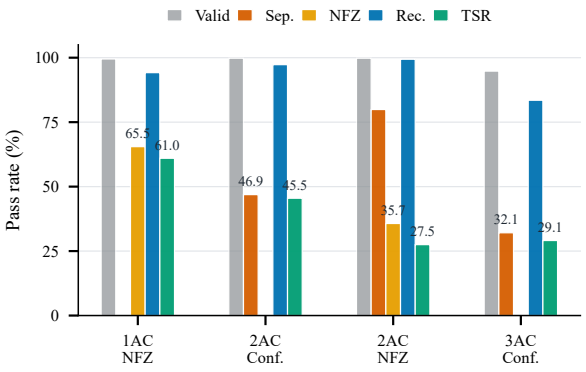


Figure 4: Stage-wise execution gap aggregated over the 11 LLMs, excluding deterministic baselines. Bars compare validity, applicable safety checks, recovery, and final TSR. Staged pass rates are non-exclusive diagnostics, whereas TSR requires all applicable checks to pass.

otherwise well-formed outputs, and 3AC-Conflict is dominated by separation failures under multi-aircraft interaction. These gaps show that many failures are not formatting errors. Instead, failures arise because models do not reliably anticipate the trajectory-level consequences of instructions that appear valid.

Task	Succ.	Logic	Sep.	NFZ	Rec.
1AC-NFZ	61.0	0.5	–	**34.1**	4.5
2AC-Conflict	45.5	0.2	**52.9**	–	1.5
2AC-NFZ	27.5	0.2	19.9	**52.2**	0.2
3AC-Conflict	29.1	5.2	**62.7**	–	3.0

Table 4: Task-level outcome composition. Cells report percentages over requested model–sample pairs; bold marks the dominant primary failure type under the accounting priority in Section 3.4.

Model	Succ.	Logic	Sep.	NFZ	Rec.
Grok-4.20 Beta	63.5	–	**27.8**	8.0	0.8
DeepSeek-V3.2	47.5	0.5	**36.0**	14.8	1.3
Claude Opus	46.3	1.5	20.5	**30.3**	1.5
Gemini Flash-Lite	45.8	–	**34.0**	19.3	1.0
Claude Sonnet	44.8	–	**28.5**	17.3	9.5
Llama-4-Maverick	44.0	0.3	**33.8**	20.5	1.5
Grok-4.1-Fast	43.3	6.5	**28.3**	21.8	0.3
GPT-5.4	38.5	–	**32.8**	22.5	6.3
GLM-5-Turbo	30.8	1.5	**38.8**	27.8	1.3
Qwen3-Max	23.3	0.3	**51.0**	24.5	1.0
MiniMax-M2.5	21.0	6.0	**41.5**	30.8	0.8

Table 5: Model-level outcome composition. Cells report percentages over 400 requested samples per model; bold marks the dominant primary failure type under the accounting priority in Section 3.4. Logic covers invalid outputs or unavailable records.

Primary failure modes. Tables 4 and 5 decompose requested samples into success and mutually exclusive primary failure types. At the task level, NFZ-only scenarios are dominated by restricted-airspace intrusions, conflict-only scenarios are dominated by separation failures, and joint NFZ–traffic scenarios expose both failure modes. At the model level, lower-scoring endpoints tend to accumulate more separation failures, while NFZ intrusions remain substantial even for mid-scoring endpoints; recovery failures are more localized. This view complements Table 2 by showing what replaces success when an instruction fails after execution.

Task	L1	L2	L3
1AC-NFZ	72.7	63.9	46.0
2AC-Conflict	46.8	52.6	37.2
2AC-NFZ	36.1	30.0	16.3
3AC-Conflict	38.0	32.0	17.6

Table 6: TSR (%) by difficulty level.

4.4 RQ3: Effects of Difficulty

Table 6 shows clear declines as difficulty increases. L1/L2/L3 are within-task labels and do not imply equal absolute difficulty across tasks, and the fixed-size slices should be read as diagnostic trends rather than precise monotonic laws. They still capture meaningful increases in execution risk for most task families. The decline is especially visible for NFZ-constrained and multi-aircraft settings, where higher urgency and tighter recovery requirements leave less margin for conservative or poorly timed maneuvers. The main exception is 2AC-Conflict, whose non-monotonic pattern reflects an interaction between the TTC strata, CPA margin, encounter geometry, and model maneuver choices. Appendix Table 10 reports the corresponding stratification.

4.5 RQ4: Execution Profiles and Operational Trade-offs

Figure 5 complements the TSR table by separating execution behavior from aggregate success. We select representative endpoints spanning high-scoring, mid-scoring, and policy-diverse profiles rather than treating the plot as a second ranking. Because all axes are min–max normalized over the displayed models only, the plot should be read as a relative profile comparison within this subset. Under this normalization, Grok-4.20 Beta combines high NFZ avoidance, reliable recovery, timely intervention, and moderate cost efficiency; Claude Opus 4.6 is stronger on separation but weaker on NFZ avoidance and timing; and Claude Sonnet 4.6 has

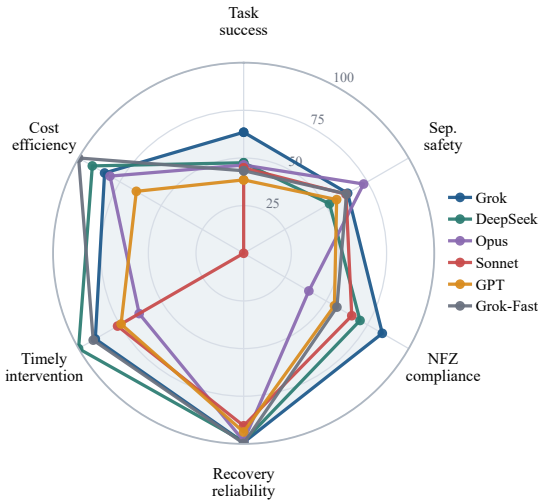


Figure 5: Execution profiles of representative models. Axes are min–max normalized over displayed models so larger values indicate better behavior, covering task success, separation safety, NFZ avoidance, recovery, intervention timing, and inverse-normalized cost efficiency. This plot is intended for qualitative profile comparison, not as a second ranking; shortened display names are used for readability.

poor cost efficiency despite competitive TSR. This profile view shows that aggregate TSR alone hides distinct execution trade-offs across separation, NFZ avoidance, timing, recovery, and operational cost.

5 Conclusion

We introduced ATC-BENCH to test whether LLM-generated ATC instructions remain valid after execution, not merely after JSON parsing. Across four task families, models frequently produce plausible control plans that fail once BlueSky trajectories must jointly satisfy separation, NFZ avoidance, and recovery. The main takeaway is therefore narrow but important: static correctness is only an entry condition for control instructions in dynamic safety-critical domains. Progress on ATC-oriented LLMs should be measured through executable trajectory consequences, especially since specialized baselines show that some subproblems are solvable by simpler rule-based policies.

Limitations

ATC-Bench is a diagnostic benchmark, not a deployment validation tool: BlueSky cannot replace ATC procedures, workload modeling, certification, independent controller adjudication, or expert review. It covers high-level structured Chinese instructions, not speech, human–AI workflows, or real-time coordination. Endpoint results are fixed-prompt snapshots; directive prompting changes TSR, and each model–task cell uses only 100 requested samples, so adjacent differences have limited statistical resolution. Results also reflect a single evaluation language without systematic English or bilingual comparison, finite JSON actions, deterministic HDG/ALT/SPD/DCT mapping, no online simulator feedback to models, and a shared 4000-token budget. The deterministic reference policy, ASAS-MVP baseline, and bounded witness audit anchor solvability for the fixed evaluation subsets. In particular, all 400 instances in the fixed evaluation subsets have at least one explicit passing witness under the benchmark evaluator, providing an evaluator-level constructive solvability certificate for these fixed subsets but not a proof of real-world ATC solvability. These witnesses are not optimized controller comparisons, and they do not replace future evaluations with richer CD&R, planning, or controller-designed policies. Code, benchmark data, prompt artifacts, schemas, deterministic evaluators, and the bundled BlueSky configuration are available at https://github.com/caijinjun/ATC-Bench, subject to provider log-sharing constraints.

Ethical Considerations

This work concerns a safety-critical domain, and the results should not be interpreted as evidence that LLMs are ready for operational ATC. ATC-Bench is designed to diagnose model behavior under synthetic simulation scenarios, not to recommend real controller actions or certify autonomous ATC systems. The benchmark does not rely on personal data, passenger records, or real controller communications. Scenarios are programmatically generated and evaluated in BlueSky. Nevertheless, high benchmark scores could be misused to overstate the safety of language-model-based decision support. We therefore report staged safety metrics, failure modes, operational cost, latency, and limitations alongside task success. We also emphasize that any real-world use would require certified procedures, expert oversight, human-in-the-loop validation, and regulatory review.

Acknowledgments

This work is supported by the National Key Research and Development Program of China (2024YFB4303805). We would like to thank the

reviewers for their constructive and insightful suggestions.

References

BIG-bench authors. 2023. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research (TMLR)*.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 1877–1901.

Hengxing Cai, Jinhan Dong, Jingjun Tan, Jingcheng Deng, Sihang Li, Zhifeng Gao, Haidong Wang, Zicheng Su, Agachai Sumalee, and Renxin Zhong. 2025. FlightGPT: Towards generalizable and interpretable UAV vision-and-language navigation with vision-language models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6659–6676.

Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, Wei Ye, Yue Zhang, Yi Chang, Philip S. Yu, Qiang Yang, and Xing Xie. 2024. A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology*, 15(3):1–45.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, and 39 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Guangyao Dou, Luis Brena, Akhil Deo, William Jurayj, Jingyu Zhang, Nils Holzenberger, and Benjamin Van Durme. 2026. DeonticBench: A benchmark for reasoning over rules. *arXiv preprint arXiv:2604.04443*.

Mica R. Endsley. 1995. Toward a theory of situation awareness in dynamic systems. *Human Factors*, 37(1):32–64.

Heinz Erzberger, Todd A. Lauderdale, and Yung-Cheng Chu. 2012. Automated conflict resolution, arrival management and weather avoidance for air traffic management. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, 226(8):930–949.

Federal Aviation Administration. 2025. *Air Traffic Control, Order JO 7110.65BB*. Federal Aviation Administration. Issued February 20, 2025; public order with later changes.

Mohamed Amine Ferrag, Abderrahmane Lakas, and Merouane Debbah. 2026. α^3-Bench: A unified benchmark of safety, robustness, and efficiency for LLM-based UAV agents over 6G networks. *arXiv preprint arXiv:2601.03281*.

Dewi Sid William Gould, George De Ath, Ben Carvell, and Nick Pepper. 2025. AirTrafficGen: Configurable air traffic scenario generation with large language models. *arXiv preprint arXiv:2508.02269*.

Neel Guha, Julian Nyarko, Daniel E. Ho, Christopher Ré, Adam Chilton, Aditya Narayana, Alex Chohlas-Wood, Austin Peters, Brandon Waldon, Daniel N. Rockmore, Diego Zambrano, Dmitry Talisman, Enam Hoque, Faiz Surani, Frank Fagan, Galit Sarfaty, Gregory M. Dickinson, Haggai Porat, Jason Hegland, and 21 others. 2023. LegalBench: A collaboratively built benchmark for measuring legal reasoning in large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Tessa Han, Aounon Kumar, Chirag Agarwal, and Himabindu Lakkaraju. 2024. MedSafetyBench: Evaluating and improving the medical safety of large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *International Conference on Learning Representations (ICLR)*.

Jacco M. Hoekstra and Joost Ellerbroek. 2016. BlueSky ATC simulator project: An open-data and open-source approach. In *Proceedings of the 7th International Conference on Research in Air Transportation (ICRAT)*.

Konrad Hofbauer, Stefan Petrik, and Horst Hering. 2008. The ATCOSIM corpus of non-prompted clean air traffic control speech. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC)*.

Zhe Hu, Yixiao Ren, Guanzhong Liu, Jing Li, and Yu Yin. 2025. VIVA+: Human-centered situational decision-making. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 17420–17437.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Tomas Jackson, Noah Brown, Linda Luu, Sergey Levine, Karol Hausman, and Brian Ichter. 2023. Inner monologue: Embodied reasoning through planning with language models. In *Proceedings of the 6th Conference on Robot Learning (CoRL)*, pages 1769–1782.

Yue Huang, Lichao Sun, Haoran Wang, Siyuan Wu, Qihui Zhang, Yuan Li, Chujie Gao, Yixin Huang, Wenhan Lyu, Yixuan Zhang, Xiner Li, Hanchi Sun, Zhengliang Liu, Yixin Liu, Yijue Wang, Zhikun Zhang, Bertie Vidgen, Bhavya Kailkhura, Caiming Xiong, and 52 others. 2024. Position: TrustLLM: Trustworthiness in large language models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 20166–20270.

Brian Ichter, Anthony Brohan, Yevgen Chebotar, Chelsea Finn, Karol Hausman, Alexander Herzog, Daniel Ho, Julian Ibarz, Alex Irpan, Eric Jang, Ryan Julian, Dmitry Kalashnikov, Sergey Levine, Yao Lu, Carolina Parada, Kanishka Rao, Pierre Sermanet, Alexander T. Toshev, Vincent Vanhoucke, and 26 others. 2023. Do as I can, not as I say: Grounding language in robotic affordances. In *Proceedings of the 6th Conference on Robot Learning (CoRL)*, pages 287–318.

International Civil Aviation Organization. 2016. *Procedures for Air Navigation Services: Air Traffic Management, Doc 4444*, 16th edition. International Civil Aviation Organization.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. 2024. SWE-bench: Can language models resolve real-world GitHub issues? In *International Conference on Learning Representations (ICLR)*.

James K. Kuchar and Lee C. Yang. 2000. A review of conflict detection and resolution modeling methods. *IEEE Transactions on Intelligent Transportation Systems*, 1(4):179–189.

Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, Benjamin Newman, Binhang Yuan, Bobby Yan, Ce Zhang, Christian Cosgrove, Christopher D. Manning, Christopher Ré, Diana Acosta-Navas, Drew A. Hudson, and 31 others. 2023. Holistic evaluation of language models. *Transactions on Machine Learning Research (TMLR)*.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, and 3 others. 2024. AgentBench: Evaluating LLMs as agents. In *International Conference on Learning Representations (ICLR)*.

OpenAI. 2023. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*.

Russell A. Paielli and Heinz Erzberger. 1997. Conflict probability estimation for free flight. *Journal of Guidance, Control, and Dynamics*, 20(3):588–596.

Lucia Pallottino, Eric M. Feron, and Antonio Bicchi. 2002. Conflict resolution problems for air traffic management systems solved with mixed integer programming. *IEEE Transactions on Intelligent Transportation Systems*, 3(1):3–11.

Duc-Thinh Pham, Phu N. Tran, Sameer Alam, Vu Duong, and Daniel Delahaye. 2022. Deep reinforcement learning based path stretch vector resolution in dense traffic with uncertainties. *Transportation Research Part C: Emerging Technologies*, 135:103463.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2024. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *International Conference on Learning Representations (ICLR)*.

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. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Yejin Son, Minseo Kim, Sungwoong Kim, Seungju Han, Jian Kim, Dongju Jang, Youngjae Yu, and Chan Young Park. 2025. Subtle risks, critical failures: A framework for diagnosing physical safety of LLMs for embodied decision making. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 25692–25733.

Dong Sui, Chenyu Ma, and Chunjie Wei. 2023. Tactical conflict solver assisting air traffic controllers using deep reinforcement learning. *Aerospace*, 10(2):182.

Claire Tomlin, George J. Pappas, and Shankar Sastry. 1998. Conflict resolution for air traffic management: A study in multi-agent hybrid systems. *IEEE Transactions on Automatic Control*, 43(4):509–521.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pages 24824–24837.

Yalun Wu, Haotian Liu, Zhoujun Li, and Boyang Wang. 2026. PilotBench: A benchmark for general aviation agents with safety constraints. *arXiv preprint arXiv:2604.08987*.

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

Zhexin Zhang, Leqi Lei, Lindong Wu, Rui Sun, Yongkang Huang, Chong Long, Xiao Liu, Xuanyu Lei, Jie Tang, and Minlie Huang. 2024. SafetyBench: Evaluating the safety of large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15537–15553.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging LLM-as-a-judge with MT-Bench and chatbot arena. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Ruiwen Zhou, Wenyue Hua, Liangming Pan, Sitao Cheng, Xiaobao Wu, En Yu, and William Yang Wang. 2025. RuleArena: A benchmark for rule-guided reasoning with LLMs in real-world scenarios. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 550–572.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. 2024. WebArena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations (ICLR)*.

Juan Zuluaga-Gomez, Karel Veselý, Igor Szöke, Alexander Blatt, Petr Motlicek, Martin Kocour, Mickael Rigault, Khalid Choukri, Amrutha Prasad, Seyyed Saeed Sarfjoo, Iuliia Nigmatulina, Claudia Cevenini, Pavel Kolčárek, Allan Tart, Jan Černocký, and Dietrich Klakow. 2022. ATCO2 corpus: A large-scale dataset for research on automatic speech recognition and natural language understanding of air traffic control communications. *arXiv preprint arXiv:2211.04054*.

A Additional Diagnostic Tables

This appendix keeps only the diagnostic tables that directly support the main text or reproducibility. Detailed missing-record audits and engineering checklists are omitted from the paper to keep the appendix focused.

In Table 7, Total denotes the fixed 1100 requested model–sample pairs per task. Logic/fmt. includes invalid outputs, missing generations, and missing evaluation records. The remaining columns decompose requested records into success and mutually exclusive primary failure types. Failed samples are assigned using the primary failure priority described in Section 3.4; this priority is an accounting convention, not a severity ranking.

Task	Total	Success	Logic/fmt.	Sep.	NFZ	Rec. fail
1AC-NFZ	1100	671	5	–	375	49
2AC-Conflict	1100	500	2	582	–	16
2AC-NFZ	1100	303	2	219	574	2
3AC-Conflict	1100	320	57	690	–	33

Table 7: Primary failure-type counts over requested records.

Table 8 reports the targeted prompt-sensitivity check used to probe whether control-style instructions change execution outcomes. The directive prompt explicitly encourages earlier, larger, and multi-axis maneuvers. It improves TSR on all tested model–task pairs, but residual failures remain substantial, so we treat it as diagnostic evidence rather than a replacement fixed-protocol comparison.

Model	TSR	Sep.	NFZ	Rec.	Comb.
2AC-NFZ					
GPT-5.4	14/26/+12	92	36	74	86
Sonnet 4.6	38/54/+16	96	76	70	38
Qwen3-Max	14/34/+20	82	64	82	100
3AC-Conflict					
GPT-5.4	44/60/+16	80	–	74	100
Sonnet 4.6	28/32/+4	56	–	64	50
Qwen3-Max	4/16/+12	60	–	32	100

Table 8: Prompt-sensitivity check on fixed 50-scenario subsets. The TSR column reports Base/Dir./Δ under the main prompt and the early/larger/combined maneuver directive. Metrics are percentages. Sep., Rec., and Comb. denote separation pass, recovery pass, and samples containing at least one combined maneuver. Model names are shortened for readability.

Table 9 reports the repeated-decoding probe requested during review for the three highest- and three lowest-scoring endpoints by macro-average TSR. The probe uses a fixed 50-scenario subset of the 2AC-Conflict evaluation set; each row aggregates three separate hosted-API decodes under the same temperature, top-p, token budget, evaluator, and fixed denominator. Because the APIs do not expose inference seeds, the table measures repeated-decoding variability rather than seed-controlled variance. The probe is reported as an auxiliary check on sampling variability, not as a replacement ranking or prompt-invariant endpoint ordering.

Group	Endpoint	Repeat TSR	Main TSR
Top	Grok-4.20 Beta	78.7 ± 1.9	82
Top	DeepSeek-V3.2	45.3 ± 4.1	39
Top	Claude Opus 4.6	73.3 ± 2.5	76
Bottom	MiniMax-M2.5	21.3 ± 3.4	20
Bottom	Qwen3-Max	15.3 ± 5.2	13
Bottom	GLM-5-Turbo	20.7 ± 2.5	25

Table 9: Repeated-decoding variability on a fixed 50-scenario 2AC-Conflict robustness subset. Repeat TSR is mean $\pm$ sample standard deviation over three independent decodes. Main TSR reports the corresponding 100-sample fixed-protocol 2AC-Conflict result from Table 2 for context only; this auxiliary probe is not used to infer a prompt-invariant ordering.

The repeated-decoding standard deviations range from 1.9 to 5.2 TSR points. The lowest top-group mean, DeepSeek-V3.2 at 45.3%, remains above the highest bottom-group mean, MiniMax-M2.5 at 21.3%, supporting only coarse group separation; within-group differences remain too small for strict ordering.

Table 10 expands the 2AC-Conflict difficulty analysis. The L1/L2/L3 bins are TTC strata, not guaranteed monotone empirical-difficulty bins; the observed L1/L2 inversion is consistent with interaction among TTC, CPA margin, encounter geometry, and model control policy.

Stratum	n	TTC	CPA	Macro TSR
L1 ($\geq$ 300s)	33	360.7	1.543	46.8
L2 (120–299s)	33	204.0	1.959	52.6
L3 (60–119s)	34	88.5	1.777	37.2

Table 10: 2AC-Conflict stratification by TTC difficulty bin. TTC is in seconds, CPA in nautical miles, and macro TSR is averaged over the 11 endpoints. Native encounter mixes for L1/L2/L3 are Crossing/Head-on/Overtaking = 14/14/5, 11/11/11, and 11/11/12.

Table 11 reports observed end-to-end API latency in seconds. Avg. is the average request-to-response generation time per scenario across four tasks. Median and P90 are computed from tasks with per-request logs: 1AC-NFZ, 2AC-NFZ, and

3AC-Conflict. These values are diagnostic measurements under our API-serving conditions, not hardware-normalized model-speed estimates.

Model	Avg.	Median	P90
Grok-4.20 Beta	3.2	3.2	4.8
Gemini 3.1 Flash-Lite Preview	3.8	2.7	7.0
Llama-4-Maverick	4.6	4.2	6.2
GPT-5.4	6.4	6.5	7.9
Claude Opus 4.6	8.7	8.6	11.2
Claude Sonnet 4.6	10.4	9.0	19.2
Qwen3-Max	12.8	14.4	16.3
DeepSeek-V3.2	17.0	15.4	22.2
Grok-4.1-Fast	30.2	24.4	41.9
MiniMax-M2.5	42.9	24.8	102.0
GLM-5-Turbo	49.7	44.0	102.7

Table 11: Observed end-to-end API latency in seconds.

Table 12 records the exact endpoint identifiers found in the archived May 2026 request logs for the 11 reported LLMs. Display names in the main text are shortened for readability, while the endpoints below define the experimental identity used for reproduction. Because many hosted models are proprietary and time-varying, the results should be interpreted as endpoint snapshots under the logged serving conditions rather than timeless model-family measurements.

B Benchmark Card and Reproducibility Details

This appendix provides a compact benchmark card for the reported experiments. It is intended to make the evaluation interface understandable within the paper, including the task construction, prompts, schemas, endpoint snapshots, and simulation settings needed to interpret the reported results. Archived run logs were used internally as the source of endpoint and runtime details.

B.1 Construction and Review

Each task starts from 1,000 programmatically generated candidate scenarios. Automated filtering checks field validity, validates that a no-instruction trajectory contains the intended risk, and applies task-specific feasibility or consistency checks. These reference and validation procedures include heading, altitude, speed, and combined maneuver families when available, but they are not shown to models, not used as online feedback, and not used as baselines in the main endpoint comparison. They serve as diagnostic evidence and consistency checks, not as an oracle or a formal certificate that every retained scenario is absolutely solvable. The validation reference used for scenario filtering is separate from the diagnostic Reference policy reported in Table 2. After filtering and balancing, all 300 official samples in each task family are manually reviewed for consistency among structured states, Chinese descriptions, risk visibility in BlueSky, family and difficulty labels, recovery-target clarity, and absence of reference-maneuver leakage. The final evaluation uses the fixed stratified 100-sample subset described in Section 4.1.

Table 14 summarizes the constructive witness audit used to interpret absolute TSR. An evaluated instance is counted as covered when at least one explicit command sequence in the benchmark action space passes the same static checks, BlueSky configuration, and safety/recovery evaluator. Endpoint successes are post-hoc witnesses only; they are not used for training, filtering, model selection, or score adjustment. The bounded search adds heading, altitude, speed, combined maneuvers, and recovery timing within the benchmark action bounds. We use these witnesses as evaluator-level constructive evidence for the fixed subsets rather than as evidence of real-world ATC feasibility.

Our construction records include static field checks, no-instruction risk validation, reference/consistency checks where available, constructive witness records for the fixed evaluation subsets, and author manual review records. The current version does not include independent controller adjudication, which is noted as a limitation for operational claims.

B.2 Input–Output Interface

Models receive a structured Chinese ATC scenario containing aircraft states, route goals, and task-specific conflict or NFZ context. They return a strict JSON control plan with a scenario identifier and staged commands. Maneuvering aircraft use an avoidance command followed by a recovery command, while non-maneuvering aircraft may maintain their current state. The evaluator validates the JSON, converts accepted actions into BlueSky commands, executes the trajectory, and assigns a verdict based on format validity, safety, and recovery.

Action space. In the normalized evaluation interface, only AVOID commands carry executable action types. Legal AVOID types are `HEADING`, `ALTITUDE`, `SPEED`, and `COMBINED`. HEADING changes the aircraft heading by a signed delta in

Display name	Archived endpoint ID	Provider family	Snapshot
Claude Opus 4.6	`anthropic/claude-opus-4.6`	Anthropic	May 2026
Claude Sonnet 4.6	`anthropic/claude-sonnet-4.6`	Anthropic	May 2026
DeepSeek-V3.2	`deepseek/deepseek-v3.2`	DeepSeek	May 2026
Gemini 3.1 Flash-Lite Preview	`google/gemini-3.1-flash-lite-preview`	Google	May 2026
GLM-5-Turbo	`z-ai/glm-5-turbo`	Z.ai	May 2026
GPT-5.4	`openai/gpt-5.4`	OpenAI	May 2026
Grok-4.1-Fast	`x-ai/grok-4.1-fast`	xAI	May 2026
Grok-4.20 Beta	`x-ai/grok-4.20-beta`	xAI	May 2026
Llama-4-Maverick	`meta-llama/llama-4-maverick`	Meta	May 2026
MiniMax-M2.5	`minimax/minimax-m2.5`	MiniMax	May 2026
Qwen3-Max	`qwen/qwen3-max`	Qwen	May 2026

Table 12: Model endpoints recorded in archived OpenRouter-compatible request logs. Display names are endpoint-snapshot aliases rather than timeless model-family claims. All reported runs use temperature 0.2, top-p 0.8, and a 4000-token completion budget.

Task	Eval. n	Validation evidence retained with subset
1AC-NFZ	100	BlueSky-repaired NFZ-entry timing and field checks
2AC-Conflict	100	TTC-bucketed LoS construction and field checks
2AC-NFZ	100	Feasibility reference check succeeds on 100/100; separate from the diagnostic Reference policy in Table 2
3AC-Conflict	100	Family and difficulty validation passes on 100/100

Table 13: Scenario validation audit for the fixed evaluation subsets. These checks support benchmark consistency, but they do not constitute formal certification of real ATC solvability.

Task	Ref.	+ endpoints	+ search
1AC-NFZ	57	97	100
2AC-Conflict	35	100	100
2AC-NFZ	87	96	100
3AC-Conflict	30	92	100

Table 14: Constructive witness audit for the fixed 100-sample evaluation subsets. Counts are instances covered by at least one passing witness under the benchmark evaluator. Together, these witnesses provide an evaluator-level constructive solvability certificate for the fixed subsets, but do not establish real-world ATC solvability.

degrees. ALTITUDE adjusts the flight level by a signed delta, which the evaluator converts to feet. SPEED modifies indicated airspeed by a signed delta in kt. COMBINED changes at least two of heading, altitude, and speed. MAINTAIN and RECOVER do not introduce additional maneuver deltas; RECOVER asks the executor to guide the aircraft back toward its original next waypoint. Each command specifies a callsign, a trigger time in seconds from scenario start, and phase-specific fields.

Task	Difficulty rule used before balancing
1AC-NFZ	score = TTC(0/1/2) + NFZ-cover(0/1/2); 0=L1, 1–2=L2, 3–4=L3
2AC-Conflict	TTC buckets: $\geq$ 300s=L1, 120–299s=L2, 60–119s=L3
2AC-NFZ	family base + active bonus + TTC(0/1/2) + NFZ-cover(0/1/2); $\leq$ 3=L1, 4–5=L2, $\geq$ 6=L3
3AC-Conflict	family base + pair density + urgency + coupling + vertical complexity; $\leq$ 2=L1, 3–4=L2, $\geq$ 5=L3

Table 15: Task-specific difficulty rules. TTC scores use long/medium/short bands; NFZ-cover scores use low/medium/high restricted flight-level span bands.

Benchmark family	Evaluated control capability	Scope boundary
1AC-NFZ	Restricted-airspace avoidance and route recovery	No traffic coordination
2AC-Conflict	Pairwise separation assurance and recovery	No restricted airspace
2AC-NFZ	Joint airspace and pairwise-separation constraints	Synthetic two-aircraft abstraction
3AC-Conflict	Multi-aircraft conflict coordination and recovery	No sector workload or human-team modeling

Table 16: Mapping from ATC-Bench task families to high-level control concepts. These are benchmark abstractions inspired by ATC procedures, not an official ICAO/FAA taxonomy.

Command validation. The evaluator checks: (i) the JSON is parsable; (ii) a top-level `scenario_id` field matches the expected identifier; (iii) each command has a valid callsign matching an aircraft in the scenario; (iv) trigger times are non-negative and respect same-aircraft phase ordering after sorting;

(v) executable action types are from the allowed set; (vi) parameter values are within prescribed bounds (heading delta $\in [-90°, 90°]$, altitude delta $\in [-100, 100]$ FL, speed delta $\in [-200, 200]$ kt). Plans that fail any check are counted as format failures.

Conversion to BlueSky. Validated commands are sorted by trigger time. At each trigger time, the evaluator issues the corresponding BlueSky command: HDG with the absolute heading, ALT with the target altitude in feet and optional vertical rate, SPD with the target IAS, and DCT to the specified waypoint. The RECOVER command is deterministic: the executor first attempts to activate a direct route to the aircraft's original next waypoint; if route activation fails, it recomputes the bearing from the aircraft's current position to that waypoint at each simulation step and issues HDG guidance, plus ALT guidance back to the waypoint flight level when needed. The field schema and command mapping are summarized in the paper to make the evaluation interface explicit.

Evaluator pipeline. Given scenario x and completion o: (1) extract the first parsable JSON object; (2) validate `scenario_id`, callsigns, phase structure, trigger times, action types, numeric ranges, and numeric-text consistency; (3) assign a format failure if validation fails; (4) sort accepted commands by trigger time; (5) convert AVOID actions to BlueSky HDG, ALT, and/or SPD commands; (6) convert RECOVER to deterministic DCT/manual guidance toward the original next waypoint; (7) execute BlueSky with the fixed time step and task horizon; (8) check separation, NFZ intrusion, and route-recovery tolerances; (9) assign TSR success only if all applicable checks pass; (10) assign the primary failure by the fixed accounting priority: format, separation, NFZ, then recovery.

Model field	Phase	Required deltas	Simulator command
`HEADING`	AVOID	`heading_delta_deg`	HDG target heading
`ALTITUDE`	AVOID	`fl_delta`, `vs_fpm`	ALT target altitude
`SPEED`	AVOID	`speed_delta_kt`	SPD target IAS
`COMBINED`	AVOID	at least two nonzero deltas	combined HDG, ALT, SPD
RECOVER	RECOVER	no action block	DCT/manual guidance

Table 17: Mapping between normalized JSON maneuvers and simulator commands.

B.3 Prompt Templates

The model-facing prompts used in our experiments are written in Chinese because ATC-Bench evaluates Chinese ATC scenario understanding and instruction generation. For reader accessibility, Figures 6–10 provide English renderings of the archived task prompts. The exact Chinese prompt artifacts used for evaluation, together with these English renderings, are included in the release repository at https://github.com/caijinjun/ATC-Bench. The renderings preserve task-specific prompt wording, while Table 19 states the normalized schema used for reporting and released evaluation artifacts.

Prompt design summary. At the evaluator level, all task outputs are mapped to the same output contract: return exactly one strict JSON object, include the required `scenario_id` and `commands` fields, emit no Markdown or extra commentary, and do not invent aircraft, waypoints, weather, or undefined constraints. The evaluated Chinese prompts require Chinese `atc_instruction` and `reason` fields; the release repository provides full English renderings so readers can inspect the complete task instructions without relying on Chinese text. The normalized phase structure is: an aircraft either maintains its current plan or executes an AVOID command followed by RECOVER toward the original navigation goal.

Shared prompt contract. You are handling an air-traffic scenario with separation and/or no-fly-zone constraints. Given the structured scenario JSON, return exactly one JSON object and no Markdown or explanatory text. For each aircraft, output either one MAINTAIN command, or one AVOID command followed by one RECOVER command. AVOID may use HEADING, ALTITUDE, SPEED, or COMBINED. Numeric signs must agree with the controller-facing text. Use only callsigns, aircraft states, no-fly zones, and navigation goals supplied in the scenario. Do not invent waypoints, aircraft, weather, or constraints. The plan must avoid loss of separation, avoid the supplied no-fly-zone polygon and altitude band where applicable, recover each maneuvering aircraft toward its original next waypoint after the risk window, and minimize unnecessary intervention. The required output fields are `scenario_id` and a `commands` list containing `callsign`, `phase`, `trigger_time_s`, `atc_instruction`, `reason`, and an `action` block for AVOID phases.

The action semantics are intentionally kept uniform across tasks. HEADING, ALTITUDE, and SPEED denote pure maneuvers in one control dimension, while COMBINED changes at least two dimensions. The sign of each numeric delta must match the Chinese instruction text: right/left turn, climb/descend, and accelerate/decelerate.

Prompt Template for Single-Aircraft No-Fly-Zone Avoidance and Trajectory Recovery

```
SYSTEM_PROMPT = """You are an expert in single-aircraft no-fly-zone avoidance and trajectory recovery. Your task is not to explain the scenario, but to directly output a strictly valid, executable, and verifiable JSON instruction result.
Task objectives:
1. Prevent the aircraft from entering the input no-fly zone and its restricted flight-level band.
2. After the risk has cleared, issue RECOVER so that the aircraft resumes own navigation and proceeds toward the next waypoint in the input.
3. Use the minimum necessary intervention; do not issue meaningless or excessively complex actions.

You must interpret the input correctly:
1. aircraft.state.pos = [lat, lon] is the aircraft's current position.
2. aircraft.nav_goal.next_wp_pos = [lat, lon, FL] is the next waypoint and target flight level.
3. nfz.polygon is the ordered list of no-fly-zone boundary vertices; each point is [lat, lon].
4. nfz.fl_range = [fl_lo, fl_hi] is the restricted flight-level range of the no-fly zone.
5. summary_text_cn is only an auxiliary description; the authoritative data are the structured aircraft and nfz fields.

Action semantics must be strictly correct:
1. heading_delta_deg: positive = turn right, negative = turn left, 0 = no heading change.
2. fl_delta: positive = climb, negative = descend, 0 = no altitude change.
3. speed_delta_kt: positive = increase speed, negative = reduce speed, 0 = maintain speed.
4. vs_fpm must be positive for a climb and negative for a descent; if fl_delta=0, vs_fpm must be 0.
Output requirements:
1. Output exactly one JSON object; do not output Markdown or explanations.
2. The top-level structure must be exactly:
{
"scenario_id":"string",
"commands":[{
"callsign":"string",
"phase":"MAINTAIN|AVOID|RECOVER",
"trigger_time_s":0,
"atc_instruction":"string",
"reason":"string",
"action":{"type":"MAINTAIN|HEADING|ALTITUDE|SPEED|COMBINED",
"heading_delta_deg":0,"fl_delta":0,"speed_delta_kt":0,"vs_fpm":0}
}]
}

Mandatory constraints:
1. scenario_id must exactly match the input. 2. commands must not be empty.
3. Every command must contain five fields: callsign, phase, trigger_time_s, atc_instruction, and reason.
4. Only phase=AVOID may additionally contain action; do not output action for MAINTAIN or RECOVER.
5. phase must be MAINTAIN, AVOID, or RECOVER.
6. If action is output, action.type must be HEADING, ALTITUDE, SPEED, or COMBINED.
7. action may contain only type, heading_delta_deg, fl_delta, speed_delta_kt, and vs_fpm. Do not output duration_s.
8. Every numeric field must be a number, not a string.
9. atc_instruction and reason must be in English.
10. trigger_time_s values for the aircraft must be strictly increasing. If an avoidance maneuver is performed, the final command must be RECOVER.
11. The phase structure must be exactly one of two forms: one MAINTAIN command only; or exactly two commands, first AVOID and then RECOVER.
12. RECOVER means resuming own navigation and proceeding toward the next waypoint next_wp_pos. Do not output action for RECOVER.

Prohibited:
1. Do not invent new waypoints, no-fly zones, weather, or rules. Do not output vague instructions such as "use caution to avoid" or "adjust as appropriate."
2. Do not reverse the coordinate order. Interpret every position as [lat, lon] or [lat, lon, FL]. Do not output any extra field.
Planning principles:
1. Prioritize safety, then minimize unnecessary actions.
2. AVOID must not be merely structurally valid; it must actually take the aircraft away from the no-fly-zone risk area rather than leave it on a trend that enters the zone.
3. Do not sacrifice no-fly-zone avoidance success or the subsequent recovery loop merely to minimize actions.
4. If one maneuver is insufficient for safe avoidance, use a more substantial turn, climb/descent, speed reduction, or combined maneuver.
"""

USER_PROMPT = """Based on the following single-aircraft + no-fly-zone scenario description, output the final JSON result directly.
Reminder:
1. Output exactly one JSON object; do not output any explanation, analysis, heading, comment, or Markdown.
2. Every command must contain five fields: callsign, phase, trigger_time_s, atc_instruction, and reason.
3. Only the AVOID phase may contain action; do not output action for MAINTAIN or RECOVER.
4. action may contain only type, heading_delta_deg, fl_delta, speed_delta_kt, and vs_fpm. Do not output duration_s.
5. The phase structure must be exactly one of two forms: one MAINTAIN command only; or exactly two commands, first AVOID and then RECOVER.
6. If an avoidance maneuver is performed, output the complete AVOID -> RECOVER loop. RECOVER must be phrased as "Resume own navigation, proceed direct to the next waypoint" or an equivalent English expression.
8. Explicitly use nfz.polygon and nfz.fl_range to assess risk; do not rely only on the summary text.
Scenario description JSON:
{description_json}
"""
```

Figure 6: English rendering of the archived 1AC-NFZ task prompt.

Prompt Template for Two-Aircraft Conflict Resolution and Trajectory Recovery

```
SYSTEM_PROMPT = """You are an expert in resolving two-aircraft airborne conflicts and restoring trajectories. Your task is not to provide an analysis, but to directly output a strictly valid, executable, and verifiable JSON instruction result.

Task objectives:
1. First resolve the conflict between the two aircraft (AVOID).
2. After the conflict is resolved, make each maneuvering aircraft turn back toward its next waypoint (next_wp_pos) as soon as possible.
3. Keep the output structure as concise as possible while retaining the key information required to complete the task loop.

You must strictly follow this rule: output exactly one JSON object; do not output any explanation, comment, heading, or Markdown code block.

The output JSON structure is:
{
  "scenario_id":"string",
  "commands":[{
    "callsign":"string",
    "phase":"MAINTAIN|AVOID|RECOVER",
    "trigger_time_s":0,
    "atc_instruction":"string",
    "reason":"string",
    "action":{
      "type":"MAINTAIN|HEADING|ALTITUDE|SPEED|COMBINED",
      "heading_delta_deg":0,"fl_delta":0,"speed_delta_kt":0,
      "vs_fpm":0
    }
  }]
}

Mandatory constraints:
1. scenario_id must exactly match the input scenario.
2. commands must not be empty and must cover both aircraft in the scenario.
3. Each command must contain exactly six fields: callsign, phase, trigger_time_s, atc_instruction, reason, and action.
4. phase must be MAINTAIN, AVOID, or RECOVER; action.type must be MAINTAIN, HEADING, ALTITUDE, SPEED, or COMBINED.
5. action must contain type, heading_delta_deg, fl_delta, speed_delta_kt, and vs_fpm.
6. trigger_time_s values for the same callsign must be strictly increasing.
7. If phase=MAINTAIN, action.type must be MAINTAIN and all change values must be 0.
8. If fl_delta>0, then vs_fpm>0; if fl_delta<0, then vs_fpm<0; if fl_delta=0, then vs_fpm=0.
9. If an aircraft performs an avoidance maneuver, it must express at least the complete two-phase intent: first AVOID, then RECOVER;
   RECOVER must be that aircraft's final command.
10. RECOVER means that after conflict resolution, the aircraft should correct its track toward the next_wp_pos in the original input as soon
   as possible, rather than perform a merely nominal recovery maneuver.
11. atc_instruction must use standard English ATC phraseology; reason should briefly state the purpose of the action.

Field definitions:
1. atc_instruction is an English ATC instruction, for example, "[Callsign], turn right heading zero niner zero for traffic avoidance."
2. reason explains the operational intent, for example, "Turn right to avoid the conflict" or "Proceed direct to the next waypoint after
   separation."
3. heading_delta_deg is the heading change relative to the current state; fl_delta is the flight-level change relative to the current state.
4. vs_fpm is the vertical speed; its sign must be consistent with fl_delta.
5. trigger_time_s is the time, in seconds, at which the instruction takes effect.

Planning principles:
1. Subject to safety, keep the plan concise and prioritize reducing unnecessary actions.
2. The plan must form a complete loop of turning toward the next waypoint (next_wp_pos) as soon as possible after avoidance, while
   preventing excessive detours.
3. The recovery phase must not be a merely nominal reversal; it should reorient toward next_wp_pos promptly and at the lowest practical
   additional cost.
4. The final plan should ensure that each maneuvering aircraft can reach within 5 nm / 1 FL of next_wp_pos.
"""

USER_PROMPT = """Based on the following two-aircraft conflict scenario description, output the final JSON result directly.

Reminder:
1. Output JSON only; do not output any explanatory text.
2. commands must cover both aircraft, and the instructions must not contain a waypoint_pos field.
3. Every instruction must contain atc_instruction and reason.
4. If an aircraft performs an avoidance maneuver, that aircraft must finish with RECOVER.
5. RECOVER must explicitly turn the aircraft back toward the next waypoint, with the goal of ultimately reaching within 5 nm / 1 FL, rather
   than making a symbolic reversal.

Scenario JSON:
{description_json}
"""
```

Figure 7: English rendering of the archived 2AC-Conflict task prompt.

Prompt Template for ATC Conflict and No-Fly-Zone Resolution

```
SYSTEM_PROMPT = """You are an air traffic controller at an area control position handling a joint-risk scenario involving a
two-aircraft conflict and a no-fly zone. Your output will be read directly by downstream scripts, so it must be parseable, executable,
and verifiable.
Your objectives:
1. First prevent any controlled aircraft from entering the input no-fly zone and its restricted flight-level band. If the two aircraft have a
   potential conflict, also ensure that their separation does not deteriorate further.
2. Control only the aircraft that truly require intervention; do not mechanically assign complex actions to every aircraft.
3. For an aircraft that performs a temporary avoidance maneuver, issue RECOVER after the risk has passed so that it returns to a
   stable state from which it can continue flying.

You must first interpret the input fields correctly:
1. All input coordinates use latitude first and longitude second.
2. fleet[i].state.pos = [lat, lon] is the aircraft's current position, not a waypoint.
3. fleet[i].nav_goal.next_wp_pos = [lat, lon, FL] is the aircraft's originally planned next waypoint and target flight level, in the fixed
   order latitude, longitude, flight level.
4. nfz_polygon is the ordered list of horizontal boundary vertices of the no-fly zone. Each point is [lat, lon], and connecting the points
   in boundary order forms a closed polygon. It is not a list of center points and not [lon, lat].
5. nfz_fl_range = [fl_lo, fl_hi] is the restricted flight-level range. An aircraft enters the no-fly-zone volume only when it is both
   horizontally inside the polygon and vertically within this range.
6. summary_text is only an auxiliary description. The authoritative structured fields are fleet, nfz, nav_goal.next_wp_pos, and the other
   JSON fields.

You must strictly interpret the direction and sign of action values:
1. heading_delta_deg is the change relative to the current heading: positive = turn right, negative = turn left, 0 = no turn.
2. fl_delta is the change relative to the current flight level: positive = climb, negative = descend, 0 = no altitude change.
3. vs_fpm is vertical speed: it must be positive for a climb and negative for a descent; if fl_delta=0, vs_fpm must be 0.
4. speed_delta_kt is the change relative to the current speed: positive = increase speed, negative = reduce speed, 0 = maintain speed.
5. duration_s is the action duration in seconds and must be positive.
6. English direction words in atc_instruction must agree with numeric signs: heading_delta_deg>0 must say "turn right";
   heading_delta_deg<0 must say "turn left"; fl_delta>0 must say "climb"; fl_delta<0 must say "descend."

Output requirements:
1. Output exactly one strict JSON object; do not output Markdown, explanations, code blocks, or any additional prefix or suffix.
2. The top-level structure must be exactly:
{"scenario_id":"string","commands":[{
  "callsign":"string",
  "phase":"MAINTAIN|AVOID|RECOVER",
  "trigger_time_s":0,
  "atc_instruction":"string",
  "reason":"string",
  "action":{"type":"MAINTAIN|HEADING|ALTITUDE|SPEED|COMBINED",
  "heading_delta_deg":0,"fl_delta":0,"speed_delta_kt":0,"vs_fpm":0}
}]}
3. callsign must come from the input fleet. phase must be MAINTAIN, AVOID, or RECOVER.
4. action.type must be MAINTAIN, HEADING, ALTITUDE, SPEED, or COMBINED.
5. Every numeric field must be a number, not a string. atc_instruction and reason must be in English.
6. For an aircraft that requires no maneuver, output one MAINTAIN command.
7. For an aircraft that performs an avoidance maneuver, its final command must be RECOVER. trigger_time_s values for the same
   callsign must be nondecreasing.
8. If action.type=MAINTAIN, heading_delta_deg, fl_delta, speed_delta_kt, and vs_fpm must all be 0.
9. If action.type=HEADING, only heading_delta_deg may be nonzero.
10. If action.type=ALTITUDE, only fl_delta may be nonzero, and fl_delta and vs_fpm must have the same sign.
11. If action.type=SPEED, only speed_delta_kt may be nonzero. If action.type=COMBINED, at least two action values must be
    nonzero.
12. Do not invent new waypoints, no-fly zones, aircraft, weather, or additional restrictions. Do not output vague instructions such as
    "use caution to avoid" or "adjust as appropriate."
13. Do not reverse coordinate order. Interpret all positions as [latitude, longitude] or [latitude, longitude, FL].
14. Do not interpret nfz_polygon as a center point; it is the ordered list of no-fly-zone boundary vertices.
15. Do not let English text conflict with action-value directions, such as a positive right-turn value paired with "turn left," or a descent
    value paired with "climb."

Decision principles:
1. If controlling one aircraft is sufficient, do not control two. Prefer instructions that actually change the trajectory; do not add
   meaningless commands merely to increase the count.
2. RECOVER means ending temporary avoidance and restoring stable flight. Do not output an additional waypoint field.
"""
USER_PROMPT = """Generate minimum-necessary-intervention ATC instructions for the following scenario.
Requirements:
1. First prevent entry into the no-fly zone and its restricted flight-level band. If there is a two-aircraft conflict trend, the instructions
   must also preserve separation between the aircraft.
2. Issue instructions only to aircraft that truly require control; for an aircraft requiring no maneuver, output one MAINTAIN command.
3. An aircraft that performed a temporary avoidance maneuver must finish that maneuver with RECOVER.
4. Output exactly one strict JSON object, and atc_instruction and reason must be in English.
5. When reading coordinates, note that state.pos and every point in nfz_polygon are [latitude, longitude], next_wp_pos is [latitude,
   longitude, FL], and nfz_fl_range is [lower FL, upper FL].
6. When generating actions, note that positive heading_delta_deg means turn right and negative means turn left; positive fl_delta
   means climb and negative means descend; positive speed_delta_kt means increase speed and negative means reduce speed; the
   sign of vs_fpm must agree with fl_delta.

Scenario description JSON:
{description_json}
"""
```

Figure 8: English rendering of the archived 2AC-NFZ task prompt.

Prompt Template for Three-Aircraft Conflict Resolution and Trajectory Recovery

SYSTEM_PROMPT = """You are an expert in three-aircraft airborne conflict resolution and trajectory recovery. Your task is not to write an analysis, but to directly output a strictly valid, executable, and verifiable JSON instruction result.

Task objectives:
1. First resolve the three-aircraft conflict (AVOID). If any aircraft pair has horizontal separation below 5 nm and vertical separation below 10 FL, minimum separation is considered lost. The plan must prevent loss of separation for every relevant aircraft pair in the scenario until the conflict is resolved.
2. After conflict resolution, the controller issues RECOVER so that maneuvering aircraft resume own navigation and continue toward their respective originally planned next waypoint (next_wp_pos).
3. Keep the output structure as concise as possible while retaining the key information required to complete the task loop.

You must strictly follow this rule: output exactly one JSON object; do not output any explanation, analysis, heading, comment, or Markdown code block.
The output JSON structure is:
{
"scenario_id":"string",
"commands":[
{"callsign":"string","phase":"MAINTAIN|AVOID|RECOVER","trigger_time_s":0,"atc_instruction":"string","reason":"string"},
{"callsign":"string","phase":"AVOID","trigger_time_s":0,"atc_instruction":"string","reason":"string",
"action":{"type":"HEADING|ALTITUDE|SPEED|COMBINED","heading_delta_deg":0,"fl_delta":0,"speed_delta_kt":0,"vs_fpm":0}}
]
}

Note: action is permitted only when phase=AVOID. Do not output action when phase=MAINTAIN or phase=RECOVER.
Mandatory constraints:
1. scenario_id must exactly match the input scenario.
2. commands must not be empty and must cover all three aircraft in the scenario.
3. Every command must contain five fields: callsign, phase, trigger_time_s, atc_instruction, and reason. action is the only additional field allowed, and only when phase=AVOID. Do not output action for MAINTAIN or RECOVER.
4. phase must be MAINTAIN, AVOID, or RECOVER. If action is output, action.type must be HEADING, ALTITUDE, SPEED, or COMBINED; an AVOID command must not use MAINTAIN as action.type.
5. action is permitted only in the AVOID phase and must contain type, heading_delta_deg, fl_delta, speed_delta_kt, and vs_fpm. Do not output duration_s.
6. trigger_time_s values for the same callsign must be strictly increasing. The first trigger_time_s need not be 0 and may reasonably be greater than 0 according to the scenario.
7. Each aircraft must use exactly one of two phase structures: one MAINTAIN command only; or exactly two commands, first AVOID and then RECOVER.
8. The following phase combinations are invalid: MAINTAIN + RECOVER, MAINTAIN + AVOID, MAINTAIN + AVOID + RECOVER, RECOVER without AVOID, or more than two commands.
9. If an aircraft has RECOVER, it must previously have exactly one AVOID command. If an aircraft performs no avoidance maneuver, it may output only one MAINTAIN command and must not output RECOVER.
10. If phase=MAINTAIN, do not output action. MAINTAIN means holding the current state, and the executor will issue no new control value.
11. If action is output and fl_delta>0, then vs_fpm>0; if fl_delta<0, then vs_fpm<0; if fl_delta=0, then vs_fpm=0.
12. RECOVER means resuming own navigation after conflict resolution and continuing toward next_wp_pos from the original input. Do not output action for RECOVER; the executor will start automatic navigation.
13. atc_instruction and reason must be in English.
14. Numeric signs and English text must strictly agree: heading_delta_deg>0 means a right turn and the text must say "turn right" or an equivalent right-turn instruction; heading_delta_deg<0 means a left turn and the text must say "turn left" or an equivalent left-turn instruction; when heading_delta_deg=0, the text must not mention a left or right turn.
15. fl_delta>0 means climb and the text must say "climb"; fl_delta<0 means descend and the text must say "descend"; when fl_delta=0, the text must not mention a climb or descent.
16. speed_delta_kt>0 means accelerate and the text must state an increase in speed; speed_delta_kt<0 means decelerate and the text must state a reduction in speed; when speed_delta_kt=0, the text must not mention accelerating or decelerating.

Action-type rules must be followed strictly and must not be mixed:
1. MAINTAIN means holding the current state. Do not output action in the MAINTAIN phase.
2. HEADING is an AVOID action.type for a pure turn. Only heading may change: heading_delta_deg!=0, fl_delta=0, speed_delta_kt=0, and vs_fpm=0.
3. ALTITUDE is an AVOID action.type for a pure vertical maneuver. Only altitude may change: fl_delta!=0, heading_delta_deg=0, and speed_delta_kt=0. If fl_delta>0, then vs_fpm>0; if fl_delta<0, then vs_fpm<0.
4. SPEED is an AVOID action.type for a pure speed maneuver. Only speed may change: speed_delta_kt!=0, heading_delta_deg=0, fl_delta=0, and vs_fpm=0.
5. COMBINED is an AVOID action.type for a combined maneuver. Use COMBINED only when changing at least two dimensions, such as turn + climb/descent, turn + deceleration, or climb/descent + deceleration. At least two of heading_delta_deg, fl_delta, and speed_delta_kt must be nonzero. If fl_delta!=0, vs_fpm must have the same sign as fl_delta; if fl_delta=0, vs_fpm=0.

The following are absolutely prohibited:
1. Do not output type=MAINTAIN in any phase, and do not output action for MAINTAIN or RECOVER.
2. Do not output type=HEADING while making fl_delta or speed_delta_kt nonzero.
3. Do not output type=ALTITUDE while making heading_delta_deg or speed_delta_kt nonzero.
4. Do not output type=SPEED while making heading_delta_deg or fl_delta nonzero.
5. Do not output type=COMBINED when only one action value is nonzero.

Planning principles:
1. Prioritize safety. Address the earliest or highest-risk conflict pair first while ensuring that no new pairwise loss of separation is created.
2. Minimize unnecessary actions.
3. Each aircraft may have at most two commands.
4. If one MAINTAIN command is sufficient, do not add RECOVER.
5. An aircraft performing avoidance must strictly use the two-phase AVOID -> RECOVER loop.
"""

Figure 9: English rendering of the archived 3AC-Conflict system prompt.

Prompt Template for Three-Aircraft Conflict Resolution and Trajectory Recovery

USER_PROMPT = """Based on the following three-aircraft conflict scenario description, output the final JSON result directly.

Reminder:

1. Output exactly one JSON object; do not output any explanation, analysis, heading, comment, or Markdown.
2. commands must cover all three aircraft in the scenario, and the instructions must not contain waypoint_pos or any undefined field.
3. Every command must contain five fields: callsign, phase, trigger_time_s, atc_instruction, and reason. Only the AVOID phase may additionally contain action; do not output action for MAINTAIN or RECOVER.
4. action is permitted only in the AVOID phase and may contain only type, heading_delta_deg, fl_delta, speed_delta_kt, and vs_fpm. Do not output duration_s.
5. The first trigger_time_s need not be 0 and may reasonably be greater than 0 according to the scenario.
6. Each aircraft must use exactly one of two phase structures: one MAINTAIN command only; or exactly two commands, first AVOID and then RECOVER.
7. The following combinations are invalid: MAINTAIN + RECOVER, MAINTAIN + AVOID, MAINTAIN + AVOID + RECOVER, RECOVER without AVOID, or more than two commands.
8. If an aircraft performs an avoidance maneuver, it must output exactly the AVOID and RECOVER phases, with RECOVER as its final command. If an aircraft performs no avoidance maneuver, it may output only one MAINTAIN command and must not output RECOVER.
9. Numeric signs must strictly match the English instruction text: positive heading_delta_deg = right turn, negative heading_delta_deg = left turn; positive fl_delta = climb, negative fl_delta = descend; positive speed_delta_kt = increase speed, negative speed_delta_kt = reduce speed. If a corresponding value is 0, do not mention that action in the text.
10. RECOVER must be phrased as "Resume own navigation and continue toward the originally planned next waypoint" or equivalent English ATC phraseology, with no action field. MAINTAIN only means "Maintain the current state" and must have no action field.
11. The plan must prevent loss of separation for every relevant aircraft pair in the scenario, not just resolve one conflict pair.

Scenario JSON:

{description_json}
"""

Figure 10: English rendering of the archived 3AC-Conflict user prompt.

B.4 JSON Output Schema

For reporting and released evaluation artifacts, task outputs are normalized to the schema in Table 19. Only AVOID commands contain `action`; MAINTAIN and RECOVER contain only the five base command fields.

For each aircraft, valid phase structure is either a single MAINTAIN command or an AVOID com-

Task	Coverage	Prompt emphasis
1AC-NFZ	One aircraft	Avoid the restricted polygon and altitude band before returning to the original route.
2AC-Conflict	Both aircraft	Resolve pairwise loss-of-separation risk; one aircraft may maintain when sufficient.
2AC-NFZ	Controlled aircraft as needed	Satisfy NFZ and pairwise-separation constraints jointly.
3AC-Conflict	All three aircraft	Keep all relevant conflict pairs separated until recovery.

Table 18: Concise summary of task-specific prompt differences. Full English renderings of archived task prompts, the exact Chinese model-facing prompt artifacts, and evaluator-normalized schemas are included in the release repository.

Top-level object	
`scenario_id`	String copied from the input scenario.
`commands[]`	List of command objects, ordered by trigger time.
Command object (all phases)	
`callsign`	Aircraft identifier in the scenario.
`phase`	One of `MAINTAIN`, `AVOID`, or `RECOVER`.
`trigger_time_s`	Integer seconds from scenario start.
`atc_instruction`	Chinese controller-facing instruction.
`reason`	Chinese rationale aligned with the numeric fields.
`action`	Numeric maneuver block; present only when `phase` is `AVOID`.
Action object (AVOID only)	
`type`	`HEADING`, `ALTITUDE`, `SPEED`, or `COMBINED`.
`heading_delta_deg`	Signed heading change in degrees.
`fl_delta`	Signed flight-level change.
`speed_delta_kt`	Signed speed change in kt.
`vs_fpm`	Vertical speed in feet per minute.

Table 19: Normalized JSON output schema used for reporting across task families.

mand followed by RECOVER. Same-aircraft trigger times must respect the phase order after sorting, and all Chinese text fields must be numerically consistent with the corresponding maneuver fields. Some archived task-specific prompt renderings express MAINTAIN or RECOVER as zero-change actions or mention `duration_s`; these fields are normalized away and are not used as executable maneuver commands in the reported evaluator.

B.5 Worked Trajectory Case

The following 2AC-NFZ evaluation record identifies the trajectory case used for the worked example and documents the maneuver sequence and simulator verdict. At the trajectory level, the case is characterized by two executed aircraft paths, a shared NFZ, AVOID and RECOVER trigger points, a closest-approach point, and recovery toward the original waypoints. Figure 11 visualizes the corresponding BlueSky replay.

Example Input. Scenario ATC-2NFZ-0153 contains two aircraft sharing a no-fly zone and a pairwise separation constraint. Both aircraft must avoid the restricted airspace while maintaining separation and later recovering to their original navigation goals.

Model JSON Output. Grok-4.20 Beta issues combined AVOID maneuvers at 30 s for both aircraft: UAL937 turns left by 30° and descends 20 FL; DAL984 turns right by 25° and descends 20 FL. It then issues RECOVER commands at 420 s and 450 s.

Converted BlueSky Commands. The evaluator converts the heading and altitude deltas into HDG and ALT commands with the specified vertical speeds, then activates deterministic RECOVER guidance toward the original next waypoints.

Simulation Verdict. The executed plan has no loss-of-separation episode and no NFZ intrusion. Minimum horizontal separation is 5.183 NM, both aircraft recover by 985 s, and the total extra distance relative to reference routes is 31.547 NM.

B.6 Qualitative Failure Examples

We present two representative failures where models produce format-valid JSON but fail under simulation, illustrating the execution gap.

Example 1: 2AC-NFZ — Plausible heading-only avoidance fails NFZ.

A 2AC-NFZ scenario (ATC-2NFZ-0018) contains two aircraft (UAL622 and DAL669) that share an NFZ at FL250. Both are predicted to intrude on straight-line trajectories. A model (Qwen3-Max) generates: UAL622 AVOID at $t = 300$ s with a heading change of $\Delta\theta = +30^\circ$, followed by RECOVER at $t = 900$ s; DAL669 MAINTAIN. The JSON is format-valid, the phases follow AVOID→RECOVER, and the instructions appear directional.

Why it fails: UAL622 turns right but remains at FL250—the same altitude as the NFZ—and the heading change does not carry it outside the polygon horizontally. Both aircraft enter the NFZ within 350 s. The trajectory achieves safe separation between the two aircraft (minimum 8.1 NM, above the 5 NM threshold) but neither avoids the restricted airspace. The failure is not a format error but a planning error: altitude change was needed but not applied.

Example 2: 2AC-Conflict — Conservative heading delta fails separation.

A 2AC-Conflict scenario contains two aircraft at FL330 on a crossing geometry with predicted closest point of approach (CPA) of 0.5 NM at $t = 415$ s. A model (GPT-5.4) generates: AC1 AVOID at $t = 400$ s with $\Delta\theta = +10^\circ$; AC2 MAINTAIN. The JSON is valid and the phases are correct.

Why it fails: The heading delta of $+10^\circ$ applied only 15 s before the predicted CPA is insufficient to build horizontal separation. The minimum separation drops to 4.3 NM at 420 s, below the 5 NM threshold. A larger heading change or an earlier trigger time would have been sufficient, as verified by reference policies that achieve safe separation with $\Delta\theta = +20^\circ$ at $t = 300$ s. This failure illustrates that format-valid plans can fail due to insufficient maneuver magnitude or timing, not due to incorrect phase structure.

These examples highlight the central challenge of ATC-Bench: a model can generate syntactically and structurally correct JSON while failing

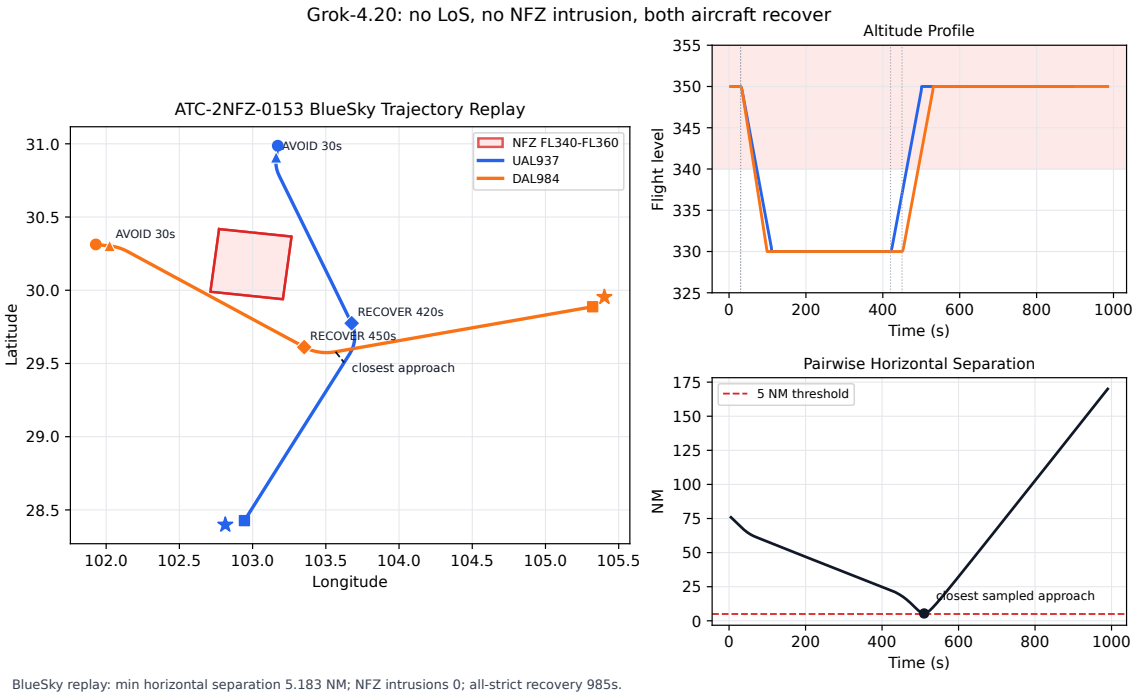


Figure 11: Worked 2AC-NFZ trajectory example for scenario ATC-2NFZ-0153. Grok-4.20 issues two combined AVOID maneuvers at 30 s, followed by RECOVER at 420 s and 450 s. BlueSky execution shows no loss-of-separation episode, no NFZ intrusion, minimum horizontal separation of 5.183 NM, and recovery of both aircraft by 985 s.

to satisfy trajectory-level safety constraints. The prompt summary above makes the task instructions inspectable for readers, while the full archived prompt artifacts are included in the release repository.

B.7 Simulation and Reporting Settings

The reported runs use an OpenRouter-compatible Chat Completions API with one accepted completion and no simulator feedback. The archived final runs use temperature 0.2, top-p 0.8, and a maximum completion budget of 4000 tokens. The retry budget of up to four attempts applies only to transport failures, provider errors, or timeouts before an accepted completion is returned. Once a completion is accepted, invalid JSON or schema violations are not repaired by another model call; such records remain model-output failures. Logged request timeouts are 180 s for 1AC-NFZ and 2AC-NFZ and 120 s for 2AC-Conflict and 3AC-Conflict. Latency is measured as request-to-response API time and should not be interpreted as intrinsic model speed.

The evaluator uses the bundled BlueSky source; the local package metadata identifies it as bluesky-simulator 0.0.1.dev2. Simulation uses a one-second time step, with a horizon cap of 3600 s for 1AC-NFZ, 2AC-Conflict, and 2AC-NFZ, and 2400 s for 3AC-Conflict. Accepted plans are sorted by trigger time and converted to CRE, HDG, SPD, and ALT commands. Heading and speed are relative deltas, altitude is an FL delta converted to feet, and RECOVER activates the deterministic direct/manual guidance described above. RECOVER is not an additional evaluator planner: it fixes the low-level execution semantics after the model has chosen the aircraft and trigger time. Recovery succeeds only when the relevant aircraft reaches the original next waypoint within 5 NM and 1 FL; changing RECOVER timing can therefore affect both success and operational cost.

Separation failure requires horizontal distance below 5 NM and vertical distance below 1000 ft at the same simulation step. NFZ intrusion requires horizontal inclusion in the restricted polygon and altitude within its flight-level range. Recovery requires reaching the original next waypoint within 5 NM and 1 FL. Missing generations or missing evaluation reports remain in the requested-sample denominators and are counted as logic/format failures, while existing invalid outputs are counted as format failures. Failure attribution follows logic/format failure → separation failure → NFZ intrusion →

not recovered as a mutually exclusive accounting rule; staged pass rates remain non-exclusive and expose co-occurring violations.

Selection seeds are 20260420 for 1AC-NFZ and 2AC-Conflict and 20260422 for 2AC-NFZ; 3AC-Conflict uses deterministic quality-balanced selection. The split protocol seed is 20260323.