

Self-Corrected Speech Reasoning: Diagnosing and Improving Condition Updating via PACER

Zijie Zeng¹, Liwen Shi¹, Yunji¹, Bowen Wang¹, Nanjie Li², Xiaoyong Guo², Zeyu Chen¹,
Xinzheng Sun¹, Xuyang Xiong¹, Haoyang Wu¹, Sheng Zhang^{1,†}

¹Tsinghua Shenzhen International Graduate School

²The School of Computer Science and Technology, Xinjiang University

Correspondence: zeng-zj24@mails.tsinghua.edu.cn, zhang_sh@mail.tsinghua.edu.cn

Abstract

Speech large language models (Speech LLMs) are increasingly expected to reason in everyday spoken interactions, where users often revise task conditions through self-correction. Unlike ordinary disfluencies, self-corrections alter the premise structure of a problem: obsolete conditions must be discarded, while final corrected conditions must guide reasoning. We formulate this setting as self-corrected speech reasoning and study it as a condition-update problem. We construct SCUR-Bench (Spoken Condition-Update Reasoning Benchmark), a controlled Chinese benchmark of 13,295 instances organized by reasoning difficulty and self-correction difficulty, and introduce a three-path diagnostic protocol to analyze speech-side recovery, condition use, and reasoning. Our analysis reveals a recurrent failure pattern in current Speech LLMs: corrected conditions may be perceived, transcribed, or even mentioned, yet still fail to serve as active premises in reasoning. We term this final-condition adoption failure. A controlled English parallel evaluation exhibits the same recovery–adoption gap, providing preliminary evidence that this failure is not unique to Chinese. To mitigate it, we propose **PACER**ⁱ, a perception-aware condition-enhanced reinforcement framework based on GRPO. On Qwen2.5-Omni-7B, PACER reduces end-to-end error and condition-related errors by 39.8% and 67.5%, respectively, relative to the untrained baseline.

1 Introduction

Speech large language models (Speech LLMs) have recently progressed from speech recognition and audio understanding to end-to-end speech interaction and speech reasoning, moving closer to real-world voice-based interaction with users (Zhang et al., 2023; Rubenstein et al., 2023; Tang et al.,

2024; Chu et al., 2024, 2023; Xie and Wu, 2024; Zeng et al., 2024; Fang et al., 2025; Xu et al., 2025). However, natural speech often contains self-corrections, where a speaker states a condition and later revises it. Unlike filled pauses or repetitions, self-corrections change the operative task constraint: models must suppress obsolete conditions and reason with the final valid ones. We therefore formulate self-corrected speech reasoning as a condition-update problem.

Existing studies have evaluated Speech LLMs in terms of audio understanding, spoken reasoning, and disfluency robustness, showing that corrections, restarts, and repetitions can substantially degrade model performance (Gupta et al., 2021; Liu et al., 2025; Gosai et al., 2025; Lin et al., 2026; Modi et al., 2026). However, most evaluations focus on aggregate degradation under broadly disfluent speech. They rarely trace how self-correction changes the model’s condition state, nor do they clearly distinguish failures caused by speech perception, logical reasoning, or failure to adopt the corrected final condition. As a result, controlled data and evaluation protocols for fine-grained diagnosis in self-corrected speech reasoning remain limited.

To address this gap, we construct a controlled diagnostic benchmark for Chinese self-corrected speech reasoning. Inspired by text-based reasoning QA and complex reasoning benchmarks that control difficulty through reasoning steps, variable counts, and logical branches (Weston et al., 2015; Cobbe et al., 2021; Wu et al., 2025), our benchmark organizes examples along two dimensions: self-correction complexity and reasoning difficulty. Each example includes structured condition annotations. We further design a three-path diagnostic protocol that analyzes model behavior across speech perception, logical reasoning, and final-condition adoption, enabling fine-grained behavioral diagnosis.

[†]Corresponding author.

ⁱ<https://github.com/ordealm/PACER>

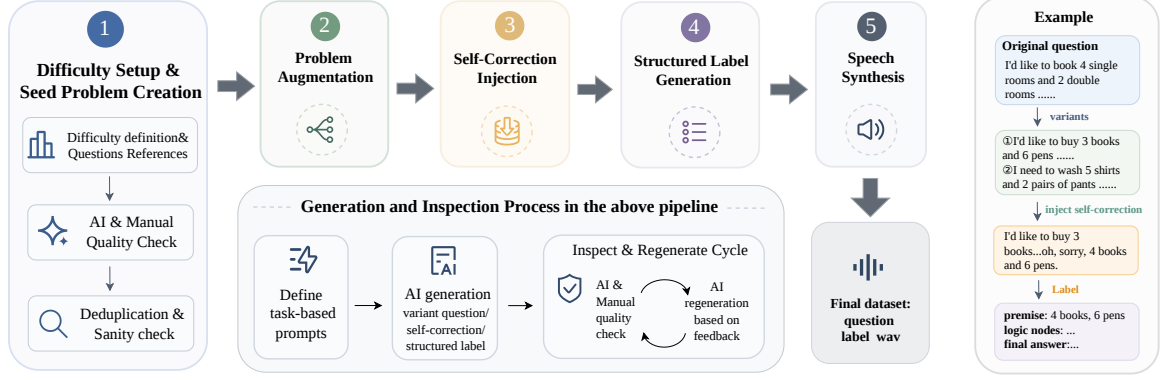


Figure 1: Dataset construction pipeline for SCUR-Bench. Seed problems are augmented into surface-diverse variants while preserving intrinsic reasoning difficulty and condition-update structure; the examples therefore illustrate structure-preserving augmentation rather than cross-domain generalization. Each stage combines AI-assisted generation with manual quality control, followed by iterative inspection and regeneration when needed.

Our analysis shows that errors in Chinese self-corrected speech reasoning are not fully explained by speech perception or text-based reasoning alone. Across the evaluated models, we observe a recurring condition-state update error pattern, which we call final-condition adoption failure. In such cases, corrected information may be recoverable from speech, transcribed, or even mentioned in the response, but still fail to serve as an active premise in the reasoning chain. Under our evaluation setting, condition-related errors constitute an important source of end-to-end speech reasoning failures, suggesting that the evaluated Speech LLMs do not yet consistently demonstrate reliable condition-state updating over self-corrected spoken input.

To mitigate this issue, we propose PACER, Perception-Aware Condition-Enhanced Reinforcement for Speech Reasoning, a GRPO-based optimization framework (Shao et al., 2024). Inspired by perception-aware optimization (Wang et al., 2026b), PACER does not aim to improve general ASR accuracy. Instead, it combines structured condition-aware rewards with speech-evidence contrast, encouraging sensitivity to speech-side information that may affect condition-grounded reasoning while promoting the use of final valid conditions. Experiments show that PACER reduces condition-related errors and improves end-to-end robustness under self-corrected spoken input compared with final-answer-only reinforcement.

The main contributions of this work are as follows:

- **Diagnostic task and benchmark.** We for-

mulate self-corrected speech reasoning as a condition-update problem and construct SCUR-Bench, a controlled benchmark with structured condition annotations and a hierarchical three-path diagnostic protocol.

- **Failure diagnosis.** We identify final-condition adoption failure as a recurring pattern among the evaluated Speech LLMs: corrected information may be recoverable, transcribed, or mentioned, yet still fail to serve as an active premise in subsequent reasoning. A controlled English parallel evaluation reveals a similar recovery–adoption gap, providing preliminary evidence beyond the Chinese setting.
- **Optimization method.** We propose PACER, a perception-aware condition-enhanced reinforcement framework that combines structured condition-aware rewards with speech-evidence contrast to reduce condition-related errors in self-corrected speech reasoning.

2 Task Setup and Diagnostic Framework

2.1 Task Definition

We study self-corrected speech reasoning, where a model receives a spoken question with natural self-corrections and must reason over the conditions ultimately endorsed by the speaker. Unlike standard spoken question answering, earlier conditions may be invalidated by later utterances, while the corrected conditions determine the gold answer.

Formally, each instance is represented as

$$(x, q, \mathcal{V}, \mathcal{O}, A), \quad (1)$$

where x denotes the spoken question, q denotes

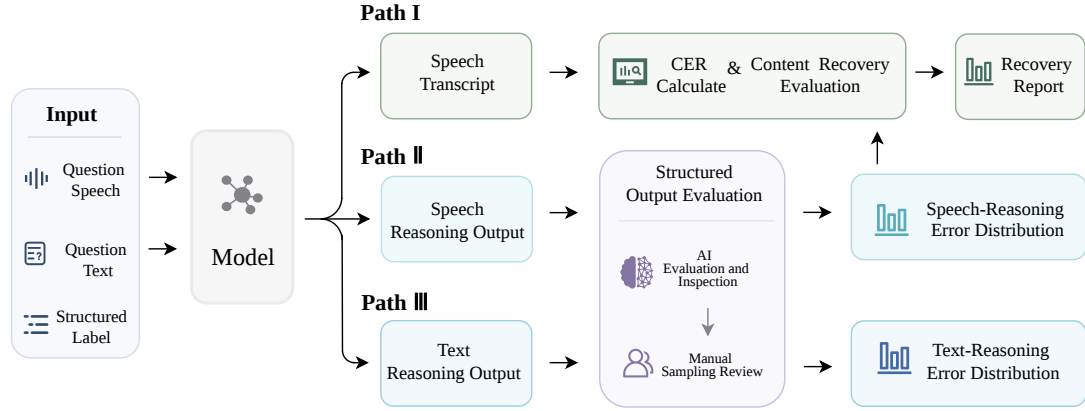


Figure 2: Three-path diagnostic protocol for analyzing speech-side recovery, end-to-end condition use, and text-side reasoning in self-corrected speech reasoning.

its text counterpart preserving the self-correction expressions, $\mathcal{V}$ is the set of final valid conditions, $\mathcal{O}$ is the set of obsolete conditions invalidated by corrections, and $\mathcal{A}$ is the set of acceptable answers. Given x , the model generates a response $\hat{y}$, which is correct if the extracted answer matches any element in $\mathcal{A}$.

For example, in “the price is 400 yuan, no, 380 yuan; apply a 10% discount, actually 15%”, 400 yuan and 10% are obsolete, while 380 yuan and 15% are final valid conditions. The answer should therefore be computed as $380 \times 0.85 = 323$. The diagnostic protocol below analyzes whether errors arise from speech perception, reasoning, or condition use.

2.2 Dataset Construction and Annotation

Existing resources are insufficient for studying self-corrected speech reasoning at training scale. Prior reasoning benchmarks control task complexity through reasoning steps, condition counts, or discrete operations (Weston et al., 2015; Dua et al., 2019; Cobbe et al., 2021; Patel et al., 2021), but are primarily text-based and do not model spoken self-corrections that revise task conditions. Conversely, self-repair and disfluency studies examine repair organization, disfluency detection, and robustness to non-canonical speech (Schegloff et al., 1977; Levelt, 1983; Honnibal and Johnson, 2014; Gupta et al., 2021), but typically lack reasoning-oriented condition annotations and supervision at the scale required for Speech LLM training. We therefore construct a Chinese self-corrected speech reasoning dataset for controlled diagnosis and optimization.

We organize the dataset along two dimensions:

intrinsic reasoning difficulty and self-correction difficulty. The former captures the complexity of solving the clean problem, while the latter captures the structural complexity of condition revision in speech. Intrinsic reasoning difficulty has three levels: L1 for single-step calculation or direct inference, L2 for two- to four-step single-chain reasoning or simple multi-variable judgment, and L3 for longer reasoning chains, multi-variable composition, or multi-branch reasoning. Self-correction difficulty has six levels: D0 for no correction, D1/D2 for single-/multi-slot immediate correction, D3/D4 for single-/multi-slot chained correction, and D5 for long-distance retrospective correction. We exclude L1D5 because retrospective correction requires sufficient intervening context and is often unnatural in single-step problems. D4 and D5 are retained as controlled single-turn stress tests of complex condition revision rather than as naturalistic models of multi-turn correction; analogous dependencies may arise more naturally across dialogue turns, which we leave for future work.

The construction pipeline, shown in Figure 1, consists of seed-problem preparation, source-level splitting, difficulty-preserving augmentation, self-correction insertion, structured annotation, quality control, and speech synthesis. Seed problems are split at the source level before augmentation to separate the Train+Dev pool from SCUR-Test, thereby reducing source leakage between model development and final evaluation. SCUR-Dev is subsequently held out from unused instances in the Train+Dev pool and is never used for parameter updates in the reported PACER experiments. LLMs are then used to generate difficulty-preserving vari-

Model	Final Answer		Condition-Related Errors			Audio Condition Use		Transcription
	Text Err.	Audio Err.	Text C/E	Audio C/E	Audio C/All	Adopt.	OldCond	VCR
Fun-Audio-Chat-8B	8.63	18.07	23.72	60.89	11.00	85.92	7.33	90.19
Qwen2.5-Omni-7B	21.77	27.91	32.10	51.80	14.46	83.46	9.74	93.63
MiniCPM-o-2.6	31.53	50.48	41.02	67.22	33.94	71.56	29.84	74.24
R1-AQA	69.72	85.94	62.96	86.73	74.54	35.06	68.07	83.09
Qwen2-Audio-7B-Instruct	71.97	86.87	65.90	88.21	76.63	33.44	69.27	79.17

Table 1: Core behavioral diagnostics for self-corrected speech reasoning. All values are percentages. C/E = CondErr/Err, C/All = CondErr/All, Adopt. = corrected-condition adoption, OldCond = obsolete-condition usage, and VCR = ValidCondRecall. Lower is better except Adopt. and VCR.

ants and insert self-corrections under predefined D-level constraints. At each generation and annotation stage, AI-assisted checks and human verification are jointly applied to identify inconsistent corrections, ambiguous answers, invalid reasoning paths, and mismatched difficulty labels. Each retained instance is annotated with final valid conditions, obsolete conditions, reference reasoning nodes, and acceptable final answers.

We name the resulting benchmark SCUR-Bench (Spoken Condition-Update Reasoning Benchmark), comprising 13,295 instances: 10,400 in SCUR-Train, 405 in SCUR-Dev, and 2,490 in SCUR-Test. The current PACER experiments use a balanced subset of SCUR-Train for optimization. Speech inputs are synthesized from the annotated Chinese questions using Edge-TTS. Further construction rules, validation prompts, synthesis details, matrix statistics, and the data release plan are provided in Appendix B.

2.3 Diagnostic Protocol and Core Metrics

Based on the annotated benchmark, we design a three-path black-box diagnostic protocol to evaluate final-answer correctness and condition use, including whether models adopt final valid conditions, suppress obsolete ones, and use corrected conditions as effective reasoning premises.

As shown in Figure 2, the text reasoning path uses the textual counterpart q as a speech-free reference, the speech reasoning path uses the spoken question x as the end-to-end setting, and the transcription path asks the model to transcribe x to estimate whether correction-relevant evidence is recoverable from speech. These paths support behavioral comparison rather than mechanistic decomposition.

For reasoning outputs, we use LLM-assisted reference evaluation with human verification to judge final-answer correctness, corrected-condition adop-

tion, obsolete-condition usage, mention-only condition use, and error source. An error is condition-related if its source label is condition_only or both. We define

$$\text{CondErr/Err} = \frac{\sum_i \mathbb{I}[c_i=0 \wedge e_i \in \mathcal{E}_c]}{\max(1, \sum_i \mathbb{I}[c_i=0])} \times 100\%, \quad (2)$$

where $\mathcal{E}_c = \{\text{condition_only}, \text{both}\}$, c_i indicates final-answer correctness, and e_i is the evaluator-assigned error source. CondErr/Err measures the share of erroneous answers involving condition misuse, while CondErr/All reports the same error type over all samples.

We also report Adoption, OldCond, Mention-only, CER, and ValidCondRecall. Adoption and OldCond measure use of final valid conditions and obsolete conditions, respectively. Mention-only measures cases where required final valid conditions are mentioned but not used as reasoning premises; these conditions are corrected in D1–D5 and unrevised in D0. CER and ValidCondRecall are computed on transcriptions; the latter measures whether final valid conditions are recoverable in speech-reasoning error cases.

We audit 100 randomly sampled examples for evaluator reliability, obtaining 98.00% agreement on final-answer correctness ($\kappa = 0.932$) and 93.00% on error-source classification ($\kappa = 0.776$). Detailed metric definitions, judge settings, and reliability analyses are provided in Appendix C.

3 Diagnostic Analysis

Using the three-path protocol in Section 2.3, we first compare diagnostic patterns across models and then examine how errors vary with correction and reasoning difficulty. Table 1 reports the core cross-model metrics.

Cross-model diagnostic patterns. Across models, speech input consistently increases final-answer errors. For example, Fun-Audio-Chat-8B

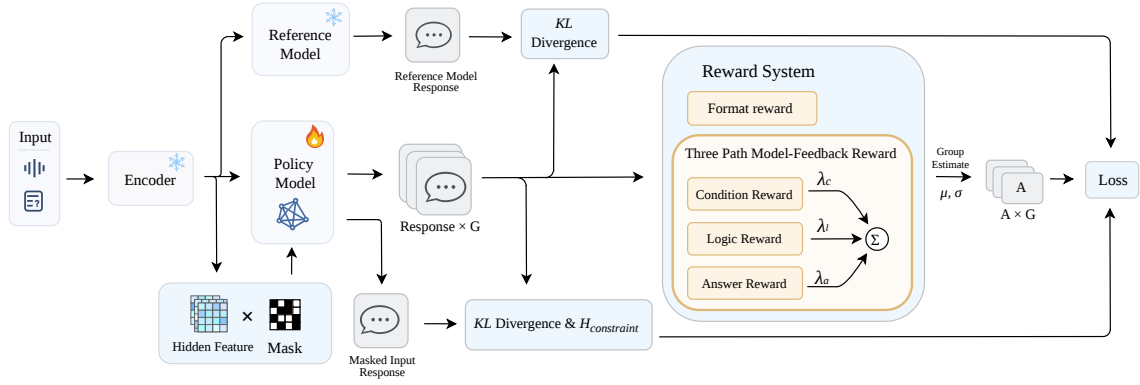


Figure 3: PACER training framework with GRPO, structured rewards, and speech-evidence contrast.

Metric	Correction difficulty						Reasoning difficulty		
	D0	D1	D2	D3	D4	D5	L1	L2	L3
Mention-only ↓	7.52	14.12	19.97	16.57	23.52	21.11	8.11	15.61	25.51
C/All ↓	3.84	12.44	16.55	16.37	25.56	18.32	8.20	14.34	20.30
C/E ↓	25.93	51.85	52.63	58.73	67.91	42.48	86.67	60.71	41.51
Err. ↓	14.81	24.00	31.44	27.88	37.64	43.13	9.46	23.61	48.90

Table 2: No-training Qwen2.5-Omni-7B audio reasoning results by correction and reasoning difficulty. All values are percentages.

Language	Audio Err.↓	CondErr/All↓	Adopt.↑	OldCond↓	VCR↑
Chinese	27.91%	14.46%	83.46%	9.74%	93.63%
English	32.21%	13.49%	83.85%	8.47%	92.20%

Table 3: Controlled Chinese–English comparison. The English parallel set exhibits the same qualitative recovery–adoption gap observed in Chinese.

risers from 8.63% text error to 18.07% audio error, while Qwen2.5-Omni-7B rises from 21.77% to 27.91%. The error composition also shifts: condition-related errors among erroneous predictions increase from 23.72% to 60.89% for Fun-Audio-Chat-8B, and from 41.02% to 67.22% for MiniCPM-o-2.6. Thus, self-corrected speech exposes failures closely tied to condition use rather than uniformly degrading performance.

The adoption and obsolete-condition metrics reveal condition-state maintenance issues beyond final-answer accuracy. Fun-Audio-Chat-8B and Qwen2.5-Omni-7B retain corrected-condition adoption rates above 83%, yet still show obsolete-condition leakage. In contrast, R1-AQA and Qwen2-Audio-7B-Instruct show adoption rates below 36% and obsolete-condition usage above 68%. The three-way protocol further separates failure sources: R1-AQA and Qwen2-Audio-7B-Instruct already fail substantially on text input, MiniCPM-o-2.6 shows a larger text–audio gap and relatively low

valid-condition recall, whereas Fun-Audio-Chat-8B and Qwen2.5-Omni-7B have high VCR, reaching 90.19% and 93.63%, but still produce substantial audio condition-related errors. This contrast suggests that recovering corrected conditions from speech is necessary but not sufficient; models must also integrate those conditions into the reasoning process.

English parallel evaluation. To examine whether the recovery–adoption gap is specific to Chinese speech, we further construct a controlled English parallel evaluation set that preserves the reasoning structures and correction patterns of the Chinese test instances. As shown in Table 3, a similar pattern also emerges in English: the model achieves a VCR of 92.20%, whereas corrected-condition adoption reaches 83.85%, with obsolete-condition use remaining at 8.47%. This result further illustrates that accurately recovering a spoken correction and consistently carrying that correction into downstream reasoning are related but distinct challenges. The English evaluation provides an initial indication that this gap is not unique to Chinese, although broader cross-lingual validation will require independently collected speech data and evaluation across additional languages and models.

Difficulty-wise behavior. Table 2 further examines how the two difficulty axes shape condition-use behavior, using NoTrain Qwen2.5-Omni-7B audio results. Along the correction-difficulty axis, audio error rises from 14.81% on D0 to 43.13% on D5, while C/All increases from 3.84% to 18.32%. C/E also stays high in harder correction groups, reaching 67.91% on D4, indicating that a large share of audio errors involve condition misuse.

Setting	Mention-Only ↓	Adopt. ↑	OldCond ↓	CondErr/All ↓	CondErr/Err ↓	CUF/CondErr ↓	Err. ↓
Gold-text NoTrain	10.61	88.36	3.43	6.99	32.10	53.45	21.77
NoTrain	16.90	83.46	9.74	14.46	51.80	75.00	27.91
Whisper ASR + Qwen2.5-Omni	19.26	81.89	13.93	18.63	57.07	75.22	32.65
SFT-only	12.06	92.90	0.51	7.31	21.90	15.93	33.37
GRPO-Answer	9.55	88.87	1.23	5.54	28.99	47.83	19.12
GRPO-Structured	8.45	89.73	0.68	5.22	28.57	43.85	18.27
PACER	7.96	90.64	0.58	4.70	27.99	41.03	16.79
w/o Logic Reward	8.50	90.01	0.85	5.18	30.07	44.96	17.23
w/o Condition Reward	7.93	89.50	0.95	5.46	31.92	44.85	17.11
w/o Answer Reward	9.15	89.53	0.53	5.70	33.02	35.92	17.27

Table 4: Main results, additional baselines, and reward-component ablations on end-to-end self-corrected speech reasoning. All values are percentages. Mention-Only denotes stated but unused final valid conditions. CUF/CondErr denotes condition-related errors involving missing final-condition use or obsolete-condition use. Other metrics follow Table 1.

Along the reasoning-difficulty axis, audio error increases from 9.46% on L1 to 48.90% on L3, suggesting that the L-axis reflects increasing reasoning burden. The rise of Mention-only cases further suggests a condition-use gap: in harder cases, models may state the final valid condition without reliably using it as an active reasoning premise. This trend validates the need to evaluate correction complexity and reasoning complexity jointly, since either axis can amplify failures in condition-state maintenance.

Together, these results highlight final-condition adoption failure: correction-relevant information may be recoverable from speech and even mentioned in the response, but not consistently used as an effective premise in the reasoning chain. We treat this as an observable output pattern rather than an asserted internal mechanism. These findings motivate condition-state maintenance as a targeted optimization objective: models should both use final valid conditions as active reasoning premises and suppress obsolete ones. Additional results and a representative example are provided in Appendix D.

4 PACER: Perception-Aware Condition-Enhanced Reinforcement

4.1 Optimization Framework

Motivated by the three-path analysis, which shows that recovered conditions are not always used as reasoning premises, PACER optimizes the language-side policy under self-corrected speech input while keeping the speech encoder frozen. It is perception-aware only in that optimization depends on speech representations and their perturbations;

it targets observable condition-use behavior rather than general-purpose transcription accuracy.

Given a self-corrected speech input x , the frozen encoder produces $z = E_{\text{sp}}(x)$. The old policy $\pi_{\theta_{\text{old}}}$ samples G candidate responses $\{y_i\}_{i=1}^G$ conditioned on z . Each response receives a scalar reward R_i , which is normalized within the sampled group to obtain the GRPO advantage:

$$\hat{A}_i = \frac{R_i - \mu_R}{\sigma_R + \epsilon_{\text{std}}}, \quad (3)$$

where μ_R and σ_R are the group-level reward mean and standard deviation, and ϵ_{std} is a small constant for numerical stability.

For token-level optimization, the policy ratio is

$$r_{i,t}(\theta) = \frac{\pi_{\theta}(y_{i,t}|y_{i,<t},z)}{\pi_{\theta_{\text{old}}}(y_{i,t}|y_{i,<t},z)}. \quad (4)$$

PACER uses the clipped GRPO surrogate:

$$\ell_{i,t}^{\text{clip}}(\theta) = \min \left(r_{i,t}(\theta) \hat{A}_i, \text{clip}(r_{i,t}(\theta), 1 - \epsilon_{\text{clip}}, 1 + \epsilon_{\text{clip}}) \hat{A}_i \right). \quad (5)$$

This surrogate provides the GRPO backbone (Shao et al., 2024). PACER differs from standard GRPO through two task-specific components: structured condition-aware rewards and speech-evidence contrast. Because final valid conditions are often recoverable in the transcription path, the contrast term is not designed to improve general ASR or low-level speech perception. Instead, perturbing speech representations provides a speech-side sensitivity signal, while structured rewards guide this sensitivity toward condition-grounded reasoning: adopting final valid conditions, suppressing obsolete ones, and grounding the final answer in the updated condition state.

Setting	Mention-Only ↓	Adopt. ↑	OldCond ↓	CondErr/All ↓	CondErr/Err ↓	CUF/CondErr ↓	Err. ↓
<i>Condition-score weight</i>							
$\lambda_c = 0.20$	8.97	90.14	0.72	5.38	31.24	38.06	17.23
$\lambda_c = 0.30$	8.39	90.07	0.83	5.26	30.82	41.22	17.07
$\lambda_c = 0.35$	7.96	90.64	0.58	4.70	27.99	41.03	16.79
$\lambda_c = 0.40$	9.01	89.55	0.65	4.74	27.31	44.92	17.35
<i>Speech mask ratio</i>							
Mask = 0.3	8.76	90.41	0.61	4.62	27.64	49.57	16.71
Mask = 0.5	7.96	90.64	0.58	4.70	27.99	41.03	16.79
Mask = 0.7	8.78	90.29	0.37	5.46	31.19	36.76	17.51
Mask = 0.9	8.22	90.42	0.67	5.38	31.83	40.30	16.91
<i>Speech-evidence contrast coefficient</i>							
$\gamma = 0.0005$	8.40	90.54	0.67	5.18	31.08	42.64	16.67
$\gamma = 0.001$	7.96	90.64	0.58	4.70	27.99	41.03	16.79
$\gamma = 0.002$	7.91	90.02	0.74	5.26	30.68	41.22	17.15
$\gamma = 0.005$	7.95	90.11	0.79	5.06	28.70	46.03	17.63
$\gamma = 0.01$	8.90	90.59	0.63	4.74	27.63	42.37	17.15

Table 5: Parameter sensitivity of PACER. All metric values are percentages. Metric definitions follow Table 4. The default setting is $\lambda_c = 0.35$, Mask = 0.5, and $\gamma = 0.001$.

4.2 Structured Condition-Aware Reward

Final-answer rewards alone are insufficient for self-corrected speech reasoning: a model may reach the correct answer by chance, rely on obsolete conditions, or merely mention a valid condition without using it. PACER therefore uses a structured condition-aware reward that evaluates condition adoption, reasoning consistency, answer correctness, and output structure.

For the i -th candidate response y_i , the overall reward is

$$R_i = w_j R_i^{\text{judge}} + w_f R_i^{\text{fmt}}, \quad (6)$$

where R_i^{judge} is the primary semantic reward, R_i^{fmt} is a lightweight format regularizer, and w_j, w_f are their weights. In PACER, answer correctness is mainly assessed by the online judge rather than by surface-form answer matching.

The online judge maps y_i together with $(\mathcal{V}_i, \mathcal{O}_i, A_i)$ into three structured signals: condition adoption S_i^{cond} , reasoning consistency S_i^{logic} , and answer correctness S_i^{ans} . Each response is parsed into condition, reasoning, and final-answer fields, and three Qwen3-14B (Yang et al., 2025a) judge prompts produce structured JSON outputs for condition evidence extraction, reasoning-process verification, and final-answer checking. These outputs distinguish mentioned versus actually used conditions, identify reasoning errors, and verify final-condition-grounded answers; scalar scores are computed by predefined aggregation rules. The

judge reward is

$$R_i^{\text{judge}} = \text{clip} \left(\lambda_c S_i^{\text{cond}} + \lambda_l S_i^{\text{logic}} + \lambda_a S_i^{\text{ans}}, 0, 1 \right), \quad (7)$$

where all scalar weights are hyperparameters.

The condition score combines positive adoption evidence with condition-level penalties. Let m_i be the mention ratio of final valid conditions and u_i their use ratio in the reasoning chain:

$$S_i^{\text{cond}} = \text{clip} \left(\alpha_m m_i + \alpha_u u_i - \sum_{e \in \mathcal{C}^-} \kappa_e g_i^e, 0, 1 \right). \quad (8)$$

Here, $\mathcal{C}^-$ covers obsolete-condition reliance, old-new conflicts, invalid condition usage, and wrong-premise-driven reasoning; g_i^e and κ_e denote the trigger strength and penalty weight for error type e . This separates shallow mention from effective condition adoption.

The logic score evaluates reasoning under the adopted conditions, without redundantly penalizing premise-level condition errors already handled by (S_i^{cond}) . Let $(\mathcal{V}^{\text{logic}})$ denote reasoning verdicts such as chain completeness, computational correctness, state-update consistency, self-consistency, and final-answer support. Let $(v_i^k \in \{0, 1\})$ indicate whether the k -th verdict is satisfied, and let $(\mathcal{E}_i^{\text{logic}})$ be the set of triggered reasoning errors. We summarize the logic scorer as

$$S_i^{\text{logic}} = \text{clip} \left(\sum_{k \in \mathcal{V}^{\text{logic}}} \mu_k v_i^k - \sum_{e \in \mathcal{E}_i^{\text{logic}}} \nu_e, 0, 1 \right). \quad (9)$$

where μ_k weights the contribution of the k -th reasoning verdict and ν_e penalizes the e -th reasoning error.

The answer score (S_i^{ans}) is binary: 1 only if the answer is correct and grounded in the final valid conditions, and 0 otherwise. Finally, (R_i^{fnt}) encourages a stable and parseable “condition–reasoning–answer” structure without assessing mathematical correctness. Additional prompt templates, parsing rules, score definitions, and hyperparameter settings are detailed in Appendix F.

4.3 Speech-Evidence Contrast and PACER Objective

Motivated by the diagnostic analysis, PACER extends GRPO-based reinforcement learning (Shao et al., 2024) to target the gap between speech-side condition recovery and downstream condition use: corrected conditions may be recoverable from speech, but still fail to serve as active reasoning premises. We use speech-evidence contrast not as a generic ASR-improvement objective, but as an auxiliary sensitivity signal whose behavioral effect is guided by structured rewards toward final-condition adoption and condition-grounded reasoning.

Given the intact speech representation $z = E_{\text{sp}}(x)$, we apply a random representation-level mask to obtain $\tilde{z} = M(z)$. PACER measures the next-token distribution discrepancy between the intact and perturbed branches:

$$D_{i,t}^{\text{sp}} = D_{\text{KL}}(\pi_{\theta}(\cdot | y_{i,<t}, z) \parallel \pi_{\theta}(\cdot | y_{i,<t}, \tilde{z})). \quad (10)$$

Since the mask is random and not condition-aware, $D_{i,t}^{\text{sp}}$ is not a correction-span detector. Inspired by perception-aware policy optimization (Wang et al., 2026b), we use it as a speech-side contrastive sensitivity term that encourages the policy to remain responsive to speech-side evidence, while the structured reward determines whether this sensitivity is translated into correct final-condition use. Importantly, random masking is not intended to identify the corrected condition and is therefore not a direct condition-adoption objective. Final-condition adoption is directly encouraged by the condition-aware structured reward, whereas speech-evidence contrast serves as an auxiliary regularizer that discourages the optimized policy from becoming insensitive to speech-side evidence. Thus, the reward specifies what condition-use behavior to reinforce, while the contrast helps preserve the speech sensitivity needed to support that behavior.

To stabilize training, PACER applies an entropy constraint over both branches. For $b \in \{z, \tilde{z}\}$, the

token-level predictive entropy is

$$H_{i,t}^b = - \sum_{v \in \mathcal{V}_{\text{tok}}} \pi_{\theta}(v | y_{i,<t}, b) \log \pi_{\theta}(v | y_{i,<t}, b). \quad (11)$$

The entropy regularizer is

$$\Omega_{i,t}^{\text{ent}} = \sum_{b \in \{z, \tilde{z}\}} \eta_b H_{i,t}^b, \quad (12)$$

where η_b is the branch-specific entropy coefficient.

The final PACER objective is

$$J_{\text{PACER}}(\theta) = \mathbb{E}_{\{y_i\}_{i=1}^G \sim \pi_{\theta_{\text{old}}}(\cdot | z)} \left[\frac{1}{G} \sum_{i=1}^G \frac{1}{|y_i|} \sum_{t=1}^{|y_i|} \left(\ell_{i,t}^{\text{clip}}(\theta) - \beta D_{i,t}^{\text{ref}} + \gamma D_{i,t}^{\text{sp}} - \Omega_{i,t}^{\text{ent}} \right) \right], \quad (13)$$

where $D_{i,t}^{\text{ref}}$ is the token-level KL penalty to the frozen reference policy, $D_{i,t}^{\text{sp}}$ is the speech-evidence contrast term, and β and γ control their strengths. The term $\ell_{i,t}^{\text{clip}}(\theta)$ follows the clipped policy objective in GRPO, while the additional contrast and entropy terms introduce speech-evidence sensitivity and training stabilization.

5 Experiments

5.1 Experimental Setup

We use Qwen2.5-Omni-7B (Xu et al., 2025) as the base Speech LLM and evaluate on the full test split. During RL, we freeze the speech encoder and update only the text-side policy. To balance the L/D matrix, we sample 500 examples from each valid cell, yielding 8,500 training examples.

We compare NoTrain, GRPO-Answer, GRPO-Structured, and PACER, corresponding to no task-specific training, final-answer-only GRPO, structured condition-aware GRPO, and our full method with speech-evidence contrast, respectively. To isolate the effects of textual input, external ASR, and supervised fine-tuning, we additionally include: (1) a Whisper-cascaded baseline that first transcribes speech using Whisper large-v3 (Radford et al., 2023) and then feeds the transcription to Qwen2.5-Omni-7B for text-based reasoning without task-specific ASR fine-tuning; (2) a Gold-text baseline; and (3) an SFT-only baseline with frozen perception encoders. We report the diagnostic metrics defined in Section 2.3; additional diagnostic measures, model-selection criteria, and implementation details for the SFT and Whisper baselines are provided in Appendix C.1, while PACER-specific hyperparameters are provided in Appendix F.1.

5.2 Main Results and Ablations

Table 4 summarizes the main results and ablations. Compared with NoTrain, PACER reduces the overall error rate from 27.91% to 16.79%, a 39.8% relative reduction, and CondErr/All from 14.46% to 4.70%, a 67.5% relative reduction. Mention-Only decreases from 16.90% to 7.96%, corrected-condition adoption increases from 83.46% to 90.64%, and obsolete-condition usage drops from 9.74% to 0.58%. Among the settings reported in Table 4, PACER achieves the lowest CondErr/All and overall error rate, while maintaining one of the lowest Mention-Only rates.

The additional baselines further clarify the source of condition-update failures. Gold-text NoTrain reduces the overall error rate to 21.77% and CondErr/All to 6.99%, while its Logic Accuracy remains nearly unchanged relative to NoTrain (83.47% vs. 83.42%), indicating that removing speech-side uncertainty facilitates condition recovery but does not eliminate downstream reasoning errors. By contrast, Whisper ASR + Qwen2.5-Omni reaches an overall error rate of 32.65% and CondErr/All of 18.63%, with CUF/CondErr remaining essentially unchanged (75.22% vs. 75.00%); this degradation is specific to the evaluated cascade rather than ASR-mediated reasoning in general.

SFT-only exhibits a different trade-off. It achieves the highest corrected-condition adoption rate (92.90%), the lowest obsolete-condition usage (0.51%), and the lowest CondErr/Err and CUF/CondErr (21.90% and 15.93%). However, its Logic Accuracy after condition adoption is only 70.12%, compared with 87.15% for PACER, and its overall error rate reaches 33.37%. Additional fine-grained diagnostic metrics are reported in Appendix D. Thus, successful condition adoption does not necessarily imply successful downstream reasoning. PACER instead achieves a stronger balance between maintaining the updated condition state and reasoning correctly from it.

Among the reinforcement-learning variants, GRPO-Answer reduces the overall error rate to 19.12%, while structured condition-aware rewards further improve it to 18.27%. PACER lowers it to 16.79% and further reduces CondErr/All from 5.22% to 4.70%, supporting the complementary role of speech-evidence contrast when guided by structured supervision. Reward-component ablations show the same overall pattern: removing any component weakens the balance between answer

correctness and condition-state maintenance, although individual metrics may occasionally improve.

5.3 Sensitivity Analysis and Remaining Errors

Table 5 shows that PACER is relatively stable across moderate hyperparameter settings, although no single configuration dominates every metric. The default condition-score weight ($\lambda_c = 0.35$) achieves the lowest overall error rate (16.79%) and CondErr/All (4.70%) among the tested values, while alternative weights improve only isolated condition-related metrics.

Similar trade-offs appear for the speech mask ratio and contrast coefficient. Mask (= 0.3) and $\gamma = 0.0005$ yield the lowest overall error rates (16.71% and 16.67%, respectively), whereas the default Mask (= 0.5) and $\gamma = 0.001$ provide a better balance across condition-state metrics. Larger perturbations or contrast coefficients do not consistently improve performance, supporting the role of speech-evidence contrast as an auxiliary sensitivity signal rather than an independently dominant objective.

Remaining errors mainly involve incomplete use of multiple corrected conditions, long-chain calculation mistakes, and complex retrospective corrections, suggesting that revised condition states become harder to maintain when updates interact with multi-step reasoning.

6 Conclusion

This paper studies self-corrected speech reasoning as a condition-update problem. We build a controlled Chinese benchmark with structured condition annotations and a three-path diagnostic protocol. Our analysis shows that Speech LLMs may recover or mention corrected conditions without using them as active reasoning premises, leading to final-condition adoption failures and obsolete-condition leakage.

We propose PACER, which combines structured condition-aware rewards with speech-evidence contrast under GRPO. On Qwen2.5-Omni-7B, PACER improves end-to-end speech reasoning while reducing condition-related errors and condition-update failures, suggesting that robust self-corrected speech reasoning requires active condition-state maintenance beyond transcription accuracy or final-answer supervision.

7 Limitations

This work has several limitations. First, SCUR-Bench is a controlled diagnostic benchmark based primarily on synthesized Chinese single-turn speech and therefore does not fully capture spontaneous phenomena such as hesitations, prosodic variation, background noise, implicit corrections, or cross-turn revisions. D4 and D5 should accordingly be viewed as controlled single-turn stress tests rather than naturalistic models of multi-turn correction.

Second, the English parallel evaluation provides preliminary evidence that the recovery–adoption gap is not unique to Chinese, but it preserves the reasoning and correction structures of the Chinese test set and is evaluated on only one model. Broader multilingual conclusions require independently collected speech across additional languages, speakers, and model families.

Finally, PACER is currently trained and evaluated only on Qwen2.5-Omni-7B, so its effectiveness across different Speech LLM architectures remains unverified. We also do not systematically evaluate broader capability preservation. Although freezing the speech encoder and Talker and applying KL regularization help limit capability drift, and D0 accuracy improves from 85.19% to 87.75%, these results do not establish preservation of capabilities such as speaker identification, emotion recognition, general audio understanding, or speech-generation quality.

References

- Ron Artstein and Massimo Poesio. 2008. [Survey article: Inter-coder agreement for computational linguistics](#). *Computational Linguistics*, 34(4):555–596.
- Ted Byrt, Janet Bishop, and John B. Carlin. 1993. [Bias, prevalence and kappa](#). *Journal of Clinical Epidemiology*, 46(5):423–429.
- Jean Carletta. 1996. [Assessing agreement on classification tasks: The kappa statistic](#). *Computational Linguistics*, 22(2):249–254.
- Qian Chen, Luyao Cheng, Chong Deng, Xiangang Li, Jiaqing Liu, Chao-Hong Tan, Wen Wang, Junhao Xu, Jieping Ye, Qinglin Zhang, Qiquan Zhang, and Jingren Zhou. 2025. [Fun-Audio-Chat technical report](#). *arXiv preprint arXiv:2512.20156*.
- Yiming Chen, Xianghu Yue, Chen Zhang, Xiaoxue Gao, Robby T. Tan, and Haizhou Li. 2026. [VoiceBench: Benchmarking llm-based voice assistants](#). *Transactions of the Association for Computational Linguistics*, 14:378–398.
- Yunfei Chu, Jin Xu, Qian Yang, Haojie Wei, Xipin Wei, Zhifang Guo, Yichong Leng, Yuanjun Lv, Jinzheng He, Junyang Lin, Chang Zhou, and Jingren Zhou. 2024. [Qwen2-Audio technical report](#). *arXiv preprint arXiv:2407.10759*.
- Yunfei Chu, Jin Xu, Xiaohuan Zhou, Qian Yang, Shiliang Zhang, Zhijie Yan, Chang Zhou, and Jingren Zhou. 2023. [Qwen-Audio: Advancing universal audio understanding via unified large-scale audio-language models](#). *arXiv preprint arXiv:2311.07919*.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. [Training verifiers to solve math word problems](#). *arXiv preprint arXiv:2110.14168*.
- Jacob Cohen. 1960. [A coefficient of agreement for nominal scales](#). *Educational and Psychological Measurement*, 20(1):37–46.
- DeepSeek-AI. 2026. [DeepSeek-V4-Flash](#). API released: 2026-04-24; Accessed: 2026-05-26.
- Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. [DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2368–2378, Minneapolis, Minnesota. Association for Computational Linguistics.
- Qingkai Fang, Yan Zhou, Shoutao Guo, Shaolei Zhang, and Yang Feng. 2025. [LLaMA-omni 2: LLM-based real-time spoken chatbot with autoregressive streaming speech synthesis](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 18617–18629, Vienna, Austria. Association for Computational Linguistics.
- Alvan R. Feinstein and Domenic V. Cicchetti. 1990. [High agreement but low kappa: I. the problems of two paradoxes](#). *Journal of Clinical Epidemiology*, 43(6):543–549.
- Google DeepMind. 2026. [Gemini 3.1 Pro](#). Accessed: 2026-05-26.
- Advait Gosai, Tyler Vuong, Utkarsh Tyagi, Steven Li, Wenjia You, Miheer Bavare, Arda Uçar, Zhongwang Fang, Brian Jang, Bing Liu, and Yunzhong He. 2025. [Audio MultiChallenge: A multi-turn evaluation of spoken dialogue systems on natural human interaction](#). *arXiv preprint arXiv:2512.14865*.
- Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shiron Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao,

- Zhuoshu Li, Ziyi Gao, Aixin Liu, and 175 others. 2025. [DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning](#). *Nature*, 645(8081):633–638.
- Aditya Gupta, Jiacheng Xu, Shyam Upadhyay, Diyi Yang, and Manaal Faruqi. 2021. [Disfl-QA: A benchmark dataset for understanding disfluencies in question answering](#). In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3309–3319, Online. Association for Computational Linguistics.
- Matthew Honnibal and Mark Johnson. 2014. [Joint incremental disfluency detection and dependency parsing](#). *Transactions of the Association for Computational Linguistics*, 2:131–142.
- Yixuan Hou, Heyang Liu, Yuhao Wang, Ziyang Cheng, Ronghua Wu, Qunshan Gu, Yanfeng Wang, and Yu Wang. 2025. [SOVA-Bench: Benchmarking the speech conversation ability for LLM-based voice assistant](#). In *Interspeech 2025*, pages 5713–5717.
- Wenxuan Huang, Bohan Jia, Zijie Zhai, Shaosheng Cao, Zheyu Ye, Fei Zhao, Zhe Xu, Yao Hu, and Shao-hui Lin. 2026. Vision-r1: Incentivizing reasoning capability in multimodal large language models. In *The Fourteenth International Conference on Learning Representations*.
- Willem J. M. Levelt. 1983. [Monitoring and self-repair in speech](#). *Cognition*, 14(1):41–104.
- Gang Li, Jizhong Liu, Heinrich Dinkel, Yadong Niu, Junbo Zhang, and Jian Luan. 2025a. [Reinforcement learning outperforms supervised fine-tuning: A case study on audio question answering](#). *arXiv preprint arXiv:2503.11197*.
- Zongxia Li, Wenhao Yu, Chengsong Huang, Rui Liu, Zhenwen Liang, Fuxiao Liu, Jingxi Che, Dian Yu, Jordan Boyd-Graber, Haitao Mi, and Dong Yu. 2025b. [Self-rewarding vision-language model via reasoning decomposition](#). *arXiv preprint arXiv:2508.19652*.
- Guan-Ting Lin, Chen Chen, Zhehuai Chen, and Hung-yi Lee. 2026. [Full-Duplex-Bench-v3: Benchmarking tool use for full-duplex voice agents under real-world disfluency](#). *arXiv preprint arXiv:2604.04847*.
- Hongcheng Liu, Yixuan Hou, Heyang Liu, Yuhao Wang, Yanfeng Wang, and Yu Wang. 2025. [VocalBench-DF: A benchmark for evaluating speech llm robustness to disfluency](#). *arXiv preprint arXiv:2510.15406*.
- Smit Nautambhai Modi, Gandharv Mahajan, Marc Weter, and Randall Welles. 2026. [EchoChain: A full-duplex benchmark for state-update reasoning under interruptions](#). *arXiv preprint arXiv:2604.16456*.
- OpenAI. 2026. [GPT-5.4](#). Accessed: 2026-05-26.
- OpenBMB. 2025. MiniCPM-o 2.6: A GPT-4o level MLLM for vision, speech, and multimodal live streaming on your phone. <https://openbmb.vercel.app/minicpm-o-2-6-en>. Technical report.
- Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. [Are NLP models really able to solve simple math word problems?](#) In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2080–2094, Online. Association for Computational Linguistics.
- Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine Mcleavey, and Ilya Sutskever. 2023. Robust speech recognition via large-scale weak supervision. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 28492–28518. PMLR.
- Ahmad Rezaei, Mohsen Gholami, Saeed Ranjbar Alvar, Kevin Cannons, Mohammad Asiful Hossain, Weimin Zhou, Shunbo Zhou, Yong Zhang, and Mohammad Akbari. 2026. [CPPO: Contrastive perception for vision language policy optimization](#). *arXiv preprint arXiv:2601.00501*.
- Paul K. Rubenstein, Chulayuth Asawaroengchai, Duc Dung Nguyen, Ankur Bapna, Zalán Borsoš, Félix de Chaumont Quitry, Peter Chen, Dalia El Badawy, Wei Han, Eugene Kharitonov, Hannah Muckenhirn, Dirk Padfield, James Qin, Danny Rozenberg, Tara Sainath, Johan Schalkwyk, Matt Sharifi, Michelle Tadmor Ramanovich, Marco Tagliasacchi, and 11 others. 2023. [AudioPaLM: A large language model that can speak and listen](#). *arXiv preprint arXiv:2306.12925*.
- Emanuel A. Schegloff, Gail Jefferson, and Harvey Sacks. 1977. [The preference for self-correction in the organization of repair in conversation](#). *Language*, 53(2):361–382.
- John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. [Proximal policy optimization algorithms](#). *arXiv preprint arXiv:1707.06347*.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. [DeepSeekMath: Pushing the limits of mathematical reasoning in open language models](#). *Preprint*, arXiv:2402.03300.
- Elizabeth Ellen Shriberg. 1994. *Preliminaries to a Theory of Speech Disfluencies*. Ph.d. dissertation, University of California, Berkeley, Berkeley, CA.
- Huajie Tan, Yuheng Ji, Xiaoshuai Hao, Xiansheng Chen, Pengwei Wang, Zhongyuan Wang, and Shanghang Zhang. 2025. [Reason-RFT: Reinforcement fine-tuning for visual reasoning of vision language models](#). In *The Thirty-Ninth Annual Conference on Neural Information Processing Systems*. NeurIPS 2025 Poster.

- Changli Tang, Wenyi Yu, Guangzhi Sun, Xianzhao Chen, Tian Tan, Wei Li, Lu Lu, Zejun MA, and Chao Zhang. 2024. [SALMONN: Towards generic hearing abilities for large language models](#). In *The Twelfth International Conference on Learning Representations*.
- Dingdong Wang, Junan Li, Jincenzi Wu, Dongchao Yang, Xueyuan Chen, Tianhua Zhang, and Helen M. Meng. 2026a. [MMSU: A massive multi-task spoken language understanding and reasoning benchmark](#). In *The Fourteenth International Conference on Learning Representations*.
- Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xiong-Hui Chen, Jianxin Yang, Zhenru Zhang, Yuqiong Liu, An Yang, Andrew Zhao, Yang Yue, Shiji Song, Bowen Yu, Gao Huang, and Junyang Lin. 2025. [Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for LLM reasoning](#). In *Advances in Neural Information Processing Systems*, volume 38.
- Zhenhailong Wang, Xuehang Guo, Sofia Stoica, Haiyang Xu, Hongru WANG, Hyeonjeong Ha, Xiushi Chen, Yangyi Chen, Ming Yan, Fei Huang, and Heng Ji. 2026b. [Perception-aware policy optimization for multimodal reasoning](#). In *The Fourteenth International Conference on Learning Representations*.
- Xumeng Wen, Zihan Liu, Shun Zheng, Shengyu Ye, Zhirong Wu, Yang Wang, Zhijian Xu, Xiao Liang, Junjie Li, Ziming Miao, Jiang Bian, and Mao Yang. 2026. [Reinforcement learning with verifiable rewards implicitly incentivizes correct reasoning in base LLMs](#). In *The Fourteenth International Conference on Learning Representations*.
- Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M. Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. 2015. [Towards AI-Complete question answering: A set of prerequisite toy tasks](#). *arXiv preprint arXiv:1502.05698*.
- Junru Wu, Tianhao Shen, Linxi Su, and Deyi Xiong. 2025. [C²RBench: A Chinese complex reasoning benchmark for large language models](#). In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 21031–21050, Vienna, Austria. Association for Computational Linguistics.
- Wenyi Xiao and Leilei Gan. 2025. [Fast-slow thinking GRPO for large vision-language model reasoning](#). In *Advances in Neural Information Processing Systems. NeurIPS 2025 Spotlight*.
- Zhifei Xie and Changqiao Wu. 2024. [Mini-Omni: Language models can hear, talk while thinking in streaming](#). *arXiv preprint arXiv:2408.16725*.
- Jin Xu, Zhifang Guo, Jinzheng He, Hangrui Hu, Ting He, Shuai Bai, Keqin Chen, Jialin Wang, Yang Fan, Kai Dang, Bin Zhang, Xiong Wang, Yunfei Chu, and Junyang Lin. 2025. [Qwen2.5-Omni technical report](#). *arXiv preprint arXiv:2503.20215*.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025a. [Qwen3 technical report](#). *Preprint*, arXiv:2505.09388.
- Wanqi Yang, Yanda Li, Yunchao Wei, Meng Fang, and Ling Chen. 2025b. [SpeechR: A benchmark for speech reasoning in large audio-language models](#). *arXiv preprint arXiv:2508.02018*.
- Qiyang Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gao-hong Liu, Juncai Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, and 17 others. 2025. [DAPO: An open-source LLM reinforcement learning system at scale](#). In *Advances in Neural Information Processing Systems*, volume 38.
- Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. 2025. [Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model?](#) *arXiv preprint arXiv:2504.13837*.
- Aohan Zeng, Zhengxiao Du, Mingdao Liu, Kedong Wang, Shengmin Jiang, Lei Zhao, Yuxiao Dong, and Jie Tang. 2024. [GLM-4-Voice: Towards intelligent and human-like end-to-end spoken chatbot](#). *arXiv preprint arXiv:2412.02612*.
- Dong Zhang, Shimin Li, Xin Zhang, Jun Zhan, Pengyu Wang, Yaqian Zhou, and Xipeng Qiu. 2023. [SpeechGPT: Empowering large language models with intrinsic cross-modal conversational abilities](#). In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 15757–15773, Singapore. Association for Computational Linguistics.
- Jingyi Zhang, Jiaxing Huang, Huanjin Yao, Shunyu Liu, Xikun Zhang, Shijian Lu, and Dacheng Tao. 2025. [R1-VL: Learning to reason with multimodal large language models via step-wise group relative policy optimization](#). In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1859–1869.
- Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, Jingren Zhou, and Junyang Lin. 2025. [Group sequence policy optimization](#). *arXiv preprint arXiv:2507.18071*.

A Additional Related Work

A.1 Speech-Language Models and Spoken-Agent Benchmarks

Recent advances in speech-language and omni-modal large language models have enabled increasingly natural spoken interactions, where systems are expected to understand spoken input

and, in some cases, generate spoken responses in an end-to-end manner. Representative models include SpeechGPT (Zhang et al., 2023), SALMONN (Tang et al., 2024), LLaMA-Omni 2 (Fang et al., 2025), and Qwen2.5-Omni (Xu et al., 2025). Related efforts further explore large-scale audio-language modeling, real-time spoken dialogue, and end-to-end voice interaction (Rubenstein et al., 2023; Chu et al., 2023, 2024; Xie and Wu, 2024; Zeng et al., 2024). These studies demonstrate the growing capability of large language models to process speech, audio, and multimodal signals within unified conversational frameworks.

Alongside model development, several benchmarks have been proposed to evaluate spoken language understanding, speech-centered reasoning, and voice-agent interaction. Examples include VoiceBench (Chen et al., 2026), MMSU (Wang et al., 2026a), SOVA-Bench (Hou et al., 2025), and SpeechR (Yang et al., 2025b). These benchmarks cover important aspects of speech-based intelligence, including audio understanding, instruction following, spoken reasoning, and conversational evaluation. However, they generally do not specifically target the problem of tracking corrected task conditions in self-repaired spoken input. This gap motivates our focus on self-correction and condition updating, which we discuss next.

A.2 Self-Repair, Disfluency Robustness, and Condition Updating

Self-repair, also referred to as self-correction, is a common phenomenon in spontaneous speech. It has long been studied in conversation analysis and psycholinguistics as part of repair organization and speech monitoring, where speakers detect problems in their own utterances and revise previously produced speech (Schegloff et al., 1977; Levelt, 1983). In spoken language processing, self-correction is often examined together with broader disfluency phenomena, including filled pauses, repetitions, restarts, and repairs. These phenomena have motivated research on disfluency analysis, detection, and removal (Shriberg, 1994; Honnibal and Johnson, 2014).

Recent studies further show that disfluent and non-canonical spoken inputs can degrade language understanding, spoken question answering, and voice-assistant performance, making robustness to natural speech an important challenge for speech and language models (Gupta et al., 2021; Liu et al., 2025; Gosai et al., 2025). Related benchmarks

on full-duplex and interruption-aware voice-agent evaluation also highlight the difficulty of maintaining correct interaction states when users revise, interrupt, or update previously stated information (Lin et al., 2026; Modi et al., 2026).

However, most disfluency-oriented evaluations treat disfluencies as generic input perturbations and primarily measure the overall degradation in accuracy or response quality. By contrast, self-correction may explicitly invalidate previously stated task conditions and introduce new final conditions. This requires models not only to tolerate noisy spoken input, but also to identify the corrected content, update the condition state, and reason according to the final valid information.

This perspective also distinguishes self-corrected speech reasoning from conventional reasoning benchmarks. Prior reasoning datasets often control task complexity through reasoning steps, condition counts, or discrete operations (Weston et al., 2015; Dua et al., 2019; Cobbe et al., 2021; Patel et al., 2021). In self-corrected speech, however, task difficulty arises from both the intrinsic complexity of solving the underlying problem and the structural complexity of revising conditions from spoken input. Therefore, in this work, we formalize self-correction as a condition-updating problem in speech reasoning and organize our diagnostic benchmark along two dimensions: intrinsic reasoning difficulty and self-correction difficulty.

A.3 Reinforcement Learning for Reasoning and Perception-Enhanced Optimization

Reinforcement learning has become a key paradigm for enhancing the reasoning capabilities of large language models (Schulman et al., 2017). Recent reasoning systems and training paradigms based on GRPO and RLVR have further advanced reinforcement-learning-based optimization for reasoning-oriented models (Shao et al., 2024; Guo et al., 2025; Wen et al., 2026). Subsequent studies continue to improve and analyze policy optimization algorithms and training dynamics for large-scale reasoning models (Yu et al., 2025; Yue et al., 2025; Zheng et al., 2025; Wang et al., 2025).

This paradigm has also been extended to multimodal reasoning, where R1-style reinforcement learning and GRPO-based optimization have been explored for vision-language models (Huang et al., 2026; Zhang et al., 2025; Tan et al., 2025; Xiao

and Gan, 2025). Nevertheless, many such methods still rely heavily on verifiable final answers or rule-based rewards, which may be sparse and may fail to reveal whether the model has correctly perceived, grounded, and used task-relevant evidence (Wang et al., 2026b; Li et al., 2025b). This limitation is particularly important for self-corrected speech reasoning, because answer correctness alone cannot guarantee that the model has recovered the corrective information, updated the condition state, and consistently followed the final valid conditions.

A related line of work addresses this issue through perception-aware or perception-enhanced optimization, treating perceptual failure as a major source of errors in multimodal reasoning (Wang et al., 2026b; Li et al., 2025b; Rezaei et al., 2026). Inspired by this perspective, PACER introduces structured condition-awareness rewards and speech-evidence contrast within the GRPO framework.

B Dataset Construction Details

B.1 Overall Construction Pipeline

We construct the dataset through a multi-stage pipeline that combines AI-assisted generation, iterative quality control, manual verification, structured annotation, and speech synthesis. Starting from clean Chinese seed questions, the pipeline produces difficulty-preserving variants, inserts spoken self-correction expressions, annotates structured reasoning labels, and synthesizes the final audio inputs.

The dataset itself is built in Chinese. The question field in the released data contains Chinese spoken-style reasoning questions, and speech synthesis is performed from this Chinese text. For readability in this paper, some examples in the tables are shown as English translations of the original Chinese records. The released construction code implements the full pipeline with executable scripts for question augmentation, self-correction insertion, label generation and checking, and TTS synthesis.

B.2 Difficulty Definitions and Data Distribution

We define dataset difficulty along two orthogonal dimensions: intrinsic reasoning difficulty and self-correction difficulty. The intrinsic reasoning axis is inspired by text-based reasoning QA and complex reasoning benchmarks that control dif-

ficulty through reasoning steps, variable counts, and logical branches (Weston et al., 2015; Cobbe et al., 2021; Wu et al., 2025). In contrast, the self-correction axis is designed to capture the structural complexity of spoken condition revision. Table 7 summarizes the definitions used during data synthesis and quality control.

Table 6 reports the data distribution across reasoning difficulty levels and self-correction difficulty levels. The original Train+Dev pool contains 10,805 examples, of which 405 are held out as SCUR-Dev, leaving 10,400 examples in SCUR-Train; SCUR-Test contains 2,490 examples. The L1–D5 combination is not included because long-distance retrospective correction is mainly used for questions with richer condition structures and is often unnatural for single-step problems.

B.3 Question Generation and Quality Control

Our data synthesis process uses a family of Chinese prompts rather than a single prompt. These prompts cover clean-question augmentation, candidate quality control, failed-case regeneration, self-correction insertion, self-correction quality control, label generation, and label quality control. Due to space constraints, Table 8 summarizes the prompt families and their key constraints; the full raw prompts, validation instructions, generation scripts, configuration templates, and output schemas are provided in the supplementary materials and released code.

We first prepare clean seed questions according to predefined reasoning difficulty levels. These questions contain no self-correction expressions and are inspected for commonsense plausibility, solvability, self-consistency, answer determinacy, difficulty-level consistency, and TTS suitability. Duplicate or near-duplicate questions are flagged with LLM assistance and then manually removed or revised. To reduce source leakage, seed questions are split before augmentation.

For clean-question augmentation, each seed question is expanded into variants while preserving the original reasoning structure and difficulty. The prompts allow changes to numerical conditions, entities, roles, units, scenarios, and information order, but prohibit changes that substantially alter the reasoning difficulty or leak the answer. Generated candidates are checked by a QC model and categorized as pass, review, or reject. Candidates marked as review or reject are either discarded, manually inspected, or sent to regeneration. For

Logic difficulty	Split	D0	D1	D2	D3	D4	D5	All
L1	Train	750	668	605	666	607	–	3296
	Dev	25	25	25	25	25	–	125
	Test	196	171	151	164	111	–	793
L2	Train	802	644	638	634	559	585	3862
	Dev	25	25	25	25	25	25	150
	Test	183	125	133	145	133	111	830
L3	Train	675	540	527	500	500	500	3242
	Dev	25	25	25	23	12	20	130
	Test	168	154	139	143	112	151	867
All	Train	2227	1852	1770	1800	1666	1085	10400
	Dev	75	75	75	73	62	45	405
	Test	547	450	423	452	356	262	2490

Table 6: Distribution of SCUR-Bench across intrinsic reasoning difficulty and self-correction difficulty. SCUR-Train contains 10,400 instances, SCUR-Dev contains 405 instances, and SCUR-Test contains 2,490 instances. L1–L3 denote intrinsic reasoning difficulty and D0–D5 denote self-correction difficulty. “–” indicates an invalid L/D combination.

seed questions with insufficient accepted variants, a failed-case regeneration prompt uses the original seed question, accepted good cases, and failed review cases as context to produce additional candidates.

For self-corrected samples, corrections must target computable condition slots, such as numbers, units, prices, time, duration, distance, thresholds, proportions, frequencies, or quantities. Corrections are not allowed to target only wording clarification. The final corrected conditions must remain mathematically valid, commonsensically plausible, and uniquely solvable. When a correction affects dependent quantities, thresholds, or explanatory consequences, the dependent information is updated to avoid internal contradiction. Correction cues must appear before the final question request and should be natural for spoken Chinese.

The prompts also enforce level-specific correction structures. D1 follows a local immediate-correction pattern such as “8 yuan, no, 6 yuan”. D2 contains multiple immediate corrections over different condition slots. D3 follows a chained single-slot pattern such as “8 yuan, no, 7 yuan, no, 6 yuan”. D4 requires chained corrections over multiple independent slots. D5 uses a retrospective cue after intervening condition information, such as revising an earlier price after another complete condition has already been stated. This design ensures that the assigned D label reflects the actual correction structure rather than merely the presence of correction words.

Across all generation stages, samples are filtered

according to shared validity criteria: they must be natural, uniquely solvable, logically self-consistent, compatible with TTS synthesis, and consistent with the assigned L/D difficulty labels. Cases with inconsistent corrections, ambiguous answers, invalid arithmetic, answer leakage, unnatural spoken form, or mismatched difficulty labels are removed, revised, or regenerated.

B.4 Structured Annotation Schema

Each finalized self-corrected question is annotated with anchor labels used for diagnostic evaluation and reward computation. In our implementation, these fields are stored under anchors: `premises` corresponds to the final valid conditions $\mathcal{V}$, `invalid_info` corresponds to obsolete conditions $\mathcal{O}$, `logic_nodes` records the reference reasoning process, and `final_answers` stores one or more acceptable final answers. The value before the correction cue is recorded as `invalid_info`, while the corrected value is used as an effective premise for solving the problem.

The structured labels are created and verified together with the final question. During verification, we check whether each obsolete condition is actually superseded by a later correction, whether each final valid condition is necessary or relevant for the reference solution, and whether the reasoning nodes support the final answer. Cases with ambiguous answer sets, inconsistent corrections, invalid arithmetic, mismatched labels, missing final conditions, or unnatural spoken form are removed or revised. Accepted records may additionally contain

Level	Definition
L1	Single-step computation or direct reasoning based on a single condition. The problem can be solved through one explicit operation or decision.
L2	Two-to-four-step single-chain reasoning, or branch/decision making after a single-step operation over two to three variables. The problem requires combining multiple conditions in a clear reasoning order.
L3	Five-or-more-step reasoning over a single variable, or multi-variable, multi-step, and multi-branch reasoning. The problem may involve condition composition, variable binding, or branch selection.
D0	No self-correction. This level serves as the clean baseline for measuring the intrinsic reasoning difficulty of the original problem.
D1	Single immediate correction. One condition slot is revised once, following a local “wrong value $\rightarrow$ corrected value” pattern, with no intervening information between the erroneous value and the corrected value.
D2	Multiple immediate corrections. Two to three different condition slots are each revised once, and each revision follows a local “wrong value $\rightarrow$ corrected value” pattern.
D3	Single-slot chained correction. The same condition slot is overwritten two or three times consecutively, forming a local correction chain.
D4	Multi-slot chained correction. At least two different condition slots each contain a local correction chain, and the chains remain independent.
D5	Long-distance retrospective correction. The input contains multiple corrected computable slots, and at least one earlier condition is revised after intervening condition information using a weak retrospective cue.

Table 7: Definitions of intrinsic reasoning difficulty and self-correction difficulty. L1–L3 denote reasoning difficulty, while D0–D5 denote self-correction difficulty.

construction metadata such as `_qc_info`, including the accepted flag, QC epoch, and check time. This metadata is used for construction auditing and is not required by downstream training or evaluation.

Table 9 shows representative examples across self-correction difficulty levels. The examples are English translations for presentation; the released dataset stores the original Chinese questions and Chinese labels.

B.5 Speech Synthesis

We synthesize audio from the final annotated Chinese question field using Edge-TTS. For each utterance, one of five Mandarin voices is randomly selected: zh-CN-XiaoxiaoNeural, zh-CN-YunxiNeural, zh-CN-YunjianNeural, zh-CN-XiaoyiNeural, and zh-CN-YunyangNeural. Voice sampling is performed per utterance with a fixed random seed of 1234. The speaking rate is set to +15% and the volume to +0%. Synthesis is performed in asynchronous batches of 25. Each output audio file is named according to the sample ID, and the corresponding JSONL record stores the generated `wav_path`.

We perform a light TTS-compatibility check before synthesis and an output validation check after synthesis. The checks cover abnormal symbols, overly long sentences, unnatural pauses, pronunciation ambiguity, synthesis failures, and missing audio files. Failed or abnormal generations are

excluded or regenerated through script-based and manual filtering. The output JSONL preserves the original text and annotation fields and appends `wav_path`, allowing the final manifest to be directly consumed by downstream audio reasoning data loaders.

The released construction code provides the exact synthesis script, configurable environment variables, voice list, batching settings, and output metadata format. We do not specify sampling rate or codec in the paper because the synthesis script does not require these parameters as dataset-level configuration.

B.6 AI Assistance in Dataset Construction

During dataset construction, we use publicly available commercial LLM APIs for generation and verification assistance. Specifically, Gemini-3.1-Pro (Google DeepMind, 2026) and GPT-5.4 (OpenAI, 2026) are used for seed problem preparation and difficulty-preserving augmentation, while DeepSeek-V4-Flash (DeepSeek-AI, 2026) and GPT-5.4 are used for self-correction insertion, label generation, annotation checking, and consistency verification. All API-based assistance uses the provider-hosted model versions available during data construction and evaluation; prompts, decoding settings, and output schemas are released in the supplementary materials. Their outputs are not directly accepted as final annotations. Instead, generated candidates and labels are cross-checked

Prompt family	Purpose	Main constraints
Difficulty-preserving augmentation	Generate clean variants from seed reasoning problems	Preserve the overall reasoning difficulty and cognitive load of the seed problem; ensure solvability, naturalness, TTS friendliness, and diversity; prohibit self-correction expressions and answer leakage.
Candidate quality control	Filter generated clean variants	Check commonsense validity, mathematical consistency, ambiguity, TTS naturalness, difficulty preservation, diversity from accepted cases, and similarity to sibling candidates.
Failed-case regeneration	Regenerate candidates for insufficient or failed variants	Use accepted good cases and failed review cases as context; avoid repeated errors; maintain the original knowledge type and difficulty level; generate diverse new candidates.
Self-correction insertion	Inject D1–D5 self-correction patterns into clean questions	Insert correction expressions according to the target D level while preserving the original problem structure, reasoning type, and final solvability.
Self-correction quality control	Validate generated self-corrected questions	Check whether the correction structure matches the assigned D level, whether each correction falls on a computable condition slot, and whether the final question remains natural, unambiguous, and solvable.
Label generation and checking	Produce and verify structured anchors	Extract final valid conditions, obsolete conditions, reasoning nodes, and final answers; reject cases with ambiguous answers, inconsistent corrections, invalid arithmetic, or mismatched labels.

Table 8: Prompt families used in data synthesis and quality control.

through iterative QC, script-based validation, and manual review to filter inconsistent corrections, ambiguous answers, invalid reasoning paths, answer mismatches, unnatural spoken forms, and mismatched difficulty labels.

This AI-assisted and human-verified procedure improves construction efficiency while maintaining the validity of structured condition annotations. To support reproducibility, we provide the full raw prompts, validation instructions, generation scripts, TTS synthesis script, API configuration templates, decoding settings, and output schemas in the supplementary materials and released construction code.

B.7 Dataset Release Plan

The full dataset contains more than ten thousand paired text–audio instances, and the complete synthesized audio package is over 2GB. Due to file-size constraints, the supplementary materials include the complete metadata, annotation files, construction scripts, evaluation scripts, prompt files, configuration templates, and a representative subset of audio files. These materials are sufficient to inspect the data format, verify the annotation schema, reproduce the construction pipeline, and run the evaluation protocol on the provided subset.

The code, configuration files, prompts, and evaluation scripts are publicly available in our [GitHub](#)

[repository](#). The complete dataset, including all synthesized audio files, JSONL manifests, and structured annotations, will be released on Hugging Face, with the corresponding dataset link maintained in the GitHub repository.

B.8 Artifact License and Terms

The released artifacts are provided for research and reproducibility purposes. The dataset metadata, structured annotations, prompts, configuration files, construction scripts, training scripts, and evaluation scripts will be released under the license specified in the accompanying public repository. Users of the artifacts are responsible for following the terms of that license and for complying with the licenses and usage restrictions of any third-party components used with the artifacts, including underlying foundation models, model checkpoints, APIs, and text-to-speech services. Our release does not grant additional rights beyond those permitted by the corresponding third-party licenses and terms of use.

C Evaluation Details

C.1 Baseline and Model Selection Details

Selection of diagnostic models. The models used in the cross-model diagnostic analysis were selected to support a consistent comparison under the three-path protocol rather than to provide an exhaustive survey of available Speech LLMs.

Level	Question	Final valid conditions	Obsolete conditions	Answer
D0	There are 14 people ahead of you in a bank queue. Each person takes 3 minutes on average. How long do you expect to wait?	14 people ahead; 3 minutes per person	None	42 minutes
D1	In a livestream tipping scenario, each tip is 6 yuan, no, 7 yuan. You tip 9 times. What is the total amount?	7 yuan per tip; 9 tips	6 yuan per tip	63 yuan
D2	Each virtual gift costs 8 yuan, no, 10 yuan. You bought 7 gifts, sorry, 5 gifts. How much should you pay in total?	10 yuan per gift; 5 gifts	8 yuan per gift; 7 gifts	50 yuan
D3	In a livestream tipping scenario, each tip is 6 yuan, no, 7 yuan, no, 8 yuan. You tip 9 times. What is the total amount?	8 yuan per tip; 9 tips	6 yuan per tip; 7 yuan per tip	72 yuan
D4	An online order takes 15 minutes to pack, let me correct that, 12 minutes, no, 10 minutes. Delivery takes 20 minutes, sorry, 25 minutes, correction, 22 minutes. How long does it take from placing the order to receiving it?	Packing takes 10 minutes; delivery takes 22 minutes	Packing takes 15/12 minutes; delivery takes 20/25 minutes	32 minutes
D5	A wallet originally had 500 yuan. He spent 78 yuan on books, no, 85 yuan, and then received 120 yuan from his mother. Wait, I think the original wallet amount was wrong; it was actually 520 yuan. How much money is left now?	Initial amount is 520 yuan; book cost is 85 yuan; received 120 yuan	Initial amount is 500 yuan; book cost is 78 yuan	555 yuan

Table 9: Representative examples of structured annotations across self-correction difficulty levels. The examples are English translations of original Chinese dataset records. D0 contains no obsolete condition, while D1–D5 contain increasingly complex correction structures.

We primarily considered whether a model can directly accept raw speech input, support Chinese speech, produce textual responses suitable for structured reasoning analysis, and provide a stable checkpoint or reproducible inference interface. We also sought functional diversity across the selected systems. Accordingly, the evaluated models include Fun-Audio-Chat-8B (Chen et al., 2025), Qwen2.5-Omni-7B (Xu et al., 2025), MiniCPM-o 2.6 (OpenBMB, 2025), R1-AQA (Li et al., 2025a), and Qwen2-Audio-7B-Instruct (Chu et al., 2024), spanning general speech dialogue, unified omni-modal, general audio-language, and reasoning-oriented audio question-answering systems. This diversity allows us to examine whether condition-update failures are restricted to a particular model design or can also emerge across speech-capable systems with different training objectives and architectural emphases. Commercial or rapidly changing hosted systems were not required for this comparison because fixed and reproducible inference interfaces were prioritized.

Qwen2.5-Omni-7B was further selected as the base model for PACER because the optimization framework requires access to trainable parameters and intermediate speech representations, as well as

the ability to freeze speech-related modules while updating the language-side policy. Qwen2.5-Omni-7B satisfies these implementation requirements through its open and trainable architecture. Its diagnostic behavior is also closely aligned with the phenomenon studied in this work: the model recovers most final valid conditions in the transcription path, with a ValidCondRecall of 93.63%, while still exhibiting substantial condition-related errors in end-to-end speech reasoning. This recovery–adoption gap makes it a suitable base model for investigating whether condition-aware reinforcement can improve the downstream use of recovered speech information. The selection of Qwen2.5-Omni-7B for PACER does not imply that the method is architecture-specific; however, as discussed in the Limitations section, the current work does not establish the effectiveness of PACER across additional base models.

SFT-only baseline. The SFT-only baseline uses the same Qwen2.5-Omni-7B backbone and performs supervised fine-tuning directly from speech input to the target structured reasoning response. Each training example contains the waveform and a target textual response. The input prompt instructs the model to listen to the speech and produce three

fields corresponding to the known conditions, reasoning process, and final answer. The written form of the question is not provided as an additional input, so the model must obtain task information from the speech signal itself. The supervised loss is applied only to tokens belonging to the assistant target response; prompt-side and non-target tokens are masked from the language-modeling loss.

During SFT, the audio encoder and visual encoder are frozen, while parameters outside these frozen perception modules remain trainable. This setting isolates supervised adaptation of the downstream multimodal and language-side components without directly updating the speech encoder. Audio is converted to mono when necessary, resampled to 16 kHz, and truncated to a maximum duration of 30 s. The maximum multimodal token sequence length is 2,048. We explicitly keep the text-token truncation settings separate from the audio preprocessing settings so that the textual maximum length does not inadvertently truncate the waveform or its speech features.

The SFT model is trained for 2 epochs on two GPUs with a per-device batch size of 4 and gradient accumulation of 1, corresponding to an effective batch size of 8. We use a learning rate of 2×10^{-5} , a cosine learning-rate scheduler, and a warmup ratio of 0.05. Training is performed in bfloat16 with gradient checkpointing and DeepSpeed ZeRO-3. Checkpoints are saved every 200 optimization steps, with at most three checkpoints retained. No separate in-training evaluation file is supplied in this SFT configuration. The audio and visual perception encoders remain frozen throughout optimization. This baseline is therefore intended to test whether direct supervised adaptation alone is sufficient to learn explicit condition updating from the same type of speech input used by PACER.

Whisper large-v3 + Qwen2.5-Omni baseline. For the cascaded baseline, we first transcribe each speech input using Whisper large-v3 (Radford et al., 2023) and then provide the resulting transcription to the untrained Qwen2.5-Omni-7B for text-based reasoning. Whisper is used only for inference and receives no task-specific fine-tuning on SCUR-Bench. Transcription is performed with the transcribe task on CUDA using FP16 inference. The decoding language is explicitly specified according to the evaluation language; for example, language=en is used for the English parallel evaluation. Each utterance is transcribed independently

from its waveform, and the recognized text is stored together with the original sample metadata before being passed to the downstream reasoning stage.

The Whisper inference implementation distributes input files across available GPUs but does not otherwise modify the model or the decoding objective. In the two-GPU configuration used in our implementation, input files are partitioned across two independent workers, each of which loads Whisper large-v3 on its assigned GPU. Resume support is enabled so that already transcribed examples can be skipped after interrupted runs. For each successful transcription, we retain the recognized text, detected language, and segment-level output, although only the recognized transcription is required by the downstream reasoning baseline.

The second stage uses Qwen2.5-Omni-7B without task-specific training and treats the Whisper transcription as textual input for the same reasoning task. Thus, this baseline explicitly separates speech recognition from downstream reasoning and provides a comparison with direct end-to-end speech reasoning. It should not be interpreted as an optimized ASR-LLM cascade: neither Whisper nor the downstream Qwen2.5-Omni-7B is specifically adapted for this pipeline. Consequently, the performance of this baseline characterizes the particular Whisper large-v3 + Qwen2.5-Omni configuration evaluated here rather than the general capability of cascaded ASR-LLM systems.

C.2 Three-Path Diagnostic Protocol

We evaluate each model through three complementary paths: text reasoning, speech reasoning, and transcription. The text reasoning path takes the textual self-corrected question as input and evaluates whether the model can handle condition updates without speech perception difficulty. The speech reasoning path takes the corresponding speech input and serves as the main end-to-end evaluation setting. The transcription path asks the model to transcribe the speech input only, and is used to estimate whether final valid conditions are recoverable from the speech signal.

In both the text reasoning and speech reasoning paths, the model is instructed to produce a structured response containing known conditions, reasoning process, and final answer. In the transcription path, the model is explicitly instructed to only convert the speech into text and not answer the question. This separation allows us to compare speech-free reasoning, end-to-end speech reason-

ing, and speech-side condition recovery.

For reasoning outputs, we use LLM-assisted structured evaluators based on DeepSeek-V4-Flash (DeepSeek-AI, 2026) and GPT-5.4 (OpenAI, 2026), followed by human verification. The same evaluator set is used for transcription-side condition recovery checks. The evaluator receives the question, the model response, annotated final valid conditions, obsolete conditions, reference reasoning nodes, and acceptable final answers. It judges condition adoption, obsolete-condition usage, reasoning correctness, final-answer correctness, and error source. The final answer is judged by semantic equivalence to the annotated acceptable answer set, while error attribution is judged separately as condition-related, logic-related, both, or none.

For transcription outputs, the evaluators identify whether the model output is a transcription, an answer, or a mixed response, and then check which final valid conditions are recovered, which are missed, and which obsolete conditions are mentioned. They are instructed not to judge mathematical correctness or final-answer correctness. Disagreements or uncertain cases between the two evaluators are resolved through human review.

C.3 Metric Definitions

Character Error Rate. We report character error rate (CER) to measure general transcription quality. Following standard ASR evaluation practice for Chinese, both the reference question and the model transcription are normalized by removing spaces and punctuation and retaining Chinese characters, English letters, and digits. CER is computed as

$$\text{CER} = \frac{S + D + I}{N_{\text{ref}}} \times 100\%, \quad (14)$$

where S , D , and I denote the numbers of substitutions, deletions, and insertions, respectively, and N_{ref} is the number of characters in the normalized reference text.

Valid Condition Recall. CER measures overall transcription fidelity, but it does not directly indicate whether the final valid conditions required for reasoning are recovered. We therefore report Valid Condition Recall (VCR). For each instance i , let $\mathcal{V}_i$ denote the annotated set of final valid conditions. A condition $v \in \mathcal{V}_i$ is considered recovered if the transcription semantically expresses the same final condition, even if the wording is not identical. Obsolete conditions are not penalized in VCR, because the goal is to measure whether the final

usable conditions are present in the speech-side output.

For the subset $\mathcal{S}$ used in transcription diagnosis, VCR is computed as

$$\text{VCR} = \frac{\sum_{i \in \mathcal{S}} \sum_{v \in \mathcal{V}_i} \mathbb{I}[v \text{ is recovered}]}{\sum_{i \in \mathcal{S}} |\mathcal{V}_i|} \times 100\%. \quad (15)$$

In our diagnostic analysis, $\mathcal{S}$ is the set of speech-reasoning error cases unless otherwise specified. This setting tests whether end-to-end speech reasoning errors occur even when final valid conditions are recoverable from the speech input.

Final-Answer Error Rate. Let $c_i \in \{0, 1\}$ denote whether the final answer of instance i is correct. The answer is considered correct if the model’s final answer is semantically equivalent to one of the acceptable answers in the annotated answer set A_i . Equivalent numerical forms, such as integer and decimal variants, are treated as correct when they express the same value.

For a set of instances $\mathcal{S}$, the final-answer error rate is

$$\text{Err} = \frac{\sum_{i \in \mathcal{S}} \mathbb{I}[c_i = 0]}{|\mathcal{S}|} \times 100\%. \quad (16)$$

We report this metric separately for text reasoning and speech reasoning, denoted as Text Err. and Audio Err.

Corrected-Condition Adoption. Corrected-condition adoption measures whether the model uses the final valid conditions as active reasoning premises. For each instance i , let $\mathcal{V}_i$ be the set of final valid conditions and let $\hat{\mathcal{V}}_i$ be the subset judged to be actually adopted and used by the model. The instance-level adoption score is

$$a_i = \frac{|\hat{\mathcal{V}}_i|}{|\mathcal{V}_i|}. \quad (17)$$

The reported Adoption score is macro-averaged over valid difficulty cells:

$$\text{Adoption} = \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} \frac{1}{|\mathcal{S}_c|} \sum_{i \in \mathcal{S}_c} a_i \times 100\%, \quad (18)$$

where $\mathcal{C}$ is the set of valid L/D difficulty cells and $\mathcal{S}_c$ is the set of evaluation instances in cell c .

Obsolete-Condition Usage. Obsolete-condition usage measures whether the model relies on invalidated conditions during reasoning. Let $\mathcal{O}_i$ be the set of obsolete conditions and let $\hat{\mathcal{O}}_i$ be the subset of obsolete conditions judged to be used by the

model in reasoning or in the final conclusion. The instance-level obsolete-condition usage score is

$$o_i = \begin{cases} \frac{|\hat{\mathcal{O}}_i|}{|\mathcal{O}_i|}, & |\mathcal{O}_i| > 0, \\ 0, & |\mathcal{O}_i| = 0. \end{cases} \quad (19)$$

The reported OldCond score is macro-averaged over difficulty cells:

$$\text{OldCond} = \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} \frac{1}{|\mathcal{S}_c|} \sum_{i \in \mathcal{S}_c} o_i \times 100\%. \quad (20)$$

Lower OldCond indicates better suppression of invalidated conditions.

Condition-Related Error. For each incorrect response, the evaluator assigns an error source label:

$$e_i \in \{\text{none}, \text{condition_only}, \text{logic_only}, \text{both}\}.$$

A condition-related error is defined as an incorrect answer whose error source is either condition_only or both. These errors include failure to adopt corrected conditions, use of obsolete conditions, mixing old and new conditions, or deriving the final answer from an invalid premise.

Let

$$\mathcal{E}_c = \{\text{condition_only}, \text{both}\}.$$

We report two condition-related error metrics. CondErr/Err measures the proportion of answer errors that involve condition misuse:

$$\text{CondErr/Err} = \frac{\sum_i \mathbb{I}[c_i = 0 \wedge e_i \in \mathcal{E}_c]}{\max(1, \sum_i \mathbb{I}[c_i = 0])} \times 100\%. \quad (21)$$

CondErr/All measures the proportion of all examples that are both incorrect and condition-related:

$$\text{CondErr/All} = \frac{\sum_i \mathbb{I}[c_i = 0 \wedge e_i \in \mathcal{E}_c]}{N} \times 100\%, \quad (22)$$

where N is the total number of evaluated examples.

Mention-Only. Mention-Only quantifies shallow condition recognition without effective reasoning use. An instance is counted as Mention-Only if at least one final valid condition is mentioned or listed in the response but is not used in the subsequent reasoning chain. The reported rate is

$$\text{Mention-Only} = \frac{\sum_i \mathbb{I}[m_i = 1]}{N} \times 100\%, \quad (23)$$

where $m_i = 1$ indicates a mention-only failure.

Condition-Update Failure. Condition-update failure among condition-related errors, denoted CUF/CondErr, measures how many condition-related errors specifically arise from failure to update the active condition state. Let $u_i = 1$ indicate that instance i is diagnosed as a condition-update failure, including non-adoption of corrected conditions, obsolete-condition residue, or old-new condition mixing that affects the answer. Then

$$\text{CUF/CondErr} = \frac{\sum_i \mathbb{I}[c_i = 0 \wedge e_i \in \mathcal{E}_c \wedge u_i = 1]}{\max(1, \sum_i \mathbb{I}[c_i = 0 \wedge e_i \in \mathcal{E}_c])} \times 100\%. \quad (24)$$

D Additional diagnostic Results

D.1 English Parallel Evaluation

To examine whether the recovery–adoption gap is specific to Chinese, we construct an item-level English parallel version of SCUR-Test. The English set preserves the reasoning structure, numerical relationships, and correction dependencies of each Chinese instance, while adapting the wording and self-correction expressions to English. We evaluate the untrained Qwen2.5-Omni-7B using the same diagnostic protocol as in the Chinese evaluation.

As shown in Table 10, the English parallel set exhibits the same qualitative recovery–adoption pattern as Chinese. Although VCR remains high at 92.20%, corrected-condition adoption is 83.85%, and obsolete conditions are still used in 8.47% of cases. The corresponding Chinese values are 93.63%, 83.46%, and 9.74%. Thus, in both languages, recovery of final valid conditions from speech does not guarantee their use as active reasoning premises. Because the English experiment uses one model and a controlled item-level parallel set, we interpret this result as preliminary cross-lingual evidence rather than broad language generalization.

D.2 Full Three-Path Diagnostic Results

Table 11 reports the full overall results of the three-way diagnostic protocol on the test set. The table complements the main diagnostic results by jointly presenting transcription, text reasoning, and end-to-end speech reasoning metrics.

The transcription path measures whether correction-relevant information is recoverable from speech. CER_{Err} and CER_{Cond} denote the average character error rate on answer-error samples and condition-main-error samples, respectively. VCR_{Err} and VCR_{Cond} denote valid-condition recall on the corresponding subsets, and FullCond denotes the proportion of answer-error samples

Language	Text Err. ↓	Audio Err. ↓	Audio C/E ↓	Audio C/All ↓	Adopt. ↑	OldCond ↓	VCR ↑
Chinese	21.77	27.91	51.80	14.46	83.46	9.74	93.63
English	26.81	32.21	41.88	13.49	83.85	8.47	92.20

Table 10: Chinese and item-level English parallel evaluation of untrained Qwen2.5-Omni-7B. All values are percentages. C/E denotes CondErr/Err, C/All denotes CondErr/All, Adopt. denotes corrected-condition adoption, OldCond denotes obsolete-condition usage, and VCR denotes valid-condition recall on speech-reasoning error cases. The English evaluation is intended as a controlled cross-lingual check rather than evidence of broad multilingual generalization.

in which all final valid conditions are recovered. These transcription metrics are computed on error cases because they are intended to diagnose failed speech reasoning.

The text reasoning path removes the speech perception burden and evaluates whether the model can handle condition updates from text. The speech reasoning path is the main end-to-end setting, where the model must recover correction-relevant evidence from audio, update the active condition state, and reason toward the final answer. For both paths, Adopt. measures final-condition adoption, OldCond measures obsolete-condition usage, Logic@Adopt measures logical correctness after adopting the final conditions, C/All denotes condition-related errors over all samples, and Err. denotes the final-answer error rate.

D.3 Qualitative Diagnostic Example

Table 12 shows a representative Qwen2.5-Omni-7B case. For readability, the Chinese source question and model outputs are presented as faithful English renderings, while CER is computed from the original Chinese transcription and reference. The transcription is faithful, and the text-reasoning output is correct, but the speech-reasoning output still uses an obsolete condition.

D.4 No-training difficulty trend

In the no-training setting, Qwen2.5-Omni-7B exhibits two distinct failure trends. As self-correction difficulty increases, especially from immediate corrections to chained and retrospective corrections, the model shows more mention-only behavior, obsolete-condition usage, and condition-related errors. As reasoning difficulty increases from L1 to L3, the final-answer error rate rises sharply even when corrected conditions are adopted, indicating that condition tracking and downstream reasoning difficulty contribute complementary sources of failure.

E Human-AI Agreement on Error Analysis

To assess the reliability of the automatic error analysis, we compare AI-based annotations with human annotations on a matched sample of 100 examples. The two annotation sets correspond to exactly the same 100 examples with one-to-one sample matching; therefore, differences between the two sets are not caused by sample mismatch. We examine both sample-level agreement on individual diagnostic fields and metric-level consistency between aggregate statistics derived from the two annotation sources.

Importantly, this agreement analysis serves as a *standalone reliability audit* of the automatic evaluator. In the evaluation pipeline used for the reported experimental results, AI-generated diagnostic annotations are subsequently reviewed by human annotators, and the verified annotations are used in the final analysis. The audit reported here therefore evaluates the quality of the initial automatic judgments independently of this additional human quality-control step.

Table 13 uses abbreviated field names for compactness: AnsCorr denotes answer correctness; OldUse denotes whether an obsolete condition is used in reasoning; LogicOK denotes logical correctness given the conditions actually used; CondCause denotes whether condition misuse is the primary cause of the final failure; ErrType denotes error source type; and FailSrc denotes the primary failure source.

Agreement measures. For a field with N valid matched annotations, the observed agreement is

$$P_o = \frac{1}{N} \sum_{i=1}^N \mathbb{I}(a_i = h_i), \quad (25)$$

where a_i and h_i denote the AI and human labels for example i , respectively. We additionally report Cohen’s κ (Cohen, 1960), which adjusts ob-

Model	Transcription					Text Reasoning					Speech Reasoning				
	CER _{Err}	CER _{Cond}	VCR _{Err}	FullCond	VCR _{Cond}	Adopt.	OldCond	Logic@Adopt	C/All	Err.	Adopt.	OldCond	Logic@Adopt	C/All	Err.
Fun-Audio-Chat-8B	16.35	20.02	90.19	78.56	86.08	92.64	1.21	93.02	2.05	8.63	85.92	7.33	91.64	11.00	18.07
Qwen2.5-Omni-7B	17.93	13.67	93.63	85.65	86.23	88.36	3.43	83.47	6.99	21.77	83.46	9.74	83.42	14.46	27.91
MiniCPM-o-2.6	82.66	88.47	74.24	47.02	70.23	84.73	10.56	77.67	12.93	31.53	71.56	29.84	71.24	33.94	50.48
R1-AQA	21.96	21.43	83.09	67.56	83.38	64.28	37.57	43.76	43.90	69.72	35.06	68.07	39.51	74.54	85.94
Qwen2-Audio-7B-Instruct	26.41	28.48	79.17	62.43	76.84	62.56	40.64	41.89	47.43	71.97	33.44	69.27	40.79	76.63	86.87

Table 11: Full overall three-way diagnostic metrics on the test set. All values are percentages. CER_{Err} and CER_{Cond} denote the average character error rate on answer-error samples and condition-main-error samples, respectively. VCR_{Err} and VCR_{Cond} denote valid-condition recall on the corresponding subsets. FullCond denotes the proportion of answer-error samples in which all final valid conditions are recovered. Adopt. denotes final-condition adoption, OldCond denotes obsolete-condition usage, Logic@Adopt denotes logical correctness under adopted conditions, C/All denotes condition-related errors over all samples, and Err. denotes final-answer error rate. Lower is better for CER, OldCond, C/All, and Err.; higher is better for VCR, FullCond, Adopt., and Logic@Adopt.

Item	Content
Source question	A meeting lasts 50 minutes, corrected to 45 minutes. The break between two meetings is 15 minutes, corrected to 10 minutes. If Manager Wang needs to attend 3 consecutive meetings, how many minutes are needed from the start of the first meeting to the end of the last meeting, including the breaks?
Reference answer	155 minutes.
Transcription	A meeting lasts 50 minutes, corrected to 45 minutes. The break between two meetings is 15 minutes, corrected to 10 minutes. If Manager Wang needs to attend 3 consecutive meetings, how many minutes are needed from the start of the first meeting to the end of the last meeting, including the breaks? CER = 1.03%.
Text reasoning output	Known conditions: Each meeting lasts 45 minutes. The break between two meetings is 10 minutes. Manager Wang needs to attend 3 consecutive meetings. Reasoning process: First, compute the total duration of the 3 meetings: 3 meetings * 45 minutes/meeting = 135 minutes. Then compute the total break duration between the meetings: 2 breaks * 10 minutes/break = 20 minutes. Finally, add the meeting duration and the break duration: 135 minutes + 20 minutes = 155 minutes. Final answer: 155 minutes.
Speech reasoning output	Known conditions: Each meeting lasts 50 minutes. The break between two meetings is 10 minutes. Manager Wang needs to attend 3 consecutive meetings. Reasoning process: First, compute the total duration of the 3 meetings: 3 meetings * 50 minutes/meeting = 150 minutes. Then compute the break duration between the meetings. Since Manager Wang needs to attend 3 consecutive meetings, there are 2 breaks: 2 breaks * 10 minutes/break = 20 minutes. Finally, add the meeting duration and the break duration: 150 minutes + 20 minutes = 170 minutes. Answer: 170 minutes.
Error source	Condition-related error. The speech reasoning output uses the obsolete condition “each meeting lasts 50 minutes” instead of the corrected condition “each meeting lasts 45 minutes”.

Table 12: A qualitative diagnostic example for Qwen2.5-Omni-7B. The model outputs are shown for the transcription, text-reasoning, and speech-reasoning paths.

served agreement for agreement expected from the marginal label distributions:

$$\kappa = \frac{P_o - P_e}{1 - P_e}, \quad (26)$$

where

$$P_e = \sum_{c \in \mathcal{C}} p_c^{\text{AI}} p_c^{\text{Human}} \quad (27)$$

denotes the expected chance agreement over the label set $\mathcal{C}$.

Cohen’s κ can be strongly affected by class prevalence when one label dominates the sample, producing high observed agreement but a much smaller chance-corrected value. This behavior is commonly discussed as the prevalence paradox of κ (Feinstein and Cicchetti, 1990; Byrt et al., 1993). We therefore additionally report prevalence-adjusted bias-adjusted kappa (PABAK) for binary fields, following the prevalence- and

bias-adjustment framework discussed by Byrt (Byrt et al., 1993):

$$\text{PABAK} = 2P_o - 1. \quad (28)$$

PABAK is included as a complementary descriptive statistic to make the influence of severe prevalence imbalance more transparent, rather than as a replacement for Cohen’s κ . We therefore interpret raw agreement, κ , PABAK, and the underlying class distribution jointly.

Agreement is calculated only when a valid human label is available for the corresponding field. All 100 examples remain matched across the audit, but field-specific valid sample sizes can therefore be smaller than 100. In particular, two human OldUse entries are blank, leaving 98 valid annotations for this field; these samples remain available for the other diagnostic fields.

Field	Valid N	Agree.	κ	PABAK
AnsCorr	100	98.00%	0.932	0.960
OldUse	98	97.96%	-0.010	0.959
LogicOK	100	94.00%	0.716	0.880
CondCause	100	94.00%	0.370	0.880
ErrType	100	93.00%	0.776	–
FailSrc	100	92.00%	0.744	–

Table 13: Sample-level agreement between human and AI annotations. Valid N denotes the number of matched examples with a valid human annotation for the corresponding field. PABAK is additionally reported for binary fields as a complementary measure under label imbalance.

Interpretation of OldUse. OldUse exhibits 97.96% observed agreement but a Cohen’s κ of -0.010 . Among its 98 valid human annotations, only one example is labeled positive, corresponding to a positive-class prevalence of approximately 1.02%. Under this highly imbalanced distribution, the expected chance agreement is $P_e \approx 0.980$, which is very close to the observed agreement $P_o \approx 0.980$. Consequently, disagreement on only a small number of instances can substantially reduce κ , even though the overall agreement rate remains high. This pattern is consistent with the well-known prevalence sensitivity of Cohen’s κ (Feinstein and Cicchetti, 1990; Byrt et al., 1993). The corresponding PABAK value of 0.959 provides a complementary view of overall agreement under this extreme prevalence distribution.

At the same time, the high raw agreement and PABAK should not be interpreted as establishing robust positive-class detection from this audit alone, because the number of positive OldUse cases is extremely small. We therefore interpret this field cautiously and primarily use it as part of the aggregate diagnostic analysis.

OldUse should also be distinguished from corrected-condition adoption. Avoiding an obsolete condition does not necessarily mean that the corrected condition has been successfully adopted. A model may fail to use the obsolete condition while still omitting the corrected condition, merely mentioning it without using it as an active premise, or reasoning from another incorrect premise. The low prevalence of explicit OldUse is therefore compatible with the broader corrected-condition adoption failures analyzed in this work.

Interpretation of CondCause. CondCause shows an observed agreement of 94.00%, with

Cohen’s $\kappa = 0.370$ and PABAK = 0.880. The human annotations contain four positive CondCause examples out of 100, again resulting in substantial class imbalance. For this field, the expected chance agreement is $P_e \approx 0.905$, which partially accounts for the difference between raw agreement and Cohen’s κ .

CondCause is also intrinsically more difficult to annotate than simple error presence. It indicates whether condition misuse is the *primary cause* of the final failure. A response can simultaneously contain condition-use errors, logical errors, calculation mistakes, or other reasoning failures. Human and AI evaluators may therefore agree that several problems are present while differing in which error should be regarded as the primary cause. Its lower κ consequently reflects both class imbalance and the difficulty of fine-grained causal attribution in responses containing multiple co-occurring errors. The high raw agreement and supplementary PABAK nevertheless provide a complementary indication of overall agreement under this imbalanced distribution.

Agreement on the remaining fields. The principal diagnostic fields show substantially stronger chance-corrected agreement. AnsCorr achieves $\kappa = 0.932$, LogicOK achieves $\kappa = 0.716$, ErrType achieves $\kappa = 0.776$, and FailSrc achieves $\kappa = 0.744$. Carletta (Carletta, 1996) discusses values above approximately 0.67 as supporting tentative conclusions in classification-based annotation studies. More generally, however, agreement statistics should be interpreted in relation to annotation difficulty and the underlying class distribution rather than through a single universal threshold (Artstein and Poesio, 2008). We therefore report both raw and chance-corrected statistics and explicitly discuss fields affected by extreme prevalence or causal ambiguity.

Metric-level consistency. Table 14 compares aggregate error-analysis metrics independently derived from AI and human annotations. ErrRate denotes the overall error rate; ErrN denotes the number of error samples; CondErrN denotes the number of condition-related errors; CondErr/Err denotes the proportion of condition-related errors among erroneous responses; CondErr/All denotes their proportion among all responses; OldUseRate denotes obsolete-condition usage rate; LogicOKRate denotes the proportion of logically correct reasoning given the used conditions; and Cor-

Metric	AI	Human
N	100	100
ErrRate	19.00%	17.00%
ErrN	19	17
CondErrN	7	6
CondErr/Err	36.84%	35.29%
CondErr/All	7.00%	6.00%
OldUseRate [†]	1.02%	1.02%
LogicOKRate	88.00%	88.00%
CorrUseRate	87.29%	94.75%

Table 14: Metric-level comparison between AI-based and human-based error analysis. [†]OldUseRate is computed over the 98 examples with valid human OldUse annotations.

rUseRate denotes the average corrected-condition usage rate.

For OldUseRate, the comparison is computed over the 98 examples with valid human OldUse annotations rather than treating the two missing human annotations as negative cases. Other metrics are computed over their corresponding valid matched samples.

At the aggregate level, AI- and human-derived metrics show similar overall patterns. The overall error rate differs by 2.00 percentage points, while CondErr/All differs by 1.00 percentage point. On the valid OldUse subset, both annotation sources yield an aggregate OldUseRate of 1.02%, and LogicOKRate is identical at 88.00%. The largest quantitative difference occurs for CorrUseRate, for which the AI evaluator gives 87.29% compared with 94.75% from human annotation, a difference of 7.46 percentage points. This larger discrepancy is consistent with corrected-condition usage being a finer-grained judgment than final-answer correctness or coarse error categorization.

Overall, the audit indicates that the AI evaluator provides reasonably reliable initial judgments for aggregate diagnostic analysis, while agreement on rare-event and fine-grained causal fields is more sensitive to class prevalence and annotation ambiguity. We therefore distinguish this standalone reliability audit from the final evaluation procedure: automatic annotations provide the initial structured judgments, after which human review supplies an additional level of quality control, and the verified annotations are used for the reported analysis. The final conclusions are thus supported by both the automatic diagnostic framework and subsequent human verification rather than by the AI evaluator alone.

Hyperparameter	Value
Base model	Qwen2.5-Omni-7B
Trainable part	Text-side LLM
Speech encoder	Frozen
Visual branch	Frozen
LM head	Frozen
Epochs	2
Learning rate	5×10^{-7}
Scheduler	Linear
Warmup ratio	0.05
Batch size/device	4
Grad. accumulation	1
Generations G	8
Max completion length	512
Max prompt length	None
Temperature	0.9
Top- p	0.9
Repetition penalty	1.1
Reference KL β	0.05
Checkpointing	Enabled
DeepSpeed	ZeRO-3

Table 15: Main optimization and generation hyperparameters.

Reward / PACER term	Value
Judge weight w_j	1.0
Format weight w_f	0.3
Condition weight λ_c	0.35
Logic weight λ_l	0.35
Answer weight λ_a	0.30
Mask ratio	0.5
Mask strategy	Random
Speech contrast γ	0.001
Entropy coeff. η_z	0.01
Entropy coeff. $\eta_{\bar{z}}$	0.01
Judge model	Qwen3-14B
Judge max tokens	225
Judge max length	4096

Table 16: Reward and speech-evidence contrast default hyperparameters.

F PACER Training Details

F.1 Training Hyperparameters

The main optimization and generation hyperparameters are shown in Table 15. Reward and PACER-specific hyperparameters are shown in Table 16. Unless otherwise stated, the default setting is $\lambda_c = 0.35$, Mask = 0.5, and $\gamma = 0.001$.

The training data use an anchor-based JSONL format. Each example contains a speech input and structured reference annotations, including final valid conditions, obsolete conditions, reference reasoning nodes, and final answers. During training, the prompt contains the speech input and a fixed response-format instruction. The written question

text is not exposed to the policy prompt, so condition recovery and condition use must be grounded in the speech input.

For reproducibility, the supplementary materials and released code include the PACER training scripts, reward-judge prompts, response-format instructions, masking implementation, configuration files, and hyperparameter templates. These files specify the GRPO training pipeline, reward aggregation rules, speech-evidence masking procedure, model-calling settings, and evaluation commands used after training.

F.2 Reward Setting

We denote the unit-interval clipping operator as

$$\text{clip}_{[0,1]}(x) = \min(1, \max(0, x)). \quad (29)$$

The total reward follows the main-text formulation:

$$R_i = w_j R_i^{\text{judge}} + w_f R_i^{\text{fmt}}. \quad (30)$$

The judge reward combines condition, logic, and answer scores:

$$R_i^{\text{judge}} = \text{clip}_{[0,1]} \left(\lambda_c S_i^{\text{cond}} + \lambda_l S_i^{\text{logic}} + \lambda_a S_i^{\text{ans}} \right). \quad (31)$$

We use $\lambda_c = 0.35$, $\lambda_l = 0.35$, and $\lambda_a = 0.30$ in the default setting.

The online judge evaluates each rollout response through three Qwen3-14B routes. Instead of directly producing scalar scores, each route outputs a structured JSON object, which is then parsed and converted into reward components by predefined aggregation rules. The condition route extracts evidence about final valid conditions, obsolete conditions, and old-new conflicts. The logic route verifies reasoning completeness, correctness, self-consistency, and error types. The answer route checks whether the final answer is correct and follows the final corrected conditions.

The condition route uses the following output schema:

```
{
  "mentioned_corrected_conditions": [],
  "mentioned_old_conditions": [],
  "missing_corrected_conditions": [],
  "extra_condition_like_claims": [],
  "used_corrected_conditions": [],
  "used_old_conditions": [],
  "old_new_conflict_conditions": []
}
```

The logic route uses the following output schema:

```
{
  "covered_logic_steps": [],
  "used_invalid_in_logic": [],
  "error_types": [],
  "logic_verdicts": {
    "key_chain_complete": 0,
    "calculation_correct": 0,
    "state_updates_correct": 0,
    "self_consistent": 0,
    "supports_final_answer": 0
  },
  "alternate_reasoning_correct": 0
}
```

The answer route uses the following output schema:

```
{
  "student_final_claims": [],
  "matched_reference_answers": [],
  "missing_reference_answers": [],
  "extra_wrong_claims": [],
  "contains_conflicting_answers": 0,
  "follows_final_correction": 1,
  "is_correct": 1,
  "match_type": "exact"
}
```

The condition score separates mention-only evidence from effective adoption into the reasoning state. Let N_c be the number of final valid conditions. Define the mention and use ratios from the condition-route JSON as

$$m_i = \frac{|\text{mentioned_corrected_conditions}|}{\max(1, N_c)}, \quad (32)$$

$$u_i = \frac{|\text{used_corrected_conditions}|}{\max(1, N_c)}.$$

The positive adoption evidence is

$$S_{i,+}^{\text{cond}} = \alpha_m m_i + \alpha_u u_i, \quad (33)$$

where $\alpha_m = 0.30$ and $\alpha_u = 0.70$. This setting gives larger weight to actual use than to shallow mention.

Negative condition evidence is aggregated as a weighted penalty set $\mathcal{C}^-$:

$$S_i^{\text{cond}} = \text{clip}_{[0,1]} \left(S_{i,+}^{\text{cond}} - \sum_{e \in \mathcal{C}^-} \kappa_e g_i^e \right). \quad (34)$$

Here, g_i^e denotes the triggered strength of the e -th negative condition signal for response y_i . In

our instantiation, $\mathcal{C}^-$ contains four channels. The first channel corresponds to obsolete-condition usage. Its value is determined by how many obsolete conditions remain active in the response, i.e., how many items appear in `used_old_conditions`; this count is capped at 2 for stability, so repeated obsolete-condition use does not dominate the entire condition score. The second channel corresponds to old–new conflict. It is activated when the response simultaneously retains final corrected conditions and obsolete conditions in a conflicting manner, as reflected by a non-empty `old_new_conflict_conditions` field. The third channel corresponds to invalid-condition usage inside the reasoning chain. It is activated when the logic route detects that invalid or obsolete information has actually entered downstream reasoning, i.e., when `used_invalid_in_logic` is non-empty. The fourth channel corresponds to wrong-premise-driven reasoning. It is activated when the logic route explicitly identifies `wrong_premise_error`, indicating that the subsequent reasoning process is built on an incorrect or outdated premise rather than on the final valid conditions.

The corresponding penalty weights are $\kappa_{\text{old}} = 0.08$, $\kappa_{\text{conflict}} = 0.05$, $\kappa_{\text{invalid}} = 0.08$, and $\kappa_{\text{wrong}} = 0.05$. As a result, the obsolete-condition penalty is upper-bounded by 0.16.

The logic score evaluates the reasoning chain under the adopted conditions. The logic route returns five binary verdicts $v_i^k \in \{0, 1\}$, corresponding to key-chain completeness, calculation correctness, state-update correctness, self-consistency, and whether the reasoning supports the final answer. Concretely, key-chain completeness checks whether the response contains the essential intermediate reasoning steps instead of jumping directly to the answer; calculation correctness checks whether numerical or symbolic computations are correct; state-update correctness checks whether intermediate state changes are tracked correctly in multi-step problems; self-consistency checks whether the reasoning remains internally non-contradictory; and support for the final answer checks whether the derived reasoning is sufficient to justify the stated final answer. Denoting this verdict set by $\mathcal{V}^{\text{logic}}$, we use a start-from-one deduction form:

$$S_{i,0}^{\text{logic}} = 1 - \sum_{k \in \mathcal{V}^{\text{logic}}} \mu_k (1 - v_i^k). \quad (35)$$

The verdict deduction weights are

$$\begin{aligned} \mu_{\text{chain}} &= 0.15, & \mu_{\text{calc}} &= 0.22, & \mu_{\text{state}} &= 0.12, \\ \mu_{\text{self}} &= 0.12, & \mu_{\text{sup}} &= 0.20. \end{aligned} \quad (36)$$

In addition, the judge outputs a set of triggered reasoning error types $\mathcal{E}_i^{\text{logic}}$. Let $h_i^e = \mathbb{I}[e \in \mathcal{E}_i^{\text{logic}}]$. We apply an extra deduction

$$P_i^{\text{logic}} = \sum_{e \in \mathcal{E}^{\text{logic}}} \nu_e h_i^e, \quad (37)$$

where the instantiated weights are

$$\begin{aligned} \nu_{\text{missing}} &= 0.08, & \nu_{\text{contra}} &= 0.10, \\ \nu_{\text{multi}} &= 0.10, & \nu_{\text{unsup}} &= 0.12, \end{aligned} \quad (38)$$

$$\begin{aligned} \nu_{\text{unit}} &= 0.08, & \nu_{\text{state}} &= 0.08, \\ \nu_{\text{formula}} &= 0.15, & \nu_{\text{arith}} &= 0.08. \end{aligned} \quad (39)$$

These correspond to missing-key-step, contradiction, multi-branch, unsupported-answer, unit-conversion, state-update, formula, and arithmetic errors, respectively.

The final logic score is

$$S_i^{\text{logic}} = \text{clip}_{[0,1]} \left(S_{i,0}^{\text{logic}} - P_i^{\text{logic}} \right). \quad (40)$$

If the judge marks the reasoning as an alternative but valid solution path, we preserve partial credit by

$$S_i^{\text{logic}} \leftarrow \max(S_i^{\text{logic}}, 0.85). \quad (41)$$

The answer score is a strict binary judgment. Only exact or equivalent matches are treated as correct. Define

$$c_i^{\text{ans}} = \mathbb{I}[\text{match_type} \in \{\text{exact}, \text{equivalent}\}]. \quad (42)$$

Let

$$g_i^{\text{miss}} = \mathbb{I}[|\text{missing_reference_answers}| > 0], \quad (43)$$

$$g_i^{\text{extra}} = \mathbb{I}[|\text{extra_wrong_claims}| > 0], \quad (44)$$

$$g_i^{\text{conf}} = \mathbb{I}[\text{contains_conflicting_answers} = 1], \quad (45)$$

$$g_i^{\text{corr}} = \mathbb{I}[\text{follows_final_correction} = 0]. \quad (46)$$

The final answer score is

$$\begin{aligned} S_i^{\text{ans}} &= \mathbb{I} \left[c_i^{\text{ans}} = 1 \wedge g_i^{\text{miss}} = 0 \right. \\ &\quad \wedge g_i^{\text{extra}} = 0 \wedge g_i^{\text{conf}} = 0 \\ &\quad \left. \wedge g_i^{\text{corr}} = 0 \right]. \end{aligned} \quad (47)$$

The format reward encourages a parseable condition–reasoning–answer structure but does not evaluate mathematical correctness:

$$\begin{aligned} R_i^{\text{fmt}} &= \min \left(0.20 q_i^{\text{cond}} + 0.20 q_i^{\text{logic}} \right. \\ &\quad \left. + 0.30 q_i^{\text{ans}} + 0.30 q_i^{\text{num}}, 1 \right). \end{aligned} \quad (48)$$

Here q_i^{cond} , q_i^{logic} , and q_i^{ans} indicate whether the response contains recognizable condition, reasoning, and answer fields, while q_i^{num} indicates whether the answer field contains an explicit answer value.

F.3 Speech-Evidence Masking

PACER applies masking at the speech-representation level. Given the intact speech representation z , a random temporal mask produces $\tilde{z} = M(z)$. The default mask ratio is 0.5. The mask is applied over valid speech frames and does not target padding frames.

F.4 Evaluation Settings After Training

After training, PACER is evaluated on held-out self-corrected speech reasoning sets. Model responses are generated from speech inputs and then evaluated with a structured external evaluator using the same anchor annotations: final valid conditions, obsolete conditions, reasoning nodes, and final answers.

The evaluation reports condition-level, reasoning-level, and answer-level metrics. Condition-level metrics measure whether final corrected conditions are adopted and whether obsolete conditions remain active. Reasoning-level metrics measure whether corrected conditions enter the reasoning chain and whether old conditions still affect reasoning. Answer-level metrics measure strict final-answer correctness under the final corrected conditions.

G Additional Difficulty-Level Evaluation Results

This appendix provides additional difficulty-level evaluation results for PACER. The main paper already reports training-objective ablations, reward-component ablations, speech-mask effects, error-source breakdowns, and parameter sensitivity analyses. Accordingly, this appendix focuses on complementary analyses by intrinsic reasoning difficulty and self-correction difficulty, together with representative qualitative cases that illustrate condition-state recovery.

Unless otherwise specified, all results are reported on the full test split of 2,490 examples, and all metric values are percentages.

Model naming. For readability, we use short method names throughout this appendix. **NoTrain** denotes the untrained Qwen2.5-Omni-7B baseline. **GRPO-Answer** uses only final-answer correctness as the reward signal. **GRPO-Structured** uses answer-level reward and structured condition-state reward, but does not use the speech-evidence contrast mechanism. **PACER** denotes the full method.

Metrics. **Acc.** denotes final-answer accuracy. **Adopt.** denotes corrected-condition adoption. **Old-InChain** denotes the rate at which obsolete conditions enter the reasoning chain. **CondErr/All** denotes condition-related errors among all examples. **Logic@Adopt** measures whether the reasoning remains logically correct after adopting corrected conditions.

G.1 Breakdown by Reasoning Difficulty

Table 17 reports results by intrinsic reasoning difficulty. L1 denotes single-step calculation or direct inference, L2 denotes two- to four-step reasoning or simple multi-variable judgment, and L3 denotes longer reasoning chains, multi-variable composition, or multi-branch reasoning. PACER improves performance across all reasoning levels, with especially clear gains on L2 and L3. L3 remains the hardest subset, suggesting that long-chain reasoning continues to be challenging even when condition-state maintenance is improved.

G.2 Breakdown by Self-Correction Difficulty

Table 18 reports results by self-correction difficulty. D0 is a no-correction control setting. D1 and D2 denote single-slot and multi-slot immediate correction, respectively. D3 and D4 denote single-slot and multi-slot chained correction. D5 denotes long-distance retrospective correction. PACER mainly improves correction-bearing examples, especially D2–D5, where the model must revise the active condition state rather than answer from the initially perceived information.

G.3 Qualitative Case Studies of Condition-State Recovery

To make the recovery behavior concrete, we present two representative D2 examples in full. The original benchmark metadata and model outputs are in Chinese. For readability in this appendix, we present faithful English renderings of the original question and model responses while preserving the underlying condition updates, reasoning steps, and final predictions. In both examples, all non-PACER variants fail, whereas PACER correctly updates and propagates the corrected condition state.

These two cases make the behavioral difference explicit. In both examples, the main advantage of PACER is not merely answer-level improvement, but reliable recovery of the corrected condition state under multi-slot correction, followed by consistent downstream reasoning.

Level	Setting	Acc. $\uparrow$	Adopt. $\uparrow$	OldInChain $\downarrow$	CondErr/All $\downarrow$	Logic@Adopt $\uparrow$
L1	NoTrain	90.54	86.51	8.04	8.20	98.16
L1	GRPO-Answer	98.23	92.01	1.26	1.26	99.41
L1	GRPO-Structured	98.99	93.09	0.60	0.88	99.75
L1	PACER	98.74	93.04	0.42	0.88	99.35
L2	NoTrain	76.39	85.21	11.12	14.34	88.47
L2	GRPO-Answer	86.99	91.07	1.51	4.34	90.26
L2	GRPO-Structured	89.76	92.04	0.47	3.73	92.16
L2	PACER	90.48	93.28	0.36	3.01	93.81
L3	NoTrain	51.10	79.18	9.79	20.30	66.08
L3	GRPO-Answer	59.17	84.07	0.94	10.61	66.88
L3	GRPO-Structured	58.25	84.62	0.95	10.61	65.76
L3	PACER	62.05	86.00	0.94	9.80	70.33

Table 17: Breakdown by reasoning difficulty.

D-level	Setting	Acc. $\uparrow$	Adopt. $\uparrow$	OldInChain $\downarrow$	CondErr/All $\downarrow$	Logic@Adopt $\uparrow$
D0	NoTrain	85.19	92.12	0.00	3.84	88.22
D0	GRPO-Answer	86.29	92.64	0.00	3.66	89.01
D0	GRPO-Structured	86.11	92.93	0.20	3.47	87.94
D0	PACER	87.75	91.74	0.00	3.47	89.74
D1	NoTrain	76.00	84.26	7.91	12.44	86.69
D1	GRPO-Answer	84.44	86.21	1.58	4.22	86.94
D1	GRPO-Structured	84.89	89.66	0.19	3.11	86.43
D1	PACER	85.56	91.54	0.41	4.00	87.15
D2	NoTrain	68.56	83.56	13.19	16.55	83.63
D2	GRPO-Answer	80.85	92.09	2.92	5.20	85.72
D2	GRPO-Structured	81.56	91.88	0.97	4.02	84.80
D2	PACER	85.11	93.15	0.24	2.60	88.43
D3	NoTrain	72.12	79.68	12.18	16.37	85.22
D3	GRPO-Answer	83.63	86.74	0.90	4.65	86.54
D3	GRPO-Structured	84.73	87.40	1.13	4.87	88.14
D3	PACER	85.84	90.09	0.90	3.54	89.07
D4	NoTrain	62.36	77.14	16.07	25.56	80.85
D4	GRPO-Answer	76.40	87.64	1.15	8.99	82.76
D4	GRPO-Structured	78.37	87.86	0.90	8.99	84.20
D4	PACER	79.21	87.95	0.85	8.15	87.18
D5	NoTrain	56.87	84.30	8.81	18.32	72.14
D5	GRPO-Answer	64.89	87.45	0.66	9.16	73.50
D5	GRPO-Structured	66.79	88.10	0.66	9.92	75.87
D5	PACER	67.56	88.74	1.33	9.16	78.42

Table 18: Breakdown by self-correction difficulty.

Setting	Model response (English rendering)	Error diagnosis
Case 1 (multi-slot immediate correction). Question: A stationery store has three piles of notebooks. The first pile has 32 notebooks, or rather 35. The second pile has 38 notebooks. The third pile has 20 notebooks, sorry, it should be 25. The owner sells 7 notebooks from each pile. How many notebooks remain in the store? Gold answer: 77 notebooks.		
NoTrain	<i>Known conditions:</i> The first pile has 32 notebooks, the second pile has 38 notebooks, and the third pile has 20 notebooks, or rather 25. The owner sells 7 notebooks from each pile. <i>Reasoning:</i> After selling 7 from each pile, the first pile has $32 - 7 = 25$ notebooks left, the second pile has $38 - 7 = 31$ left, and the third pile has $20 - 7 = 13$ left. Adding them together gives $25 + 31 + 13 = 69$. <i>Answer:</i> 69 notebooks remain in the store.	Retains obsolete conditions (32 and 20) and never fully updates to the corrected state. Condition-centric failure. Final prediction: 69.
GRPO-Answer	<i>Known conditions:</i> The first pile has 32 notebooks, the second pile is 35, the third pile has 38 notebooks, the third pile has 25 notebooks, and the owner sells 7 notebooks from each pile. <i>Reasoning:</i> After selling 7 from each pile, the first pile has $32 - 7 = 25$ left, the second pile has $35 - 7 = 28$ left, and the third pile has $25 - 7 = 18$ left. Adding them gives $25 + 28 + 18 = 71$. <i>Answer:</i> 71 notebooks remain in the store.	Still preserves the obsolete first-pile value 32 and also shows slot confusion across piles. Condition-centric failure. Final prediction: 71.
GRPO-Structured	<i>Known conditions:</i> The first pile has 32 notebooks, the second pile has 38 notebooks, the third pile has 25 notebooks, and the owner sells 7 notebooks from each pile. <i>Reasoning:</i> After selling 7 from each pile, the first pile has $32 - 7 = 25$ left, the second pile has $38 - 7 = 31$ left, and the third pile has $25 - 7 = 18$ left. Adding them gives $25 + 31 + 18 = 74$. <i>Answer:</i> 74 notebooks remain in the store.	Still carries the obsolete first-pile value 32 into the reasoning chain. Condition-centric failure. Final prediction: 74.
PACER	<i>Known conditions:</i> The first pile has 35 notebooks, the second pile has 38 notebooks, and the third pile has 25 notebooks. The owner sells 7 notebooks from each pile. <i>Reasoning:</i> After selling 7 from each pile, the first pile has $35 - 7 = 28$ notebooks left, the second pile has $38 - 7 = 31$ left, and the third pile has $25 - 7 = 18$ left. Adding them gives $28 + 31 + 18 = 77$. <i>Answer:</i> 77 notebooks remain in the store.	Correctly updates all corrected quantities and propagates them through the full reasoning chain. Correct final prediction: 77.

Table 19: Full qualitative comparison for a representative D2 example. The original instance and outputs are in Chinese; the table shows faithful English renderings for readability. All non-PACER variants fail to maintain the corrected condition state, whereas PACER updates the active condition set consistently.

Setting	Model response (English rendering)	Error diagnosis
Case 2 (multi-slot immediate correction). Question: In an office printer network, the main queue currently has 4 connected computers, and the backup queue has 1. You then add 2 computers to the backup queue, sorry, it should be 3. Next, 1 computer from the backup queue is transferred to the main queue, sorry, it should be 2. How many connected computers are in the main queue and the backup queue at the end? Gold answer: main queue = 6, backup queue = 2.		
NoTrain	<i>Known conditions:</i> The main queue currently has 4 connected computers, and the backup queue has 1. <i>Reasoning:</i> 1. Add 3 computers to the backup queue, so the backup queue becomes $1 + 3 = 4$. 2. Transfer 1 computer from the backup queue to the main queue, so the main queue becomes $4 + 1 = 5$. 3. Transfer 2 more computers to the main queue, so the main queue becomes $5 + 2 = 7$. 4. The backup queue finally becomes $4 - 2 = 2$. <i>Answer:</i> The main queue has 7 connected computers, and the backup queue has 2.	Mixes obsolete and corrected transfer counts by applying both 1 and 2. Primary failure is condition-state inconsistency. Final prediction: main 7, backup 2.
GRPO-Answer	<i>Known conditions:</i> The main queue currently has 4 connected computers, and the backup queue has 1. <i>Reasoning:</i> 1. Add 3 computers to the backup queue, so it becomes $1 + 3 = 4$. 2. Transfer 1 computer from the backup queue to the main queue, so the main queue becomes $4 + 1 = 5$. 3. Then transfer 2 more computers from the backup queue to the main queue, so the main queue becomes $5 + 2 = 7$. 4. The backup queue then becomes $4 - 2 = 2$. <i>Answer:</i> The main queue has 7 connected computers, and the backup queue has 2.	Again mixes obsolete and corrected transfer counts. Failure is still driven by incorrect condition-state maintenance. Final prediction: main 7, backup 2.
GRPO-Structured	<i>Known conditions:</i> The main queue currently has 4 connected computers, the backup queue has 1, 3 computers are added to the backup queue, and 2 computers are transferred from the main queue to the backup queue. <i>Reasoning:</i> 1. Initially, the main queue has 4 computers and the backup queue has 1. 2. After adding 3 computers to the backup queue, it becomes $1 + 3 = 4$. 3. After transferring 2 computers from the main queue to the backup queue, the main queue becomes $4 - 2 = 2$, and the backup queue becomes $4 + 2 = 6$. <i>Answer:</i> The main queue has 2 connected computers, and the backup queue has 6.	Uses the updated number 2 but reverses the transfer direction. The corrected event is not preserved consistently in the reasoning chain. Final prediction: main 2, backup 6.
PACER	<i>Known conditions:</i> The main queue currently has 4 connected computers, and the backup queue has 1. Then 3 computers are added to the backup queue, and 2 computers are transferred from the backup queue to the main queue. <i>Reasoning:</i> 1. Initially, the main queue has 4 computers and the backup queue has 1. 2. Add 3 computers to the backup queue, so it becomes $1 + 3 = 4$. 3. Transfer 2 computers to the main queue, so the main queue becomes $4 + 2 = 6$, and the backup queue becomes $4 - 2 = 2$. <i>Answer:</i> The main queue has 6 connected computers, and the backup queue has 2.	Correctly updates both corrected operations and preserves the transfer direction. Correct final prediction: main 6, backup 2.

Table 20: Full qualitative comparison for another representative D2 example. The original instance and outputs are in Chinese; the table shows faithful English renderings for readability. PACER is the only variant that correctly updates and propagates the corrected transfer event.