# REINFORCEMENT LEARNING VIA STRUCTURED DELIBERATION AND FINE-GRAINED ATTRIBUTION

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Reinforcement learning with verifiable rewards (RLVR) improves the reasoning capabilities of large language models by optimizing policies against a single reward signal derived from the final outcome. However, real-world reasoning tasks often involve multiple valid answers or more open-ended objectives that a single outcome signal cannot adequately capture or distinguish. In such settings, outcome-only RLVR assigns a homogeneous advantage to the entire reasoning trajectory, ignoring both the complex reasoning structure and the heterogeneous contributions of individual components. We introduce **s**tructured **o**ptimization via **d**eliberation and **a**ttribution (SODA), a framework that addresses these limitations through structured deliberation and fine-grained attribution. SODA first makes the structure of complex reasoning explicit by decomposing intermediate reasoning into semantically meaningful, locally evaluable decision units, and then attributes credit to each individual unit based on its contribution in the context of the remaining response. These credits are computed via local edits, yielding differentiated training signals without additional policy rollouts or a learned critic. We instantiate SODA on set-valued QA, where each question may have a different number of semantically distinct answers. Reasoning is structured into complementary stages of candidate proposal and selective refinement to produce correct and non-redundant answers. Experiments across multiple datasets and base models show consistent improvements over state-of-the-arts, with ablations demonstrating complementary gains from structured deliberation and fine-grained attribution.

## 1 INTRODUCTION

Reinforcement learning with verifiable rewards (RLVR) has achieved substantial success in training large language models (LLMs) for reasoning tasks, particularly those with a single verifiable answer (see e.g., Shao et al., 2024). However, many real-world questions seek multiple semantically distinct answers, with the number of valid answers varying across inputs. For example, identifying plausible diagnoses from a patient's symptoms requires considering several possible conditions rather than returning a single diagnosis (Figure 1). This setting, known as *set-valued question answering (QA)*, illustrates a broader class of reasoning problems that require models to coordinate multiple plausible outputs and decide how many to retain in the final response. Prompting alone does not reliably elicit such adaptive output cardinality, as evidenced by the weak association between predicted and ground-truth answer-set sizes in our motivating experiment (Section 2). Recent work also highlights difficulties in candidate generation and selection (Puri et al., 2026; Hu et al., 2026), motivating methods that better structure and coordinate these interdependent stages of reasoning.

These components of reasoning can be viewed as a sequence of *intermediate decisions*. Although RLVR can directly reward the quality of a generated answer set, standard outcome-based supervision such as Group Relative Policy Optimization (GRPO; Shao et al., 2024) provides limited guidance for the individual decisions. A shared response-level advantage is applied homogeneously across the entire trajectory, despite the heterogeneous contributions of individual decisions: some improve the final output, while others introduce errors, redundancy or omissions. As a result, useful information in the intermediate reasoning can remain underutilized: a correct but discarded proposal, for example, receives no distinct credit for having been generated. This raises a natural question:

*Can we exploit task structure to directly evaluate individual decision contributions and derive fine-grained training signals?*



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

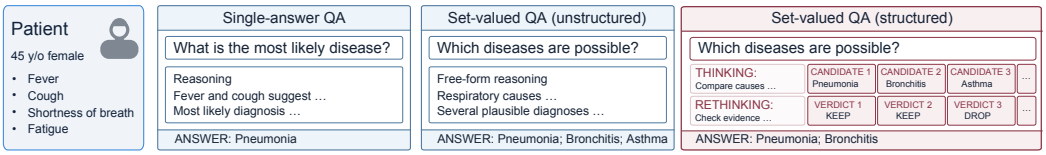


Figure 1: Response structures for single-answer and set-valued question answering (QA).

Certain generation tasks, such as set-valued QA, support direct evaluation of individual decisions through three structural ingredients: (1) *identifiable decision units*, which together form the structured components of reasoning and can be isolated as semantically meaningful, clearly delineated spans in the generated text; (2) *admissible local edits*, which modify a decision while preserving a valid scoring object without regeneration; and (3) *task-defined utilities*, which score the original and edited objects at training time to quantify the decision's contribution. In set-valued QA, these ingredients correspond to candidate proposals and keep/drop decisions, candidate removal and selection flips, and metrics such as recall and $F_1$, respectively. Similar structure arises in other tasks, such as software test-suite generation (Alshahwan et al., 2024) and document reranking (Sun et al., 2023).

Making this structure explicit in intermediate reasoning provides a natural basis for attributing credit to individual decisions in context. Decisions that are otherwise *implicit* in free-form generation can be articulated as *explicit* intermediate steps that can be inspected by the model itself or external verifiers. Figure 1 illustrates this distinction: *structured* set-valued QA exposes candidate answers and keep/drop decisions as explicit units for local evaluation and credit assignment. While prior work on structured reasoning, such as subproblem decomposition (Zhou et al., 2023), tree-based search (Yao et al., 2023), and iterative reconsideration (Madaan et al., 2023), has established the value of structured deliberation, we need a principled way to evaluate each decision's contribution to turn this structure into decision-specific training signals.

Building on this principle, we introduce **Structured Optimization via Deliberation and Attribution (SODA)**. SODA organizes intermediate reasoning into two stages, THINKING and RETHINKING: THINKING develops explicit intermediate content, while RETHINKING revisits it through context-dependent decisions. This organization exposes semantically meaningful decision units as identifiable targets for evaluation and credit assignment. For each unit, SODA compares a task-defined utility before and after an admissible local edit, while holding the remaining structure fixed. The resulting utility difference measures the unit's contribution in context and provides a distinct training signal to its corresponding token span. The remaining free-form reasoning spans receive group-relative credits by comparing task-defined utilities across responses to the same input. Credit computation requires only utility rescoring, without additional policy rollouts or a learned critic.

We instantiate SODA in set-valued QA, where THINKING proposes candidate answers and RETHINKING assigns keep/drop verdicts. Each candidate receives credit based on how much its removal reduces the candidate pool's coverage of the ground-truth answers, evaluating its contribution to the available answer hypotheses. Each keep/drop decision receives credit based on the change in final-set $F_1$ when that decision is reversed, evaluating whether it improves the selection. Jointly, these signals connect and balance complementary exploration in THINKING with selective refinement in RETHINKING to produce correct and non-redundant answers.

Our **contributions** are summarized as follows:

- *Conceptually*, we frame reasoning structure as a basis for credit attribution. We characterize the conditions under which individual reasoning decisions can be modified in isolation, with their contributions quantified by the resulting changes in task performance. This first bridges the design of intermediate reasoning structure with the construction of fine-grained training signals.
- *Methodologically*, we introduce SODA, which makes reasoning decisions explicit through two-stage deliberation and translates their local utility contributions into decision-specific training signals without additional policy rollouts or a learned critic. For set-valued QA, we establish theoretical properties showing how these credits align with attainable answer-set quality.
- *Empirically*, experiments across multiple set-valued QA datasets and base models show improved answer-set quality, with ablations supporting the contributions of structured deliberation and fine-grained credit assignment at each stage.

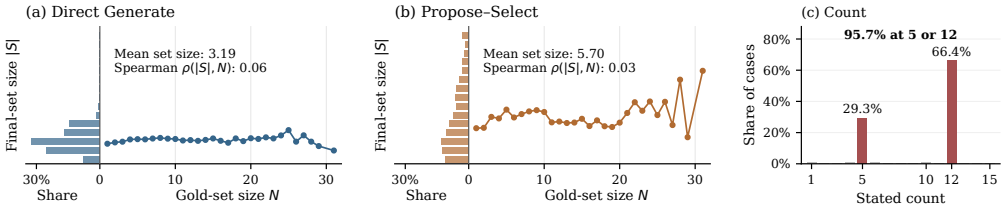


Figure 2: Prompting Qwen3-8B on DDXPlus without task-specific training. **(a,b)** Final-set sizes under *Direct Generate* and *Propose–Select* prompts. **(c)** Answer counts predicted by the LLM. Without targeted training, LLMs show limited adaptation of answer-set size to individual questions.

### 1.1 Related Work

**Reinforcement Learning with Verifiable Rewards (RLVR).** Group Relative Policy Optimization (GRPO) (Shao et al., 2024) computes group-relative advantages from outcome rewards and broadcasts each response's advantage to all tokens. Subsequent work identifies optimization issues in this formulation: Liu et al. (2025) remove normalization terms to mitigate length and difficulty biases; Yu et al. (2025) improve training stability through revised clipping, sampling, and loss aggregation. These refinements improve optimization but still broadcast a single response-level advantage to all tokens, without explicitly distinguishing their contributions within the response.

**Fine-grained and Counterfactual Credit Assignment.** Fine-grained signals can be derived from process reward models (Wang et al., 2024), generative critiques (Xie et al., 2025), reward aggregation (Cheng et al., 2025), and model-internal signals (Sheng et al., 2026; Shan et al., 2026; He et al., 2026; Jin et al., 2026). Other approaches derive step or process signals through Monte Carlo continuations (Kazemnejad et al., 2025; Guo et al., 2025), grouping trajectories by shared states or tree structure (Feng et al., 2025; Ji et al., 2026), tracking reference-answer likelihood improvements (Wang et al., 2026b;a), or contrasting token predictions under different prompts (Li et al., 2026a). An important approach is counterfactual attribution, which evaluates contributions by comparing original and modified outcomes. Related methods use token perturbations (Li et al., 2026b), resampled continuations (Samanta et al., 2026), or reasoning-path subset evaluation (Zhang et al., 2026b). SODA instead assigns decision-level credit by directly rescoring semantically meaningful local edits to a structured object, without downstream regeneration or a learned critic.

**Structured Reasoning.** Prior work uses structure to guide multi-step reasoning and the exploration of solutions through explicit steps (Wei et al., 2022), subproblem decomposition (Zhou et al., 2023), tree-based exploration (Yao et al., 2023), and graph-based aggregation and refinement (Besta et al., 2024). Other approaches structure the revision of intermediate content to identify errors and improve initial solutions, using prompted feedback (Madaan et al., 2023), reinforcement learning for self-correction (Kumar et al., 2025), reflection and exploration (Shen et al., 2025), and self-verification (Ma et al., 2025). More closely related, Jiang et al. (2026) use an external LLM to transform reference reasoning chains into verifiable subproblems and assign advantages to answer spans accordingly. SODA instead designs intermediate reasoning structure explicitly for local attribution, organizing decisions so that admissible edits expose their utility contributions in context.

**Set-valued Question Answering.** Set-valued QA requires generating a set of semantically distinct answers whose cardinality varies across questions. Training-time methods optimize $F_1$ (Zhang et al., 2026a), elicit answer sets of a fixed size using recall-based rewards (Puri et al., 2026), or assign credit to retrieval steps (Shi et al., 2026). Inference-time methods use conformal calibration to construct answer sets with controlled miscoverage risk (Hu et al., 2026). We introduce structured intermediate reasoning into set-valued QA to support fine-grained credit assignment.

## 2 Motivating Study: Intermediate Structure under Prompting

In this section, we use zero-shot prompting to examine the challenges of set-valued generation. We examine how explicit candidate proposal and selection affect model behavior and what limitations remain. We evaluate Qwen3-8B on DDXPlus (Tchango et al., 2022) under three prompting settings: (1) *Direct Generate*, which uses unrestricted free-form reasoning to generate a final answer set,

choosing both its contents and size; (2) *Propose–Select*, which first proposes 15 candidate answers, then assigns each a keep/drop verdict to form the final answer set; and (3) *Count*, which predicts only the number of valid answers without generating them. Prompts are provided in Appendix C.3.

**Observation 1: prompting yields limited size adaptation.** Answer-set sizes from *Direct Generate* correlate weakly with ground-truth sizes (Figure 2a), while explicit count predictions concentrate on a few values (Figure 2c). These patterns suggest that prompting alone struggles to adjust the number of answers adaptively to each question.

**Observation 2: explicit structure helps, but leaves a gap.** *Propose–Select* improves overall performance (Appendix E.1) and produces larger and more varied final answer-set sizes than *Direct Generate*, but its final-set size remains weakly associated with gold-set size (Figure 2b). Although the structure makes the candidate proposal and selection explicit, prompting alone does not reliably guide which answers to retain or how many to include for each question.

**Implication: from structured deliberation to targeted learning.** These observations suggest that making proposal and selection explicit at inference time is not sufficient for reliable adaptation. However, this structure reveals distinct sources of error: a valid answer may be missing because it was never proposed or was later discarded. This motivates evaluating individual decisions through local edits to derive separate training signals for different components of reasoning.

# 3 METHODOLOGY

**S**tructured **o**ptimization via **d**eliberation and **a**ttribution (SODA) organizes intermediate reasoning into explicit decision units whose local utility contributions can be directly evaluated and used as span-specific training signals. In this section, we introduce SODA in a general setting, and instantiate the framework for set-valued QA in Section 4.

## 3.1 STRUCTURED DELIBERATION AND APPLICABILITY

To make individual decision contributions directly evaluable, we consider a class of generation tasks whose structure supports verifiable local edits. Specifically, we require: (1) *identifiable decision units*, which together form the structured components of reasoning and can be isolated as semantically meaningful, clearly delineated spans in the generated text; (2) *admissible local edits*, which modify a decision while preserving a valid scoring object; and (3) *task-defined utilities*, which score the original and edited response at training time to quantify the decision's contribution.

To expose task structure for credit assignment, SODA uses a structured prompt to organize intermediate reasoning into THINKING and RETHINKING. Given an input $q$, a policy $\pi_\theta(\cdot \mid q)$ generates

$$y = \big(h^{\text{think}}, u_1, \ldots, u_M, h^{\text{rethink}}, v_1, \ldots, v_M, h^{\text{outcome}}\big),$$

where $M$ is the prescribed number of decision units in each stage. In THINKING, free-form reasoning $h^{\text{think}}$ precedes a collection of *thinking units*, $\mathcal{U} = (u_1, \ldots, u_M)$, each representing explicit intermediate content, such as a candidate answer. In RETHINKING, free-form reasoning $h^{\text{rethink}}$ precedes *rethinking units*, $\mathcal{V} = (v_1, \ldots, v_M)$, where each $v_j$ is a verdict specifying how $u_j$ is used. The outcome span $h^{\text{outcome}}$ presents the final answer.

This structure supports credit assignment to each component of reasoning separately.

## 3.2 LOCAL UTILITY-BASED CREDITS FOR STRUCTURED DECISION UNITS

Structured decision units make explicit two types of reasoning: intermediate content ($u_j$) and its use ($v_j$). We assign credit to them through the utility difference induced by an admissible local edit, holding the remaining units fixed. This difference measures the decision's local contribution in context and provides a training signal for its corresponding token span. Let $U_{\text{think}}(\mathcal{U})$ and $U_{\text{rethink}}(\mathcal{U}, \mathcal{V})$ be task-defined utility functions evaluating the intermediate collection and the decisions about its use, respectively. These utilities may rely on reference information or a verifier available during training.

**Thinking-unit credit.** Let $\mathcal{U}_{-j}$ denote the collection after removing $u_j$, with other units unchanged. The credit measures the utility lost upon removal. A positive value rewards content that contributes beyond what the remaining collection provides, accounting for redundancy and complementarity:

$$\Delta_j^{\text{think}} = U_{\text{think}}(\mathcal{U}) - U_{\text{think}}(\mathcal{U}_{-j}). \tag{1}$$

**Rethinking-unit credit.** For each observed decision $v_j$, the task specifies an admissible alternative $\widetilde{v}_j$ representing a different choice about how to use $u_j$. For example, a keep decision may be replaced with a drop decision, or vice versa. Holding $\mathcal{U}$ and all other rethinking units fixed, define $\mathcal{V}^{(j)} = (v_1, \ldots, \widetilde{v}_j, \ldots, v_M)$. Positive credit favors the observed decision over the specified alternative; negative credit identifies an improving replacement. The corresponding credit is

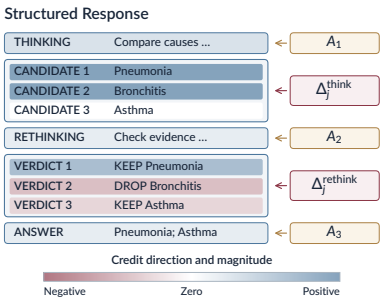


Figure 3: Credit assignment in SODA. The reference answers are Pneumonia and Bronchitis.

$$\Delta_j^{\text{rethink}} = U_{\text{rethink}}(\mathcal{U}, \mathcal{V}) - U_{\text{rethink}}(\mathcal{U}, \mathcal{V}^{(j)}). \tag{2}$$

Both credits are computed by directly rescoring structured objects, without regenerating downstream text, additional policy rollouts, or a learned critic. They are applied to the corresponding unit spans in the original response without group centering.

### 3.3 Group-Relative Credits for Free-Form Spans

The remaining free-form spans receive group-relative credits. For each input $q$, we sample $G$ responses from the current policy. For response $g$, task-defined utility scores $r_{1,g}$, $r_{2,g}$, and $r_{3,g}$ measure the overall quality of the intermediate collection, the reconsidered result, and the final output, respectively. Unlike the local credits $\Delta$, these scores evaluate the corresponding outputs as a whole. The utility functions used to compute these overall scores may be the same as those used to compute local credits, but the framework does not require this (see Section 4.2 for an example).

Following Liu et al. (2025), we mean-center these scores within each group to derive the group-relative credits, $A_{1,g}$, $A_{2,g}$, and $A_{3,g}$, and assign them to $h^{\text{think}}$, $h^{\text{rethink}}$, and $h^{\text{outcome}}$, respectively:

$$A_{k,g} = r_{k,g} - 1/G \sum_{g'=1}^{G} r_{k,g'}, \qquad k \in \{1, 2, 3\}. \tag{3}$$

### 3.4 Policy Optimization

For response $g$, each token receives an advantage according to its enclosing span: free-form spans receive the corresponding group-relative credit $A_{k,g}$, while decision-unit spans receive $\Delta_{g,j}^{\text{think}}$ or $\Delta_{g,j}^{\text{rethink}}$. Each credit applies to the tokens within its span. Figure 3 illustrates this assignment in set-valued QA. Following the formulation of GRPO (Shao et al., 2024), these token-level advantages weight an importance-sampling policy objective; the loss formulation and implementation details are provided in Appendix A.1.

## 4 Instantiation and Theoretical Analysis for Set-Valued QA

In this section, we instantiate SODA in set-valued QA, where THINKING proposes candidate answers and RETHINKING determines which candidates to retain. We define the task-specific utilities and local edits, then theoretically show how the resulting credits relate to attainable answer-set quality.

### 4.1 Task and Response Structure

For a question $q$, let $\mathcal{Y}_q^*$ denote the reference answer set and $N = |\mathcal{Y}_q^*| \geq 1$ its cardinality. The model proposes candidates $\mathcal{U} = (u_1, \ldots, u_M)$ and assigns each candidate a keep/drop decision $v_j$. The retained candidates form a subcollection $S$. The prompt fixes the candidate budget at $M$, while the model determines the output size $|S|$.

We treat CANDIDATE and VERDICT lines in the response format (Appendix C.1) as decision-unit spans, and the remaining text as the corresponding free-form spans. We use maximum one-to-one matching between candidate entries and reference answers: each candidate and each reference answer can participate in at most one matched pair. Matching is recomputed after each local edit. For a candidate collection $A$, let $\mathrm{TP}(A)$ denote the maximum number of such pairs with $\mathcal{Y}_q^*$.

We evaluate answer-set quality using a smoothed $F_1$ score with a smoothing constant $\varepsilon > 0$, allowing the metric to distinguish answer-set sizes even when no answer is matched:

$$F_\varepsilon(A) = \frac{2\,\mathrm{TP}(A) + \varepsilon}{|A| + N + \varepsilon}.$$

### 4.2 Credit Assignment

**Thinking-unit credits.** We use $U_{\mathrm{think}}(\mathcal{U}) = \mathrm{TP}(\mathcal{U})/N$ to measure candidate-pool recall, so positive credit rewards additional coverage rather than correctness alone. Removing candidate $u_j$ yields

$$\Delta_j^{\mathrm{think}} = \frac{1}{N}\left[\mathrm{TP}(\mathcal{U}) - \mathrm{TP}(\mathcal{U}_{-j})\right]. \tag{4}$$

**Rethinking-unit credits.** We use $U_{\mathrm{rethink}}(\mathcal{U}, \mathcal{V}) = F_\varepsilon(S)$ to evaluate the retained answer set, so negative credit identifies a keep/drop decision whose flip improves answer-set quality. Let $S^{(j)}$ denote the selection obtained by reversing $v_j$, removing entry $j$ if kept and adding it if dropped:

$$\Delta_j^{\mathrm{rethink}} = F_\varepsilon(S) - F_\varepsilon(S^{(j)}). \tag{5}$$

**Free-form span credits.** We score $h^{\mathrm{think}}$ by the *candidate-pool ceiling*, the best answer-set $F_\varepsilon$ attainable from its candidate pool, placing candidate-pool potential and final answer quality on a common utility scale:

$$C(\mathcal{U}) = \max_{S \subseteq \mathcal{U}} F_\varepsilon(S) = \frac{2\,\mathrm{TP}(\mathcal{U}) + \varepsilon}{\mathrm{TP}(\mathcal{U}) + N + \varepsilon}.$$

We score both $h^{\mathrm{rethink}}$ and $h^{\mathrm{outcome}}$ by $F_\varepsilon(S)$, measuring the quality of the final answer set. Thus, for response $g$, the three scores are

$$r_{1,g} = C(\mathcal{U}_g), \qquad r_{2,g} = r_{3,g} = F_\varepsilon(S_g). \tag{6}$$

Centering these scores according to Equation 3 gives the credits $A_{1,g}$, $A_{2,g}$, and $A_{3,g}$ for the corresponding free-form spans.

### 4.3 Theoretical Properties

We examine whether thinking-unit credits align with attainable answer-set quality and whether rethinking-unit credits identify improving selection changes.

We use the maximum one-to-one matching definition of TP from Section 4.1. The matching is recomputed after each local edit. The following results hold for $N \geq 1$ and $\varepsilon > 0$.

**Proposition 1** (Thinking-unit credit and attainable utility)**.** *For every candidate $j$,*

$$C(\mathcal{U}) - C(\mathcal{U}_{-j}) = \frac{N(2N+\varepsilon)}{\big(\mathrm{TP}(\mathcal{U}) + N + \varepsilon\big)\big(\mathrm{TP}(\mathcal{U}) + N + \varepsilon - 1\big)}\,\Delta_j^{\mathrm{think}}.$$

Within a fixed candidate pool, thinking-unit credit is proportional to each candidate's marginal contribution to attainable $F_\varepsilon$. This connects recall-based local credit with the pool-ceiling score used for the free-form thinking span.

**Proposition 2** (Selection optimality under single flips)**.** *For any indexed selection $S$ from $\mathcal{U}$,*

$$\Delta_j^{\mathrm{rethink}} \geq 0 \text{ for all } j \quad \Longleftrightarrow \quad F_\varepsilon(S) = C(\mathcal{U}).$$

Whenever a selection falls below the ceiling, rethinking-unit credits identify an improving flip: removing an entry without reducing TP or adding one that increases TP. This guarantee holds within a fixed candidate pool under the stated assumptions. Proofs are provided in Appendix D.

Table 1: Results on DDXPlus with Qwen3-8B. The two untrained generation rows exclude empty or unparseable outputs (4,686/5,000 retained for direct generation; 5,000/5,000 for structured generation). A dash (−) indicates undefined metrics: pool recall for methods without candidate proposal and Spearman $\rho$ for constant-size outputs. Throughout the tables, bold and underlined values indicate the best and second-best quality and correlation scores.

| | Proposal quality | Answer-set quality | | | Set-size behavior | |
|---|---|---|---|---|---|---|
| Method | Pool recall | $F_1$ | Precision | Recall | Mean $\lvert S\rvert$ | Spearman $\rho$ |
| *Without task-specific training* | | | | | | |
| Direct generation | – | 0.150 | 0.289 | 0.132 | 3.19 | 0.06 |
| Structured generation | 0.276 | 0.171 | 0.270 | 0.180 | 5.70 | 0.03 |
| *With task-specific training* | | | | | | |
| Fixed $K = 6$ | – | 0.481 | 0.636 | 0.469 | 6.00 | – |
| Fixed $K = 9$ | – | 0.478 | 0.524 | 0.546 | 9.00 | – |
| Fixed $K = 12$ | – | 0.447 | 0.434 | **0.588** | 12.00 | – |
| GRPO (direct) | – | <u>0.576</u> | 0.760 | 0.492 | 6.01 | <u>0.87</u> |
| GRPO (structured) | 0.507 | 0.531 | 0.707 | 0.459 | 5.91 | 0.81 |
| GRPO-$\lambda$ (structured) | 0.472 | 0.524 | 0.760 | 0.430 | 5.10 | 0.83 |
| OAR-G (structured) | <u>0.522</u> | 0.561 | <u>0.764</u> | 0.476 | 5.84 | 0.84 |
| SODA | **0.628** | **0.657** | **0.821** | <u>0.580</u> | 6.48 | **0.89** |

# 5 Experimental Results

## 5.1 Experimental Setup

**Datasets.** We train separate policies on four datasets: DDXPlus (Tchango et al., 2022), our primary benchmark of synthetic patient cases with symptoms, medical history, and diagnosis annotations; ProtoQA (Boratko et al., 2020) for commonsense QA; BioASQ (Krithara et al., 2023) for biomedical QA; and MIMIC-III (Johnson et al., 2016) for ICD-9 diagnosis coding from discharge summaries. All predictions are normalized before matching. DDXPlus and ProtoQA use no additional alias mappings, BioASQ uses official aliases, and MIMIC-III lists diagnosis titles associated with ICD-9 codes in the prompt. Preparation, matching, and prompt details are provided in Appendix B.1.

**Training.** Our main experiments use Qwen3-8B with 32 questions per batch, 8 responses per question, and 15 candidates for structured reasoning. Full configurations are provided in Appendix B.2.

**Baselines.** We first evaluate the untrained base model under both direct and structured prompting. We compare SODA with five trained baselines (configurations are provided in Appendix B.3):

- **GRPO (direct)** generates the final answer set directly, using outcome-only supervision.
- **GRPO (structured)** uses the same thinking/rethinking structure and candidate budget as SODA, but retains outcome-only supervision with a shared response-level advantage.
- **Fixed-$K$** (Puri et al., 2026) optimizes recall while requiring exactly $K$ distinct answers. We train separate policies at $K \in \{6, 9, 12\}$.
- **GRPO-$\lambda$** (Parthasarathi et al., 2025) uses eligibility traces over token log-probabilities for credit assignment, with the same response structure and candidate budget as SODA.
- **OAR-G** (Li et al., 2026b) assigns token-level credit using outcome-sensitivity signals, with the same response structure and candidate budget as SODA.

**Evaluation.** We generate one response per question. We report precision, recall, and unsmoothed $F_1$, computed by comparing each predicted answer set with its reference set. We also report the mean predicted set size and the Spearman correlation ($\rho$) between predicted and reference set sizes.

## 5.2 Main Results

We first compare SODA with the full baseline suite on DDXPlus, then evaluate its effectiveness across additional datasets and base models. Each setting uses separately trained policies.

Table 2: Results across additional datasets with Qwen3-8B.

| | | Proposal quality | Answer-set quality | | | Set-size behavior | |
|---|---|---|---|---|---|---|---|
| Dataset | Method | Pool recall | $F_1$ | Precision | Recall | Mean $\lvert S\rvert$ | Spearman $\rho$ |
| ProtoQA | GRPO (structured) | 0.445 | 0.312 | 0.347 | 0.309 | 4.11 | 0.090 |
| | SODA | **0.493** | **0.332** | **0.388** | **0.313** | 3.74 | **0.112** |
| BioASQ | GRPO (structured) | 0.308 | 0.224 | 0.241 | 0.227 | 2.17 | **0.738** |
| | SODA | **0.398** | **0.242** | **0.285** | **0.230** | 1.55 | 0.630 |
| MIMIC-III | GRPO (structured) | 0.714 | 0.551 | 0.548 | 0.620 | 4.89 | 0.659 |
| | SODA | **0.860** | **0.606** | **0.599** | **0.679** | 5.14 | **0.699** |

Table 3: Results across additional base models on DDXPlus.

| | | Proposal quality | Answer-set quality | | | Set-size behavior | |
|---|---|---|---|---|---|---|---|
| Model | Method | Pool recall | $F_1$ | Precision | Recall | Mean $\lvert S\rvert$ | Spearman $\rho$ |
| gpt-oss-20b | GRPO (structured) | 0.321 | 0.286 | **0.552** | 0.228 | 3.01 | 0.297 |
| | SODA | **0.552** | **0.407** | 0.419 | **0.490** | 10.22 | **0.307** |
| Qwen3-30B-A3B | GRPO (structured) | 0.250 | 0.193 | 0.575 | 0.139 | 1.54 | 0.094 |
| | SODA | **0.567** | **0.500** | **0.622** | **0.480** | 6.35 | **0.589** |

**Comparison with baselines on DDXPlus.** Table 1 shows that SODA achieves the highest pool recall, final-set $F_1$, precision, and cardinality correlation among the evaluated methods. Its final-set recall is slightly below Fixed $K = 12$, which is constrained to return more answers per question. Notably, structured generation outperforms direct generation before training, yet GRPO (structured) underperforms GRPO (direct) after training. One possible explanation is that structured intermediate reasoning creates opportunities for candidate exploration and reconsideration, but shared outcome-level credit provides limited guidance for learning these distinct decisions. The improvement of SODA over GRPO (structured) supports the value of decision-specific credit assignment for learning to use the intermediate structure effectively.

**Results across datasets.** Table 2 reports results with Qwen3-8B on ProtoQA, BioASQ, and MIMIC-III. SODA achieves better proposal and final answer-set quality on all three datasets. Cardinality correlation also improves on ProtoQA and MIMIC-III. On BioASQ, despite lower cardinality correlation, SODA improves precision while maintaining similar recall, yielding better answer-set quality. Paired bootstrap confidence intervals for the $F_1$ improvements are provided in Appendix E.2.

**Results across base models.** We further evaluate SODA using gpt-oss-20b and Qwen3-30B-A3B on DDXPlus (Table 3). The consistent gains over outcome-only GRPO with structured responses across all three base models support the effectiveness of fine-grained credit assignment within structured reasoning across model families and sizes.

### 5.3 Ablation Studies

We examine the contributions of local credit assignment on DDXPlus with Qwen3-8B by independently enabling or disabling local credits for THINKING and RETHINKING units (Table 4). When disabled, each unit receives its corresponding stage's group-relative credit; free-form span credits remain unchanged. Detailed span-level credit assignments are provided in Appendix Table A.1.

Both THINKING- and RETHINKING-unit credits individually improve answer-set quality. Combining them yields the highest $F_1$, precision, and recall, supporting their complementary contributions. Although using only RETHINKING-unit credits achieves slightly higher candidate-pool recall, it does not translate into better final answers. The full method achieves higher final-answer quality despite slightly lower pool recall, suggesting more effective selection from the available candidates.

Additional comparisons of centering strategies are reported in Appendix Table E.3.

Table 4: Ablation of local credits on DDXPlus with Qwen3-8B. A checkmark (✓) indicates that local credits are enabled. The gray row marks SODA.

| Local credit | | Proposal quality | Answer-set quality | | | Set-size behavior | |
|---|---|---|---|---|---|---|---|
| Thinking | Rethinking | Pool recall | $F_1$ | Precision | Recall | Mean $\lvert S\rvert$ | Spearman $\rho$ |
| – | – | 0.595 | 0.570 | 0.698 | 0.518 | 6.79 | 0.852 |
| – | ✓ | **0.633** | 0.614 | 0.754 | <u>0.559</u> | 6.98 | 0.860 |
| ✓ | – | 0.621 | <u>0.636</u> | <u>0.800</u> | 0.557 | 6.40 | **0.890** |
| ✓ | ✓ | <u>0.628</u> | **0.657** | **0.821** | **0.580** | 6.48 | <u>0.887</u> |

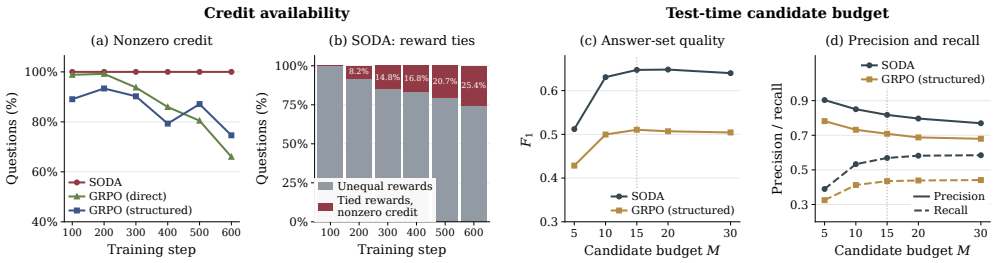


Figure 4: Diagnostic analyses on DDXPlus. **(a)** Fraction of questions with at least one nonzero token credit. **(b)** Decomposition of active questions for SODA. **(c,d)** Performance under varying test-time candidate budgets; models are trained with $M = 15$ and evaluated without retraining.

### 5.4 Diagnostic Analyses

**Credit availability under outcome ties.** We resample responses from each method's checkpoints on a held-out set and compute token credits under its corresponding credit rule. Sampling details are provided in Appendix B.4. As shown in Figure 4, as training progresses, credit availability decreases for baselines but remains high for SODA. When all responses to a question receive the same outcome reward, group centering yields zero outcome-based advantages, thus providing no training signals. By directly evaluating individual decisions, SODA can still assign nonzero local credits. It can also assign opposite credit signs within a response, distinguishing beneficial from detrimental decisions. These results show that SODA provides more persistent and differentiated training signals when outcome-level feedback no longer distinguishes sampled responses.

**Robustness to test-time candidate budgets.** We vary the candidate budget at inference without retraining models. SODA outperforms the baseline at every tested budget (Figure 4). Apart from the smallest budget, final-set $F_1$ remains largely stable: larger pools increase recall while reducing precision. This stability indicates that the learned proposal and selection behavior remains effective beyond the training budget. Detailed results are provided in Table E.4.

## 6 Conclusion

We introduce SODA, a framework that uses intermediate reasoning structure for fine-grained credit assignment. By making semantic decisions explicit and locally editable, SODA evaluates their contributions through task-defined utility differences and assigns differentiated training signals without additional policy rollouts or a learned critic. In set-valued QA, these signals supervise both candidate exploration and subsequent selection. Experiments demonstrate their benefits across multiple datasets and base models.

The applicability of SODA depends on task structures supporting local decision edits and task-specific utilities for evaluating their effects without downstream regeneration. Therefore, extending the framework to broader reasoning tasks requires identifying suitable decision structures and utilities. Within set-valued QA, a further direction is preference-conditioned training, enabling a single model to adjust its precision–recall trade-off according to user requirements at inference time.

## AI USE STATEMENT

We used AI assistants to support code development, manuscript drafting and editing. The authors reviewed and verified the resulting code, analyses, and text, and take full responsibility for the paper's accuracy, originality, and conclusions.

## REPRODUCIBILITY STATEMENT

Appendix A specifies span-level credit assignment, policy optimization, and ablation settings. Appendix B documents dataset preparation, training configurations, baseline implementations, evaluation protocols, and computational resources. Appendix C provides full prompts and response formats. Appendix D proves the theoretical results under their stated assumptions, and Appendix E reports additional results and analyses. Our implementation is available in an anonymous repository: `https://anonymous.4open.science/r/soda/`.

## REFERENCES


Nadia Alshahwan, Jubin Chheda, Anastasia Finogenova, Beliz Gokkaya, Mark Harman, Inna Harper, Alexandru Marginean, Shubho Sengupta, and Eddy Wang. Automated unit test improvement using large language models at meta. In *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering*, FSE 2024, pp. 185–196, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400706585. doi: 10.1145/3663529.3663839. URL https://doi.org/10.1145/3663529.3663839.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michał Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. Graph of thoughts: solving elaborate problems with large language models. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'24/IAAI'24/EAAI'24. AAAI Press, 2024. ISBN 978-1-57735-887-9. doi: 10.1609/aaai.v38i16.29720. URL https://doi.org/10.1609/aaai.v38i16.29720.

Michael Boratko, Xiang Lorraine Li, Tim O'Gorman, Rajarshi Das, Dan Le, and Andrew McCallum. ProtoQA: A question answering dataset for prototypical common-sense reasoning. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1122–1136, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.85. URL https://aclanthology.org/2020.emnlp-main.85/.

Jie Cheng, Gang Xiong, Ruixi Qiao, Lijun Li, Chao Guo, Junle Wang, Yisheng Lv, and Fei-Yue Wang. Stop summation: Min-form credit assignment is all process reward model needs for reasoning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=3Sxby0hH1q.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for LLM agent training. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=QXEhBMNrCW.

Yiran Guo, Lijie Xu, Jie Liu, Dan Ye, and Shuang Qiu. Segment policy optimization: Effective segment-level credit assignment in RL for large language models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=9osvTOYbT4.

Yuhang He, Haodong Wu, Siyi Liu, Hongyu Ge, Hange Zhou, Keyi Wu, Zhuo Zheng, Qihong Lin, Zixin Zhong, and Yongqi Zhang. Where hindsight credit can reside: A signed-capacity view of token updates in RLVR, 2026. URL https://arxiv.org/abs/2604.11056.

Anqi Hu, Zhiyuan Wang, Zijun Jia, and Bo Fu. MiRD: Reliable set-valued prediction for open-ended question answering via miscoverage risk decomposition, 2026. URL https://arxiv.org/abs/2605.27091.

Yuxiang Ji, Ziyu Ma, Yong Wang, Guanhua Chen, Xiangxiang Chu, and Liaoni Wu. Tree search for LLM agent reinforcement learning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=ZpQwAFhU13.

Xitai Jiang, Zihan Tang, Wenze Lin, Yang Yue, Shenzhi Wang, and Gao Huang. From reasoning chains to verifiable subproblems: Curriculum reinforcement learning enables credit assignment for LLM reasoning, 2026. URL https://arxiv.org/abs/2605.22074.

Hongbo Jin, Rongpeng Zhu, Zhongjing Du, Xu Jiang, Jingqi Tian, Qiaoman Zhang, and Jiayu Ding. DGPO: Distribution guided policy optimization for fine grained credit assignment, 2026. URL https://arxiv.org/abs/2605.03327.

Alistair EW Johnson, Tom J Pollard, Lu Shen, Li-wei H Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. MIMIC-III, a freely accessible critical care database. *Scientific data*, 3(1):160035, 2016.

Amirhossein Kazemnejad, Milad Aghajohari, Eva Portelance, Alessandro Sordoni, Siva Reddy, Aaron Courville, and Nicolas Le Roux. VinePPO: Refining credit assignment in RL training of LLMs. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=Myx2kJFzAn.

Anastasia Krithara, Anastasios Nentidis, Konstantinos Bougiatiotis, and Georgios Paliouras. BioASQ-QA: A manually curated corpus for biomedical question answering. *Scientific data*, 10(1):170, 2023.

Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, John D Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, Lei M Zhang, Kay McKinney, Disha Shrivastava, Cosmin Paduraru, George Tucker, Doina Precup, Feryal Behbahani, and Aleksandra Faust. Training language models to self-correct via reinforcement learning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=CjwERcAU7w.

Shufan Li, Konstantinos Kallidromitis, Akash Gokul, Yuta Kyuragi, and Aditya Grover. Guidance contrastive token credit assignment for discrete policy optimization, 2026a. URL https://arxiv.org/abs/2605.29198.

Ziheng Li, Liu Kang, Feng Xiao, Luxi Xing, Qingyi Si, Zhuoran Li, Weikang Gong, Deqing Yang, Yanghua Xiao, and Hongcheng Guo. Outcome-grounded advantage reshaping for fine-grained credit assignment in mathematical reasoning. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 24681–24693, San Diego, California, United States, July 2026b. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.1132. URL https://aclanthology.org/2026.acl-long.1132/.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding R1-Zero-like training: A critical perspective. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=5PAF7PAY2Y.

Ruotian Ma, Peisong Wang, Cheng Liu, Xingyan Liu, Jiaqi Chen, Bang Zhang, Xin Zhou, Nan Du, and Jia Li. $S^2$R: Teaching LLMs to self-verify and self-correct via reinforcement learning. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 22632–22654, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1104. URL https://aclanthology.org/2025.acl-long.1104/.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-Refine: Iterative refinement with self-feedback. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=S37hOerQLB.

James Mullenbach, Sarah Wiegreffe, Jon Duke, Jimeng Sun, and Jacob Eisenstein. Explainable prediction of medical codes from clinical text. In Marilyn Walker, Heng Ji, and Amanda Stent (eds.), *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 1101–1111, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1100. URL https://aclanthology.org/N18-1100/.

Prasanna Parthasarathi, Mathieu Reymond, Boxing Chen, Yufei Cui, and Sarath Chandar. GRPO-$\lambda$: Credit assignment improves LLM reasoning, 2025. URL https://openreview.net/forum?id=iRWqcnBlLQ.

Isha Puri, Mehul Damani, Idan Shenfeld, Marzyeh Ghassemi, Jacob Andreas, and Yoon Kim. Reaching beyond the mode: RL for distributional reasoning in language models, 2026. URL https://arxiv.org/abs/2603.24844.

Ankur Samanta, Akshayaa Magesh, Ayush Jain, Youliang Yu, Daniel Jiang, Kavosh Asadi, Kaveh Hassani, Paul Sajda, Jalaj Bhandari, and Yonathan Efroni. Credit assignment with resets in language model reasoning, 2026. URL https://arxiv.org/abs/2605.25507.

Yingyu Shan, Yuhang Guo, Zihao Cheng, Zeming Liu, Xiangrong Zhu, Xinyi Wang, Jiashu Yao, Wei Lin, Hongru Wang, and Heyan Huang. Learning from own solutions: Self-conditioned credit assignment for reinforcement learning with verifiable rewards, 2026. URL https://arxiv.org/abs/2606.18810.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Maohao Shen, Guangtao Zeng, Zhenting Qi, Zhang-Wei Hong, Zhenfang Chen, Wei Lu, Gregory W. Wornell, Subhro Das, David Daniel Cox, and Chuang Gan. Satori: Reinforcement learning with chain-of-action-thought enhances LLM reasoning via autoregressive search. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=j4FXxMiDjL.

Yuepeng Sheng, Yuwei Huang, Shuman Liu, Anxiang Zeng, and Haibo Zhang. ESPO: Entropy importance sampling policy optimization, 2026. URL https://arxiv.org/abs/2512.00499.

Qiming Shi, Zhaolu Kang, Yunfan Zhou, Di Weng, and Yingcai Wu. SPADER: Step-wise peer advantage with diversity-aware exploration rewards for multi-answer question answering, 2026. URL https://arxiv.org/abs/2606.00593.

Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. Is ChatGPT good at search? investigating large language models as re-ranking agents. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 14918–14937, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.923. URL https://aclanthology.org/2023.emnlp-main.923/.

Arsène Fansi Tchango, Rishab Goel, Zhi Wen, Julien Martel, and Joumana Ghosn. DDXPlus: a new dataset for automatic medical diagnosis. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

Jingyi Wang, Lei Zhu, Tengjin Weng, Song-Li Wu, Haochen Tan, Jierun Chen, Chaofan Tao, Haoli Bai, Lu Hou, Lifeng Shang, and Xiao-Ping Zhang. GRPO-VPS: Enhancing group relative policy optimization with verifiable process supervision for effective reasoning. In *ICLR 2026 Workshop on Logical Reasoning of Large Language Models*, 2026a. URL https://openreview.net/forum?id=Ise1xvtUF6.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-Shepherd: Verify and reinforce LLMs step-by-step without human annotations. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9426–9439, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.510. URL https://aclanthology.org/2024.acl-long.510/.

Xiangwei Wang, Wei Wang, Ken Chen, Nanduni Nimalsiri, and Saman Halgamuge. Discovering process-outcome credit in multi-step LLM reasoning, 2026b. URL https://arxiv.org/abs/2602.01034.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

Guofu Xie, Yunsheng Shi, Hongtao Tian, Ting Yao, and Xiao Zhang. CAPO: Towards enhancing LLM reasoning through generative credit assignment, 2025. URL https://arxiv.org/abs/2508.02298.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: deliberate problem solving with large language models. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Juncai Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Ru Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Yonghui Wu, and Mingxuan Wang. DAPO: An open-source LLM reinforcement learning system at scale. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=2a36EMSSTp.

Fengji Zhang, Xinyao Niu, Chengyang Ying, Guancheng Lin, Zhongkai Hao, Zhou Fan, Chengen Huang, Jacky Keung, Bei Chen, and Junyang Lin. $A^2$Search: Ambiguity-aware question answering with reinforcement learning. In *The Fourteenth International Conference on Learning Representations*, 2026a. URL https://openreview.net/forum?id=3CPzUWIoNf.

Wentao Zhang, Haoyu Zhang, Xinke Jiang, Yuxuan Cheng, Yuhan Pan, Miao Li, Zhipeng Qiao, Tao Feng, Zhen Tao, and Dengji Zhao. Fishing out free riders: Shapley-based reward attribution for parallel reasoning via reinforcement learning, 2026b. URL https://arxiv.org/abs/2607.18979.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V Le, and Ed H. Chi. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WZH7099tgfM.

# APPENDIX

The appendix first collects the details needed to reproduce our experiments. Appendix A specifies span-level credits, policy optimization, and ablation settings. Appendix B describes dataset preparation, training configurations, baseline implementations, evaluation protocols, and computational resources. Appendix C provides full prompts and response formats. Appendix D then proves the theoretical results in Section 4. Finally, Appendix E presents additional prompting results, uncertainty estimates, ablations, and test-time candidate-budget analysis.

## A METHOD IMPLEMENTATION DETAILS

This section specifies the span-level credits and policy update used with Sections 3–4, followed by the credit assignments in the ablation settings. Experimental settings and baseline implementations appear in Appendix B; full prompts appear in Appendix C.

### A.1 SPAN-LEVEL CREDITS AND POLICY OPTIMIZATION

We use the QA credit definitions in Section 4. The structured scoring object consists of the indexed candidates $\mathcal{U} = (u_1, \ldots, u_M)$ and their decisions $\mathcal{V} = (v_1, \ldots, v_M)$. Thinking-unit credit removes one candidate index; rethinking-unit credit reverses the corresponding keep/drop decision while holding the other decisions fixed. Edits change the scoring object, and credits are assigned to spans in the original response.

For each input $q$, we sample $G$ responses from $\pi_{\text{old}}$. For response $g$, each token receives a credit $\alpha_{g,t}$ according to its enclosing span:

$$\alpha_{g,t} = \begin{cases} A_{1,g}, & \text{if } t \text{ belongs to a free-form thinking span,} \\ \Delta^{\text{think}}_{g,j}, & \text{if } t \text{ belongs to thinking unit } j, \\ A_{2,g}, & \text{if } t \text{ belongs to a free-form rethinking span,} \\ \Delta^{\text{rethink}}_{g,j}, & \text{if } t \text{ belongs to rethinking unit } j, \\ A_{3,g}, & \text{if } t \text{ belongs to the outcome span.} \end{cases} \tag{7}$$

We minimize the importance-sampling loss

$$\mathcal{L}_q(\theta) = -\sum_{g=1}^{G} \sum_{t=1}^{T_g} \frac{\pi_\theta(y_{g,t} \mid q, y_{g,<t})}{\pi_{\text{old}}(y_{g,t} \mid q, y_{g,<t})} \alpha_{g,t}, \tag{8}$$

where $T_g$ is the response length, $\pi_\theta$ is the current policy, and $\pi_{\text{old}}$ is the sampling policy.

We train with an importance-sampling policy objective, without ratio clipping or additional advantage standardization. Each rollout batch receives one optimizer update. As shown in Equation 8, the loss sums token-level contributions across each question's responses without normalization by response length. Assigned credits are used directly as advantages, with prompt tokens excluded from the loss.

### A.2 CREDIT ASSIGNMENT IN ABLATION SETTINGS

Table A.1 specifies the signal assigned to each span. For response $g$, define

$$A_{1,g} = C(\mathcal{U}_g) - \frac{1}{G} \sum_{g'=1}^{G} C(\mathcal{U}_{g'}), \tag{9}$$

$$A^{\text{G}}_{2,g} = A^{G}_{3,g} = F_\varepsilon(S_g) - \frac{1}{G} \sum_{g'=1}^{G} F_\varepsilon(S_{g'}), \tag{10}$$

$$A^{\text{B}}_{2,g} = A^{B}_{3,g} = F_\varepsilon(S_g) - \frac{1}{|\mathcal{B}(g)|} \sum_{g' \in \mathcal{B}(g)} F_\varepsilon(S_{g'}). \tag{11}$$

Here $\mathcal{B}(g)$ is the lower or upper half of the question's sampled group, ranked by candidate-pool ceiling; ties use stable sort position. With $G = 8$, each bucket contains four responses. The THINKING stage always uses full-group centering. Only the rethinking stage and outcome signals change between Group and Bucket; $A_{3,g} = A_{2,g}$ under either strategy. Local credits are the raw utility differences in Equations 4 and 5, without centering. No group-standard-deviation normalization is applied.

Each response's free-form spans receive $A_{1,g}$, $A_{2,g}$, and $A_{3,g}$ in the thinking, rethinking, and outcome stages, respectively. Enabling a local credit replaces the corresponding unit span's stage-level credit; it does not add another term. Disabling it leaves the stage-level credit in place, rather than zeroing the span.

Table A.1: Span-level credit assignment in the local-credit ablations. Checkmarks enable local credits for the corresponding unit tokens; otherwise, those tokens retain their stage-level credit. Each entry is the complete assigned credit, not an additive term. The gray row is SODA. Local credits refer to unit $j$ of response $g$, with $g$ suppressed.

| | Local credit | | Thinking spans | | Rethinking spans | | Outcome |
|---|---|---|---|---|---|---|---|
| Centering | Think | Rethink | Free-form | Unit $u_j$ | Free-form | Unit $v_j$ | |
| Group | – | – | $A_{1,g}$ | $A_{1,g}$ | $A^{\mathrm{G}}_{2,g}$ | $A^{\mathrm{G}}_{2,g}$ | $A^{\mathrm{G}}_{2,g}$ |
| Group | – | ✓ | $A_{1,g}$ | $A_{1,g}$ | $A^{\mathrm{G}}_{2,g}$ | $\Delta^{\mathrm{rethink}}_j$ | $A^{\mathrm{G}}_{2,g}$ |
| Group | ✓ | – | $A_{1,g}$ | $\Delta^{\mathrm{think}}_j$ | $A^{\mathrm{G}}_{2,g}$ | $A^{\mathrm{G}}_{2,g}$ | $A^{\mathrm{G}}_{2,g}$ |
| Group | ✓ | ✓ | $A_{1,g}$ | $\Delta^{\mathrm{think}}_j$ | $A^{\mathrm{G}}_{2,g}$ | $\Delta^{\mathrm{rethink}}_j$ | $A^{\mathrm{G}}_{2,g}$ |
| Bucket | – | – | $A_{1,g}$ | $A_{1,g}$ | $A^{\mathrm{B}}_{2,g}$ | $A^{\mathrm{B}}_{2,g}$ | $A^{\mathrm{B}}_{2,g}$ |
| Bucket | – | ✓ | $A_{1,g}$ | $A_{1,g}$ | $A^{\mathrm{B}}_{2,g}$ | $\Delta^{\mathrm{rethink}}_j$ | $A^{\mathrm{B}}_{2,g}$ |
| Bucket | ✓ | – | $A_{1,g}$ | $\Delta^{\mathrm{think}}_j$ | $A^{\mathrm{B}}_{2,g}$ | $A^{\mathrm{B}}_{2,g}$ | $A^{\mathrm{B}}_{2,g}$ |
| Bucket | ✓ | ✓ | $A_{1,g}$ | $\Delta^{\mathrm{think}}_j$ | $A^{\mathrm{B}}_{2,g}$ | $\Delta^{\mathrm{rethink}}_j$ | $A^{\mathrm{B}}_{2,g}$ |

# B Experimental Setup

## B.1 Dataset Preparation

For each dataset, we train separate policies on its training split and evaluate them on its held-out split. We describe input and reference construction and dataset-specific matching rules here. Full prompt templates are provided in Appendices C.1 and C.2.

### B.1.1 DDXPlus

We use DDXPlus from the `aai530-group6/ddxplus` Hugging Face repository. After independently shuffling the source training and test splits with seed 42, we retain 25,000 training cases and 5,000 held-out cases. The main experiments use the first 19,200 training cases without replacement for 600 training steps, in the same order for all methods.

**Input and gold-set construction.** We convert each case's structured evidence into a natural-language clinical profile containing age, sex, symptoms, and antecedents. The reference answer set is formed from the pathology entries in `DIFFERENTIAL_DIAGNOSIS`. The resulting gold-set sizes have mean 9.21 and range 1–32 in training, and mean 9.28 and range 1–31 in the held-out set.

**Answer matching.** Following the exact-match evaluation setting of Puri et al. (2026), we normalize answers by lowercasing text, removing ASCII punctuation and the articles `a`, `an`, and `the`, and collapsing whitespace. Both training rewards and DDXPlus evaluation use normalized exact matching against reference diagnosis labels. Predictions are matched to reference diagnoses one-to-one, so each reference diagnosis contributes at most one true positive, while unmatched predictions remain in the precision denominator.

### B.1.2 PROTOQA

ProtoQA (Boratko et al., 2020) contains commonsense questions with multiple plausible answers.

**Input and gold-set construction.** We use the source training and validation splits, retaining 8,782 training questions and 980 validation questions for evaluation.

**Answer matching.** Training and evaluation use normalized exact matching: we lowercase text, remove ASCII punctuation and the articles `a`, `an`, and `the`, and collapse whitespace. Matching is one-to-one, so each slot contributes at most one true positive. We use no external alias dictionary, substring matching, or semantic similarity scorer.

### B.1.3 BIOASQ

BioASQ (Krithara et al., 2023) contains biomedical questions with expert-provided reference answers.

**Input and gold-set construction.** We use the official data splits, retaining 4,893 training questions and 400 evaluation questions of both the factoid and list types.

**Answer matching.** Training and evaluation use normalized exact matching against the accepted answer expressions supplied by the dataset: we lowercase text, remove ASCII punctuation and the articles `a`, `an`, and `the`, and collapse whitespace. A prediction is correct if it matches an accepted expression of a reference answer after normalization. Each reference answer can be matched only once, so listing multiple expressions of the same answer does not increase recall.

### B.1.4 MIMIC-III

We construct a diagnosis-coding task from MIMIC-III (Johnson et al., 2016), using hospital discharge summaries as inputs and ICD-9 diagnoses as targets. The prompt provides a menu of 50 diagnosis labels.

**Input and gold-set construction.** We use discharge summaries and diagnosis codes from MIMIC-III v1.4. We select the 50 most frequent diagnosis codes across all admissions with discharge summaries before splitting the data. Each admission's reference answer set contains the official titles of its selected diagnosis codes. Following Mullenbach et al. (2018), we use hospital admissions as examples. We construct a patient-level split using seed 42, yielding 44,348 training and 5,006 evaluation admissions.

**Answer matching.** Outputs are the ICD-9 `LONG_TITLE` strings verbatim (e.g. `"Unspecified essential hypertension"`). The menu shown to the model (below) is these 50 titles listed in a fixed order (sorted alphabetically), identical across every example.

## B.2 TRAINING CONFIGURATION

The retained Tinker v4 configurations for the reported model families support the common settings in Table B.1. The models are `Qwen/Qwen3-8B`, `Qwen/Qwen3-30B-A3B`, and `openai/gpt-oss-20b`. The Qwen runs specify `qwen3_disable_thinking`; gpt-oss uses `gpt_oss_low_reasoning`. LoRA is created with rank 32 through Tinker.

In all experiments, the training data are shuffled with seed 42 and used without replacement. Training is limited to 600 steps and stops sooner if the available data are exhausted.

Sampling sets the generation limit and renderer stop sequences; temperature, top-$p$, and top-$k$ otherwise use SDK defaults. Training combines task instructions and the question in one user message.

## B.3 BASELINE IMPLEMENTATIONS

**Shared configuration.** All baselines use Qwen3-8B with rank-32 LoRA, a learning rate of $2\times10^{-5}$, $G = 8$ sampled responses per question, and a batch size of $B = 32$ questions. Each rollout batch receives one optimizer update. Methods using the shared structured prompt adopt a candidate budget of $M = 15$. Evaluation uses the fixed 5,000-case DDXPlus panel, greedy decoding, a 2,048-token generation limit, and an 8,192-token context limit.

Table B.1: Training configuration for the experiments.

| Parameter | Value |
|---|---|
| Base model | Qwen3-8B |
| Questions per batch | 32 |
| Responses per question ($G$) | 8 |
| Candidate budget ($M$) | 15 |
| Random seed | 42 |
| Smoothing parameter ($\varepsilon$) | 1 |
| Credit coefficients | 1 |
| Updates per rollout batch | 1 |
| Loss API | `importance_sampling` |
| Fine-tuning method | LoRA, rank 32 |
| Learning rate | $2 \times 10^{-5}$ |
| Optimizer and parameters | Adam ($\beta_1$=0.9, $\beta_2$=0.95, $\epsilon$=$10^{-8}$) |
| Learning-rate schedule | Constant |
| Input truncation | MIMIC-III note only: 8,000 tokens |
| Maximum generation length | 12,288 tokens |
| Training sampling temperature | 1.0 |
| Other sampling parameters | top-$p = 1.0$; top-$k$ unrestricted |

**GRPO (direct).** GRPO (direct) uses the direct-generation prompt and smoothed final-set $F_1$ as the scalar reward. The group-centered credit $A^{\mathrm{G}}_{2,g}$ is broadcast to all response tokens.

**GRPO (structured).** GRPO (structured) uses the same THINKING/RETHINKING structure as SODA. The scalar reward is smoothed final-set $F_1$, and the group-centered credit $A^{\mathrm{G}}_{2,g}$ is broadcast to all response tokens. This baseline isolates the effect of structured deliberation from that of fine-grained credit assignment.

**Fixed-$K$** (Puri et al., 2026). Fixed-$K$ instructs the model to generate exactly $K$ answers and uses recall as the scalar reward. Responses receive zero reward if the parsed answer count differs from $K$ or if normalized answers contain duplicates. The budgets $K \in \{6, 9, 12\}$ were fixed before evaluation.

**OAR-G** (Li et al., 2026b). OAR-G uses the answer content of the last `FINAL SET:` line as the outcome span. Following the hyperparameter configuration in the original paper, Gaussian perturbations of pre-answer response embeddings produce a clean-versus-perturbed outcome-distribution KL signal, with noise standard deviation $0.01$. Gradient $\times$ Input attribution scores undergo log transformation and min–max normalization, followed by bi-level reshaping with threshold $0.4$ and boost factor 2. The resulting weights are normalized to mean one over attributed tokens, while answer and end tokens retain the scalar advantage. Responses without a detectable answer span fall back to uniform credit.

**GRPO-$\lambda$** (Parthasarathi et al., 2025). We implement the algorithm following the default hyperparameter configuration: $\lambda = 0.99$, $\gamma = 1$, clipping width $0.2$, reference-KL coefficient $0.04$, a negative-advantage floor of $-0.1$, and a gradient-norm cap of 1. We use the shared structured prompt and group-mean-centered smoothed final-set $F_1$ as the advantage.

### B.4 Evaluation Protocol

**Generation protocol.** We generate one response per question using greedy decoding. It applies the tokenizer chat template to a system instruction and a user question, with model-specific thinking/reasoning options. Models are loaded in bfloat16 with a LoRA adapter. Generation and context limits are 2,048 and 8,192 tokens, respectively. The checkpoint diagnostics in Figure 4 instead use a fixed 256-question test subset shuffled with seed 42, $G = 8$, temperature 1, top-$p = 1$, and generation/context caps of 2,048/8,192 tokens. They preserve each method's training prompt layout and disable model thinking.

**Parsing and answer matching.** The parser extracts the last `FINAL SET` line. Candidate and verdict indices use their last parsed occurrence. Equivalent but textually different answers can remain and count as unmatched predictions after one-to-one gold-slot matching. DDXPlus training and

evaluation use normalized exact matching against the reference labels. For datasets whose references provide multiple accepted forms per gold slot, a prediction may match any of those forms.

**Metric aggregation.** Precision, recall, and $F_1$ are computed per question and macro-averaged. Pool recall is averaged over all evaluation questions. $\rho$ is computed across questions from final-set size and gold-slot count, using average ranks for ties.

### B.5 Computational Resources and Code Availability

We conduct training using Tinker[1] SDK v0.25.0. Evaluation uses Tinker sampling, or self-hosted inference on NVIDIA RTX 6000 Blackwell and RTX A6000 GPUs. To ensure fair comparisons, we faithfully align the task and evaluation protocols across methods.

Our implementation is available in an anonymous repository: `https://anonymous.4open.science/r/soda/`.

## C Prompts and Response Formats

### C.1 Response Formats

Dataset-specific task instructions (Appendix C.2) are combined with the response-format instructions below: the dataset part supplies the case/question text (and, for MIMIC-III, the label menu), and the instructions below specify how the model must structure its answer.

**Direct generation**. The GRPO (direct) baseline generates the final answer set without intermediate structure.

```
You are given a problem whose answer is a SET of one or more
distinct answers.

Think step by step about which answers are correct. Some
questions have a single correct answer, others have many --
decide how many apply.

End with the last line exactly in this format:
FINAL SET: first answer; second answer
The FINAL SET line must contain ONLY the answers, separated by
semicolons, with no explanation and no numbering. Include every
answer you judge correct and nothing else.

Write everything as plain text. Do not use JSON, tool calls, or
code blocks.

[dataset-specific input]
```

**Structured prompt**. SODA and all the structured baselines use the same candidate–selection format described in full in Section 4: `THINKING`, `CANDIDATE`, `RETHINKING`, `VERDICT`, and `FINAL SET`.

```
You are given a problem whose answer is a SET of one or more
distinct answers.

Write your response in exactly this structure:

Step 1 -- Think step by step about which answers are plausible,
considering different interpretations and lines of evidence.

Then list exactly 15 candidate answers, one per line, RANKED
from most likely to least likely, in exactly this format:
CANDIDATE 1: most likely answer
```

[1] `https://thinkingmachines.ai/tinker/`

```
CANDIDATE 2: next most likely answer
CANDIDATE 3: ...
Give all 15 even if you are unsure about the later ones -- this
is a shortlist to choose from, not your answer. Each candidate
must be a short answer phrase (a few words), not a sentence or
explanation. Candidates must be genuinely different answers, not
paraphrases, restatements, or qualified variants of each other.

Step 2 -- Write a line saying only RETHINKING: and below it
re-examine each candidate on its own merits.

Then output one verdict line per candidate, in this format:
VERDICT 1: keep
VERDICT 2: drop
VERDICT 3: ...
Write keep only if you judge that candidate genuinely correct;
otherwise write drop. Work through them one at a time, deciding
each on its own merits. How many you keep is entirely your
decision -- some questions have a single correct answer, others
have many.

End with the last line exactly in this format:
FINAL SET: first kept answer; second kept answer
The FINAL SET line must list exactly the candidates you marked
keep, separated by semicolons, with no explanation and no
numbering.

Write everything as plain text. Do not use JSON, tool calls, or
code blocks.

[dataset-specific input]
```

**Fixed-$K$.** The Fixed-$K$ baseline (Appendix B.3) requests exactly $K$ distinct answers, with $K \in \{6, 9, 12\}$ substituted for {K} in the prompt below.

```
You are given a problem whose answer is a SET of one or more
distinct answers.

Think step by step about which answers are correct.

End with the last line exactly in this format:
FINAL SET: first answer; second answer
The FINAL SET line must contain ONLY the answers, separated by
semicolons, with no explanation and no numbering. Give exactly
{K} distinct answers you judge correct -- no more, no fewer, and
no repeats or rephrasings of the same answer.

Write everything as plain text. Do not use JSON, tool calls, or
code blocks.

[dataset-specific input]
```

### C.2 Dataset-Specific Inputs

The following task inputs are combined with the response instructions above. Dataset construction and reference answers are described in Appendix B.1.

**DDXPlus.** The model receives the clinical case in the following format. The pathology vocabulary is not included in the prompt.

```
Age: <age>
Sex: <sex>
```

```
Symptoms:
- <finding question>: <value>
- ...

Antecedents:
- <finding question>: <value>
- ...
```

**ProtoQA.** The `[dataset-specific input]` placeholder in Appendix C.1 is filled with ProtoQA's stripped `normalized-question` field, e.g. `"At the beach, name something that might protect you from the sun."`

**BioASQ.** The `[dataset-specific input]` placeholder in Appendix C.1 is filled with the BioASQ question text verbatim, e.g. `"Which gene therapies are under investigation for Duchenne muscular dystrophy?"`

**MIMIC-III.** The full 50-title menu is the 50 most frequent ICD-9 `LONG_TITLE` strings selected from the full eligible corpus as described above; we do not reproduce all 50 here for space but they are fully determined by that selection rule and are released with the code.

```
Possible diagnoses (choose only from this list; use the exact
wording given):
- <diagnosis title 1>
- <diagnosis title 2>
- ...
- <diagnosis title 50>

The following is a hospital discharge summary for one inpatient
admission. Based on it, identify which of the diagnoses above
apply to this admission.

[discharge summary]
```

### C.3 Motivating-Study Prompts

We evaluate Qwen3-8B without task-specific training under three prompting settings. The exact prompts are reproduced below, with placeholders indicating where each clinical case is inserted. Results and generation and parsing settings appear in Appendix E.1.

**Direct Generate.** The model generates an answer set directly, as a single user turn (system and case are not separated). The reported count is the number of answers in the parsed `FINAL SET` line.

```
You are given a problem whose answer is a SET of one or more
distinct answers.

Think step by step about which answers are correct. Some
questions have a single correct answer, others have many --
decide how many apply.

End with the last line exactly in this format:
FINAL SET: first answer; second answer
The FINAL SET line must contain ONLY the answers, separated by
semicolons, with no explanation and no numbering. Include every
answer you judge correct and nothing else.

Write everything as plain text. Do not use JSON, tool calls, or
code blocks.

[clinical case]
```

**Count.** The model estimates the number of diagnoses without naming them, as a single user turn. The reported count is the first integer parsed from the response.

```
You will be shown a patient case (demographics and symptoms).
Based on it, estimate how many DISTINCT diagnoses a clinician
would consider plausible as part of the differential for this
case.

Answer with ONLY a single integer on its own, nothing else --
no explanation, no range, no words.

Case:
[clinical case]

Number of plausible diagnoses:
```

**Propose–Select.** The untrained model uses the candidate–selection prompt in Appendix C.1, with $M = 15$, and returns the candidates marked keep. The reported count is the parsed final-set size.

## D Theoretical Proofs

We use the notation of Section 4. Fix a question $q$ with reference answers $\mathcal{Y}_q^*$, where $N = |\mathcal{Y}_q^*| \geq 1$, and a candidate collection $\mathcal{U} = (u_1, \ldots, u_M)$.

**Matching and scoring conventions.** Candidate entries and reference answers form a bipartite graph under the task's answer-matching rule. A candidate may be compatible with multiple reference answers, but each candidate and each reference answer can participate in at most one matched pair. For any indexed subcollection $A$ of $\mathcal{U}$, $\mathrm{TP}(A)$ denotes the cardinality of a maximum matching in the induced graph.

Entries are distinguished by their indices, including entries with identical text, and $|A|$ counts these entries. For a fixed $\varepsilon > 0$, define

$$\mathrm{Recall}(A) = \frac{\mathrm{TP}(A)}{N}, \qquad F_\varepsilon(A) = \frac{2\,\mathrm{TP}(A) + \varepsilon}{|A| + N + \varepsilon}.$$

All indexed subcollections, including the empty one, are admissible in the analysis. In particular, $\mathrm{TP}(\varnothing) = 0$ and $F_\varepsilon(\varnothing) = \varepsilon/(N + \varepsilon)$. The candidate-pool ceiling is

$$C(\mathcal{U}) = \max_{S \subseteq \mathcal{U}} F_\varepsilon(S),$$

where the maximum ranges over indexed subcollections.

**Local edits.** The selection $S$ contains exactly the candidate entries assigned a keep decision. A thinking edit removes only entry $j$ from $\mathcal{U}$. A rethinking edit reverses only decision $v_j$, removing entry $j$ from $S$ if kept and adding it if dropped. All other entries and decisions remain fixed. The compatibility relations of the remaining entries are unchanged, but the maximum matching is recomputed after every edit.

These edits operate on the structured scoring objects; they do not regenerate or revise the surrounding reasoning text. Throughout the proofs, write $m = \mathrm{TP}(\mathcal{U})$.

*Proof of Proposition 1.* We first establish the candidate-pool ceiling. A maximum matching of $\mathcal{U}$ has $m$ matched candidate entries. Retaining exactly these entries yields a selection with size and true-positive count both equal to $m$. For any other selection $S$, let $t = \mathrm{TP}(S)$ and $s = |S|$. Since $t \leq m$ and $s \geq t$,

$$F_\varepsilon(S) = \frac{2t + \varepsilon}{s + N + \varepsilon} \leq \frac{2t + \varepsilon}{t + N + \varepsilon} \leq \frac{2m + \varepsilon}{m + N + \varepsilon}.$$

The second inequality follows because

$$h(x) = \frac{2x + \varepsilon}{x + N + \varepsilon}$$

is strictly increasing for $x \geq 0$, with

$$h'(x) = \frac{2N + \varepsilon}{(x + N + \varepsilon)^2} > 0.$$

Consequently,

$$C(\mathcal{U}) = \max_{S \subseteq \mathcal{U}} F_\varepsilon(S) = h(m),$$

including the case $m = 0$, where the empty selection attains the ceiling.

Deleting one candidate entry reduces the maximum matching size by at most one: removing its incident edge, if any, from a maximum matching leaves a matching of size at least $m - 1$. Deletion cannot increase the maximum matching size. Therefore,

$$\mathrm{TP}(\mathcal{U}) - \mathrm{TP}(\mathcal{U}_{-j}) \in \{0, 1\}.$$

If deletion leaves the matching size unchanged, both $C(\mathcal{U}) - C(\mathcal{U}_{-j})$ and $\Delta_j^{\mathrm{think}}$ are zero. Otherwise, $\Delta_j^{\mathrm{think}} = 1/N$, and

$$h(m) - h(m-1) = \frac{2N + \varepsilon}{(m + N + \varepsilon)(m + N + \varepsilon - 1)}.$$

Combining the two cases gives

$$C(\mathcal{U}) - C(\mathcal{U}_{-j}) = \frac{N(2N + \varepsilon)}{(m + N + \varepsilon)(m + N + \varepsilon - 1)} \Delta_j^{\mathrm{think}},$$

as claimed. The coefficient is well-defined and positive for $m \geq 0$, $N \geq 1$, and $\varepsilon > 0$. □

**Sum of thinking-unit credits.** Deleting candidate $j$ reduces the maximum matching size if and only if that candidate is matched in every maximum matching of $\mathcal{U}$. Indeed, a maximum matching that leaves $j$ unmatched remains feasible after its deletion; conversely, a matching of size $m$ after deletion would also be a maximum matching of $\mathcal{U}$ that leaves $j$ unmatched.

There are at most $m$ such candidate indices, since every maximum matching contains exactly $m$ candidate entries. Hence

$$\sum_{j=1}^{M} \Delta_j^{\mathrm{think}} = \frac{|\{j : \mathrm{TP}(\mathcal{U}_{-j}) < m\}|}{N} \leq \frac{m}{N} = \mathrm{Recall}(\mathcal{U}). \tag{12}$$

Thus, thinking-unit credits measure coverage lost under individual deletions. Their sum need not equal total pool recall: alternative candidates can preserve the maximum matching size when any one of them is removed.

*Proof of Proposition 2.* Let

$$m = \mathrm{TP}(\mathcal{U}), \qquad s = |S|, \qquad t = \mathrm{TP}(S), \qquad D = s + N + \varepsilon, \qquad F = F_\varepsilon(S).$$

Since $t \leq \min(s, N)$ and $\varepsilon > 0$, we have $0 < F \leq 1$.

For a kept index, define $c_j = \mathrm{TP}(S) - \mathrm{TP}(S^{(j)})$; for a dropped index, define $c_j = \mathrm{TP}(S^{(j)}) - \mathrm{TP}(S)$. Adding or removing one candidate changes the maximum matching size by at most one, so $c_j \in \{0, 1\}$. Direct substitution yields

$$\Delta_j^{\mathrm{rethink}} = \begin{cases} \dfrac{2c_j - F}{D - 1}, & v_j = \mathrm{keep}, \\ \dfrac{F - 2c_j}{D + 1}, & v_j = \mathrm{drop}. \end{cases} \tag{13}$$

Both applicable denominators are positive. Removing a kept entry without reducing TP therefore improves utility and gives negative credit to the observed keep decision. Likewise, adding a dropped entry that increases TP improves utility and gives negative credit to the observed drop decision.

We now show that every selection below the ceiling admits one of these improving edits.

*Case 1:* $s > t$. Choose a maximum matching of $S$. At least one retained candidate is unmatched. Removing this candidate preserves that matching, and hence preserves $\mathrm{TP}(S) = t$ while reducing the selection size by one. This yields a strictly improving removal.

*Case 2:* $s = t$ *and* $F_\varepsilon(S) < C(\mathcal{U})$. Strict monotonicity of $h$ implies $t < m$. A maximum matching of $S$ matches all its candidate entries. Viewed in the full graph of $\mathcal{U}$, this matching is not maximum, so there exists an augmenting path.

The candidate endpoint of this path must lie outside $S$, because every candidate in $S$ is already matched. Every internal candidate on the path is matched by the current matching and therefore belongs to $S$. Thus, adding the single outside candidate makes the entire augmenting path available. Reassigning matches along the path increases TP from $t$ to $t + 1$. This yields a strictly improving addition.

These cases exhaust all suboptimal selections, because $s \geq t$. Therefore, if all rethinking-unit credits are nonnegative, the selection must attain $C(\mathcal{U})$.

Conversely, if $F_\varepsilon(S) = C(\mathcal{U})$, no single flip can increase utility: every $S^{(j)}$ is an admissible indexed subcollection of $\mathcal{U}$. Hence $\Delta_j^{\mathrm{rethink}} \geq 0$ for every $j$. This proves the equivalence. □

Together, these results show that thinking-unit credits align with attainable answer-set quality, while rethinking-unit credits identify an improving local edit whenever the current selection is suboptimal. The guarantees concern the indexed scoring objects and single-entry edits defined above, with maximum matching recomputed after each edit. They do not establish convergence of policy optimization.

# E Additional Results and Analyses

## E.1 Additional Prompting Results

Table E.1 supplements the three settings in Section 2. *Propose–Select* improves recall and $F_1$ over *Direct Generate* with slightly lower precision, while final-set cardinality correlation remains weak. Count prediction has a higher correlation, but its predictions concentrate on a few values. Full prompts appear in Appendix C.3.

All settings use Qwen3-8B without additional training, with greedy decoding and non-thinking mode. Figure 2(a,b), Table E.1, and the two untrained generation rows in Table 1 exclude empty or unparseable outputs, retaining 4,686 cases for *Direct Generate* and all 5,000 for *Propose–Select*. The reported generation metrics are therefore conditional on a nonempty parsed output; the retained samples differ between methods.

Table E.1: Supplementary prompting results, excluding empty or unparseable generation outputs. $|S|$ is the mean final-set size for *Direct Generate* and *Propose–Select*, and the mean stated estimate for *Count*. $\rho$ is its correlation with gold-set size.

| Setting | $\lvert S\rvert$ | $\rho$ | Precision | Recall | $F_1$ |
|---|---|---|---|---|---|
| *Direct Generate* | 3.19 | 0.06 | 0.289 | 0.132 | 0.150 |
| *Propose–Select* | 5.70 | 0.03 | 0.270 | 0.180 | 0.171 |
| *Count* | 9.67 | 0.31 | – | – | – |

## E.2 Evaluation-Sample Uncertainty

Table E.2 reports uncertainty in the macro-$F_1$ difference between SODA and GRPO (structured) for Tables 1 and 2. For each dataset, we resample evaluation cases with replacement, keeping the two methods' predictions paired, and recompute the mean per-case $F_1$ difference. We use 20,000 resamples (seed 42) and report the 2.5th and 97.5th percentiles. These per-dataset intervals are not adjusted for multiple comparisons and capture evaluation-case sampling uncertainty conditional on the fixed checkpoints and predictions, rather than training-seed or checkpoint-selection uncertainty.

The intervals exclude zero on DDXPlus, ProtoQA, and MIMIC-III. For BioASQ, the positive point estimate is accompanied by an interval that includes zero, so this evaluation does not establish a reliable positive improvement at the 95% confidence level.

Table E.2: Paired bootstrap confidence intervals for the improvement in macro-$F_1$ of SODA over GRPO (structured) with Qwen3-8B. $\Delta F_1 = F_1(\text{SODA}) - F_1(\text{GRPO})$. Intervals quantify evaluation-sample uncertainty for fixed policies.

| Dataset | Testing set size | GRPO (structured) | SODA | $\Delta F_1$ | 95% CI for $\Delta F_1$ |
|---|---|---|---|---|---|
| DDXPlus | 5,000 | 0.5312 | 0.6566 | $+0.1254$ | $[+0.1214, +0.1294]$ |
| ProtoQA | 980 | 0.3120 | 0.3316 | $+0.0196$ | $[+0.0107, +0.0284]$ |
| BioASQ | 400 | 0.2240 | 0.2424 | $+0.0184$ | $[-0.0089, +0.0458]$ |
| MIMIC-III | 5,006 | 0.5508 | 0.6055 | $+0.0547$ | $[+0.0499, +0.0595]$ |

### E.3 Additional Ablation Results

Table E.3 supplements the ablation study in the main paper by checking the effect of centering strategy: for each configuration, we compare Group and Bucket centering for RETHINKING and outcome credits. Group subtracts the mean final-set utility over all responses to the same question. Bucket ranks responses by candidate-pool ceiling $F_1$ as a proxy for overall quality, divides them into lower and upper halves, and subtracts the mean final-set utility within each half. THINKING scores always use full-group centering; local credits remain uncentered and unstandardized.

With local credits enabled, bucket centering provides no additional benefit over group centering. One possible explanation is that fine-grained credits already evaluate structured decisions in context, leaving bucket centering to affect only the free-form rethinking and outcome spans. Its benefit may therefore be limited, while splitting the group reduces the number of responses used to estimate each baseline. These results support group centering for SODA.

Table E.3: Ablation of local credits and centering strategies on DDXPlus with Qwen3-8B. A checkmark (✓) indicates that local credits are enabled; otherwise, those tokens retain the group-relative credit. Free-form span utilities remain unchanged. The gray row marks the SODA setting used in the main experiments. Bold and underlined values indicate the best and second-best quality and correlation scores across configurations, respectively.

| | Local credit | | Proposal quality | Answer-set quality | | | Set-size behavior | |
|---|---|---|---|---|---|---|---|---|
| Centering | Thinking | Rethinking | Pool recall | $F_1$ | Precision | Recall | Mean $\lvert S\rvert$ | Spearman $\rho$ |
| Group | | | 0.595 | 0.570 | 0.698 | 0.518 | 6.79 | 0.852 |
| | | ✓ | **0.633** | 0.614 | 0.754 | 0.559 | 6.98 | 0.860 |
| | ✓ | | 0.621 | 0.636 | 0.800 | 0.557 | 6.40 | <u>0.890</u> |
| | ✓ | ✓ | <u>0.628</u> | **0.657** | **0.821** | **0.580** | 6.48 | 0.887 |
| Bucket | | | 0.610 | 0.584 | 0.682 | 0.549 | 7.56 | 0.854 |
| | | ✓ | 0.597 | 0.604 | 0.764 | 0.533 | 6.45 | 0.868 |
| | ✓ | | 0.621 | 0.637 | 0.780 | <u>0.568</u> | 6.54 | **0.893** |
| | ✓ | ✓ | 0.603 | <u>0.639</u> | <u>0.813</u> | 0.560 | 6.25 | 0.874 |

### E.4 Test-Time Candidate Budgets

Table E.4 reports answer-set quality and set-size behavior as the test-time candidate budget varies for the fixed trained policies. We uniformly sample 1,000 questions without replacement from the held-out set. Empty or unparseable outputs remain in the evaluation. The sweep holds the trained policies fixed and varies only the requested candidate budget within its local evaluation pipeline.

Table E.4: Performance under different test-time candidate budgets on a fixed random subset of 1,000 DDXPlus test questions (seed 42). All methods and budgets use the same subset. Policies trained with $M = 15$ are evaluated without further training.

| | | Answer-set quality | | | Set-size behavior | |
|---|---|---|---|---|---|---|
| $M_{\text{test}}$ | Method | $F_1$ | Precision | Recall | Mean $\lvert S\rvert$ | Spearman $\rho$ |
| 5 | GRPO (structured) | 0.428 | 0.782 | 0.325 | 3.14 | 0.676 |
| | SODA | **0.512** | **0.903** | **0.390** | 3.15 | **0.728** |
| 10 | GRPO (structured) | 0.500 | 0.732 | 0.413 | 5.04 | 0.774 |
| | SODA | **0.631** | **0.850** | **0.533** | 5.44 | **0.858** |
| 15 | GRPO (structured) | 0.511 | 0.709 | 0.435 | 5.68 | 0.786 |
| | SODA | **0.647** | **0.818** | **0.569** | 6.37 | **0.878** |
| 20 | GRPO (structured) | 0.507 | 0.687 | 0.439 | 5.98 | 0.789 |
| | SODA | **0.648** | **0.797** | **0.582** | 6.77 | **0.872** |
| 30 | GRPO (structured) | 0.504 | 0.680 | 0.441 | 6.17 | 0.771 |
| | SODA | **0.640** | **0.769** | **0.585** | 7.04 | **0.875** |