CROSS-TASK TRANSFER VIA MULTI-ROLE REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning with verifiable rewards is often used to train a language model for a single role that includes a problem-solving task and an objective based on answer correctness. Humans often learn differently; they decompose a problem or draw on relations that span examples, pursuing goals that are not directly tied to the correct answer to any single problem. Inspired by this, we ask whether training a language model on the same examples with auxiliary roles, alongside the primary one, can improve its test performance. We define each role by a task, what the model is shown and asked for, and an objective, how its output is rewarded. We formalize this setting as *Role-Based Reinforcement Learning (RBRL)*, in which one model is trained in several roles. To evaluate our approach, we use GSM-Symbolic with a single distractor clause, which is known to substantially reduce LLM accuracy, and define two auxiliary roles: one that identifies the distractor clause within a problem and another that writes a method to solve every problem sharing a template. When the model is trained in these multiple roles, where the auxiliary roles include distractors but the primary role does not, the resulting model solves more problems with distractors at test time than when trained in the primary role alone. We interpret this gain in terms of *cross-task transfer*. Specifically, the training in the auxiliary tasks improves model performance on the primary task at test time. When distractors are also added to the primary role training, the model improves further, suggesting that the benefits of each role may be additive. We also observe the same additive pattern on reading comprehension tasks. An auxiliary role that states what a passage implies without actually asking any questions improves question answering on RACE-C and on five other datasets we did not train the model on.

1 INTRODUCTION

Human learners can improve at completing a task by practicing something other than the task itself. Comparing problems of the same type reveals a general schema that carries over to new problems (Gick & Holyoak, 1983), and identifying the subgoals of a problem before solving it improves performance on later problems that require additional procedures (Catrambone, 1998). In both studies, the practice tasks share problems and some procedures with the test task, but participants are never asked to answer the problems during practice. We distinguish such practice by whether it operates **across instances**, as comparing several problems does, or **within an instance**, as identifying the subgoals of a problem does.

Reinforcement learning with verifiable rewards (RLVR) (Guo et al., 2025) is often used to train a language model for a single role. Inspired by these human studies, we ask: *can training a language model on the same examples with auxiliary roles, alongside the primary one, further improve its test performance?* To study this, we define *Role-Based Reinforcement Learning (RBRL)*, in which one model is trained in several roles over the same content. A *role* is both a task, what the model is shown and asked for, and an objective, how its output is rewarded. We call the primary role typically used for RLVR the REASONER. It is shown the problem, asked for a solution, and rewarded when the solution's answer matches the ground-truth answer.

To test this idea in a controlled way, we impose conditions that isolate the effects of auxiliary roles. Auxiliary roles never pass their outputs to the reasoner during training and are never invoked at

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.

test time. We train every model on the same number of training rollouts, so adding an auxiliary role reallocates rollouts from the reasoner rather than adding training. Under these conditions, an auxiliary role can affect the reasoner only through updates to the model parameters. This lets us interpret gains as *cross-task transfer* from the auxiliary role tasks to the reasoner's task over the same examples.

On GSM-Symbolic (Mirzadeh et al., 2025), where a single distractor clause in a problem is known to sharply reduce accuracy, we define two auxiliary roles around this failure mode, neither of which tasks the model to produce an answer directly. In the DETECTOR role, the model reads the problem and is rewarded for identifying its distractor clause, a relation **within an instance**. In the ABSTRACTOR role, it reads several instances of a problem template and is rewarded for producing a method that solves held-out instances, a relation **across instances**. We test in two settings that differ in whether the reasoner trains on problems with distractors. First, we train the model in the primary role on problems without distractors, which we call *clean*, while the auxiliary role training has problems with distractors. We view a role that improves on the primary task in this setting as a *substitutive* role. In the second setting, we also include distractors in the primary role training. We then interpret a role that provides further improvements as an *additive* role.

We find cross-task transfer in both settings. With the reasoner trained on clean problems, either auxiliary role recovers roughly a third to a half of the gain from training the reasoner on problems with distractors directly, so both roles are substitutive. With the reasoner trained on problems with distractors, both roles still improve it, so both are additive, but by different amounts. The detector adds about a tenth of the gain the reasoner itself makes from training on those problems, and the abstractor adds at least a third of it. We trace this difference to the number of instances the abstractor is shown. When we narrow the abstractor's view to the reasoner's, so that it is shown one problem, the additivity matches the detector's.

To test whether additivity holds beyond GSM-Symbolic, we define an UNDERSTANDER role for reading comprehension that, like the abstractor, operates across instances. Each question about a passage is an instance: the understander is shown the passage without its questions, writes what the passage implies, and is rewarded by the accuracy with which the model then answers every question about the passage from that statement, so it is rewarded for content useful across questions rather than for any one of them. We train it on RACE-C, a multiple-choice comprehension dataset, alongside a reasoner that trains on the same questions, and find that it improves the reasoner on RACE-C and on five further comprehension datasets the model is not trained on. Taken together, we find cross-task transfer between roles that share training data, and that a role can be substitutive, additive, or both. An additive role shows that the same data holds more training signal than the primary role alone extracts.

2 RELATED WORK

Cross-task transfer. Multi-task learning trains one network on several tasks at once, each with its own labels (Caruana, 1997). Auxiliary tasks, added only to help the main task and dropped at test time, go back to features used as outputs (Caruana & de Sa, 1996) and reach reinforcement learning as auxiliary control and prediction tasks over shared weights (Jaderberg et al., 2017). Taskonomy maps which visual tasks help which (Zamir et al., 2018), and later work groups tasks by how they affect one another under joint training (Standley et al., 2020).RBRL can be viewed as extending the line of work initiated by UNREAL (Jaderberg et al., 2017) to language models. Auxiliary roles in RBRL take the place of auxiliary tasks in this line, trained over shared weights and dropped at test time. We ask which auxiliary roles help the main role and when.

Cross-task transfer in language models. An intermediate task helps a target task most when it demands higher-level reasoning (Pruksachatkun et al., 2020), and a second task can be dropped at test time, as when a model is trained to write explanations alongside its labels (Li et al., 2022; Hsieh et al., 2023). Under RLVR, RL on mathematics carries to science, coding, and planning (Huan et al., 2025), and RL on compositions of skills carries to a task that received no RL (Yuan et al., 2026). In all of these, the transfer is measured on tasks across datasets. RBRL asks whether transfer exists on different roles within the same dataset.

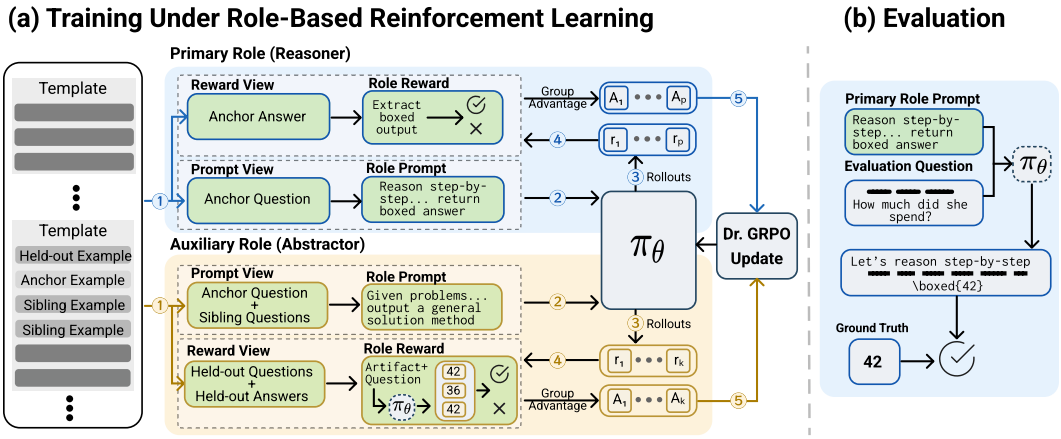

Figure 1: (a) Training. Every role is anchored at the same example, here a question of one GSM-Symbolic template. A task is its prompt view with its prompt, and an objective is its reward view with its reward. The abstractor calls π_θ to answer held-out siblings, and their average accuracy is the reward. Advantages are formed within each role. (b) Evaluation. No auxiliary role is invoked.

Multi-role RL. Recent work has one model occupy more than one position in its own training loop: proposing problems and solving them (Zhao et al., 2025; Liu et al., 2025a), or training a critic, judge, teacher, or hinter from the same backbone whose reward is the solver's subsequent gain (Lin et al., 2026; Chen et al., 2025; Ning et al., 2024; Xia et al., 2026). RLAD trains an abstraction generator rewarded by a solver's accuracy given the abstraction, which the solver reads at test time (Qu et al., 2026). In each, the second role exists to improve the solver, and it feeds the solver or supplies its reward, so the solver's gain cannot be separated from the extra input or signal it receives. Our auxiliary roles never feed the reasoner, the reasoner never reads their output, and no role is trained to reward another. This isolates transfer through the parameters, which lets us ask whether an auxiliary role helps the reasoner at all, rather than assume it does.

3 Role-Based Reinforcement Learning

In its simplest form, RLVR is used to train a model on examples that pair questions with ground-truth answers. For each example, the model is shown a question and a prompt that asks it to produce the answer. A reward function compares the answer with the ground truth. Each example is thus split between what the model sees and what the reward function sees.

Humans can learn from more than one example at a time, and some of what a learner produces, such as a method or a summary, has no answer key: its value is whether it helps on later problems. RLVR in the form above includes neither. We therefore define a *role* as a task together with an objective, and generalize both. A *view* is drawn from the dataset around an anchor example and can be part of that example, the whole example, or a set of related examples, such as other instances of the same template or other questions about the same passage. The *task* of a role is a prompt view together with a prompt that asks for a particular kind of output from it. The *objective* of a role is defined by a reward function together with a reward view, which is what the output is scored on. For the reasoner, the reward view is the ground-truth answer. For a role whose output has no ground-truth target, the reward view is a set of related problems, and the reward function measures the downstream performance the output enables by calling the model on those problems. Role-Based Reinforcement Learning (RBRL) trains one model in several roles over one dataset.

3.1 Framework

Let $\mathcal{D}$ be a dataset of examples and θ the parameters of an LLM π_θ. We define a role $k \in \mathcal{K}$ as a triple (ϕ_k, ρ_k, r_k) of a *view function*, a *prompt template*, and a *reward function*. A training step draws an anchor example $e \sim \mathcal{D}$ and, for each role, constructs the role's view $(x_{k,\rho}, x_{k,r}) = \phi_k(e; \mathcal{D})$.

The prompt template renders the first part of the view as a prompt, $\rho_k(x_{k,\rho})$, from which rollouts are sampled, $y \sim \pi_\theta(\cdot \mid \rho_k(x_{k,\rho}))$. The reward function uses the second part of the view to calculate $r_k(x_{k,r}, y)$. The task of the role is defined by the first part of its view together with the prompt template, and its objective is to maximize its expected reward,

$$\max_\theta \ \mathbb{E}_{y\sim\pi_\theta(\cdot|\rho_k(x_{k,\rho}))}[r_k(x_{k,r}, y)]\,. \tag{1}$$

A single policy acts in every role (Figure 1a), every role is anchored at the same e and draws from the same $\mathcal{D}$, and what a role collects around e is part of the role. The roles share the weights and are optimized jointly by maximizing the expected reward summed over roles,

$$J(\theta) \ = \ \mathbb{E}_{e\sim\mathcal{D}} \sum_{k\in\mathcal{K}} \mathbb{E}_{y\sim\pi_\theta(\cdot|\rho_k(x_{k,\rho}))}[r_k(x_{k,r}, y)]\,, \qquad (x_{k,\rho}, x_{k,r}) = \phi_k(e;\, \mathcal{D}). \tag{2}$$

In our implementation, all roles are optimized in one joint update with Dr. GRPO (Liu et al., 2025b) (Figure 1a), and advantages are computed within each role, using only its own n_k rollouts to form the group baseline, where n_k is the share of the run's fixed budget of N trainable rollouts that role k receives, $\sum_k n_k = N$.

3.2 The Primary Role: Reasoner

The **primary role** is the reasoner, trained on question and answer pairs. The prompt view is the question of the example, $x_{\mathrm{pri},\rho} = q(e)$. The prompt asks for step-by-step reasoning toward a boxed answer. The reward view is the gold answer, $x_{\mathrm{pri},r} = a^\star(e)$, and the reward function checks for an exact match,

$$r_{\mathrm{pri}}(a^\star(e), y) \ = \ \mathbb{1}[\hat{a}(y) = a^\star(e)]\,, \tag{3}$$

with $\hat{a}(\cdot)$ the boxed answer. This is the only prompt used at evaluation, and the reasoner trained alone on the full budget is the baseline against which every model trained with a role is compared.

3.3 Rewarding an Artifact

Our auxiliary roles produce an *artifact* $u(y)$, a span extracted from the rollout. The artifact has no ground-truth target of its own, and we do not learn a reward model to predict its quality. Instead, the model is asked to solve downstream problems given the artifact, and the accuracy of those answers is the artifact's reward. The artifact is scored only by what it lets the model do, so the reward stays verifiable. Similar indirect scoring appears in self-play (Liu et al., 2025a), learning by teaching (Ning et al., 2024) and RLAD (Qu et al., 2026). The view for role k collects this *scored set* $C_k(e) \subseteq \mathcal{D}$ along with what the prompt renders, so $C_k(e)$ is part of $x_{k,r}$. The role also fixes a *reward prompt* $\kappa_k(c, u)$ that presents a problem c together with the artifact. The reward samples m answers for each problem in the scored set,

$$\mathrm{acc}_\theta(u;\, C) \ = \ \frac{1}{m\,|C|} \sum_{c\in C} \sum_{j=1}^{m} \mathbb{1}\Big[\hat{a}(y_c^{(j)}) = a^\star(c)\Big]\,, \qquad y_c^{(j)} \sim \pi_\theta(\cdot \mid \kappa_k(c, u))\,. \tag{4}$$

These m reward rollouts per problem only score the artifact. They are not trained on and do not count against the training rollout budget N. A degenerate artifact, one that is empty, over length, or states an answer, receives zero reward.

3.4 Isolating Cross-Task Transfer

To isolate the effect of auxiliary roles, we impose three conditions. **Role independence**: ϕ_{pri} never invokes π_θ and never receives another role's output, so nothing passes from an auxiliary role to the primary role during training. **Primary-only evaluation**: at test time the model receives only $\rho_{\mathrm{pri}}(x_{\mathrm{pri},\rho})$, and no auxiliary output is produced. **Budget conservation**: $\sum_k n_k = N$ in every model, the baseline sets $n_{\mathrm{pri}} = N$, and auxiliary roles take their rollouts from the primary role's share, so adding a role takes rollouts away from the primary role rather than adding training. Under these conditions, an auxiliary role can affect the primary role only through updates to the weights they share, so any gain is cross-task transfer from the auxiliary role's task to the primary one.

Table 1: GSM-NoOp accuracy (%). The reasoner is trained on clean problems only (**Reasoner without Distractors**) or on problems with distractors as well (**Reasoner with Distractors**); the abstractor is scored on problems with distractors in every model and the detector sees both kinds. **Unseen** problems back no training template.

Training Conditions	**Reasoner without Distractors**				**Reasoner with Distractors**			
Testing Conditions	**Clean**		**With distractor**		**Clean**		**With distractor**	
	Unseen	**Seen**	**Unseen**	**Seen**	**Unseen**	**Seen**	**Unseen**	**Seen**
Qwen3-4B-Base								
Base model	91.9	80.3	54.1	54.7				
Reasoner	94.7	93.6	58.3	63.1	95.4	93.3	72.6	75.3
+ Detector	92.2	92.3	**65.4**	68.2	**95.5**	93.7	74.5	76.2
+ Abstractor	**95.1**	**93.8**	63.1	**69.2**	93.9	**94.6**	**78.7**	**80.4**
Qwen3-8B-Base								
Base model	94.0	84.3	62.3	63.4				
Reasoner	**96.1**	93.5	62.4	67.7	**96.0**	94.5	75.2	78.8
+ Detector	95.5	96.2	**68.4**	**74.3**	93.9	**95.5**	76.8	81.1
+ Abstractor	95.7	**97.1**	68.0	74.1	**96.0**	95.2	**83.6**	**84.5**

4 Arithmetic under Distraction: Detector and Abstractor

Arithmetic word problems expose a particular failure mode of LLMs, which are easily distracted by irrelevant context (Shi et al., 2023). GSM-Symbolic (Mirzadeh et al., 2025) shows that adding a single distractor clause, information that looks relevant but is not required to answer the question, reduces base model accuracy from 91.9% to 54.1% at 4B and from 94.0% to 62.3% at 8B in the models we evaluate (Table 1). The distractor clause gives structure within a problem, and the dataset's templates give structure across problems. We use these two forms of structure to define auxiliary roles. The detector asks the model to distinguish a distractor from information needed for the solution, while the abstractor asks it to provide a method to solve instances across the template.

4.1 Setup

We divide the available GSM templates (Mirzadeh et al., 2025) into train (*seen*) and test (*unseen*) sets. We train on 49 GSM-Symbolic templates, sampling $5{,}782$ training instances ($1{,}954$ clean, $3{,}828$ with a distractor clause), and evaluate on an external audited No-Op benchmark (Sturgeon, 2026) with independently written distractor clauses: 204 unseen and 134 seen pairs of a clean problem and its twin with a distractor. Appendix A describes the data construction, benchmark audit, and splits.

Models and optimization. We use Qwen3-4B-Base and Qwen3-8B-Base with rank 64 LoRA adapters on the attention and MLP projections, trained with Dr. GRPO at a constant learning rate of 2×10^{-5} and KL coefficient $\beta = 0.02$, with batch size 128 and sampling temperature 1.0. The optimizer settings were searched at each model size, which are then held fixed for every dataset and every model (Appendix B). Every model trains $N{=}16$ rollouts per example. The reasoner baseline spends all 16 on reasoning, and a model with one auxiliary role gives 8 to the reasoner and 8 to the role. Models with both auxiliary roles are reported in Appendix G.

Selection and evaluation. Each model uses its best dev checkpoint, selected on a split disjoint from all reported benchmarks (Appendix C). Every evaluation uses only the reasoner prompt, reporting average correctness over 8 samples. Appendix K reports standard errors across the samples.

Experimental settings. We train the reasoner in two settings that differ in whether it sees problems with distractors. In the *Reasoner without Distractors* setting, the reasoner trains on clean problems only, while the auxiliary roles train on the full pool of problems, with and without distractors. A role is *substitutive* if it improves test performance on problems with distractors in this setting, since the reasoner never trains on them. In the *Reasoner with Distractors* setting, the reasoner also trains on the full pool, and a role is *additive* if it improves the reasoner further.

4.2 ROLES

Within an instance: DETECTOR. A model can be tasked to find the distractor clause within the problem. The prompt view, like the reasoner's, is the anchor problem itself, $x_{\mathrm{det},\rho} = q(e)$. Since a third of the problems carry no distractor, the prompt asks for a verdict $v(y) \in \{\mathsf{yes}, \mathsf{no}\}$ on whether the problem carries one and, if yes, for a quote of the clause as a warning $w(y)$. The reward view is the problem, its gold answer and whether it carries a distractor, $\delta(e) \in \{0, 1\}$. The reward first checks the verdict against $\delta(e)$. The reward for no on a clean problem is 1. On a problem with a distractor the reward prompt κ_{det} deletes the quoted clause from the problem and issues the reasoner prompt, and the reward is the policy's accuracy on what remains,

$$r_{\mathrm{det}}(e, y) \;=\; \mathbb{1}[v(y) = \delta(e)] \cdot \begin{cases} 1 & \delta(e) = 0, \\ g_{\mathrm{det}}(w, e) \cdot \mathrm{acc}_\theta(w;\, \{e\}) & \delta(e) = 1, \end{cases} \tag{5}$$

where $g_{\mathrm{det}} \in \{0, 1\}$ requires a warning that is present, non-degenerate, and does not leak the answer (Appendix F.1).

Across instances: ABSTRACTOR. Where the detector looks inside the example, the abstractor looks above it. Its view draws $\ell + h$ instances of the anchor's template (Figure 1a), the anchor among them, and splits them into a shown set S of ℓ instances and a withheld set H of h instances, so that $x_{\mathrm{abs},\rho} = \{q(c) \,:\, c \in S\}$ and $x_{\mathrm{abs},r} = H$. The prompt renders the shown instances and asks for a general solution for the template, a method $u(y)$ that solves every instance. The reward prompt κ_{abs} presents each withheld instance with the method alone, so the method must be complete, and the reward is the policy's accuracy on the withheld set,

$$r_{\mathrm{abs}}(H, y) \;=\; \mathrm{acc}_\theta(u(y);\, H)\,. \tag{6}$$

The abstractor sees $\ell = 3$ instances of a template and is scored on $h = 2$ withheld instances with distractors, with $m = 4$ samples per instance for both roles.

4.3 RESULTS AND DISCUSSION

Both roles are substitutive. We report the results in Table 1 for both settings. We focus on accuracy on the unseen problems with distractors, as the seen ones follow, and accuracy on clean problems sits at 92–97% for every trained model at both scales. In both settings, the differences from the reasoner on clean problems go both ways and stay within a few percent, so an auxiliary role leaves accuracy on problems without a distractor clause essentially unchanged. The **Reasoner without Distractors** gains little over the base model on unseen problems with distractors ($54.1\% \to 58.3\%$ at 4B, $62.3\% \to 62.4\%$ at 8B). Adding an auxiliary role, which is never asked to produce an answer, raises it by at least 4.8% at 4B and 5.6% at 8B. Training the reasoner on problems with distractors directly raises it by 14.3% and 12.8% over the reasoner without distractors, so each role recovers between a third and a half of that gain. This is cross-task transfer, and both roles are substitutive.

How the effect appears. We ask where and how the roles affect the output, comparing the detector and the abstractor on the unseen problems with distractors and their clean twins. In the Reasoner without Distractors setting, most of the gain lies on problems the reasoner solves clean but fails once the distractor clause is added (Appendix J). A role changes how the clause is handled and leaves the arithmetic around it as it was. The reasoner and the role model agree step for step until they reach the distractor clause, and part only on whether its number enters the final computation. The change a role leaves in the model is small, and it shows in the output only at the distractor clause.

Both roles are additive; the abstractor adds far more. Once the reasoner is trained on problems with distractors, the two roles part ways in how much they add. The **Reasoner with Distractors** gains 18.5% over the base model at 4B and 12.9% at 8B. The detector adds about a tenth of the reasoner's own gain on top of this, 1.9% and 0.9% at 4B, and 1.6% and 2.3% at 8B, on unseen and seen problems, respectively. The abstractor adds another 6.1% at 4B, a third of the reasoner's own gain, and 8.4% at 8B, two thirds of it, with 5.1% and 5.7% on seen problems. Both roles are therefore additive, the detector by about a tenth of the reasoner's own gain and the abstractor by at least a third of it. We ask below why the two roles differ by this much, first by asking which problems each role repairs, and then by narrowing the abstractor's view toward the reasoner's.

Which problems each role repairs. We count a problem as repaired when no sample of the reasoner solves it and a majority of the role model's samples do, and as broken in the opposite case. With the reasoner trained on problems with distractors, the detector repairs 4 problems and breaks 4 at 4B, and repairs 2 and breaks 1 at 8B, so it leaves the set of solved problems about where it found it. The abstractor repairs 13 problems and breaks 1 at 4B, and repairs 12 and breaks none at 8B, and its repairs include 3 of the detector's 4 at 4B and both of the detector's 2 at 8B. The abstractor repairs many problems while breaking almost none. The detector's gain comes instead as robustness, raising the fraction of correct samples on problems the reasoner already solves some of the time, rather than as repair. This suggests that the reasoner extracts most of what the detector is rewarded for from the answer reward on problems with distractors on its own.

Narrowing the abstractor's view toward the reasoner's. We hypothesize that the abstractor's additional gain comes from operating across instances, therefore we narrow its view toward the reasoner. The results of this are shown in Table 2 which varies the two components of the abstractor's view, with the prompt, the reward function, and the budget unchanged. The prompt view is the set S of ℓ shown instances, one or three, and the reward view is either the shown instances themselves, $H = S$, or $h = 2$ held-out instances. With $\ell = 1$ and $H = S$, the abstractor reads the problem the reasoner reads and is scored on it, and differs from the reasoner only in being asked for a method rather than an answer. A method written from one instance and scored on that instance has to hold for nothing beyond it, so all the reward can select for is whether the method keeps the quantities the instance needs and drops its clause, the relation the detector is rewarded for, and this view transfers at the detector's level. Each component of the view matters on its own. Scoring on held-out instances raises accuracy on unseen problems with distractors at either ℓ, and showing three instances instead of one raises it under either reward view, the larger of the two effects. Accuracy rises through the four views, 77.2%, 78.9%, 80.3%, and 83.6%, and the widest, three shown and scored on held-out ones, transfers the most. The more instances the method has to hold across, the more the role adds. The gap between the two roles' additive gains is therefore a matter of view: an abstractor confined to the reasoner's view adds what the detector adds, and the abstractor of Table 1 adds more because its view reaches across instances.

Table 2: The abstractor's view varied at 8B, in the Reasoner with Distractors setting. The last row is the abstractor of Table 1.

Role	Prompt view	Reward view	Unseen	Seen
Reasoner	1	shown	75.2	78.8
+ Detector	1	shown	76.8	81.1
+ Abstractor	1	shown	77.2	81.8
	1	withheld	78.9	83.7
	3	shown	80.3	**85.6**
	3	withheld	**83.6**	84.5

Given these findings, we would expect the detector's gain to shrink further with an increase in model capability, as a stronger reasoner learns the relation on its own. At the same time, we expect the abstractor's gain to hold until accuracy on the benchmark saturates, though we do not test this.

5 Reading Comprehension across Sources: Understander

When learning from a book, humans often build an understanding of the material before answering questions about it. We reduce this to its smallest unit, a passage and its questions, and test it on reading comprehension. The understander role tests whether additive roles exist beyond GSM-Symbolic (Section 4), and without a known failure mode to define the role around. We train on multiple-choice reading comprehension from one source, RACE-C (Liang et al., 2019), and evaluate on RACE-C and, without further training, on five other comprehension datasets.

5.1 Setup and Role

We train on 50 RACE-C passages (266 questions) and report on the held-out RACE-C test split and on QuAIL (Rogers et al., 2020), CosmosQA (Huang et al., 2019), LSAT-RC and LSAT-LR (Wang et al., 2022), and QuALITY (Pang et al., 2022). Appendix F shows the dev curves with the deployed checkpoint marked, and Appendix A the splits and the transfer datasets. Models, optimization, budget and selection are those of Section 4.1.

Table 3: Reading comprehension. The model only trained on RACE-C. Accuracy (%) reported.

Role	RACE-C	QuAIL	CosmosQA	LSAT-RC	LSAT-LR	QuALITY	Mean
Qwen3-4B-Base							
Base model	72.1	62.6	60.6	62.3	53.9	58.4	61.6
Reasoner	83.1	74.6	75.9	71.8	62.9	71.7	73.4
+ Understander	**84.2**	**77.6**	**78.0**	**74.0**	**63.2**	**72.1**	**74.8**
Qwen3-8B-Base							
Base model	80.4	71.3	67.7	68.3	61.0	68.8	69.6
Reasoner	85.8	77.1	78.0	77.0	65.9	75.5	76.6
+ Understander	**86.5**	**80.0**	**79.5**	**77.6**	**66.8**	**76.5**	**77.8**

Across instances: UNDERSTANDER. Like the abstractor, the understander operates across instances, where each question about a passage is an instance. For an anchor question e on a passage with text $t(e)$, the view collects the passage and the set H of every question asked about it, $x_{\mathrm{und},\rho} = t(e)$ and $x_{\mathrm{und},r} = \{(t(e), c, a^\star(c)) : c \in H\}$, each question with its options and gold answer. The prompt renders the passage alone, with no question, and asks for what the passage implies. This statement is the artifact $u(y)$. The reward prompt κ_{und} presents the passage, a question, and the artifact, and asks for the answer without reasoning, so any inference has to be in the artifact. The reward is the policy's accuracy over the questions, $r_{\mathrm{und}}(x_{\mathrm{und},r}, y) = \mathrm{acc}_\theta(u(y); H)$ at $m = 4$ samples per question. Since the answering model also sees the passage, the reward measures what the artifact adds to the passage, and since it is scored on every question, the role is rewarded for what is useful across questions rather than for any one of them.

5.2 RESULTS AND DISCUSSION

The understander is additive. We report the results in Table 3. The understander raises accuracy on every dataset at both scales (standard errors in Appendix K). The mean gain ($+1.4\%$ and $+1.2\%$) is 12% of the reasoner's improvement over the base model at 4B and 17% at 8B. The reasoner trains on every question and the understander improves it further, so the understander is additive, on a task with no obvious failure mode to target.

The gain is larger beyond the training source than on it. At 4B, $+1.1\%$ on RACE-C against $+3.0\%$ on QuAIL and $+2.1\%$ on CosmosQA; at 8B, $+0.7\%$ on RACE-C against $+2.9\%$ and $+1.5\%$. The role does more than fit the training corpus. What it is rewarded for, structure shared by several questions about a passage, is shared across comprehension tasks, while the reasoner's view of one question at a time is specific to its source. At both scales, the largest gains are on QuAIL and CosmosQA, the two datasets closest to RACE-C in form, short narrative passages with multiple-choice questions. On the datasets further from that form, the logical-reasoning arguments of LSAT-LR and the far longer passages of QuALITY, the gains are about 1% or less. The gain carries best to tasks that share the structure the role was trained on, and it reaches beyond the dataset the role trains on.

Table 4: Budget split (reasoner:understander rollouts), Qwen3-4B-Base, panel mean.

Split	Mean
base	61.6
16:0	73.4
12:4	75.0
8:8	74.8
4:12	74.6
0:16	65.8

Narrowing the reward view. Following the view ablation on GSM-Symbolic (Table 2), we ask whether the same holds here: does the understander's gain come from being rewarded across questions? We test this at 4B with the prompt, reward function, and rollout budget unchanged. The understander still reads each passage without its questions, but its reward view is narrowed from every question about the passage to the anchor question alone, $H = e$. Thus, each understanding is rewarded only for the question solved by the reasoner in the same round. This removes the gain. The narrowed role reaches 72.6% on average, compared with 74.8% for the understander and 73.4% for the reasoner, and exceeds the reasoner only on RACE-C. The two roles therefore differ only in how many questions an understanding must serve. An understanding rewarded across every question about its passage transfers to comprehension datasets the model never trains on, whereas one

rewarded for a single question does not. As with the abstractor, the role adds signal through the instances its view spans beyond the reasoner's.

Budget split. Table 4 varies the split of the fixed 16-rollout budget. Anywhere between a quarter and three quarters of the budget on the role gives essentially the same answer ($+1.6\%$, $+1.4\%$, $+1.2\%$ on the panel mean), so the result does not depend on the ratio. We used an even split throughout without tuning it. When the reasoner is removed entirely, the model loses 7.6% to the reasoner alone, yet lands 4.2% above the base model. This understander only is the counterpart of the Reasoner without Distractors setting on GSM-Symbolic, and a stricter one, since here the model never trains toward an answer at all. The only gradient it receives comes from the understander's objective, so the understander is substitutive as well as additive.

6 Conclusion

Humans learn from little data, and part of how they do so is by practicing on the same material in more than one way (Gick & Holyoak, 1983; Catrambone, 1998). Language models, on the other hand, are data hungry by comparison, though recent work shows that a small number of examples can improve performance substantially during post-training (Ye et al., 2025). In this paper we asked whether auxiliary roles can help draw more signal from the same examples, even without test-time invocation, and formalized Role-Based Reinforcement Learning to test this. We find that the same examples hold more training signal than the primary role alone extracts. Within a fixed rollout budget and evaluated under the reasoner prompt alone, a model trained in several roles improves in the primary role, on arithmetic with distractors and on reading comprehension. In the domains we chose, the additional signal comes from the role's view. Narrowing the abstractor's view to the single instance the reasoner reads reduces its gain, and narrowing the understander's reward view to the one question the reasoner solves removes its improvement entirely. Since that structure is a property of how examples relate rather than of any one of them, we hypothesize that across-instance roles will add signal on datasets in general, wherever such relations can be found. Interestingly, we also find that the gain falls most on examples that differ from the training distribution. On GSM-Symbolic, the abstractor improves the reasoner more on problems from unseen templates than on seen ones, and the widest view yields the largest gain on unseen problems. On reading comprehension, the understander improves the reasoner more on the five transfer datasets than on RACE-C itself.

Future Works. In the domains we chose, roles whose view spans several instances show the largest gains, both within the training distribution and outside it. Although this holds for these two domains, we ask whether it is true in general. Where the difficulty of a dataset lies within an instance rather than across them, as in multi-step proofs, a within-instance role that names the intermediate results may add more than any across-instance one, and which kind of role a dataset rewards is itself a question the framework can ask. We also find every role we study to be both substitutive and additive; whether there exist roles that are additive but not substitutive, supplying something the reasoner can use only once it has the primary skill, is a question for future work, and if such roles exist the order in which roles train would matter. We also ask whether the view itself can be learned by the model. In our study, we exploit the views that come with the dataset, a template above its instances and a passage above its questions, yet most datasets carry no such structure. We believe that a retriever role can be trained to construct the view of a role that generalizes across instances, collecting for each anchor the examples that belong with it, so that the view function is learned rather than given. Similarly, one can define a meta-role, whose output is another role, its view, prompt, and reward. Self-adapting models already generate the data they then train on (Zweiger et al., 2025; Wang et al., 2026), which suggests that this loop is within reach.

Limitations. The models come from one family and two sizes, so we cannot say how transfer changes with scale beyond the trend between them, though we expect substitutive gains to persist as models improve, since a more capable model should make more of a signal it was never trained to answer from, and prior work finds cross-task transfer under RL in Llama-3.1-8B (Yuan et al., 2026). We show that transfer occurs and how it varies with the role, but not what an auxiliary role changes in the shared weights. The fixed budget counts trained rollouts only, and the rollouts that score an artifact are generated as well, so a model with a role may take more wall-clock time than the reasoner (Appendix I).

Reproducibility statement

Every model in the paper trains from a publicly available base model (Qwen3-4B-Base, Qwen3-8B-Base) on data we construct from public sources; Appendix A gives the construction of the GSM-Symbolic training pool, the external No-Op benchmark and its seen/unseen split, and the RACE-C and transfer-panel splits. Appendix B lists every hyperparameter, what was swept and on which dataset, and what was held fixed; Appendix C states the checkpoint-selection and evaluation protocol; Appendix E reproduces every prompt verbatim; Appendix I reports training cost per model. Code, configurations, data-construction scripts and the evaluation outputs behind every table will be released. During review, it has been submitted as supplementary material.

AI use statement

Generative AI tools were used in this work for drafting and editing the appendices, writing code, and literature search. The research ideas and contributions presented in this paper originated from and were formalized by the authors; AI assistants were used to support their implementation and exposition. The authors reviewed all AI-assisted work and take responsibility for the final content, including all text, claims, and artifacts produced with the aid of generative AI.

Ethics statement

The work trains and evaluates language models on arithmetic word problems and reading-comprehension questions from public datasets; it involves no human subjects, no personal data, and no application with foreseeable direct harm.

References

Rich Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997. doi: 10.1023/A:1007379606734.

Rich Caruana and Virginia R. de Sa. Promoting poor features to supervisors: Some inputs work better as outputs. In *Advances in Neural Information Processing Systems 9*, 1996.

Richard Catrambone. The subgoal learning model: Creating better examples so that students can solve novel problems. *Journal of Experimental Psychology: General*, 127(4):355–376, 1998. doi: 10.1037/0096-3445.127.4.355.

Yixing Chen, Yiding Wang, Siqi Zhu, Haofei Yu, Tao Feng, Muhan Zhang, Mostofa Patwary, and Jiaxuan You. Multi-agent evolve: LLM self-improve through co-evolution. *arXiv preprint arXiv:2510.23595*, 2025.

Mary L. Gick and Keith J. Holyoak. Schema induction and analogical transfer. *Cognitive Psychology*, 15(1):1–38, 1983.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645:633–638, 2025.

Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alexander Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. In *Findings of the Association for Computational Linguistics: ACL 2023*, 2023.

Maggie Huan, Yuetai Li, Tuney Zheng, Xiaoyu Xu, Seungone Kim, Minxin Du, Radha Poovendran, Graham Neubig, and Xiang Yue. Does math reasoning improve general LLM capabilities? understanding transferability of LLM reasoning. *arXiv preprint arXiv:2507.00432*, 2025.

Lifu Huang, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Cosmos QA: Machine reading comprehension with contextual commonsense reasoning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2019.

Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z. Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. In *International Conference on Learning Representations (ICLR)*, 2017.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. RACE: Large-scale ReAding comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2017.

Shiyang Li, Jianshu Chen, Yelong Shen, Zhiyu Chen, Xinlu Zhang, Zekun Li, Hong Wang, Jing Qian, Baolin Peng, Yi Mao, Wenhu Chen, and Xifeng Yan. Explanations from large language models make small reasoners better. *arXiv preprint arXiv:2210.06726*, 2022.

Yichan Liang, Jianheng Li, and Jian Yin. A new multi-choice reading comprehension dataset for curriculum learning. In *Proceedings of the 11th Asian Conference on Machine Learning (ACML)*, 2019.

Jianbo Lin, Xiaomin Yu, Yi Xin, Yifu Guo, Zhuosong Jiang, Zhongqi Yue, Weishi Wang, Heqing Zou, Chengwei Qin, and Hui Xiong. ICRL: Learning to internalize self-critique with reinforcement learning. *arXiv preprint arXiv:2605.15224*, 2026.

Bo Liu, Chuanyang Jin, Seungone Kim, Weizhe Yuan, Wenting Zhao, Ilia Kulikov, Xian Li, Sainbayar Sukhbaatar, Jack Lanchantin, and Jason Weston. SPICE: Self-play in corpus environments improves reasoning. *arXiv preprint arXiv:2510.24684*, 2025a.

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 *Conference on Language Modeling*, 2025b.

Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. GSM-Symbolic: Understanding the limitations of mathematical reasoning in large language models. In *International Conference on Learning Representations (ICLR)*, 2025.

Xuefei Ning, Zifu Wang, Shiyao Li, Zinan Lin, Peiran Yao, Tianyu Fu, Matthew B. Blaschko, Guohao Dai, Huazhong Yang, and Yu Wang. Can LLMs learn by teaching for better reasoning? a preliminary study. In *Advances in Neural Information Processing Systems 37*, 2024.

Richard Yuanzhe Pang, Alicia Parrish, Nitish Joshi, Nikita Nangia, Jason Phang, Angelica Chen, Vishakh Padmakumar, Johnny Ma, Jana Thompson, He He, and Samuel R. Bowman. QuALITY: Question answering with long input texts, yes! In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2022.

Yada Pruksachatkun, Jason Phang, Haokun Liu, Phu Mon Htut, Xiaoyi Zhang, Richard Yuanzhe Pang, Clara Vania, Katharina Kann, and Samuel R. Bowman. Intermediate-task transfer learning with pretrained language models: When and why does it work? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 5231–5247, 2020.

Yuxiao Qu, Anikait Singh, Yoonho Lee, Amrith Setlur, Ruslan Salakhutdinov, Chelsea Finn, and Aviral Kumar. RLAD: Training LLMs to discover abstractions for solving reasoning problems. In *International Conference on Learning Representations*, 2026.

Anna Rogers, Olga Kovaleva, Matthew Downey, and Anna Rumshisky. Getting closer to AI complete question answering: A set of prerequisite real tasks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

Vivswan Shah, Randy Cogill, Hanwei Yue, Gopinath Chennupati, and Rinat Khaziev. Regularization through reasoning: Systematic improvements in language model classification via explanation-enhanced fine-tuning. *arXiv preprint arXiv:2511.02044*, 2025.

Rulin Shao, Shuyue Stella Li, Rui Xin, et al. Spurious rewards: Rethinking training signals in RLVR. *arXiv preprint arXiv:2506.10947*, 2025.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H. Chi, Nathanael Schärli, and Denny Zhou. Large language models can be easily distracted by irrelevant context. In *International Conference on Machine Learning (ICML)*, 2023.

Trevor Standley, Amir Zamir, Dawn Chen, Leonidas Guibas, Jitendra Malik, and Silvio Savarese. Which tasks should be learned together in multi-task learning? In *International conference on machine learning*, pp. 9120–9132. PMLR, 2020.

Benjamin Sturgeon. Revisiting GSM-Symbolic: Do 2026 frontier models still fail at confounded grade school math? `https://www.benjaminsturgeon.com/inkhaven-day-8/`, 2026. Dataset: `https://huggingface.co/datasets/Experimental-Orange/gsm-noop-audited`.

Siyuan Wang, Zhongkun Liu, Wanjun Zhong, Ming Zhou, Zhongyu Wei, Zhumin Chen, and Nan Duan. From LSAT: The progress and challenges of complex reasoning. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30:2201–2216, 2022.

Zekun Wang, Anant Gupta, Zihan Dong, and Christopher J. MacLellan. Self-consolidating language models: Continual knowledge incorporation from context. *arXiv preprint arXiv:2605.07076*, 2026.

Yu Xia, Canwen Xu, Zhewei Yao, Julian McAuley, and Yuxiong He. Learning to hint for reinforcement learning. *arXiv preprint arXiv:2604.00698*, 2026.

Yixin Ye, Zhen Huang, Yang Xiao, Ethan Chern, Shijie Xia, and Pengfei Liu. LIMO: Less is more for reasoning. In *Second Conference on Language Modeling*, 2025. URL `https://openreview.net/forum?id=T2TZ0RY4Zk`.

Lifan Yuan, Weize Chen, Yuchen Zhang, Ganqu Cui, Hanbin Wang, Ziming You, Ning Ding, Zhiyuan Liu, Maosong Sun, and Hao Peng. From $f(x)$ and $g(x)$ to $f(g(x))$: LLMs learn new skills in RL by composing old ones. In *International Conference on Learning Representations*, 2026.

Amir R. Zamir, Alexander Sax, William Shen, Leonidas J. Guibas, Jitendra Malik, and Silvio Savarese. Taskonomy: Disentangling task transfer learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3712–3722, 2018.

Andrew Zhao, Yiran Wu, Yang Yue, Tong Wu, Quentin Xu, Yang Yue, Matthieu Lin, Shenzhi Wang, Qingyun Wu, Zilong Zheng, and Gao Huang. Absolute zero: Reinforced self-play reasoning with zero data. In *Advances in Neural Information Processing Systems 38*, 2025.

Adam Zweiger, Jyothish Pari, Han Guo, Yoon Kim, and Pulkit Agrawal. Self-adapting language models. In *Advances in Neural Information Processing Systems 38*, 2025.

A Datasets and Splits

A.1 GSM-Symbolic with distractor clauses

Templates. GSM-Symbolic releases its templates but not the generator, so we implemented the template language ourselves. The interpreter is certified rather than assumed correct ($242/250$ templates): the released annotated questions invert into variable assignments, and we re-derive each gold from the template's answer expression over the recovered assignment. This certifies $242/250$ templates (99.0% of questions inverted, 98.3% of golds re-derived); the failures are templates that disagree with their own released data, float-boundary cases, and unsamplable draws. We use the `p1` and `p2` configurations, the harder rewrites of the GSM8K seeds, and drop `main` as saturated. Templates whose base accuracy under an untrained Qwen3-4B-Base lies outside 5–90% are dropped: Dr. GRPO's advantage is zero in any group where every rollout agrees, so a template is uninformative at both ends. Screening keeps 84 of 144 total `p1` + `p2` templates provided by GSM-Symbolic (Mirzadeh et al., 2025); it also removes an accidental difficulty imbalance between splits. Clause authoring (below) yields verified clauses for 81 of the 84. Of these, we select 49 templates for our training set. We leave the remaining to allow a larger *unseen* pool of questions.

Clauses. The distractor clauses are not released with the templates. We author them with Qwen3.8-27B, prompted with each template's surface text, variable definitions and answer expression and asked for one clause of a given type; each clause is itself parameterised, so it is redrawn per base assignment rather than pasted. Correctness is checked structurally by a program, never judged by a model. A clause may declare only *fresh* variables; since the gold is the evaluation of the template's answer expression, an expression that never references those variables cannot change value, so the clause is provably no-op. We then execute the check, rendering both twins from the same draw and comparing golds on sampled assignments, rather than relying on the argument. Further gates reject name collisions, empty domains, unresolved placeholders, echoes of the authoring prompt, and clauses that state the gold. This yields verified clauses of three types: same domain and different entity, same entity and different time or scope, and sounds-operative-but-unused; the last is not used in the reported pool.

Training pool. 49 templates, 40 base assignments each (two templates carry 35 and 39). Every base is rendered once clean and once per available clause type, L2 for every template and L3 for 47 of them, giving $5{,}782$ instances, $1{,}954$ clean and $3{,}828$ with a distractor clause, with each clean instance and its twin with a distractor sharing their template, base assignment and gold. The Reasoner without Distractors models train the reasoner on the $1{,}954$ clean instances only; in every model the abstractor is scored on instances with distractors and the detector draws its inputs from the whole pool, clean and with distractors, so a third of its inputs carry no clause.

Benchmark. Accuracy is reported on an external audited No-Op set (Sturgeon, 2026), a third-party release in the style of the No-Op variant described by Mirzadeh et al. (2025): GSM8K problems extended with distractor clauses, written and audited independently of this work by two auditors labelling each clause truly irrelevant, ambiguous or actually relevant. We keep only the truly-irrelevant rows of the stronger auditor (338 of 945) and emit each as a pair, the clean problem and its twin with the distractor, sharing a gold. Every item retains its original GSM8K identifier, so the set splits by whether that problem also backs one of our training templates: 204 pairs over 50 identifiers that it does not (*unseen*) and 134 pairs over 34 that it does (*seen*). The build asserts that no trained identifier appears in the unseen file. Re-scoring every reported checkpoint on all 945 candidate pairs lowers absolute accuracy (unaudited golds can be wrong) but leaves every contrast in the paper in place.

Selection split. 429 held-out instances (145 clean, 284 with a distractor) of the training templates: same templates, base assignments disjoint from training, zero instance overlap with training or with the benchmark.

A.2 Reading comprehension

Training. 50 RACE-C passages (Liang et al., 2019) carrying 266 questions, drawn at the passage level. RACE-C is the college-level extension of RACE (Lai et al., 2017).

In-domain evaluation. The remaining RACE-C pool is split at the passage level into a development set of 150 passages (777 questions) and a test set of 271 passages (1,420 questions). Selection uses the development set; all reported in-domain numbers are on the test set.

Transfer panel. The deployed checkpoint is evaluated without further training on five further multiple-choice comprehension datasets:

dataset	passages	questions
QuAIL	120	720
CosmosQA	2985	2985
LSAT-RC	80	539
LSAT-LR	1010	1016
QuALITY	115	2086

These span short passages with commonsense questions (CosmosQA), mid-length narrative and expository passages (QuAIL), analytical and logical reasoning (LSAT-RC, LSAT-LR), and long-document comprehension (QuALITY, mean passage length roughly an order of magnitude above RACE-C). The training distribution is RACE-C alone, so every column but the first is a transfer result.

B HYPERPARAMETERS

Swept. Learning rate × gradient-norm clip × KL coefficient, run at each model size for the reasoner-only model and for the one-role model, selecting on development accuracy on RACE-C; the two searches converged on the same values. The selected values were then used unchanged for every dataset and every model.

	swept values	selected
learning rate	5×10^{-6} – 1.6×10^{-4} (size-dependent)	2×10^{-5} (4B, 8B)
gradient-norm clip	0.5, 1.0	0.5
KL coefficient β	0, 0.02	0.02

Held fixed everywhere. LoRA rank 64, α 256, dropout 0.05 on the q, k, v, o, gate, up and down projections; Dr. GRPO with per-role group baselines; constant learning-rate schedule; batch size 128; rollout batch 8; two inner epochs per batch; training sampling temperature 1.0, top-p 0.95; 16 trained rollouts per prompt; 768-token limit on training generations for every role.

C SELECTION AND EVALUATION PROTOCOL

Checkpoint selection. Every run trains for a fixed horizon and saves a checkpoint every 4 (RACE-C), 8 (GSM-Symbolic at 4B), or 14 (GSM-Symbolic at 8B) optimizer steps. One checkpoint per run is deployed: the checkpoint with the highest development accuracy, where the development split is disjoint from every reported benchmark (Appendix A). On GSM-Symbolic, the selection statistic is accuracy on the 284 held-out instances containing a distractor; on RACE-C, it is accuracy on the 150-passage development split. The selection rule is fixed in advance and applied identically to every model; no model is evaluated at a checkpoint selected using the benchmark on which it is subsequently reported. On GSM-Symbloic, all 4B runs train for 541 steps and all 8B runs for 847 steps, horizons that cover the peak dev performance of all roles. On RACE-C, all runs were trained for 298 total steps since we use a much smaller dataset and the models converged faster.

Evaluation. Every reported number is avg@8: eight samples per question at temperature 0.6, top-p 0.95, a 1024-token limit and evaluation seed 123, scored by exact match on the boxed answer. Every model, including every model trained with a role, is evaluated under the reasoner prompt alone (Appendix E); no artifact is generated or supplied at evaluation.

Peak-then-decline. Dev-argmax reads each model at its best, and the baselines are trained well past that point. The GSM-Symbolic reasoner-only baseline peaks at step 432 of 541 at 4B and 602 of 847 at 8B and averages 3.9% and 3.0% below its peak thereafter. The baseline's ceiling is therefore

a property of its objective rather than of insufficient training, which supports the comparisons of Sections 4.3 and 5.2.

D TRAINING PROCEDURE

Algorithm 1 gives one training step of RBRL. For each example in the batch, every role with a non-zero allocation builds its view, samples its n_k rollouts, scores them, and forms advantages against its own group mean; the rollouts of all roles are assembled into one batch and a single Dr. GRPO update is taken on it. The reasoner is a role like any other, with $\phi_{\text{pri}}(e) = e$ and exact-match reward, and the reasoner-only baseline is the allocation $n_{\text{pri}} = N$. Any samples a reward function draws from π_θ (Equation 4) are used inside r_k and never enter $\mathcal{B}$.

Algorithm 1: Role-Based Reinforcement Learning, one training step

Require: policy π_θ; dataset $\mathcal{D}$; roles $\{(\phi_k, \rho_k, r_k)\}_{k \in \mathcal{K}}$; allocation $\{n_k\}$ with $n_k \geq 0$ and $\sum_k n_k = N$; batch size B

Sample a batch of examples $E \sim \mathcal{D}$ with $|E| = B$;
$\mathcal{B} \leftarrow \varnothing$
foreach $e \in E$ ***and*** *role* k *with* $n_k > 0$ **do**
  $x_{k,\rho}, x_{k,r} \leftarrow \phi_k(e; \mathcal{D})$ `// the role's view; may collect siblings of` e
  $y^{(1)}, \ldots, y^{(n_k)} \sim \pi_\theta(\cdot \mid \rho_k(x_{k,\rho}))$ `// the role's` n_k `trained rollouts`
  $R^{(i)} \leftarrow r_k(x_{k,r}, y^{(i)})$ for $i = 1, \ldots, n_k$ `// may sample` π_θ `again (Eq. 4); those samples are discarded`
  $\hat{A}^{(i)} \leftarrow R^{(i)} - \frac{1}{n_k} \sum_{j=1}^{n_k} R^{(j)}$ `// group baseline within the role`
  $\mathcal{B} \leftarrow \mathcal{B} \cup \{(\rho_k(x_{k,\rho}),\, y^{(i)},\, \hat{A}^{(i)})\}_{i=1}^{n_k}$
Update θ with Dr. GRPO on $\mathcal{B}$ `// one update, every role`

E PROMPTS

Here, we present every prompt used by a reported model, verbatim. Braces mark the slot a field is inserted into. All models are base models, so prompts are plain completions with no chat template.

E.1 GSM-SYMBOLIC

Reasoner (training and evaluation)

```
Solve the math word problem.  Think step-by-step, then give the final
answer as exactly one boxed number, e.g.  \boxed{42}.

Problem:
{question}

Reason step by step, then output \boxed{<final number>}.
```

Abstractor (view: ℓ instances of one template)

```
Below are {ℓ} versions of the same underlying math problem.  The
numbers, names, and objects differ; the structure and solution
procedure are the same.  IMPORTANT: some versions contain an extra
sentence that looks relevant but is NOT used to get the answer -- a
distractor.

Problem 1:
{instance 1}

...
Problem {ℓ}:
{instance ℓ}

Compare the versions.  A quantity that MATTERS recurs across versions
playing the same role; a distractor sentence appears in only some
```

```
versions, describes a different person, time, or scope, or never
enters the computation.  Then write a METHOD for solving ANY version
of this problem:  refer to quantities by what they mean (e.g.  "the
number of puppies with spots"), never by their specific values.  State
which given quantities matter, what to compute from them and in what
order, and state succinctly what to ignore -- as a GENERAL rule
describing the KIND of irrelevant information (e.g.  "any sentence
about a person other than the subject"), not the specific sentences
in these versions.  Write the method itself (not an example) between a
<method> tag and a </method> tag.
```

Abstractor downstream template κ_{abs} (one withheld instance; reward only)

```
Use the method below to solve the problem.
Method:
{method}
Problem:
{question}
Think step-by-step following the method, then give the final answer as
one boxed number, e.g.  \boxed{42}.
```

Detector (view: the example itself)

```
Below is a math word problem.  It may or may not contain an extra
sentence that looks relevant but is NOT needed to answer the question
-- a distractor.
Problem:
{question}
Decide whether this problem contains such an irrelevant sentence.  A
sentence that MATTERS is needed to compute the answer; a distractor
states a quantity the question never uses, or describes a different
person, place, time, or object than the question asks about.
Do NOT solve the problem and do NOT state the answer.
Reason briefly, then give your verdict as exactly one of \boxed{yes} or
\boxed{no}.
If the verdict is \boxed{yes}, then after it write a WARNING listing
the irrelevant sentence or sentences, one per line, each line starting
with a dash.
COPY EACH SENTENCE EXACTLY as it appears in the problem above --
word for word, with the same numbers, names and punctuation.  Do NOT
paraphrase it, shorten it, summarise it, or describe it.  A line that
is not an exact copy is worthless.
Write the warning between a <warning> tag and a </warning> tag.
```

Detector downstream fallback when a quoted span cannot be located verbatim (reward only)

```
Solve the math word problem.  Think step-by-step, then give the final
answer as exactly one boxed number, e.g.  \boxed{42}.
Note:  the sentence or sentences below appear in this problem but are
NOT needed for the answer -- ignore them:
{warning}
Problem:
{question}
Reason step by step, then output \boxed{<final number>}.
```

The detector's downstream template κ_{det} has no prompt of its own: it deletes every verbatim-quoted sentence from the problem and issues the reasoner prompt on the result, using the fallback above when a quote cannot be located verbatim and the plain reasoner prompt when the verdict is no.

The two prompt controls of Table 10 draw each rollout's prompt from a pool of two: the reasoner prompt above and one of the following. Each is the role's own instruction folded into an answer-

scored solve, with its wording taken from the role prompt above it so that the two models differ only in where the work happens and what it is scored on.

Prompt control, detector instruction (second prompt; the first is the reasoner prompt)

```
Solve the math word problem.  It may or may not contain an extra
sentence that looks relevant but is NOT needed to answer the question
-- a distractor.  A sentence that MATTERS is needed to compute the
answer; a distractor states a quantity the question never uses, or
describes a different person, place, time, or object than the question
asks about.
Problem:
{question}
First say whether the problem contains such an irrelevant sentence,
and if it does, name it.  Then ignore it and solve the problem step by
step using only the sentences that matter.
Reason step by step, then output \boxed{<final number>}.
```

Prompt control, abstractor instruction (second prompt; the first is the reasoner prompt)

```
Solve the math word problem.  Some sentences may look relevant but are
NOT needed to get the answer.
Problem:
{question}
First write a METHOD for solving ANY version of this kind of problem:
refer to quantities by what they mean, never by their specific values;
state which given quantities matter, what to compute from them and in
what order, and what kind of information to ignore.  Then apply the
method to this problem step by step.
Reason step by step, then output \boxed{<final number>}.
```

E.2 READING COMPREHENSION

Reasoner (training and evaluation, every dataset)

```
Solve the multiple-choice question using the passage.
Passage:
{passage}
Question:
{question}
Options:
A. {option A}
B. {option B}
...
Think step-by-step and return your final answer as exactly one boxed
letter (e.g., \boxed{A}, \boxed{B}, etc.).
```

Understander (view: the passage, no questions)

```
Read the passage and write ONLY the interpretive conclusions a reader
needs that are NOT stated outright.  Do NOT summarize, restate, or
describe the passage or its structure -- the reader already has the
passage in front of them.  Be brief:  a few sharp lines, not prose, no
headers.
State only:
- the key INFERENCES the passage forces but never says (what must be
true; what the author would agree or disagree with)
- the author's ATTITUDE/stance and the specific evidence that fixes it
- the intended meaning of any pivotal/figurative/unusual word AS USED
HERE, and the referent of any genuinely ambiguous pronoun
- any single decisive contrast, cause-effect link, exception, or
```

```
qualification a question could hinge on
Commit to conclusions; skip anything already explicit in the
text.  Wrap your understanding strictly inside <understanding> and
</understanding> tags.
Passage:
{passage}
```

Understander downstream template κ_{und} (one withheld question; reward only)

```
Use the provided passage and understanding to solve the
multiple-choice question.
Passage:
{passage}
Understanding:
{understanding}
Question:
{question}
Options:
A. {option A}
...
Using only the information above, output your final choice as exactly
one boxed letter (e.g., \boxed{A}, \boxed{B}, etc.).  Do not explain or
show any reasoning.  After the boxed letter, terminate the response.
```

The understander's downstream solver answers directly, with no chain of thought and a 16-token limit, so the artifact rather than the solver's own reasoning has to carry the inference. An understanding cut off by the 768-token training limit receives zero reward.

F Additional Results

F.1 GSM-Symbolic: Detector downstream template

The detector's downstream template κ_{det} deletes the quoted clause from the problem and issues the plain reasoner prompt on the result; when the quote is verbatim this reproduces the clean twin. If a quoted span cannot be located verbatim, the clause is instead prepended to the reasoner prompt as material to ignore, since deleting a guessed span destroys information rather than mis-describing it.

F.2 GSM-Symbolic: Repairs and breaks under other definitions

Section 4 counts a problem as repaired when no sample of the reasoner solves it and a majority of the role model's samples do, and as broken in the reverse case. The threshold matters, because a role can move a problem by one sample or by all eight, and a definition that credits any movement counts noise while one that demands unanimity counts almost nothing. We therefore repeat the count under four other definitions in Table 5, from the loosest, where one sample on either side suffices, through a majority threshold and a shift of at least four samples, to unanimity, where all eight samples must agree. We report both settings at both sizes, and we include the model trained with both auxiliary roles. Every count uses the 8 samples per problem of the reported models on the 204 unseen problems with distractors, and we count a detector repair that the abstractor also repairs as shared.

In the Reasoner with Distractors setting, three things hold across the definitions. First, every role breaks problems as well as repairing them. The detector breaks about as many as it repairs once a repair has to reach a majority of samples, 4 against 4 at 4B and 1 against 2 at 8B. Second, the abstractor's net gain in problems exceeds the detector's under every definition, and the abstractor is the only role whose repairs survive the strictest definition short of unanimity, 13 repairs against 1 break at 4B and 12 against none at 8B. Third, the share of the detector's repairs that the abstractor covers depends on the definition. It reads 10 of 11 and 9 of 11 under the loosest definition and 7 of 10 and 5 of 9 under the majority definition, and the strict definition leaves counts too small to separate the two roles further. The model trained with both roles repairs mostly problems the abstractor

Table 5: Problems repaired and broken by each role on the 204 unseen GSM-NoOp problems with distractors, under definitions of increasing strictness. r and a are the samples, out of 8, that solve a problem for the reasoner and for the role model; a repair satisfies the stated condition and a break is the reverse. Cells read repaired / broken. **Shared** counts the detector's repairs that the abstractor also repairs; **Both** is the model trained with both auxiliary roles (Appendix G). † is the definition of Section 4.

Definition	Qwen3-4B-Base Detector	Abstractor	Shared	Both	Qwen3-8B-Base Detector	Abstractor	Shared	Both
Reasoner with Distractors								
Any sample (r=0, $a\geq$1)	11 / 7	20 / 4	10	22 / 1	11 / 4	20 / 2	9	18 / 1
Majority ($r<4$, $a\geq4$)	10 / 6	21 / 9	7	16 / 6	9 / 5	20 / 5	5	20 / 4
Shift of four ($a-r\geq4$)	7 / 5	19 / 7	5	14 / 4	6 / 3	20 / 2	3	18 / 3
Majority from none (r=0, $a\geq4$)†	4 / 4	13 / 1	3	9 / 0	2 / 1	12 / 0	2	9 / 0
Unanimity (r=0, a=8)	1 / 0	1 / 0	0	1 / 0	0 / 0	1 / 0	0	1 / 0
Reasoner without Distractors								
Any sample (r=0, $a\geq$1)	22 / 2	17 / 4	13	14 / 6	14 / 5	15 / 2	8	15 / 4
Majority ($r<4$, $a\geq4$)	21 / 6	18 / 10	14	21 / 10	17 / 8	11 / 1	7	17 / 5
Shift of four ($a-r\geq4$)	15 / 3	17 / 3	9	21 / 6	14 / 4	8 / 1	5	14 / 2
Majority from none (r=0, $a\geq4$)†	4 / 0	2 / 0	2	5 / 1	6 / 1	1 / 0	0	3 / 0
Unanimity (r=0, a=8)	0 / 0	0 / 0	0	0 / 0	0 / 0	0 / 0	0	0 / 0

repairs, under every definition, and it breaks at most one problem under the loosest definition and none under the strict one.

In the Reasoner without Distractors setting, the detector repairs as many problems as the abstractor or more under most definitions, in line with Table 1, where it is at least as strong as the abstractor in that setting. Under unanimity, where a problem must move from never solved to always solved or the reverse, at most one problem moves for any role in the Reasoner with Distractors setting and none in the other. Eight samples rarely agree on both sides of a change.

F.3 Reading Comprehension: Development curves of the reported runs

Figure 2 shows the RACE-C development accuracy of every reported run in Table 3, unsmoothed, with the checkpoint that was deployed (the dev argmax) marked. Selection never looks at a reported benchmark.

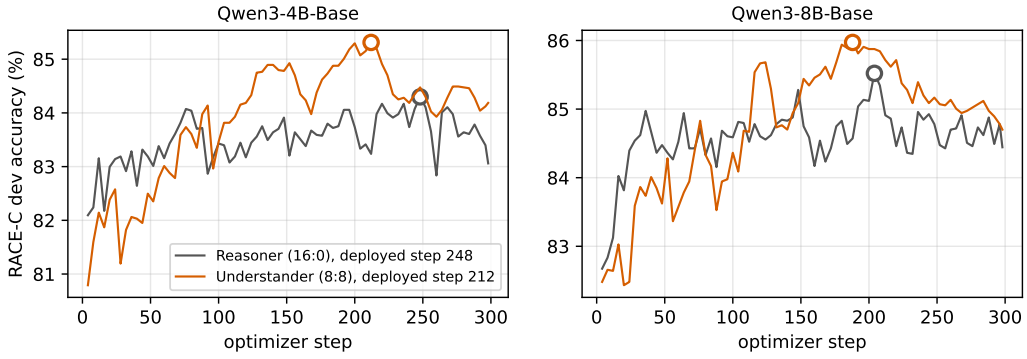

Figure 2: RACE-C development accuracy by optimizer step for the reported Reasoner and Understander runs at both model sizes. Circles mark the deployed checkpoint; the curves are not smoothed.

F.4 Reading Comprehension: More from the same data

Table 6 trains the Reasoner on twice the data, 100 passages (503 questions). Doubling the corpus improves the RACE-C column by +0.4 and lowers the panel mean by 0.5, with QuAIL, CosmosQA

and LSAT-LR down by 1.3, 0.8 and 2.3: more of one corpus fits that corpus. The UNDERSTANDER trained on 50 passages is ahead of the 100-passage REASONER on every dataset and by +1.9 on the mean. This is the pattern of Section 5: the reasoner's objective attaches to the training corpus, and the role's does not. The 100-passage run trains for 559 steps against 298; reading it at the dev argmax within 256 steps compares the models at one optimizer horizon, and its development accuracy at the uncapped argmax (step 448) is within 0.1% of the capped one.

Table 6: Training-set size, Qwen3-4B-Base: 50 passages (266 questions) versus 100 (503).

Train set	Role	RACE-C	QuAIL	CosmosQA	LSAT-RC	LSAT-LR	QuALITY	Mean
50 passages	Reasoner	83.1	74.6	75.9	71.8	62.9	71.7	73.4
50 passages	Understander	**84.2**	**77.6**	**78.0**	**74.0**	**63.2**	**72.1**	**74.8**
100 passages	Reasoner	83.5	73.3	75.1	73.3	60.6	71.9	72.9

F.5 READING COMPREHENSION: A FROZEN SCORING POLICY

The Understander's reward is the accuracy of the current model itself on withheld questions, so the model that scores the artifact improves as training proceeds. We compare this current-model scorer with two stationary alternatives: a frozen base model and a model snapshot refreshed every eight rounds. As shown in Table 7, all three scorers produce nearly identical results: the frozen and lagged scorers reach 74.0 and 74.7 on the panel mean, compared with 74.8 for the current-model scorer, while all three remain within about 1% of one another on the five transfer datasets. This matches the grouping used for the reward: all rollouts of a passage are scored by one model at a given step, so the scorer's level cancels in the group baseline, while its drift across steps only rescales advantages. We therefore use the current model as the scorer throughout.

Table 7: Scorer stationarity, Qwen3-4B-Base. The Understander's artifact is scored either by the frozen base model, or a model snapshot refreshed every eight rounds (lagged).

Scorer	RACE-C	QuAIL	CosmosQA	LSAT-RC	LSAT-LR	QuALITY	Mean
Frozen	83.8	77.2	77.1	73.6	61.6	71.0	74.0
Lagged	84.3	77.3	77.7	74.0	62.9	71.9	74.7

G TWO AUXILIARY ROLES IN ONE MODEL

We also train models with both auxiliary roles, in which the reasoner gets 8 rollouts while the detector and the abstractor get 4 each (8:4:4), so every model still trains 16 rollouts per example. Table 8 reports both settings of Section 4 at both scales, next to each role alone. Standard errors are in Appendix K and training times in Appendix I.

In both settings, the model trained with both auxiliary roles follows the abstractor within a few percent on unseen problems with distractors, 63.8% against 63.1% and 70.4% against 68.0% without distractors, and 78.3% against 78.7% and 81.4% against 83.6% with them, so the detector adds little to the abstractor as well. With the reasoner trained on problems with distractors, it repairs 9 problems and breaks none at each size under the definition of Section 4.3, and 8 of the 9 are problems the abstractor repairs, at both sizes. Appendix F.2 gives the counts under other definitions.

H ALTERNATIVE EXPLANATIONS RULED OUT

The gains of Section 4 admit other plausible explanations, each of which we found unlikely and checked regardless. Every check is at 4B in the Reasoner with Distractors setting, where the reasoner trains on problems with distractors and the abstractor adds +6.1 and +5.1 on top of it (Table 1); the models below are in Table 9, with standard errors in Appendix K.

The prompt, not the objective. The role's instruction, folded into the reasoner's own prompt, might carry the gain by itself. We give the reasoner a second prompt, drawn per rollout, that folds a role's

Table 8: A model with both auxiliary roles (8:4:4, eight reasoner rollouts and four per auxiliary role) next to each role alone (8:8), GSM-NoOp accuracy (%) in both settings of Table 1. Every model trains 16 rollouts per example.

Training Conditions	**Reasoner without Distractors**				**Reasoner with Distractors**			
Testing Conditions	**Clean**		**With distractor**		**Clean**		**With distractor**	
	Unseen	**Seen**	**Unseen**	**Seen**	**Unseen**	**Seen**	**Unseen**	**Seen**
Qwen3-4B-Base								
Reasoner	94.7	93.6	58.3	63.1	95.4	93.3	72.6	75.3
+ Detector	92.2	92.3	**65.4**	68.2	**95.5**	93.7	74.5	76.2
+ Abstractor	**95.1**	**93.8**	63.1	**69.2**	93.9	**94.6**	**78.7**	80.4
+ Detector + Abstractor	92.7	93.6	63.8	67.2	94.7	93.7	78.3	**80.6**
Qwen3-8B-Base								
Reasoner	96.1	93.5	62.4	67.7	**96.0**	94.5	75.2	78.8
+ Detector	95.5	96.2	68.4	74.3	93.9	95.5	76.8	81.1
+ Abstractor	95.7	**97.1**	68.0	74.1	**96.0**	95.2	**83.6**	84.5
+ Detector + Abstractor	**96.2**	96.5	**70.4**	**76.1**	95.6	**96.3**	81.4	**85.7**

Table 9: Alternative explanations, Qwen3-4B-Base, setting with distractors, mean ± per-draw standard error: the reasoner at three rollout budgets, the abstractor model, and the abstractor model with its rewards permuted within each group.

Model	**Clean**		**With distractor**	
	Unseen	**Seen**	**Unseen**	**Seen**
Reasoner, 8 rollouts	95.5 ±0.3	94.5 ±0.2	70.7 ±0.8	75.1 ±0.8
Reasoner, 16 rollouts	95.4 ±0.4	93.3 ±0.4	72.6 ±0.4	75.3 ±0.7
Reasoner, 32 rollouts	95.5 ±0.4	91.0 ±0.4	71.6 ±0.7	74.4 ±0.8
+ Abstractor (8:8)	93.9 ±0.5	94.6 ±0.3	78.7 ±0.5	80.4 ±0.7
+ Abstractor, rewards shuffled	95.3 ±0.4	91.2 ±0.4	66.8 ±0.8	67.8 ±0.7

instruction into the solve, asking it to name the distractor or to write a method before answering, and keep the reasoner's view, reward, parser and budget (Table 10; both control prompts are reproduced beside the role prompts in Appendix E). For the detector, which shares the reasoner's view, the control is exact. The only remaining difference is whether the judgement is an instruction inside an answer-scored generation or a separate rollout scored on whether the instance carries a clause. The detector's instruction gives -2.6 on unseen problems with distractors and $+0.5$ on seen, against $+1.9$ and $+0.9$ for the role; the abstractor's gives -1.4 and -5.5 against $+6.1$ and $+5.1$. The abstractor's instruction cannot show the reasoner the instances its reward is scored on, so the gain comes from the objective and its view of the data. The control forms a separate Dr. GRPO group for each prompt variant, so that prompt identity is never part of an advantage. Pooling the two variants into one group is the naive implementation and does worse still, 69.4 on unseen problems with distractors (-3.2) and 71.5 on seen (-3.8), with training reward falling over the run and a dev-argmax at step 104 of 540. Either way the control is null.

A second objective as a regularizer. An auxiliary role might help not through what it rewards but by perturbing the shared weights with a second gradient of any kind, as random rewards (Shao et al., 2025) or random explanation tokens (Shah et al., 2025) can. We train the abstractor model with the abstractor's rewards permuted at random within each example's group before advantages are computed. The role's rollouts receive the same rewards in the same proportions, so the update keeps its magnitude, but the reward no longer says which rollout was better. Selected by the same rule, at its dev-argmax at the last checkpoint of the run, this model reaches 66.8 on unseen problems with distractors and 67.8 on seen, against 78.7 and 80.4 for the abstractor and 72.6 and 75.3 for the reasoner alone. A random reward does not merely remove the gain, it costs the reasoner about 6%, so what transfers is what the abstractor's reward selects, not the presence of a second objective.

More reasoning rollouts. The abstractor model trains the reasoner on eight rollouts per example and is compared with a reasoner on sixteen, so the reasoner might simply be at the wrong budget.

Table 10: Prompt-only controls, Qwen3-4B-Base, problems with distractors, mean ± per-draw standard error: the reasoner keeps its view, reward and budget and adds the role's instruction to its prompt.

Role	**Unseen**	**Seen**
Reasoner	72.6 ±0.4	75.3 ±0.7
+ Detector instruction	70.0 ±0.7	75.8 ±0.6
+ Abstractor instruction	71.2 ±1.2	69.8 ±0.8

Trained with 8, 16 and 32 rollouts per example, everything else unchanged, the reasoner reaches 70.7, 72.6 and 71.6 on unseen problems with distractors and 75.1, 75.3 and 74.4 on seen ones, each at its own dev-selected step. Doubling or halving its rollouts does not move it, and the abstractor model is ahead with half of them.

I COMPUTE

Training runs on one H200 per run. Wall-clock is the training loop only (job startup and vLLM initialisation excluded) for RACE-C, and the total of all Slurm jobs a run occupied for GSM-Symbolic, where several runs were resumed once after a walltime limit. Every model trains 16 rollouts per prompt; models trained with a role additionally spend the downstream solves that score each artifact ($m \times |C_k(e)|$ per auxiliary rollout), which is where their extra time goes.

setting	model	4B (h)	8B (h)
RACE-C	Reasoner (16:0)	4.1	5.6
RACE-C	Understander (8:8)	6.6	9.5
RACE-C	Understander (12:4)	5.6	7.6
GSM	Reasoner, without distractors	10.3	21.9
GSM	+ Abstractor	13.2	26.7
GSM	+ Detector	9.6	19.0
GSM	+ both (8:4:4)	10.9	22.5
GSM	Reasoner, with distractors	10.7	19.9
GSM	+ Abstractor	13.0	23.5
GSM	+ Detector	9.7	16.0
GSM	+ both (8:4:4)	10.6	22.6
GSM	+ detector instruction (Table 10)	6.8	–
GSM	+ abstractor instruction (Table 10)	7.6	–

J QUALITATIVE EXAMPLES

The roles barely change the surface of a generation. On the unseen benchmark problems with distractors, explicit dismissal of the clause rises only from 11% of the generations of the 8B reasoner without distractors to 15% under the DETECTOR and 13% under the ABSTRACTOR; the typical divergence is silent, both models deriving the same quantities and differing only in whether the clause's number enters the final computation. The two items below show it.

J.1 A DETECTOR-MODEL DIVERGENCE ON GSM-NOOP

The item below is from the unseen benchmark, with its distractor; the clean twin is solved by every model. The inserted clause is in bold. Both generations are from the Reasoner without Distractors setting at 4B (reasoner trained on clean problems only, with and without the DETECTOR role), one sample each at the evaluation temperature; the two generations share their first five steps almost verbatim and diverge only in the last.

Problem (gold 100)

Mei qualified for a spot on the swimming team, so she went shopping for some athletic gear. She bought a jersey for $22, a pair of swimming leggings for $9 more than the jersey cost, and a pair of cleats that were originally $78 but were on sale for two-fourths price. She had a coupon for $4 off the package of $12 athletic headbands that she also bought. **She also received a $15 gift card from her coach for making the team, which she is saving to use on a future purchase.** How much did she spend on athletic gear?

Reasoner, clean data only: answers 85

[Steps 1--5: jersey $22; leggings $22 + $9 = $31; cleats $78 × 1/2 = $39; headbands $12 − $4 = $8; total $22 + $31 + $39 + $8 = $100.]
Step 6: Subtract the gift card. Mei received a $15 gift card, which she saved to use on a future purchase. So, the amount she actually spent is: $100 − $15 = $85.
Final Answer: \boxed{85}

Reasoner, clean data only, + Detector: answers 100

[Steps 1--5 identical.]
Step 6: Gift card. Mei received a $15 gift card, which she is saving for future use. This does not affect the total amount she spent on athletic gear.
Final Answer: Mei spent $100 on athletic gear. \boxed{100}

Both models give the gift card its own step under the same heading; one subtracts and one declines to. Across the benchmark this is the modal failure of the baselines: the answer equals the gold combined with the distractor's number by a single operation, overwhelmingly subtraction.

J.2 An abstractor-model divergence on GSM-NoOp

The same pattern under the Abstractor, from the Reasoner without Distractors setting at 8B, on an unseen problem with a distractor. Both generations derive the same per-pickaxe rates (wooden 5, silver 10, mithril 16 pounds per day) and diverge at the clause.

Problem (gold 9920)

One gnome can mine 5 pounds of ore per day with wooden pickaxe. He can mine two times as much with a silver pickaxe and 60% more with a mithril pickaxe than with a silver pickaxe. How many pounds of ore can 20 gnomes with mithril pickaxes mine in a month with 31 days, **given that 5 of the gnomes previously used wooden pickaxes before upgrading to mithril**?

Reasoner, clean data only: answers 7440

Step 2: Determine the number of gnomes using mithril pickaxes. Total gnomes: 20. Gnomes previously using wooden pickaxes: 5. Gnomes upgraded to mithril pickaxes: $20 - 5 = 15$.
Step 3: Calculate the total ore mined by gnomes with mithril pickaxes in a month. Ore mined per gnome per day: 16 pounds. Number of gnomes with mithril pickaxes: 15. Number of days in a month: 31. Total ore mined: $15 \times 16 \times 31$.
Step 4: $15 \times 16 = 240$. $240 \times 31 = 7440$.
Final Answer: \boxed{7440}

Reasoner, clean data only, + Abstractor: answers 9920

Step 4: Calculate the total ore mined by 20 gnomes with mithril pickaxes in one day. If 20 gnomes use mithril pickaxes, the total ore mined per day is $20 \times 16 = 320$ pounds.

```
Step 5:  In 31 days, the total ore mined is 320 × 31 = 9920 pounds.
Step 6:  Account for the 5 gnomes who previously used wooden pickaxes.
The problem states that 5 of the gnomes previously used wooden
pickaxes before upgrading to mithril.  However, since all 20 gnomes
are now using mithril pickaxes, this information does not affect the
calculation.  Thus, the total ore mined remains 9920 pounds.
Final Answer:  \boxed{9920}
```

The clause is read by both models; one converts it into a subtraction of gnomes and one gives it a step and sets it aside. Section 4.3 locates the gain across the benchmark.

J.3 Understandings written by the deployed 8B understander

Verbatim artifacts from the RACE-C-trained 8B Understander under its training prompt, on CosmosQA passages (an unseen transfer target, chosen for its short passages). The artifact is never produced at evaluation; these show what the role was trained to write.

Passage

```
HOLY GUACAMOLE GUYS! I had my fingers on the trigger the day WAMU was
acquired and their shares dropped under $0.20 to buy a couple thousand
worth.  I was THIS close to doing it.  Got cold feet at the end and
did n't go through.
```

Understanding

```
- The author is expressing regret for not purchasing shares of WAMU
when they were undervalued.
- The author's attitude is one of missed opportunity and hesitation.
- The word ``trigger'' is used figuratively here to mean the point of
action or decision.
- The pronoun ``it'' in ``I was THIS close to doing it'' refers to
purchasing the shares.
```

The question asked of this passage is what might have happened had the author bought the shares (gold: they could now sell them for a great deal of money); the first two lines already entail it.

Passage

```
Oh Rufus how could you?!  I know its my fault for leaving the 5 pound
bag of dog treats on the counter and leaving the house, but did you
really have to eat them all while I was gone?!  [...]
```

Understanding

```
- The passage implies that Rufus's actions are inexcusable despite the
author's own mistake.
- The author's attitude is one of frustration and disappointment,
evidenced by the exclamation marks and the rhetorical question.
- The word ``nose level'' is used figuratively to indicate that the
bag of dog treats was easily accessible to Rufus.
- The passage highlights a contrast between the author's oversight and
Rufus's deliberate consumption of the treats.
```

The question asks why Rufus ate the treats (gold: because he really likes them); "deliberate consumption" resolves it.

K Standard errors

Every accuracy in the paper is an avg@8: eight samples per problem at the evaluation temperature, scored and averaged (Appendix C). The main-text tables report the means. This appendix repeats every main-text table with a standard error on each cell. Each model is one training run, so these errors cover sampling at evaluation and nothing else.

Table 11: Reasoner without Distractors setting of Table 1, GSM-NoOp accuracy (%) as mean ± per-draw standard error over the eight samples per problem (Appendix K).

Role	Clean		With distractor	
	Unseen	Seen	Unseen	Seen
Qwen3-4B-Base				
Base model	91.9 ±0.5	80.3 ±1.5	54.1 ±2.5	54.7 ±2.5
Reasoner	94.7 ±0.5	93.6 ±0.4	58.3 ±0.9	63.1 ±1.0
+ Detector	92.2 ±0.4	92.3 ±0.6	**65.4 ±1.2**	68.2 ±0.8
+ Abstractor	**95.1 ±0.2**	**93.8 ±0.4**	63.1 ±0.4	**69.2 ±1.3**
+ Detector + Abstractor (8:4:4)	92.7 ±0.7	93.6 ±0.4	63.8 ±0.5	67.2 ±1.1
Qwen3-8B-Base				
Base model	94.0 ±0.4	84.3 ±1.6	62.3 ±1.9	63.4 ±2.3
Reasoner	96.1 ±0.4	93.5 ±0.4	62.4 ±0.7	67.7 ±0.7
+ Detector	95.5 ±0.2	96.2 ±0.2	68.4 ±0.5	74.3 ±0.4
+ Abstractor	95.7 ±0.3	**97.1 ±0.2**	68.0 ±0.9	74.1 ±0.9
+ Detector + Abstractor (8:4:4)	**96.2 ±0.1**	96.5 ±0.4	**70.4 ±0.7**	**76.1 ±0.5**

Table 12: Reasoner with Distractors setting of Table 1, GSM-NoOp accuracy (%) as mean ± per-draw standard error over the eight samples per problem (Appendix K).

Role	Clean		With distractor	
	Unseen	Seen	Unseen	Seen
Qwen3-4B-Base				
Base model	91.9 ±0.5	80.3 ±1.5	54.1 ±2.5	54.7 ±2.5
Reasoner	95.4 ±0.4	93.3 ±0.4	72.6 ±0.4	75.3 ±0.7
+ Detector	**95.5 ±0.4**	93.7 ±0.2	74.5 ±0.9	76.2 ±0.6
+ Abstractor	93.9 ±0.5	**94.6 ±0.3**	**78.7 ±0.5**	80.4 ±0.7
+ Detector + Abstractor (8:4:4)	94.7 ±0.4	93.7 ±0.4	78.3 ±1.0	**80.6 ±0.6**
Qwen3-8B-Base				
Base model	94.0 ±0.4	84.3 ±1.6	62.3 ±1.9	63.4 ±2.3
Reasoner	**96.0 ±0.2**	94.5 ±0.3	75.2 ±0.5	78.8 ±0.5
+ Detector	93.9 ±0.5	95.5 ±0.5	76.8 ±0.7	81.1 ±0.6
+ Abstractor	**96.0 ±0.3**	95.2 ±0.3	**83.6 ±0.8**	84.5 ±0.6
+ Detector + Abstractor (8:4:4)	95.6 ±0.2	**96.3 ±0.3**	81.4 ±0.5	**85.7 ±0.6**

Per-draw standard error. Sample i of every problem, for $i = 1, \ldots, 8$, forms one replicate of the benchmark. Its accuracy a_i is the mean over problems of the correctness of sample i. Tables 11 to 15 report the cell mean $\bar{a}$ with $\mathrm{SD}(a_1, \ldots, a_8)/\sqrt{8}$; a panel mean carries the standard error of the eight per-sample panel means. This is the sampling error of the avg@8 estimate for a fixed checkpoint on a fixed problem set. It does not cover variation across training runs, nor across problem sets.

All evaluations run in a batch-invariant decoding mode that makes every generation reproducible bit for bit, so every number can be regenerated exactly from the checkpoints.

Table 13: View ladder of Table 2, Qwen3-8B-Base, setting with distractors, problems with distractors, mean ± per-draw standard error. **Shown** is the number of instances of the template in the abstractor's prompt; the reward scores the method on those instances (shown) or on two others (withheld). One run per cell.

Role	Shown	Scored on	Unseen	Seen
Reasoner	–	–	75.2 ±0.5	78.8 ±0.5
+ Detector	–	–	76.8 ±0.7	81.1 ±0.6
+ Abstractor	1	shown	77.2 ±0.4	81.8 ±0.5
	1	withheld	78.9 ±0.5	83.7 ±0.6
	3	shown	80.3 ±0.5	85.6 ±0.3
	3	withheld	83.6 ±0.8	84.5 ±0.6

Table 14: Table 3 as mean ± per-draw standard error over the eight samples per question (Appendix K); the last column is the panel mean with the standard error of the per-draw panel means.

Role	RACE-C	QuAIL	CosmosQA	LSAT-RC	LSAT-LR	QuALITY	Mean
Qwen3-4B-Base							
Base model	72.1 ±0.9	62.6 ±0.9	60.6 ±1.2	62.3 ±0.8	53.9 ±0.9	58.4 ±1.4	61.6 ±0.8
Reasoner	83.1 ±0.1	74.6 ±0.3	75.9 ±0.2	71.8 ±0.3	62.9 ±0.4	71.7 ±0.2	73.4 ±0.1
+ Understander	84.2 ±0.1	77.6 ±0.2	78.0 ±0.3	74.0 ±0.4	63.2 ±0.2	72.1 ±0.4	74.8 ±0.2
+ Understander, $H=\{e\}$	83.7 ±0.3	74.5 ±0.7	74.9 ±0.6	71.7 ±0.5	59.4 ±0.6	71.7 ±0.3	72.6 ±0.4
Qwen3-8B-Base							
Base model	80.4 ±0.6	71.3 ±0.9	67.7 ±0.5	68.3 ±0.7	61.0 ±0.6	68.8 ±0.5	69.6 ±0.5
Reasoner	85.8 ±0.1	77.1 ±0.2	78.0 ±0.1	77.0 ±0.2	65.9 ±0.2	75.5 ±0.3	76.6 ±0.1
+ Understander	86.5 ±0.2	80.0 ±0.2	79.5 ±0.5	77.6 ±0.4	66.8 ±0.4	76.5 ±0.2	77.8 ±0.2

Table 15: Budget split of Table 4 (reasoner:understander rollouts), Qwen3-4B-Base, panel mean ± standard error of the per-draw panel means.

Split	Mean
base	61.6 ±0.8
16:0	73.4 ±0.1
12:4	75.0 ±0.2
8:8	74.8 ±0.2
4:12	74.6 ±0.2
0:16	65.8 ±0.7

Table 16: Budget split (reasoner:understander rollouts) on Qwen3-4B-Base, trained on RACE-C. Accuracy (%) reported.

Split	RACE-C	QuAIL	CosmosQA	LSAT-RC	LSAT-LR	QuALITY	Mean
Base model	72.1 ±0.9	62.6 ±0.9	60.6 ±1.2	62.3 ±0.8	53.9 ±0.9	58.4 ±1.4	61.6 ±0.8
16:0	83.1 ±0.1	74.6 ±0.3	75.9 ±0.2	71.8 ±0.3	62.9 ±0.4	71.7 ±0.2	73.4 ±0.1
12:4	**84.5 ±0.2**	77.0 ±0.3	77.6 ±0.1	**75.4 ±0.2**	62.3 ±0.2	**73.3 ±0.4**	**75.0 ±0.2**
8:8	84.2 ±0.1	**77.6 ±0.2**	**78.0 ±0.3**	74.0 ±0.4	**63.2 ±0.2**	72.1 ±0.4	74.8 ±0.2
4:12	84.3 ±0.1	76.7 ±0.1	77.4 ±0.2	74.1 ±0.4	63.1 ±0.4	72.1 ±0.4	74.6 ±0.2
0:16	76.3 ±0.7	66.9 ±0.8	64.8 ±0.9	65.6 ±0.5	56.1 ±0.6	64.9 ±1.3	65.8 ±0.7