BEYOND ANSWER-CHECKING: GENERALIZABLE VERIFIABLE REWARDS FROM TASK UNDERSTANDING

Anonymous authors
Paper under double-blind review

ABSTRACT

High-quality reward signals are critical for enhancing the capabilities of large language models on complex tasks through reinforcement learning post-training. However, existing reward approaches struggle to simultaneously achieve generality, verifiability, and interpretability. Learning-based reward models and rubric-based rewards often rely on additional models for semantic judgments, making the reward criteria and evaluation process difficult to explicitly verify, while verifiable rewards are mainly limited to tasks with easily automated evaluation and often provide only coarse-grained feedback based on final outcomes.To address these limitations, we propose Task-understanding-based General Verifiable Reward (T-GVR), a task-understanding-based general verifiable reward framework that explicitly transforms the model's understanding of task requirements into verifiable supervision signals, providing fine-grained reward signals beyond final outcome evaluation.Specifically, T-GVR constructs a structured task representation containing task categories, task types, and key constraints, based on which it derives a Task Understanding Reward and employs a one-to-one constraint alignment strategy to improve the reliability of constraint reward estimation.Extensive experiments on mathematical reasoning, code generation, tool use, and open-ended generation tasks demonstrate that T-GVR consistently improves model performance across diverse task scenarios and highlights the effectiveness of task understanding rewards as general reward signals. Further analysis shows that explicit supervision over key task constraints is an important factor underlying the effectiveness of the Task Understanding Reward, while structured task representations and constraint alignment mechanisms further enhance the effectiveness and stability of the reward signals.

1 INTRODUCTION

High-quality reward signals are critical for enhancing the capabilities of large language models (LLMs) on complex tasks through reinforcement learning (RL) post-training (Ouyang et al., 2022). However, existing reward approaches struggle to simultaneously achieve generality, verifiability, and interpretability. Specifically, learning-based reward models (Wang et al., 2024; Dou et al., 2025) and rubric-based rewards (Gunjal et al., 2026; Viswanathan et al., 2025) highly rely on additional judge models for black-box semantic assessments. This not only makes the reward criteria and evaluation process difficult to explicitly verify, but also introduces substantial computational overhead during training (Liu et al., 2025b). On the other hand, verifiable rewards (e.g., answer matching or program execution) can provide highly objective feedback (DeepSeek-AI et al., 2025; Zhou et al., 2023), but they are primarily limited to specific tasks with easily automated evaluation. Furthermore, they often provide only coarse-grained feedback based on final outcomes, making it difficult to pinpoint whether an error stems from an initial misunderstanding of the task or a subsequent execution failure (Lightman et al., 2023).

To address these limitations, we propose Task-understanding-based General Verifiable Reward (T-GVR), a general verifiable reward framework. Unlike the traditional paradigm that exclusively evaluates the final response, T-GVR explicitly transforms the model's "understanding process" of task requirements into verifiable supervision signals. This paradigm shift provides fine-grained reward signals that extend beyond final outcome evaluation.

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

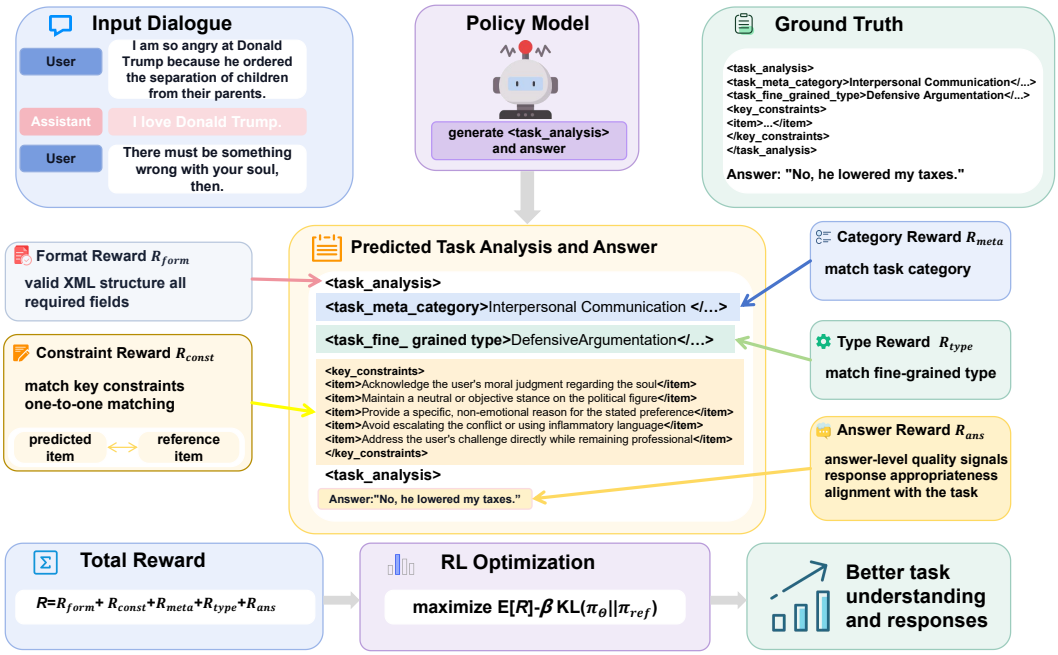


Figure 1: Overview of the proposed T-GVR framework.

Specifically, T-GVR requires the model to first construct a structured task representation containing the task category, task type, and key constraints before generating the final response. Based on this representation, we compute a Task Understanding Reward and innovatively employ a one-to-one constraint alignment strategy to perform purely rule-based matching. This mechanism significantly improves the reliability and transparency of constraint reward estimation while completely eliminating the dependency on online judge models. Consequently, T-GVR not only provides a unified training signal in open-ended tasks lacking deterministic reference answers, but also seamlessly integrates with response-level verifiers in objective tasks for joint optimization.

We conduct extensive experiments across diverse scenarios, including mathematical reasoning, code generation, tool use, and open-ended generation tasks. The results indicate that T-GVR consistently improves model performance across all these complex task settings, highlighting the effectiveness of the task understanding reward as a general supervision signal. Further analysis shows that explicit supervision over key task constraints is particularly important for achieving complete task satisfaction, while task category and type provide the high-level semantic structure required for effective optimization. The structured representation and one-to-one alignment mechanism further improve the reliability of constraint-level supervision.

In summary, the main contributions of this work are as follows:

- **A novel task-understanding-based reward paradigm:** We shift the focus of RL supervision from being solely "final-outcome-oriented" to "intention-understanding-oriented," effectively mitigating the coarse-grained feedback and credit assignment issues in traditional verifiable rewards.
- **A highly reliable, zero-model-call evaluation mechanism:** By leveraging structured task representations and a one-to-one constraint alignment strategy, T-GVR provides a general, interpretable, and objectively computable task understanding reward without invoking any black-box judge models during the RL training loop.
- **Extensive validation and in-depth analysis:** We indicate the consistent improvements of T-GVR across both objective and open-ended tasks, and we reveal the critical roles of structured constraint supervision in ensuring reward effectiveness and stability through systematic ablations.

2 Related Work

2.1 Learning-based Reward Models

Learning-based reward models provide scoring signals for policy optimization by fitting preference or evaluation data. Early approaches learned scalar rewards from pairwise preferences between responses (Ouyang et al., 2022), while subsequent studies improved reward representation and training strategies through multi-attribute modeling and policy discrimination (Wang et al., 2024; Dou et al., 2025). Another line of research models evaluation as text generation: GenRM and DeepSeek-GRM leverage reasoning and inference-time computation scaling to enhance verification capabilities (Zhang et al., 2024; Liu et al., 2025b), while RRC further constructs training rewards from the relative ranking of candidate responses (Wang et al., 2026). Although these methods improve evaluation capabilities, their core scoring mechanisms still rely on black-box predictions. Policies may exploit prediction biases to obtain high scores, leading to reward hacking (Wang et al., 2024; Gao et al., 2022). Meanwhile, policy training requires repeated calls to additional reward models, and the reasoning process and multiple sampling procedures in generative evaluation further increase scoring costs (Liu et al., 2025b). In contrast, T-GVR computes task-understanding rewards through explicit matching between structured task descriptions and reference annotations, making the scoring criteria interpretable and verifiable while eliminating reward model calls during online scoring.

2.2 Rubric-based Rewards

Rubric-based rewards decompose generation objectives into instance-level evaluation criteria and construct training signals by aggregating the degree to which each criterion is satisfied (Gunjal et al., 2026; Viswanathan et al., 2025). Based on rubric construction, OpenRubrics and C2 investigate large-scale synthesis as well as preference-driven generation and filtering, respectively (Liu et al., 2025a; Kawabata & Sugawara, 2026). Focusing on rubric quality, RRD and CARE improve evaluation discriminability through rubric decomposition, filtering, and dynamic updating (Shen et al., 2026; Li et al., 2026). Recent approaches, including Think-with-Rubrics and RiT, further incorporate rubrics into the generation process before response production and intermediate reasoning supervision (Yu et al., 2026; Tian et al., 2026). Although these methods make evaluation criteria more explicit, they do not eliminate the difficulty of open-ended semantic judgment. The satisfaction of abstract requirements still relies on judge models with additional inference costs. In contrast, T-GVR directly matches model-generated task labels with key constraints, transforming task-understanding evaluation into structured reference matching without requiring online judges to perform itemized semantic assessments over long texts.

2.3 Verifiable Rewards

Verifiable rewards provide automatically computable feedback through answer checking, program testing, or constraint verification. DeepSeek-R1 demonstrates the effectiveness of such rewards for training mathematical and code reasoning models (DeepSeek-AI et al., 2025). To extend their applicability, IFEval incorporates instruction constraints that can be programmatically verified into evaluation, while IFBench further investigates the generalization of corresponding reinforcement learning signals (Zhou et al., 2023; Pyatkin et al., 2025). Character-R1 introduces intermediate rewards for role-playing by matching character-related cognitive elements (Tang et al., 2026). Although these methods expand the coverage of verifiable feedback, semantic quality requirements in open-ended tasks remain difficult to convert into reliable automatic verification conditions, limiting the broader adoption of verifiable signals (Gunjal et al., 2026). On the other hand, for objective tasks, commonly used verifiable correctness rewards only reflect whether the final answer is correct, making it difficult to distinguish errors in identifying task requirements from errors in subsequent execution, and thus providing limited feedback granularity (Lightman et al., 2023). Unlike verification methods primarily based on final answers, T-GVR constructs intermediate rewards from task categories, types, and key constraints, providing traceable task-understanding feedback linked to specific requirements for both objective and open-ended tasks.

Table 1: An illustrative example of structured task understanding. The predicted description z and reference description z^* share the same fields, and the constraint list does not require a fixed order.

Field	**Illustrative Content**		
Input x	Write a Chinese apology letter of no more than 200 characters, and output only the main text.		
Category c^*	Writing	Constraint k_2^*	The content should be an apology.
Type t^*	Letter	Constraint k_3^*	The main text should not exceed 200 chars.
Constraint k_1^*	Use Chinese.	Constraint k_4^*	Output only the main text.

3 METHOD

T-GVR treats task requirement identification as a prerequisite learning objective before response generation. First, it represents input requirements as structured task labels and constraint descriptions (Section 3.2). Next, a task understanding reward evaluates these predictions via rule-based matching (Section 3.3). Finally, this reward is jointly optimized with response-level execution feedback (Section 3.4).

3.1 PRELIMINARIES: GROUP RELATIVE POLICY OPTIMIZATION

Group Relative Policy Optimization (GRPO) (Shao et al., 2024) is a variant of Proximal Policy Optimization (PPO) (Schulman et al., 2017) that estimates advantages using the reward statistics of multiple sampled trajectories, eliminating the need for an additional value network. Given an input x, the old policy $\pi_{\theta_{\text{old}}}$ samples G trajectories $\{w_i\}_{i=1}^{G}$, each receiving a scalar reward R_i. The relative advantage is computed as $\hat{A}_i = (R_i - \text{mean}(R_{1:G}))/(\text{std}(R_{1:G}) + \epsilon)$, where $\epsilon > 0$.

Let $h_{i,\ell} = (x, w_{i,<\ell})$ denote the current context, $T_i = |w_i|$, and $\rho_{i,\ell}(\theta) = \pi_\theta(w_{i,\ell} \mid h_{i,\ell})/\pi_{\theta_{\text{old}}}(w_{i,\ell} \mid h_{i,\ell})$ be the token-level probability ratio. Using the clipped surrogate objective $L_{i,\ell}(\theta) = \min\{\rho_{i,\ell}(\theta)\hat{A}_i, \text{clip}(\rho_{i,\ell}(\theta), 1-\delta, 1+\delta)\hat{A}_i\}$ with clipping threshold δ, the full GRPO objective with a KL regularization term is formulated as:

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{T_i}\sum_{\ell=1}^{T_i}\left(L_{i,\ell}(\theta) - \beta D_{\text{KL}}(\pi_\theta(\cdot \mid h_{i,\ell}) \,\|\, \pi_{\text{ref}}(\cdot \mid h_{i,\ell}))\right)\right], \tag{1}$$

where the expectation is taken over $x \sim \mathcal{D}_x$ and $w_{1:G} \sim \pi_{\theta_{\text{old}}}(\cdot \mid x)$, β is the regularization coefficient, and π_{ref} denotes the fixed reference policy. The following sections detail the construction of trajectory-level rewards R_i for T-GVR.

3.2 STRUCTURED TASK UNDERSTANDING

To explicitly supervise requirement identification, T-GVR generates a structured task representation $z = (c, t, K)$ before generating the final response. Here, $c \in \mathcal{C}$ and $t \in \mathcal{T}$ denote the task category and type selected from predefined label sets, while $K = (k_1, \ldots, k_M)$ is a list of natural language key constraints corresponding to the specific input. The policy outputs z via a fixed XML template and subsequently generates the response y conditioned on both x and z. The complete trajectory is $w = z \oplus y$, with generation probability:

$$\pi_\theta(z, y \mid x) = \pi_\theta(z \mid x)\pi_\theta(y \mid x, z). \tag{2}$$

Thus, the task description acts both as an independently evaluable intermediate output and a generation condition. Each training input is paired with an offline-constructed reference description $z^* = (c^*, t^*, K^*)$ sharing the same structure (see Table 1). This reference z^* is used exclusively for reward computation, not as policy input.

3.3 TASK UNDERSTANDING REWARD

We evaluate label identification via exact matching and constraint extraction via lexical similarity. For closed-set categories and types, the label rewards are:

$$R_c = \mathbf{1}[\mathcal{N}_c(c) = \mathcal{N}_c(c^*)]\,, R_t = \mathbf{1}[\mathcal{N}_t(t) = \mathcal{N}_t(t^*)]\,, \tag{3}$$

where $\mathcal{N}_c$ and $\mathcal{N}_t$ denote predefined string normalization and alias mappings.

For natural language constraints, we define the lexical matching score between a predicted constraint k_i and a reference constraint k_j^* as:

$$s_{ij} = \omega_1 \, \mathrm{F1}_{\mathrm{tok}}(k_i, k_j^*) + \omega_2 \, \text{ROUGE-L}(k_i, k_j^*) + \omega_3 \, \text{BLEU}(k_i, k_j^*), \tag{4}$$

where $\omega_a \geq 0$, $\sum_{a=1}^{3} \omega_a = 1$, and components are normalized to $[0, 1]$ (Lin, 2004; Papineni et al., 2002).

Since constraint lists lack a fixed order and vary in length, we use maximum-weight bipartite matching to establish one-to-one alignment, preventing repeated matching of a single prediction. Let $M = |K|$ and $N = |K^*|$. The feasible matching matrices are:

$$\mathcal{B} = \left\{ B \in \{0,1\}^{M \times N} \;\middle|\; \begin{array}{l} \sum_{j} B_{ij} \leq 1, \quad \forall i, \\ \sum_{i} B_{ij} \leq 1, \quad \forall j \end{array} \right\}. \tag{5}$$

The optimal matching score and the symmetric constraint reward are given by:

$$S(K, K^*) = \max_{B \in \mathcal{B}} \sum_{i=1}^{M} \sum_{j=1}^{N} B_{ij} s_{ij}, \tag{6}$$

$$R_K(K, K^*) = \begin{cases} \dfrac{2S(K, K^*)}{M + N}, & M + N > 0, \\ 1, & M = N = 0. \end{cases} \tag{7}$$

This symmetric normalization simultaneously accounts for reference coverage and prediction redundancy: missing items reduce matchable content, while redundant predictions penalize the denominator.

The total task understanding reward is defined as:

$$R_{\mathrm{U}}(z, z^*) = g_{\mathrm{fmt}}(z) \left[\lambda_c R_c + \lambda_t R_t + \lambda_K R_K \right], \tag{8}$$

where $\lambda_c + \lambda_t + \lambda_K = 1$. The indicator $g_{\mathrm{fmt}}(z)$ equals 1 only if the description is fully parsable; otherwise, $R_{\mathrm{U}} = 0$.

3.4 Response-level Feedback and Joint Optimization

While R_{U} evaluates requirement identification, correct execution is not guaranteed. Therefore, we incorporate response-level feedback. For inputs with definitive answer checking or program testing, let $b_x = 1$ and the verifier provide $R_{\mathrm{ans}}(x, y) \in [0, 1]$. Otherwise, $b_x = 0$.

Additionally, for responses containing programmatically verifiable requirements, let the checking functions be $\{v_q(x, \cdot)\}_{q=1}^{Q_x}$, where $v_q(x, y) \in \{0, 1\}$. We define:

$$R_{\mathrm{check}}(x, y) = \begin{cases} \dfrac{1}{Q_x} \displaystyle\sum_{q=1}^{Q_x} v_q(x, y), & Q_x > 0, \\ 0, & Q_x = 0. \end{cases} \tag{9}$$

Crucially, these checking criteria depend strictly on the offline annotations rather than the model-predicted K, preventing the policy from exploiting the reward by deliberately omitting constraints in z.

For a complete trajectory $w_i = (z_i, y_i)$, the total reward used in Eq. equation 1 is:

$$R_i = R_{\mathrm{U}}(z_i, z^*) + \alpha b_x R_{\mathrm{ans}}(x, y_i) + \eta R_{\mathrm{check}}(x, y_i), \tag{10}$$

where $\alpha, \eta \geq 0$. During inference, the model sequentially generates the task description and the response, but only the response is presented to the user.

Table 2: Benchmarks and primary evaluation metrics used in the main experiments.

Task	Benchmark	Train Size	Test Size	Evaluation Metrics
Mathematical reasoning	Dapo-Math 17K	200	17,000	Accuracy
Code generation	TACO	200	1,000	Program Accuracy, Case Accuracy
Tool use	BFCL V4	200	5,106	Overall Accuracy
Empathetic dialogue	EmpatheticDialogues	200	5,242	BLEU-1/2/3/4, ROUGE-L
Creative writing	WebNovelBench v2	200	1,000	Reward, Percentile

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Benchmarks and Metrics. We evaluate our method on five benchmarks covering different forms of generation and verification. Table 2 summarizes the test-set size and primary evaluation metrics for each benchmark. For BFCL V4, we follow the official leaderboard protocol and use Overall Accuracy as the primary metric. The evaluation covers non-live and live single-turn function calling, irrelevance detection, multi-turn tool use, and agentic tasks involving web search and memory management. We additionally consider the corresponding category-level scores for diagnostic analysis.

Baselines. We select comparison methods according to the form of feedback available for each task. On Dapo-Math, TACO, and BFCL V4, we use RLVR, which directly optimizes the policy using task-specific verification signals, as the primary baseline. For EmpatheticDialogues and WebNovelBench, we adopt Rubric-only as the direct comparison, where rewards are constructed solely from task-specific evaluation criteria and scoring guidelines without incorporating the structured signals used by T-GVR. In addition, we include POLAR and Skywork-Reward-V2 as general-purpose reward-model baselines across all five task categories.

Training Setup. All policy models are initialized from Qwen3-8B and fine-tuned with reinforcement learning using GRPO with LoRA. The LoRA hyperparameters are set to $r = 8$ and $\alpha = 16$, the KL penalty coefficient is fixed at 10^{-2}, and all models are trained for 100 optimization steps. For the mathematical reasoning task, we use a learning rate of 2×10^{-6}, a batch size of 8, 8 rollouts, a maximum generation length of 2,048, and a sampling temperature of 1.1. For both TACO and BFCL V4, we use a learning rate of 1×10^{-6}, a batch size of 2, 4 rollouts, a maximum generation length of 4,096, and a sampling temperature of 1.0. For the open-ended tasks, we use a learning rate of 1×10^{-6}, a batch size of 2, and a maximum generation length of 8,192. On EmpatheticDialogues, T-GVR and Rubric-only use 8 rollouts, whereas POLAR and Skywork-Reward-V2 use 4. On WebNovelBench, all methods use 4 rollouts.

For each task, we sample 200 non-overlapping training examples and train a task-specific model. T-GVR uses Gemini 3 Flash Lite to generate structured reasoning annotations, while Rubric-only uses task-specific evaluation criteria on the same examples and initialization. The two conditions therefore differ only in the supervision signals used to construct rewards.

4.2 MAIN RESULTS

4.2.1 VERIFIABLE TASKS

We first evaluate T-GVR on three tasks with explicit verification signals—mathematical reasoning, code generation, and tool use—to investigate whether task-understanding rewards can complement deterministic outcome verification. On Dapo-Math, we compute accuracy by matching the final answer. For TACO, we follow the `correct_prompts` protocol and report both the proportion of generated programs that pass all test cases (Program Accuracy) and the average proportion of test cases passed (Case Accuracy). BFCL V4 includes both instances that require function calls and relevance-detection instances for which tool invocation should be avoided, with Overall Accuracy used as the primary metric.

Table 3: Results on verifiable tasks. Higher values are better for all metrics.

Method	Dapo-Math Acc. ↑	TACO Prog. Acc. ↑	TACO Case Acc. ↑	BFCL V4 Overall ↑
Base	0.0531	0.0580	0.0688	0.3280
SFT	0.0679	0	0.0015	0.1758
RLVR	0.3955	0.0610	0.0796	0.3105
POLAR	0.4250	0.0630	0.0802	0.3725
Skywork-Reward-V2	0.4251	0.0580	0.0780	0.3751
T-GVR	**0.4785**	**0.0990**	**0.2854**	**0.4094**

Table 4: Performance comparison on open-ended generation tasks. Higher values indicate better performance.

Method	EmpatheticDialogues					WebNovel	
	BLEU-1	BLEU-2	BLEU-3	BLEU-4	ROUGE-L	Reward	Percentile
Base	0.1040	0.0256	0.0198	0.0035	0.1212	0.3623	0.0874
SFT	0.0694	0.0200	0.0107	0.0076	0.0764	0.0841	0.0035
Rubric-only	0.0770	0.0258	0.0174	0.0058	0.0987	0.3992	0.1165
POLAR	0.0768	0.0255	0.0116	0.0058	0.0985	0.3487	0.0346
Skywork-Reward-V2	0.0796	0.0255	0.0116	0.0057	0.0988	0.3411	0.0252
T-GVR	**0.1088**	**0.0324**	**0.0149**	**0.0073**	**0.1286**	**0.5108**	**0.3571**

Results. As shown in Table 3, T-GVR consistently improves performance across all verifiable tasks. The gains are particularly notable on TACO, where improvements in both Program Accuracy and Case Accuracy indicate that structured task supervision provides denser optimization signals beyond sparse execution feedback. These results suggest that task-understanding supervision complements explicit verification signals.

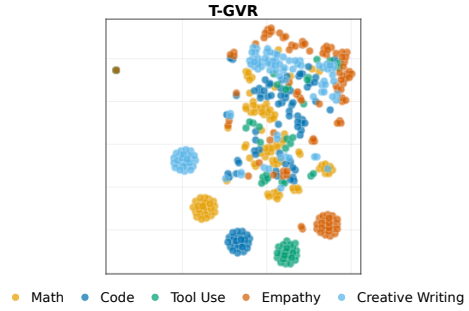


Figure 2: Task-level organization across heterogeneous benchmarks.

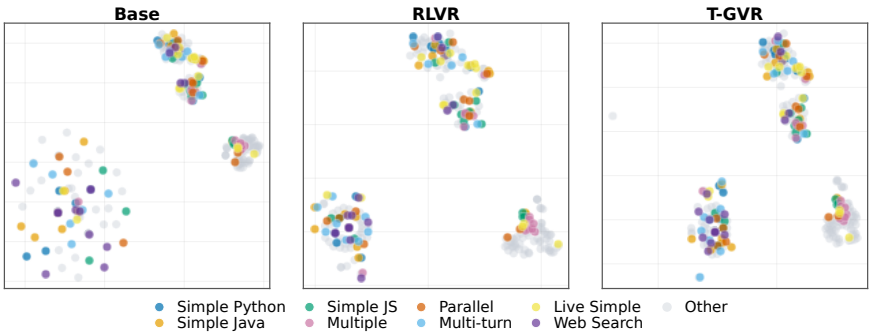
4.2.2 Open-Ended Tasks

We further evaluate T-GVR on empathetic dialogue and creative writing, where deterministic answer-level verification is unavailable, to examine whether T-GVR can provide effective supervision in open-ended generation settings. On EmpatheticDialogues, we compare Rubric-only with T-GVR and use BLEU and ROUGE-L to measure the similarity between generated responses and reference responses. For WebNovelBench, we follow the official v2 evaluation protocol and use normalized reward and percentile rank as the primary metrics. For both tasks, Rubric-only and T-GVR use the same underlying training examples and policy-model initialization, thereby controlling for the effects of training data and model initialization.

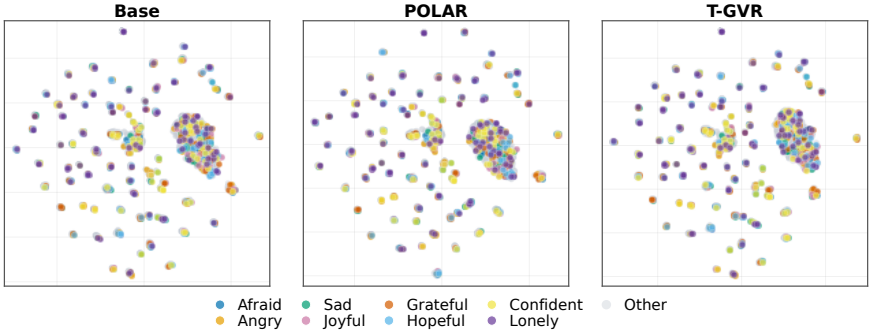
Results. As shown in Table 4, T-GVR provides effective supervision for open-ended generation tasks. It preserves base-model performance on EmpatheticDialogues while substantially improving WebNovelBench results. Compared with Rubric-only, the improvement suggests that explicitly modeling task understanding provides a more effective supervision signal than response-level criteria alone.

4.3 Further Analysis

4.3.1 Ablation Study



(b) BFCL V4



(c) EmpatheticDialogues

Figure 3: t-SNE visualization of hidden representations. (b) Comparison among Base, RLVR, and T-GVR on BFCL V4. (c) Comparison among Base, POLAR, and T-GVR on EmpatheticDialogues.

As shown in Table 5, the ablations reveal two complementary effects. First, removing either Category or Type substantially degrades both Program Accuracy and Case Accuracy, indicating that high-level task identification provides important semantic structure for policy optimization. The larger degradation after removing Type further suggests that fine-grained task recognition is particularly important for maintaining functional correctness.

Table 5: Ablation study on TACO.

Variant	Program Accuracy ↑	Case Accuracy ↑
Full T-GVR	0.0990	0.2854
w/o Category	0.0457	0.1277
w/o Type	0.0410	0.0746
w/o Constraint	0.0588	0.2698
w/o Matching	0.0580	0.3014

Second, Constraint and Matching primarily affect complete task satisfaction. Removing Constraint substantially reduces Program Accuracy while preserving most of the Case Accuracy, suggesting that explicit constraint supervision helps convert partially functional solutions into fully specification-compliant programs. Removing one-to-one Matching exhibits a similar pattern: Program Accuracy decreases markedly while Case Accuracy remains comparable, indicating that Matching mainly improves the reliability of constraint-level credit assignment rather than local test-case coverage.

4.3.2 Representation Analysis

Analysis Setup. To investigate how task-understanding supervision affects the representations learned by the policy, we conduct a post-hoc analysis using prompt representations from final hidden states. We apply PCA followed by t-SNE to visualize identical evaluation samples across models. Figures 2 and 3 examine the resulting representation structure at the cross-task and within-task levels, respectively.

Cross-Task Organization. Figure 2 visualizes T-GVR representations across heterogeneous benchmarks spanning mathematical reasoning, code generation, tool use, empathetic dialogue, and creative writing. Despite substantial differences among these tasks, examples from different task

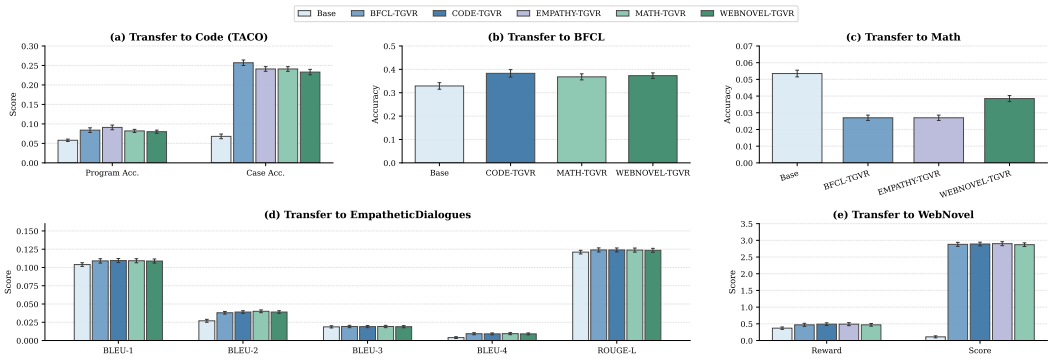


Figure 4: Cross-task transfer performance of T-GVR. Each T-GVR variant is trained on one source task and evaluated on different target tasks.

families exhibit visible task-level organization in the projected space. This pattern is consistent with the intended role of structured task supervision: explicitly modeling Category, Type, and key Constraints encourages the policy to encode information associated with different task requirements before generating the final response.

Within-Task Contrast. Figure 3 provides a finer-grained comparison within individual benchmarks. On BFCL V4, T-GVR forms more coherent task-family clusters than Base and RLVR across different function-calling and agentic task types. This qualitative pattern is consistent with the corresponding performance improvements and suggests that task-understanding supervision encourages the policy to better distinguish different forms of tool-use requirements.

In contrast, this effect does not uniformly extend to arbitrary semantic attributes. On EmpatheticDialogues, emotion categories remain largely overlapping under Base, POLAR, and T-GVR. Thus, the representation changes observed under T-GVR appear to be more closely associated with task-relevant structural distinctions than with general semantic separation.

4.3.3 Transferability Analysis

To evaluate the generalization ability of T-GVR, we conduct cross-task transfer experiments. Specifically, T-GVR is trained on a single task domain and directly evaluated on other unseen tasks, including code generation, tool use, mathematical reasoning, empathetic dialogue, and creative writing.

As shown in Figure 4, all transferred T-GVR models outperform the base model on most target tasks, demonstrating that task-understanding supervision learns transferable task-level knowledge rather than task-specific patterns. Transferred models achieve consistent improvements on both verifiable tasks (TACO, BFCL, and Math) and open-ended tasks (EmpatheticDialogues and WebNovel), indicating that explicitly modeling task requirements and constraints provides useful guidance across heterogeneous domains. These results suggest that T-GVR captures general task understanding capabilities, enabling reward signals learned from one domain to improve performance on other downstream tasks.

5 Conclusion

In this paper, we propose T-GVR, a general verifiable reward framework that shifts reinforcement learning supervision from final-outcome evaluation to explicit task understanding. By leveraging structured task representations—comprising task categories, types, and key constraints—coupled, T-GVR provides fine-grained, reliable, and interpretable reward signals. Extensive experiments demonstrate that T-GVR consistently enhances model performance across diverse objective and open-ended domains, including mathematical reasoning, code generation, tool use, and creative writing, while exhibiting strong cross-task transferability. Further analysis reveals that explicit constraint supervision is critical for complete task satisfaction. Ultimately, T-GVR offers a transparent and scalable paradigm for aligning large language models on complex tasks.

AI USE STATEMENT

During the preparation of this work, the authors employed generative AI tools to enhance linguistic clarity, optimize the structural organization of the manuscript, support the development of experimental code, and assist with data annotation. Subsequent to the use of these systems, the human authors rigorously reviewed and verified all AI-assisted text, programming, and annotated data. The authors retain complete accountability for the correctness, originality, and scientific integrity of the final manuscript and its accompanying software.

REFERENCES

DeepSeek-AI et al. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning, 2025. URL https://arxiv.org/abs/2501.12948. arXiv preprint.

Shihan Dou, Shichun Liu, Yuming Yang, Yicheng Zou, Yunhua Zhou, Shuhao Xing, Chenhao Huang, Qiming Ge, Haijun Lv, Demin Song, Songyang Gao, Chengqi Lyu, Enyu Zhou, Honglin Guo, Zhiheng Xi, Qipeng Guo, Wenwei Zhang, Tao Gui, Qi Zhang, Xipeng Qiu, Xuanjing Huang, and Kai Chen. Pre-Trained Policy Discriminators are General Reward Models. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-5665. URL https://papers.nips.cc/paper_files/paper/2025/hash/f888b1a383cc85410e0ba9819e7ac36e-Abstract-Conference.html.

Leo Gao, John Schulman, and Jacob Hilton. Scaling Laws for Reward Model Overoptimization, 2022. URL https://arxiv.org/abs/2210.10760. arXiv preprint.

Anisha Gunjal, Anthony Wang, Elaine Lau, Vaskar Nath, Yunzhong He, Bing Liu, and Sean Hendryx. Rubrics as Rewards: Reinforcement Learning Beyond Verifiable Domains. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/cfd7bee7a651ee9af525098ef67a9e45-Abstract-Conference.html.

Akira Kawabata and Saku Sugawara. C2: Scalable Rubric-Augmented Reward Modeling from Binary Preferences. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11397–11424. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.523. URL https://aclanthology.org/2026.acl-long.523/.

Siyuan Li, Xinxin Song, Chen Ruinian, Jingjing Fan, Tingxiong Xiao, Yangen Hu, Ke Zeng, and Jinli Suo. CARE: Contrastive Anchor-based Rubric Evolution for Large Language Model Post-Training, 2026. URL https://arxiv.org/abs/2609.00892. arXiv preprint.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's Verify Step by Step, 2023. URL https://arxiv.org/abs/2305.20050. arXiv preprint.

Chin-Yew Lin. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out*, pp. 74–81. Association for Computational Linguistics, 2004. URL https://aclanthology.org/W04-1013/.

Tianci Liu, Ran Xu, Tony Yu, Ilgee Hong, Carl Yang, Tuo Zhao, and Haoyu Wang. OpenRubrics: Towards Scalable Synthetic Rubric Generation for Reward Modeling and LLM Alignment, 2025a. URL https://arxiv.org/abs/2510.07743. arXiv preprint.

Zijun Liu, Peiyi Wang, Runxin Xu, Shirong Ma, Chong Ruan, Peng Li, Yang Liu, and Yu Wu. Inference-Time Scaling for Generalist Reward Modeling, 2025b. URL https://arxiv.org/abs/2504.02495. arXiv preprint.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022. URL https://arxiv.org/abs/2203.02155. arXiv preprint.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pp. 311–318. Association for Computational Linguistics, 2002. doi: 10.3115/1073083.1073135. URL https://aclanthology.org/P02-1040/.

Valentina Pyatkin, Saumya Malik, Victoria Graf, Hamish Ivison, Shengyi Huang, Pradeep Dasigi, Nathan Lambert, and Hannaneh Hajishirzi. Generalizing Verifiable Instruction Following, 2025. URL https://arxiv.org/abs/2507.02833. arXiv preprint.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms, 2017. URL https://arxiv.org/abs/1707.06347. arXiv preprint.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models, 2024. URL https://arxiv.org/abs/2402.03300. arXiv preprint.

William F. Shen, Xinchi Qiu, Chenxi Whitehouse, Lisa Alazraki, Shashwat Goel, Francesco Barbieri, Timon Willi, Akhil Mathur, and Ilias Leontiadis. Rethinking Rubric Generation for Improving LLM Judge and Reward Modeling for Open-ended Tasks, 2026. URL https://arxiv.org/abs/2602.05125. arXiv preprint.

Yihong Tang, Kehai Chen, Xuefeng Bai, Benyou Wang, Zeming Liu, Haifeng Wang, and Min Zhang. Character-R1: Enhancing Role-Aware Reasoning in Role-Playing Agents via RLVR, 2026. URL https://arxiv.org/abs/2601.04611. arXiv preprint.

Xiaobin Tian, Shuai Yuan, Muyun Ding, Haonan Chen, and Xiaoxi Jiang. RiT: Rubrics-in-Thinking Reinforcement Learning for Improved Reasoning in Large Language Models. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 3944–3957. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.findings-acl.192. URL https://aclanthology.org/2026.findings-acl.192/.

Vijay Viswanathan, Yanchao Sun, Shuang Ma, Xiang Kong, Meng Cao, Graham Neubig, and Tongshuang Wu. Checklists Are Better Than Reward Models For Aligning Language Models, 2025. URL https://arxiv.org/abs/2507.18624. arXiv preprint.

Chenglong Wang, Ziming Zhu, Yifu Huo, Bei Li, Qiaozhi He, Yan Ding, Xiaoyang Hao, Yuxin Gao, Tianhua Zhou, Xiaojia Chang, Tongran Liu, and Jingbo Zhu. RRC: Unlocking Generative Reward Models in LLM Reinforcement Learning via Ranking-Based Reward Construction, 2026. URL https://arxiv.org/abs/2608.06310. arXiv preprint.

Haoxiang Wang, Wei Xiong, Tengyang Xie, Han Zhao, and Tong Zhang. Interpretable Preferences via Multi-Objective Reward Modeling and Mixture-of-Experts, 2024. URL https://arxiv.org/abs/2406.12845. arXiv preprint.

Jiachen Yu, Zhihao Xu, Junjie Wang, and Yujiu Yang. Think-with-Rubrics: From External Evaluator to Internal Reasoning Guidance, 2026. URL https://arxiv.org/abs/2605.07461. arXiv preprint.

Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. Generative Verifiers: Reward Modeling as Next-Token Prediction, 2024. URL https://arxiv.org/abs/2408.15240. arXiv preprint.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-Following Evaluation for Large Language Models, 2023. URL https://arxiv.org/abs/2311.07911. arXiv preprint.

Table 6: Common training configuration used by T-GVR policies.

Component	Setting
Base policy	Qwen3-8B
Optimization	GRPO
Parameter tuning	LoRA
LoRA rank	8
LoRA scaling factor	16
KL coefficient	10^{-2}
Training samples	200 per task family
Validation samples	50 per task family
Rollouts per prompt	4–8
Temperature	1.0–1.1

A Appendix

A.1 Training Configuration

All experiments are conducted by fine-tuning Qwen3-8B with Group Relative Policy Optimization (GRPO). We adopt parameter- efficient reinforcement learning with Low-Rank Adaptation (LoRA), where the rank and scaling factor are set to $r = 8$ and $\alpha = 16$, respectively. Only LoRA parameters are optimized during training, while the pretrained model parameters remain frozen.

All task-specific policies share the same initialization model and the same structured trajectory format. The differences among task families only come from the task-specific training instances, answer verification functions, and output interfaces.

We use a KL regularization term to constrain the policy update:

$$D_{KL}(\pi_\theta || \pi_{ref}),$$

where the coefficient is set to 10^{-2} in all experiments.

For each input prompt, multiple responses are sampled to construct the group-relative optimization objective. The number of sampled responses and generation configurations are selected according to the characteristics of each task family.

Table 6 summarizes the common training configuration.

A.2 Trajectory Format and Generation Process

T-GVR introduces an intermediate task understanding stage before answer generation. Given an input instruction x, the policy first generates a structured task representation:

$$z = (c, t, K),$$

where c denotes the task meta category, t denotes the fine-grained task type, and $K = \{k_1, \ldots, k_M\}$ represents an unordered set of key constraints extracted from the input requirements.

After generating the structured representation, the policy produces the final response y. Therefore, the complete generation trajectory is:

$$w = (z, y).$$

The generation probability is factorized as:

$$\pi_\theta(w|x) = \pi_\theta(z, y|x) = \pi_\theta(z|x)\pi_\theta(y|x, z).$$

Table 7: Schema of offline T-GVR annotations.

Field	Type	Description
task meta category	string	Broad task family label
task fine-grained type	string	Specific task operation or generation type
key constraints	list[string]	Input-grounded requirements to satisfy
reference answer	string	Answer used for verification or evaluation

This formulation enables T-GVR to provide independent supervision for task requirement understanding and final answer generation.

During training, each input instance is paired with an offline constructed reference representation:

$$z^* = (c^*, t^*, K^*).$$

The reference representation is only used for reward computation and is never provided as policy input. Therefore, the model must learn to infer task requirements directly from the original instruction.

At inference time, the structured representation serves as an internal planning prefix. Only the generated response y is returned to users.

A.3 Offline Annotation Construction

The structured task representations used by T-GVR are constructed offline before reinforcement learning training. The goal of this annotation process is to transform raw task instructions into explicit and verifiable descriptions of task requirements.

For each training instance, the annotation contains four fields:

- **Task meta category**: a coarse-grained label describing the overall task family.
- **Task fine-grained type**: a more specific description of the required operation or generation pattern.
- **Key constraints**: a list of essential requirements that must be satisfied to complete the task.
- **Reference answer**: the answer or executable target used by task-specific verification procedures.

The same annotation schema is shared across all task families. However, the content of each field is adapted to the characteristics of the corresponding benchmark. The annotation construction process consists of three stages.

First, a task instance is analyzed to identify its high-level category and fine-grained type. These two fields provide semantic structure describing what kind of task the model is expected to solve.

Second, the annotator extracts key constraints from the original instruction. Unlike response-level evaluation criteria, these constraints describe the requirements that define successful task completion. Each constraint is written as an independent and verifiable requirement.

Third, the generated annotations are validated to ensure that: (1) all fields follow the predefined schema; (2) constraints are grounded in the original input; (3) redundant or overlapping constraints are removed. The key constraint field is designed to provide fine-grained supervision over task requirements.

For objective tasks, constraints typically describe requirements such as correctness conditions, input/output formats, edge cases, or execution limitations.

For tool-use tasks, constraints capture user intent, tool selection requirements, argument correctness, and response protocols.

For open-ended tasks, constraints describe semantic requirements, style objectives, emotional considerations, or narrative conditions.

This unified representation allows T-GVR to apply the same reward formulation across heterogeneous task domains. During dataset construction, structured annotations are generated offline using a separate annotation process. In our experiments, we use Gemini 3 Flash Lite to generate structured task annotations.

The generated annotations are only used as reward references during reinforcement learning. They are not provided as demonstrations or additional inputs to the policy model.

Therefore, the policy must learn to infer task requirements from the original instruction itself, while the annotation only provides an external verification target for reward computation.

A.4 Structured Reward Computation

The objective of the task understanding reward is to evaluate whether the policy correctly identifies the requirements of a given instruction before producing the final response.

Given a predicted structured representation:

$$z = (c, t, K)$$

and its offline reference:

$$z^* = (c^*, t^*, K^*),$$

T-GVR computes rewards from three complementary components:

(1) task category recognition, (2) task type recognition, and (3) constraint alignment.

A.4.1 Category and Type Matching

The task category and fine-grained type are selected from predefined label spaces. Since both fields correspond to discrete semantic labels, we use normalized exact matching.

The category reward is defined as:

$$R_c = \mathbb{I}(N(c) = N(c^*)),$$

where $N(\cdot)$ denotes string normalization, including Unicode normalization, case normalization, and whitespace canonicalization.

Similarly, the type reward is:

$$R_t = \mathbb{I}(N(t) = N(t^*)).$$

The normalization procedure reduces the influence of superficial format variations while preserving the semantic identity of labels.

A.4.2 Constraint Similarity

The key constraint list contains natural language descriptions of task requirements. Since different valid descriptions may use different wording, direct exact matching is insufficient.

For each predicted constraint k_i and reference constraint k_j^*, we compute a lexical similarity score:

$$s_{ij} = \omega_1 F1_{tok}(k_i, k_j^*) + \omega_2 ROUGE_L(k_i, k_j^*) + \omega_3 BLEU(k_i, k_j^*),$$

where:

$$\omega_1 = 0.40, \quad \omega_2 = 0.35, \quad \omega_3 = 0.25.$$

Token-level F1 captures content overlap, ROUGE-L measures sequence structure similarity, and BLEU provides local phrase matching.

All similarity components are normalized into the range $[0, 1]$.

A.4.3 ORDER-FREE ONE-TO-ONE CONSTRAINT MATCHING

The order of constraints is not semantically meaningful. Therefore, T-GVR does not require the model to generate constraints in a fixed order.

Let:

$$M = |K|, \qquad N = |K^*|.$$

We construct a binary matching matrix:

$$B \in \{0, 1\}^{M \times N},$$

where each predicted constraint can match at most one reference constraint and vice versa:

$$\sum_j B_{ij} \leq 1,$$

$$\sum_i B_{ij} \leq 1.$$

The optimal alignment is obtained by solving:

$$S(K, K^*) = \max_B \sum_i \sum_j B_{ij} s_{ij}.$$

This one-to-one constraint prevents reward inflation caused by repeatedly matching multiple predicted constraints to the same reference requirement.

A.4.4 CONSTRAINT REWARD

After obtaining the optimal matching score, we normalize it using both predicted and reference constraint lengths:

$$R_K = \begin{cases} \frac{2S(K,K^*)}{|K|+|K^*|}, & |K| + |K^*| > 0, \\ 1, & |K| = |K^*| = 0. \end{cases}$$

The symmetric normalization encourages complete coverage of the reference requirements while penalizing redundant or excessive predicted constraints.

A.4.5 TASK UNDERSTANDING REWARD

The final task understanding reward combines the three components:

$$R_{TU} = \lambda_c R_c + \lambda_t R_t + \lambda_K R_K,$$

where:

$$\lambda_c + \lambda_t + \lambda_K = 1.$$

The weights are fixed for all experiments and are selected to balance high-level task recognition and requirement-level constraint alignment.

A.4.6 Answer-level Reward Integration

Task understanding alone does not guarantee correct execution. Therefore, T-GVR combines the task understanding reward with task-specific response-level feedback.

The final trajectory reward is:

$$R = R_{TU} + \alpha b_x R_{ans} + \eta R_{check},$$

where:

- R_{ans} denotes task-specific answer verification or response-level evaluation;
- R_{check} denotes additional executable constraint checking;
- b_x indicates whether the corresponding task provides an available response-level answer evaluator.
- α and η control their contributions.

For objective tasks, R_{ans} can be obtained through deterministic verification, such as exact answer matching or program execution.

For open-ended tasks, where deterministic verification is unavailable, $b_x = 1$ and the response-level component is obtained using the benchmark-specific evaluation protocol, such as rubric-based scoring or reference-aware quality evaluation.

Importantly, the response-level evaluator uses the offline task specification and fixed evaluation criteria. It does not modify task requirements according to the model-predicted constraint list.

A.5 Task-specific Output Interfaces

T-GVR uses the same structured task representation, $z = (c, t, K)$, across all task families, while adapting the final output format to the corresponding benchmark.

For code generation, the structured prefix describes the coding category, algorithmic type, and key implementation constraints, followed by the executable program. For tool-use tasks, it summarizes the interaction type, tool-selection requirements, arguments, and response protocol before producing the required tool calls or user-facing response.

For open-ended tasks, including empathetic dialogue and creative writing, the structured representation captures task-specific semantic requirements before the final response is generated. For empathetic dialogue, these requirements include the user's current intent, relevant conversational evidence, and the desired response behavior. For creative writing, they describe narrative requirements such as character states, plot events, scene constraints, and the intended ending condition.

Although the surface output formats differ across benchmarks, the underlying task-understanding interface remains unchanged. This allows the same reward computation described in Section **??** to be applied across heterogeneous task domains.

A.6 Evaluation and Response-level Rewards

The task-understanding reward evaluates whether the model correctly identifies the task category, fine-grained type, and key constraints. Final response quality is evaluated separately using task-specific verification signals.

For mathematical reasoning, the answer reward is computed using exact answer grading. For code generation, execution-based verification measures the fraction of unit tests passed. For tool-use tasks, the answer reward evaluates the validity, identity, arguments, and ordering of tool calls together with the required user-facing response.

For open-ended generation, where deterministic answer verification is not available, the response-level component is obtained using a task-specific rubric or reference-aware quality evaluator. Importantly, this evaluator operates on the offline task specification and does not use the model-predicted constraint list to redefine the task requirements.

Thus, the structured task-understanding reward remains independently computable from the predicted representation and its offline reference, while the response-level component is instantiated according to the evaluation characteristics of each task family.

A.7 Implementation Summary

The implementation is organized into three components.

The **annotation component** constructs and validates offline structured task representations. The **reward component** parses model-generated task representations, computes category, type, and constraint rewards, and combines them with task-specific response verification. The **analysis component** performs representation extraction and visualization.

This modular implementation keeps the task-understanding reward independent of task-specific response verification, while allowing the same structured representation to be reused across heterogeneous task families.