R-Log: Incentivizing Log Analysis Capability in LLMs via Reasoning-based Reinforcement Learning

Yilun Liu[1,2,*], Ziang Chen[1,2,*], Song Xu[2], Minggui He[2], Shimin Tao[2], Weibin Meng[2], Yuming Xie[2]
Tao Han[2], Chunguang Zhao[2], Jingzhou Du[2], Daimeng Wei[2], Shenglin Zhang[1], Yongqian Sun[1†]

[1] Nankai University, China
[2] Huawei, China

{liuyilun3,chenziang8,xusong26,heminggui,taoshimin,mengweibin3,yuming.xie,billow.han}@huawei.com
{zhaochunguang4,dujingzhou,weidaimeng}@huawei.com,{zhangsl,sunyongqian}@nankai.edu.cn

Abstract

The growing complexity of log data in modern software systems has prompted the use of Large Language Models (LLMs) for automated log analysis. Current approaches typically rely on direct supervised fine-tuning (SFT) on log-label pairs. However, this exacerbates the domain discrepancy between general-purpose LLMs and specialized log data, causing overfitting. Furthermore, SFT's imbalanced loss computation often allows lengthy contexts to overwhelm critical, concise details in model answers, leading to hallucinations. To address these limitations, we propose R-Log, a novel reasoning-based paradigm that mirrors the structured, step-by-step analytical process of human engineers. This approach enhances generalizability by learning the underlying rules behind conclusions. We further employ Reinforcement Learning (RL) to optimize the model within a simulated O&M environment, thereby reducing hallucinations by directly rewarding correct outcomes. R-Log is first cold-started on a curated dataset of 2k+ reasoning trajectories, guided by 13 strategies from manual O&M practices, to establish an initial reasoning capability. This ability is then refined via RL using a joint reward function. Empirical evaluations on real-world logs show that R-Log outperforms existing methods across five log analysis tasks, particularly in unseen scenarios (by 228.05%). We also designed R-Log-fast with 5x speedup while keeping 93% of the efficacy.

CCS Concepts

• **Computing methodologies** → **Natural language processing**; *Machine learning*; • **Networks** → Network monitoring.

Keywords

Log analysis, Inference-time Reasoning, Reinforcement Learning

---

* Equal Contribution. † Corresponding author.

Figure 1: An real case from evaluation illustrating the reasoning-based nature of R-Log. Through human-like step-by-step reasoning, R-Log avoided the hallucinated "a minute" by existing LLMs and successfully identified the short response time "1ms" from the error log.

1 Introduction

Automated log analysis plays a pivotal role in modern software engineering, serving as a fundamental practice for ensuring system reliability, performance, and security. To facilitate the comprehensive understanding of diverse log events within complex software

systems and aid human Operation and Maintenance (O&M) engineers in managing the sheering volume of incidents, a variety of sub-tasks are studied in the field of log analysis. These sub-tasks range from enhancing log interpretation [35] and log parsing (into variables and templates) [64], to applying logs for practical problem-solving, such as anomaly detection [27], root cause analysis [4] and solution recommendation [33].

However, the increasing complexity of software modules and the growing customization of analysis scenarios pose new challenges: training and deploying massive specialized models for each scenario, task or domain can lead to significant cost [33, 54]. In practice, the performances of these specialized models are further hampered by insufficient historical logs, as online environments may continuously generate new log patterns unseen during training [34, 35]. To address this, Liu *et al.* proposed LogLM [33], fine-tuning a single large language model (LLM) for multiple sub-tasks of log analysis, given LLMs' strong capability acquired in pre-training [11].

Nevertheless, LLMs are typically trained on generic plain text corpora (*i.e.*, natural language), which unavoidably introduces distribution discrepancy [23, 49] when being directly fine-tuned on domain-specific and highly structured log texts (*e.g.*, IP addresses, status codes and file paths, *etc.*). For example, LogLM relies on direct answer fitting via Supervised Fine-Tuning (SFT) using log-label pairs without learning any rules behind the analytical conclusions. In this paper, we hypothesize that, incentivizing capability in general-purpose LLMs for specialized, complex domains like log analysis may require a paradigm shift: **from directly fitting answers (X→Y) to learning the reasoning trajectories of human experts in natural language to derive answers through reasoning (X→(R,Y))**. This shift is inspired by observations on real-world software development and O&M practices, where human engineers decompose complex problems from unfamiliar domains into sub-steps and follow a trajectory (*e.g.*, procedures outlined in O&M manuals [14]) to reason toward final conclusions [22]. Secondly, with the increasing of task contexts and scenarios complexity, direct SFT on longer and more heterogeneous sequences (*e.g.*, multi-task training) often struggles to converge to the optimal [5], suggesting the need for more sophisticated learning algorithms tailored to complex scenarios (*e.g.*, Reinforcement Learning (RL)).

To this end, we propose R-Log, a novel paradigm to incentivize log analysis capabilities in LLMs by driving the model to reason (*i.e.*, output thinking trajectories as shown in Fig. 1) first before getting the answer through RL. The key innovations of R-Log are:

(1) From X→Y to X→(R,Y). Existing methods directly train LLMs on log (X)-label (Y) pairs, expecting the model to spontaneously learn intrinsic patterns for solving log analysis problems. However, the distribution discrepancy between general texts and logs makes LLMs difficult to learn the intrinsic patterns, potentially leading to overfitting on historical logs (*i.e.*, learning by rote [20]). Consequently, the X→Y paradigm may result in limited generalizability to handle unseen scenarios (*e.g.*, performance of LogLM degrades sharply for a untrained new task in Section 4.6). In contrast, R-Log's learning objective encompasses both the answer and an explicit reasoning process aligned with human practices (*i.e.*, X→(R,Y)), facilitating the learning of universal rules behind the answers (*e.g.*, the steps in Fig. 1), thereby enhancing generalization (*i.e.*, outperforming LogLM in the unseen task by 228.05%).

Additionally, the model's explicit reasoning trajectory enhances its transparency and trustworthiness.

(2) From SFT to RL. With the increasing complexity of log analysis scenarios, handling complex tasks with flexible instructions and heterogeneous answers has become a preferred requirement for LLMs [33]. However, SFT inherently optimizes in a token-by-token "memorization" manner [5], where losses are evenly distributed across tokens (*i.e.*, a word-wise comparison with reference answers), making convergence difficult for long and complex sequences [19]. For instance, in Fig. 1, a user asks to interpret the error log with natural language (a popular scenario in recent studies [33, 34]), and the reference answer is complex beyond a short label (*e.g.*, "normal" or "abnormal"). Since the computation of loss in SFT is evenly distributed across the long reference answer, some minor key point like the "1ms" will be overwhelmed by other information, leading to hallucinations (*i.e.*, "1ms" was identified as "a minute" by existing LLMs in Fig. 1). Another case is when fine-tuning the model with reasoning trajectories, the longer reasoning part R receives more signals than the shorter Y, potentially preventing the model from producing correct conclusions (further discussed in Section 4.4)[1]. R-Log overcomes this by modeling log analysis as an O&M agent interacting with a heterogeneous environment that simulates real-world O&M practices. The agent takes actions on system-triggered log incidents that require diverse analysis skills (*e.g.*, log parsing, or root cause analysis), receives rewards based on the correctness of its analysis, and updates parameters using RL algorithms. therefore, it better aligns with human practices and prioritizes correct conclusions over mere token matching, as indicated by the steady performance improvement by RL in Fig. 3(a).

To achieve this, we first collected expertise from human log analysts and summarized it into 13 reasoning templates that reflects thinking patterns of professional engineers in various log analysis tasks. We then instantiated these reasoning templates with real-world logs and constructed a dataset of 2k+ samples with X→(R,Y) pairs, named the Log Reasoning Dataset. R-Log's training is two-staged: in the first stage, we perform SFT on the base model using Log Reasoning Dataset as a cold-start, enabling the LLM to imitate human expert reasoning strategies; in the second stage, we employ RL on the cold-started model to further calibrate its reasoning paradigms. Evaluation results indicate that R-Log outperforms specialized models and general-purpose LLMs across five log analysis tasks, particularly in handling unseen tasks. Our contributions are:

- We propose a novel reasoning-based log analysis paradigm that enables LLMs to learn the reasoning strategies of human experts rather than merely fitting labels, enhancing both generalization and interpretability.
- We design a novel RL-based training algorithm for log analysis, allowing the model to adapt to more complex scenarios and contexts.
- We open-source the Log Reasoning Dataset[2] containing 2k+ real-world reasoning trajectories in log analysis as well as the 13 curated typical reasoning strategies from manual practices, facilitating future research.

---

[1]A straightforward solution may be to balance the loss with different weights, but it is hardly feasible in practice to precisely label different parts and assign proper weights.
[2]Code and dataset are available at https://github.com/lunyiliu/R-Log

2 Related Work

2.1 Reasoning-based LLMs

Since the pioneering advancement of DeepSeek-R1 [8] on solving math problems, the reasoning-based paradigm (also called inference-time scaling) has become a promising alternative for LLMs on solving complex, domain-specific and expertise-required problems. Reasoning-based LLMs are characterized by its scaled reasoning steps before outputting final answer during inference time, thereby are particularly suitable for domains that naturally require a thinking procedure to reach the conclusion. For example, in addition to mathematics, the advantage of reasoning-based LLMs has been further verified on domains such as code generation [29], legal field [6], financial reasoning [45] and machine translation [16].

Given the complexity and expertise required to perform the tasks of log analysis, as well as the pervasive existence of reasoning procedures as recommended by manuals [14, 24] in real-world software O&M practices, it is motivated for reasoning-based paradigm to be applied in it. And R-Log advances by demonstrating the advantage of reasoning-based LLMs in software log analysis, through its carefully designed training paradigm tailored for log analysis.

2.2 Reinforcement Learning

With the increasing complexity in applied scenarios of AI systems, the theory of RL develops to overcome the learning goals that are infeasible for traditional gradient-based algorithms (*e.g.*, non-differentiable) [57]. By simulating the real-world learning environment, an agent takes actions according a certain probability distribution (called policy) and receives reward signals from the environment, iteratively converging to an optimized policy for the specific environment [3]. Recently, RL techniques have achieved significant success in building powerful LLMs, aligning the model with human expectations through the RLHF (Reinforcement Learning with Human Feedback) algorithm. Compared with RLHF which requires training a critic model to simulate human feedbacks, another popular alternative is GRPO (Group Relative Policy Optimization) [47], a simplified RL algorithm adopted by DeepSeek-R1 without training a critic model, which uses relative advantages among a group of sampled answers to define the reward signals. By utilizing GRPO, recent studies have explored performing software failure localization and change management [50, 60].

Compared with existing studies, R-Log firstly applied RL into the general field of log analysis. Beyond a specific sub-task (*e.g.*, log anomaly detection), we designed a joint reward function to measure actions in a heterogeneous O&M environment with various analysis tasks, enabling a more generalized policy for log analysis.

2.3 Log Analysis

2.3.1 Task-specific Approaches.

To handle a specific challenge or facilitate a specific analysis scenario, new sub-tasks in log analysis and specialized approaches for the sub-tasks continuously emerges. Typical sub-tasks are:

(1) Log Parsing, which aims at parsing raw logs into templates and variables, thereby reducing its sheering volumes and facilitates further analysis. A log template only retains static parts in a log and replaces dynamic variables with a symbol of <*>. Log parsing approaches can be divided into coarse-level methods, which focus on extracting common parts as templates in raw logs [9, 13, 17, 38, 61], and fine-level methods, which focus on identifying variables within raw logs [21, 30, 39, 53].

(2) Log Anomaly Detection, which classifies anomalous events from normal behaviors within system logs. Traditional approaches require massive historical logs as training samples to model the anomaly patterns within systems [10, 40, 62], thereby can hardly handle the frequently updated online environment. In contrast, semantic-based approaches identifies anomalous patterns based on understanding semantics within log templates [34, 42, 44], thereby can achieve better results with a small amount of training samples.

(3) Log Interpretation, which aims at aiding human engineers in interpreting logs by describing the logs in natural language. Liu *et al.* proposed a systematic criteria for evaluating the performance of log interpretation and tested advanced LLMs on this sub-task using specifically designed prompts [34, 35].

(4) Root Cause Analysis, which predicts the root causes of system events recorded by logs. Chen *et al.* [4] implemented an LLM-empowered system that predicts cloud incidents' root cause type.

(5) Solution Recommendation, which provides mitigating solutions to crashes indicated by system logs. Ahmed *et al.* [1] firstly explored this sub-task by fine-tuning LLMs to handle cloud incidents and recommend mitigation steps for engineers.

(6) Log Variable Classification, which recognizes the categories of dynamic variables within logs to aid downstream analysis. Compared with log parsing which merely identifies appearance of variables, this sub-task requires a more comprehensive understanding to logging patterns within the system and predicts the specific type of variables (*e.g.*, an string of numbers can either be an object amount or a status code) [21, 30].

2.3.2 Unified Approaches.

With the development of LLMs, the boundary between sub-tasks in log analysis has been blurred. By modeling log-label pairs in different sub-tasks into a unified format of instruction-response, LogLM achieved utilizing a single LLM to perform various log analysis tasks [33]. Ji *et al.* utilize continuous pre-training on interpretable knowledge to transform a general-purpose LLM into a specialized model tailored for log analysis [23]. Gou *et al.* [15] proposed OWL, an LLM trained for question-and-answering (QA) tasks in IT operation.

However, existing approaches fit the LLM with answers only, limiting its potential in understanding reasons behind analysis answers. In contrast, R-Log is trained on human-aligned reasoning trajectories and is further enhanced through RL, thereby better incentivizing log analysis capabilities within LLMs.

3 Methodology

An overview of R-log is presented in Fig. 2. We first studied typical reasoning strategies human O&M engineers employs when conducting log analysis in practice, and summarized into 13 task-specific reasoning templates tailored for various log analysis sub-tasks. Then, we instantiated these templates with real-world logs into reasoning trajectories (R) from the log to be analyzed (X) to the conclusion (Y), handling five specific log analysis sub-tasks. Using this dataset containing X→(R,Y) samples, we trained R-Log with a two-stage strategy. In the first stage, by directly SFT the foundation
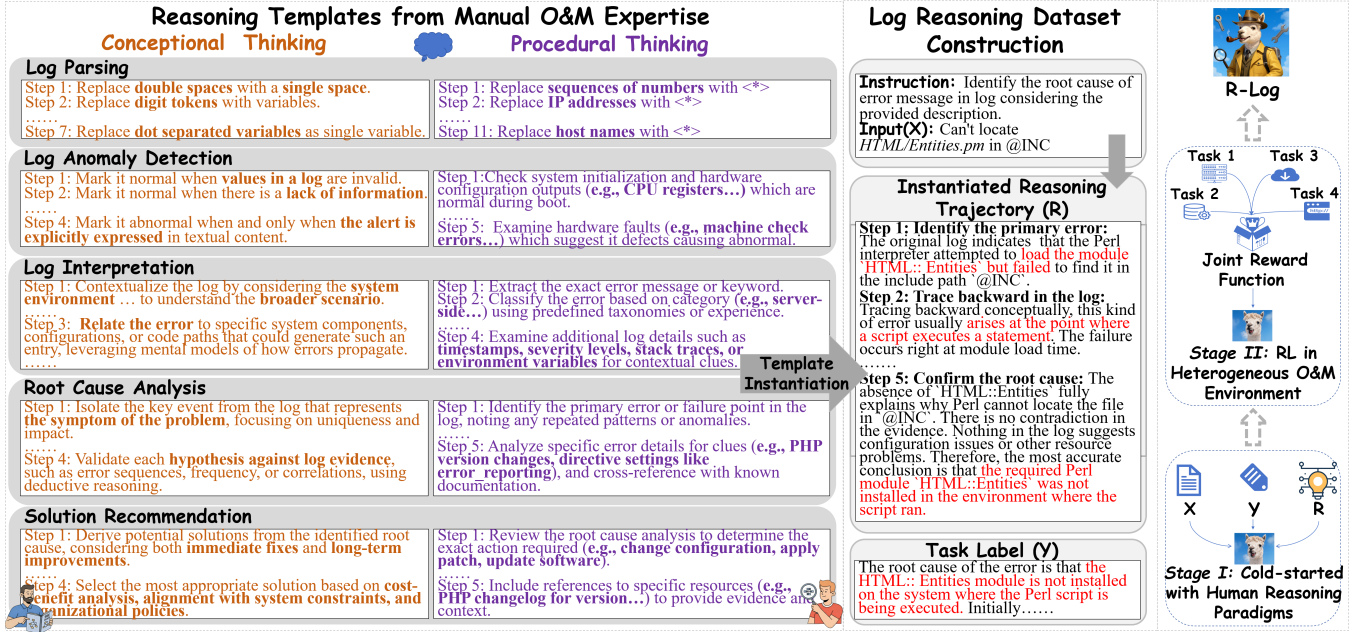
Figure 2: Illustration on the construction of the human-aligned Log Reasoning Dataset and two-staged training of R-Log. Highlighted parts reflects natures of "conceptional" or "procedural" in thinking strategies, and the real-world logs in instantiation.

model using X→(R,Y) samples, the model is cold-started with the pre-constructed reasoning paradigms imitating human experts. In the second stage, we designed a joint reward function to guide the further optimization of reasoning paradigms in a heterogeneous O&M environment with diverse log analysis scenarios.

3.1 Reasoning Strategies in Manual O&M Expertise

As revealed by Jefferies *et al.* [22] in a psychological study on human software engineers, senior engineers tend to "decompose a problem more richly into minimally interacting parts," suggesting the importance of reasoning strategies in software engineering. In order to built a comprehensive and human-aligned reasoning ability for LLMs, we started by investigating reasoning strategies of human O&M engineers in log analysis, from both the angle of cognitive psychology and task-specific practice guidelines.

3.1.1 Conceptional Thinking v.s. *Procedural Thinking.* By studying individual cognitive differences of software engineers, Bill [7] concluded two primary ways to build reasoning strategies for engineers: (1) by abstracting from development experience and (2) by learning specific rules from training programs. This taxonomy, aligning with modern cognitive psychology [51], reveals two distinct reasoning patterns adopted by software engineers: conceptional thinking and procedural thinking. In the context of software log analysis, conceptional thinking style seeks to abstract patterns and strategies from experiences to guide the log analysis tasks (*e.g.*, judging anomaly by overall semantics in logs), while procedural thinkers tend to follow a checklist-like reasoning trace focusing on examining specific attributes (*e.g.*, checking specific status codes in logs to determine

failure). These distinct thinking styles reflect the diversity of human reasoning strategies, and guide through our dataset curation.

3.1.2 Task-specific Reasoning Strategies. To ensure a comprehensive coverage of analysis scenarios, we investigated human reasoning strategies in five typical log analysis sub-tasks (Section 2.3.1 introduced these sub-tasks), encompassing both conceptional and procedural thinking styles. The sources of these strategies include manually-extracted rules for performing sub-tasks [20, 34], software O&M manuals and guidelines [14, 24] and interviews with O&M practitioners.

Reasoning Strategies for Log Parsing and Anomaly Detection. These two sub-tasks have been widely investigated by community. Liu *et al.* [34] proposed a series of high-level steps (*e.g.*, judging by semantics like "textual alert content") to perform anomaly detection, which we utilize as a conceptional reasoning strategy for anomaly detection. Huang *et al.* [20] extracted procedural and conceptional rules for both log parsing and anomaly detection. Specifically, the procedural reasoning strategy for log parsing focuses on examining specific variable types such as "replacing IP addresses with..." or "replacing time indicators with..."; while the conceptional strategy abstract several patterns for being recognized as variables, such as "digit tokens" and "path-like token". These prior knowledge from practice can serve as a reasoning paradigm for log analysis.

Reasoning Strategies for Log Interpretation, Root Cause Analysis and Solution Recommendation (IRS). The IRS sub-tasks are not as well-studied as the two tasks above in automated log analysis. However, interpreting log content, finding root cause and employing solutions are essential skills for O&M engineers in practice. Therefore, we seek wisdom from existing software O&M manuals which

Table 1: Statistics on Reasoning Templates and Log Reasoning Dataset.

Tasks[a]	# Reasoning Templates	Template Source	# X→(R,Y) Samples	Avg. Len. of R	Domain
L.P.	2	Extracted Rules	200	128.97	HDFS
			200	105.70	Hadoop
			200	83.14	Zookeeper
			200	61.66	BGL
			200	80.41	HPC
			200	114.80	Linux
			200	124.64	Proxifier
A.D.	3	Extracted Rules	194	98.28	BGL
			138	102.98	Spirit
L.I.	2	Manuals & Interviews	300	257.47	
R.C.	3		300	320.97	Apache
S.R.	3		300	344.76	
All	13		2632	171.01	-

[a] **L.P.**, **A.D.**, **L.I.**, **R.C.**, **S.R.** represent the sub-tasks of log parsing, anomaly detection, log interpretation, root cause analysis, solution recommendation.

provide strategic guidelines for engineers. An official O&M guidance document [14] stated recommended steps for engineers to locate root causes, from recognizing key problems in logs, determining direct cause, to tracing back to the root cause. Kent *et al.* [24], in a log management guidance, suggested thinking strategies for proposing practical solutions for logged errors. The strategy involves self-reflection, proposing several possible corrective actions and assessing them to determine a optimized solution.

In addition, we also make interviews with a small group of O&M practitioners from Huawei to share their reasoning strategies for the IRS scenarios. Each interview usually lasts 20 to 30 minutes, the interviewee is queried to share their experiences in solving triggered system problems, from interpreting error logs, locating root causes, to taking mitigating actions. From their shared cases and experiences, both conceptional and procedural thinking strategies are summarized by authors. For instance, the conceptional thinking strategy for the sub-task of log interpretation involves contextualizing with broader scenario, emphasizing general implications of logs and relating the logs with system environments before generating the final interpretations; while the procedural strategy firstly searches for existence of specific error message, classifies the error categories and examining attributes such as timestamp or variables.

3.2 Construction of Log Reasoning Dataset

3.2.1 Human Reasoning Templates for Log Analysis. Upon acquiring the reasoning strategies, we crafted them into reasoning templates with a unified format of "Step 1:..." to "Step n:.." until reaching the analysis goals. As shown in Fig. 2, each sub-task is equipped with at least one conceptual and one procedural thinking strategy, ensuring the diversity of our reasoning templates. Statistics and sources of these templates are shown in Table 1. For a sub-task S_i, we denote the template set for this sub-task as T_i, $i \in [1, 5]$. These reasoning templates reflect how human O&M engineers reasons in different log analysis scenarios and guide through our curation of the instantiated reasoning trajectories in Log Reasoning Dataset.

3.2.2 Instantiating Templates to Reasoning Trajectories for Real-world Logs. To facilitate the learning of human thinking patterns for LLMs, we instantiated these curated templates into diverse reasoning trajectories handling real-world logs. In other words, under a specific analysis task, the specific reasoning process (R) to reach task-expected outputs (Y) given real-world logs (X), is generated following the reasoning steps in templates. To ensure a comprehensive coverage of various analysis scenarios, we use the open-source training dataset from LogLM [33] as the initial data source. This dataset contains a total of 2600+ (X,Y) samples, encompassing five distinct log analysis sub-tasks, with real-world logs from diverse domains such as supercomputers, distributed systems, operating systems, and software applications. For every (X,Y) sample of sub-task S_i in the source dataset, the goal of the instantiation is to generate the reasoning trajectory R from X to Y, under the guidance of a random template $t \in T_i$.

Following recent studies [16, 32], we automation this process using advanced LLMs by a role-playing prompt [25], which assumes the LLM to be an O&M engineer who needs to analyze a given log to reach the expected conclusion and asks the LLM to output the inner monologue of the engineer guided by the reasoning template. The prompts are slightly fine-tuned to suit different sub-tasks, and the version for anomaly detection is as follows:

> Assume that you are a DevOps engineer with extensive experience in log analysis and a strong ability to detect anomalies in logs. Now you have both the unstructured log and its labeled status (normal or abnormal), Your task is to construct the entire analysis process from Original Log to Log Label into an inner monologue, based on the following Reasoning Guidance: {Reasoning Template}. You must strictly follow Reasoning Guidance step by step without skipping steps and output the chain-of-thought trajectory from Original Log to Log Label. Note that the monologue should purely be starting from the Original Log without leaking any information in the labeled status, since it reflects the step-by-step internal mental activity of the engineer who doesn't know the answer at first. The reference to the guidance should be specific combining the input log, avoiding using vague phrases like 'according to Step *'. Make sure every reasoning step in the monologue exists in the actual given log characteristics and do not propose anything outside it. Original Log: {Log X}; Log Label: {Label Y}.

Statistics of the final Log Reasoning Dataset is shown in Table 1. Guided by the curated 13 human-aligned reasoning templates, 2600+ distinct reasoning trajectories R are generated from (X,Y) pairs, with an average length of 171.01 words. Notably, analysis output Y for the three IRS tasks is generally longer than the sub-task of log parsing and anomaly detection (*i.e.*, chunks of analytic words *v.s.* a parsed template or a conclusion of normal/abnormal), thereby leading to a significantly longer reasoning trajectories R in average.

3.3 Training of R-Log

3.3.1 Stage I: Cold Start on Log Reasoning Dataset. Recent studies on reasoning-based LLMs reveal the importance of initial policy

at the beginning of RL [8, 16], which not only facilitates the convergence of policy, but also pose a significant impact on the final learnt policy through RL. The human prior knowledge in our Log Reasoning Dataset can provide a sound paradigm for the model to imitate during the cold-start phase, where the model acquires its initial reasoning abilities on log analysis tasks and serves as a sound initial policy for the RL stage.

To achieve this, we simply perform a token-by-token SFT on the foundation model θ, using the Log Reasoning Dataset, denoted by D. For the convenience of reward computing in the RL stage, R and Y are wrapped by two pairs of special tokens into "<think>R</think>" and "<answer>Y</answer>" in the desired outputs to control the format. The learning goal of the cold-start phase becomes:

$$\theta_c = \arg\max_{\theta} \sum_{(X,Y,R)\in D} \log P(\,[R;Y]\mid[I;X],\theta), \quad (1)$$

where I is a short task-specific descriptional instruction accompanying every (X,Y) sample in the source dataset (e.g., "Find the root cause of errors in the following log.") and $[*;*]$ means concatenation of two strings.

3.3.2 Stage II: RL on Joint Rewards in Heterogeneous O&M Environment. In the second stage, we further optimize the cold-started policy θ_c through RL to enhance its reasoning capabilities in a heterogeneous O&M environment. By reusing (X,Y) samples in the Log Reasoning Dataset, this environment is characterized by its diversity, simulating system-side events encompassing five distinct log analysis sub-tasks (as denoted by S_i for $i \in [1,5]$) and logs sourced from various domains. The agent (i.e., the LLM with an initial policy θ_c) interacts with the environment by taking an action (i.e., generating a reasoning trajectory R' and final answer Y') based on the current state (the input log X and its accompanying instruction I simulating system-triggered events). The environment then provides a reward signal r to guide the policy optimization. For the specific parameter updating strategy, we employ the GRPO algorithm (as introduced in Section 2.2), which demonstrates effective in various complex reasoning scenarios [8, 16, 60], by employing group-based advantage estimation. The GRPO algorithm first samples multiple answers from the policy for the same input, compute reward r for each answer, and encourages the answers with a higher relative reward among the group.

The joint reward function r is designed as follows to assess the quality of the agent's generated response to various events in the simulated environment:

$$r = \begin{cases} -20 \cdot \omega_f & \text{if } \delta_f = 0 \\ \omega_f + \text{F1-score}(Y,Y') \cdot \delta_v + \\ \quad (1-\delta_v)(1 - \dfrac{\text{EditDistance}(Y,Y')}{\max(|Y|,|Y'|)}) & \text{if } i = 1 \\ \omega_f + \mathbb{1}(Y'=Y) & \text{if } i = 2 \\ \omega_f + \dfrac{\text{BLEU}(Y,Y') + \sum_{n\in 1,2,L}\text{ROUGE-}n(Y,Y')}{400} & \text{if } i \in 3,4,5. \end{cases}$$

$$(2)$$

The first term is a format checking that penalizes structural errors. If the response strictly adheres to the required structure, containing both a reasoning part enclosed in "<think>...</think>" and a non-empty answer part enclosed in "<answer>...</answer>", δ_f is 1 and the model will receive a small reward. If the format

checking fails, δ_f becomes 0 and a severe penalty is applied. ω_f is a hyperparameter controlling weights of the format reward. Following He *et al.* [16], we set ω_f to be 0.1, a small value to avoid distraction from answer quality. All answer rewards are normalized to $[0,1]$ to ensure consistent optimization across tasks, making the overall range of r to be $[-2, 1.1]$.

Upon passing format checking, r is determined by evaluating the extracted answer Y' based on the specific sub-task S_i:

(1) $i = 1$ (Log Parsing). This term encourages accurate recognition of both variable and template parts in Y'. δ_v is 1 if the ground-truth template Y contains variables (i.e.,"<*>") and becomes 0 otherwise. The F1-Score is computed by extracting variables from Y and Y' (treating each variable as a token) and calculating the binary F1-Score with "variable" as the positive label, which evaluates the accuracy of variable extraction in a range of $[0,1]$. For template parts, we use edit distance [28] and normalize it to $[0,1]$ to measure the similarity between the predicted and ground-truth templates.

(2) $i = 2$ (Anomaly Detection). The reward value for this sub-task is binary ($\{0,1\}$): it rewards only if the predicted answer Y' is either 'normal' or 'abnormal' and matches the ground truth Y. $\mathbb{1}$ is the indicator function, reflecting the binary nature of this sub-task.

(3) $i = 3, 4, 5$ (IRS tasks). This term seeks to comprehensively evaluate correctness of semantics in Y' by a combination of BLEU [43] and ROUGE [31] scores. BLEU measures the n-gram (i.e., n consecutive words) precision between Y and Y', emphasizing lexical overlap. ROUGE-1, ROUGE-2, and ROUGE-L measure recall based on unigram (i.e., single word), bigram (i.e., two consecutive words), and longest common subsequence, respectively, assessing content similarity and semantic fluency. These metrics are the most typical ones for comprehensively evaluating semantic correctness in generative text outputs [2, 12, 55] like the IRS tasks' outputs. The scores are normalized by 400 (the maximum possible sum where each metric is within $[0,100]$).

This joint reward function ensures that the model is penalized for structural errors while being rewarded for semantic accuracy tailored to each sub-task's requirements, facilitating effective learning in the heterogeneous O&M environment. Note that we didn't impose any reward signal directly on the model-generated reasoning trajectory R', since there might be distinct reasoning paths to reach the reference answers. The initial policy θ_c is equipped with a human-aligned reasoning paradigm and is expected to self-adapt its strategies according to signals received from the rewards on answers through the RL stage, converging to θ_R, i.e., R-Log.

4 Experiment

4.1 Implementation Details

Following recent studies [16, 60], Qwen2.5-7B [59] is selected as the foundation model θ due to its steady performance among open-source LLMs. Our main implementation of R-Log, denoted by R-Log-7B, was trained through the two stages according to Eq. (1) and (2). For the code-start phase and the implementation of baselines, SFT is performed using the framework of LLaMAFactory [63]. We keep the hyperparameters of SFT the same as LogLM [33], training 480 steps with a learning rate of 2×10^{-5} and a batch size of 32. For the RL phase, VeRL [48] is utilized as the framework. Following He *et*

Table 2: Comparing R-Log with Existing Methods on the Sub-task of Log Parsing.

Methods	HDFS		Hadoop		Zookeeper		BGL		HPC		Linux		Proxifier		Avg.	
	RI[a]	F1	RI	F1	RI	F1	RI	F1	RI	F1	RI	F1	RI	F1	RI	F1
Spell [9]	0.871	0.000	0.721	0.058	0.102	0.045	0.503	0.536	0.882	0.000	0.706	0.091	0.621	0.000	0.629	0.104
Drain [17]	0.914	0.389	0.647	0.068	0.787	0.225	0.822	0.397	0.119	0.002	0.695	0.225	0.822	0.500	0.687	0.258
LogParse [39]	0.907	0.632	0.349	0.502	0.982	0.348	0.992	0.665	0.194	0.330	0.825	0.588	0.490	0.334	0.677	0.486
LogStamp [53]	0.954	0.523	0.927	0.594	0.992	0.275	0.984	0.818	0.949	0.434	0.760	0.658	0.811	0.438	0.911	0.534
Qwen2.5-7B-Instruct[59]	0.923	0.832	0.896	0.583	0.946	0.767	0.914	0.482	0.938	0.682	0.810	0.768	0.683	0.833	0.870	0.707
OWL-7B [15]	0.741	0.233	0.779	0.178	0.742	0.115	0.706	0.049	0.635	0.023	0.668	0.042	0.591	0.267	0.695	0.130
LogLM-7B [33]	0.878	0.815	0.857	0.671	0.910	0.709	0.859	0.545	0.887	0.508	0.796	0.866	0.568	0.799	0.822	0.702
Qwen2.5-7B-SFT	0.900	0.706	0.902	0.768	0.936	0.809	0.988	0.839	0.949	0.519	0.773	0.723	0.559	0.759	0.861	0.732
LogPrompt [35]	0.890	0.863	0.879	0.763	0.948	0.889	0.964	0.865	0.934	0.759	0.758	0.766	0.567	0.653	0.849	0.794
SuperLog-7B [23]	0.978	0.977	**0.982**	0.942	0.998	0.815	0.976	0.700	0.974	0.727	**0.999**	0.914	**0.998**	0.938	0.986	0.859
R-Log-7B	**0.995**	**0.998**	0.945	**0.943**	**0.999**	**0.993**	**0.996**	**0.937**	**0.996**	**0.954**	0.989	**0.922**	0.991	**0.945**	**0.987**	**0.956**

[a] **RI** stands for RandIndex. **F1** stands for variable-level F1-score.

al. [16], the GRPO training configuration consists of 480 training steps, a learning rate of 4×10^{-7}, a batch size of 16, and 8 rollouts.

4.2 Research Questions & Key Findings

We empirically studied the following research questions (RQ) and report key findings for each RQ.

RQ1: Can R-Log outperform existing methods, especially LLM-based methods, in diverse log analysis tasks?

Key Findings of RQ1: In Section 4.3, we compare R-Log with various approaches encompassing task-specialized approaches and general-purpose LLMs, on open-source log analysis benchmarks across five sub-tasks. R-Log exhibits strong performance against existing approaches, highlighting its strong application potential in various log analysis scenarios.

RQ2: Does R-Log benefit from its reasoning-based training paradigm with RL?

Key Findings of RQ2: In Section 4.4, we conduct an ablation study on training stages of R-Log. The results indicate that incorporating only the first stage (*i.e.*, SFT with both R and Y) improves model performance significantly against plain SFT (only with Y), while the RL stage further improves performance over the first stage. This finding substantiates the benefits of our proposed reasoning-based paradigm with RL.

RQ3: Are the human-aligned reasoning strategies important in training R-Log?

Key Findings of RQ3: In Section 4.5, we examine the importance of human reasoning strategies by eliminating the reasoning templates from the dataset construction process (*i.e.*, the reasoning trajectories are generated without any guidance). The result reveals a significant advantage of using the human guidance over free-style generation on task performances, highlighting the importance of human O&M experience in building reasoning ability for R-Log.

RQ4: Can R-Log generalize to new analysis scenarios?

Key Findings of RQ4: We examine the performance of R-Log and LogLM on a new sub-task unseen from training. The result indicates that R-Log significantly outperforms LogLM in the challenging new scenario, highlighting its strong generalization ability to solve unseen problems via step-by-step reasoning.

RQ5: How to balance between efficacy and efficiency for R-Log?

Key Findings of RQ5: The "think-before-answer" nature of R-Log inevitably sacrifices efficiency due to the long reasoning trajectories. In Section 4.7, we experimented with an interesting alternative: changing nothing except swapping the output order in training phase to be "answer before think". By using this mode, R-Log first outputs a short answer and can stop when the "<think>" token appears (by setting it as an end token), achieving exact the same speed with non-thinking LLMs. Empirical result suggests only a minor performance drop for the "reversed" version while saving time, indicating a promising trade-off in actual application.

4.3 RQ1: Benchmarking on Log Analysis Tasks

4.3.1 Evaluation Datasets. We evaluate the performance of R-Log using the open-source benchmark from LogLM [33], encompassing the five log analysis sub-tasks and human-inspected reference answers. All the logs in the test suites are from real-world scenarios, encompassing 10 distinct domains. For log parsing and anomaly detection, the templates [18] and anomaly labels [41] are manually annotated by domain experts. In addition, we applied the improved annotations for anomaly detection from Xu *et al.* [58], who identified and corrected around 10% of the annotation errors in the original datasets by [41]. For the IRS tasks, the logs are from user posts in technical forums and the reference answers are the highest voted answers [56] with human examinations by [33]. The ratio between training samples and testing samples is 1:9 for log parsing and anomaly detection and 8:2 for IRS tasks, with strict separation between the sets to avoid data leakage [33]. All baselines (if trainable) were trained using the same amount of task-specific training data to ensure fair comparison.

4.3.2 Baselines. We compare R-Log with the following three groups of baselines for comprehensiveness: **(1) Task-specific methods**, including algorithms and models targeted for a single sub-task such as Spell [9], Drain [17], LogParse [39] and LogStamp [53]; **(2) General-purpose LLMs**, including various powerful LLMs with strong language processing ability such as LLaMA series [11] and Qwen-2.5-7B-Instruct [59] (the official chatting model of the Qwen-2.5 series); **(3) Domain-specialized LLMs**, including LLMs that incorporated domain knowledge for multiple sub-tasks within

Table 3: Benchmarking with Existing LLMs on Log Interpretation, Root Cause Analysis, and Solution Recommendation.

Methods	Log Interpretation				Root Cause Analysis				Solution Recommendation			
	BLEU	R-1[a]	R-2	R-L	BLEU	R-1	R-2	R-L	BLEU	R-1	R-2	R-L
LLaMA-3-70B [11]	0.507	7.984	2.121	5.864	0.172	6.876	1.273	4.382	0.529	7.640	1.557	5.368
LLaMA-3.1-405B [11]	4.466	28.563	11.106	17.071	1.416	12.984	3.694	8.162	2.018	18.768	5.287	12.045
Qwen2.5-7B-Instruct[59]	7.779	39.028	14.730	28.695	7.002	35.331	15.710	26.796	3.039	24.912	8.524	18.804
Qwen2.5-7B-SFT	2.041	33.852	14.140	9.289	1.605	34.210	13.586	8.873	2.032	17.474	5.588	13.084
OWL-7B [15]	2.566	28.211	8.289	19.123	1.947	20.893	5.671	14.718	0.953	21.620	5.006	15.574
SuperLog-7B [23]	4.521	32.938	12.104	20.277	3.576	27.713	8.812	17.102	3.467	27.104	7.655	17.890
LogLM-7B [33]	9.826	40.462	22.545	35.810	10.281	38.741	16.803	27.524	3.970	28.083	9.332	20.183
R-Log-7B	**16.671**	**48.593**	**24.754**	**43.880**	**13.064**	**41.453**	**19.518**	**36.492**	**6.759**	**33.587**	**12.838**	**29.925**

[a] **R-1** stands for ROUGE-1. **R-2** stands for ROUGE-2. **R-L** stands for ROUGE-L.

Table 4: Benchmarking on LLM-based Anomaly Detection.

Methods	BGL			Spirit		
	Pre[a]	Rec	F1	Pre	Rec	F1
Qwen2.5-7B-Instruct[59]	0.129	**0.934**	0.227	0.269	**0.949**	0.419
OWL-7B [15]	0.081	0.197	0.115	0.230	0.943	0.221
SuperLog-7B [23]	0.385	0.197	0.261	**0.778**	0.051	0.097
Qwen2.5-7B-SFT	0.429	0.723	0.539	0.314	0.699	0.433
LogLM-7B [33]	0.400	0.605	0.482	0.429	0.728	0.540
LogPrompt [35]	0.447	0.829	0.581	0.402	0.794	0.533
R-Log-7B	**0.493**	0.908	**0.639**	0.446	0.934	**0.603**

[a] **Pre**, **Rec**, **F1** stands for Precision, Recall and F1-score.

log analysis. This category includes LogPrompt [35] which drives advanced LLMs with specially-designed prompts for log analysis, OWL-7B [15] and SuperLog-7B [23] which trained 7B-sized LLMs with QA pairs related to IT operation and log analysis, and LogLM-7B [33] which utilizes instruction tuning to build multi-task log analysis ability for LLMs. We reproduced these models to ensure a fair comparison under the same experimental settings (*e.g.*, foundation model). In addition, we also fine-tuned the foundation model θ with the same in-domain data as R-Log for each sub-task, denoted by Qwen-2.5-SFT.

4.3.3 Log Parsing. Table 2 displays the benchmarking result for log parsing. Following existing studies *et al.* [33, 39, 52, 53], RandIndex [46] and variable-level F1-score [34] are utilized as the metric for measuring the accurate recognition of template and variable, respectively. RandIndex assesses the accuracy of log clustering (*i.e.*, whether two logs with the same template are accurately clustered together), regardless of the correctness of variables in the extracted templates. In contrast, F1-score focuses on successful identification of variables in logs, thereby serving as a more challenging metric. The result indicates that R-Log achieves a steadily strong performance on both template extraction and variable recognition, compared with existing methods which may struggle on this online scenario where majority of the logs are unseen from training (*i.e.*, a training-test ratio of 1:9).

4.3.4 Anomaly Detection. We report F1-score of anomalies as the metric for a comprehensive evaluation on the sub-task of anomaly

detection. The F1-score is computed as the harmonic average of Precision and Recall, where Precision measures the percentage of correctly classified ones in all of model's predicted anomalies, and Recall measures the recall rate of golden abnormal logs. An imbalance on Precision and Recall reflects failure of the system in anomaly detection and will lead to low F1-score. For instance, in Table 4, the high Recall and low Precision of the Qwen model means it classified most logs as abnormal, while the low Recall and high Precision of SuperLog in the dataset of Spirit means most logs are classified as normal. In contrast, R-Log achieves the highest F1-score, indicating its practical ability in detecting anomalies.

4.3.5 IRS Tasks. As shown in Table 3, R-Log continuously outperforms existing LLMs in both BLEU and Rouge scores. The advantages in BLEU indicates a more precise and readable answer with less hallucinated contents, while the high Rouge scores suggest a better recall of the key points in reference answers, such as failure causes in the sub-task of root cause analysis and mitigating steps in the sub-task of solution recommendation.

4.4 RQ2: Ablation Study on Training Stages

For this ablation study, three LLMs are compared under the same experimental settings, except the following differences: **(1) R-Log:** The exact R-Log-7B model with full cold-start and RL stages; **(2) Cold-start only:** This model didn't go through the RL stage and was only fine-tuned on Log Reasoning Dataset with (X,R,Y) samples; **(3) Plain-SFT only:** This model is trained with the same Log Reasoning Dataset, but only on (X,Y) samples without the reasoning trajectories. The scores reported in Fig. 3 are averaged across multiple evaluation datasets for the five sub-tasks (*e.g.*, averaging BGL and Spirit for anomaly detection). For log parsing and anomaly detection, we report the F1-score; for IRS tasks, we report the average the four semantic scores. **All models are aligned in its number of trained tokens** (by upsampling the training set of (2) and (3)) to offset benefits by just seeing more data in training.

As shown in Fig. 3(a), the advantage of "Cold-start only" over "Plain-SFT only" suggests the benefits of introducing reasoning-based paradigms into LLMs, where the step-by-step reasoning process facilitates the learning of problem-solving abilities in complex log analysis tasks compared with fitting only on the log-label pairs. However, as discussed in Section 1, SFT flattens the supervision signals to every word (*i.e.*, R and Y). Since R is significantly longer
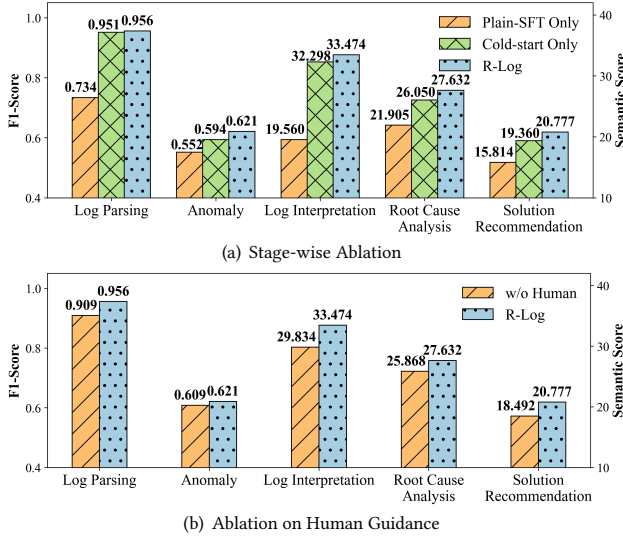
Figure 3: Ablation study on (a) training stages and (b) human reasoning templates. Scores are averaged across domains.

than Y, the model may fail to optimize its answer quality. After the RL stages, the model performance continuously advances on the five sub-tasks, suggesting the effectiveness of the reward signals on answers where the model further optimizes its reasoning paradigms by handling log events in a heterogeneous O&M environment. The improvement by RL in log parsing is less significant than other four sub-tasks, probably due to the relative easier nature of the task.

4.5 RQ3: Ablation Study on Human Strategies

We compare the following two groups: **(1) R-Log:** The exact RL-enhanced R-Log-7B model cold-started with human-aligned reasoning paradigm; **(2) W/o human:** The same setting with R-Log, with only difference in cold-start stage. Human-aligned reasoning guidance (*i.e.*, the templates) are removed from the prompt for instantiation of reasoning trajectories, resulting in a free-style generation of reasoning trajectories reflecting the advanced LLM's paradigm.

As shown in Fig. 3(b), without the incorporation of human-aligned reasoning templates, the performance continuously degrades in all sub-tasks. These reasoning templates comprehensively reflect diverse thinking strategies human engineers adopt in O&M practices, leading to high-quality reasoning trajectories with less diverted or hallucinated content than those purely generated by LLMs. These trajectories then provide a initial reasoning paradigm for R-Log, facilitating its further optimization through RL.

4.6 RQ4: Generalization on Unseen Task

We excluded *Log Variable Classification*, a challenging sub-task requiring extensive log-related knowledge as described in Section 2.3.1, from the training phase of R-Log to examine its generalization ability. The test set comprises 14000 logs from seven domains. Li *et al.* [30] manually classified each variable within these logs into the category of Object ID (OID), Location Indicator (LOI),

Table 5: F1-score of Variable Categories in an Unseen Task.

Methods	OID[a]	LOI	OBN	TDA	CRS	OBA
LogLM-7B [33]	0.226	0.269	0.037	0.166	0.101	0.031
R-Log-7B	**0.567**	**0.665**	**0.188**	**0.714**	**0.244**	**0.344**

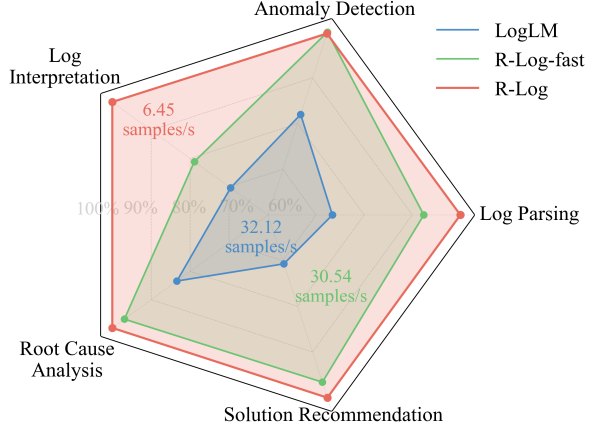[a] **OID**, **LOI**, **OBN**, **TDA**, **CRS**, **OBA** are variable classes.

Figure 4: "Think-before-answer" (R-Log) *v.s.* "Answer-before-think" (R-Log-fast), for a trade-off between efficacy and efficiency. The radar displays the relative percentage of baselines' average performances in comparison to R-Log's.

Object Name (OBN), Time/Duration of an Action (TDA), Computing Resources (CRS) and Object Amount (OBA), leading to a total of 29711 annotations of variable category. Our task instruction contains short descriptions of these categories and a random example as format guidance. We report F1-score of these categories as a metric for accurate classification of variables.

As shown in Table 5, despite not trained with any labels on variable categories, R-Log still demonstrates a strong performance compared with LogLM, especially on the categories of OID (0.567 *v.s.* 0.226), LOI (0.665 *v.s.* 0.269) and TDA (0.714 *v.s.* 0.166). Without training on the categories, the model must deduce variable types purely from the log context. By decomposing complex problems into intermediate steps through a step-by-step, human-like reasoning paradigm, R-Log generalizes its log analysis knowledge more effectively, leading to its advantage on this new task.

4.7 RQ5: Fastening via Answer-before-thinking

A recent study [26] on reasoning-based LLMs revealed an interesting phenomenon: the model may already "know" the answers at the beginning of its reasoning process. This inspired us to design a fastening version of R-Log (denoted as R-Log-fast), by switching the position of R and Y in the training dataset to induce an answer-first model. R-Log-fast first outputs answer Y, then outputs R, which can be intercepted by stopping at the "<think>" token, thereby avoiding the lengthy R and achieving the same speed as non-thinking LLMs.

As shown in Fig. 4, the performance reduction of R-Log-fast is tolerable compared with its efficiency enhancement (nearly 5x faster).

Despite using no reasoning at inference time, R-Log-fast, trained on full reasoning data with RL, still significantly outperforms LogLM, makes it a promising candidate for online deployments where latency is critical. We also noted the relatively significant performance reduction of R-Log-fast on log interpretation, which is consistent with Fig. 1, where small but critical details like "1ms" in this sub-task are easily missed without fully step-by-step reasoning.

5 Threats to Validity

Our study has several limitations:

(1) Extra Latency by Reasoning: Reasoning-based LLMs introduce extra latency that may burden online log analysis [37]. To alleviate this, we propose R-Log-fast with an 'Answer-first' strategy, fastening the inference while preserving performance, thereby achieving a practical balance between efficiency and accuracy.

(2) Limited Size of Foundation Model: Due to tight budget of resources, we didn't verify R-Log using larger foundation models, reducing the confidence of results. However, 7B is the most popular model size [6, 16, 36] for reasoning-based LLMs which balances between accuracy and speed in deployment.

(3) Fairness of Experimental Comparisons: In Section 4.3, discrepancies between R-Log's training data and baselines raised fairness concerns. However, fully unifying data is impractical due to proprietary, pre-trained baselines. To ensure fairness, in Section 4.4, we applied token alignment to offset extra-data effects. Experiments confirm R-Log remains superior.

6 Conclusion

In this paper, we present R-Log, a model that redefines LLM-based log analysis through a reasoning-first learning paradigm. By using RL with reasoning trajectories instead of SFT, R-Log learns the underlying rules of analysis. This method enables SOTA performance on known tasks and, more importantly, exceptional generalization to unseen problems by decomposing them into logical steps. This capability, combined with the verifiable trust offered by its transparent reasoning, was instrumental in its success for managing complex software systems. Future work include testing on more datasets, tasks and larger models, and refining rewards.

References

[1] Toufique Ahmed, Supriyo Ghosh, Chetan Bansal, Thomas Zimmermann, Xuchao Zhang, and Saravan Rajmohan. 2023. Recommending Root-Cause and Mitigation Steps for Cloud Incidents using Large Language Models. In *ICSE 2023*.

[2] Razieh Baradaran, Razieh Ghiasi, and Hossein Amirkhani. 2022. A survey on machine reading comprehension systems. *Natural Language Engineering* 28, 6 (2022), 683–732.

[3] Matthew Botvinick, Jane X Wang, Will Dabney, Kevin J Miller, and Zeb Kurth-Nelson. 2020. Deep reinforcement learning and its neuroscientific implications. *Neuron* 107, 4 (2020), 603–616.

[4] Yinfang Chen, Huaibing Xie, Minghua Ma, Yu Kang, Xin Gao, Liu Shi, Yunjie Cao, Xuedong Gao, Hao Fan, Ming Wen, et al. 2024. Automatic root cause analysis via large language models for cloud incidents. In *Proceedings of the Nineteenth European Conference on Computer Systems*. 674–688.

[5] Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V Le, Sergey Levine, and Yi Ma. 2025. SFT Memorizes, RL Generalizes: A Comparative Study of Foundation Model Post-training. In *Forty-second International Conference on Machine Learning*.

[6] Xu Chu, Zhijie Tan, Hanlin Xue, Guanyu Wang, Tong Mo, and Weiping Li. 2025. Domaino1s: Guiding llm reasoning for explainable answers in high-stakes domains. *arXiv preprint arXiv:2501.14431* (2025).

[7] Bill Curtis. 1984. Fifteen years of psychology in software engineering: Individual differences and cognitive science. In *Proceedings of the 7th international conference on Software engineering*. 97–106.

[8] DeepSeek-AI. 2025. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. In *arXiv preprint arXiv:2501.12948*.

[9] Min Du and Feifei Li. 2016. Spell: Streaming parsing of system event logs. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, 859–864.

[10] Min Du, Feifei Li, Guineng Zheng, and Vivek Srikumar. 2017. Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*. 1285–1298.

[11] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The Llama 3 Herd of Models. *arXiv preprint arXiv:2407.21783* (2024).

[12] Wafaa S El-Kassas, Cherif R Salama, Ahmed A Rafea, and Hoda K Mohamed. 2021. Automatic text summarization: A comprehensive survey. *Expert systems with applications* 165 (2021), 113679.

[13] Qiang Fu, Jian-Guang Lou, Yi Wang, and Jiang Li. 2009. Execution anomaly detection in distributed systems through unstructured log analysis. In *2009 ninth IEEE international conference on data mining*. 149–158.

[14] DOE Guideline. 1992. Root cause analysis guidance document. *US Department of Energy: Washington* (1992).

[15] Hongcheng Guo, Jian Yang, Jiaheng Liu, Liqun Yang, Linzheng Chai, Jiaqi Bai, Junran Peng, Xiaorong Hu, Chao Chen, Dongfeng Zhang, et al. 2024. OWL: A Large Language Model for IT Operations. In *The Twelfth International Conference on Learning Representations*.

[16] Minggui He, Yilun Liu, Shimin Tao, Yuanchang Luo, Hongyong Zeng, Chang Su, Li Zhang, Hongxia Ma, Daimeng Wei, Weibin Meng, et al. 2025. R1-t1: Fully incentivizing translation capability in llms via reasoning learning. *arXiv preprint arXiv:2502.19735* (2025).

[17] Pinjia He, Jieming Zhu, Zibin Zheng, and Michael R Lyu. 2017. Drain: An online log parsing approach with fixed depth tree. In *2017 IEEE international conference on web services (ICWS)*. IEEE, 33–40.

[18] Shilin He, Jieming Zhu, Pinjia He, and Michael R Lyu. 2020. Loghub: a large collection of system log datasets towards automated log analytics. *arXiv preprint arXiv:2008.06448* (2020).

[19] Falko Helm, Nico Daheim, and Iryna Gurevych. 2025. Token Weighting for Long-Range Language Modeling. In *Findings of the Association for Computational Linguistics: NAACL 2025*. 1440–1459.

[20] Xin Huang, Ting Zhang, and Wen Zhao. 2025. LogRules: Enhancing Log Analysis Capability of Large Language Models through Rules. In *Findings of the Association for Computational Linguistics: NAACL 2025*. 452–470.

[21] Yintong Huo, Yuxin Su, Cheryl Lee, and Michael R Lyu. 2023. SemParser: A Semantic Parser for Log Analytics. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE, 881–893.

[22] Robin Jeffries, Althea A Turner, Peter G Poison, and Michael E Atwood. 2013. The processes involved in designing software. In *Cognitive skills and their acquisition*. Psychology Press, 255–283.

[23] Yuhe Ji, Yilun Liu, Feiyu Yao, Minggui He, Shimin Tao, Xiaofeng Zhao, Su Chang, Xinhua Yang, Weibin Meng, Yuming Xie, Boxing Chen, Shenglin Zhang, and Yongqian Sun. 2025. Adapting Large Language Models to Log Analysis with Interpretable Domain Knowledge. In *Proceedings of the 34rd ACM International Conference on Information and Knowledge Management*.

[24] Karen Kent and Murugiah Souppaya. 2006. NIST SP 800–92, Guide to computer security log management. *National Institute of Standards and Technology (NIST)* (2006).

[25] Aobo Kong, Shiwan Zhao, Hao Chen, Qicheng Li, Yong Qin, Ruiqi Sun, Xin Zhou, Enzhi Wang, and Xiaohang Dong. 2023. Better zero-shot reasoning with role-play prompting. *arXiv preprint arXiv:2308.07702* (2023).

[26] Keito Kudo, Yoichi Aoki, Tatsuki Kuribayashi, Shusaku Sone, Masaya Taniguchi, Ana Brassard, Keisuke Sakaguchi, and Kentaro Inui. 2024. Think-to-talk or talk-to-think? when llms come up with an answer in multi-step reasoning. *arXiv preprint arXiv:2412.01113* (2024).

[27] Van-Hoang Le and Hongyu Zhang. 2022. Log-based Anomaly Detection with Deep Learning: How Far Are We?. In *2022 IEEE/ACM International Conference on Software Engineering (ICSE)*. IEEE, 1356–1367.

[28] Vladimir I Levenshtein et al. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, Vol. 10. Soviet Union, 707–710.

[29] Dacheng Li, Shiyi Cao, Chengkun Cao, Xiuyu Li, Shangyin Tan, Kurt Keutzer, Jiarong Xing, Joseph E Gonzalez, and Ion Stoica. 2025. S*: Test time scaling for code generation. *arXiv preprint arXiv:2502.14382* (2025).

[30] Zhenhao Li, Chuan Luo, Tse-Hsun Peter Chen, Weiyi Shang, Shilin He, Qingwei Lin, and Dongmei Zhang. 2023. Did We Miss Something Important? Studying and Exploring Variable-Aware Log Abstraction. In *ICSE 2023*.

[31] Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*. 74–81.

[32] Yilun Liu, Minggui He, Feiyu Yao, Yuhe Ji, Shimin Tao, Jingzhou Du, Duan Li, Jian Gao, Li Zhang, Hao Yang, et al. 2024. What Do You Want? User-centric Prompt Generation for Text-to-image Synthesis via Multi-turn Guidance. *arXiv preprint arXiv:2408.12910* (2024).

[33] Yilun Liu, Yuhe Ji, Shimin Tao, Minggui He, Weibin Meng, Shenglin Zhang, Yongqian Sun, Yuming Xie, Boxing Chen, and Hao Yang. 2025. Loglm: From task-based to instruction-based automated log analysis. In *2025 IEEE/ACM 47th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE, 401–412.

[34] Yilun Liu, Shimin Tao, Weibin Meng, Jingyu Wang, Wenbing Ma, Yuhang Chen, Yanqing Zhao, Hao Yang, and Yanfei Jiang. 2024. Interpretable online log analysis using large language models with prompt strategies. In *Proceedings of the 32nd IEEE/ACM International Conference on Program Comprehension*. 35–46.

[35] Yilun Liu, Shimin Tao, Weibin Meng, Feiyu Yao, Xiaofeng Zhao, and Hao Yang. 2024. Logprompt: Prompt engineering towards zero-shot and interpretable log analysis. In *Proceedings of the 2024 IEEE/ACM 46th International Conference on Software Engineering: Companion Proceedings*. 364–365.

[36] Zhaowei Liu, Xin Guo, Fangqi Lou, Lingfeng Zeng, Jinyi Niu, Zixuan Wang, Jiajie Xu, Weige Cai, Ziwei Yang, Xueqian Zhao, et al. 2025. Fin-r1: A large language model for financial reasoning through reinforcement learning. *arXiv preprint arXiv:2503.16252* (2025).

[37] Xiaoxue Ma, Yishu Li, Jacky Keung, Xiao Yu, Huiqi Zou, Zhen Yang, Federica Sarro, and Earl T Barr. 2025. Practitioners' expectations on log anomaly detection. *IEEE Transactions on Software Engineering* (2025).

[38] Adetokunbo AO Makanju, A Nur Zincir-Heywood, and Evangelos E Milios. 2009. Clustering event logs using iterative partitioning. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. 1255–1264.

[39] Weibin Meng, Ying Liu, Federico Zaiter, et al. 2020. Logparse: Making log parsing adaptive through word classification. In *2020 29th International Conference on Computer Communications and Networks (ICCCN)*. 1–9.

[40] Weibin Meng, Ying Liu, Yichen Zhu, et al. 2019. LogAnomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs.. In *IJCAI*, Vol. 19. 4739–4745.

[41] Adam Oliner and Jon Stearley. 2007. What supercomputers say: A study of five system logs. In *37th annual IEEE/IFIP international conference on dependable systems and networks (DSN'07)*. IEEE, 575–584.

[42] Jonathan Pan, Wong Swee Liang, and Yuan Yidi. 2024. RAGLog: Log Anomaly Detection using Retrieval Augmented Generation. In *2024 IEEE World Forum on Public Safety Technology (WFPST)*. IEEE, 169–174.

[43] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*. 311–318.

[44] Jiaxing Qi, Shaohan Huang, Zhongzhi Luan, Shu Yang, Carol Fung, Hailong Yang, Depei Qian, Jing Shang, Zhiwen Xiao, and Zhihui Wu. 2023. Loggpt: Exploring chatgpt for log-based anomaly detection. In *2023 IEEE International Conference on High Performance Computing & Communications, Data Science & Systems, Smart City & Dependability in Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys)*. IEEE, 273–280.

[45] Lingfei Qian, Weipeng Zhou, Yan Wang, Xueqing Peng, Jimin Huang, and Qian-qian Xie. 2025. Fino1: On the transferability of reasoning enhanced llms to finance. *arXiv e-prints* (2025), arXiv–2502.

[46] William M Rand. 1971. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association* 66, 336 (1971), 846–850.

[47] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300* (2024).

[48] Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2024. HybridFlow: A Flexible and Efficient RLHF Framework. *arXiv preprint arXiv: 2409.19256* (2024).

[49] Alexander Smith, William Martinez, Sophia Garcia, Benjamin Thomas, Olivia Davis, and Weimang Ye. 2024. Understanding Distribution Shift in LLMs: Methods, Evaluations, and Challenges. *Preprint on ResearchGate* (02 2024). doi:10.13140/RG.2.2.33962.32963

[50] Yongqian Sun, Weihua Kuang, Chao Shen, Xidao Wen, Tinghua Zheng, Heng Liu, Shenglin Zhang, Bo Wu, and Dan Pei. 2025. Enhancing Interpretability in Software Change Management with Chain-of-Thought Reasoning. *arXiv preprint arXiv:2507.09315* (2025).

[51] David Tall, Eddie Gray, Maselan Bin Ali, Lillie Crowley, Phil DeMarois, Mercedes McGowen, Demetra Pitta, Marcia Pinto, Michael Thomas, and Yudariah Yusof. 2001. Symbols and the bifurcation between procedural and conceptual thinking. *Canadian Journal of Math, Science & Technology Education* 1, 1 (2001), 81–104.

[52] Shimin Tao, Yilun Liu, Weibin Meng, Zuomin Ren, Hao Yang, Xun Chen, Liang Zhang, Yuming Xie, Chang Su, Xiaosong Oiao, et al. 2023. Biglog: Unsupervised large-scale pre-training for a unified log representation. In *2023 IEEE/ACM 31st International Symposium on Quality of Service (IWQoS)*. IEEE, 1–11.

[53] Shimin Tao, Weibin Meng, Yimeng Cheng, Yichen Zhu, Ying Liu, Chunning Du, Tao Han, Yongpeng Zhao, Xiangguang Wang, and Hao Yang. 2022. Logstamp: Automatic online log parsing based on sequence labelling. *ACM SIGMETRICS Performance Evaluation Review* 49, 4 (2022), 93–98.

[54] Alex Strick van Linschoten. 2024. The State of LLM Operations or LLMOps: Why Everything is Hard. https://www.zenml.io/blog/state-of-llmops-why-everything-is-hard.

[55] Haoran Wang, Yue Zhang, and Xiaosheng Yu. 2020. An overview of image caption generation methods. *Computational intelligence and neuroscience* 2020, 1 (2020), 3062706.

[56] Jiabo Wang, Guojun Chu, Jingyu Wang, Haifeng Sun, Qi Qi, Yuanyi Wang, Ji Qi, and Jianxin Liao. 2024. LogExpert: Log-based Recommended Resolutions Generation using Large Language Model. In *Proceedings of the 2024 ACM/IEEE 44th International Conference on Software Engineering: New Ideas and Emerging Results*. 42–46.

[57] Xu Wang, Sen Wang, Xingxing Liang, Dawei Zhao, Jincai Huang, Xin Xu, Bin Dai, and Qiguang Miao. 2022. Deep reinforcement learning: A survey. *IEEE Transactions on Neural Networks and Learning Systems* 35, 4 (2022), 5064–5078.

[58] Song Xu, Yilun Liu, Minggui He, Mingchen Dai, Ziang Chen, Chunguang Zhao, Jingzhou Du, Shimin Tao, Weibin Meng, Shenglin Zhang, Yongqian Sun, Boxing Chen, and Daimeng Wei. 2025. RationAnomaly: Log Anomaly Detection with Rationality via Chain-of-Thought and Reinforcement Learning.

[59] Qwen An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxin Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yi-Chao Zhang, Yunyang Wan, Yuqi Liu, Zeyu Cui, Zhenru Zhang, Zihan Qiu, Shanghaoran Quan, and Zekun Wang. 2024. Qwen2.5 Technical Report. *ArXiv* abs/2412.15115 (2024). https://api.semanticscholar.org/CorpusID:274859421

[60] Lingzhe Zhang, Yunpeng Zhai, Tong Jia, Chiming Duan, Siyu Yu, Jinyang Gao, Bolin Ding, Zhonghai Wu, and Ying Li. 2025. ThinkFL: Self-Refining Failure Localization for Microservice Systems via Reinforcement Fine-Tuning. *arXiv preprint arXiv:2504.18776* (2025).

[61] Shenglin Zhang, Weibin Meng, et al. 2007. Syslog processing for switch failure diagnosis and prediction in datacenter networks. In *IEEE/ACM 25th International Symposium on Quality of Service (IWQoS'17)*. 1–10.

[62] Xu Zhang, Yong Xu, Qingwei Lin, Bo Qiao, Hongyu Zhang, Yingnong Dang, Chunyu Xie, Xinsheng Yang, Qian Cheng, Ze Li, et al. 2019. Robust log-based anomaly detection on unstable log data. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 807–817.

[63] Yaowei Zheng, Richong Zhang, Junhao Zhang, YeYanhan YeYanhan, and Zheyan Luo. 2024. LlamaFactory: Unified Efficient Fine-Tuning of 100+ Language Models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, Yixin Cao, Yang Feng, and Deyi Xiong (Eds.). Association for Computational Linguistics, Bangkok, Thailand, 400–410. doi:10.18653/v1/2024.acl-demos.38

[64] Jieming Zhu, Shilin He, Jinyang Liu, Pinjia He, Qi Xie, Zibin Zheng, and Michael R Lyu. 2019. Tools and benchmarks for automated log parsing. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE, 121–130.