X-R2: Explainable Research Reasoning with Knowledge Graphs for Scientific Reasoning

Anonymous ACL submission

Abstract

Large language models (LLMs) are increasingly used for scientific question answering, yet genomics reasoning still suffers from limited reasoning capability and insufficient explainability. Moreover, under a controlled knowledge-increment setting, it remains unclear which knowledge channel is truly necessary to obtain correct answers. To address these challenges, we propose **X-R2**, an iterative reasoning framework that performs mandatory question decomposition given a query Q and optional document evidence $\mathcal{D}$, while explicitly separating three additional knowledge channels: (i) entity-centric knowledge, (ii) KG triple knowledge, and (iii) parametric knowledge. At each iteration, X-R2 extracts step-level entities and constraints, acquires evidence from the enabled channels, and triggers re-decomposition when a self-check detects missing or inconsistent support, thereby improving both accuracy and explainability.

To enable fine-grained attribution, we construct **X-R2 Bench**. Experiments on this benchmark show that X-R2 consistently improves performance over direct one-shot generation and confirm that instance-aligned KG triples provide the largest marginal gains.

1 Introduction

This paper focuses on improving the **accuracy** and **explainability** of large language models in knowledge-intensive genomics reasoning tasks. We propose **X-R2** (*e*xplainable *r*esearch *r*easoning with knowledge graphs for scientific reasoning), an iterative reasoning framework that performs multi-step inference over evidence, explicitly satisfies implicit constraints, and recovers missing intermediate facts when needed, thereby producing **correct and verifiable** answers.

A central challenge in genomics reasoning is that a question may be answerable only when the system can identify which knowledge source is decisive for correctness on that specific instance. However, existing reasoning and retrieval-augmented systems are often evaluated end-to-end as monolithic pipelines, making it difficult to conduct controlled analysis or attribute failures to missing evidence versus reasoning gaps.

To enable such controlled analysis, we construct **X-R2 Bench**, a genomics research reasoning benchmark that couples each question with (i) step-level decompositions, (ii) aligned KG(Knowledge Graphs) triples (KG triple knowledge), and (iii) curated entity-centric resources. Each instance can be executed under knowledge switches that selectively enable/disable access to entity-centric knowledge, KG triple knowledge, and parametric knowledge, thereby revealing when a method truly needs each component. We consider document evidence $\mathcal{D}$ when available and three additional knowledge channels: entity-centric knowledge, KG triple knowledge, and parametric knowledge. Reliable scientific reasoning requires not only access to these channels, but also an explicit mechanism to progressively expose and satisfy hidden constraints during inference.

Figure 1 illustrates why this design is necessary: compared with single-step parametric answering, one-shot KG-augmented reasoning, and unconstrained multi-step chain-of-thought reasoning, X-R2 introduces an inference-time control structure that progressively exposes constraints, acquires evidence step by step across knowledge channels, and revises the plan when evidence is insufficient or inconsistent, improving both accuracy and traceable attribution. Our contributions are as follows:

- **X-R2 reasoning framework.** We propose an iterative inference framework for scientific reasoning in which **question decomposition is mandatory**, followed by step-specific ex-

traction of entities and relations/constraints, controlled KG-triple retrieval and parametric-knowledge elicitation, step reasoning, and self-check-driven re-decomposition.

- **X-R2 Bench.** We introduce **X-R2 Bench**, a genomics research reasoning benchmark with step-level decompositions, aligned KG triple evidence, and curated entity-centric resources. The benchmark provides knowledge switches for controlled ablations across knowledge channels, enabling fine-grained attribution of when each source is necessary for correctness.

- **Knowledge weight analysis.** We conduct an empirical study comparing iterative reasoning with direct generation baselines under different knowledge availability settings, analyzing both accuracy and knowledge-source necessity.

2 Related Work

Multi-step reasoning and error accumulation. Recent work improves controllability by making inference more explicit via multi-step decomposition, iterative reasoning, or structured search (Trivedi et al., 2023; Yao et al., 2023; Shinn et al., 2023; Madaan et al., 2023; Lee et al., 2024b; Li et al., 2025a). A key limitation is error accumulation: early mistakes in decomposition or intermediate conclusions can propagate and dominate later steps. X-R2 addresses this by coupling stepwise reasoning with explicit self-checking and revision, enabling re-decomposition when the current state is unreliable.

External knowledge augmentation and harmful context. RAG and related approaches augment LLMs with external context, but retrieval can be noisy or irrelevant and may even hurt performance (Yoran et al., 2023; Maekawa et al., 2024; Shen et al., 2024; Hagström et al., 2025; Peng et al., 2025; Li et al., 2025a). This motivates stepwise evidence acquisition and robustness under retrieval errors. X-R2 retrieves/elicits knowledge step by step and revises the plan when evidence is insufficient or inconsistent.

Planning, iterative retrieval, and context sufficiency. To improve grounded reasoning, recent studies explore planning-then-generation and stepwise (iterative) retrieval for knowledge-intensive reasoning (Trivedi et al., 2023; Lee et al., 2024b; Lyu et al., 2024; Asai et al., 2024; Li et al., 2025b), and examine when retrieved context is sufficient or insufficient to support a correct answer, as well as how RAG systems should abstain or remain robust under retrieval errors (Maekawa et al., 2024; Yan et al., 2024; Hagström et al., 2025; Joren et al., 2025; Peng et al., 2025). However, these methods are still commonly evaluated as monolithic pipelines, making controlled knowledge inputs and fine-grained attribution difficult.

Which knowledge source matters in genomics reasoning? Beyond end-to-end accuracy, scientific QA benefits from identifying which knowledge source is necessary for correctness. Prior work studies parametric vs. non-parametric memories and evaluates grounding and context utilization (Mallen et al., 2023; Lee et al., 2024a; Shen et al., 2024; Hagström et al., 2025; Maekawa et al., 2024; Peng et al., 2025), and recent hybrid settings combine textual and relational evidence (Zhu et al., 2025; Lee et al., 2025). However, controlled attribution across entity-centric knowledge, KG triple knowledge, and parametric knowledge remains underexplored in genomics, partly due to the lack of benchmarks with aligned evidence and switchable knowledge access. We address this gap with X-R2 Bench and knowledge switches for instance-level necessity analysis.

3 Knowledge Taxonomy & Problem Setup

3.1 Knowledge Taxonomy

Scientific questions often hide multiple sub-goals and constraints. We treat the document context $\mathcal{D}$ (e.g., an abstract) as input evidence: it provides instance-specific textual evidence and should be prioritized for grounding when available. Beyond $\mathcal{D}$, we formalize three additional knowledge channels that can support inference and enable controlled attribution:

(i) Entity-centric knowledge provides lightweight signals for entity grounding, including canonical names, aliases/synonyms, entity types, and simple lexical or type constraints (e.g., gene vs. locus vs. phenotype). It is used to resolve mentions in $(Q, \mathcal{D})$ into stable entity identifiers and to constrain retrieval queries and step-level reasoning.

(ii) KG triple knowledge is relational evidence retrieved from an external knowledge graph in

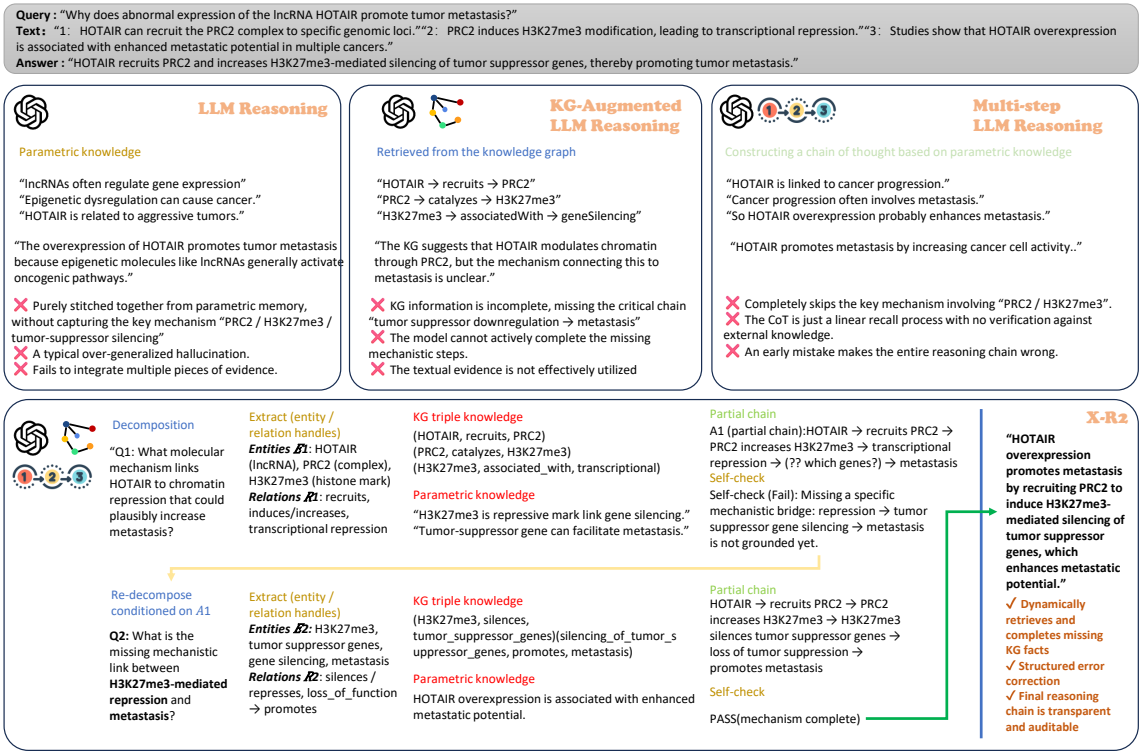

Figure 1: **Motivating example for X-R2.** Scientific reasoning often requires completing missing links between evidence and the final claim. Pure parametric answering may overgeneralize; one-shot KG augmentation can retrieve correct but disconnected triples; and vanilla CoT may drift without verification. X-R2 addresses these failure modes with mandatory decomposition, step-wise entity/relation extraction, controlled KG-triple retrieval and parametric-knowledge elicitation, and self-check-driven re-decomposition.

the form of triples (or short paths) anchored on grounded entities. It complements $\mathcal{D}$ by supplying structured relations (e.g., INTERACTS_WITH, CAUSES, ASSOCIATED_WITH) that may be missing, implicit, or scattered across the document evidence.

(iii) Parametric knowledge refers to background facts and patterns stored in model weights. While it can fill in missing intermediate facts when external evidence is incomplete, it is also the least verifiable channel; therefore, we treat it as a controlled, optional source that can be explicitly enabled/disabled (or elicited via a dedicated prompt) for analysis.

Together, $\mathcal{D}$ and the three channels define the knowledge space used by X-R2 and underpin the knowledge switches used throughout our controlled evaluation.

3.2 Problem Setup

We define the task and evaluation protocol shared by **X-R2** and all baselines.

Task. Given a query Q and an document context $\mathcal{D}$, the system produces a prediction $\hat{A}$. When document access is enabled ($s_{\text{DOC}}=1$), $\hat{A}$ should be grounded in $\mathcal{D}$ under the corresponding setting. Each instance also includes a reference answer $A^\star$, and performance is measured by comparing $\hat{A}$ against $A^\star$.

Available inputs. The mandatory input is Q, with $\mathcal{D}$ provided when available. In addition, an instance may include optional resources for controlled analysis: (i) entity-centric knowledges (canonical names, types, and aliases), and (ii) instance-aligned KG evidence as a small set of KG triples (or short paths). These resources are packaged by the benchmark. When KG access is enabled ($s_{\text{KG}}=1$), the system may also retrieve additional triples from an external KG store.

Output and evaluation format. We normalize $\hat{A}$ into a short-answer form (e.g., a list of canonical strings) for deterministic evaluation, and compute exact-match/normalized-match scores against $A^\star$. When evidence is insufficient under a given set-

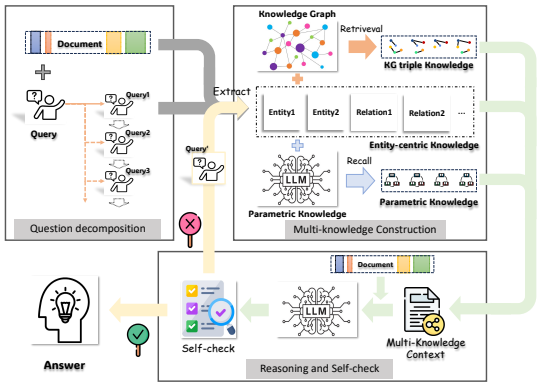

Figure 2: **X-R2 overview.** Given a query Q and optional document evidence $\mathcal{D}$, X-R2 performs mandatory question decomposition, X-R2 performs mandatory question decomposition to expose step-level subquestions. At each step, the model extracts entity/relation cues, retrieves KG triple knowledge from an external knowledge graph, and (optionally) elicits parametric knowledge via a controlled parametric-recall prompt. The model then conducts step reasoning and runs a self-check to decide whether to stop and output the final answer or to re-decompose and continue the next iteration.

ting, the model may abstain rather than speculate; under accuracy-style metrics, abstentions are counted as incorrect.

Controlled analysis via knowledge switches. To attribute behavior to different knowledge components, we evaluate each instance under binary knowledge switches that independently enable/disable access to: document evidence (s_{DOC}), entity-centric knowledge (s_{ENT}), KG triple knowledge (s_{KG}), and parametric knowledge (s_{PAR}). Comparing results across switch configurations reveals which knowledge source(s) are necessary for correctness on that instance.

4 Method

Figure 2 summarizes the end-to-end inference loop of X-R2. At a high level, X-R2 iterates over decompose $\rightarrow$ extract $\rightarrow$ retrieve/recall $\rightarrow$ reason $\rightarrow$ self-check. In each iteration, entity-centric signals guide KG retrieval and optional parametric elicitation, after which a self-check determines whether the current intermediate answer is sufficient or the system should re-decompose and continue.

4.1 Iterative Multi-step Inference

At iteration t with state A_{t-1} ($A_0 = \varnothing$), X-R2 executes: Decompose $\rightarrow$ Extract $\rightarrow$ RetrieveKG $\rightarrow$ Recall $\rightarrow$ Reason $\rightarrow$ Check. Concretely, Q_t, (E_t, R_t), KG_t, M_t, A_t, stop_t are produced in order from $(Q, \mathcal{D}, A_{t-1})$; if stop_t=1, the system returns $\hat{A}$=A_t, otherwise it continues with A_t and re-invokes decomposition.

Question decomposition. Given $(Q, \mathcal{D}, A_{t-1})$, the model produces a decomposition object

$$\mathsf{Decomp}_t = \{\texttt{subquestions}, \texttt{constraints}, \texttt{required\_evidence}, \texttt{current}\}, \quad (1)$$

where `current` specifies the step target Q_t. For $t = 1$, the model decomposes from $(Q, \mathcal{D})$. For $t > 1$, it re-decomposes conditioned on $(A_{t-1}, Q, \mathcal{D})$ to generate refined follow-up questions that address remaining gaps.

Step-specific extraction. Conditioned on $(Q_t, \mathcal{D})$, the model extracts step-specific entities E_t and relation/constraint cues R_t:

$$(E_t, R_t) = g(Q_t, \mathcal{D}). \quad (2)$$

Here R_t may include relation types, directions, conditions, required attributes, or comparison operators, serving as structured handles for downstream knowledge acquisition.

KG triple retrieval. Using (E_t, R_t) as retrieval intent, X-R2 collects KG triple knowledge KG_t as a compact set of triples (and optionally short paths) from an external knowledge graph.

Controlled parametric recall. In parallel to KG retrieval, X-R2 may elicit related parametric knowledge induced by (E_t, R_t):

$$M_t = h(E_t, R_t). \quad (3)$$

Operationally, M_t is obtained via a controlled parametric-recall prompt that encourages the model to surface relevant background knowledge while keeping it separable from retrieved KG triples and document evidence.

Step reasoning and self-check. The model produces an intermediate answer by reasoning over the combined context:

$$A_t = f(Q_t, \mathcal{D}, E_t, R_t, KG_t, M_t). \quad (4)$$

After generating A_t, the model performs a self-check:

$$\mathsf{stop}_t = \mathsf{Check}(A_t, Q, \mathcal{D}), \quad \mathsf{stop}_t \in \{0, 1\}, \quad (5)$$

Algorithm 1 X-R2: Iterative multi-step multi-source reasoning

Require: Query Q, document $\mathcal{D}$, max iterations $T_{\max}$
$A_0 \leftarrow \varnothing$
for $t = 1$ **to** $T_{\max}$ **do**
  $\mathsf{Decomp}_t \leftarrow \mathsf{Decompose}(Q, \mathcal{D}, A_{t-1})$ // mandatory
  $Q_t \leftarrow \mathsf{Decomp}_t.\mathsf{current}$
  $(E_t, R_t) \leftarrow \mathsf{Extract}(Q_t, \mathcal{D})$
  $KG_t \leftarrow (\mathsf{s\_KG} = 1)\ ?\ \mathsf{RetrieveKG}(E_t, R_t)\ :\ \varnothing$
  $M_t \leftarrow (\mathsf{s\_Par} = 1)\ ?\ \mathsf{RecallParametric}(E_t, R_t)\ :\ \varnothing$
  $A_t \leftarrow \mathsf{Reason}(Q_t, \mathcal{D}, E_t, R_t, KG_t, M_t)$
  $\mathsf{stop}_t \leftarrow \mathsf{Check}(A_t, Q, \mathcal{D})$
  if $\mathsf{stop}_t = 1$ **then**
    return $\hat{A} \leftarrow A_t$
  end if
end for
return $\hat{A} \leftarrow A_{T_{\max}}$

Algorithm 2 X-R2 Bench construction (decomposition + KG alignment)

Require: QA records $\mathcal{R}$, KG triples $\mathcal{G}$, top-K, max fix rounds F
Build an inverted index over normalized entity strings in $\mathcal{G}$
for each record $r \in \mathcal{R}$ **do**
  $(\mathsf{decomp}, \mathsf{seeds}, \mathsf{kg\_query}) \leftarrow \mathsf{LLM\_Extract}(Q, \mathcal{D})$
  $\mathsf{kg\_hits} \leftarrow \mathsf{RetrieveSubgraph}(\mathcal{G}, \mathsf{seeds}, K)$
  $x \leftarrow \mathsf{LLM\_Build}(r, \mathsf{decomp}, \mathsf{seeds}, \mathsf{kg\_hits})$
  if not $\mathsf{Validate}(x)$ **then** $x \leftarrow \mathsf{LLM\_Fix}(x)$ (repeat up to F)
  Write x to benchmark JSONL (preserving $A^\star$ exactly)
end for

deciding whether A_t sufficiently answers the original query and is consistent with $\mathcal{D}$. If $\mathsf{stop}_t = 1$, X-R2 terminates and outputs $\hat{A} = A_t$; otherwise it continues to iteration $t{+}1$ by re-invoking decomposition. We cap the maximum number of iterations $T_{\max}$; if reached, the system outputs the latest A_t.

Algorithm summary. Alg. 1 summarizes the iterative multi-step inference loop.

5 Benchmark Construction

To support controlled analysis of iterative multi-step reasoning with heterogeneous knowledge sources, we construct **X-R2 Bench**, where each instance is packaged with (i) an explicit decomposition and step trace, (ii) step-level entity/relation seeds for KG retrieval, and (iii) an instance-aligned KG evidence subgraph with provenance. The resulting instances are directly consumable by our inference scripts.

5.1 Data Sources and Target Format

We start from a QA dataset in JSONL format, where each record contains an identifier, a question Q, a document context $\mathcal{D}$ (in our setup, an abstract), and a reference answer $A^\star$. We also assume an external KG stored as JSONL triples with fields such as `subject`, `relation`, and `object`.

Each benchmark instance is a single JSON object with three core blocks:

- **Input/answer:** `input` (question and document) and `reference_answer`.
- **Reasoning interface:** `decomposition` and a step-aligned `step_trace` (same length), plus `entity_slot` for canonical entities and relation/constraint cues.
- **Aligned evidence:** `kg.subgraph` containing a sentence map for $\mathcal{D}$ and KG triples with provenance (doc sentence IDs or stable KG IDs).

We provide the complete schema, field definitions, and an example instance in Appendix A.

5.2 Construction Pipeline

Algorithm 2 summarizes a three-stage pipeline.

Stage 1 (LLM): decomposition and retrieval seeds. Given $(Q, \mathcal{D})$, an LLM produces a short decomposition and retrieval seeds (`seed_entities`, `seed_relations`) plus a concise `kg_query`.

Stage 2 (lexical): KG subgraph retrieval. We retrieve top-K KG triples by lexical matching over the KG store using the extracted seeds, and assign stable `kg_ids`.

Stage 3 (LLM): instance assembly with validation. We prompt an LLM with the schema, the QA record, and retrieved KG hits to produce a complete instance JSON, and validate structural constraints (e.g., `len(step_trace)=len(decomposition)`), optionally running a small fix loop.

6 Experiments

We evaluate (i) the effectiveness of our iterative multi-step reasoning procedure, (ii) its generalization across different LLM backbones, and (iii) controlled attribution across document evidence and three knowledge channels (entity-centric signals, KG triple knowledge, and parametric knowledge).

6.1 Benchmark and Tasks

We conduct experiments on **X-R2 Bench**, a genomics-oriented scientific reasoning benchmark built from paper abstracts. Each instance is a JSON object that includes an input question Q, an abstract $\mathcal{D}$, a reference answer $A^\star$, and structured fields used by our method, such as a decomposition plan (`decomposition`), step-level traces (`step_trace`), entity slots (`entity_slot`), and an aligned evidence subgraph (`kg.subgraph`) with `sentence_map` and KG/doc triples.

Scale. X-R2 Bench contains **410** instances in total (each instance corresponds to one task type paired with one abstract). We use a single held-out evaluation split (no train/dev/test partition), as our focus is on controlled inference-time analysis.

Task types. Each instance belongs to one of five task types (stored in the `task` field):

- `methods_and_techniques`: core methods/technologies used in the study (up to two).
- `research_field`: the research field/domain issue addressed by the study.
- `study_type`: the study type/angle (e.g., basic research, database development).
- `datasets`: datasets/cohorts used (e.g., TCGA, 1000 Genomes).
- `genes_and_loci`: explicitly mentioned genes or loci.

External KG. We use an external domain KG with **98,799** triples. When enabled, the system can retrieve triples from this KG using step-level seed entities/relations.

6.2 Models, Baselines, and Settings

Models. We evaluate three representative LLMs: **Qwen3-8B** (Yang et al., 2025), **DeepSeek-V3** (DeepSeek-AI et al., 2024), and **ChatGPT-5.1** (OpenAI, 2025a,b). We use each model through a unified prompting interface with deterministic decoding (below).

Methods compared.

- **Direct (one-shot).** A single-pass baseline that answers Q given the enabled context blocks.
- **Iterative (ours).** Iterative multi-step reasoning in `dynamic` mode, producing step-level subquestions Q_t, intermediate answers A_t, and a termination decision.

Deterministic decoding. All runs use `temperature=0` and `top_p=0`, with fixed settings for reproducibility.

Iteration and KG context budget. Unless otherwise stated, we set the maximum number of iterative steps to $T_{\max} = 5$ (via `-max_steps 5`) and truncate the KG triples fed into the prompt to top-$K = 5$ triples (via `-max_kg_triples 5`). This keeps the evidence budget small and emphasizes step-wise evidence selection.

Controlled switches. Each instance includes four binary knowledge switches: `s_Doc` (document evidence), `s_Ent` (entity-centric resources), `s_KG` (KG triple knowledge), and `s_Par` (parametric knowledge). When `s_Par=1`, we enable a controlled parametric-recall prompting mechanism to elicit parametric knowledge; otherwise the parametric channel is disabled. Our implementation supports per-run overrides of these switches, which we use for ablations and bucket attribution (Sec. 6.5–6.7).

6.3 Evaluation Metrics

Accuracy (task-aware matching). We follow the exact matching logic implemented in our evaluation script. Both gold and predictions are normalized by lowercasing and whitespace cleanup. Predictions are required to be `list[str]` (a JSON array of strings). Correctness is computed by task:

- **`genes_and_loci`, `datasets`, `methods_and_techniques`:** we mark an instance correct iff the predicted set is non-empty and covers the gold set (i.e., $\text{Gold} \subseteq \text{Pred}$).
- **`research_field` and `study_type`:** we mark an instance correct iff there is non-empty overlap between normalized strings, or one side is a substring of the other after concatenation (robust to minor phrasing differences).

Abstain rate. We report the fraction of instances where the system abstains (empty `pred_answer` or explicit abstain field), which reflects evidence insufficiency under stricter switch settings.

Average steps. For the iterative method, we report the average number of executed steps until

termination, computed from the length of the produced `steps` / stop decision in the `dynamic` JSON output.

Bucketed accuracy. To attribute which knowledge channel is necessary, we define three buckets using controlled re-runs: `needs-KG` (KG triple knowledge), `needs-Entity` (entity-centric knowledge), and `needs-Parametric` (parametric knowledge). We report bucketed accuracy as well as the bucket sizes.

6.4 Main Results

Table 1 compares iterative reasoning with direct one-shot generation on X-R2 Bench. For ChatGPT-5.1, iterative reasoning improves overall accuracy from 61.8 to 68.9 (+7.1), while keeping abstention low (1–2%). The average number of executed steps is 2.84, indicating that the procedure typically converges early and does not require the full $T_{\max}$ budget.

Discussion. The consistent gain suggests that explicitly surfacing latent constraints via mandatory decomposition, and then selecting evidence step-by-step, mitigates overgeneralization in genomics QA. Notably, the abstain rate increases only slightly (+0.5), indicating that the higher accuracy is not obtained by simply refusing harder instances.

6.5 Ablation and Attribution via Controlled Switches

We perform ablations by overriding the four switches at inference time, while keeping decoding deterministic and budgets fixed. Specifically, we test: (i) disabling KG triple knowledge (`s_KG=0`), (ii) disabling entity-centric resources (`s_Ent=0`), (iii) disabling parametric knowledge (`s_Par=0`), (iv) reducing the iterative depth to $T_{\max} = 1$ (to isolate the effect of iteration beyond mandatory decomposition), and (v) **KG shuffled**—permuting retrieved triples across instances while preserving the same triple budget, to stress-test whether gains require correct instance-level KG alignment.

What contributes most? Disabling KG triple knowledge causes the largest drop (68.9 $\rightarrow$ 59.7, -9.2), larger than removing entity slots (-4.4) or parametric recall (-2.1). This indicates that, under a fixed evidence budget (top-K=5), correctly retrieved KG triples provide the strongest marginal benefit in X-R2 Bench.

Is KG alignment real or incidental? The **KG shuffled** control largely removes the gain (61.1 vs 68.9), even though the prompt still contains the same number of triples. This suggests the improvement is not a prompt-length artifact: it depends on instance-aligned KG evidence, supporting the correctness/validity of X-R2 Benchs KG alignment interface.

Bucketed accuracy. Using the controlled re-runs described in Sec. 6.3, we further report bucketed performance. This directly quantifies when correctness depends on KG triple knowledge, entity-centric knowledge, or parametric knowledge, under the same instance-aligned interface.

Interpretation. The largest bucket is `needs-KG` (186 instances), and it also exhibits the largest within-bucket gain (+16.8), consistent with the ablation that KG triples are the dominant contributor. The `needs-Entity` bucket remains substantial, highlighting that entity-centric grounding signals are often a prerequisite for effective KG retrieval and constraint satisfaction. Finally, `needs-Parametric` captures cases where safe background recall is still necessary (e.g., canonical expansions, common aliases, or domain conventions not present in the abstract/KG context under the top-K budget).

6.6 Generalization Across LLMs

To test whether improvements are model-specific, we evaluate both methods on three different LLMs: Qwen3-8B (Yang et al., 2025), DeepSeek-V3 (DeepSeek-AI et al., 2024), and ChatGPT-5.1 (OpenAI, 2025a,b). We keep the same evaluation protocol, $T_{\max} = 5$, and top-$K = 5$ KG context.

Discussion. We observe consistent gains of +7–8 points across all backbones, suggesting the benefit comes from the inference procedure (decomposition + step-wise evidence use) rather than a model-specific artifact. The largest absolute improvement appears on the weaker backbone (Qwen3-8B), consistent with the hypothesis that structured intermediate supervision at inference time offers stronger regularization when the base model is less reliable in direct one-shot extraction.

6.7 Why X-R2 Bench is Necessary

A core goal of this work is to attribute when correctness requires different knowledge access

Method	Overall Acc.	Abstain ↓	Avg. Steps	Δ vs Direct
Direct (one-shot)	**61.8**	**0.7**	–	–
Iterative (ours)	**68.9**	**1.2**	**2.84**	**+7.1**

Table 1: Main results on X-R2 Bench (410 instances) using ChatGPT-5.1.

Setting (ours)	Overall Acc.	Abstain ↓	Avg. Steps	Notes
Full (default)	68.9	1.2	2.84	Doc+Entity+KG+Param.
w/o KG	59.7	1.5	3.05	`s_KG=0`
w/o Entity	64.5	1.3	2.96	`s_Ent=0`
w/o Parametric	66.8	1.2	2.91	`s_Par=0`
KG shuffled	61.1	1.3	2.98	permute triples across instances
$T_{\max}$=1	63.2	0.9	1.00	no multi-step refinement

Table 2: Ablation results via switch overrides (ChatGPT-5.1). "KG shuffled" breaks instance-level KG alignment while keeping the same evidence budget.

Bucket	#Instances	Direct Acc.	Iterative Acc.
needs-KG	186	41.4	58.2
needs-Entity	98	52.0	61.5
needs-Parametric	51	39.2	47.8

Table 3: Bucketed accuracy by controlled switch attribution (ChatGPT-5.1). Remaining 75 instances are doc-sufficient under our protocol.

Model	Direct Acc.	Iterative Acc.	Δ
Qwen3-8B	44.6	52.8	+8.2
DeepSeek-V3	56.3	64.1	+7.8
ChatGPT-5.1	61.8	68.9	+7.1

Table 4: Cross-model results on X-R2 Bench under the same protocol ($T_{\max} = 5$, top-$K = 5$).

modes. This is difficult to do reliably without (i) **aligned KG triples** at the instance level and (ii) **controlled switches** that can disable document evidence, entity-centric knowledge, KG triples, or parametric knowledge without changing the task interface.

Aligned KG enables controlled attribution. Because each instance includes an aligned evidence subgraph (`kg.subgraph`) with provenance (`sentence_map` and KG IDs), we can toggle `s_KG` or `s_Ent` and re-run the same instances under identical prompting and budgets. This produces bucket labels such as `needs-KG`, `needs-Entity`, and `needs-Parametric` (Table 3) in a reproducible manner, which is not directly supported by standard QA benchmarks lacking KG alignment and switchable interfaces.

Benchmark validity via alignment stress-test. The **KG shuffled** control in Table 2 preserves the triple budget but destroys instance-level alignment. Its substantial degradation (68.9 → 61.1) empirically supports that the benchmarks KG signal is not incidental: correctly aligned KG triples are required to realize the full benefit, and incorrect alignment cannot be compensated for by simply providing more symbolic text.

Practical impact. Beyond reporting higher accuracy, X-R2 Bench allows us to answer why a method works: whether improvements come from better entity grounding, better retrieval of symbolic relational evidence, or safe use of background knowledge. This makes the evaluation actionable for designing reliable scientific reasoning systems.

7 Conclusions

We introduced **X-R2**, an iterative genomics reasoning framework that performs mandatory question decomposition and step-wise evidence use across document, entity-centric, KG-triple, and parametric knowledge channels. We also constructed **X-R2 Bench** with aligned evidence and binary knowledge switches (`s_Doc`, `s_Ent`, `s_KG`, `s_Par`) for controlled attribution. Experiments across three LLM backbones show consistent gains over direct one-shot generation, and ablations highlight the largest marginal benefit from instance-aligned KG triples.

Limitations.

Our controlled study centers on three knowledge channels (entity-centric signals, KG triples, and prompted parametric recall) and treats them as binary switches, which simplifies attribution but does not cover richer genomics knowledge resources, and the parametric channel remains the least verifiable. In addition, we evaluate on a single split of 410 abstract-based instances with task-aware normalized matching; this setting may not reflect performance on full-text evidence and can under-penalize over-prediction for list-style answers.

References

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2024. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *International Conference on Learning Representations (ICLR)*.

DeepSeek-AI, Aixin Liu, and 1 others. 2024. Deepseek-v3 technical report. *Preprint*, arXiv:2412.19437.

Lovisa Hagström, Sara Vera Marjanovic, Haeun Yu, Arnav Arora, Christina Lioma, Maria Maistro, Pepa Atanasova, and Isabelle Augenstein. 2025. A reality check on context utilisation for retrieval-augmented generation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 19691–19730, Vienna, Austria. Association for Computational Linguistics.

Hailey Joren, Jianyi Zhang, Chun-Sung Ferng, Da-Cheng Juan, Ankur Taly, and Cyrus Rashtchian. 2025. Sufficient context: A new lens on retrieval augmented generation systems. In *International Conference on Learning Representations (ICLR)*.

Chanhee Lee, Zhaojun Chen, Luke Zettlemoyer, Wentau Yih, and Hannaneh Hajishirzi. 2024a. How well do large language models truly ground? In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 2412–2437, Mexico City, Mexico. Association for Computational Linguistics.

Deokhee Lee, Jinyoung Kim, Hyunwoo Park, Jinyoung Lee, and Jinho Seo. 2025. HybGRAG: Hybrid retrieval augmented generation in textual-relational contexts. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 835–854, Vienna, Austria. Association for Computational Linguistics.

Myeonghwa Lee, Seonho An, and Min-Soo Kim. 2024b. PlanRAG: A plan-then-retrieval augmented generation for generative large language models as decision makers. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6537–6555, Mexico City, Mexico. Association for Computational Linguistics.

Jiacheng Li, Tianming Wang, Wenchang Xu, Yongfei Zhang, and Wenhu Chen. 2025a. R3-RAG: Learning to Retrieve, Reason and Refine for multi-hop question answering. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 9570–9587, Suzhou, China. Association for Computational Linguistics.

Yuan Li, Qi Luo, Xiaonan Li, Bufan Li, Qinyuan Cheng, Bo Wang, Yining Zheng, Yuxin Wang, Zhangyue Yin, and Xipeng Qiu. 2025b. R3-RAG: Learning step-by-step reasoning and retrieval for LLMs via reinforcement learning. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 10491–10507, Suzhou, China. Association for Computational Linguistics.

Yuanjie Lyu, Zihan Niu, Zheyong Xie, Chao Zhang, Tong Xu, Yang Wang, and Enhong Chen. 2024. Retrieve-plan-generation: An iterative planning and answering framework for knowledge-intensive LLM generation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 4683–4702, Miami, Florida, USA. Association for Computational Linguistics.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. *Preprint*, arXiv:2303.17651.

Seiji Maekawa, Hayate Iso, Sairam Gurajada, and Nikita Bhutani. 2024. Retrieval helps or hurts? a deeper dive into the efficacy of retrieval augmentation to language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 5506–5521, Mexico City, Mexico. Association for Computational Linguistics.

Alex Mallen, Yue Li, Haotian Lin, Tushar Khot, Bo Hu, Shanya Li, Zhu Yao, Dan Wang, Diyi Yang, Percy Liang, and 1 others. 2023. When not to trust language models: Investigating the effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9847–9863, Toronto, Canada. Association for Computational Linguistics.

OpenAI. 2025a. GPT-5.1: A smarter, more conversational ChatGPT. Accessed 2026-01-06.

OpenAI. 2025b. GPT-5.1 model documentation. Accessed 2026-01-06.

Xiangyu Peng, Prafulla Kumar Choubey, Caiming Xiong, and Chien-Sheng Wu. 2025. Unanswerability evaluation for retrieval augmented generation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8452–8472, Vienna, Austria. Association for Computational Linguistics.

Junhong Shen, Shuo Chen, Jie Ma, Jing Zhang, Jing Wang, and Chen Chen. 2024. Assessing "implicit" retrieval robustness of large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 8813–8830, Miami, Florida, USA. Association for Computational Linguistics.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Preprint*, arXiv:2303.11366.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10014–10037, Toronto, Canada. Association for Computational Linguistics.

Shi-Qi Yan, Jia-Chen Gu, Yun Zhu, and Zhen-Hua Ling. 2024. Corrective retrieval augmented generation. *arXiv preprint arXiv:2401.15884*.

An Yang, Qwen Team, and 1 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Preprint*, arXiv:2305.10601. NeurIPS 2023 camera-ready.

Ori Yoran, Tomer Wolfson, Ori Ram, and Jonathan Berant. 2023. Making retrieval-augmented language models robust to irrelevant context. *Preprint*, arXiv:2310.01558.

Xiangrong Zhu, Yuexiang Xie, Yi Liu, Yaliang Li, and Wei Hu. 2025. Knowledge graph-guided retrieval augmented generation. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 8912–8924, Albuquerque, New Mexico. Association for Computational Linguistics.

A X-R2 Bench Schema and Construction Details

This appendix provides the full instance schema, field definitions, and additional construction details omitted from the main text.

A.1 Full Benchmark Instance Schema

Each instance is a single JSON object (one line) with the following core fields (matching our construction and inference code):

- **Input:** `input.question` and `input.document` (with `doc_id`, `type`, `text`).
- **Reference answer:** `reference_answer`, preserved exactly as provided by the QA source.
- **Decomposition and trace:** a list of sub-questions (`decomposition`) and a per-step trace (`step_trace`), where `len(step_trace)=len(decomposition)`.
- **Entity-centric knowledge:** `entity_slot` containing canonical entities (with coarse types), relation/constraint cues, and context notes.
- **Aligned evidence subgraph:** `kg.subgraph` containing (a) `sentence_map` (sentence IDs → abstract sentences), (b) `triples` with provenance, and (c) `nodes` appearing in the triples.
- **Controlled switches:** `switches` with four binary flags `s_Doc`, `s_Ent`, `s_KG`, and `s_Par` to enable/disable access to the four knowledge sources (Sec. 3).

This packaging ensures that the reasoning script can (i) read document sentences from `kg.subgraph.sentence_map`, (ii) access KG evidence from `kg.subgraph.triples`, and (iii) respect per-instance switch settings.

A.2 Stage 3 Assembly: Evidence Provenance and Step Trace

In Stage 3, the LLM outputs a complete BenchmarkInstance JSON that includes:

- **Sentence map:** split the abstract into a small set of sentence IDs (e.g., `S0, S1, ...`) in `kg.subgraph.sentence_map`.
- **Triples with provenance:** add (a) doc-grounded triples with `source="doc"` and

prov="Sx", and (b) KG triples with source="kg" and prov=kg_id.

- **Step trace aligned to decomposition:** each step includes the current sub-question, extracted entities/relations, used sentence IDs, an intermediate answer, and a self-check decision.
- **Auxiliary fields (optional):** closed_book_recall (a few parametric claims with confidences), labels_for_evaluation (coverage targets and gold evidence sentence IDs), and a simple judge_rubric.

A.3 Validation Rules

To guarantee structural consistency, we validate each produced instance with:

- **Required keys:** presence of input, reference_answer, decomposition, step_trace, kg.subgraph.
- **Alignment constraint:** len(step_trace)=len(decomposition).
- **Type checks:** decomposition is a list of strings; step_trace is a list of step objects; kg.subgraph.triples is a list of triples with source and prov.
- **Provenance sanity:** doc-grounded triples use sentence IDs appearing in sentence_map; KG triples use stable kg_ids returned by retrieval.

When validation fails, we run a bounded fix loop (up to F rounds) by prompting the LLM with the error message and the partially formed instance.

A.4 Retrieval Details (Stage 2)

Stage 2 performs best-effort lexical retrieval using the extracted seed_entities and seed_relations: (i) exact match on normalized entity strings, (ii) fallback partial containment match over subjects/objects, and (iii) a relation-keyword filter when applicable. We keep the top-K hits (default K=40) and assign stable kg_ids for provenance.

B Example: From a Genomics QA Pair to an X-R2 Bench Instance

This appendix presents a concrete example showing how a raw genomics QA record is transformed into a benchmark instance in X-R2 Bench, together with the corresponding iterative reasoning trace visualized as a tree.

B.1 Raw QA Pair (Input Record)

ID. 54
Task. genes_and_loci
Document. abstract_54 (abstract)
Reference answer ($A^\star$).

```
["The article mainly studies BLAST",
"POG", "RNA"]
```

Document evidence ($\mathcal{D}$).

> POGs/PlantRBP (http://plantrbp.uoregon.edu/) is a relational database that integrates data from rice, Arabidopsis, and maize by placing the complete Arabidopsis and rice proteomes and available maize sequences into putative orthologous groups (POGs). Annotation efforts will focus on predicted RNA binding proteins (RBPs): i.e. those with known RNA binding domains or otherwise implicated in RNA function. POGs form the heart of the database, and were assigned using a mutual-best-hit-strategy after performing BLAST comparisons of the predicted Arabidopsis and rice proteomes. Each POG entry includes orthologs in Arabidopsis and rice, annotated with domain organization, gene models, phylogenetic trees, and multiple intracellular targeting predictions. A graphical display maps maize sequences on to their most similar rice gene model. The database can be queried using any combination of gene name, accession, domain, and predicted intracellular location, or using BLAST. Useful features of the database include the ability to search for proteins with both a specified domain content and intracellular location, the concurrent display of mutual best hits and phylogenetic trees which facilitates evaluation of POG assignments, the association of maize sequences with POGs, and the display of targeting predictions and domain organization for all POG members, which reveals consistency, or lack thereof, of those predictions.

B.2 Simulated X-R2 Bench Instance (JSON)

The following is a simulated benchmark instance that follows our X-R2 Bench schema and is directly consumable by our inference scripts.


```
{
  "id": 54,
  "q_index": 54,
  "task": "genes_and_loci",
  "input": {
    "question": "Does the study explicitly mention
        any specific genes or loci studied? Please
        list them.",
    "document": {
      "doc_id": "abstract_54",
      "type": "abstract",
      "text": "POGs/PlantRBP (http://plantrbp.
          uoregon.edu/) is a relational database
          that integrates data from rice,
          Arabidopsis, and maize by placing the
          complete Arabidopsis and rice proteomes
```

```
          and available maize sequences into
          putative orthologous groups (POGs).
          Annotation efforts will focus on
          predicted RNA binding proteins (RBPs): i.
          e. those with known RNA binding domains
          or otherwise implicated in RNA function.
          POGs form the heart of the database, and
          were assigned using a mutual-best-hit-
          strategy after performing BLAST
          comparisons of the predicted Arabidopsis
          and rice proteomes. Each POG entry
          includes orthologs in Arabidopsis and
          rice, annotated with domain organization,
           gene models, phylogenetic trees, and
          multiple intracellular targeting
          predictions. A graphical display maps
          maize sequences on to their most similar
          rice gene model. The database can be
          queried using any combination of gene
          name, accession, domain, and predicted
          intracellular location, or using BLAST.
          Useful features of the database include
          the ability to search for proteins with
          both a specified domain content and
          intracellular location, the concurrent
          display of mutual best hits and
          phylogenetic trees which facilitates
          evaluation of POG assignments, the
          association of maize sequences with POGs,
           and the display of targeting predictions
           and domain organization for all POG
          members, which reveals consistency, or
          lack thereof, of those predictions."
  }
},
"reference_answer": [],

"decomposition": [
  "Scan the abstract for explicit gene or locus
       identifiers (e.g., standard gene symbols,
       locus tags, accessions).",
  "Disambiguate biological entities (genes/loci)
       from methods, databases, species names, and
        general biological terms.",
  "If no explicit gene/locus is mentioned, output
       an abstention/empty list with evidence."
],

"entity_slot": {
  "canonical_entities": [
    {"name": "POGs/PlantRBP", "type": "database"},
    {"name": "putative orthologous groups (POGs)",
         "type": "concept"},
    {"name": "RNA binding proteins (RBPs)", "type"
         : "protein_class"},
    {"name": "Arabidopsis", "type": "species"},
    {"name": "rice", "type": "species"},
    {"name": "maize", "type": "species"},
    {"name": "BLAST", "type": "method"}
  ],
  "relation_constraint_cues": [
    {"cue": "explicitly mention", "type": "
         constraint"},
    {"cue": "specific genes or loci", "type": "
         target_type"},
    {"cue": "list them", "type": "output_format"}
  ],
  "notes": "Only accept explicit gene/locus
       identifiers; do not treat methods (e.g.,
       BLAST), concepts (POG), or general terms (
       RNA/RBP) as genes/loci."
},

"kg": {
  "subgraph": {
    "sentence_map": {
      "S0": "POGs/PlantRBP integrates data from
           Arabidopsis/rice/maize into putative
           orthologous groups (POGs).",
      "S1": "Annotation focuses on predicted RNA
           binding proteins (RBPs).",
      "S2": "POGs were assigned using a mutual-
           best-hit strategy after BLAST
           comparisons.",
      "S3": "Each POG entry includes orthologs,
           domain organization, gene models,
           phylogenetic trees, and targeting
           predictions.",
      "S4": "A graphical display maps maize
           sequences onto their most similar rice
           gene model.",
      "S5": "The database can be queried by gene
           name, accession, domain, location, or
           using BLAST."
    },
    "nodes": [
      {"id": "n_db", "name": "POGs/PlantRBP", "
           type": "database"},
      {"id": "n_pog", "name": "POGs", "type": "
           concept"},
      {"id": "n_blast", "name": "BLAST", "type": "
           method"},
      {"id": "n_rbp", "name": "RBPs", "type": "
           protein_class"}
    ],
    "triples": [
      {"kg_id": "doc_0001", "source": "doc", "prov
           ": "S0", "subject": "POGs/PlantRBP", "
           relation": "integrates_data_from", "
           object": "Arabidopsis/rice/maize"},
      {"kg_id": "doc_0002", "source": "doc", "prov
           ": "S2", "subject": "POGs", "relation":
            "assigned_by", "object": "
           mutual_best_hit_strategy"},
      {"kg_id": "doc_0003", "source": "doc", "prov
           ": "S2", "subject": "
           mutual_best_hit_strategy", "relation":
           "uses", "object": "BLAST_comparisons"},
      {"kg_id": "doc_0004", "source": "doc", "prov
           ": "S5", "subject": "POGs/PlantRBP", "
           relation": "query_by", "object": "
           gene_name/accession/domain/location/
           BLAST"}
    ]
  }
},

"switches": {"s_Doc": 1, "s_Ent": 1, "s_KG": 0, "
     s_Par": 0},

"step_trace": [
  {
    "t": 1,
    "Q_t": "Scan the abstract for explicit gene or
         locus identifiers.",
    "E_t": ["POGs/PlantRBP", "Arabidopsis", "rice"
         , "maize", "RBPs", "BLAST"],
    "R_t": ["explicit_gene_or_locus_identifier", "
         locus_tag", "accession"],
    "used_sentences": ["S0", "S3", "S4", "S5"],
    "used_kg_ids": ["doc_0004"],
    "M_t": [],
    "A_t": {"candidates": [], "note": "No explicit
         gene symbols, locus IDs, or accessions
         are present."},
    "self_check": {"sufficient_for_Q": false, "
         reason": "Disambiguate BLAST/POG/RNA/RBP
         vs genes/loci."}
  },
  {
    "t": 2,
    "Q_t": "Disambiguate genes/loci from methods,
         databases, and general terms.",
    "E_t": ["BLAST", "POGs", "RNA", "RBPs"],
    "R_t": ["is_method", "is_database_or_concept",
         "is_general_bio_term"],
    "used_sentences": ["S1", "S2", "S5"],
    "used_kg_ids": ["doc_0002", "doc_0003", "doc_0
         004"],
    "M_t": [],
    "A_t": {"decision": "BLAST is a method; POG is
         a grouping concept; RNA/RBPs are general
         terms, not specific genes/loci."},
    "self_check": {"sufficient_for_Q": true, "
         reason": "No explicit genes/loci are
         mentioned in the abstract."}
  },
  {
    "t": 3,
    "Q_t": "Output the final list of explicitly
         mentioned genes/loci (or abstain).",
    "E_t": [],
    "R_t": ["output_list", "abstain_if_none"],
    "used_sentences": ["S0", "S5"],
```

```
      "used_kg_ids": [],
      "M_t": [],
      "A_t": [],
      "self_check": {"sufficient_for_Q": true, "
          reason": "Empty list is consistent with
          the document evidence."}
    }
  ],

  "predicted_answer": []
}
```

B.3 Reasoning Trace as a Tree (with Iterative Augmentation and Correction)

```
Root: Q
 Q: Does the study explicitly mention any specific
      genes or loci studied? Please list them.
    Iteration t=1  (decompose  extract  reason
        check)
      Q1: Identify candidate "gene/locus-like"
          mentions in the abstract.
      + Context acquired
        +Evidence sentences: S2 ("BLAST comparisons
            "), S0 ("POGs/PlantRBP; POGs"), S1 ("
            RNA binding proteins")
        +Initial constraint: "must be explicit gene/
            locus identifiers"
      Extract (E1)
        E1 candidates: {BLAST, POG, RNA}   (may be a
            type confusion)
      Intermediate answer A1
        A1 (naive): [BLAST, POG, RNA]   (methods/
            concepts treated as genes/loci)
      Self-check C1 (verification trigger)
         Check-1: "Do candidates match gene/locus
             identifier patterns (symbols/locus
             tags/accessions)?"  NO
          Action: Re-decompose + add a disambiguation
              sub-goal

    Iteration t=2  (re-decompose conditioned on A1 +
        acquire missing info)
      Q2: Disambiguate candidates by entity type
          using document evidence.
      + New information introduced
        +Type rules: gene/locus must be (gene symbol
            | locus tag | accession); otherwise
            reject
        +Doc-grounded typing evidence:
            S2: "BLAST comparisons"  BLAST is a
                method
            S0: "putative orthologous groups (POGs)"
                  POG is a grouping concept
            S1: "RNA binding proteins"  RNA/RBPs are
                general biological terms/classes
      Correction (fix)
        Fix-1: BLAST  method (not gene/locus)
        Fix-2: POG    concept/grouping (not gene/
            locus)
        Fix-3: RNA    general term (not gene/locus)
      Intermediate answer A2 (revised)
        A2: No explicit genes/loci are mentioned in
            the abstract. Output  (abstain).
      Self-check C2
         Check-2: "Is the revised decision supported
             by explicit sentences?"  YES (S0/S1/
             S2)
         stop = 1

    Final output Â
       Â:  (no explicit gene/locus identifiers)
       Provenance: S0/S1/S2 (type-disambiguation
           evidence)
```