KSPLIT: WHAT TO LEARN AND WHAT TO RETRIEVE IN LLM–SLM COLLABORATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Large–small language model (LLM–SLM) collaboration enables SLMs to leverage the stronger knowledge and reasoning capabilities of LLMs while reducing dependence on repeated LLM assistance. Across different collaboration paradigms, LLM-provided information is ultimately either internalized into SLM parameters or retained externally for on-demand access. Yet existing approaches largely treat this placement as a consequence of the collaboration paradigm, leaving a fundamental question under-explored: what should the SLM learn, and what should it retrieve? We find that LLM-provided information can be broadly divided into domain-specific knowledge and generalizable reasoning capabilities, which favor different placement strategies. Domain-specific knowledge primarily serves particular domains or tasks. Parameterizing such knowledge in a capacity-limited SLM may introduce knowledge interference without necessarily improving its generalizable capabilities, making external storage more suitable. In contrast, generalizable reasoning capabilities, such as the ability to select, organize, and apply knowledge, can transfer across problems and tasks and are therefore better internalized into SLM parameters than repeatedly retrieved during inference. However, such differentiated placement is challenging because domain-specific knowledge and generalizable reasoning are often tightly coupled within the same reasoning trajectory, and their roles depend on the dependencies among reasoning steps. The key challenge is therefore to disentangle them based on these dependencies while preserving a complete reasoning path. We propose KSplit, a type-aware knowledge-placement framework for LLM–SLM collaboration. KSplit represents LLM reasoning as a typed dependency graph connecting knowledge units and reasoning operations through their functional dependencies. Dependency-Aware Knowledge Partitioning (DAKP) exploits these dependencies to disentangle domain-specific knowledge from generalizable reasoning while preserving complete reasoning paths. Functional Knowledge Placement (FKP) then internalizes generalizable reasoning through knowledge-free reasoning supervision, while compiling domain-specific knowledge into an external retrievable repository. Experiments across multiple benchmarks show that KSplit consistently improves LLM–SLM knowledge transfer, with controlled studies further validating its type-aware partitioning and placement strategy.

1 INTRODUCTION

Large language models offer strong knowledge and reasoning capabilities, but their substantial computational and deployment costs often limit their direct use on resource constrained devices. Large and small language model collaboration provides a practical alternative by allowing SLMs to leverage the stronger capabilities of LLMs while reducing dependence on repeated LLM assistance. Existing approaches realize such collaboration through answer supervision, rationale distillation, or retrieval augmented inference (Hsieh et al., 2023; Shridhar et al., 2023; Lewis et al., 2020). Despite their different formulations, the information provided by the LLM ultimately follows two destinations. It is either internalized into the parameters of the SLM or retained externally for on demand access. Parameter updates make acquired knowledge and reasoning capabilities directly available within the SLM, whereas external memory preserves information outside the model and retrieves it when needed.

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

How LLM provided information is placed therefore determines what the SLM learns internally and what it must access externally during inference.

We find that LLM provided information can be broadly divided into domain specific knowledge and generalizable reasoning capabilities, which favor different placement strategies. Domain specific knowledge mainly serves particular domains or tasks. Parameterizing such knowledge in a capacity limited SLM may consume limited model capacity and interfere with existing knowledge without necessarily improving generalizable capabilities. External storage is therefore more suitable for preserving such information and accessing it when needed. In contrast, generalizable reasoning capabilities describe how knowledge is selected, organized, and applied across problems. Since these capabilities can transfer across tasks, they are better internalized into SLM parameters than repeatedly retrieved during inference. This distinction suggests that effective LLM and SLM collaboration should first separate different types of information and then determine where each type should reside.

However, achieving such differentiated placement is nontrivial because domain specific knowledge and generalizable reasoning capabilities are often tightly coupled within the same reasoning trajectory. Individual reasoning steps may simultaneously involve task dependent factual knowledge and reusable reasoning operations, making it difficult to assign each step to a single category in isolation. Moreover, the function of a reasoning step often depends on its relationships with preceding and subsequent steps, so naive decomposition may break essential dependencies and disrupt the logical continuity of the original trajectory. The key challenge is therefore to disentangle these two types of information according to their dependency structure while preserving a coherent and complete reasoning path for subsequent learning and retrieval.

To address this challenge, we propose KSplit, a type-aware knowledge placement framework for LLM and SLM collaboration. KSplit first represents LLM reasoning as a typed dependency graph that explicitly captures knowledge units, reasoning operations, and their functional dependencies. Based on this structured representation, Dependency Aware Knowledge Partitioning, DAKP, exploits these dependencies to disentangle domain specific knowledge from generalizable reasoning while preserving complete reasoning paths. Functional Knowledge Placement, FKP, then places the separated information according to its functional characteristics. It internalizes generalizable reasoning into the SLM through knowledge free reasoning supervision while compiling domain specific knowledge into an external retrievable repository for on demand access. Experiments across multiple benchmarks show that KSplit consistently improves LLM and SLM knowledge transfer, while controlled studies further validate the effectiveness of its type-aware partitioning and placement strategy.

Our contributions are summarized as follows.

- We identify a key knowledge placement principle for LLM and SLM collaboration. Domain specific knowledge and generalizable reasoning capabilities exhibit different transfer characteristics and should therefore be assigned to different destinations rather than uniformly parameterized or externalized.
- We propose **KSplit**, a type aware knowledge placement framework that first disentangles domain specific knowledge from generalizable reasoning according to their functional dependencies and then places them separately. KSplit introduces Dependency Aware Knowledge Partitioning to preserve complete reasoning paths during separation and Functional Knowledge Placement to internalize reusable reasoning while retaining domain knowledge externally.
- We conduct extensive experiments across multiple benchmarks and controlled settings. The results show that KSplit consistently improves LLM and SLM knowledge transfer and further validate the effectiveness of dependency aware partitioning and type aware placement.

2 Related Work

External knowledge and retrieval control. Retrieval-augmented language models combine parametric generation with an external corpus, keeping factual evidence accessible and updatable without writing every change into model weights (Lewis et al., 2020; Borgeaud et al., 2022). Subsequent systems interleave retrieval with multi-step reasoning (Trivedi et al., 2023) or trigger retrieval from uncertainty and intermediate model states (Jiang et al., 2023; Su et al., 2024). Other work adapts retrieval to question complexity (Jeong et al., 2024). This line of work optimizes access to an already defined external source: how evidence is selected, acquired, and integrated for a query. KSplit

addresses the preceding allocation decision inside teacher supervision: which units should become reusable parametric capability, and which should be compiled into an editable external store. Retrieval control remains complementary once that allocation has produced the external K resource.

Reasoning distillation and parametric adaptation. Chain-of-thought prompting elicits intermediate computation from a frozen model (Wei et al., 2022; Kojima et al., 2022; Wang et al., 2022). Instruction- and explanation-based distillation transfer teacher-generated guidance and rationales to smaller models (Mukherjee et al., 2023; Li et al., 2023), while related methods distill structured reasoning traces or stepwise supervision (Shridhar et al., 2023; Hsieh et al., 2023). Parameter-efficient methods such as LoRA make this transfer practical by adapting a small parameter subset (Hu et al., 2021). These methods primarily optimize the supervision signal and update mechanism. When a complete rationale is treated as one target, factual content and reusable operations share the same parametric destination. Work separating memory recall from reasoning behavior shows that the two capabilities can be analyzed independently (Jin et al., 2025). KSplit turns this behavioral distinction into a storage rule by parameterizing graph-derived R and retaining explicit, update-sensitive K in an external module.

Structured reasoning and knowledge separation. Plans, search trees, thought graphs, and executable traces expose intermediate computation and its dependencies for inference, exploration, or faithfulness (Yao et al., 2023; Besta et al., 2024; Lyu et al., 2023). KSplit uses structure at a different stage: DAKP constructs and validates a typed dependency graph during asset creation, then compiles its K and R units for different storage interfaces. Its output is a pair of storage-ready assets rather than an inference-time graph-search policy. Viewed together, the neighboring literatures optimize complementary decisions: retrieval control governs access, distillation governs transfer, structured reasoning organizes computation, and KSplit governs the allocation of typed solution content to parametric or external storage.

3 METHOD

KSplit allocates teacher-derived content according to its functional role. We operationalize domain-specific knowledge as factual knowledge units (K), including facts, definitions, and task-specific rules retained in inspectable and editable external storage. We operationalize generalizable reasoning capabilities as reasoning operations (R) for organizing and applying evidence, which are internalized in model parameters. The central challenge is separation: teacher solutions interleave K and R, and a unit's role often depends on the surrounding reasoning dependencies rather than its surface form alone.

Figure 1 gives the complete pipeline. During training, Dependency-Aware Knowledge Partitioning (DAKP) converts teacher-assisted solutions into validated typed graphs. Functional Knowledge Placement (FKP) then compiles the accepted outputs as provenance-bearing K records and K-text-free R supervision, mapping them to an external K module and a parametric R adapter. The resulting system defines two deployment paths: R uses the parametric adapter, whereas RK augments it with external K. Fixed RK provides the core type-aware configuration; Capability-Aware Knowledge Routing (CAKR) complements it with instance-wise selection between R and RK.

3.1 PROBLEM SETUP AND FUNCTIONAL KNOWLEDGE TYPES

Let $\mathcal{D}_{\text{tr}} = \{(x_i, y_i^*)\}_{i=1}^N$ be the supervised training set. Teacher assistance constructs train-side assets; deployment uses the student and the external K module. We use K and R as functional labels relative to the current task, student, and evidence pipeline.

For an instance x_i, **K** comprises explicit propositions that a solver may need to access, including facts, definitions, qualitative laws, relations, criteria, and task- or domain-specific rules. External storage keeps this content attributable, inspectable, and replaceable. **R** comprises answer-independent operations that organize and apply available evidence, such as decomposing a problem, identifying conditions, applying a rule, comparing alternatives, integrating evidence, and checking a conclusion. R supervision excludes copied K-node text, answer labels, option text, and instance-specific conclusions that expose the target.

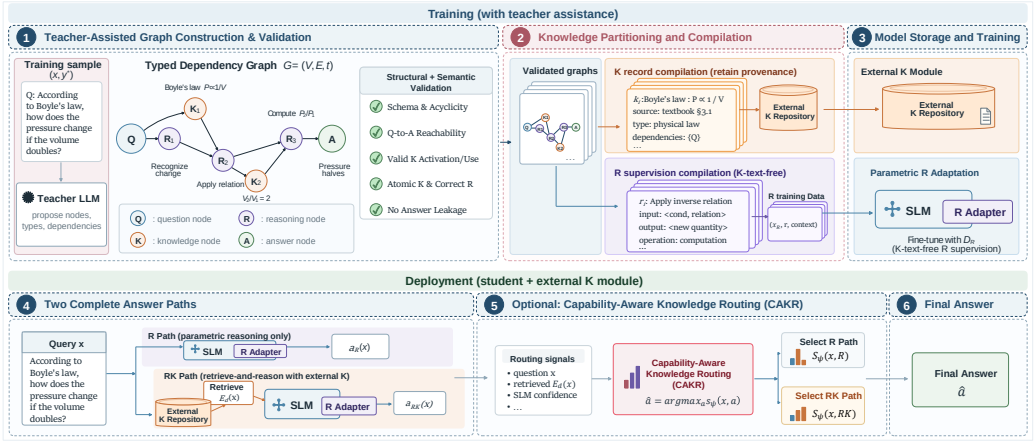


Figure 1: Overview of KSplit. DAKP constructs validated typed graphs; FKP stores reusable R parametrically and explicit K externally. The resulting R and RK paths support fixed RK and CAKR-based path selection.

After assigning functional roles from their dependencies, KSplit applies

$$\text{Allocate}(v) = \begin{cases} \text{external K module}, & t(v) = \text{K}, \\ \text{parametric R adaptation}, & t(v) = \text{R}. \end{cases} \tag{1}$$

The allocation concerns explicit content. Learned retrieval, ranking, and K-use parameters provide access to the external repository, which remains the storage location of the compiled K records.

3.2 Dependency-Aware Knowledge Partitioning

A fluent teacher solution often mixes facts, operations, and conclusions. DAKP resolves their functional roles from the dependencies that activate and consume each unit, representing the result as a typed directed graph

$$G_i = (V_i, E_i, t_i), \qquad t_i : V_i \to \{\text{Q}, \text{K}, \text{R}, \text{A}\}. \tag{2}$$

Q and A mark the problem and answer boundaries. A node is typed as K when it carries answer-relevant content whose validity may change with the underlying facts, rules, or domain assumptions; it is typed as R when it specifies a reusable evidence-processing operation without carrying the answer-bearing content itself. Typing therefore follows a node's function in the graph, with directed edges recording question activation, reasoning flow, knowledge use, and answer support. This representation preserves how facts and operations contribute to the solution while keeping their storage roles explicit.

Teacher-assisted construction jointly proposes node contents, types, and dependencies. Each candidate then passes structural and semantic quality control before entering the accepted graph pool. Structural checks enforce the schema, acyclicity, Q-to-A reachability, an ordered R backbone, and valid K activation and use. Semantic checks assess whether K is atomic and supportable, whether R expresses its declared operation, whether edges agree with node roles, and whether either type leaks the answer. Generation, repair, and review lineages vary across datasets; compilation therefore begins from the final accepted pool together with its retained provenance.

Validated partition output. Figure 3 traces one accepted solution through typed graph construction and export. The dependencies distinguish answer-bearing K content from reusable R operations even when surface form alone is ambiguous. FKP's deterministic export compiles provenance-bearing K records $\mathcal{K}^G$ and ordered, K-text-free R-supervision examples $\mathcal{D}_R$. Compilation applies node/path eligibility checks, validation, and deduplication, so the two collections need not have a one-to-one correspondence. The appendix provides the full prompts, schemas, validation rules, coverage, and node-level example. The resulting intervention evaluates graph-conditioned separation as a complete treatment; isolating the causal contribution of individual edges would require a separate intervention.

3.3 FUNCTIONAL KNOWLEDGE PLACEMENT

The compiled assets have different operational requirements. FKP assigns each one a storage and execution interface suited to its role: R becomes parametric supervision, while K becomes an explicit external resource.

Parameterizing R. Let $\mathcal{D}_R = \{(x_i, y_i^*, \rho_i)\}$ be the eligible R-supervision set, where ρ_i is the compiled reasoning interface. Starting from student parameters θ_0, we learn a parameter-efficient adapter $\Delta\theta_R$ using answer supervision and interface generation:

$$\mathcal{L}_R = \lambda_{\text{ans}}\mathcal{L}_{\text{ans}}(y_i^* \mid x_i) + \lambda_{\text{int}}\mathcal{L}_{\text{int}}(\rho_i \mid x_i). \tag{3}$$

Both terms are autoregressive token losses over answers and interfaces. R-training inputs omit the graph's externalized K text: the interface loss transfers useful operations and their order, while the answer loss couples them to task success. This objective controls the placement of the newly compiled assets while retaining the pretrained student's existing parameter state.

Externalizing and using K. For task setting d, K records are deduplicated and indexed as a repository $\mathcal{B}_d$, optionally together with official task resources when the benchmark protocol permits them. Repository composition, query construction, reranking, and aggregation are dataset-specific. Records retain source identifiers where available. Source-linked exclusions are applied when reliable mappings exist, and the accompanying manifests record incomplete historical provenance.

Retrieval returns evidence $E_d(x) = \text{Ret}_d(x; \mathcal{B}_d)$. The external module may also include K-use parameters trained on deployment-matched, K-conditioned supervision so that the student can apply the retrieved content. The explicit K records remain in $\mathcal{B}_d$, where they can be inspected or updated independently of the R adapter. Material corpus changes trigger rebuilding or revalidation of dependent indexes, K-use components, cached candidates, and routing assets.

The stored assets instantiate two complete answer actions:

$$a_R^d(x) = \text{Dec}_R^d\big(x; \Theta_R^d\big), \tag{4}$$

$$a_{RK}^d(x) = \text{Agg}_d\Big(\{\text{Dec}_{RK}^d(x, e; \Theta_{RK}^d) : e \in E_d(x)\}\Big). \tag{5}$$

Θ_R^d and Θ_{RK}^d denote the complete parameter states used by each path, while Agg_d covers joint-context or candidate-level aggregation. Depending on the dataset, RK may differ from R in both visible evidence and K-use parameters; the appendix records the complete state of each path. Applying a_{RK}^d to every instance yields fixed RK, a complete deployment policy for KSplit.

3.4 CAPABILITY-AWARE KNOWLEDGE ROUTING

R and RK can have different instance-level utility: external evidence may supply missing knowledge, but irrelevant or conflicting retrieval can perturb an otherwise sufficient R prediction. CAKR treats this heterogeneity as an answer-path composition problem over $\mathcal{A}_d(x) = \{a_R^d(x), a_{RK}^d(x)\}$.

Held-out or out-of-fold outcomes provide supervision about the current student–pipeline pair. Dataset-specific estimators score the two actions using inference-time information such as the question, retrieval statistics, candidate predictions, reader confidence, or consistency signals. We define capability awareness as the expected utility of a concrete action for the current system. With calibrated scores s_{ψ_d}, the generic policy is

$$\widehat{a}_d(x) = \arg\max_{a \in \mathcal{A}_d(x)} s_{\psi_d}(x, a), \qquad \widehat{y}(x) = \widehat{a}_d(x). \tag{6}$$

Learners, features, targets, defaults, and candidate schedules vary by dataset. In the evaluated pipelines, CAKR operates after retrieval and may also follow candidate execution. The present evaluation therefore measures answer-path selection quality, while compute-aware routing remains outside its scope. Fixed RK remains the complete non-adaptive deployment policy.

4 EXPERIMENTS

We evaluate KSplit through a sequence of linked claims. We first test whether type-aware storage yields a competitive complete system. We then examine the two storage pressures that motivate

treating K and R differently, followed by a matched intervention that compares the intended placement with its reversal. Next, we isolate whether DAKP preserves downstream utility while limiting acquisition of K assigned to the external store. Finally, we use robustness and boundary analyses to delimit where these conclusions hold. This organization separates system-level comparisons from matched mechanism evidence: each supports a different part of the argument rather than serving as a duplicate measure of the same claim.

4.1 Experimental Setup

Evaluation structure. The evaluation has four levels. Table 1 establishes end-to-end viability under each method's native interface. Figure 2 combines premise checks with a matched four-arm placement design. Table 2 evaluates the complete DAKP treatment under a Same-K control. Robustness, sensitivity, and boundary analyses then test the scope of the preceding conclusions, with exact secondary results retained in Appendix E.

Tasks and protocols. We report exact-answer accuracy on OpenBookQA, MedQA, StrategyQA, ARC-Challenge, and QuaRel (Mihaylov et al., 2018; Jin et al., 2021; Geva et al., 2021; Clark et al., 2018; Tafjord et al., 2019) under dataset-specific evaluation protocols. StrategyQA is evaluated on its labeled evaluation split. For OpenBookQA, the repository combines official facts with K extracted from training graphs; the remaining repositories use train-side graph-derived K. Appendix C specifies the splits, repository construction, and run-level protocols.

Comparison contract and implementation. Table 1 groups methods by adaptation and knowledge access: prompting uses neither task-specific training nor a repository, parametric baselines learn from answers or rationales, and retrieval or collaborative baselines retain their native external-information interfaces. This comparison tests system competitiveness; Section 4.3 tests the storage mechanism with matched branches, information access, questions, and estimands. Unless stated otherwise, trainable pipelines use Qwen2.5-3B-Instruct and the same task examples; R adaptation uses LoRA, and teacher APIs construct train-side assets. The fixed paths parameterize R, couple retrieved K with a knowledge-conditioned adapter, or combine both stores. CAKR selects after the R and RK answers have been materialized.

Reporting and uncertainty. Invalid, unparseable, and missing predictions count as incorrect. We report mean $\pm$ sample standard deviation for three-seed entries and the observed accuracy for single-run entries. Paired contrasts align predictions by question. Single-run contrasts use an item bootstrap; the controlled placement study uses 20,000 crossed seed–question bootstrap replicates with percentile 95% confidence intervals. Boldface in Table 1 records the highest point estimate, while inferential claims are tied to the reported intervals.

4.2 End-to-End Performance of KSplit

A competitive complete system. Table 1 first asks whether the proposed allocation yields a viable complete system before answer-path composition. Fixed RK has a higher point estimate than the strongest evaluated non-KSplit entry on OpenBookQA (91.40 vs. 89.73), MedQA (58.99 vs. 57.29), and QuaRel (92.03 vs. 90.76); it is within 0.22 points on ARC-Challenge (85.13 vs. 85.35) and 1.17 points on StrategyQA (71.32 vs. 72.49). The same type-aware allocation rule is therefore competitive across all five benchmarks.

Complementary fixed paths. Within KSplit, the fixed-RK point estimate exceeds fixed K on all five tasks and fixed R on four. StrategyQA is the informative exception: fixed R reaches 72.49 compared with 71.32 for fixed RK. Joint access to parametric R and external K is thus the strongest fixed policy in most settings, but adding external K is not uniformly beneficial at the point-estimate level. This variation motivates composing complete answer paths instead of assuming that one action must dominate every instance.

Residual answer-path composition. CAKR changes the fixed-RK point estimate by $+0.06$ to $+1.17$ points across the five datasets; on OpenBookQA, the $+0.27$-point mean change has a 95% CI of $[-0.27, +1.00]$. These modest, dataset-dependent gains quantify residual complementarity

Table 1: End-to-end accuracy on five knowledge-intensive reasoning benchmarks. Uncertainty denotes mean ± sample standard deviation over three fixed seeds; other entries are single runs under the corresponding dataset-specific protocol. Boldface marks the highest point estimate per benchmark, with ties retained.

Category	Method	OpenBookQA	MedQA	StrategyQA	ARC-Challenge	QuaRel
Prompting	Direct	79.20	48.86	62.01	82.59	65.04
	Zero-shot CoT	80.40	49.18	62.01	82.17	53.44
Parametric baselines	Answer-SFT	88.80 ± 1.00	56.59 ± 1.01	71.18 ± 0.76	84.44 ± 0.65	90.76 ± 1.10
	CoT Distillation / SCoTD	83.80 ± 1.06	54.18 ± 0.63	69.29 ± 0.91	83.28 ± 0.31	88.04 ± 1.19
	Distilling Step-by-Step	89.73 ± 0.31	57.29 ± 0.16	**72.49 ± 0.87**	85.35 ± 0.18	90.40 ± 0.83
Retrieval baselines	Standard RAG	76.80	55.46	53.28	80.72	54.17
	Confidence-gated RAG	80.40	55.15	56.33	82.34	72.28
Collaborative baseline	PRR	83.67 ± 0.64	54.80 ± 0.51	64.92 ± 1.33	82.48 ± 0.18	87.14 ± 0.94
Ours: fixed policies	Fixed $\mathcal{R}$	88.53 ± 0.42	57.34 ± 1.58	**72.49 ± 2.66**	84.53 ± 0.13	91.06 ± 1.21
	Fixed $\mathcal{K}$	88.93 ± 0.31	58.89 ± 1.58	69.58 ± 1.76	83.13 ± 0.36	90.16 ± 0.28
	Fixed $\mathcal{RK}$	91.40 ± 0.80	58.99 ± 1.73	71.32 ± 2.77	85.13 ± 0.36	92.03 ± 0.63
Ours: composition	CAKR	**91.67 ± 0.64**	**59.10 ± 1.77**	**72.49 ± 2.27**	**85.72 ± 0.13**	**92.09 ± 0.73**

between materialized R and RK answers. Fixed RK therefore carries the complete-system claim, while CAKR measures remaining selection headroom; Appendix E reports its route-level behavior. Because Table 1 retains each method's native interface, it establishes system viability rather than the effect of placement itself. We test that effect next.

4.3 Evidence for Type-Aware Knowledge Placement

Two distinct storage pressures. Panels (a)–(b) of Figure 2 examine the two premises behind differentiated storage. First, parameterizing factual K increases acquisition of unseen target-fact paraphrases by 3.21 points (95% CI $[2.66, 3.77]$) in the original study and by 3.07 points ($[2.60, 3.56]$) in an extension. The corresponding retention estimates are negative, -0.96 ($[-2.22, 0.34]$) and -0.67 ($[-1.57, 0.17]$), but both intervals span zero; acquisition is replicated, whereas the retained-behavior effect remains unresolved. Second, when reusable R patterns are served externally, a model-assisted audit records $7.97\times$ as many copy events and $7.82\times$ as many R-context tokens as one-time delivery of each accepted pattern. The two studies therefore expose different operational pressures—in-weight acquisition of K and repeated external delivery of R. They motivate separate treatment, but the better orientation must be tested by downstream performance.

Matched orientation test. We next ask whether the intended R-parametric/K-external orientation outperforms the reversed placement. A controlled 2×2 design varies both the parameter branch and the externally supplied information. $\mathcal{R}_0$ and $\mathcal{K}_0$ evaluate the two parameter branches without external content; $\mathcal{RK}$ instantiates the intended configuration, and $\mathcal{KR}$ reverses the stores. The direct $\mathcal{RK}-\mathcal{KR}$ contrast compares the two complete configurations. To account for pre-existing differences between the parameter branches, we also report the branch-adjusted interaction

$$\mathrm{DID} = (\mathcal{RK} - \mathcal{R}_0) - (\mathcal{KR} - \mathcal{K}_0).$$

The direct contrast answers which complete configuration performs better; DID is the stricter estimand for whether the gain from external information depends on the parameterized branch. Panels (c)–(d) of Figure 2 report both.

Complete configurations favor RK. All five direct point estimates favor the intended complete configuration: $\mathcal{RK} - \mathcal{KR}$ ranges from $+0.58$ to $+2.00$ points. OpenBookQA has the largest estimate ($+2.00$, 95% CI $[0.00, +3.92]$); the other four intervals also span zero. The cross-task regularity is therefore directional rather than evidence of a universally nonzero effect. Unlike Table 1, this comparison holds the controlled placement design fixed and directly contrasts two complete allocations.

Adjusted mechanism evidence is task-dependent. The stricter branch-adjusted result is heterogeneous. OpenBookQA has a positive DID whose interval lies above zero ($+2.42$, 95% CI

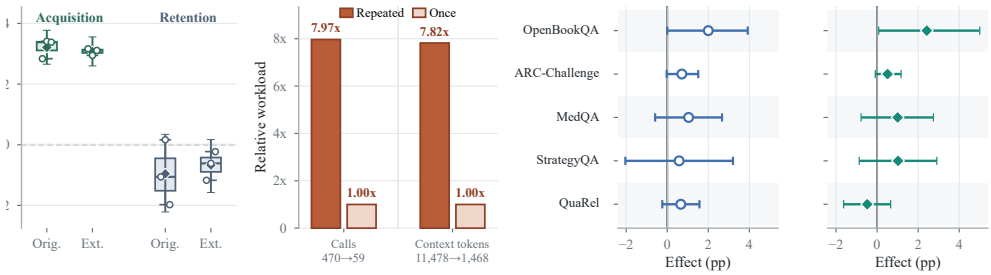


Figure 2: Type-aware storage evidence: **(a)** K acquisition and retention, **(b)** reusable-R delivery, **(c)** complete RK−KR contrast, and **(d)** branch-adjusted DID, with 95% confidence intervals.

$[+0.08, +5.00]$). ARC-Challenge ($+0.51$), MedQA ($+1.00$), and StrategyQA ($+1.02$) remain positive, but their intervals span zero. QuaRel separates the two estimands: its direct contrast is $+0.66$, whereas its DID is -0.48 (95% CI $[-1.63, +0.66]$). Thus the complete RK allocation is directionally favored on every task, while evidence for the incremental placement interaction is supported on OpenBookQA, directional on three tasks, and reversed but uncertain on QuaRel.

Evidence synthesis. The four panels provide converging but non-interchangeable evidence. The premise checks establish why K and R merit separate treatment; the direct contrast favors the intended deployable configuration; and DID shows that the branch-adjusted mechanism strength depends on the task. One explanation consistent with this pattern is complementarity: differentiated storage should matter most when a task needs both reusable operations and external factual knowledge, and less when either source is already sufficient. The present study does not manipulate that task-level mixture, so we treat complementarity as an interpretation to be tested rather than as an identified result. Appendix D reports the per-seed arms, log-odds analyses, and full placement matrix.

4.4 Separation Quality of DAKP

A two-criterion test. Useful separation must satisfy two requirements simultaneously: R supervision should retain downstream utility, and it should limit parametric acquisition of K assigned to the external store. We compare Graph-R with a Same-K Bag-R control on OpenBookQA. For each seed, both arms start from the same answer-only checkpoint and share the same 992 questions, K inventory, R-only update budget, and evaluator. Graph-R uses R targets compiled from accepted typed dependency graphs. Bag-R targets are generated from the identical K facts under a graph-free construction, matching source content while removing access to node identifiers, edges, paths, and Graph-R targets.

Utility is retained while K acquisition decreases. Table 2 shows that Graph-R retains utility and reduces K acquisition. Its dev500 advantage is $+1.13$ points, with a lower confidence bound of -0.80 above the pre-specified -1-point non-inferiority margin. On 200 K items that the starting model treats as unknown or uncertain, Graph-R assigns lower gold-answer log-odds than Bag-R (difference -0.514, 95% CI $[-0.784, -0.318]$). The Same-K treatment therefore supports graph-derived R supervision as non-inferior for downstream utility while reducing acquisition of knowledge intended for external storage.

Operational corroboration and scope. Appending local dependency text lowers node-typing Macro-F1 from 94.24 to 84.78 (-9.46, 95% CI $[-12.73, -6.32]$), supporting the use of dependencies to construct and validate the graph rather than as typing input. The Same-K estimand covers the complete graph-conditioned construction: organization, target length, and expression format remain bundled, and Bag-R targets are 57.8% longer despite per-example loss normalization. Appendix B gives the functional K/R rule, validation contract, coverage, and a worked graph; Appendix D gives the complete protocols and diagnostics.

Table 2: DAKP Same-K evaluation on OpenBookQA over three seeds. Graph-R minus Bag-R effects include 95% hierarchical paired-bootstrap intervals.

Criterion	Measure	Graph-R	Bag-R	Δ [95% CI]	Outcome
Utility retention	Dev500 acc. (%) $\uparrow$	86.47	85.33	$+1.13$ $[-0.80, +3.07]$	Non-inferior
K separation	K-probe log-odds $\downarrow$	3.391	3.905	-0.514 $[-0.784, -0.318]$	Reduced

Table 3: Teacher-workflow robustness on OpenBookQA Test500 (mean accuracy (%) $\pm$ sample SD over three seeds). Bold marks the row best; underlining marks the best fixed policy, with ties retained.

Teacher	Fixed $\mathcal{R}$	Fixed $\mathcal{K}$	Fixed $\mathcal{RK}$	CAKR
GPT	88.53 ± 0.42	88.93 ± 0.31	91.40 ± 0.80	$\mathbf{91.67 \pm 0.64}$
DeepSeek	88.67 ± 0.31	89.20 ± 0.35	89.07 ± 0.23	$\mathbf{89.40 \pm 0.20}$
Qwen	87.47 ± 0.46	85.93 ± 1.29	87.47 ± 1.15	$\mathbf{89.13 \pm 0.81}$

4.5 Robustness and Boundaries

Workflow robustness. Table 3 shows that the best fixed policy changes across GPT-, DeepSeek-, and Qwen-family workflows, while the CAKR point estimate remains higher. This coupled test supports workflow-level robustness.

Objective sensitivity. Across the frozen answer/interface loss sweep, fixed R remains within a 1.07-point range, whereas K and RK become unstable when interface supervision is removed. A separate Safe-RK follow-up exceeds both single-store paths at the two evaluated ratios. The results identify a workable neighborhood of objectives while showing that access to both stores alone does not remove sensitivity to how the interface is trained.

Boundary behavior and residual headroom. QuaRel illustrates why system, mechanism, and composition evidence must remain distinct: fixed RK exceeds both single-store paths in every seed, the direct RK$-$KR contrast is positive, and the branch-adjusted DID is negative but uncertain. CAKR changes only a small fraction of decisions in the evaluated pipelines; on OpenBookQA, five repairs and one harm coexist with an oracle accuracy of 93.00%. The current policy captures only part of this headroom, leaving improved answer-path selection as a separate opportunity.

Scope of the evidence. Together, the stress tests show that type-aware configurations remain viable across the evaluated workflows, while the preferred fixed policy varies with the workflow and task. Appendix E reports the objective sweeps and routing diagnostics; Appendix D provides the per-seed arms and complementary estimands.

5 Conclusion

We study how teacher-derived knowledge should be allocated between model parameters and external storage. KSplit uses DAKP to separate dependency-coupled domain-specific knowledge (K) from generalizable reasoning operations (R), and FKP to internalize R while retaining K in an external repository. Across five benchmarks, fixed RK has a higher point estimate than the strongest evaluated non-KSplit entry on three tasks and is within 1.17 points on the other two. In matched placement tests, the direct RK$-$KR estimates are positive on all five tasks ($+0.58$ to $+2.00$ points), while the branch-adjusted evidence is task-dependent. In the Same-K evaluation, Graph-R is non-inferior in utility ($+1.13$ points; 95% CI $[-0.80, +3.07]$) and reduces acquisition of externalized K (-0.514 log-odds; 95% CI $[-0.784, -0.318]$).

Together, these results make storage allocation a first-class empirical design choice. DAKP turns interleaved teacher solutions into validated typed graphs, while FKP compiles them into auditable, storage-specific assets. The matched evaluations show where differentiated placement helps. The mixed DID pattern, including the QuaRel boundary case, further indicates that the value of each store depends on the task and the coupled execution pipeline. KSplit provides a concrete framework for implementing this allocation and testing it with complete systems and controlled interventions. We discuss the scope of the current evidence and remaining validation questions in Appendix A.

6 AI USE STATEMENT

Generative AI tools assisted manuscript structuring, drafting and editing, experiment/result auditing, code and figure preparation, and diagnostic analysis. Reported numerical results were taken from experiment artifacts rather than generated by these tools. The authors are responsible for the manuscript, the selected claims, and verifying the scope and accuracy of this disclosure before submission.

7 REPRODUCIBILITY STATEMENT

Section 4.1 specifies the evaluation and uncertainty protocol. Appendix B records the operational K/R distinction underlying DAKP; Appendix C gives the dataset and pipeline configurations; Appendices D and E provide complete evidence for the central claims together with robustness, sensitivity, and boundary analyses; and Appendix F extends the related-work discussion. The submitted artifact retains prediction-level provenance and the result summaries used to produce the reported tables. Run-specific provenance and metadata availability are stated explicitly.

REFERENCES

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, et al. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pp. 17682–17690, 2024.

Baolong Bi, Shenghua Liu, Yiwei Wang, Yilong Xu, Junfeng Fang, Lingrui Mei, and Xueqi Cheng. Parameters vs. context: Fine-grained control of knowledge reliance in language models. In *International Conference on Learning Representations*, volume 2026, pp. 106786–106808, 2026.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, et al. Improving language models by retrieving from trillions of tokens. In *International conference on machine learning*, pp. 2206–2240. PMLR, 2022.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try ARC, the AI2 reasoning challenge. In *arXiv preprint arXiv:1803.05457*, 2018.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. StrategyQA: A question answering benchmark for implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9:346–362, 2021.

Xinyan Guan, Jiali Zeng, Fandong Meng, Chunlei Xin, Yaojie Lu, Hongyu Lin, Xianpei Han, Le Sun, and Jie Zhou. Deeprag: Thinking to retrieve step by step for large language models. In *International Conference on Learning Representations*, volume 2026, pp. 117056–117071, 2026.

Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alex Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. In *Findings of the association for computational linguistics: ACL 2023*, pp. 8003–8017, 2023.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong C Park. Adaptive-rag: Learning to adapt retrieval-augmented large language models through question complexity. In *Proceedings of the 2024 conference of the north american chapter of the association for computational linguistics: Human language technologies (volume 1: Long papers)*, pp. 7036–7050, 2024.

Zhengbao Jiang, Frank F Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. Active retrieval augmented generation. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 7969–7992, 2023.

Di Jin, Eileen Pan, Nassim Oufattole, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. What disease does this patient have? a large-scale open domain question answering dataset from medical exams. *Applied Sciences*, 11(14):6421, 2021.

Mingyu Jin, Weidi Luo, Sitao Cheng, Xinyi Wang, Wenyue Hua, Ruixiang Tang, William Yang Wang, and Yongfeng Zhang. Disentangling memory and reasoning ability in large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1681–1701, 2025.

Hailey Joren, Jianyi Zhang, Chun-Sung Ferng, Da-Cheng Juan, Ankur Taly, and Cyrus Rashtchian. Sufficient context: A new lens on retrieval augmented generation systems. In *International Conference on Learning Representations*, volume 2025, pp. 20310–20334, 2025.

Minki Kang, Seanie Lee, Jinheon Baek, Kenji Kawaguchi, and Sung Ju Hwang. Knowledge-augmented reasoning distillation for small language models in knowledge-intensive tasks. *Advances in Neural Information Processing Systems*, 36:48573–48602, 2023.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35: 22199–22213, 2022.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33: 9459–9474, 2020.

Liunian Harold Li, Jack Hessel, Youngjae Yu, Xiang Ren, Kai-Wei Chang, and Yejin Choi. Symbolic chain-of-thought distillation: Small models can also "think" step-by-step. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2665–2679, 2023.

Xiang Li, Shizhu He, Jiayu Wu, Zhao Yang, Yao Xu, Yang jun Jun, Haifeng Liu, Kang Liu, and Jun Zhao. Mode-cotd: Chain-of-thought distillation for complex reasoning tasks with mixture of decoupled lora-experts. In *Proceedings of the 2024 joint international conference on computational linguistics, language resources and evaluation (LREC-COLING 2024)*, pp. 11475–11485, 2024.

Qing Lyu, Shreya Havaldar, Adam Stein, Li Zhang, Delip Rao, Eric Wong, Marianna Apidianaki, and Chris Callison-Burch. Faithful chain-of-thought reasoning. In *Proceedings of the 13th international joint conference on natural language processing and the 3rd conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (Volume 1: Long papers)*, pp. 305–329, 2023.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2381–2391, 2018.

Subhabrata Mukherjee, Arindam Mitra, Ganesh Jawahar, Sahaj Agarwal, Hamid Palangi, and Ahmed Awadallah. Orca: Progressive learning from complex explanation traces of gpt-4. *arXiv preprint arXiv:2306.02707*, 2023.

Zhengliang Shi, Shuo Zhang, Weiwei Sun, Shen Gao, Pengjie Ren, Zhumin Chen, and Zhaochun Ren. Generate-then-ground in retrieval-augmented generation for multi-hop question answering. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7339–7353, 2024.

Kumar Shridhar, Alessandro Stolfo, and Mrinmaya Sachan. Distilling reasoning capabilities into smaller language models. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 7059–7073, 2023.

Weihang Su, Yichen Tang, Qingyao Ai, Zhijing Wu, and Yiqun Liu. Dragin: Dynamic retrieval augmented generation based on the real-time information needs of large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12991–13013, 2024.

Oyvind Tafjord, Peter Clark, Matt Gardner, Wen-tau Yih, and Ashish Sabharwal. Quarel: A dataset and models for answering questions about qualitative relationships. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 7063–7071, 2019. doi: 10.1609/aaai.v33i01.33017063.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 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)*, pp. 10014–10037, 2023.

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 2609–2634, 2023.

Shangshang Wang, Julian Asilis, Ömer Faruk Akgül, Enes Bilgin, Ollie Liu, and Willie Neiswanger. Tina: Tiny reasoning models via lora. In *International Conference on Learning Representations*, volume 2026, pp. 90106–90140, 2026.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Zhengxuan Wu, Aryaman Arora, Zheng Wang, Atticus Geiger, Dan Jurafsky, Christopher D Manning, and Christopher Potts. Reft: Representation finetuning for language models. *Advances in Neural Information Processing Systems*, 37:63908–63962, 2024.

Ling Yang, Zhaochen Yu, Tianjun Zhang, Minkai Xu, Joseph E Gonzalez, Bin Cui, and Shuicheng Yan. Supercorrect: Advancing small llm reasoning with thought template distillation and self-correction. In *International Conference on Learning Representations*, volume 2025, pp. 3386–3418, 2025.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.

Zijun Yao, Weijian Qi, Liangming Pan, Shulin Cao, Linmei Hu, Liu Weichuan, Lei Hou, and Juanzi Li. Seakr: Self-aware knowledge retrieval for adaptive retrieval augmented generation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 27022–27043, 2025.

Yichun Zhao, Shuheng Zhou, and Huijia Zhu. Probe then retrieve and reason: Distilling probing and reasoning capabilities into smaller language models. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pp. 13026–13032, 2024.

A Limitations

The scope of the conclusions follows the evaluated systems and estimands. K and R are functional labels relative to the current student, task, and evidence pipeline. Graph construction, type validation, and the reusable-R serviceability audit use model assistance; human-validated assets across additional settings therefore remain an important direction for further evaluation.

Among the branch-adjusted placement estimates, only OpenBookQA has an interval above zero, and QuaRel combines a positive direct contrast with a negative, uncertain DID. These results support a task-dependent interpretation of type-aware placement rather than a uniform advantage across all settings.

The Same-K study evaluates the complete graph-conditioned treatment, with organization, target length, and expression format still bundled. Factorial controls could further isolate these effects. CAKR currently selects after retrieval and, for some pipelines, after candidate execution. Extending it toward pre-retrieval, compute-aware routing would enable direct evaluation of latency, token, and energy savings.

B Operationalizing Knowledge Separation

This appendix makes the separation procedure auditable. We first state the functional typing rule, then give the invariant prompt contract and output schema, enumerate validation and repair checks, report coverage at the asset boundaries that are actually preserved, and walk through one accepted graph. The typing is deliberately relative to the current task, student, and evidence pipeline; it is not proposed as a universal ontology of knowledge and reasoning.

B.1 Functional Typing Rule

The construction teacher proposes node contents, node types, and dependencies jointly. There is no separately trained K/R classifier. A node is typed as K when it supplies answer-relevant content that may need to change when the relevant facts, rules, or domain assumptions change. A node is typed as R when it specifies a reusable operation for transforming, comparing, integrating, or checking available evidence without carrying the answer-bearing content itself. Table 4 turns this distinction into an annotation rule.

Table 4: Operational rule used to distinguish stored knowledge (K) from reusable reasoning supervision (R). Entity mentions alone do not determine the label; the decisive question is the node's functional role in the graph.

Type	Positive criterion	Exclusion criterion
K	An atomic, supportable fact, definition, law, property, criterion, relation, mechanism, or task/domain rule needed by the instance.	Not an instruction such as compare, eliminate, combine, verify, or map evidence to a candidate.
R	A general or domain-role operation describing how evidence and activated knowledge are used.	Must not contain the concrete fact, named answer-specific entity, option identity, or selection cue that resolves the instance.

This is a storage decision rather than a claim that the two categories are metaphysically disjoint. For example, "sunlight supplies energy for plant growth" is stored as K, whereas "apply the retrieved facts to each candidate and compare the supported state" is used as R supervision. Borderline nodes are judged together with their declared operation and incident edges. If an apparent operation embeds a case-specific fact or answer cue, it is sanitized, retyped, repaired, or rejected.

B.2 Construction Prompt and Machine-Readable Contract

The prompt files were not byte-identical across datasets. Dataset adapters supplied domain-appropriate K vocabularies, role descriptions, and, in some pipelines, minimum graph sizes. The following contract is the invariant core shared by the construction pipelines:

> Construct one directed acyclic Q–K/R–A diagnostic graph for the instance. K records *what must be known*; R records *how the question, evidence, and activated knowledge are used*. Edges denote cognitive activation or usage order, rather than information-generation dependence. Preserve exactly one question node and one answer node, expose an R-only backbone with a unique source and sink, and return only valid JSON. Do not place the gold label, option identity, or answer-selecting language in K or R content.

The canonical output schema is shown below. Optional role fields are populated according to the dataset adapter, but the graph, reasoning backbone, and separately stored gold answer are common fields.

```
{
  "nodes": [{"id": "Q|K_i|R_i|A", "type": "Q|K|R|A",
             "content": "...", "knowledge_type": "...",
             "scope": "...", "reasoning_op": "...",
             "diagnostic_role": "..."}],
  "edges": [{"from": "...", "to": "...", "relation": "..."}],
  "reasoning_nodes": ["R_i", "..."],
  "reasoning_edges": [{"from": "R_i", "to": "R_j"}],
  "reasoning_source": "R_i", "reasoning_sink": "R_j",
  "gold_answer": "..."
}
```

The gold answer is required for supervision and audit bookkeeping but is stored outside node content. The permitted edge roles are $Q \to K$ for knowledge explicitly cued by the question, $Q \to R$ for entry into the reasoning backbone, $R \to R$ for operation order, $R \to K$ for knowledge activation, $K \to R$ for downstream use, and $R \to A$ for final support. The canonical contract disallows $K \to K$ and $K \to A$: knowledge must be consumed by an explicit operation before it can support the answer.

B.3 Validation, Localized Repair, and Rejection

Every exported asset is re-validated after parsing or repair. Table 5 lists the checks that matter for reproducibility.

Table 5: Validation and repair contract. "Repair" denotes a localized correction followed by the complete validation suite; an item that still fails is not exported.

Stage	Required condition	Failure action
Schema	Valid JSON; unique node identifiers; declared node and edge fields; references resolve.	Normalize or repair; reparse.
Global topology	Exactly one Q and one A; directed acyclic graph; Q is the unique source, A the unique sink; every K/R node lies on a Q–A path.	Local edge repair, otherwise reject.
R backbone	All R nodes form a connected DAG with a unique R source and sink; backbone edges agree with the full graph.	Synchronize and reconnect; recheck.
Knowledge use	Each K node has a valid activation context and a downstream R consumer at its earliest use; no direct $K \to A$ shortcut.	Reposition or repair edges.
Semantic typing	K is atomic and supportable; R is an operation; the two do not duplicate each other.	Sanitize, retype, or request semantic repair.
Leakage	No option identity, gold label, answer string, or answer-selecting instruction appears in K/R content.	Remove leaked content; otherwise reject.
Export	K retains provenance; R preserves operation order and roles while excluding K text and answer cues; gold is separate.	Block export until valid.

Repair histories differ because the assets were produced through several generation, migration, review, and re-validation lineages. The common logic is: deterministic normalization first (for example, synchronizing R-backbone edges with the full graph), localized repair from an explicit failure report second, and a full structural and semantic recheck last. We therefore treat the final accepted asset plus its provenance as the reproducibility unit, rather than implying that every dataset was produced by one identical teacher call.

B.4 Coverage at Auditable Asset Boundaries

Table 6 reports counts at artifact boundaries preserved in the frozen release. Because the source pipelines have different histories and several controlled experiments use compiled subsets, the rows document available assets rather than a common "first-call acceptance rate."

Table 6: Auditable coverage of the graph assets and the compiled R/K snapshots used by controlled comparisons. A dash means that the corresponding intermediate boundary is not used to infer a rate.

Dataset	Train	Accepted graph boundary	Compiled rows	Qualification
OpenBookQA	4,957	4,957	992	Full consolidated accepted pool; the 992-row snapshot is the frozen subset for the controlled container comparison.
MedQA	8,178	7,816	7,814	Final hard gate rejected 2 authority inputs before export.
StrategyQA	1,374	–	1,374	Full frozen training snapshot.
ARC-Challenge	992	–	984	Frozen snapshot contains 984 rows; we do not infer the cause of the eight absent rows from the count alone.
QuaRel	1,941	–	1,941	Full frozen training snapshot.

MedQA provides the most detailed retained ledger. The final authority stage received 7,816 graphs, re-applied the current hard content gate, rejected 2, and exported 7,814 graph/R examples. Those graphs contain 22,798 external K nodes, which become 22,754 records after knowledge-store deduplication. Thus the apparent $7{,}816$ versus $7{,}814$ mismatch reflects an explicit final gate rather than a silent change in the experimental population.

B.5 Worked Accepted Graph and Deterministic Export

Figure 3 gives a compact visual example of the complete DAKP transformation, from a teacher-assisted solution through a validated typed graph to the two storage-specific exports.

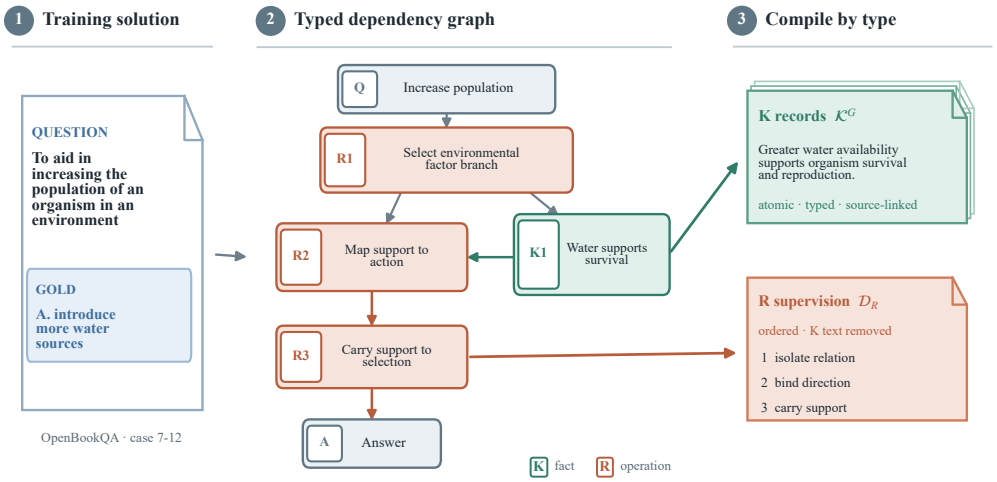


Figure 3: A provenance-locked worked example of Dependency-Aware Knowledge Partitioning (DAKP). A teacher-assisted training solution is represented as a validated Q–K–R–A dependency graph. FKP's deterministic export then produces graph-derived K records $\mathcal{K}^G$ and ordered, K-text-free R-supervision examples $\mathcal{D}_R$. Arrows denote stored graph dependencies or implemented export flow, not causal effects. Node and interface text is shortened for readability.

Table 7 reproduces an accepted OpenBookQA asset (ID `7-980`), with only line wrapping changed. The question asks what the sun is responsible for; the correct option is "plants sprouting, blooming and wilting." The example is useful because the knowledge contents and reasoning operations are visibly different without requiring a contrived boundary case.

Table 7: Worked Q–K/R–A graph from the accepted OpenBookQA pool.

ID	Type	Content
Q	Question	The sun is responsible for: (A) puppies learning tricks; (B) children growing and aging; (C) flowers wilting in a vase; (D) plants sprouting, blooming, and wilting.
R1	Reasoning	Decompose the unresolved prompt into a candidate-relation check and expose the first knowledge need.
K1	Knowledge	Sunlight provides energy needed for plant growth and development.
R2	Reasoning	Apply the retrieved facts to the candidate relation and form the supported state for comparison against the options.
K2	Knowledge	Plants go through life-cycle changes that include sprouting, blooming, and wilting.
R3	Reasoning	Integrate the supported state into a downstream selection-ready decision.
A	Answer	Make the final answer selection supported by the upstream reasoning.

The stored edges are $Q \to R1$, $R1 \to K1$, $K1 \to R2$, $R1 \to R2$, $R2 \to K2$, $K2 \to R3$, $R2 \to R3$, and $R3 \to A$. Deterministic compilation writes K1 and K2 as provenance-linked knowledge records. It separately serializes the ordered $R1 \to R2 \to R3$ operation target, excluding the two factual strings and the gold answer; the gold option remains in its dedicated supervision field. Notice that the graph need not contain equal numbers of K and R units, which is why the two exported containers can have different cardinalities.

Finally, the graph should be interpreted as an operational decomposition used to construct and audit the two containers, not as proof that its individual edges are uniquely causal. The main paper's controlled same-knowledge comparisons test the downstream utility of the resulting separation; the diagnostic analysis in Appendix D evaluates whether the stored K and R contents remain empirically distinguishable.

C Experimental Protocols

The experiments are organized around three inferential roles, and the protocol follows the same order. The main benchmark table asks whether the complete system is competitive under realistic, method-specific interfaces. The storage-orientation study uses matched arms to compare $\mathcal{RK}$ with the reversed $\mathcal{KR}$ allocation. The Same-K study asks whether graph-derived R supervision preserves task utility while reducing acquisition of the knowledge assigned to the external store. This hierarchy aligns each claim with the most specific available design: system comparison establishes viability, matched placement evaluates orientation, and Same-K evaluates the graph-conditioned separation treatment.

C.1 Evaluation Scope and Benchmark Splits

We evaluate exact-answer accuracy on five benchmarks chosen to vary the balance between reusable reasoning and task- or domain-specific knowledge. OpenBookQA and ARC-Challenge combine scientific facts with multi-step selection; MedQA emphasizes specialized medical knowledge; StrategyQA emphasizes implicit multi-hop reasoning; and QuaRel focuses on qualitative relations. This diversity is intended to expose both the opportunity for differentiated storage and its boundary cases, rather than to imply that one fixed path must dominate every task.

Only train-side graph-derived K enters the task repositories, except for OpenBookQA, where the benchmark's official open-book facts are combined with K extracted from training graphs. Evaluation questions and labels are not used to construct the repositories. Invalid, unparseable, missing, or out-of-set predictions are scored as incorrect rather than removed from the denominator.

C.2 Comparison Contract and Information Access

Table 9 states what each family is allowed to learn during training and access during inference. Rows in the end-to-end benchmark table are complete systems under their native interfaces and support the

Table 8: Dataset splits and final evaluation protocols. Confirmation sets are used only where listed for protocol verification. StrategyQA uses a labeled evaluation split because its leaderboard test labels are unavailable; the other reported test sets are the corresponding benchmark evaluation splits.

Dataset	Train	Dev	Conf.	Test	Evaluation split	Primary metric
OpenBookQA	4,957	500	–	500	Official test	Exact accuracy
MedQA	8,178	1,000	1,000	1,273	Official test	Exact accuracy
StrategyQA	1,374	343	343	229	Labeled custom test	Exact accuracy
ARC-Challenge	992	213	213	1,172	Official test	Exact accuracy
QuaRel	1,941	278	–	552	Official test	Exact accuracy

competitiveness comparison. Storage-orientation claims use the matched 2×2 placement design, while DAKP claims use the Same-K comparison.

Table 9: Information-access and comparison registry. "Native" indicates that a method retains its intended interface. These rows support system-level comparison; matched placement and Same-K provide the corresponding storage analyses.

Method family	Permitted training and inference interface	Valid interpretation
Direct / zero-shot CoT	No task-specific update; question and choices only at inference.	Prompting reference under the base student.
Answer-SFT	Gold answers on task training examples; no external repository at inference.	Parametric answer-supervision reference.
Rationale distillation	Teacher-produced complete rationales on the same task examples; no external repository at inference.	Parametric distillation reference under a native rationale target.
Standard / gated RAG	No task adapter; question, choices, and evidence from the frozen dataset-specific retriever. The gate, where used, is selected on development data.	Retrieval-system reference under a native evidence interface.
PRR	Train-side teacher–student collaboration and method-specific retrieved or collaborative context.	Collaborative reference under its native interface.
KSplit–$\mathcal{R}$	Graph-derived R adapter; question and choices, without external K, at inference.	Fixed parametric-R path.
KSplit–$\mathcal{K}$	Matched knowledge-conditioned pipeline with retrieved K.	Fixed external-K path.
KSplit–$\mathcal{RK}$	Parametric R plus retrieved K in the matched task pipeline.	Complete fixed KSplit allocation.
CAKR	Same upstream assets as the fixed paths; selects between already materialized $\mathcal{R}$ and $\mathcal{RK}$ predictions after retrieval.	Complementary answer-path selection.

The controlled placement experiment removes this interface ambiguity. For each dataset, $\mathcal{R}_0$ and $\mathcal{K}_0$ expose the two parameter branches without external content, while $\mathcal{RK}$ and $\mathcal{KR}$ add the intended and reversed external containers. All four arms use the same evaluation items and matched assets within the dataset. The direct $\mathcal{RK} - \mathcal{KR}$ contrast measures the two complete configurations; the reported difference-in-differences additionally adjusts for the $\mathcal{R}_0/\mathcal{K}_0$ branch asymmetry. The Same-K test is narrower still: Graph-R and Bag-R begin from the same answer-only checkpoint and use the same 992 OpenBookQA questions, K inventory, update budget, and evaluator. Bag-R uses the same facts through graph-free targets constructed without graph identifiers, edges, paths, or Graph-R targets.

C.3 Frozen Model and Training Registry

The reported trainable experiments use Qwen2.5-3B-Instruct as the student. Teacher APIs construct train-side assets, while benchmark predictions come from the student pipelines. Table 10 records each frozen experimental family separately so that robustness pipelines retain their complete configurations.

Three-seed tables retain the exact pseudorandom seed identifiers recorded by each frozen experiment. Run manifests preserve precision, data and repository hashes, learning rate, epoch count, sequence length, effective batch size, checkpoint identity, and code revision. Changing the teacher API can

Table 10: Implementation and experimental-asset registry. Each row denotes a frozen experimental family; "per-pipeline" means that the complete configuration is compared as a unit.

Run	Teacher source	Student	Training	K access	Answer-path policy
Main	By dataset	Qwen2.5-3B	BF16 LoRA	Dataset repo.	Post-retrieval HGB
Teacher API	GPT / DS / Qwen	Qwen2.5-3B	BF16 LoRA	Frozen repo.	Post-retrieval HGB
Loss weights	Qwen	Qwen2.5-3B	BF16 LoRA	Frozen repo.	Post-retrieval HGB
Ablations	Matched	Qwen2.5-3B	By comparison	Matched	Specified

Unless a frozen pipeline states otherwise, BF16 LoRA uses rank 16, scaling factor 32, and dropout 0.05.

alter several coupled train-side assets, so the robustness comparison evaluates transfer of the complete workflow.

C.4 Reporting, Uncertainty, and Audit Trail

Entries with uncertainty are means $\pm$ sample standard deviations over three fixed seeds. Entries without uncertainty are single completed runs under the stated dataset-specific protocol. Paired contrasts align predictions by question. Single-run contrasts use an item bootstrap; the controlled placement study uses 20,000 crossed seed–question bootstrap replicates and percentile 95% confidence intervals. Boldface in the main result table marks the largest completed point estimate within a benchmark, not statistical significance.

The paper uses a claim-matched reporting hierarchy. End-to-end accuracy supports system viability. The $\mathcal{RK} - \mathcal{KR}$ contrast and DID support the storage-orientation analysis. The Same-K utility and K-probe endpoints support the DAKP separation analysis. CAKR action counts characterize the complementary answer-path component, while the central storage claims come from matched placement and Same-K. Each claim is therefore tied to the most specific matching design available.

For auditability, the frozen release is organized around manifests rather than prose alone. Each reported family is linked to its input split, checkpoint revision where available, retrieval or repository snapshot, prediction records, evaluation receipt, and summary table. The graph construction and export provenance are documented in Appendix B; complete per-seed and diagnostic results appear in Appendix D; robustness and composition states appear in Appendix E. Together these records are intended to let a reader reconstruct what information each system saw, which quantities were selected on development data, and which frozen predictions generated each reported number.

D Complete Evidence for the Core Claims

Evidence is ordered by inferential role. Premise checks motivate differentiated storage; matched placement tests the intended orientation; and the Same-K comparison evaluates DAKP as a graph-conditioned separation treatment. We then provide benchmark-level accounting for the main result table.

D.1 Premise Checks: Why Storage Type Can Matter

The two premise checks address different operational pressures. Parameterizing factual K may make new facts locally available, whereas serving recurring R patterns from an external store may repeatedly transmit the same material. Neither observation selects the better end-to-end orientation; that question is deferred to the matched placement study.

The acquisition result replicates across the original and fixed-extension studies: both intervals are strictly positive. The retention estimates are negative in both studies, while neither independent three-seed interval excludes zero; the pooled six-seed estimate is therefore reported as a sensitivity analysis. The R audit quantifies repeated delivery within the audited pipeline. Because it uses model-assisted references and the frozen low-recall retriever, its latency estimate applies to that audited configuration.

Table 11: Premise checks motivating asymmetric storage. Panel A reports percentage-point differences for parameterizing factual K. Panel B audits repeated delivery when reusable R is externalized. These observations motivate, but do not determine, the preferred storage orientation.

Panel A: Parameterizing factual K

Analysis	Seeds	Endpoint	Difference	95% CI	Frozen gate
Original study	3	K acquisition	$+3.210$	$[+2.655, +3.773]$	Pass
Original study	3	K retention	-0.957	$[-2.216, +0.342]$	Fail
Fixed extension	3	K acquisition	$+3.073$	$[+2.600, +3.555]$	Pass
Fixed extension	3	K retention	-0.672	$[-1.574, +0.171]$	Fail
Pooled sensitivity	6	K retention	-0.814	$[-1.620, -0.034]$	Sensitivity only

Panel B: Externalizing reusable R

Audit scope	Accepted clusters	Questions	Copy events	R-context tokens	Mean latency
Frozen model-assisted v4	59/72	470	$7.966\times$	$7.819\times$	91.995 ms/query

K acquisition is K-SFT minus base-model accuracy on unseen paraphrase probes. K retention is K-SFT minus token-matched Atomic-R-SFT macro-retention accuracy. The reusable-R audit counts 10,010 repeated R-context tokens beyond one de-duplicated copy per accepted cluster.

D.2 Matched Evidence for the Storage Orientation

The placement experiment directly tests the paper's central allocation. The complete contrast compares $\mathcal{RK}$ with the reversed $\mathcal{KR}$ configuration. Because the two parameter branches may differ before external content is added, the difference-in-differences (DID) further adjusts the direct contrast by the corresponding $\mathcal{R}_0$ and $\mathcal{K}_0$ arms. Table 12 presents the dataset-level synthesis before Table 13 exposes the corresponding arm accuracies.

Table 12: Dataset-level synthesis of the controlled placement study. Effects are percentage points with 95% paired-bootstrap intervals. "Supported" requires a positive DID interval; "Directional" denotes a positive DID estimate whose interval includes zero; "Boundary" denotes disagreement between the direct contrast and DID.

Dataset	N	$\mathcal{RK} - \mathcal{KR}$	Accuracy DID	Evidence status
OpenBookQA	400	$+2.00$ $[0.00, +3.92]$	$+2.42$ $[+0.08, +5.00]$	Supported
ARC-Challenge	1,172	$+0.71$ $[-0.03, +1.51]$	$+0.51$ $[-0.09, +1.17]$	Directional
MedQA	900	$+1.04$ $[-0.59, +2.67]$	$+1.00$ $[-0.78, +2.74]$	Directional
StrategyQA	229	$+0.58$ $[-2.04, +3.20]$	$+1.02$ $[-0.87, +2.91]$	Directional
QuaRel	552	$+0.66$ $[-0.24, +1.57]$	-0.48 $[-1.63, +0.66]$	Boundary

The direct $\mathcal{RK} - \mathcal{KR}$ estimate is positive on all five datasets. The branch-adjusted DID is supported on OpenBookQA, remains positive but uncertain on ARC-Challenge, MedQA, and StrategyQA, and is negative but uncertain on QuaRel. The two estimands answer complementary questions: the direct contrast compares the complete configurations, whereas DID isolates the incremental interaction relative to the corresponding parameter branches. Complete-system comparisons therefore consistently favor the intended orientation, while the adjusted interaction remains task-dependent.

For StrategyQA, the three log-odds DIDs average $+0.023$, with a hierarchical paired-bootstrap 95% confidence interval of $[-0.042, +0.083]$. The interval spans zero, matching the directional accuracy-based result.

One mechanism-consistent interpretation is opportunity for complementarity: differentiated storage should matter more when a task needs both reusable operations and external factual knowledge, and less when one source is already sufficient. The cross-dataset pattern is compatible with this interpretation, including the QuaRel boundary, but the study does not independently manipulate or identify that dataset-level moderator.

D.3 DAKP as a Graph-Conditioned Separation Treatment

The primary DAKP test asks whether graph-derived R supervision preserves downstream utility while reducing acquisition of the held-out K. The Same-K comparison holds the starting check-

Table 13: Controlled placement by fixed seed. Accuracies are rounded to two decimals; effects use unrounded records. StrategyQA uses the original 229-example locked test. Status follows Table 12.

Dataset	Seed	$\mathcal{R}_0$	$\mathcal{RK}$	$\mathcal{K}_0$	$\mathcal{KR}$	Accuracy DID	Log-odds DID	Status
OpenBookQA	20260801	86.75	87.50	86.25	85.00	$+2.00$	$+0.297$	Supported
OpenBookQA	20260802	85.00	87.75	86.50	84.50	$+4.75$	$+0.350$	Supported
OpenBookQA	20260803	85.00	85.00	85.25	84.75	$+0.50$	$+0.305$	Supported
ARC-Challenge	20260801	84.47	84.64	84.47	84.22	$+0.43$	-0.006	Directional
ARC-Challenge	20260802	84.30	84.56	83.19	83.36	$+0.09$	$+0.322$	Directional
ARC-Challenge	20260803	83.70	84.13	84.22	83.62	$+1.02$	$+0.522$	Directional
MedQA	20260801	58.67	59.11	58.67	59.11	0.00	$+0.283$	Directional
MedQA	20260802	58.44	60.00	57.78	58.44	$+0.889$	$+0.409$	Directional
MedQA	20260803	57.44	59.44	58.00	57.89	$+2.111$	$+0.412$	Directional
StrategyQA	20260801	68.56	70.31	69.00	69.00	$+1.75$	$+0.009$	Directional
StrategyQA	20260802	72.93	73.80	74.67	75.11	$+0.44$	$+0.031$	Directional
StrategyQA	20260803	72.05	71.62	71.18	69.87	$+0.87$	$+0.029$	Directional
QuaRel	20260801	92.93	92.57	92.21	92.39	-0.54	-0.594	Boundary
QuaRel	20260802	93.66	93.12	91.67	92.39	-1.27	-0.118	Boundary
QuaRel	20260803	92.57	92.93	91.85	91.85	$+0.36$	-0.098	Boundary

point, 992 training questions, K inventory, update budget, and evaluator fixed. Graph-R uses the complete supervision treatment compiled from accepted typed graphs; Bag-R uses graph-free targets constructed without graph identifiers, edges, paths, or Graph-R targets.

Table 14: DAKP evidence on development or dedicated probe data. Panel A is the primary Same-K treatment comparison; Panel B is a separate model-assisted typing diagnostic.

Panel A: OpenBookQA Same-K comparison (three seeds)

Endpoint	Graph-R	Same-K Bag-R	Difference	95% CI	Decision
Dev500 accuracy (%)	86.47	85.33	$+1.13$	$[-0.80, +3.07]$	Non-inferior
200-K gold log-odds	3.391	3.905	-0.514	$[-0.784, -0.318]$	Lower acquisition

Panel B: Model-assisted node-typing diagnostic ($N = 400$)

Input	Accuracy	Macro-F1	K recall	R recall	Interpretation
Node text only	94.00	94.24	91.00	97.00	Reference condition
Node + local dependency text	85.00	84.78	73.00	97.00	-9.46 F1 points

Panel A uses a hierarchical paired bootstrap and a pre-specified -1-point utility non-inferiority margin. After v2 did not meet the lower-acquisition gate, the v3 design was finalized and frozen before v3 evaluation. Panel B uses balanced K/R nodes, model-assisted references rather than human gold, and a stratified paired bootstrap for the Macro-F1 difference.

Graph-R meets both pre-specified criteria. Its lower utility interval is -0.80, above the -1-point non-inferiority margin, while the K-probe log-odds difference is strictly negative. Under this treatment, graph-derived supervision therefore retains downstream utility while reducing acquisition of knowledge assigned to the external store. The estimand is the complete graph-conditioned supervision treatment: Bag-R targets are 57.8% longer on average, so organization, target length, and surface form remain bundled. At inference, the student uses the compiled assets rather than executing graph adjacency.

The typing diagnostic serves a narrower purpose. Node text alone achieves 94.24 Macro-F1 against model-assisted references, but naively appending local dependency text reduces Macro-F1 to 84.78. Dependency text is therefore not a simple classification shortcut; DAKP uses dependencies during construction and validation rather than concatenating them into the typing input. With model-assisted rather than human-gold references, the diagnostic supports operational separability in the evaluated construction pipeline.

D.4 Supplementary Benchmark Accounting

Table 15 expands the StrategyQA system results into their three fixed seeds. Table 16 then records the single-run StrategyQA retrieval baselines and the ARC-Challenge seeds underlying the fixed-path means.

Table 15: StrategyQA results on the 229-example labeled evaluation split. Entries give correct/total and accuracy by fixed seed; aggregates use unrounded accuracies.

Method	20260801	20260802	20260803	Mean ± SD
Fixed Base	145/229 (63.32)	145/229 (63.32)	145/229 (63.32)	63.32
Fixed $\mathcal{R}$	162/229 (70.74)	163/229 (71.18)	173/229 (75.55)	72.49 ± 2.66
Fixed $\mathcal{K}$	157/229 (68.56)	157/229 (68.56)	164/229 (71.62)	69.58 ± 1.76
Fixed $\mathcal{RK}$	156/229 (68.12)	167/229 (72.93)	167/229 (72.93)	71.32 ± 2.77
CAKR	163/229 (71.18)	163/229 (71.18)	172/229 (75.11)	72.49 ± 2.27

Table 16: Supporting benchmark accounting: single-run StrategyQA retrieval baselines on the 229-example evaluation split and ARC-Challenge fixed paths by seed; aggregates use unrounded accuracies.

Panel A: StrategyQA retrieval baselines

Method	Correct	Accuracy
Standard RAG	122/229	53.28
Confidence-gated RAG	129/229	56.33

Panel B: ARC-Challenge fixed paths

Seed	Fixed $\mathcal{R}$	Fixed $\mathcal{RK}$
20260801	84.56	85.41
20260802	84.64	84.73
20260803	84.39	85.24
Mean ± SD	84.53 ± 0.13	85.13 ± 0.36

Evidence summary. The premise checks motivate differentiated storage, the matched study favors the intended complete orientation directly on all five tasks, and Same-K supports DAKP as a utility-preserving treatment with lower K acquisition. Branch-adjusted evidence remains task-dependent: supported on OpenBookQA, directional on three tasks, and a boundary on QuaRel. The central claims therefore attach to the direct contrast, dataset-specific DID estimates, and the complete graph-conditioned treatment; pooled retention, edge-level attribution, and CAKR compute effects remain outside their scope.

E Robustness, Sensitivity, and Boundary Cases

This appendix separates robustness of the fixed storage paths from behavior of the complementary post-retrieval component. We first vary the teacher API, then examine sensitivity to the training objective, isolate QuaRel as a boundary case, and finally report CAKR diagnostics.

E.1 Transfer across Teacher APIs

Figure 4 summarizes teacher-workflow transfer, objective sensitivity, and the Safe-$\mathcal{RK}$ follow-up. Table 3 reports the exact transfer values in the main paper, while Table 17 retains the objective-sensitivity results.

Panel **(a)** visualizes the accuracy gap to the strongest fixed path within each teacher workflow. The ordering changes from $\mathcal{RK}$ with GPT to $\mathcal{K}$ with DeepSeek and an $\mathcal{R}$/$\mathcal{RK}$ tie with Qwen, showing workflow-level robustness alongside a pipeline-dependent preferred policy.

E.2 Sensitivity to the Training Objective

The frozen five-point sweep varies the answer/interface loss ratio. The fixed paths provide the valid cross-ratio comparison; the 15 original CAKR runs share first-stage training lineage and are therefore excluded from that comparison. A separate zero-parameter Safe-$\mathcal{RK}$ follow-up provides a lineage-matched comparison at the two ratios where it was evaluated.

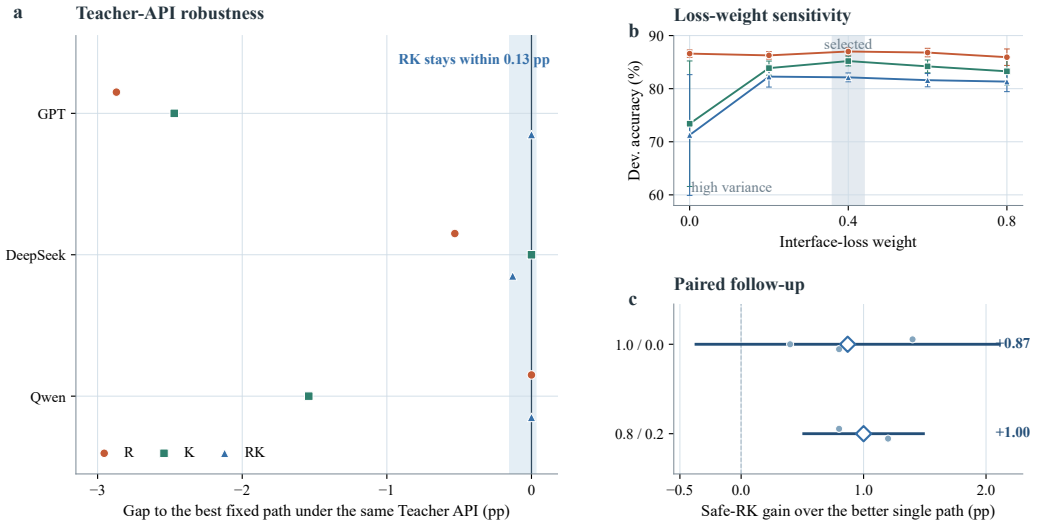


Figure 4: Robustness of the fixed type-aware storage paths on OpenBookQA. **(a)** Mean accuracy gaps to the best fixed path under each teacher API; values closer to zero are better. **(b)** Development accuracy across the original answer/interface loss-weight sweep (mean ± sample standard deviation over three seeds); the shaded setting was selected by the frozen development rule. **(c)** Seed-level Safe-$\mathcal{RK}$ gains over the better of fixed $\mathcal{R}$ and $\mathcal{K}$ in a separate follow-up; diamonds show means and horizontal lines show seed-t 95% confidence intervals. Panel **(c)** reports a separate follow-up.

Table 17: Development-set sensitivity to the answer/interface loss weights. Values are mean accuracy percentages ± sample standard deviation over three seeds.

Panel A: Original frozen five-point sweep

Answer / interface	Fixed $\mathcal{R}$	Fixed $\mathcal{K}$	Fixed $\mathcal{RK}$
1.0 / 0.0	86.60 ± 0.69	73.40 ± 11.82	71.27 ± 11.37
0.8 / 0.2	86.27 ± 0.70	83.87 ± 1.33	82.27 ± 1.97
0.6 / 0.4	87.00 ± 0.69	85.20 ± 0.92	82.13 ± 0.83
0.4 / 0.6	86.80 ± 0.80	84.20 ± 1.20	81.60 ± 1.25
0.2 / 0.8	85.93 ± 1.55	83.27 ± 2.60	81.33 ± 1.89

Panel B: Separate zero-parameter Safe-$\mathcal{RK}$ follow-up

Answer / interface	Safe-$\mathcal{RK}$	Gain over $\mathcal{R}$	Gain over $\mathcal{K}$	All seeds $> \mathcal{R}, \mathcal{K}$
1.0 / 0.0	87.47 ± 0.42	$+0.87$ $[+0.27, +1.53]$	$+14.07$	Yes
0.8 / 0.2	87.27 ± 0.50	$+1.00$ $[+0.07, +1.93]$	$+3.40$	Yes

The 0.6/0.4 ratio was selected by the frozen development rule with a 0.20-point margin. Brackets in Panel B are 95% question-cluster bootstrap confidence intervals for Safe-$\mathcal{RK}$ minus fixed $\mathcal{R}$. Safe-$\mathcal{RK}$ was evaluated only at the two listed ratios; its frozen-development CAKR policy did not exceed Safe-$\mathcal{RK}$.

Fixed $\mathcal{R}$ remains within a 1.07-point range across the sweep, whereas fixed $\mathcal{K}$ and $\mathcal{RK}$ become unstable when the interface loss is removed. The Safe-$\mathcal{RK}$ follow-up recovers a positive advantage over both single-store paths at the two evaluated ratios. These results identify a workable neighborhood of objectives together with clear sensitivity to the loss design.

E.3 QuaRel as a Boundary Case

QuaRel separates the storage result from the routing result. Fixed $\mathcal{RK}$ exceeds both single-store paths in each seed, but the branch-adjusted placement interaction in Table 12 remains the paper's boundary result. CAKR acts on only 72 of 1,656 seed-question pairs, and its five repairs are nearly offset by four harms.

Table 18: QuaRel per-seed results underlying the aggregated main comparison. Fixed $\mathcal{RK}$ and CAKR include the frozen label-blind structural-validity rule; CAKR actions remain those selected by the frozen out-of-fold policy.

Seed	Fixed $\mathcal{R}$	Fixed $\mathcal{K}$	Fixed $\mathcal{RK}$	CAKR	$\mathcal{R}$ choices	$\mathcal{RK}$ choices	$\mathcal{R}$ rate	Repairs	Harms	N
20260801	91.85	90.22	92.39	92.21	20	532	3.62%	0	1	552
20260802	89.67	90.40	91.30	91.30	31	521	5.62%	2	2	552
20260803	91.67	89.86	92.39	92.75	21	531	3.80%	3	1	552
Mean / total	91.06 ± 1.21	90.16 ± 0.28	92.03 ± 0.63	92.09 ± 0.73	72	1,584	4.35%	5	4	1,656

Additional QuaRel diagnostic. QuaRel is especially sensitive to the generation and parsing interface. In a separate deterministic diagnostic, the batch-8, both-order strict-parser run obtains 415/552 correct (75.18%), while requiring agreement between the two generated orders gives 350/552 (63.41%). A descriptive confidence gate selects threshold 0.58 on the official development set (192/278, 69.06%; 46 retrieval routes) and applies it unchanged on test, obtaining 354/552 (64.13%) with 100 retrieval routes, eight repairs, and four harms. We report these auxiliary runs as interface diagnostics; the main table follows its fixed benchmark protocol, and QuaRel remains the boundary case for adjusted placement.

E.4 CAKR Behavior and Residual Headroom

The OpenBookQA diagnostic quantifies the behavior of the current CAKR policy. CAKR changes seven of 1,500 seed-question decisions and yields a $+0.27$-point mean difference from fixed $\mathcal{RK}$; the crossed seed-question bootstrap interval includes zero.

Table 19: OpenBookQA CAKR behavior and execution diagnostics. Correct counts use the complete 500-question denominator per seed.

Seed	Fixed $\mathcal{RK}$	CAKR	Δ (pp)	$\mathcal{R}$ choices	Repairs	Harms	Oracle	Policy
20260801	461/500	462/500	$+0.20$	2	1	0	466/500	Frozen
20260802	457/500	456/500	-0.20	1	0	1	465/500	Frozen
20260803	453/500	457/500	$+0.80$	4	4	0	464/500	Frozen
Mean / total	91.40 ± 0.80	91.67 ± 0.64	$+0.27$	7 (0.47%)	5	1	93.00%	HGB

The crossed seed-question bootstrap interval for CAKR minus fixed $\mathcal{RK}$ is $[-0.27, +1.00]$ percentage points. Repairs and harms count items corrected or broken by choosing $\mathcal{R}$ instead of the default $\mathcal{RK}$ prediction. The oracle uses gold labels after both predictions are available and is diagnostic only. Retrieval, candidate scoring, and both answer paths execute before CAKR selection, so route shares measure selection behavior rather than compute savings.

The oracle reaches 93.00%, leaving residual decision headroom, while the frozen CAKR policy captures a small and uncertain portion of it. CAKR therefore complements the fixed storage paths through answer-path composition and exposes a distinct future routing problem.

What the robustness evidence establishes. Taken together, the fixed storage paths remain viable across the evaluated teacher APIs, with an ordering that depends on the complete pipeline. The loss study identifies sensitivity when interface supervision is removed, QuaRel remains the adjusted-

placement boundary, and CAKR gains remain setting-dependent. These results define the robustness scope of the separation-and-storage workflow.

F Extended Related Work

F.1 Positioning by the Design Decision

KSplit lies at the intersection of retrieval-augmented generation, reasoning distillation, and structured reasoning. Each neighboring line optimizes a distinct design decision; Table 20 locates KSplit by the decision it makes explicit. Its contribution is to organize established components around a typed placement question: which information in a teacher solution should remain externally editable, and which should become a reusable capability of the student?

Table 20: Positioning by the primary design decision represented and optimized by each line of work.

Line of work	Primary design decision	Relation to KSplit
Retrieval-augmented generation	Which external evidence to expose	Provides external memory; KSplit adds typed allocation from a teacher solution.
Reasoning-aware retrieval	How evidence supports rationale generation	Pairs retrieval with reasoning; KSplit compiles teacher traces into K/R storage assets.
Adaptive retrieval	When to retrieve or rely on context	Operates after internal and external sources already exist.
Rationale distillation and PEFT	How teacher behavior enters student parameters	KSplit first selects typed content eligible for parametric transfer.
Structured reasoning	How to organize intermediate computation	DAKP uses structure to construct and validate storage allocation.
KSplit	**Which information type should be stored where**	External K, parametric R, and complementary post-retrieval answer-path selection.

F.2 External Knowledge, Reasoning-Aware Retrieval, and Retrieval Control

Retrieval-augmented language models combine parametric generation with an external corpus, allowing factual information to be accessed without writing every update into model weights (Lewis et al., 2020; Borgeaud et al., 2022). These methods establish external memory as a practical knowledge interface. KSplit operates at the preceding allocation step, typing teacher-generated solution content as factual assets or reusable operations before assigning it to storage.

Reasoning-aware retrieval and distillation are the closest systems-level precedents. IRCoT alternates retrieval with chain-of-thought generation so that intermediate reasoning guides later evidence acquisition (Trivedi et al., 2023). KARD trains a student to generate a teacher rationale from a question and retrieved knowledge, and trains a reranker to prioritize passages useful for rationale generation (Kang et al., 2023). Probe Then Retrieve and Reason distills a probing model that constructs retrieval queries and a reasoning model that uses the retrieved knowledge (Zhao et al., 2024). Generate then Ground forms intermediate questions and candidate answers before grounding them in external evidence (Shi et al., 2024). These methods show that retrieval should support the reasoning process rather than merely match the input question.

KSplit is complementary to this progression. DAKP analyzes a teacher solution before student training, assigns interleaved units functional K or R roles, and checks their dependencies. FKP then compiles two storage-specific assets: an external K repository with a K-use interface, and R supervision for parameter-efficient adaptation. KARD and Probe Then Retrieve and Reason establish the value of pairing external knowledge with a distilled reasoner; KSplit makes the placement rule inside that pairing explicit and auditable.

Adaptive retrieval addresses a later decision. FLARE uses uncertainty during generation to trigger retrieval (Jiang et al., 2023); Adaptive RAG selects among no retrieval, single-step retrieval, and iterative retrieval according to question complexity (Jeong et al., 2024); and DRAGIN determines when and what to retrieve from information needs that arise during generation (Su et al., 2024). Sufficient Context distinguishes inadequate evidence from failure to use adequate evidence (Joren et al., 2025), while SeaKR uses internal uncertainty to assess the need for external knowledge (Yao et al., 2025). DeepRAG chooses between parametric knowledge and retrieval for individual subproblems (Guan et al., 2026), and Parameters versus Context modulates reliance when parametric and contextual knowledge conflict (Bi et al., 2026). Their central question is when or how to use

already available sources; KSplit's central question is what should be placed in each source. CAKR operates at a later stage still: in the evaluated pipeline, retrieval and both candidate answer paths execute before it selects between R and RK predictions, so its route rate measures answer-path selection rather than skipped retrieval or compute savings.

F.3 Reasoning Internalization and Parameter-Efficient Adaptation

Explanation-based distillation transfers teacher-generated instructions, rationales, or intermediate steps into smaller models (Mukherjee et al., 2023; Li et al., 2023; Shridhar et al., 2023; Hsieh et al., 2023). These studies establish that reasoning traces can provide supervision beyond final answers. Parameter-efficient methods such as LoRA make the transfer practical without updating the entire base model (Hu et al., 2021). Later work improves the form or efficiency of internalization: SuperCorrect uses structured correction traces (Yang et al., 2025), MoDE CoTD uses multiple LoRA reasoning experts to reduce cross-task interference (Li et al., 2024), ReFT intervenes in internal representations (Wu et al., 2024), and Tina studies low-cost reasoning adaptation with LoRA (Wang et al., 2026).

These methods primarily improve the supervision signal, parameterization, or optimization used to internalize behavior. Representative rationale-distillation objectives treat a complete trace as the supervision object, allowing factual statements, answer-bearing details, teacher-specific phrasing, and reusable operations to enter the same target. KSplit intervenes one stage earlier: DAKP determines the functional type of each unit, and FKP trains the student on graph-derived R supervision while keeping K externally editable. LoRA is an implementation choice in our experiments; other adaptation mechanisms could consume the same R asset. The R designation is task- and pipeline-relative and is evaluated through the matched placement and Same-K interventions.

F.4 Structured Reasoning and Memory–Reasoning Separation

Structured reasoning methods expose intermediate computation through plans, search structures, dependency graphs, or executable programs. Plan-and-Solve separates planning from step execution (Wang et al., 2023); Tree of Thoughts searches and backtracks over candidate reasoning trajectories (Yao et al., 2023); Graph of Thoughts represents dependencies among intermediate thoughts (Besta et al., 2024); and Faithful Chain-of-Thought translates natural-language analysis into an executable symbolic procedure (Lyu et al., 2023). Their structures support search, control, or faithfulness. The K-R graph instead records typed information units and dependencies so that accepted graphs can be audited and compiled into storage-specific training and retrieval assets. At inference, the student consumes these compiled assets; graph construction and validation occur during asset creation.

The closest conceptual distinction is between memory and reasoning themselves. Disentangling Memory and Reasoning Ability decomposes inference into memory-recall and reasoning actions, showing why the two capabilities should be analyzed separately (Jin et al., 2025). KSplit operationalizes a related distinction for LLM–SLM transfer, with storage allocation as its object. K denotes factual or update-sensitive information used by the current solution; R denotes operations intended to transfer across instances in the evaluated setting. The labels are functional and relative to the task, student, and evidence pipeline.

Taken together, prior work supplies powerful mechanisms for external memory, reasoning supervision, structured computation, and adaptive source use. KSplit connects these mechanisms through an experimentally testable allocation decision: converting a teacher solution into typed, dependency-linked assets and assigning them to parametric or external storage. The retriever, LoRA optimizer, graph formalism, and complementary answer-path selector support this central separation-and-storage contribution.