Semantic Compression for Sound and Complete Query Answering over Knowledge Graphs (Supplementary Materials)

I. Formal Proofs

A. Proof of Lemma 1

Lemma 1. Let $\mathcal{G}$ be an arbitrary KG and Γ a finite set of rules. Suppose $\mathrm{ConstructCompressionConfigs}(\mathcal{G}, \Gamma, N_1)$ returns two sequences $\langle \gamma_j \rangle_{j=1}^{k}$ and $\langle \Delta_j \rangle_{j=1}^{k}$, where $k \leq N_1$. Define the set of candidate rules at step i by $\Gamma_i^\ddagger = \{\gamma_j\}_{j=1}^{i}$, the corresponding compressed KG at step i by $\mathcal{G}_i^\ddagger = \mathcal{G} \setminus \bigcup_{j=1}^{i} \Delta_j$. Then, for any $i \in \{1, \ldots, k\}$, the following properties hold:

Rewritability: $\Gamma_i^\ddagger$ belongs to the FUS fragment.

Recoverability: The least model of $\mathcal{G}_i^\ddagger \cup \Gamma_i^\ddagger$ equals the original $\mathcal{G}$, i.e., $\Phi_{\mathcal{G}_i^\ddagger \cup \Gamma_i^\ddagger} = \mathcal{G}$.

Minimality: For any proper subset $\mathcal{G}' \subset \mathcal{G}_i^\ddagger$, the least model of $\mathcal{G}' \cup \Gamma_i^\ddagger$ is a proper subset of $\mathcal{G}$.

Non-redundancy: No rule in $\Gamma_i^\ddagger$ is subsumed by any other in $\Gamma_i^\ddagger$.

Proof. Fix any $i \in \{1, \ldots, k\}$ and recall $\Gamma_i^\ddagger = \{\gamma_j\}_{j=1}^{i}$ and $\mathcal{G}_i^\ddagger = \mathcal{G} \setminus \bigcup_{j=1}^{i} \Delta_j$.

Rewritability. By construction, $\mathrm{ConstructCompressionConfigs}$ appends γ_i only if $\mathrm{IsFUS}(\Gamma_i^\ddagger)$ holds. Hence $\Gamma_i^\ddagger$ belongs to the FUS fragment.

Recoverability. We show $\Phi_{\mathcal{G}_i^\ddagger \cup \Gamma_i^\ddagger} = \mathcal{G}$ via two inclusions.

($\subseteq$) Each rule in $\Gamma_i^\ddagger$ is validated to have confidence $f_\gamma = 1$, i.e., closing $\mathcal{G}$ under $\Gamma_i^\ddagger$ derives no facts beyond $\mathcal{G}$, hence $\Phi_{\mathcal{G} \cup \Gamma_i^\ddagger} = \mathcal{G}$. Since $\mathcal{G}_i^\ddagger \subseteq \mathcal{G}$, monotonicity of the least-model operator gives

$$\Phi_{\mathcal{G}_i^\ddagger \cup \Gamma_i^\ddagger} \subseteq \Phi_{\mathcal{G} \cup \Gamma_i^\ddagger} = \mathcal{G}.$$

($\supseteq$) By Definition 1, each removed set Δ_j is entailed by the remaining triples together with the current rule prefix, i.e., $\mathcal{G}_i^\ddagger \cup \Gamma_i^\ddagger \models \Delta_j$ for all $1 \leq j \leq i$. Therefore $\bigcup_{j=1}^{i} \Delta_j \subseteq \Phi_{\mathcal{G}_i^\ddagger \cup \Gamma_i^\ddagger}$. Moreover $\mathcal{G}_i^\ddagger \subseteq \Phi_{\mathcal{G}_i^\ddagger \cup \Gamma_i^\ddagger}$ holds by definition of least model. Thus,

$$\mathcal{G} = \mathcal{G}_i^\ddagger \cup \bigcup_{j=1}^{i} \Delta_j \subseteq \Phi_{\mathcal{G}_i^\ddagger \cup \Gamma_i^\ddagger}.$$

Combining both directions yields $\Phi_{\mathcal{G}_i^\ddagger \cup \Gamma_i^\ddagger} = \mathcal{G}$.

Minimality. Let $\mathcal{G}' \subset \mathcal{G}_i^\ddagger$ be any proper subset and pick $\delta \in \mathcal{G}_i^\ddagger \setminus \mathcal{G}'$. It suffices to show $\delta \notin \Phi_{\mathcal{G}' \cup \Gamma_i^\ddagger}$, because then $\Phi_{\mathcal{G}' \cup \Gamma_i^\ddagger} \subsetneq \mathcal{G}$ (not containing $\delta \in \mathcal{G}$).

Since $\mathcal{G}' \subseteq \mathcal{G}_i^\ddagger \setminus \{\delta\}$, by monotonicity we have

$$\Phi_{\mathcal{G}' \cup \Gamma_i^\ddagger} \subseteq \Phi_{(\mathcal{G}_i^\ddagger \setminus \{\delta\}) \cup \Gamma_i^\ddagger}.$$

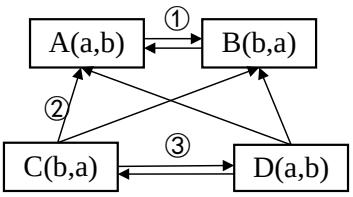


Fig. 1: Example dependency graph of triples and rules: boxes represent triples and arrows represent rules.

Hence it remains to prove

$$(\mathcal{G}_i^\ddagger \setminus \{\delta\}) \cup \Gamma_i^\ddagger \not\models \delta. \qquad (\star)$$

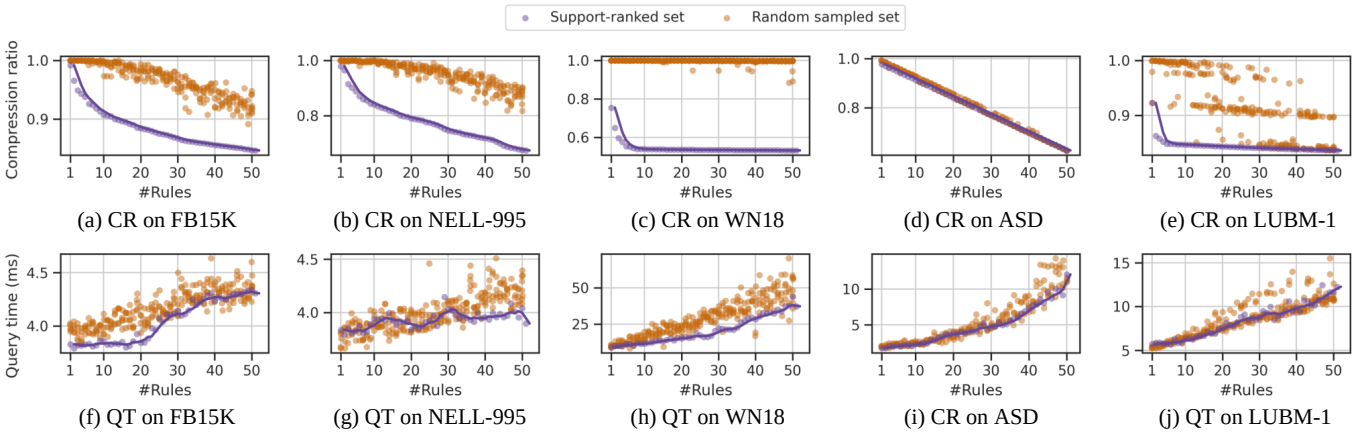
Assume to the contrary that $(\mathcal{G}_i^\ddagger \setminus \{\delta\}) \cup \Gamma_i^\ddagger \models \delta$. Let $t \leq i$ be the *smallest* index such that $(\mathcal{G}_i^\ddagger \setminus \{\delta\}) \cup \Gamma_t^\ddagger \models \delta$. Then δ becomes redundant *at step* t after adding γ_t. Moreover, this can only happen when the head predicate of γ_t matches the predicate of δ (otherwise adding γ_t cannot derive facts with $\mathrm{pred}(\delta)$). At step t, $\mathrm{IdentifyRedundantTriples}(\mathcal{G}_{t-1}^\ddagger, \Gamma_t^\ddagger, \gamma_t)$ checks all triples in $\mathcal{G}_{t-1}^\ddagger$ with that head predicate and removes exactly those that are entailed by the remaining triples under $\Gamma_t^\ddagger$. Since δ is such an entailed triple at step t, it must be included in Δ_t and thus removed, contradicting $\delta \in \mathcal{G}_i^\ddagger$. Therefore $(\star)$ holds, and minimality follows.

Non-redundancy. Assume for contradiction that there exist distinct rules $\gamma, \gamma' \in \Gamma_i^\ddagger$ such that γ is subsumed by γ'. Consider the moment when the later-added one among $\{\gamma, \gamma'\}$ was considered for inclusion in the prefix. At that time, the earlier one was already in the prefix and subsumed it. By the construction (subsumption-based pruning before acceptance), a rule that is subsumed by an already-accepted rule is not appended to the prefix (either it is discarded directly, or its redundant-triple set is empty and it is skipped). Hence the later-added rule could not be in $\Gamma_i^\ddagger$, contradicting the assumption. Therefore no rule in $\Gamma_i^\ddagger$ is subsumed by another rule in $\Gamma_i^\ddagger$. □

II. Illustrative Example

A. Triple removal example

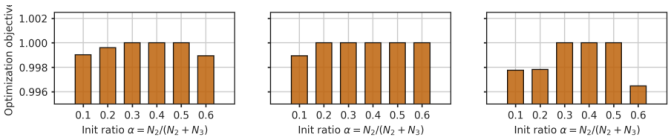
As an illustrative example, we examine how redundant triples are detected in the dependency graph of Figure 1 at Example 1.



(a) CR on FB15K (b) CR on NELL-995 (c) CR on WN18 (d) CR on ASD (e) CR on LUBM-1

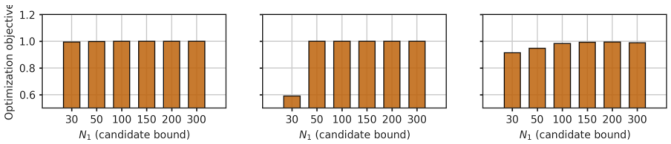
(f) QT on FB15K (g) QT on NELL-995 (h) QT on WN18 (i) CR on ASD (j) QT on LUBM-1

Fig. 2: Comparison results between different rule selection strategies in terms of compression ratio and query answering time against increasing number of selected rules.



(a) WN18 (b) ASD (c) LUBM-1

Fig. 3: Comparison results in terms of the optimization objective for different α.



(a) WN18 (b) ASD (c) LUBM-1

Fig. 4: Comparison results in terms of the optimization objective for different N_1.

Example 1. To detect redundant triples in Figure 1, `SInC` [1], [2] treats the task as a feedback-vertex-set problem. It retains the highest-degree triple in each strongly connected component and thus keeps A and C even though rule ② can infer A. `QSC` iteratively adds rules and, after each addition, removes all triples entailed by the union of the added rules and the current triple set. This process ensures that no redundant triple is overlooked, regardless of rule addition order. To illustrate why entailment checking is necessary, consider the rule addition order ③ → ② → ①. After adding ③, D is removed. After adding ②, A is removed. Although A's removal prevents the direct inference of B, B remains redundant since it can be inferred from C via rules ③ and ①. Therefore, confirming B's redundancy requires an entailment check.

III. Additional Experimental Results

A. Effectiveness of Greedy Construction of Candidate Configurations.

We compared our greedy strategy, denoted *support-ranked set*, for constructing candidate configurations, which produces top-ranked rules in descending order of support counts, with the uniformly random sampling strategy, denoted *random sampled set*, against an increasing number of rules. For each number of rules, we generated five random sets of rules and measured both the compression ratio and the query answering time. To eliminate the impact of system jitters in estimating query answering time, we ran all test queries five times and counted the average results.

As shown in Figure 2, the support-ranked sets achieve significantly better compression ratios than randomly sampled sets across all datasets, with the exception of ASD. On ASD, all sampled sets of rules showed similar compression, which stems directly from the support of rules being uniformly distributed. In terms of query answering time, support-ranked sets also exhibit an advantage over randomly sampled sets across all datasets. These findings validate that restricting the candidate space to support-ranked sets is an effective design choice. By narrowing the search space, `QSC` facilitates more efficient and targeted exploration of the optimal compression configuration.

B. Hyperparameter sensitivity.

We examine sensitivity to the BO hyperparameters. In the main paper, we vary the total evaluation budget $B = N_2 + N_3$ and report the achieved objective. In the supplementary material, we additionally vary the budget split $\alpha = N_2/(N_2 + N_3)$ (Fig. 3) and the candidate bound N_1 (Fig. 4).

Across datasets, the objective is largely insensitive to α in the tested range. In particular, $\alpha \in [0.3, 0.5]$ consistently attains near-optimal objectives, while extreme values can be slightly worse on some KGs (e.g., WN18 and LUBM-1), since they allocate too much budget to either the initial sampling stage or the BO iterations.

We also find that increasing N_1 yields diminishing returns once the candidate set is sufficiently large. WN18 is essentially unchanged across N_1, ASD stabilizes once $N_1 \geq 50$, and LUBM-1 improves up to about $N_1 \approx 150$ and then saturates.

Overall, these results suggest that $B \approx 20$ with $\alpha \in [0.3, 0.5]$ and $N_1 = 300$ is a safe default for a stable cost–quality trade-off.

IV. Implementation Details

A. Default Hyperparameters

Rule miner. For all datasets, we set the minimum support to 5, the maximum rule length to 5, the mining time limit to 100 s, and allow constants in rules.

Bayesian optimization. For all datasets, we set $N_1 = 300$, $N_2 = 10$, and $N_3 = 10$.

B. AQ Cache Design

Atomic queries (AQs) are frequently repeated across workloads. To avoid redundant rewriting, we implement a lightweight AQ cache that memoizes rewriting results offline.

Specifically, each distinct AQ is rewritten once using $\mathrm{FUSRewrite}$, and the resulting UCQ is stored in the cache. During query answering, we reuse the cached rewriting result, substitute the AQ variables with the concrete terms from the query, and answer the instantiated UCQ over the compressed KG. Since rewriting is performed offline and amortized across queries, its cost does not affect online query answering performance.

For example, consider a rewriting rule $A(X, Y) \leftarrow B(X, Z), C(X, Y)$. Given an AQ $A(a, X)$, we apply the substitution $\sigma = \{X \mapsto a,\ Y \mapsto X\}$ to obtain the instantiated UCQ $B(a, Z), C(a, X)$.

References

[1] R. Wang, D. Sun, R. K. Wong, R. Ranjan, and A. Y. Zomaya, "Sinc: Semantic approach and enhancement for relational data compression," *Knowl. Based Syst.*, vol. 258, p. 110001, 2022.

[2] R. Wang, D. Sun, and R. K. Wong, "Symbolic minimization on relational data," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 9, pp. 9307–9318, 2023.