Appendix: Technical Comparison with Rule Mining Systems

To better contextualize the contributions of our method, we provide a technical comparison between MATILDA and three competing rule mining approaches: AMIE3 [1], Popper [2], and SPIDER [3]. We focus on key algorithmic components: *candidate generation*, *pruning strategies*, and *evaluation metrics*.

Candidate Rule Generation

MATILDA. Our method constructs candidate TGDs through a symbolic traversal of the *constraint graph*, where each node represents a table-attribute pair, and edges encode potential variable bindings via joinable attributes. Candidate rules are generated by enumerating connected subgraphs that correspond to join paths, followed by assembling logical atoms for the body and head of a potential TGD.

AMIE3. AMIE3 employs a *rule expansion tree*, where nodes correspond to Horn rule prefixes, and edges represent valid rule extensions (i.e., adding one atom to the body or head while maintaining variable connectivity). Rule construction is guided by a beam search limited by variable and atom counts, ensuring connectedness and grounding of head variables. This syntactic enumeration is efficient but tailored to Horn rules with a single atom in the head.

Popper. Popper adopts a *generate-and-test* paradigm implemented via Answer Set Programming (ASP). Candidate hypotheses (sets of clauses) are enumerated via a declarative bias, including constraints on maximum literals, recursion, and typing. Popper does not explicitly model joinability but indirectly captures it via the declarative language bias and background knowledge.

SPIDER. SPIDER focuses solely on unary inclusion dependencies (INDs) and does not operate over rule bodies or logical implications. Candidates are simply all attribute pairs (A, B) such that $|\text{dom}(A)| \leq |\text{dom}(B)|$, and inclusion is tested via sorted set comparison. Unlike MATILDA, it does not construct full logical rules but instead discovers set-containment patterns.

Pruning Techniques

MATILDA. Pruning is performed at multiple stages:

- **Structural pruning** via the constraint graph limits rule enumeration to connected, joinable patterns.
- **Type pruning** filters out atom combinations with incompatible domains.
- **Support-based pruning** eliminates candidates that fail to reach a minimum frequency threshold early in the generation process.

AMIE3. AMIE3 uses a combination of syntactic and semantic pruning:

- **Redundancy pruning** eliminates logically equivalent rule variations (e.g., permuted atoms).
- **Head coverage and confidence pruning** halts expansion of rule prefixes whose statistics fall below user-defined thresholds.
- **Closedness constraints** ensure that all variables in the rule are appropriately bound.

Popper. Popper uses a *failure-driven* pruning mechanism:

- Each failed hypothesis generates constraints that prune its generalizations (if too specific) or specializations (if too general).
- Constraints are incrementally accumulated, reducing the hypothesis space without requiring full enumeration.

SPIDER. Pruning in SPIDER is minimal but includes:

- **Domain size heuristics** to skip infeasible comparisons.
- **Sorting-based early termination**, leveraging the ordering of distinct values to detect non-inclusions efficiently.

Rule Evaluation and Verification

MATILDA. We evaluate rule candidates using classical support and confidence metrics. Verification of rule satisfaction is implemented in SQL, allowing scalable and direct execution over large relational databases. Additionally, MATILDA supports *approximate* rules, tolerating exceptions, which broadens the applicability to noisy real-world data.

AMIE3. AMIE3 also computes exact support and confidence values, distinguishing itself by its *Partial Completeness Assumption (PCA)* to estimate confidence in the presence of incomplete KBs. This makes AMIE3 especially suitable for open-world settings like Wikidata or YAGO.

Popper. Evaluation in Popper is binary: a hypothesis is either consistent with all positive and negative examples or not. There is no graded notion of rule quality such as confidence or support. Rule correctness is verified via Prolog-based simulation of the background knowledge and examples.

SPIDER. Rule evaluation reduces to checking whether $s(A) \subseteq s(B)$ holds. The notion of confidence is not applicable, as SPIDER discovers only exact unary INDs and does not consider approximate or multi-attribute dependencies.

Summary of Contributions in Context

Compared to these systems, MATILDA provides the following unique features:

- **Generalization of Rule Forms:** Unlike AMIE3 and SPIDER, which are limited to Horn rules and INDs respectively, MATILDA supports TGDs with multiple atoms in both body and head, including existential quantification.

- **Unified Graph-Based Framework:** MATILDA's constraint graph serves as both a joinability index and a structural filter for rule generation—conceptually related to AMIE3's rule expansion tree but more general and database-oriented.
- **Scalable Verification:** MATILDA evaluates rule satisfaction through SQL queries, enabling fast, direct validation at scale—a technique not shared by AMIE3 or Popper.
- **Approximate Dependency Mining:** By permitting the discovery of rules with low—or even no—support or confidence, as determined by configurable thresholds, MATILDA captures statistically meaningful patterns that would otherwise be excluded by more restrictive systems.

This technical comparison positions MATILDA as a system that complements and extends the capabilities of existing rule mining systems through a novel integration of graph-based candidate generation, structural pruning, and scalable SQL-based evaluation.

References

[1] T. Lajus and F. Suchanek. Fast and expressive rule mining with AMIE 3. In *ISWC*, 2020.

[2] A. Cropper and T. Morel. Learning logic programs by discovering useful subprograms. In *IJCAI*, 2020.

[3] J. Bauckmann *et al.* Efficiently detecting inclusion dependencies. *HPI Tech. Report*, 2006.

[4] D. Papenbrock *et al.* SPIDER: Efficient detection of inclusion dependencies. *Metanome Tech. Report*, HPI, 2015.