What and why is querying workflow provenance?

The approach assumes that an execution g of a workflow \( G \) is labeled with each safe subquery \( R_i \). Thus, \( G \) is rewritten so that, using the reachability labels of nodes in g, whether reachability labels using an existing technique. At query time, given g, G and a regular path query \( R \), the approach decomposes \( R \) into a set of subqueries \( R_1, \ldots, R_k \) that are safe. For each safe subquery \( R_i \), \( G \) is rewritten so that, using the reachability labels of nodes in g, whether or not there is a path which matches \( R_i \) between two nodes can be decided in constant time.

The results of each safe subquery are then composed, possibly with some small unsafe remainder, to produce an answer to \( R \).

The approach results in an algorithm that significantly reduces the number of subqueries \( k \) over existing techniques by increasing their size and complexity, and that evaluates each subquery in time bounded by its input and output size. Experimental results demonstrate the benefit of this approach.

**Problem Statement**

Consider a graph \( G \) derived from an edge-labeled context-free graph grammar \( G \), and regular path query \( R \).

1. Pairwise regular path query decide whether there exists a path between two nodes \( u, v \) in g such that the concatenation the edge labels along the path satisfies query \( R \), denoted by \( R(u,v) = \text{true} \).
2. All-pairs regular path query find all node pairs \( (u,v) \) in two given node list \( l_1 \) and \( l_2 \) such that \( R(u,v) = \text{true} \).

**Approach Overview**

**Theoretical Results**

For a subclass of queries, namely \( \text{safe} \) queries, the time complexity is:

1. Pairwise safe regular path queries: \( O(1) \) (assuming \( |G| < |g| \))
2. All-pairs safe regular path queries: \( O(|l_1| \times |l_2| + N) \) where \( N \) is the number of reachable node pairs in \( l_1 \times l_2 \)

**Abstract**

This paper proposes a novel approach for efficiently evaluating regular path queries over provenance graphs of workflows that may include recursion.

**Introduction**

A realistic workflow and its executions/runs/provenance

**Figure 1.** A realistic workflow and its executions/runs/provenance

- After observing erroneous data when the workflow is running, answer questions like
  - "Is the workflow design wrong?" or
  - "Are all the data produced through M1(M2|M3)\(_\ast\)*M49 wrong? (Mi is a module)"

- i.e. answering regular path queries

**Example**

Example: Consider the following provenance graph \( g \) derived from workflow \( G \), and query \( R=\text{[_]*a1(_)*a2\ldots a_k(_)*} \), which favors existing approaches

- Kleene Star: \( R=a^\ast \), which favors our approach

- General queries: combination of the above and edge labels

**Experimental Results**

Data: two realistic workflows BioAID and QBLast

- IFQs: \( R=\text{[_]*a1(_)*a2\ldots a_k(_)*} \), which favors existing approaches
- Kleene Star: \( R=a^\ast \), which favors our approach
- General queries: combination of the above and edge labels

Exp-1: Time overhead (of checking query safety)

- Result: Negligible (<200 ms)

Exp-2: Query time of pairwise safe RPQs

- Result: Constant w.r.t. run size

Exp-3: Query time of all-pairs safe RPQs

- Result:
  1. We outperform baseline when queries are not too selective
  2. We achieve a major gain for Kleene Stars

**Conclusion and Future Work**

Our proposed technique is especially useful for queries that generate a lot of intermediate results and thus could be a very useful component in a cost-based query optimizer that uses statistical information to choose the right query plan and would significantly reduce the evaluation cost of lowly selective subqueries.

Future work includes building a cost-model to predicate intermediate result size and query rewriting when taking workflow specification into account.

**Figure 2a.** Query Time of All-pairs Safe IFQs

**Figure 2b.** Query Time of All-pairs Safe Kleene Stars

**Figure 3.** Query Time of All-pairs General Queries (on BioAID and QBLast)