Scaling Up Symbolic Reasoning for Relational Queries

or, speeding up *debugging & verification of database queries*

**Chenglong Wang**, Alvin Cheung, Ras Bodik
University of Washington

PAUL G. ALLEN SCHOOL
OF COMPUTER SCIENCE & ENGINEERING
• The language between human and relational databases (tables)

```
Select val
From T
Where color = red;
```
Relational Queries

• The language between human and relational databases (tables)

Select val
From T
Where color = red;

T₁ Join T₂ On T₁.color=T₂.color;
Relational Queries

• The language between human and relational databases (tables)

Select (filter & projection)

Join

Group & Aggregation

Select val
From T
Where color = red;

T₁ Join T₂ On T₁.color = T₂.color;

Select color, Sum(val)
From T
Group by color;
1982 “On Optimizing an SQL-like Nested Query” (Kim Won)

**Rewrite rules for nested queries**

$q_1$

Select $R.c_h$
From $R$
Where $R.c_h = (Select \text{Agg}(S.c_m)$
From $S$
Group By $S.cn$);

$q_2$

$S' = (Select S.cn, \text{Agg}(S.c_m)$
From $S$
Group By $S.cn$):

Select $R.c_h$
From $R$
Where $R.cn = (Select \text{Agg}(S'.cn)$
From $S'$
Where $T'.cn = R.cn$);
"The Count Bug"

Rewrite rules for nested queries

$q_1$

Select $R.c_h$
From $R$
Where $R.c_h = (Select \text{Agg}(S.c_m)$
From $S$
Where $S.c_n = S.c_p)$;

$q_2$

$S' = (Select S.c_n, \text{Agg}(S.c_m)$
From $S$
Group By $S.c_n$):

Select $R.c_h$
From $R$
Where $R.c_h = (Select \text{Agg}(S'.c_m)$
From $S'$
Where $T'.c_n = R.c_p)$;

1982 “On Optimizing an SQL-like Nested Query” (Kim Won)

1987 “Optimization of Nested SQL Queries Revisited” (Ganski & Wong)

Found a bug in the 1982 paper!

PARTS
<table>
<thead>
<tr>
<th>PNUM</th>
<th>QUN</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>8</td>
</tr>
</tbody>
</table>

SUPPLY
<table>
<thead>
<tr>
<th>PNUM</th>
<th>QUAN</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>5</td>
</tr>
</tbody>
</table>

PARTS
<table>
<thead>
<tr>
<th>PNUM</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
</tr>
</tbody>
</table>

8-10-81
6-8-78
7-3-79
10-1-78
5-7-83
[ICCE 84-2]
Reasoning Tasks

q, q’ \rightarrow \text{Verification} \quad \text{"Are two queries equivalent on ALL inputs"} \quad \Rightarrow \quad q \equiv q’

q \rightarrow \text{Property Checking (for optimization)} \quad \text{"Can the query return empty output on SOME input?"} \quad \Rightarrow \quad q(T) = \text{empty}

q, q’ \rightarrow \text{Mutation testing / Grading} \quad \text{"Find a distinguishing input between queries."} \quad \Rightarrow \quad q(T) \neq q’(T)
Relational Queries

tens to hundreds of HUGE tables
complex analytical functions

generated by computer

“analyze transition history”

highly optimized

plays important roles in industry

can’t afford 5 years to find a bug!
Check whether \( q_1 \) is equivalent to \( q_2 \) (on ALL inputs)

\[
\begin{align*}
q_1, q_2 & \quad \text{(queries)} \\
\text{assert } q_1 \neq q_2 & \quad \text{(property)} \\
q_1, q_2 & \quad \text{(queries)} \\
\text{assert } q_1 \neq q_2 & \quad \text{(property)}
\end{align*}
\]

unsatisfiable (proved \( q_1 = q_2 \))

found \( T \), \( q_1(T) \neq q_2(T) \)

unsatisfiable (proved \( q_1 = q_2 \))

found \( T \), \( q_1(T) \neq q_2(T) \)
Automatic Reasoning

Check whether $q_1$ is equivalent to $q_2$ (on ALL inputs)

$q_1, q_2$ (queries) => assert $q_1 \neq q_2$ (property)

unsatisfiable (proved $q_1 = q_2$)

found $T$, $q_1(T) \neq q_2(T)$

$q_1, q_2$ (queries) => assert $q_1 \neq q_2$ (property)

unsatisfiable (proved $q_1 = q_2$)

found $T$, $q_1(T) \neq q_2(T)$

(unbounded) equivalence is undecidable
Symbolic Reasoning

Check whether \( q_1 \) is equivalent to \( q_2 \) (on ALL inputs within a search space)

\[ q_1, q_2 \] (queries)

assert \( q_1 \neq q_2 \) (property)

unsatisfiable (proved \( q_1 = q_2 \))

found \( T \), \( q_1(T) \neq q_2(T) \)

\[ q_1, q_2 \] (queries)

\( \text{search space} \)

assert \( q_1 \neq q_2 \) (property)

unsatisfiable (proved \( q_1 = q_2 \))

found \( T \), \( q_1(T) \neq q_2(T) \)

tables with at most \( k \) rows
Symbolic Reasoning

“Check whether $q_1$, $q_2$ are equivalent on ALL tables with at most $k$ tuples”

(1) Target queries

$q_1$: Select id, val From T Where id > 1
$q_2$: Select id, val From T Where id ≠ 1

(2) Search space

tables with at most $k$ rows

(3) Property

$q_1(T) \neq q_2(T)$
Symbolic Reasoning

“Check whether $q_1$, $q_2$ are equivalent on ALL tables with at most $k$ tuples”

(1) Target queries

$q_1$: Select id, val
From T
Where id > 1

$q_2$: Select id, val
From T
Where id ≠ 1

(2) Search space

tables with at most $k$ rows

(3) Property

$q_1(T) \neq q_2(T)$
Symbolic Reasoning

“Check whether $q_1$, $q_2$ are equivalent on ALL inputs within size $k$”

(1) Target queries
$q_1, q_2$

(2) Search space

(3) Property
$T_{out1} \neq T_{out2}$

Grouping & aggregation

“Select $f$ (val) From $T$
Group By id ”
Symbolic Reasoning

“Check whether \( q_1, q_2 \) are equivalent on ALL inputs within size \( k \)”

(1) Target queries
\[ q_1, q_2 \]

(2) Search space
tables with at most \( k \) rows

(3) Property
\[ T_{out1} \neq T_{out2} \]

Grouping & aggregation
“Select \( f(val) \)
From \( T \)
Group By id”

Exponential ways to partition the table
Symbolic Reasoning

“Check whether $q_1$, $q_2$ are equivalent on ALL inputs within size $k$”

(1) Target queries
$q_1$, $q_2$

(2) Search space
tables with at most $k$ rows

(3) Property
$T_{out1} \neq T_{out2}$

- Grouping & aggregation
  “Select $f(val)$
  From $T$
  Group By id”

- Exponential ways to partition the table

- Computationally expensive
$T_{out1} \subset T_{out2} \& T_{out2} \subset T_{out1}$
Symbolic Reasoning

“Check whether $q_1$, $q_2$ are equivalent on ALL inputs within size $k$”

(1) Target queries

$q_1$, $q_2$

(2) Search space

tables with at most $k$ rows

(3) Property

$T_{out1} \neq T_{out2}$

$T_{out1} \subset T_{out2} \& T_{out2} \subset T_{out1}$

Grouping & aggregation

“Select $f(val)$

From $T$

Group By id”

Exponential ways to partition the table

Computationally expensive

$T_{out1} \subset T_{out2} \& T_{out2} \subset T_{out1}$

Solver
Symbolic Reasoning

“Check whether $q_1$, $q_2$ are equivalent on ALL inputs within size $k$”

(1) Target queries
$q_1$, $q_2$

(2) Search space
tables with at most $k$ rows

(3) Property
$T_{out1} \neq T_{out2}$

“Small Model”
A smaller search space to achieve same reasoning guarantee

Grouping & aggregation
“Select $f(val)$
From $T$
Group By id”

Exponential ways to partition the table

Computationally expensive
$T_{out1} \subset T_{out2} \& T_{out2} \subset T_{out1}$
Space Refinement

“Check whether $q_1$, $q_2$ are equivalent on ALL inputs within size $k$”

$T_{out1} \neq T_{out2}$

(queries)

(1) If exists $T \in S$ satisfying the property, we can find one in the $S'$ too.

(2) If none of tables in $S'$ satisfying the property, then no $T$ exists in $S$ too.
Space Refinement

“Check whether $q_1$, $q_2$ are equivalent on ALL inputs within size $k$”

$q_1, q_2$ (queries) $T_{out1} \neq T_{out2}$ (property)

tables with at most $k$ rows $S$ (search space)

provenance analysis

$S'$ (refined search space)

“Small Model”

(1) If exists $T \in S$ satisfying the property, we can find one in the $S'$ too.
(2) If none of tables in $S'$ satisfying the property, then no $T$ exists in $S$ too.
Insight from Property

• Many properties require only one tuple in the output to invalidate.

“Check whether \( q_1, q_2 \) are equivalent”

\[ q_1(T) \neq q_2(T) \]

 Exists a row \( r \) with different multiplicities in \( T_{out1} \) and \( T_{out2} \)

\[ r \in T_{out1}, r \notin T_{out2} \rightarrow q_1(T) \neq q_2(T) \]
Insight from the Property

- Many important properties require only one tuple in the output to be invalidated.

\[ q_1, r \in T \text{ from search space } S \]
\[ r \notin T_{out1}, r \in T_{out2} \]

\[ q_2, r \in T' \]
\[ r \in T_{out1}, r \notin T_{out2} \]

\[ T' \text{ can also distinguish } q_1 \text{ from } q_2! \]
Provenance Analysis

Assume \( r=(a, b) \) is the output tuple showing the difference between two queries

\[ r \in T_{out1}, r \notin T_{out2} \]

\( q_1 \): Select id, \( \max(\text{val}) \)
    From T
    Group By id

\( q_2 \): Select id, \( \min(\text{val}) \)
    From T
    Group By id
Provenance Analysis

Assume \( r=(a, b) \) is the output tuple showing the difference between two queries
\[ r \in T_{out1}, r \notin T_{out2} \]

\( q_1: \) Select id, \( \max(val) \)
From \( T \)
Group By id

\( q_2: \) Select id, \( \min(val) \)
From \( T \)
Group By id

Assume \( r=(a, b) \) is the output tuple showing the difference between two queries
\[ r \in T_{out1}, r \notin T_{out2} \]
Provenance Analysis

Assume \( r = (a, b) \) is the output tuple showing the difference between two queries \( r \in T_{out1}, r \notin T_{out2} \)

\( q_1: \) Select id, max(val)  
From T  
Group By id

\( q_2: \) Select id, min(val)  
From T  
Group By id

\( T' = \{ r \in T | r.id = a \} \)
Provenance Analysis

Assume \( r = (a, b) \) is the output tuple showing the difference between two queries \( r \in T_{out1}, r \notin T_{out2} \)

\( q_1: \) Select id, \( \text{max}(\text{val}) \)
From T
Group By id

\( q_2: \) Select id, \( \text{min}(\text{val}) \)
From T
Group By id

\( T' = \{ r \in T | r.id = a \} \)

\( S' = \{ T \in S | T \text{ contains only one group} \} \)

\( \text{(the group with id "a")} \)
Space Refinement

$S' = \{ T \in S \mid T \text{ contain only one group} \}$

$q_1$: Select $id$, $\max(val)$
From $T$
Group By $id$

$q_2$: Select $id$, $\min(val)$
From $T$
Group By $id$
Space Refinement

\[ S' = \{ T \in S \mid T \text{ contain only one group} \} \]

**q1:** Select id, max(val)
From T
Group By id

**q2:** Select id, min(val)
From T
Group By id

![Diagram showing query results and table refinement process]
Symbolic Provenance Analysis

- Inductively define the analysis rules for different operators
- How to combine provenance from multiple queries

If exists $T \in S$ satisfying the property, we can find one in the $S'$ too.

Analysis complexity: linear to the query size.
Experiment

Bounded Verification

“Verify two queries are equivalent on ALL inputs.”

$q_1(T) \equiv q_2(T)$

*Benchmarks: 46 rules from Apache Calcite*

Test generation

“Can the query return empty output?”

$q(T) = \text{empty}$

“Find a distinguishing input between queries.”

$q_1(T) \neq q_2(T)$

*Benchmarks: 15 student submissions & prior work*
Experiment

Bounded Verification

“Verify two queries are equivalent on ALL inputs.”

\[ q_1(T) \equiv q_2(T) \]

Benchmarks: 46 rules from Apache Calcite

Test generation

“Can the query return empty output?”

\[ q(T) = \text{empty} \]

“Find a distinguishing input between queries.”

\[ q_1(T) \neq q_2(T) \]

Benchmarks: 15 student submissions & prior work

Process

Measure solving speed with and without space refinement

1. Increase search space size until hitting 10 minutes limit without refinement

2. Re-run the same search space with space refinement
Experiment

**Bounded Verification**

“Verify two queries are equivalent on ALL inputs.”

\[ q_1(T) \equiv q_2(T) \]

*Benchmarks: 46 rules from Apache Calcite*

**Test generation**

“Can the query return empty output?”

\[ q(T) = \text{empty} \]

“Find a distinguishing input between queries.”

\[ q_1(T) \neq q_2(T) \]

*Benchmarks: 15 student submissions & prior work*

**Process**

*Measure solving speed with and without space refinement*

1. Increase search space size until hitting 10 minutes limit without refinement

2. Re-run the same search space with space refinement

*Cosette SQL Solver, Qex SQL Solver*
Experiment — Verification

Bounded Verification

“Verify that two queries are equivalent on ALL inputs with no more than k tuples.”

Result

**Cosette** with and without refinement

**Qex** with and without refinement
Bounded Verification

“Verify that two queries are equivalent on ALL inputs with no more than k tuples.”

Result

• Up to 400x speed up & little overhead.
• Speedup is independent from solver implementation!

Cosette with and without refinement

Qex with and without refinement
Experiment — Verification

Bounded Verification

“Verify that two queries are equivalent on ALL inputs with no more than k tuples.”

Result

Benefit from exponential reduction of the number of groups

Provenance is not able to refine search space

```sql
SELECT 1
FROM (SELECT * FROM emp
       WHERE emp.deptno > 7) AS t
INNER JOIN emp AS EMP0
ON t.deptno = EMP0.deptno
INNER JOIN emp AS EMP1
ON EMP0.deptno = EMP1.deptno;
```
**Experiment - Test Generation**

**Test generation**

"Can the query return empty output on SOME input?"

"Find a distinguishing input between queries."

**Result**

![Graph: cosette with and without refinement](image)

![Graph: qex with and without refinement](image)
### Experiment - Test Generation

**Test generation**

“Can the query return empty output on SOME input?”

“Find a distinguishing input between queries.”

**Result**

Reduction of the number of groups

```sql
SELECT Distinct dept_name
FROM course
WHERE credits =
(SELECT MAX(credits)
FROM course NATURAL JOIN department
WHERE title="CS"
GROUP BY dept
HAVING COUNT(DISTINCT course id)>4)
```
Limitations

- Property supported are those can be invalidate by one tuple in the output.

\[ q_1(T) \neq q_2(T) \]
\[ q_1(T) \neq \text{empty} \]
\[ \exists r, \Phi(q(T)) \]

- Improving provenance analysis precision.

"\( q_1(T) \) contains exactly 5 tuples"
"every tuple in \( q(T) \) has same multiplicity"

Generalize to arbitrary property
Summary

Scaling Up Symbolic Reasoning for Relational Queries

(1) Symbolic Reasoning

$q_1, q_2$ (queries)

(search space)

assert $q_1 \neq q_2$ (property)

tables with at most $k$ rows

unsatisfiable (proved $q_1 = q_2$)

found $T$, $q_1(T) \neq q_2(T)$

(2) Scaling Up

$q$ (query)

(search space)

tables with at most $k$ rows

(provenance analysis)

(3) Result

$\phi$ (property)

tables with at most $k$ rows

(1) smaller search space & easier to traverse
(2) equivalent for reasoning

Low analysis overhead & over 100x speed up in
(1) bounded verification
(2) test generation
Hidden Slides!
Symbolic Provenance Analysis

Multiple provenance exists for $t_{out}$ for a query $q$

Select $id$, $\min(val)$
From $T$
Where $val > 0$
Group By $id$

Choice of abstraction trades between the analysis overhead and pruning power