Worst-case Analysis for Interactive Evaluation of Boolean Provenance

Antoine Amarilli, Yael Amsterdamer
Boolean Provenance

Worst-case Analysis for Interactive Evaluation of Boolean Provenance

1. SELECT DISTINCT a.Acquired, e.Institute
2. FROM Acquisitions AS a, Roles AS r, Education AS e
3. WHERE a.Acquired = r.Organization AND
   r.Member = e.Alumni AND a.Date >= 2017.01.01 AND
   r.Role LIKE '%found%' AND e.YEAR <= year(a.Date)
### Boolean Provenance

#### Worst-case Analysis for Interactive Evaluation of Boolean Provenance

<table>
<thead>
<tr>
<th>Acquisitions</th>
<th>Roles</th>
<th>Education</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Acquired</strong></td>
<td><strong>Organization</strong></td>
<td><strong>Role</strong></td>
</tr>
<tr>
<td>A2Bdone</td>
<td>A2Bdone</td>
<td>Founder</td>
</tr>
<tr>
<td>microBarg</td>
<td>A2Bdone</td>
<td>Founding member</td>
</tr>
<tr>
<td>fPharm</td>
<td>A2Bdone</td>
<td>Founding member</td>
</tr>
<tr>
<td>Optobest</td>
<td>microBarg</td>
<td>Co-founder</td>
</tr>
<tr>
<td></td>
<td>microBarg</td>
<td>Co-founder</td>
</tr>
<tr>
<td></td>
<td>microBarg</td>
<td>CTO</td>
</tr>
</tbody>
</table>

**Input database**

\[ a_0 \land r_0 \land e_0 \]

**Output relation**

\[ (a_0 \land r_0 \land e_0) \lor (a_0 \land r_1 \land e_1) \lor (a_0 \land r_2 \land e_3) \]

1. SELECT DISTINCT a.Acquired, e.Institute
2. FROM Acquisitions AS a, Roles AS r, Education AS e
3. WHERE a.Acquired = r.Organization AND r.Member = e.Alumni AND a.Date >= 2017.01.01 AND r.Role LIKE '%found%' AND e.YEAR <= year(a.Date)
### Boolean Provenance

#### Worst-case Analysis for Interactive Evaluation of Boolean Provenance

<table>
<thead>
<tr>
<th>Acquisitions</th>
<th>Roles</th>
<th>Education</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acquired</td>
<td>Organization</td>
<td>Role</td>
</tr>
<tr>
<td>A2Bdone</td>
<td>A2Bdone</td>
<td>Founder</td>
</tr>
<tr>
<td>microBarg</td>
<td>A2Bdone</td>
<td>Founding member</td>
</tr>
<tr>
<td>fPharm</td>
<td>A2Bdone</td>
<td>Founding member</td>
</tr>
<tr>
<td>Optobest</td>
<td>microBarg</td>
<td>Co-founder</td>
</tr>
<tr>
<td></td>
<td>microBarg</td>
<td>Co-founder</td>
</tr>
<tr>
<td></td>
<td>microBarg</td>
<td>CTO</td>
</tr>
</tbody>
</table>

1. SELECT DISTINCT a.Acquired, e.Institute
2. FROM Acquisitions AS a, Roles AS r, Education AS e
3. WHERE a.Acquired = r.Organization AND
   r.Member = e.Alumni AND a.Date >= 2017.01.01 AND
   r.Role LIKE '%found%' AND e.YEAR <= year(a.Date)

\[(a_0 \land r_0 \land e_0) \lor (a_0 \land r_1 \land e_1)\]

- **Input database**
- **Output relation**

<table>
<thead>
<tr>
<th>Acquired</th>
<th>Institute</th>
<th>Relation</th>
</tr>
</thead>
<tbody>
<tr>
<td>A2Bdone</td>
<td>U. Melbourne</td>
<td>((a_0 \land r_0 \land e_0) \lor (a_0 \land r_1 \land e_1) \lor (a_0 \land r_2 \land e_3))</td>
</tr>
<tr>
<td>A2Bdone</td>
<td>U. Sau Paolo</td>
<td>((a_0 \land r_2 \land e_2))</td>
</tr>
<tr>
<td>microBarg</td>
<td>U. Melbourne</td>
<td>((a_1 \land r_2 \land e_3))</td>
</tr>
<tr>
<td>microBarg</td>
<td>U. Sau Paolo</td>
<td>((a_1 \land r_3 \land e_2) \lor (a_1 \land r_4 \land e_4))</td>
</tr>
</tbody>
</table>
For any truth valuation $val$: an output tuple $t$ evaluates to true iff it appears in the possible world of $val$

$$(a_0 \land r_0 \land e_0) \lor (a_0 \land r_1 \land e_1) \lor (a_0 \land r_2 \land e_3) = \text{False}$$
Boolean Provenance: Uses

- Deletion propagation
- Access control
- Probabilistic databases
- Consent Management
Data owners are probed on a need basis for fine-grained consent – per tuple

*Managing Consent for Data Access in Shared Databases [ICDE 2021, Drien, Amarilli, A.]*
• We can use the output iff we can derive it from input tuples with consent

• We can choose **which** variables truth values to probe

• Effectiveness depends on the answer and Boolean expressions structure
Example Evaluation

\[(a_0 \land r_0 \land e_0) \lor (a_0 \land r_1 \land e_1) \lor (a_0 \land r_2 \land e_3)\]
\[(a_0 \land r_2 \land e_2)\]
\[(a_1 \land r_3 \land e_3)\]
\[(a_1 \land r_3 \land e_2) \lor (a_1 \land r_4 \land e_4)\]

\[a_0?\]
False
False
(a_1 \land r_3 \land e_3)
(a_1 \land r_3 \land e_2) \lor (a_1 \land r_4 \land e_4)

\[a_1?\]
True

No need to ask about r_0, e_0, r_1, e_1

\[r_3?\]
True

\[e_3?\]
True

We can use an output tuple iff we can derive it from input tuples with consent
Optimizing the Worst-case Evaluation

- We are interested in a “cautious” probing strategy that minimizes the number of probed variables for any valuation

\[(w \land x) \lor (x \land y) \lor (y \land z)\]

Boolean Decision Diagram (BDD)
Three Problem Definitions (Intuitive)

Input: a set of Boolean provenance expressions

- **OPT-BDD-DEPTH**: minimize the worst-case number of probes
  - (there is always a trivial strategy that queries all variables in order)
- **DEC-BDD-DEPTH**: decide whether there exists a strategy making at most $k$ probes
- **DEC-BDD-EVASIVE**: decide whether the expressions are evasive = no strategy is better than the trivial one (making less than $n$ probes over $n$ variables)

Used in Boolean Function Learning
Previous Work

- **Expected depth optimization by testing variables of Boolean formulas**
  - Interactive Boolean Evaluation, Sequential System Testing, Active Learning, Consent management

- **Worst-case BDD Analysis**
  - Graph/ String properties
  - Construction based on input-output pairs
  - Deciding among Boolean functions

- **Other metrics**
Today

1. Model
2. General Provenance Expressions
3. Read-Once Expressions
4. Monotone Expressions
BDDs for Expression Sets

Worst-case Analysis for Interactive Evaluation of Boolean Provenance

\[
\Phi = (a_0 \land r_0 \land e_0) \lor (a_0 \land r_1 \land e_1) \lor (a_0 \land r_2 \land e_3) \\
(\land (a_0 \land r_2 \land e_2) \\
(\land (a_1 \land r_3 \land e_3) \\
(\land (a_1 \land r_3 \land e_2) \lor (a_1 \land r_4 \land e_4)
\]

\[
\varphi_0: x \land \neg x \\
\varphi_1: \text{False} \\
\varphi_2: y \lor \neg y
\]

\[
\varphi_0 \mapsto \text{False} \\
\varphi_1 \mapsto \text{False} \\
\varphi_2 \mapsto \text{True}
\]
• **BDD Depth**: maximal path length from the root to a leaf

• **Expression Set Depth**: minimal BDD depth

• Constant expression set $\iff$ depth $= 0$
• **Proposition:** DEC-BDD-DEPTH is coNP-hard, even if the input Boolean expression is in DNF/CNF and the depth upper bound is $k = 0$.

• Proof: by reduction from CNF satisfiability / DNF falsifiability. A non satisfiable CNF $\Rightarrow$ constant False $\Rightarrow$ depth 0

$\quad x \land \neg x$

false
Read-Once Provenance

\[ \Phi: (a_0 \land r_0 \land e_0) \lor (a_0 \land r_1 \land e_1) \lor (a_0 \land r_2 \land e_3) \]

Not read-once: variables repeat within/across expressions

Read once: no variable repetitions (in equivalent)

\[ \Phi: (a_0 \land r_0 \land e_0) \lor (a_0 \land r_1 \land e_1) \lor (a_0 \land r_2 \land e_3) \lor (a_1 \land r_3 \land e_2) \lor (a_1 \land r_4 \land e_4) \]

Previous work: query classes yielding read-once provenance or compiling provenance to read-once form.
E.g., SP queries
• **Proposition:** Sets of read-once of Boolean expressions (without constants), and their equivalents, are **evasive**.

• Proof: by induction

• This result does not hold if variables repeat across expressions

\[ \Phi = \{x \land y, x \lor z\} \]

```
  x
 /\    /
/  \  /  \\
true false
```

\[
\begin{array}{c}
\{\text{true, true}\} \\
\{\text{false, true}\} \\
\{\text{false, true}\} \\
\{\text{false, false}\}
\end{array}
\]
• Monotone $k$-DNF expressions: no negation, every term (conjunction) contains up to $k$ unique variables

• **In the paper:** we show a **2-way correspondence** between $k$-DNF expressions and SPJU queries

• **Question:** monotone expressions are satisfiable and falsifiable. What is the minimal depth for monotone Boolean expressions?
• **Lower bound on depth:** maximal term in DNF/clause in CNF
  • Each can be a minimal 0/1 certificate

• **Theorem:** for arbitrarily large n there exists a monotone Boolean expression with a BDD of depth **linear** in this bound
  • Term/clause size is $O(\log n)$ - exponentially smaller than “trivial” solution.
  • The BDD is optimal in this case

\[
(\psi_{i-1} \land u_i) \lor (u_i \land v_i) \lor (v_i \land \psi_{i-1}')
\]
Proof Sketch

- Recursively define: \( \psi_i = (\psi_{i-1} \land u_i) \lor (u_i \land v_i) \lor (v_i \land \psi'_{i-1}) \) where \( u_i, v_i \) are fresh variables and \( \psi'_{i-1} \) is a copy of \( \psi_{i-1} \) using fresh variables.

Let \( \psi_0 = (w_0 \land x_0) \lor (x_0 \land y_0) \lor (x_0 \land y_0) \)

- **Observation:** \( \psi_i \) cannot be evaluated without probing at least one of \( u_i, v_i \)
  - If \( u_i = v_i \) we’re done by probing both
  - Otherwise, we need to evaluate either \( \psi_{i-1} \) or \( \psi'_{i-1} \) but not both

- **Observation:** \( \psi_i \) includes \( 2^i \) copies of \( \psi_0 \) and \( n = \Theta(2^i) \) variables

- “Bad” algorithm: evaluate all copies of \( \psi_0 \) first. Each copy requires 2-4 probes.

- “Good” algorithm: evaluate \( u_i, v_i \) first, then if needed proceed to one of the \( \psi_{i-1} \) and continue recursively. We query at most \( 2i + 3 = O(\log n) \)
• When each term is of size 2, terms can be viewed as edges
• When the resulting graph is acyclic, we have the following

• **Theorem:** Given a monotone acyclic graph DNF, DEC-BDD-EVASIVE is in PTIME.
• **Proof:** We define an **non-evasiveness pattern**, which exists iff the provenance is not evasive

\[(w \land x) \lor (x \land y) \lor (y \land z)\]
Proof Sketch

Isolated vertex = non-evasive

Evasive (e.g., if all are true)

Probe every $y_i$. If all are false – no need to probe $x$.
Assume w.l.o.g $y_0$ is true.

$z_0 \land \text{True} = z_0$ absorbs $z_0 \land w_0$

$w_0$ is the new root. By recursive argument – it is non-evasive!

The other direction is by induction on the tree structure, showing having no pattern entails that any probe and any answer yields remaining sub-graphs without our pattern
Conclusion and Future Work

• Overview
  • Optimizing the BDD depth for deciding the truth value of Boolean provenance expressions
  • Results for different classes of queries and provenance shapes
  • Many open questions

• Further application domains, further query classes
Thank you!

Questions/remarks?