Worst-case Analysis for Interactive Evaluation of Boolean Provenance

Antoine Amarilli, Yael Amsterdamer

TELECOM Paris
IP PARIS
LTCI Laboratoire Traitement et Communication de l'Information

Department of Computer Science
Faculty of Exact Sciences
Bar-Ilan University

Boolean Provenance

Acquisitions

Acquired	Acquiring	Date	
A2Bdone	Zazzer	7/11/2020	a_0
microBarg	Fiffer	1/5/2017	a_1
fPharm	Fiffer	1/2/2016	a_2
Optobest	microBarg	8/8/2015	a_3

Roles

Organization	Role	Member	
A2Bdone	Founder	Usha Koirala	r_0
A2Bdone	Founding member	Pavel Lebedev	r_1
A2Bdone	Founding member	Nana Alvi	r_2
microBarg	Co-founder	Nana Alvi	r_3
microBarg	Co-founder	Gao Yawen	r_4
microBarg	CTO	Amaal Kader	r_5

Education

Alumni	Institute	Year	
Usha Koirala	U. Melbourne	2017	e_0
Pavel Lebedev	U. Melbourne	2017	e_1
Nana Alvi	U. Sau Paolo	2010	e_2
Nana Alvi	U. Melbourne	2017	e_3
Gao Yawen	U. Sau Paolo	2010	e_4
Amaal Kader	U. Cape Town	2005	e_5

Input database

```
SELECT DISTINCT a.Acquired, e.Institute
FROM   Acquisitions AS a, Roles AS r, Education AS e
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)
```

Output relation

Acquired	Institute	
A2Bdone	U. Melbourne	$(a_0 \wedge r_0 \wedge e_0) \vee (a_0 \wedge r_1 \wedge e_1) \vee (a_0 \wedge r_2 \wedge e_3)$
A2Bdone	U. Sau Paolo	$(a_0 \wedge r_2 \wedge e_2)$
microBarg	U. Melbourne	$(a_1 \wedge r_3 \wedge e_3)$
microBarg	U. Sau Paolo	$(a_1 \wedge r_3 \wedge e_2) \vee (a_1 \wedge r_4 \wedge e_4)$

Boolean Provenance

Acquisitions

Acquired	Acquiring	Date	
A2Bdone	Zazzer	7/11/2020	a_0
microBarg	Fiffer	1/5/2017	a_1
fPharm	Fiffer	1/2/2016	a_2
Optobest	microBarg	8/8/2015	a_3

Roles

Organization	Role	Member	
A2Bdone	Founder	Usha Koirala	r_0
A2Bdone	Founding member	Pavel Lebedev	r_1
A2Bdone	Founding member	Nana Alvi	r_2
microBarg	Co-founder	Nana Alvi	r_3
microBarg	Co-founder	Gao Yawen	r_4
microBarg	CTO	Amaal Kader	r_5

Education

Alumni	Institute	Year	
Usha Koirala	U. Melbourne	2017	e_0
Pavel Lebedev	U. Melbourne	2017	e_1
Nana Alvi	U. Sau Paolo	2010	e_2
Nana Alvi	U. Melbourne	2017	e_3
Gao Yawen	U. Sau Paolo	2010	e_4
Amaal Kader	U. Cape Town	2005	e_5

Input database

```
SELECT DISTINCT a.Acquired, e.Institute
FROM   Acquisitions AS a, Roles AS r, Education AS e
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)
```

$$a_0 \wedge r_0 \wedge e_0$$

Output relation

Acquired	Institute	
A2Bdone	U. Melbourne	$(a_0 \wedge r_0 \wedge e_0) \vee (a_0 \wedge r_1 \wedge e_1) \vee (a_0 \wedge r_2 \wedge e_3)$
A2Bdone	U. Sau Paolo	$(a_0 \wedge r_2 \wedge e_2)$
microBarg	U. Melbourne	$(a_1 \wedge r_3 \wedge e_3)$
microBarg	U. Sau Paolo	$(a_1 \wedge r_3 \wedge e_2) \vee (a_1 \wedge r_4 \wedge e_4)$

Boolean Provenance

Acquisitions

Acquired	Acquiring	Date	
A2Bdone	Zazzer	7/11/2020	a_0
microBarg	Fiffer	1/5/2017	a_1
fPharm	Fiffer	1/2/2016	a_2
Optobest	microBarg	8/8/2015	a_3

Roles

Organization	Role	Member	
A2Bdone	Founder	Usha Koirala	r_0
A2Bdone	Founding member	Pavel Lebedev	r_1
A2Bdone	Founding member	Nana Alvi	r_2
microBarg	Co-founder	Nana Alvi	r_3
microBarg	Co-founder	Gao Yawen	r_4
microBarg	CTO	Amaal Kader	r_5

Education

Alumni	Institute	Year	
Usha Koirala	U. Melbourne	2017	e_0
Pavel Lebedev	U. Melbourne	2017	e_1
Nana Alvi	U. Sau Paolo	2010	e_2
Nana Alvi	U. Melbourne	2017	e_3
Gao Yawen	U. Sau Paolo	2010	e_4
Amaal Kader	U. Cape Town	2005	e_5

Input database

```
SELECT DISTINCT a.Acquired, e.Institute
FROM   Acquisitions AS a, Roles AS r, Education AS e
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)
```

$$(a_0 \wedge r_0 \wedge e_0) \vee (a_0 \wedge r_1 \wedge e_1)$$

relation

Acquired	Institute	
A2Bdone	U. Melbourne	$(a_0 \wedge r_0 \wedge e_0) \vee (a_0 \wedge r_1 \wedge e_1) \vee (a_0 \wedge r_2 \wedge e_3)$
A2Bdone	U. Sau Paolo	$(a_0 \wedge r_2 \wedge e_2)$
microBarg	U. Melbourne	$(a_1 \wedge r_3 \wedge e_3)$
microBarg	U. Sau Paolo	$(a_1 \wedge r_3 \wedge e_2) \vee (a_1 \wedge r_4 \wedge e_4)$

Boolean Provenance: Possible Worlds

a_0= False, others=True

Acquisitions

Acquired	Acquiring	Date	
~~A2Bdone~~	~~Zazzer~~	~~7/11/2020~~	a_0
microBarg	Fiffer	1/5/2017	a_1
fPharm	Fiffer	1/2/2016	a_2
Optobest	microBarg	8/8/2015	a_3

A2Bdone	Founder	Usha Koirala	r_0
A2Bdone	Founding member	Pavel Lebedev	r_1
A2Bdone	Founding member	Nana Alvi	r_2
microBarg	Co-founder	Nana Alvi	r_3
microBarg	Co-founder	Gao Yawen	r_4
microBarg	CTO	Amaal Kader	r_5

Education

Alumni	Institute	Year	
Usha Koirala	U. Melbourne	2017	e_0
Pavel Lebedev	U. Melbourne	2017	e_1
Nana Alvi	U. Sau Paolo	2010	e_2
Nana Alvi	U. Melbourne	2017	e_3
Gao Yawen	U. Sau Paolo	2010	e_4
Amaal Kader	U. Cape Town	2005	e_5

Input database

For any truth valuation val:
an output tuple t evaluates to true iff it appears in the possible world of val

$(a_0 \wedge r_0 \wedge e_0) \vee (a_0 \wedge r_1 \wedge e_1) \vee (a_0 \wedge r_2 \wedge e_3)$=False

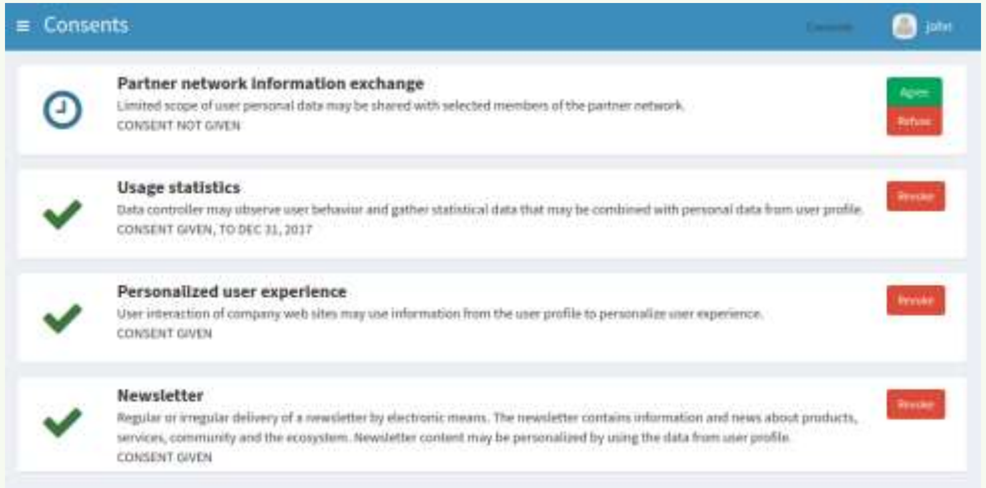
Output relation

Acquired	Institute	
~~A2Bdone~~	~~U. Melbourne~~	$(a_0 \wedge r_0 \wedge e_0) \vee (a_0 \wedge r_1 \wedge e_1) \vee (a_0 \wedge r_2 \wedge e_3)$
~~A2Bdone~~	~~U. Sau Paolo~~	$(a_0 \wedge r_2 \wedge e_2)$
microBarg	U. Melbourne	$(a_1 \wedge r_3 \wedge e_3)$
microBarg	U. Sau Paolo	$(a_1 \wedge r_3 \wedge e_2) \vee (a_1 \wedge r_4 \wedge e_4)$

Boolean Provenance: Uses

- Deletion propagation
- Probabilistic databases
- Access control
- Consent Management

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.]

Consent Management

- 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

e_5

Acquired	Institute	
A2Bdone	U. Melbourne	$(a_0 \wedge r_0 \wedge e_0) \vee (a_0 \wedge r_1 \wedge e_1) \vee (a_0 \wedge r_2 \wedge e_3)$
A2Bdone	U. Sau Paolo	$(a_0 \wedge r_2 \wedge e_2)$
microBarg	U. Melbourne	$(a_1 \wedge r_3 \wedge e_3)$
microBarg	U. Sau Paolo	$(a_1 \wedge r_3 \wedge e_2) \vee (a_1 \wedge r_4 \wedge e_4)$

Example Evaluation

$(a_0 \wedge r_0 \wedge e_0) \vee (a_0 \wedge r_1 \wedge e_1) \vee (a_0 \wedge r_2 \wedge e_3)$

$(a_0 \wedge r_2 \wedge e_2)$

$(a_1 \wedge r_3 \wedge e_3)$

$(a_1 \wedge r_3 \wedge e_2) \vee (a_1 \wedge r_4 \wedge e_4)$

a_0?

False

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

False

False

$(a_1 \wedge r_3 \wedge e_3)$

$(a_1 \wedge r_3 \wedge e_2) \vee (a_1 \wedge r_4 \wedge e_4)$

a_1?

True

False

False

$(r_3 \wedge e_3)$

$(r_3 \wedge e_2) \vee (r_4 \wedge e_4)$

r_3?

True

False

False

e_3

$e_2 \vee (r_4 \wedge e_4)$

e_3?

True

…

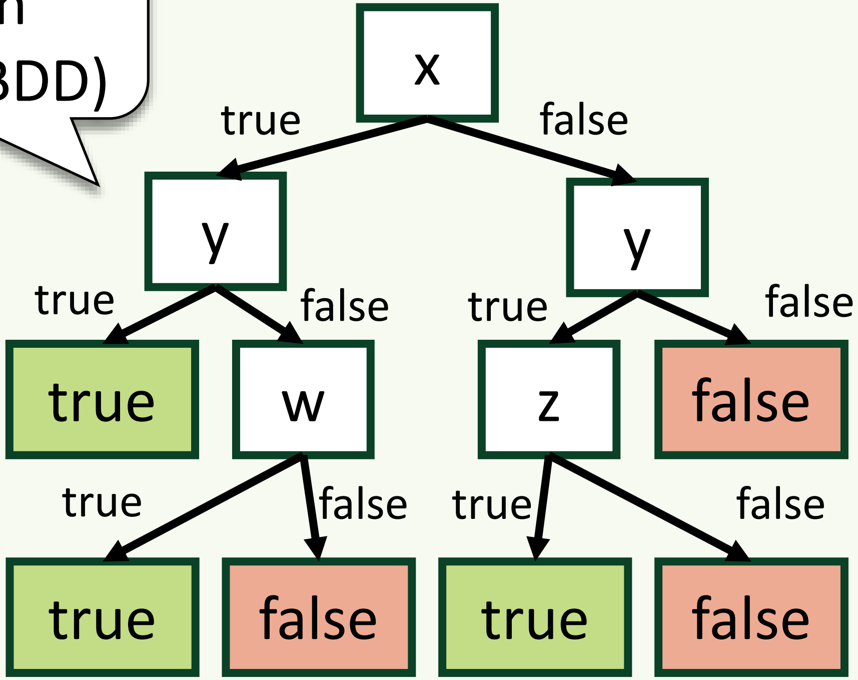
We can use an output tuple iff we can derive it from input tuples with consent

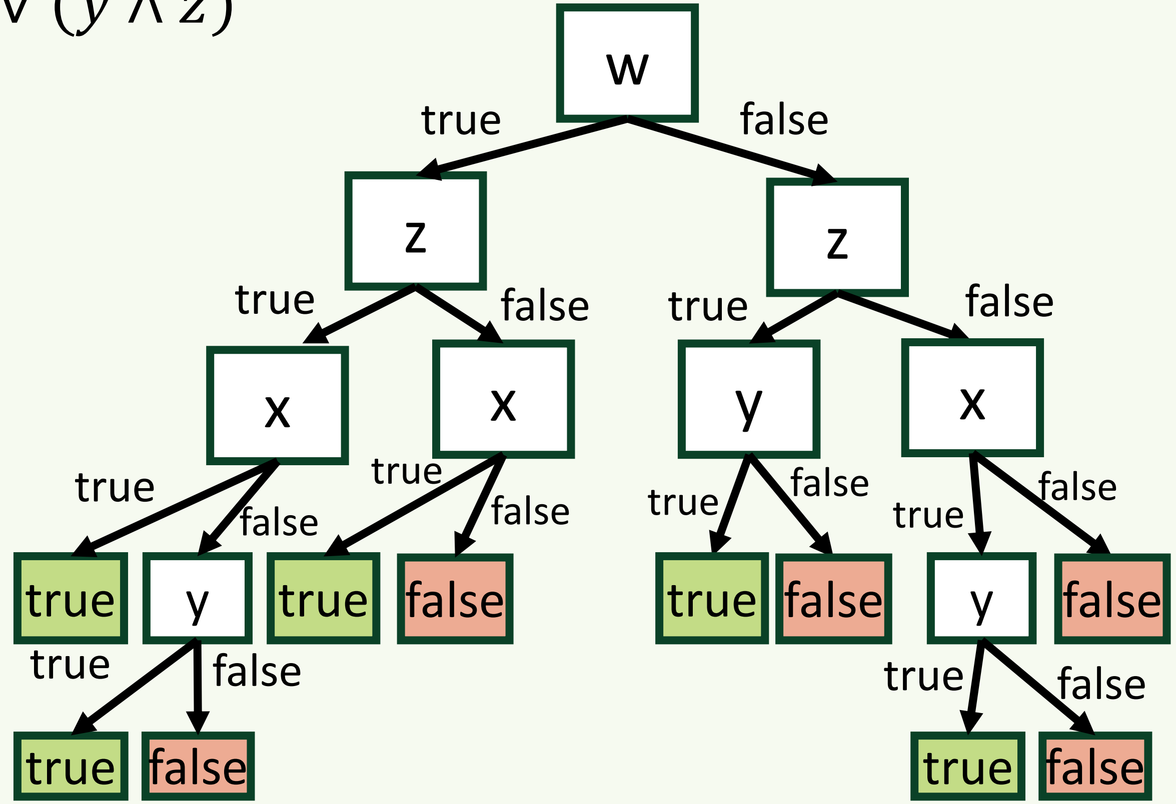
Optimizing the Worst-case Evaluation

- 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)



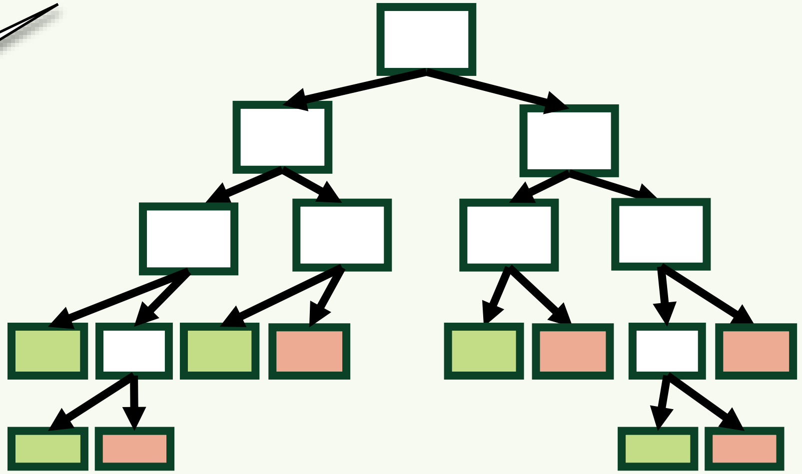


Three Problem Definitions (Intuitive)

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

$x \wedge \neg x$

false

φ_0: $x \wedge \neg x \wedge y$

φ_1: False

φ_2: $y \vee \neg y$

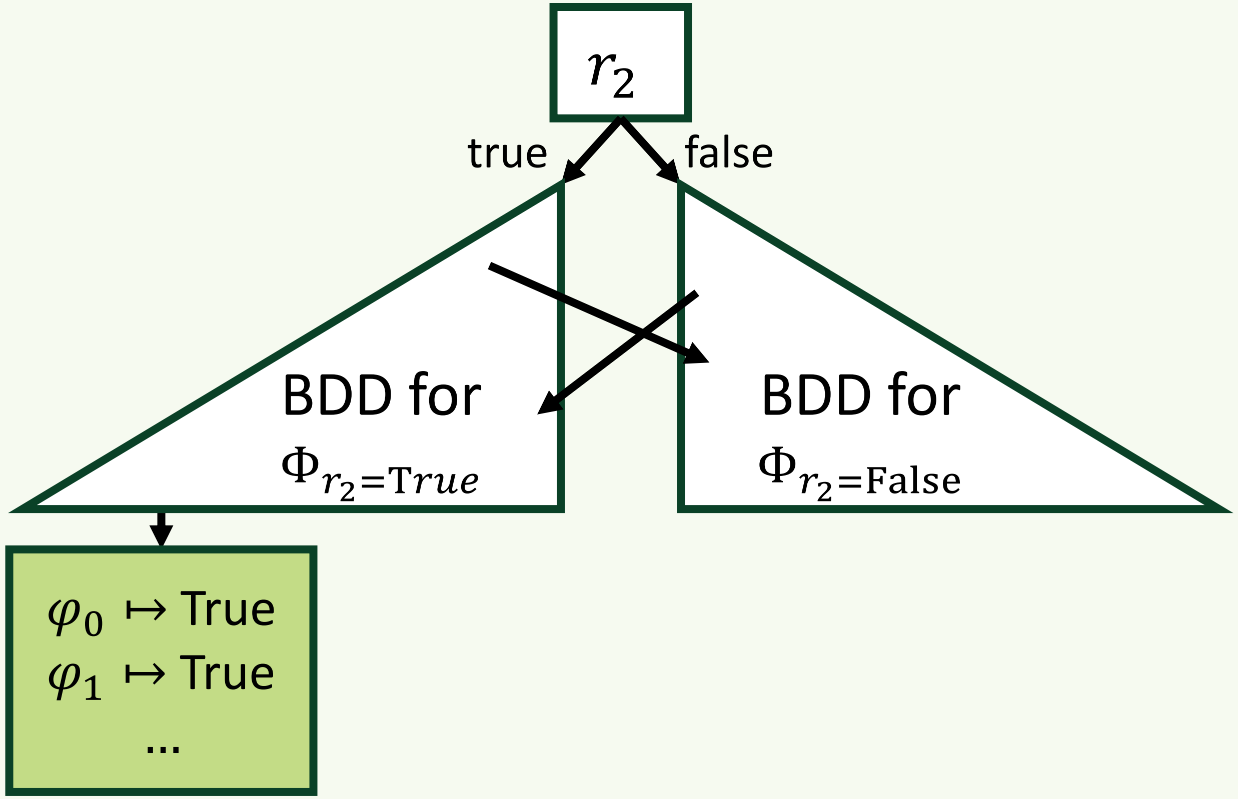
$\varphi_0 \mapsto$ False
$\varphi_1 \mapsto$ False
$\varphi_2 \mapsto$ True

Φ: $(a_0 \wedge r_0 \wedge e_0) \vee (a_0 \wedge r_1 \wedge e_1) \vee (a_0 \wedge r_2 \wedge e_3)$

$(a_0 \wedge r_2 \wedge e_2)$

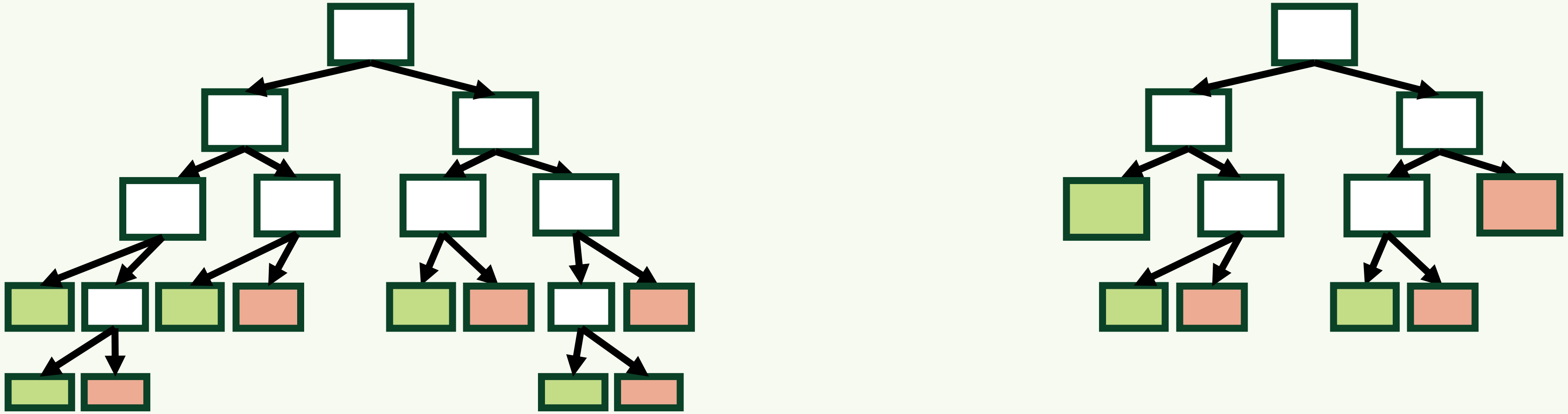
$(a_1 \wedge r_3 \wedge e_3)$

$(a_1 \wedge r_3 \wedge e_2) \vee (a_1 \wedge r_4 \wedge e_4)$



Depth

- **BDD Depth:** maximal path length from the root to a leaf
- **Expression Set Depth**: minimal BDD depth
  - Constant expression set ⇔ depth = 0



General Provenance Expressions

- **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 ⇒ constant False ⇒ depth 0

$x \land \neg x$

false

Read-Once Provenance

Φ: $(\underline{a_0} \wedge r_0 \wedge e_0) \vee (\underline{a_0} \wedge r_1 \wedge e_1) \vee (\underline{a_0} \wedge r_2 \wedge e_3)$

$(\underline{a_0} \wedge r_2 \wedge e_2)$

$(a_1 \wedge r_3 \wedge e_3)$

$(a_1 \wedge r_3 \wedge e_2) \vee (a_1 \wedge r_4 \wedge e_4)$

Not read-once: variables repeat within/across expressions

Previous work: query classes yielding read-once provenance or compiling provenance to read-once form.
E.g., SP queries

Φ: $(a_0 \wedge r_0 \wedge e_0) \vee (a_0 \wedge r_1 \wedge e_1) \vee (a_0 \wedge r_2 \wedge e_3)$

$(a_1 \wedge r_3 \wedge e_2) \vee (a_1 \wedge r_4 \wedge e_4)$

Read once: no variable repetitions (in equivalent)

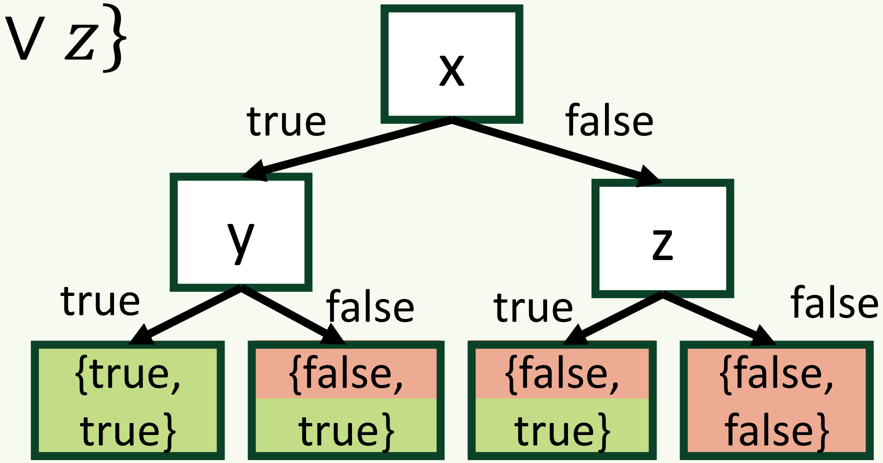
Φ: $\underline{a_0 \wedge} \big((r_0 \wedge e_0) \vee (r_1 \wedge e_1) \vee (r_2 \wedge e_3)\big)$

$\underline{a_1 \wedge} \big((r_3 \wedge e_2) \vee (r_4 \wedge e_4)\big)$

Analysis for Read-Once

- **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

$$\Phi = \{x \land y, x \lor z\}$$



Monotone Provenance

- 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?

Bounds for Monotone Provenance

- **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

$$(\psi_{i-1} \wedge u_i) \vee (u_i \wedge v_i) \vee (v_i \wedge \psi'_{i-1})$$

4 Proof Sketch

Monotone

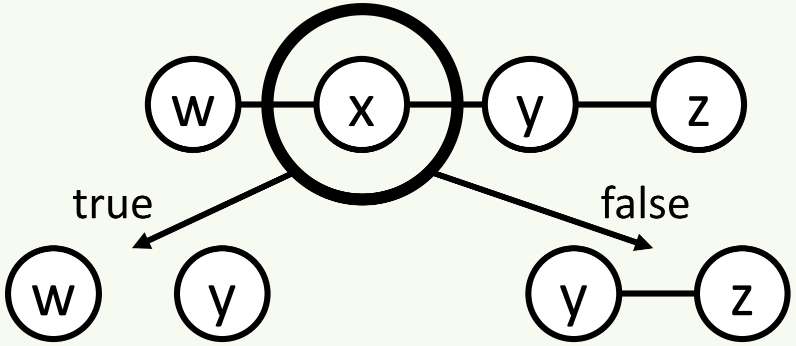
- Recursively define: $\psi_i = (\psi_{i-1} \wedge u_i) \vee (u_i \wedge v_i) \vee (v_i \wedge \psi'_{i-1})$ where u_i, v_i are fresh variables and ψ'_{i-1} is a copy of ψ_{i-1} using fresh variables.
  Let $\psi_0 = (w_0 \wedge x_0) \vee (x_0 \wedge y_0) \vee (x_0 \wedge y_0)$
- **Observation:** ψ_i 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 ψ_{i-1} or ψ'_{i-1} but not both
- **Observation:** ψ_i includes 2^i copies of ψ_0 and $n = \Theta(2^i)$ variables
- "Bad" algorithm: evaluate all copies of ψ_0 first. Each copy requires 2-4 probes.
- "Good" algorithm: evaluate u_i, v_i first, then if needed proceed to one of the ψ_{i-1} and continue recursively. We query at most $2i + 3 = O(\log n)$

Monotone Acyclic Graph DNF

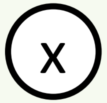
- When each term is of size 2, terms can be viewed as edges
- When the resulting graph is acyclic, we have the following

- **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

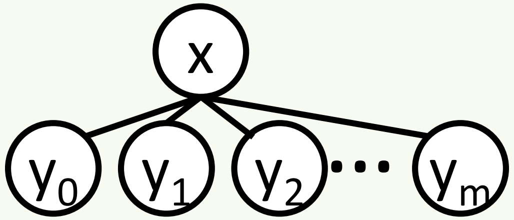
$(w \wedge x) \vee (x \wedge y) \vee (y \wedge z)$



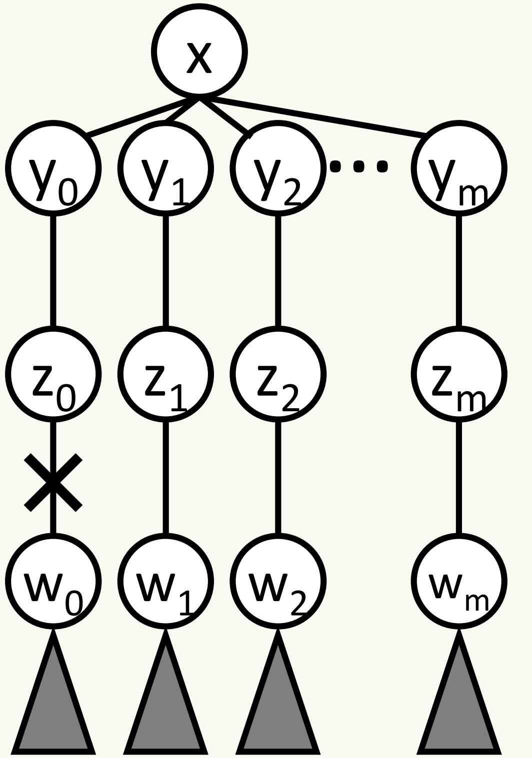
Proof Sketch



Isolated vertex
= non-evasive



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



Each child has grand-child with non-evasiveness pattern
= non-evasive

Probe every y_{i}.
If all are false – no need to probe x.
Assume w.l.o.g y_0 is true.

$z_0 \wedge \mathrm{True} = z_0$ absorbs $z_0 \wedge w_0$

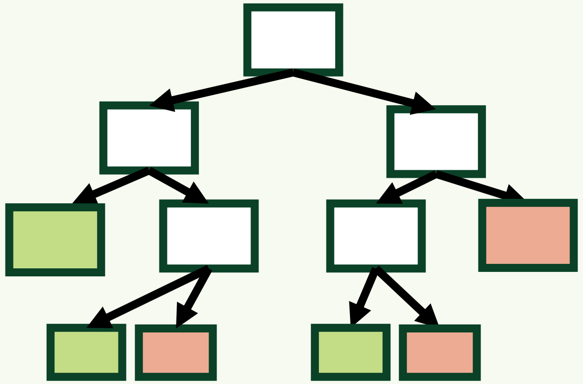
w_0 is the new root.
By recursive argument – it is non-evasive!

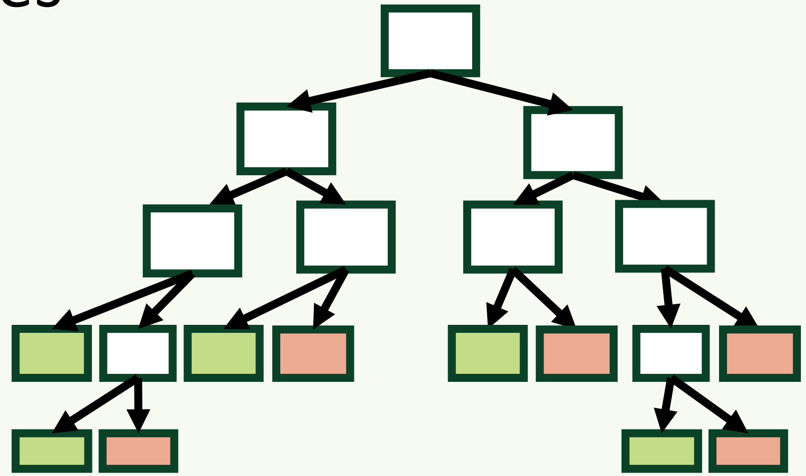
The other direction is by induction on the tree structure, showing having no pattern entails that any probe and any answer yields remaining sub-graphs without our pattern

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?