Northeastern University
Khoury College of Computer Sciences

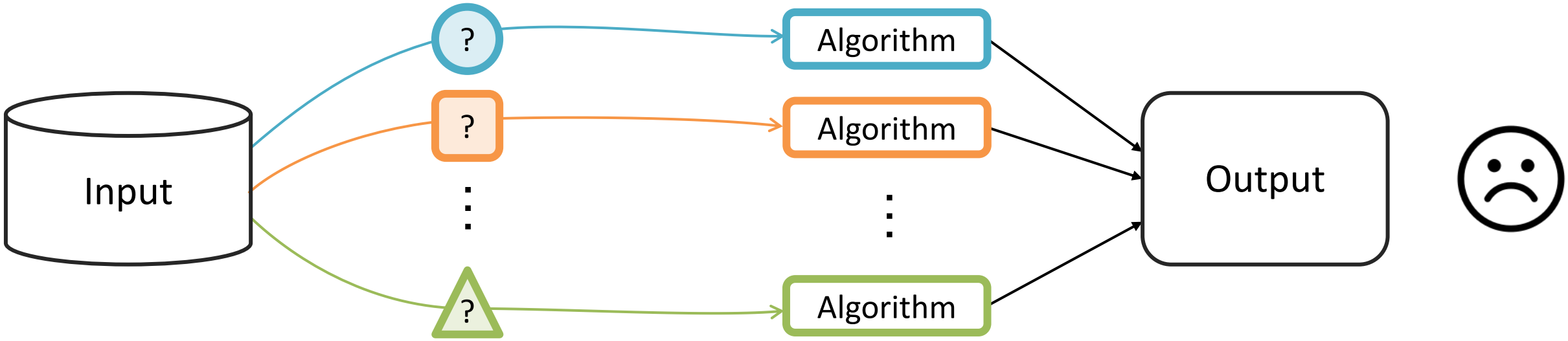
DATALAB

# A Unified Approach for Resilience and Causal Responsibility with Integer Linear Programming (ILP) and LP Relaxations

Neha Makhija
Northeastern University
(Joint work with Wolfgang Gatterbauer)
SIGMOD 2024, Santiago

https://northeastern-datalab.github.io/unified-reverse-data-management/

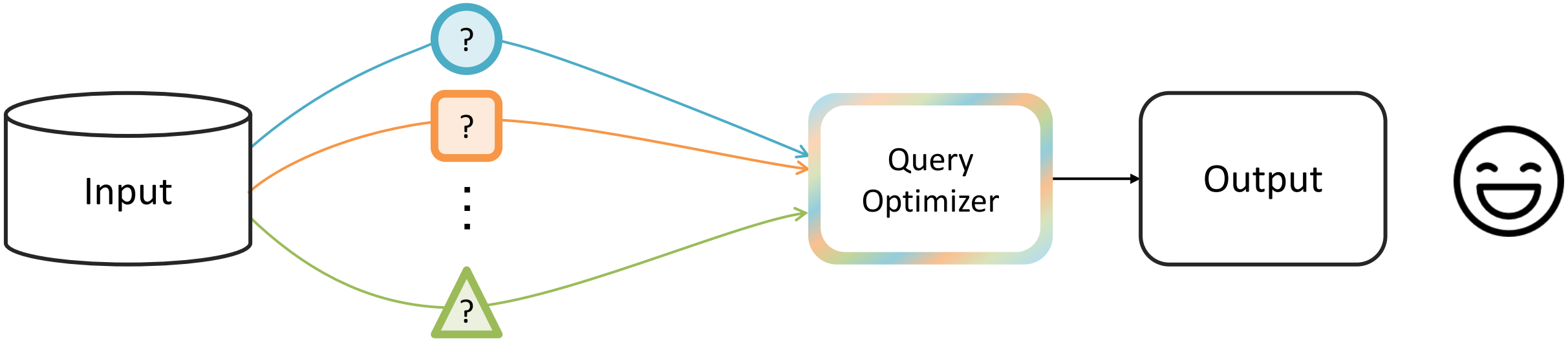
# Traditional Data Management (before query optimizers)




Many types of
user questions!

Different questions
→ Different Optimal Algorithms

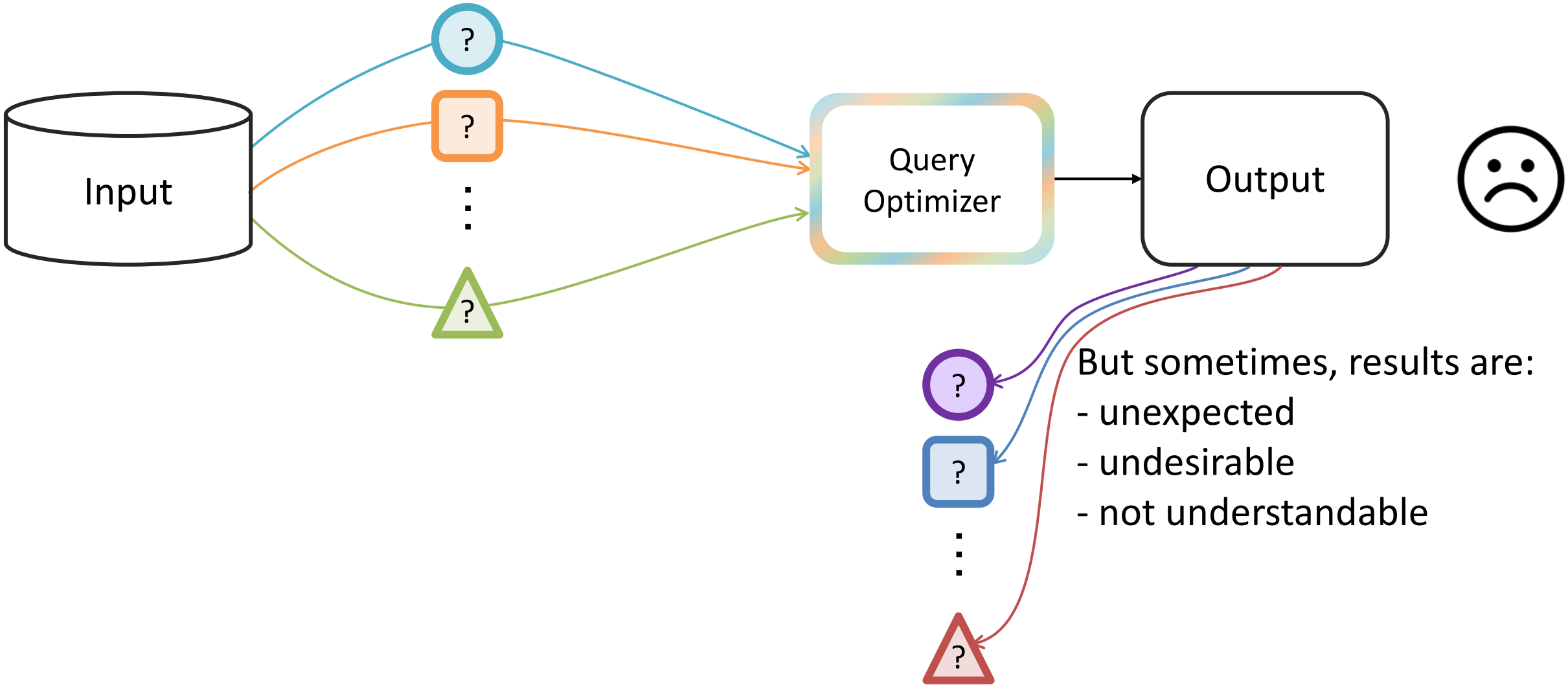
# Traditional Data Management




Many types of user questions!

Different questions
→ Query Optimizer solves everything optimally
→ Choose between different access paths

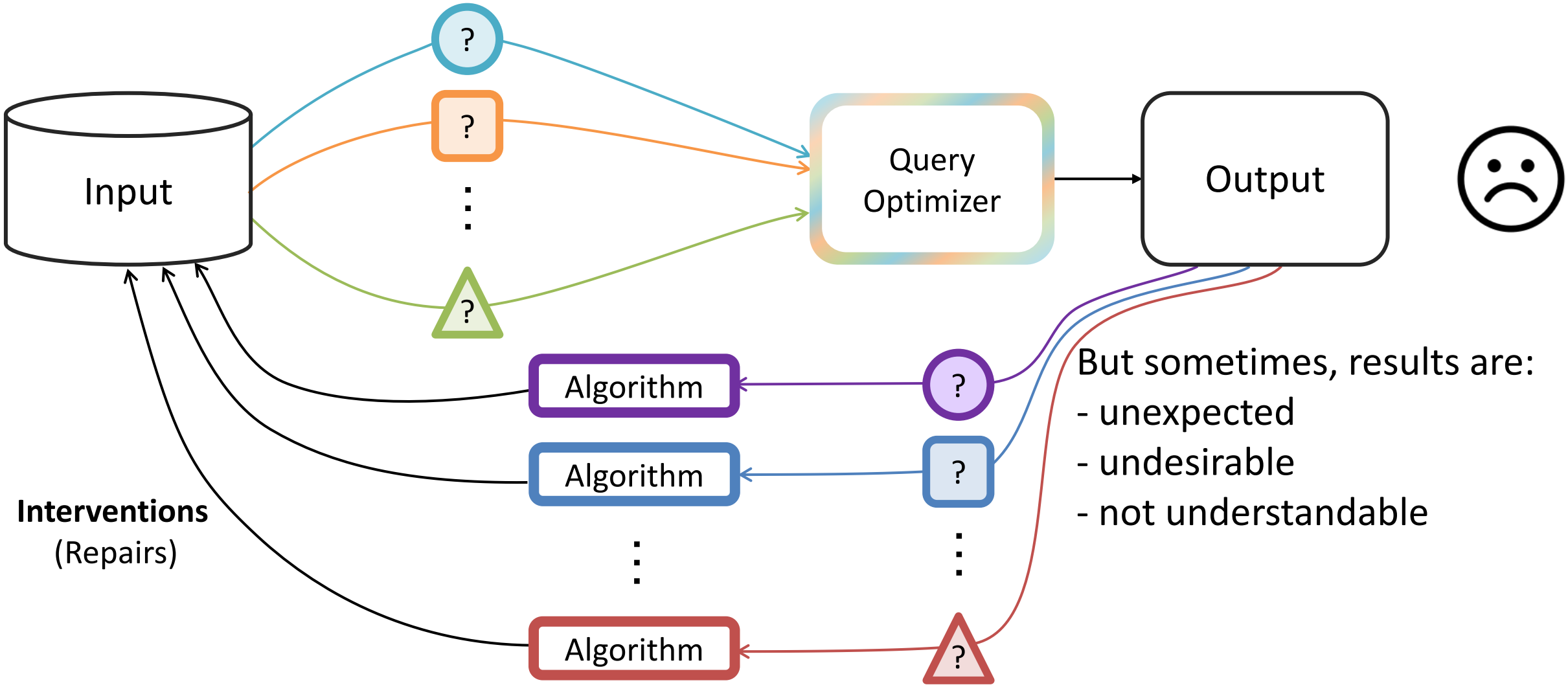
# Reverse Data Management




But sometimes, results are:
- unexpected
- undesirable
- not understandable

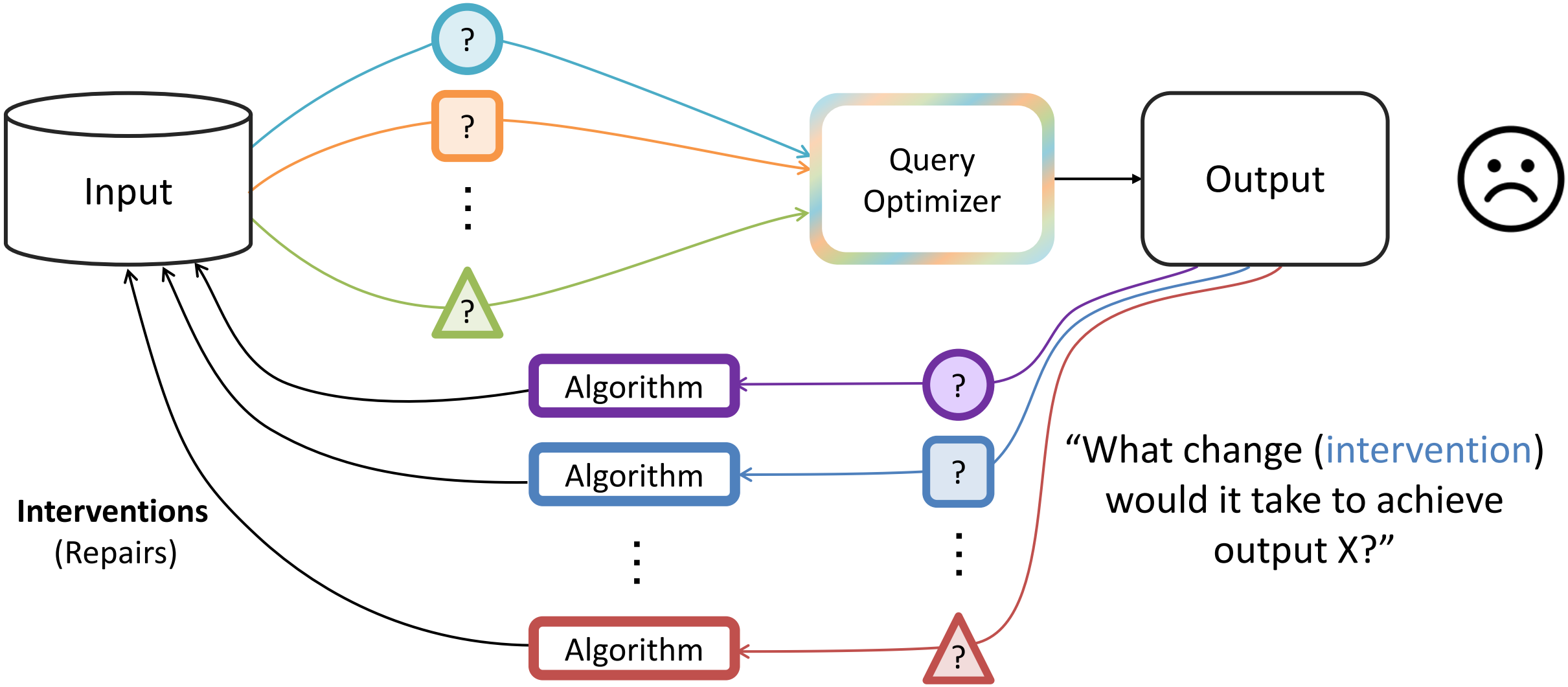
**Reverse** user questions: Why, how to, what if…

# Reverse Data Management




Meliou, Gatterbauer, Suciu. Reverse Data Management, VLDB 2011 https://doi.org/10.14778/3402755.3402803
Makhija, Gatterbauer. A Unified Approach for Resilience and Causal Responsibility with ILP and LP. SIGMOD24, Santiago. https://northeastern-datalab.github.io/unified-reverse-data-management/

# Reverse Data Management

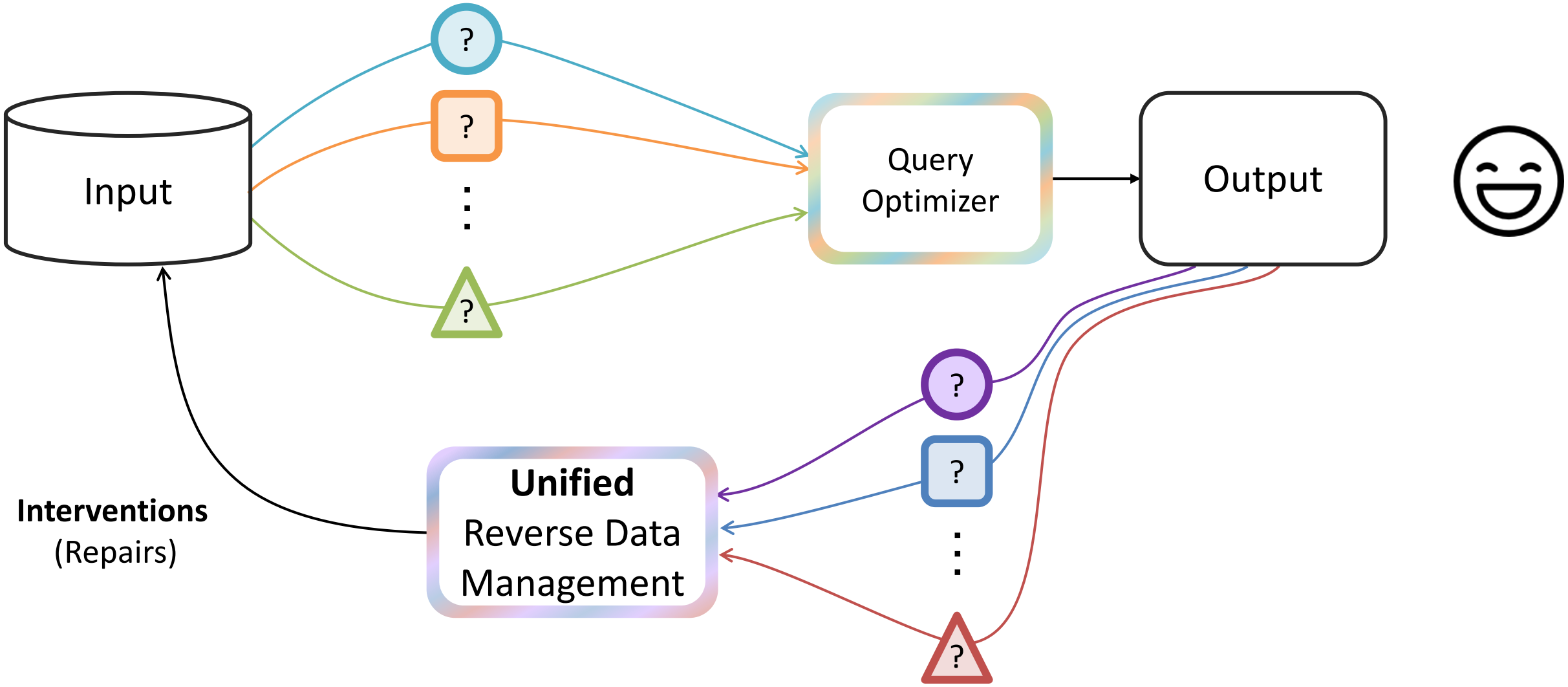



**Reverse** user questions: Why, how to, what if…


Meliou, Gatterbauer, Suciu. Reverse Data Management, VLDB 2011 https://doi.org/10.14778/3402755.3402803

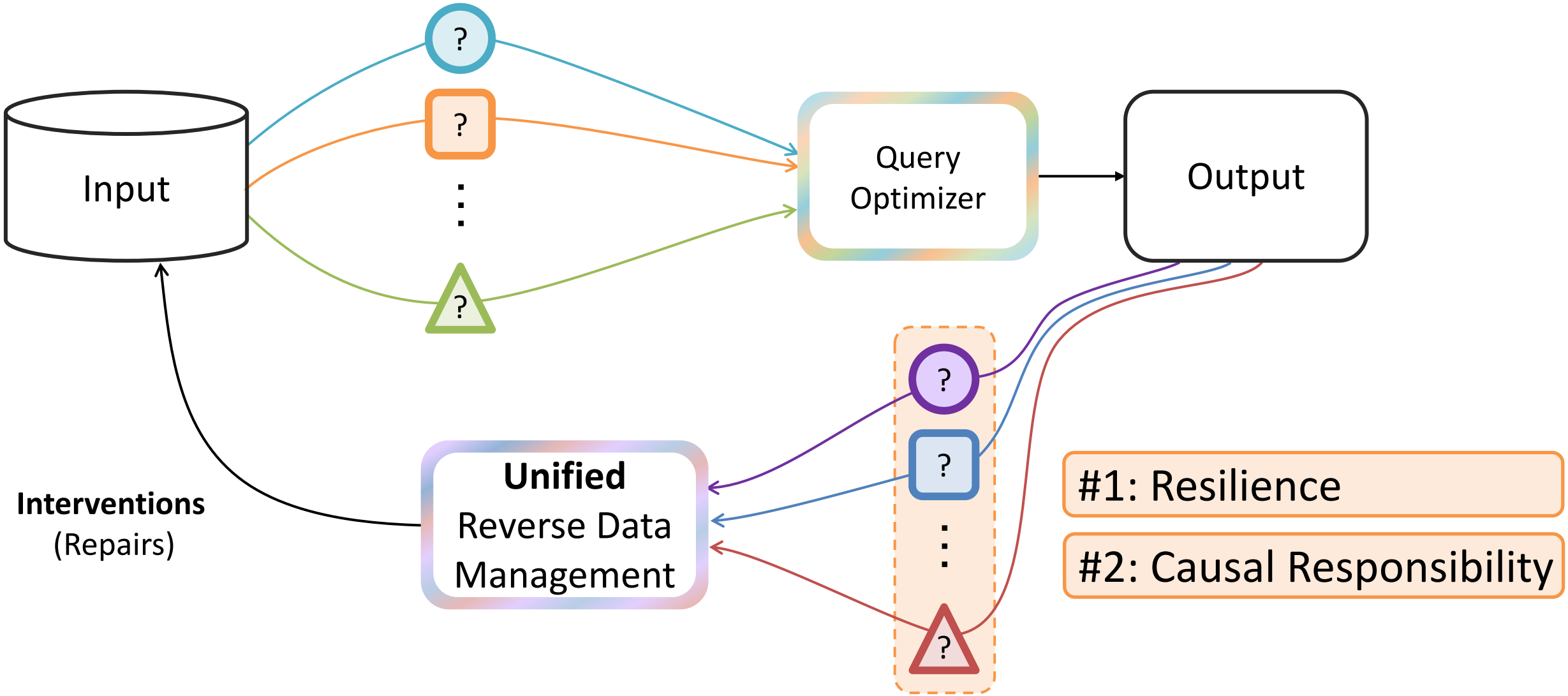
Makhija, Gatterbauer. A Unified Approach for Resilience and Causal Responsibility with ILP and LP. SIGMOD24, Santiago. https://northeastern-datalab.github.io/unified-reverse-data-management/

# Unified Reverse Data Management: Our Vision



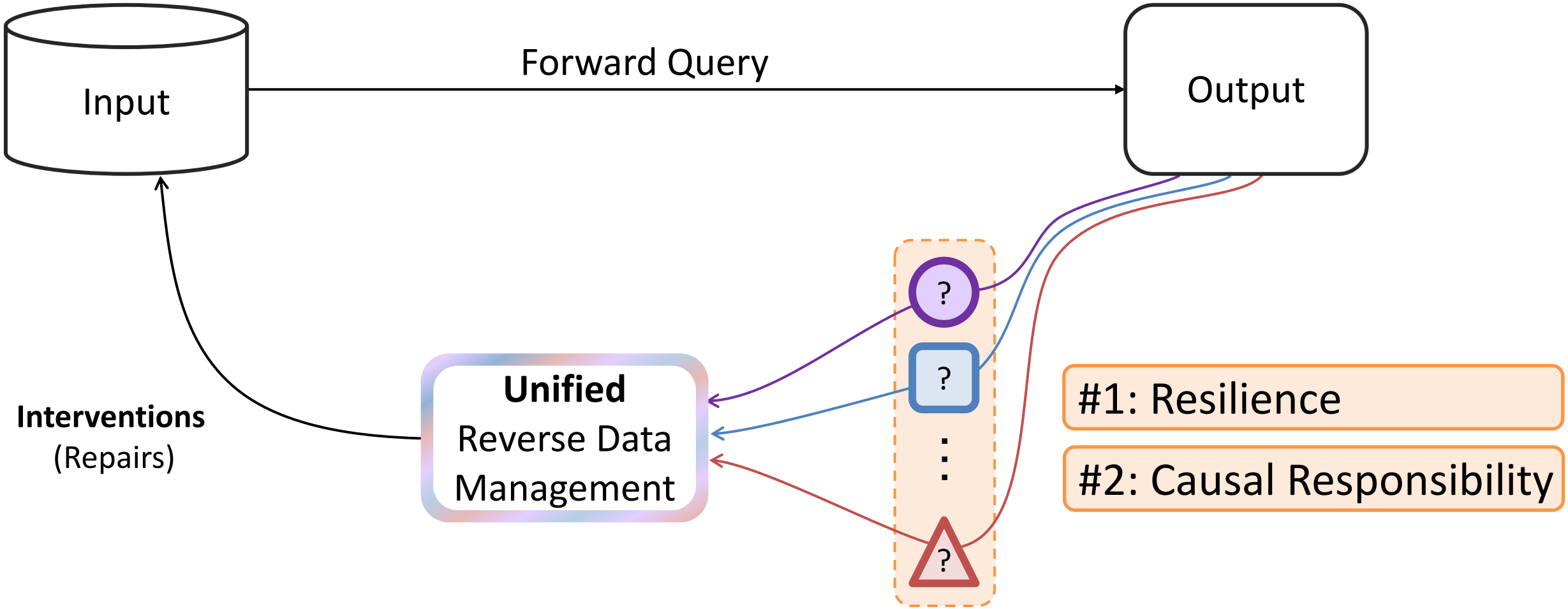

**Reverse** user questions: Why, how to, what if…

# Reverse Data Management: Our Focus



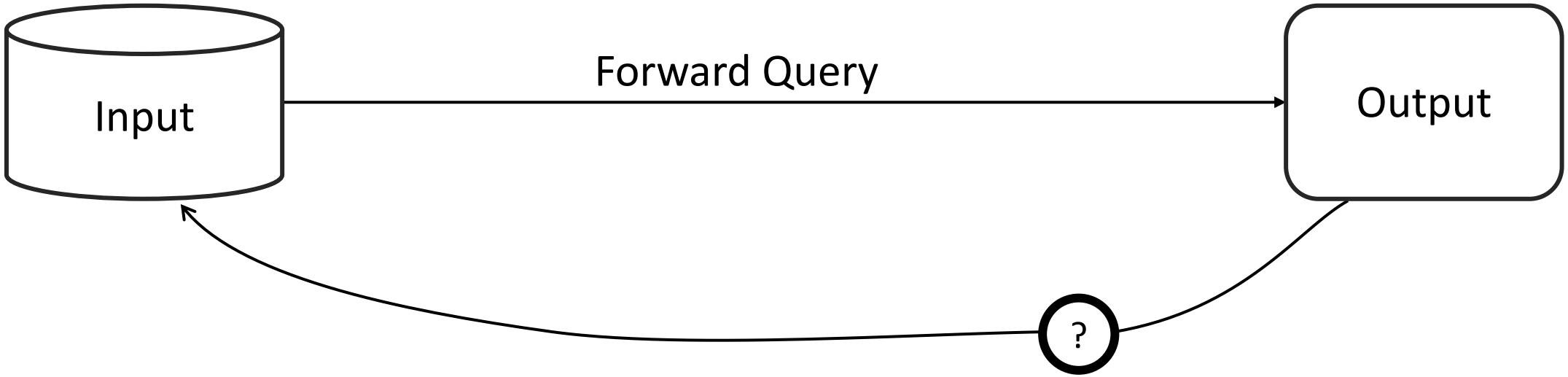

**Reverse** user questions: Why, how to, what if…

# Reverse Data Management: Our Focus




**Reverse** user questions: Why, how to, what if…

# Reverse Data Management: Resilience



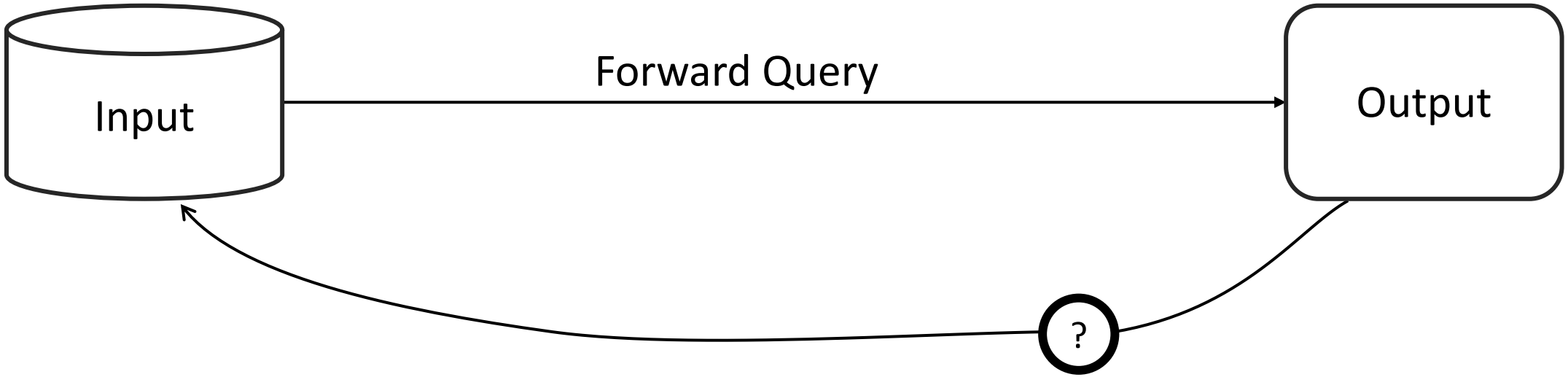

"What change (intervention) would it take to achieve no output?"

#1: Resilience

- Diagnose Points of Failure
- Equivalent to Deletion Propagation with Source Side-Effects

Freire, Gatterbauer, Immerman, Meliou. The complexity of resilience and responsibility for self-join-free conjunctive queries, PVLDB 2015 https://dl.acm.org/doi/10.14778/2850583.2850592
Makhija, Gatterbauer. A Unified Approach for Resilience and Causal Responsibility with ILP and LP. SIGMOD24, Santiago. https://northeastern-datalab.github.io/unified-reverse-data-management/

# Reverse Data Management: Resilience




"What minimum change would it take to make tuple X counterfactual?"

#2: Causal Responsibility

- Halpern-Pearl Framework of Counterfactual Causality adapted to Conjunctive Queries

Freire, Gatterbauer, Immerman, Meliou. The complexity of resilience and responsibility for self-join-free conjunctive queries, PVLDB 2015 https://dl.acm.org/doi/10.14778/2850583.2850592
Makhija, Gatterbauer. A Unified Approach for Resilience and Causal Responsibility with ILP and LP. SIGMOD24, Santiago. https://northeastern-datalab.github.io/unified-reverse-data-management/

- Unified Reverse Data Management Framework
  - Example: Resilience
- Insight #1: How to build **Unified Algorithms**
- Insight #2: How to build **Automatic Hardness Provers**
- What else is in the paper?
- Takeaways + Open Questions

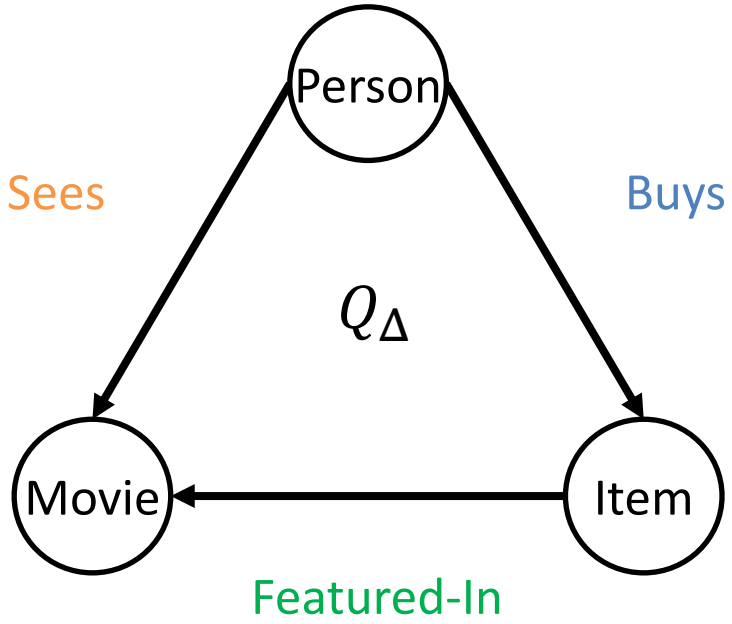
# Resilience: Example

Sees(person, movie)  Buys(person, item)  Featured-In(item, movie)

# Resilience: Example

Query:- What person sees a movie and buys an item featured in the movie?

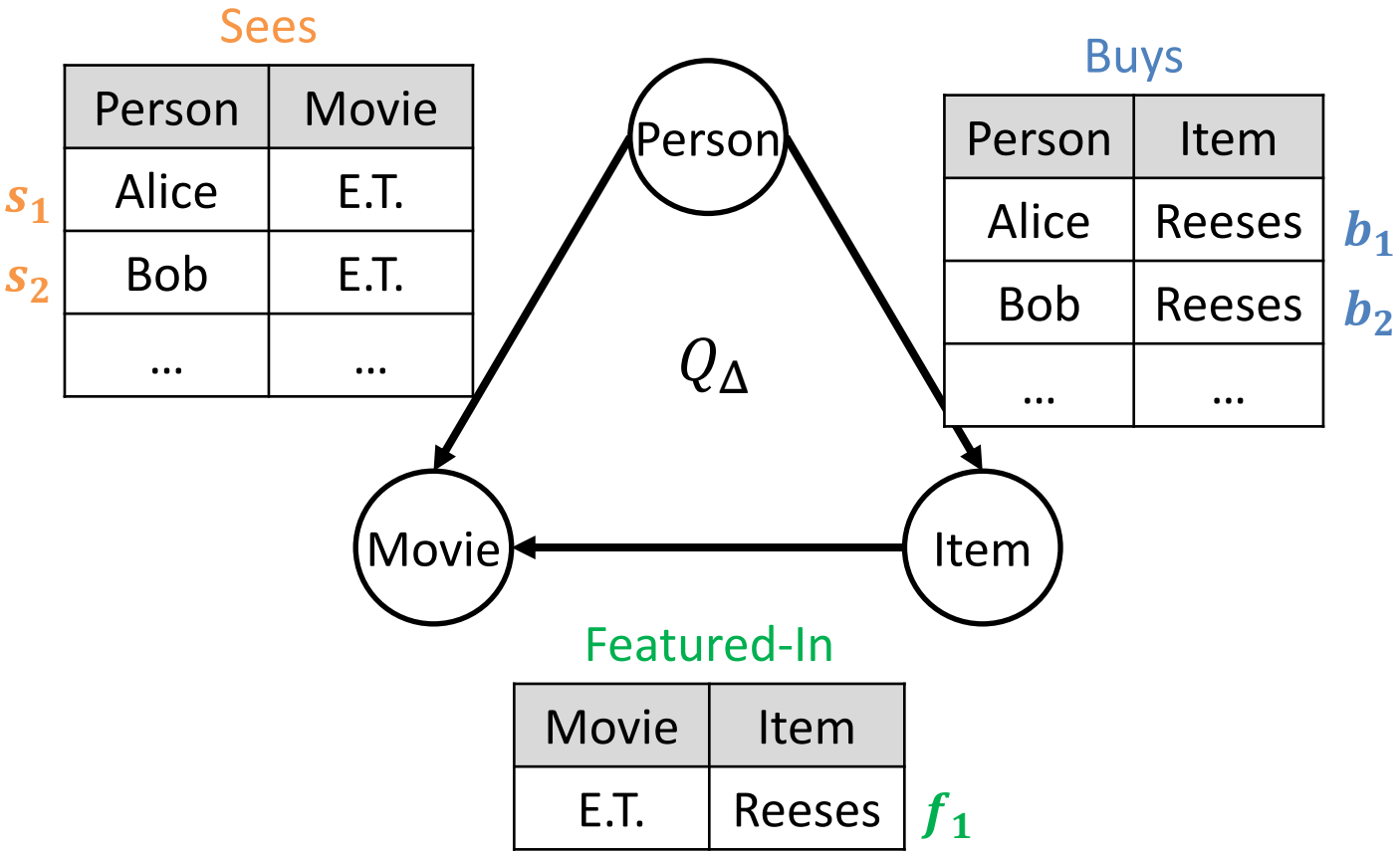
Q(person, movie, item):- Sees(person, movie), Buys(person, item), Featured-In(item, movie)



# Resilience: Example

Query:- What person sees a movie and buys an item featured in the movie?

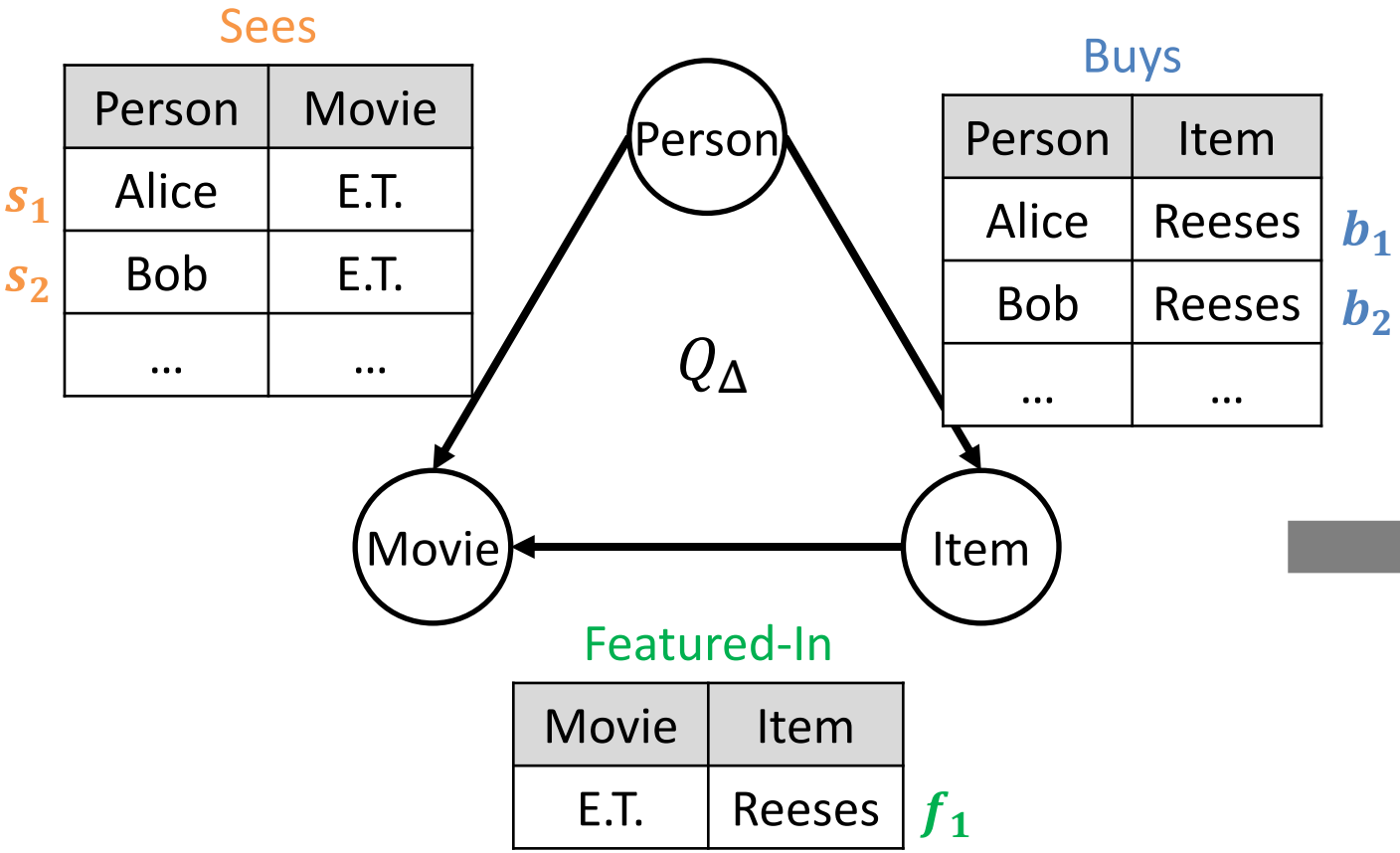
Q(person, movie, item):- Sees(person, movie), Buys(person, item), Featured-In(item, movie)



# Resilience: Example

Query:- What person sees a movie and buys an item featured in the movie?

Q(person, movie, item):- Sees(person, movie), Buys(person, item), Featured-In(item, movie)




Output

| Person | Movie | Item | |
|---|---|---|---|
| Alice | E.T. | Reeses | $s_1 b_1 f_1$ |
| Bob | E.T. | Reeses | $s_2 b_2 f_1$ |
| ... | ... | ... | ... |

Domain values

Tuples

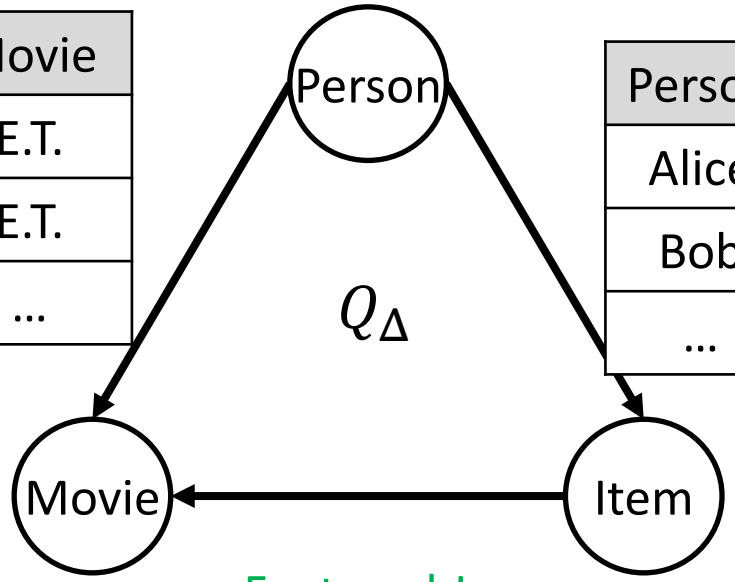
# Resilience: Example

Query:- What person sees a movie and buys an item featured in the movie?

Q(person, movie, item):- Sees(person, movie), Buys(person, item), Featured-In(item, movie)

Sees

| | Person | Movie |
|---|---|---|
| $s_1$ | Alice | E.T. |
| $s_2$ | Bob | E.T. |
| | … | … |




Buys

| Person | Item | |
|---|---|---|
| Alice | Reeses | $b_1$ |
| Bob | Reeses | $b_2$ |
| … | … | |

Featured-In

| Movie | Item | |
|---|---|---|
| E.T. | Reeses | $f_1$ |

Output

| Person | Movie | Item | |
|---|---|---|---|
| Alice | E.T. | Reeses | $s_1 b_1 f_1$ |
| Bob | E.T. | Reeses | $s_2 b_2 f_1$ |
| … | … | … | … |

Domain values

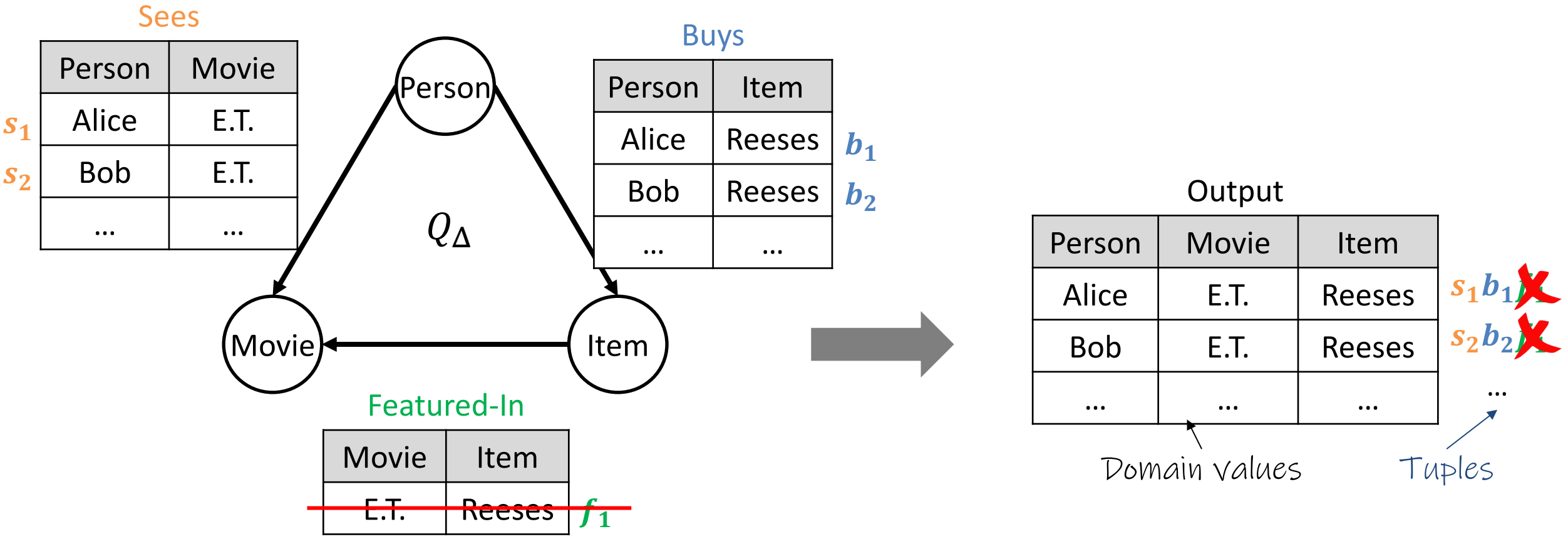
Tuples

Recall: Resilience = What minimal change would it take to delete the output?"

# Resilience: Example

Query:- What person sees a movie and buys an item featured in the movie?

Q(person, movie, item):- Sees(person, movie), Buys(person, item), Featured-In(item, movie)

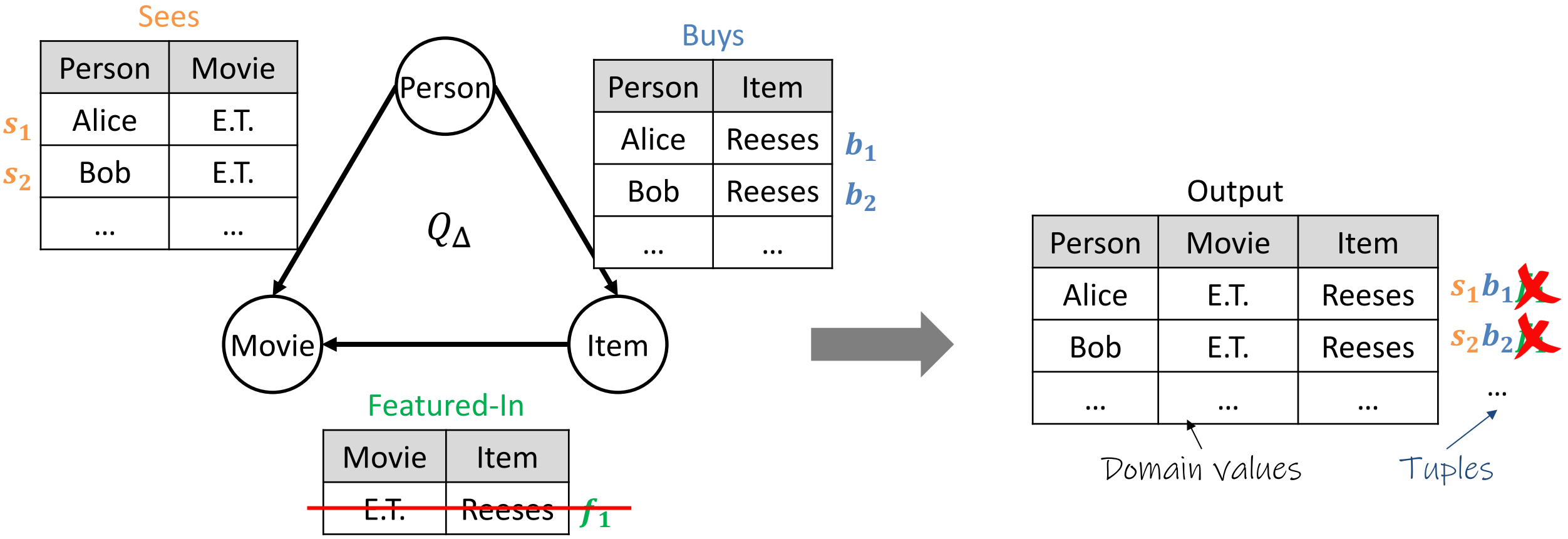



Recall: Resilience = What minimal change would it take to delete the output?"

# Resilience: Example

Query:- What person sees a movie and buys an item featured in the movie?

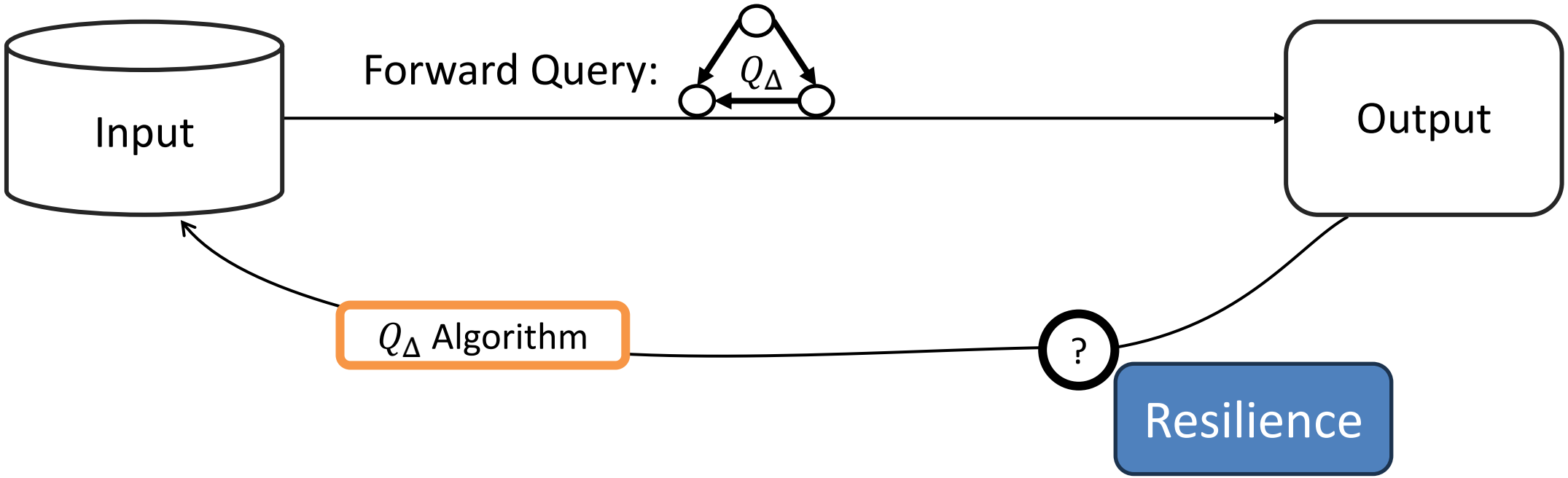
Q(person, movie, item):- Sees(person, movie), Buys(person, item), Featured-In(item, movie)




Recall: Resilience = What minimal change would it take to delete the output?"

# Resilience: Example

Q(person, movie, item):- Sees(person, movie), Buys(person, item), Featured-In(item, movie)

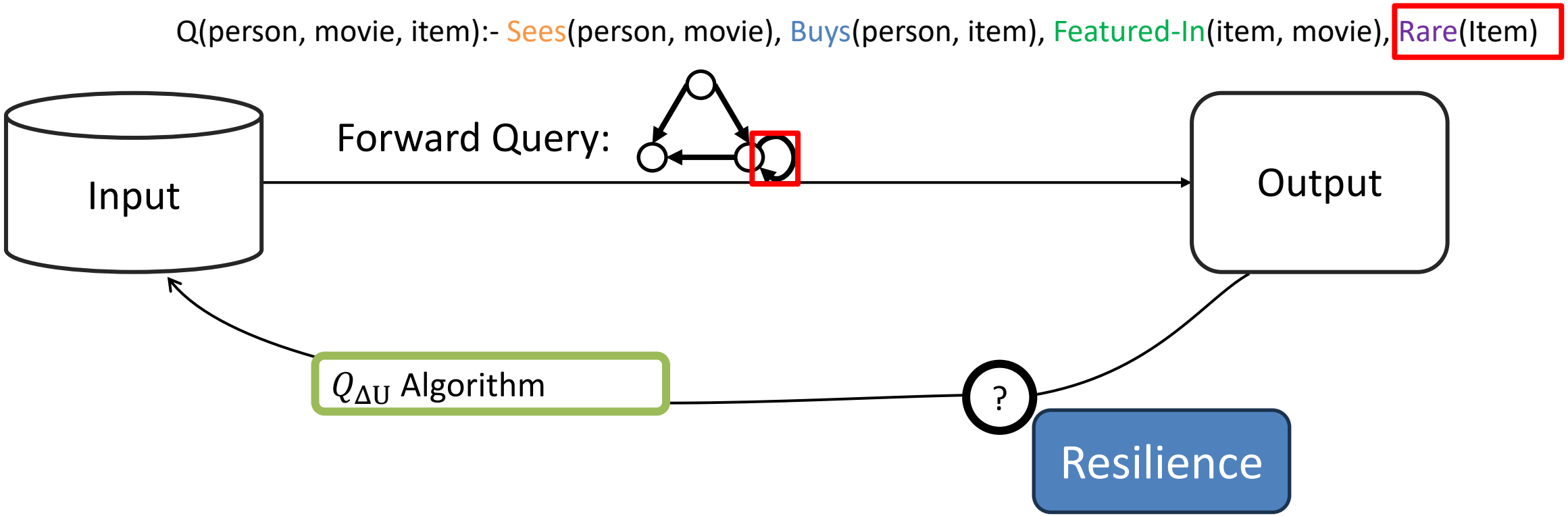



This is a NP-Hard Problem

- Optimal algorithm is exponential!

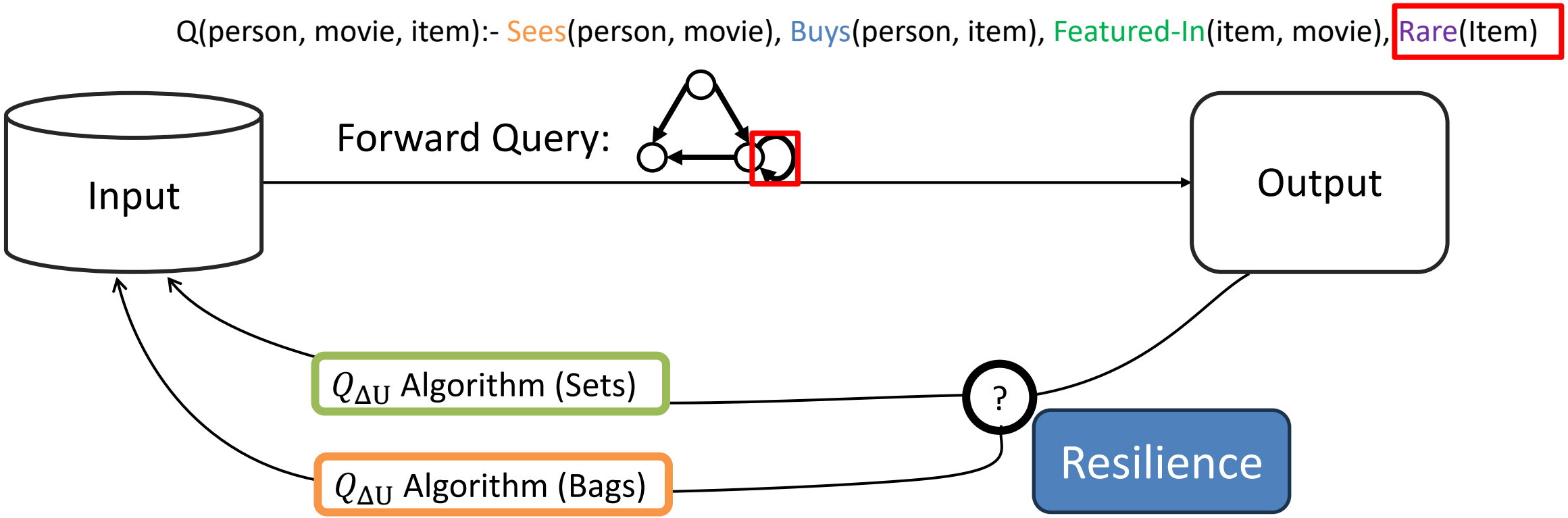
(Unless P=NP)

# Resilience: Example




Minor modification makes query PTIME

# Resilience: Example



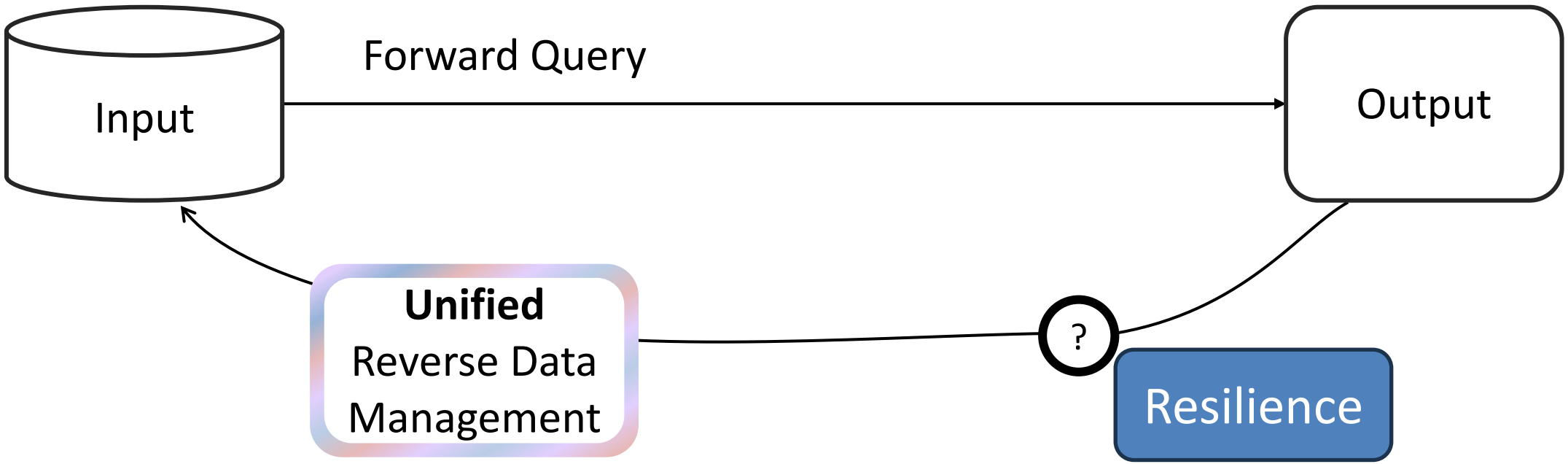

Minor modification makes query PTIME

- But only under set semantics!

If Input uses Bag semantics:

- Optimal algorithm is exponential! (Unless P=NP)

# Resilience: Example




Unified approach is *automatically* optimal

- For all known queries
- For bag / set semantics

- Unified Reverse Data Management Framework
- Insight #1: How to build **Unified Algorithms**
- Insight #2: How to build **Automatic Hardness Provers**
- What else is in the paper?
- Takeaways + Open Questions

# What is a Unified Algorithm?

1. Unified Across Complexity (Easy or Hard)
- Algorithm that can solve PTIME and NP-C problems
- Guaranteed exact PTIME termination for all known tractable cases

2. Unified Across Settings
- Set + *Bag* Semantics
- All Conjunctive Queries (including *all self-joins*)
- Can take advantage of *unspecified* Functional Dependencies

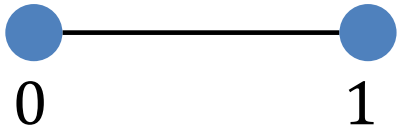
Orange keywords = contrast from previous work

Freire, Gatterbauer, Immerman, Meliou. The complexity of resilience and responsibility for self-join-free conjunctive queries, PVLDB 2015 https://dl.acm.org/doi/10.14778/2850583.2850592
Freire, Gatterbauer, Immerman, Meliou. New results for the complexity of resilience for binary conjunctive queries with self-joins, PODS 2020 https://doi.org/10.1145/3375395.3387647
Makhija, Gatterbauer. A Unified Approach for Resilience and Causal Responsibility with ILP and LP. SIGMOD24, Santiago. https://northeastern-datalab.github.io/unified-reverse-data-management/

# Detour: Integer Linear Programs

Integer Linear Program (ILP)

$$\min w^T x$$
$$s.t. Ax \leq b$$
$$x \in \{0, 1\}$$



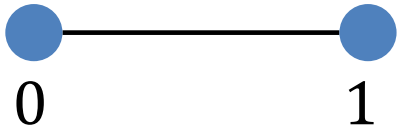

NP-Hard

Highly Optimized in Practice!

# Detour: Integer Linear Programs

Integer Linear Program (ILP)

$$\min w^T x$$
$$s.t. Ax \leq b$$
$$x \in \{0, 1\}$$




NP-Hard

Highly Optimized in Practice!

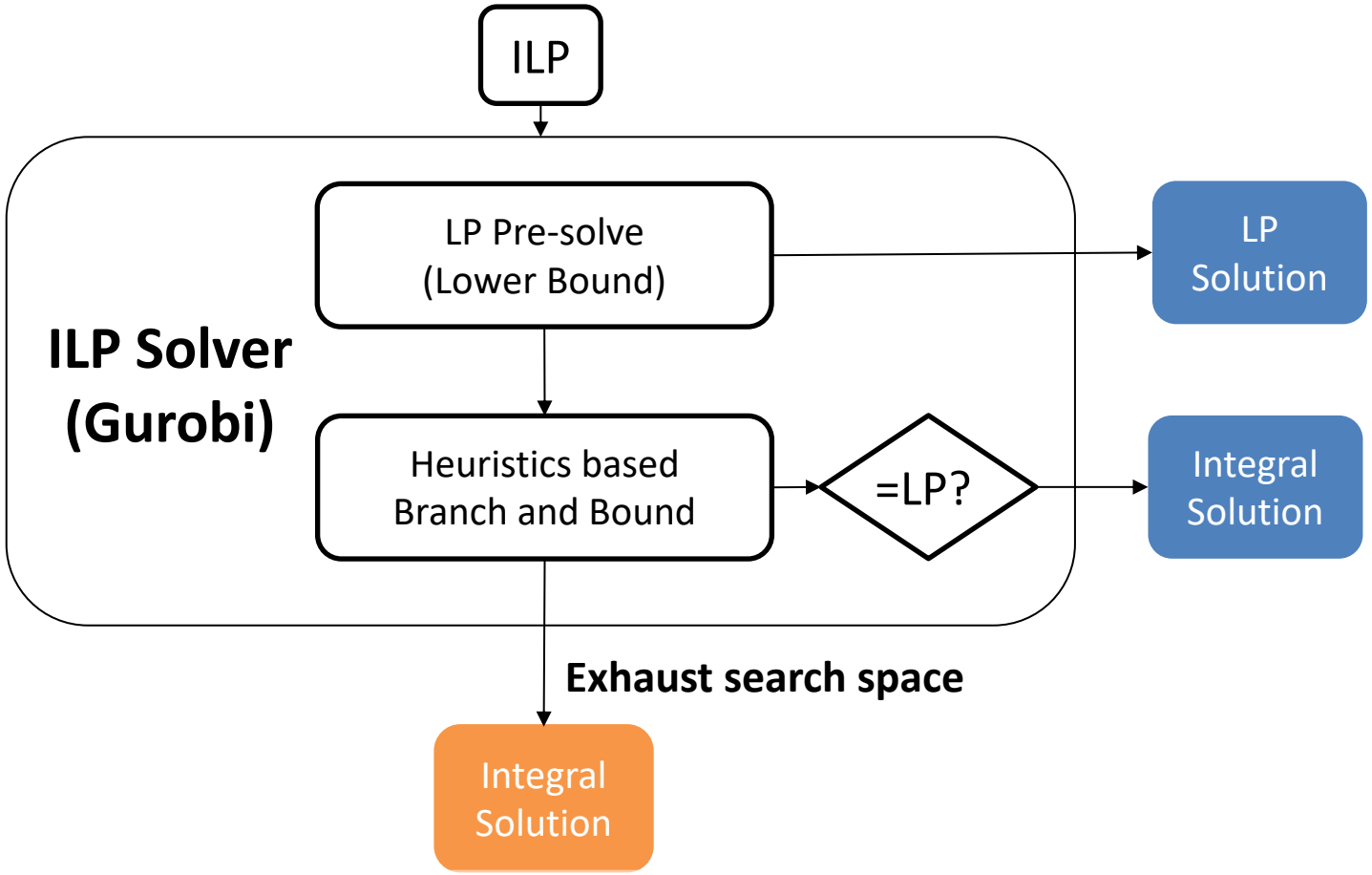
Linear Program (LP)

$$\min w^T x$$
$$s.t. Ax \leq b$$
$$x \in [0, 1]$$


PTIME

Natural Lower Bound

# Detour: Integer Linear Programs




Insight: If **LP=ILP**, then a solution can be recovered efficiently!

Based on a description of the inner workings of Gurobi (https://www.gurobi.com/resources/mixed-integer-programming-mip-a-primer-on-the-basics/ )
Makhija, Gatterbauer. A Unified Approach for Resilience and Causal Responsibility with ILP and LP. SIGMOD24, Santiago. https://northeastern-datalab.github.io/unified-reverse-data-management/

# Resilience as an Integer Linear Program (ILP)

$$\min \ \mathbf{c^T x}$$

objective vector

constraint vector

$$\mathbf{A x} \geq \mathbf{b}$$

constraint matrix

$$\mathbf{x} \in \{0,1\}^n$$

Weights if applied can go in the objective

**Theorem.**

For all known PTIME cases of resilience, the objective value of the LP relaxation is identical to the ILP.

$Q_\Delta$ Example:

Output

| Person | Movie | Item |
|---|---|---|
| Alice | E.T. | Reeses |
| Bob | E.T. | Reeses |
| … | … | … |

$s_1 b_1 f_1$

$s_2 b_2 f_1$

…

Queries with self-joins are modelled the same way. Not all constraints must have the same arity.

# Resilience as an Integer Linear Program (ILP)

$$\min \ \mathbf{c^T x}$$

objective vector

constraint vector

$$\mathbf{Ax} \geq \mathbf{b}$$

constraint matrix

$$\mathbf{x} \in \{0,1\}^n$$

$Q_\Delta$ Example:

Output

| Person | Movie | Item |
|---|---|---|
| Alice | E.T. | Reeses |
| Bob | E.T. | Reeses |
| … | … | … |

$s_1 b_1 f_1$

$s_2 b_2 f_1$

…

**Theorem.**

For all known PTIME cases of Resilience, LP=ILP

If LP=ILP, solvers can recover a solution efficiently!

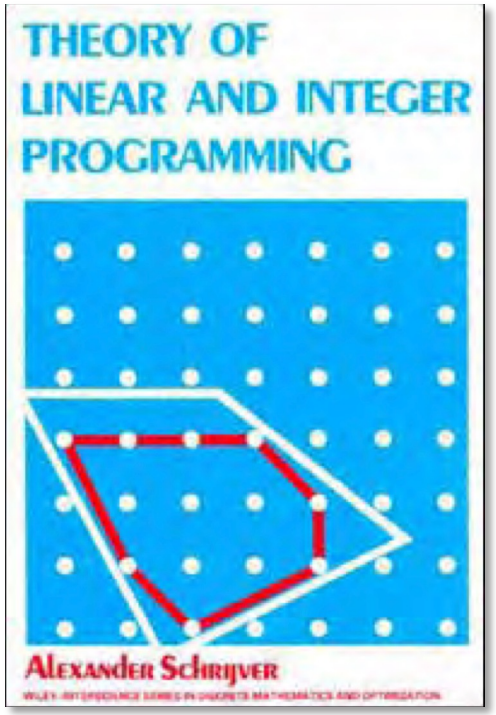
# When does LP = ILP?

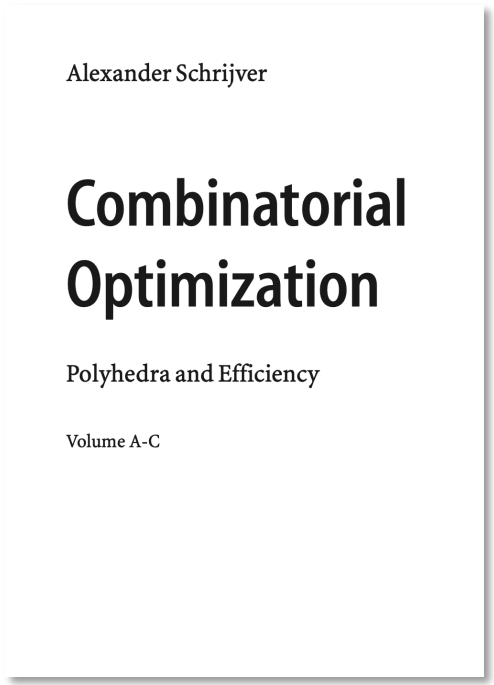
**Theorem.**

For all known PTIME cases of Resilience, LP=ILP

Our PTIME constraint matrixes need not be balanced or Totally Unimodular

The PTIME cases go beyond these known criteria!

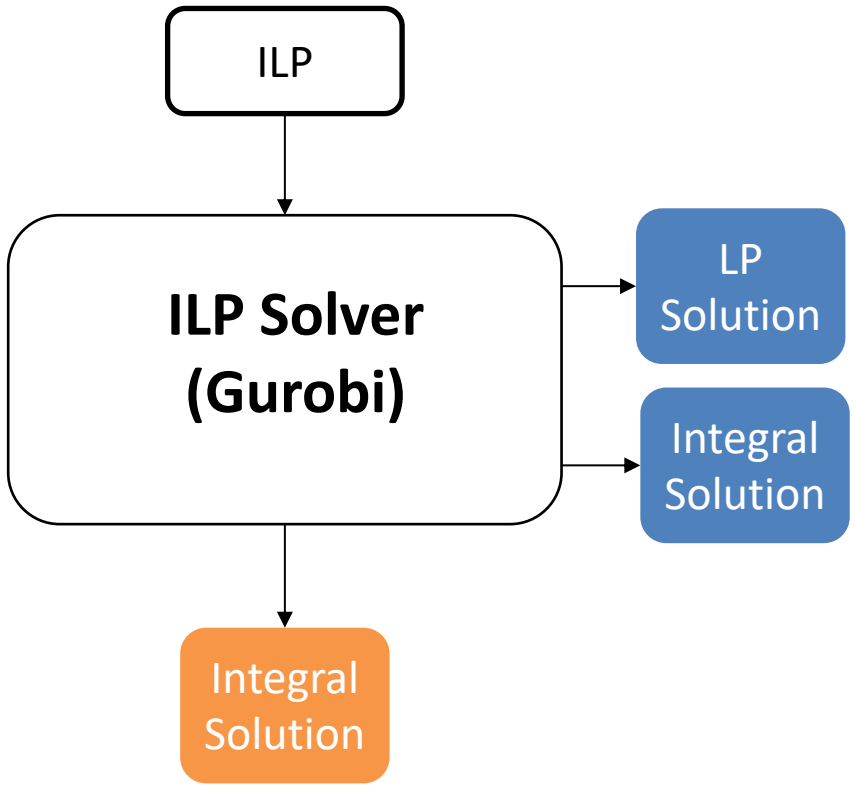




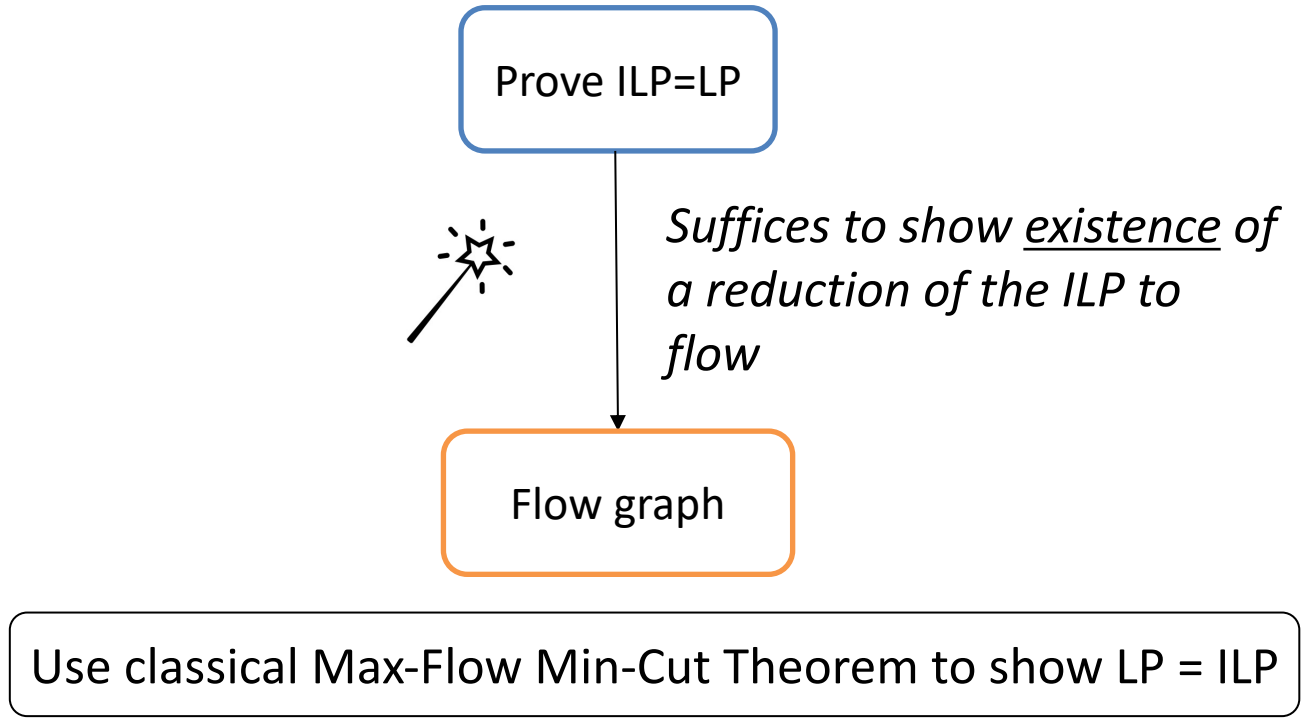

Show correspondence to a flow graph as a criterion for when **LP=ILP**

# Unified Approach for Resilience

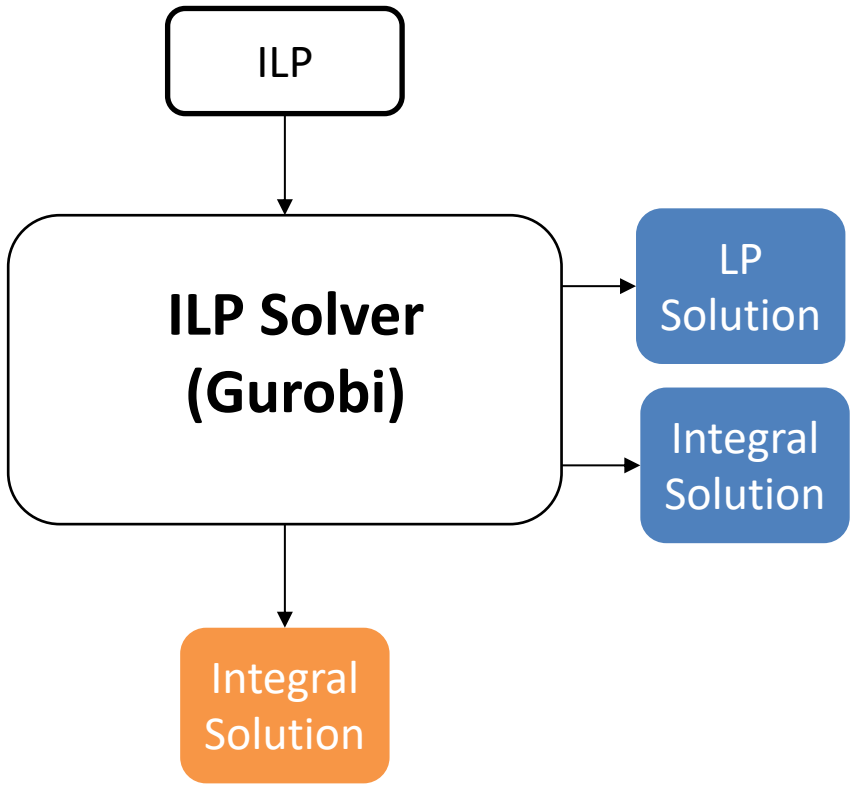



Typical Goal: Find algorithm for easy cases

Our Goal: *Just* prove case is easy

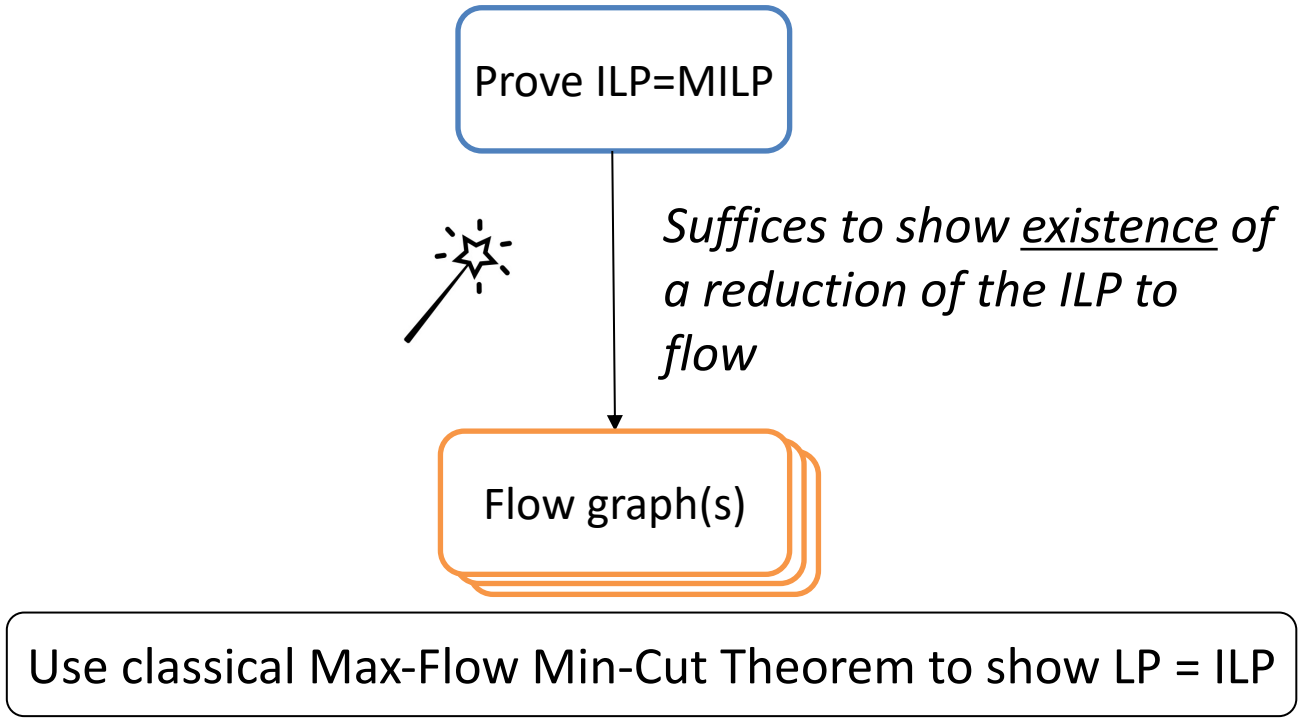


# Unified Approach for Causal Responsibility

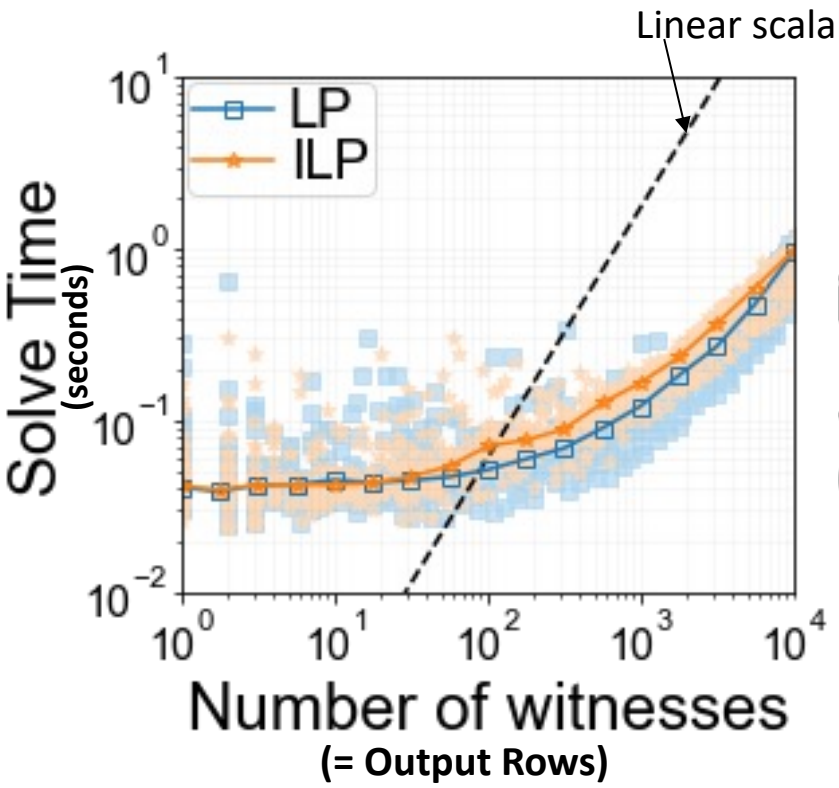



Typical Goal: Find algorithm for easy cases

Our Goal: *Just* prove case is easy

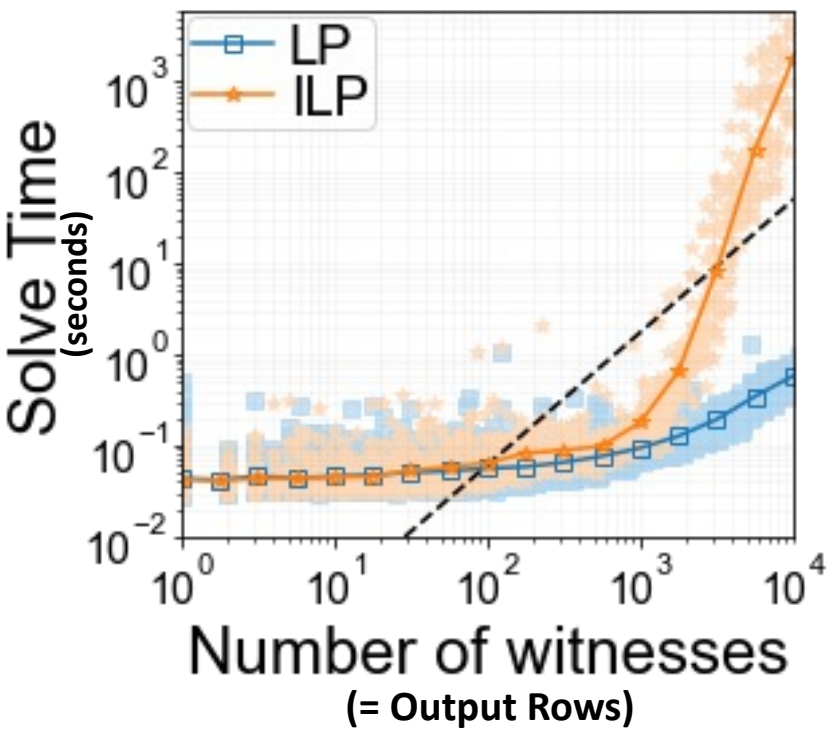


# Experiments: Unified Resilience ILP



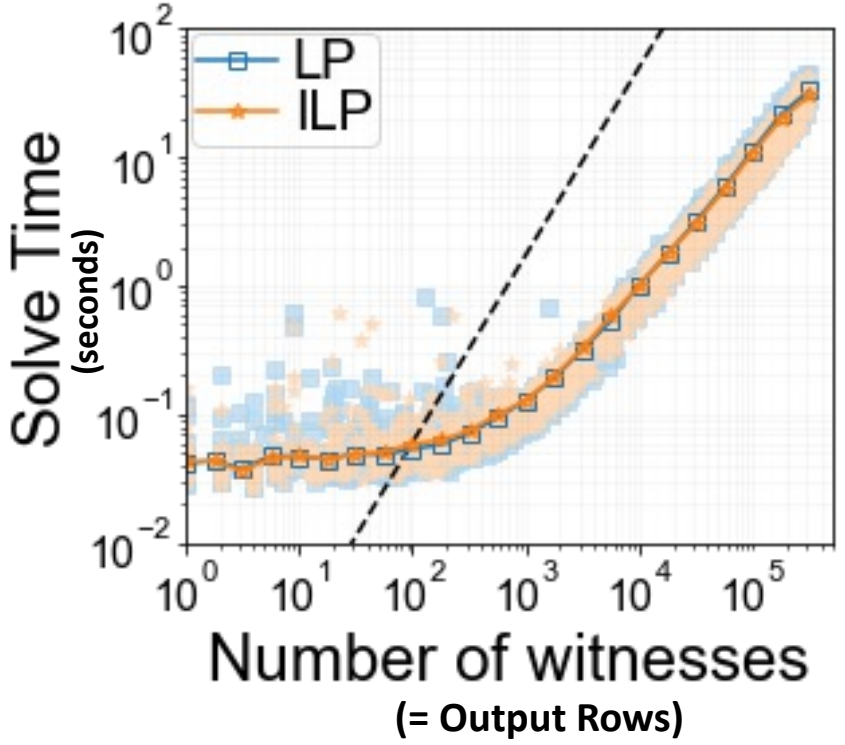

Q1: SJ-Confluence

Known PTIME




Q2: SJ-Chain

Known NP-Complete



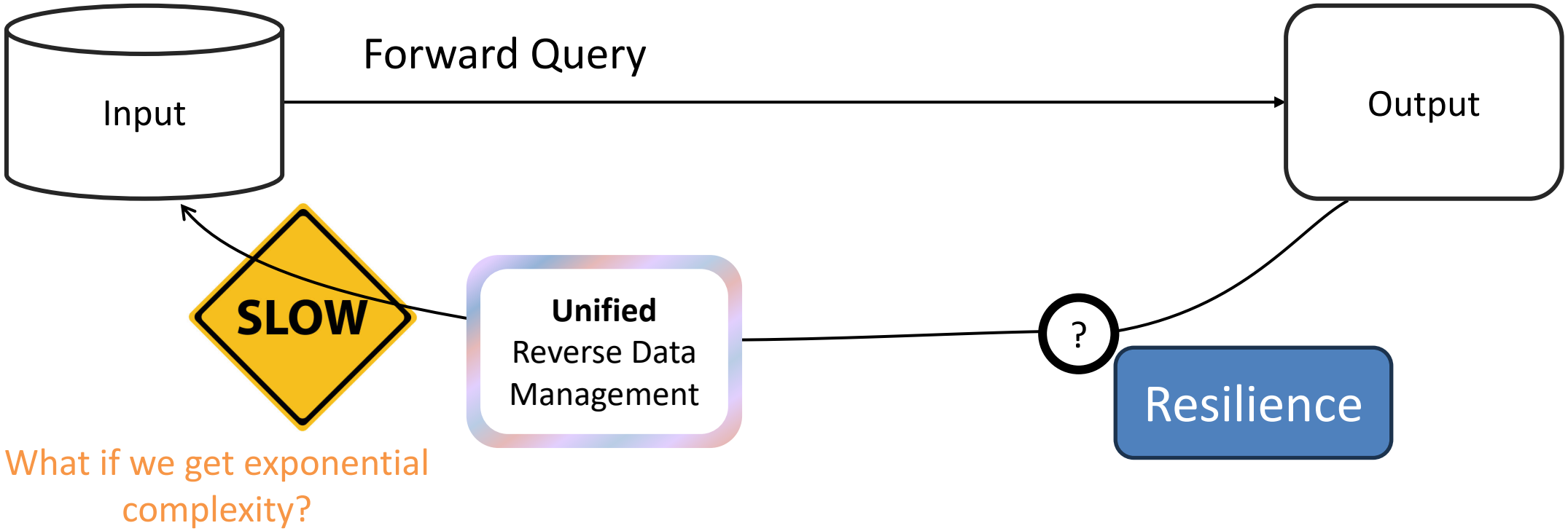

Q3: SJ-z6

Previously Open

We show NP-Complete

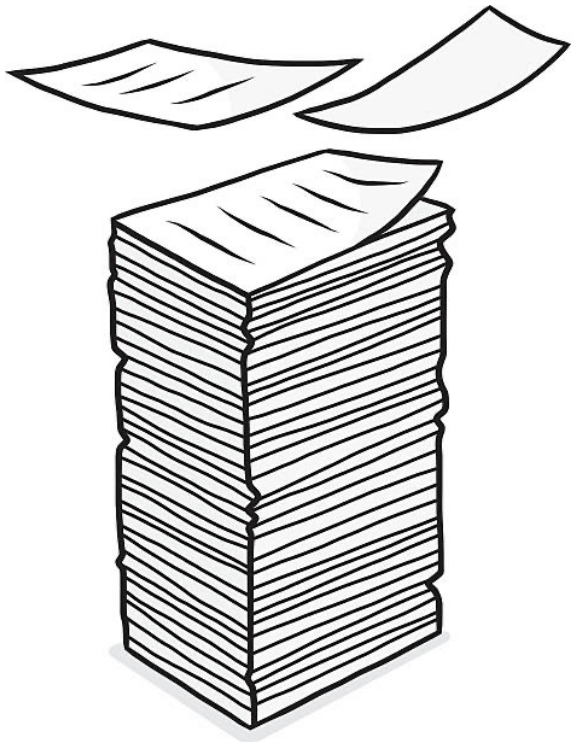
- Unified Reverse Data Management Framework
- Insight #1: How to build **Unified Algorithms**
- Insight #2: How to build **Automatic Hardness Provers**
- What else is in the paper?
- Takeaways + Open Questions

# Unified Approach to prove Hardness

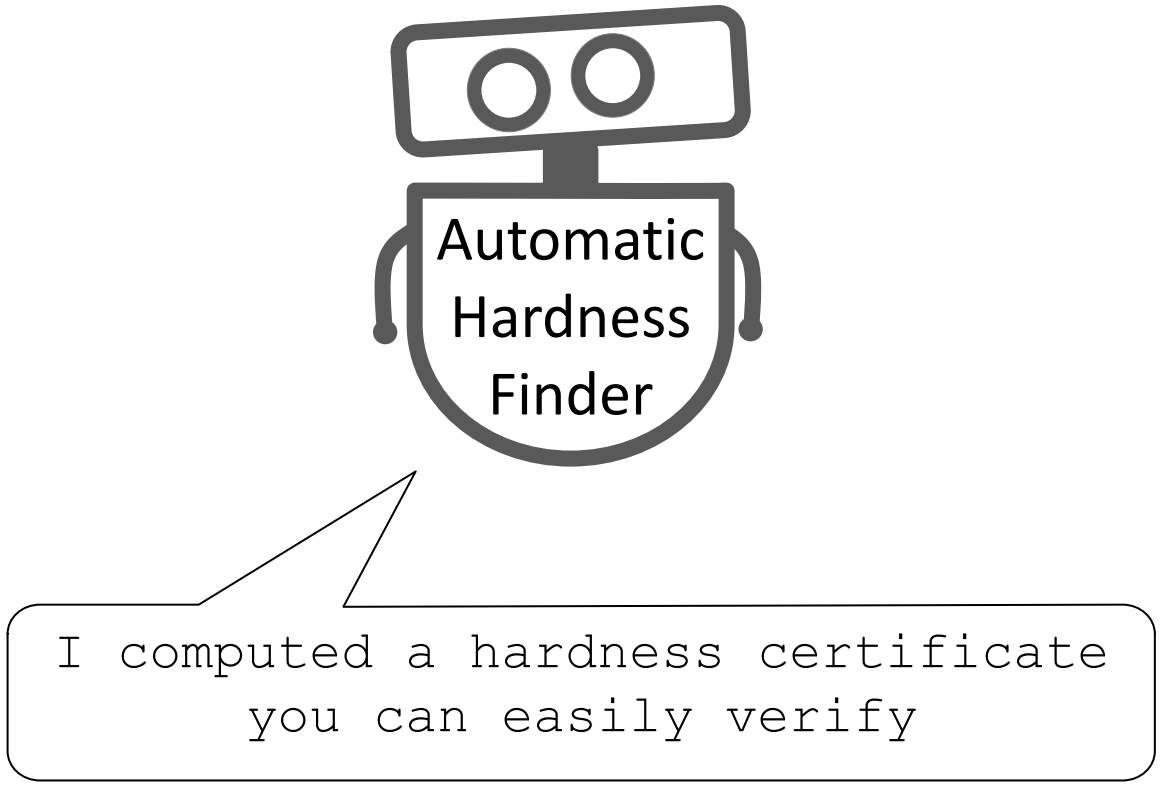



How to prove that exponential complexity is optimal (best we can do)?

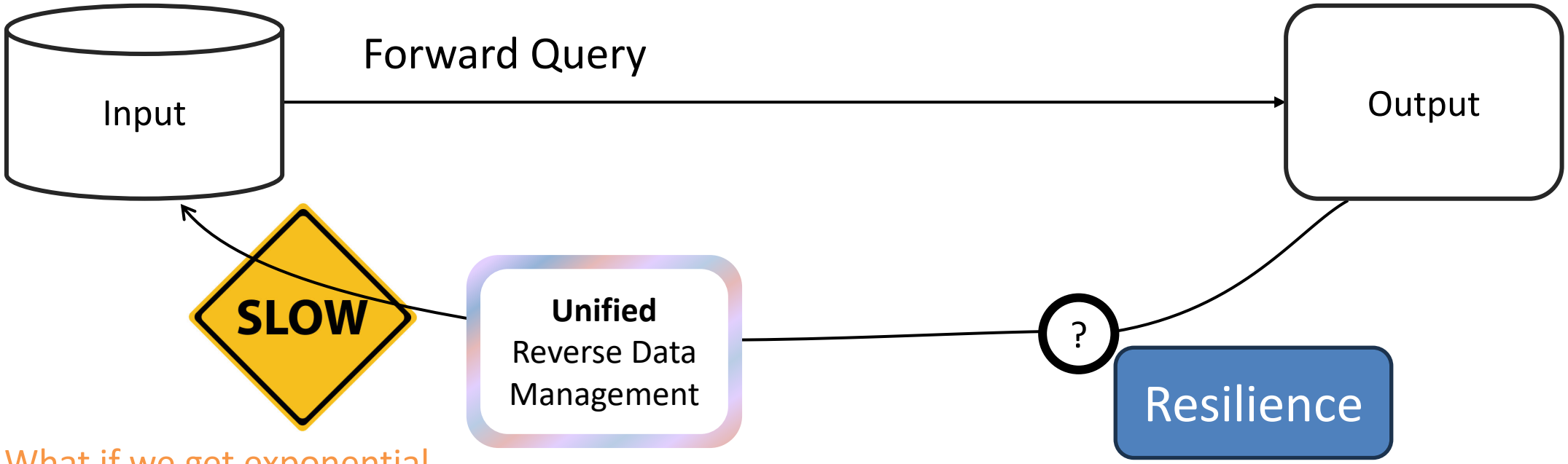
# Unified Approach to prove Hardness



*……. Hence, If you can solve Problem A, you can solve a NP-hard problems like Vertex Cover*



# Unified Approach to prove Hardness

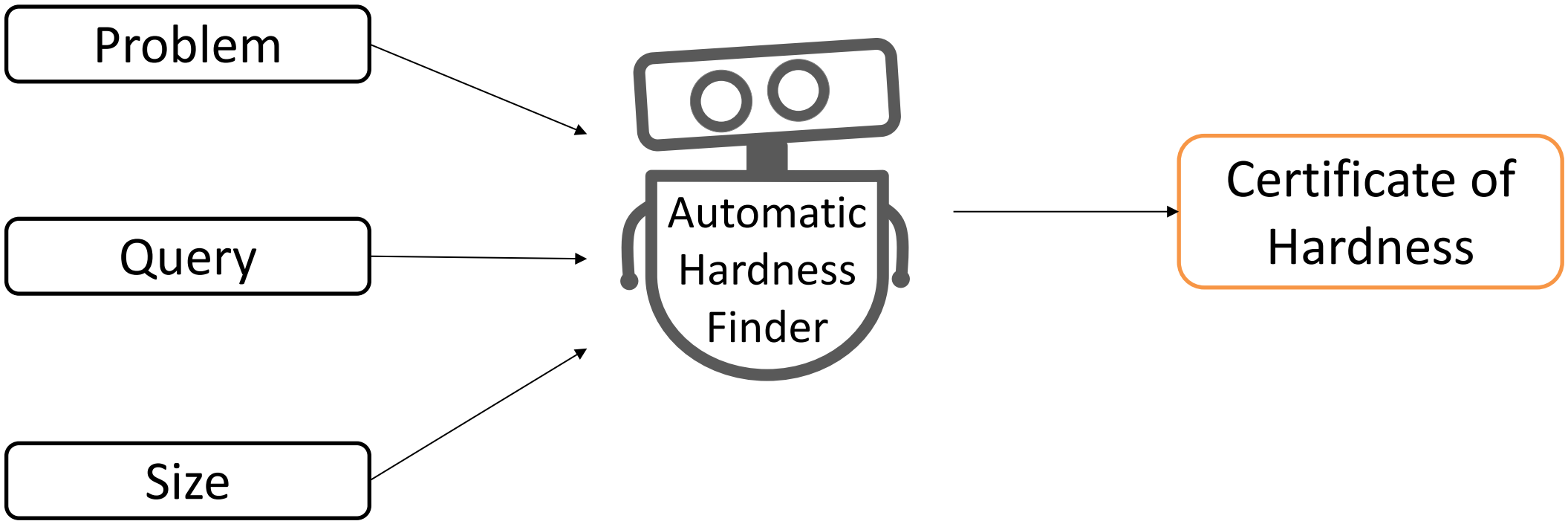



What if we get exponential complexity?

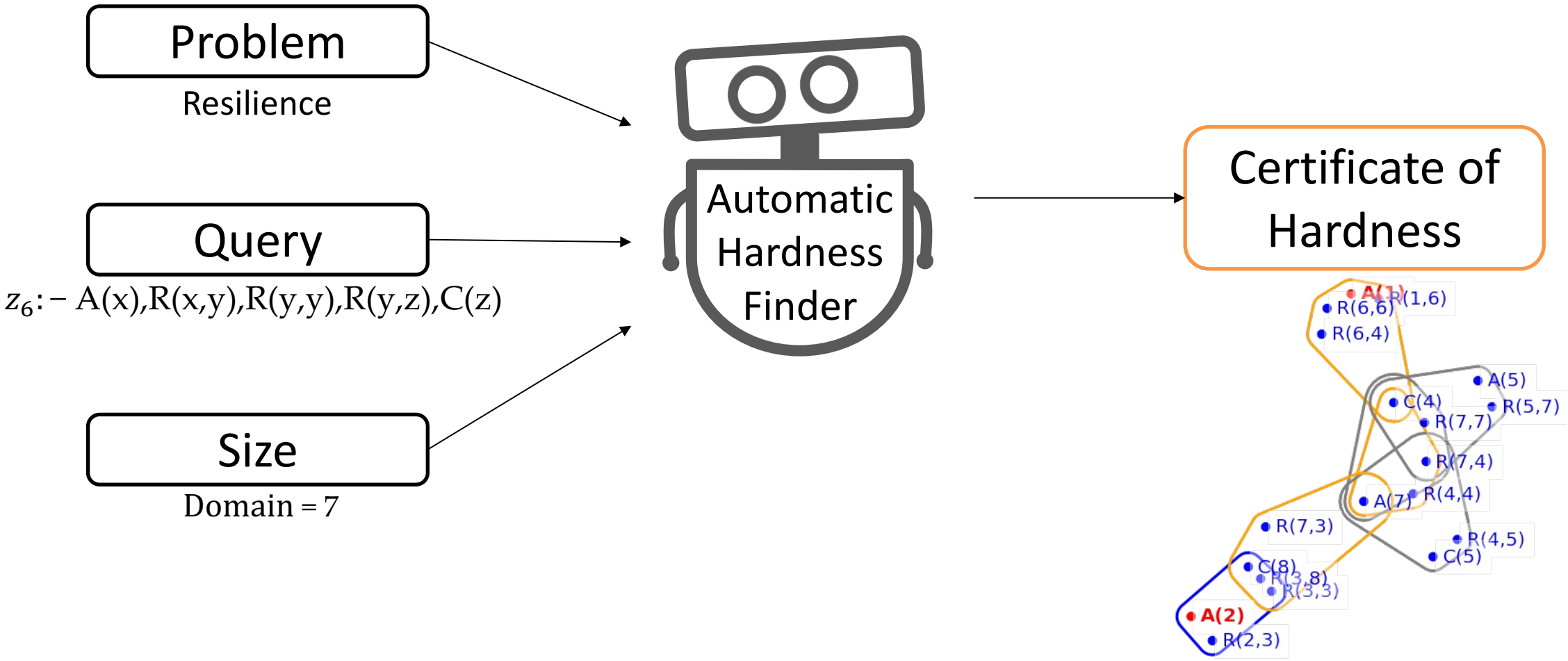
***automatically***

How to prove that exponential complexity is optimal (best we can do)?

# Unified Approach to prove Hardness



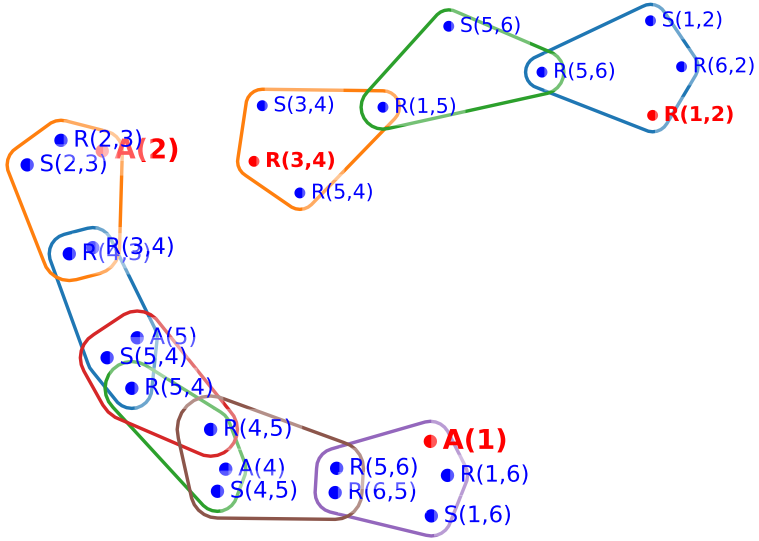
# Unified Approach to prove Hardness



# 5 New Hardness Gadgets

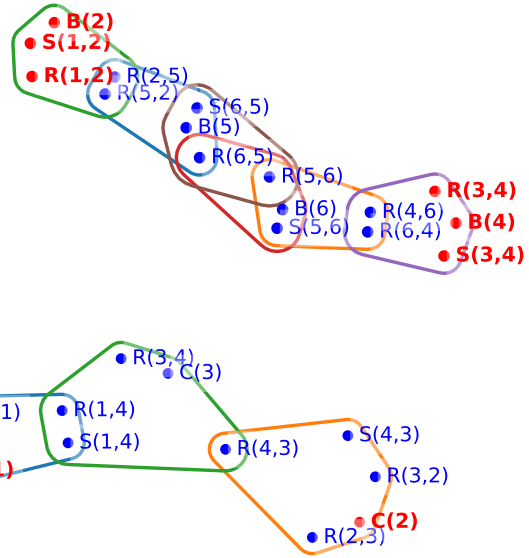
- Using the Automatic Hardness Finder, we proved 5 queries hard (out of 7 previously open from Freire+20)

$q^{S}_{3cc}$:-R(x,y),R(y,z),R(w,z),S(w,z)

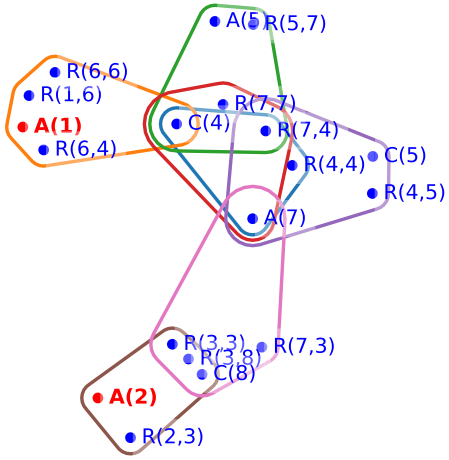



$q^{AS_{xy}}_{3perm-R}$:- A(x),S(x,y),R(x,y),R(y,z),R(z,y)

$q^{S_{xy}B}_{3perm-R}$:- S(x,y),R(x,y),B(y),R(y,z),R(z,y)




$q^{S_{xy}C}_{3perm-R}$:- S(x,y),R(x,y),R(y,z),R(z,y),C(z)




$z_6$:– A(x),R(x,y),R(y,y),R(y,z),C(z)

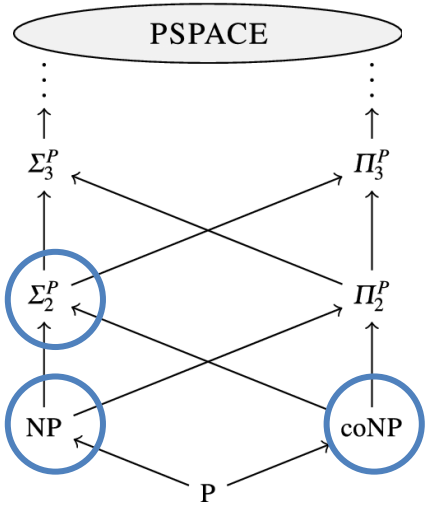
- Can recover all previous hardness results + find new ones!

# How to Build the Automatic Hardness Finder?


1. Declarative Program

2. Semantic Specification of "Hardness"
   - We show 5 properties that are sufficient to show hardness
   - These properties can be easily tested

3. Solve NP-Hard problems as a sub-routine
   - Expressivity used: $\Sigma_2^P$
   - Disjunctive Logic Programs



- Unified Reverse Data Management Framework
- Insight #1: How to build **Unified Algorithms**
- Insight #2: How to build **Automatic Hardness Provers**
- What else is in the paper?
- Takeaways + Open Questions

# What else is in the paper?

- Unified Algorithms for Resilience and Casual Responsibility
- Automatic Hardness Finder

*In this talk*

- Complexity Dichotomy for Resilience and Causal Responsibility under Bag semantics
- More tractable cases:
  - Read-Once Instances, Functional Dependencies…
- Approximation algorithms
- Experimental Verification

# Takeaways

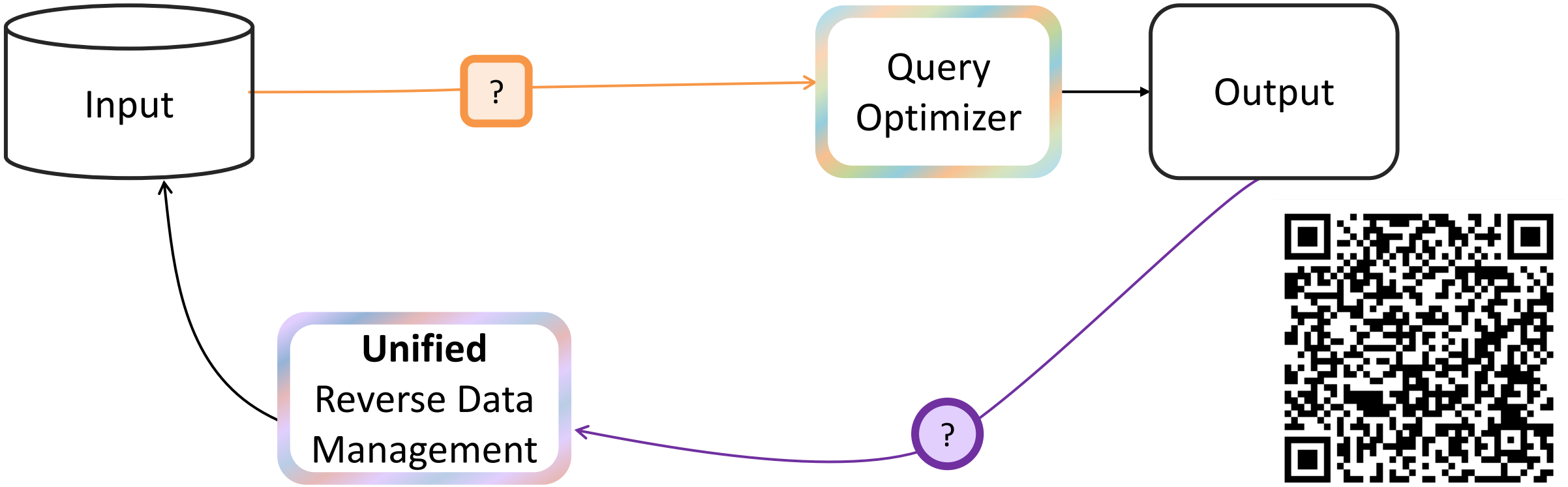
- One unified algorithm, only need to prove PTIME
- One unified hardness criterion
  - Automatic search

# Open Problems

- Which RDM problems can we solve with this unified approach?
  - Resilience
  - Minimal Factorization of Provenance of CQs (**at PODS 5pm tomorrow!**)
  - Causal Responsibility
  - …… Claim: many more: Deletion Propogation, Algorithmic Fairness, …
- Uncovering more complexity results across different problems
- Build a “Reverse Query Optimizer”

Makhija, Gatterbauer. A Unified Approach for Resilience and Causal Responsibility with ILP and LP. SIGMOD24, Santiago. https://northeastern-datalab.github.io/unified-reverse-data-management/

# Takeaways




Many more details, proofs, experiments, approximations:

- https://northeastern-datalab.github.io/unified-reverse-data-management/
- Makhija, Gatterbauer. A Unified Approach for Resilience and Causal Responsibility with Integer Linear Programming (ILP) and LP Relaxations, SIGMOD 2024
- Makhija, Gatterbauer. Minimally Factorizing the Provenance of Self-join Free Conjunctive Queries, PODS 2024

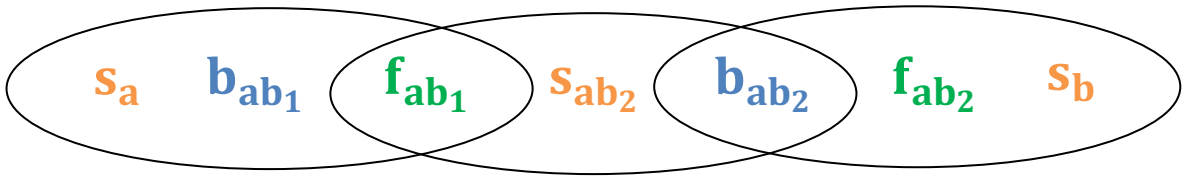
# Appendix

# Unified Approach to prove Hardness

**What is an Independent Join Path?**

Database under query $Q$, with endpoints, with 5 ***testable*** properties:

1. Data hypergraph is connected
2. Database is reduced
3. Endpoints are "valid"
4. OR – property
5. Composability

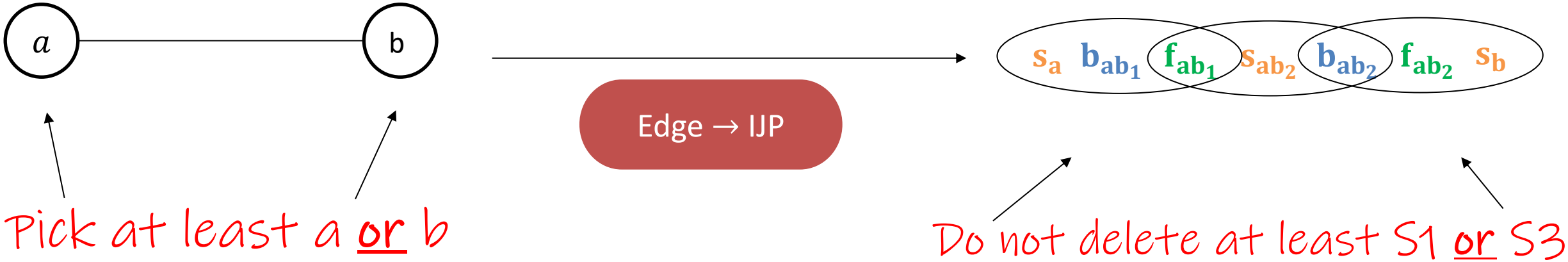



Data Hypergraph

"Key" properties:
- Semantically defined
- I will just show intuition

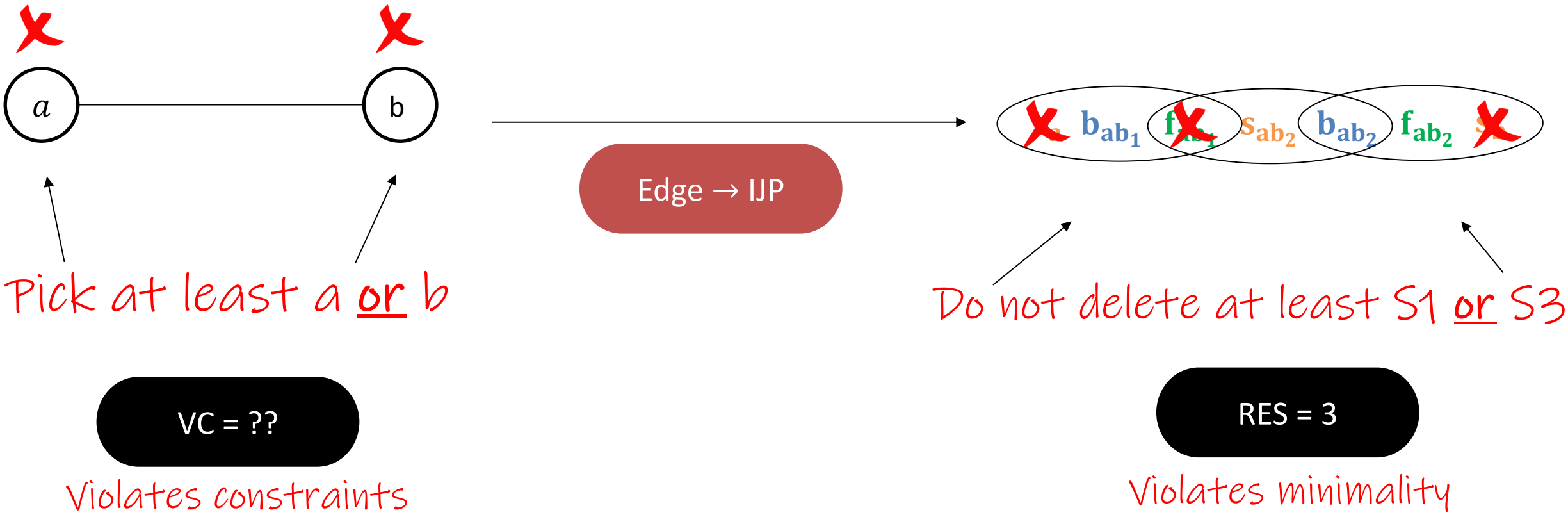
# Unified Approach to prove Hardness

**Key Property #1:** OR property



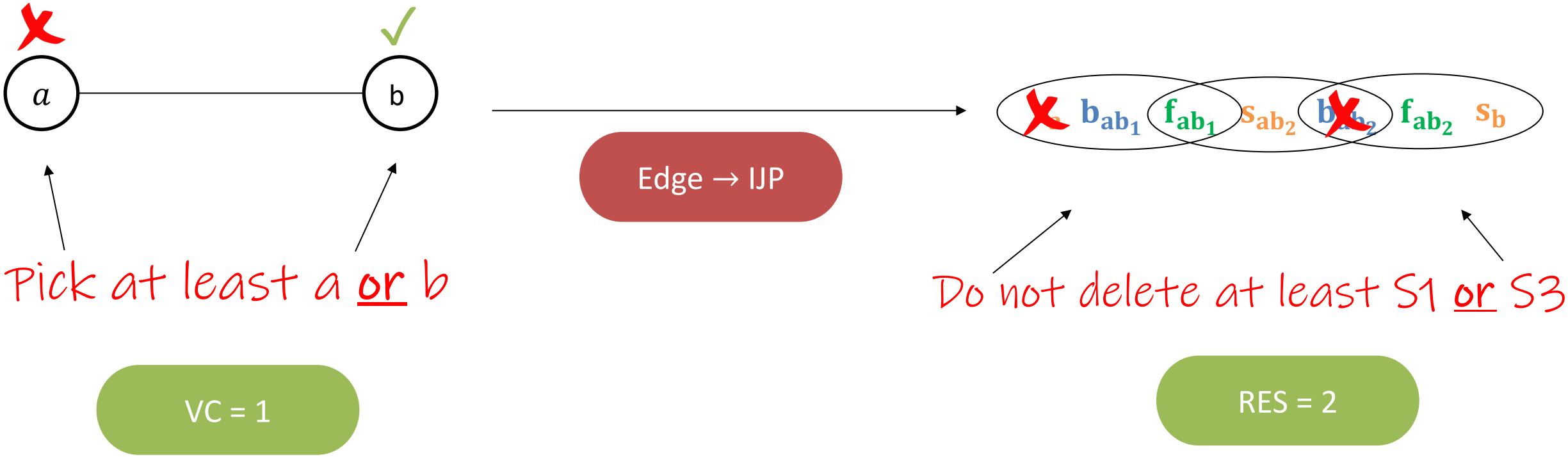
# Unified Approach to prove Hardness

**Key Property #1:** OR property



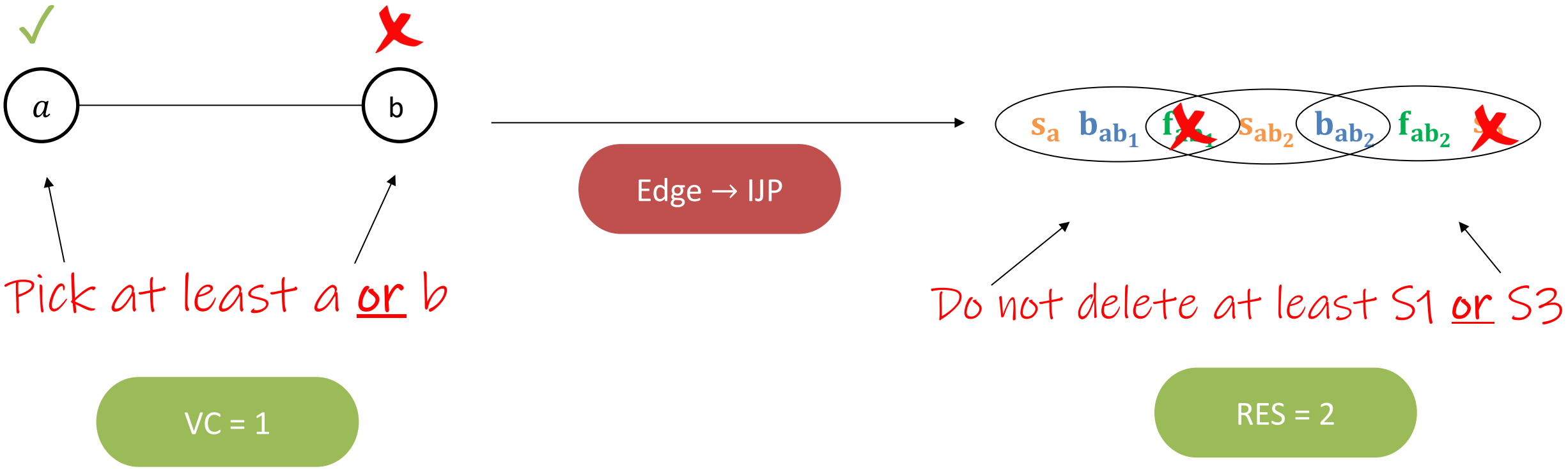
# Unified Approach to prove Hardness

**Key Property #1:** OR property



# Unified Approach to prove Hardness

**Key Property #1:** OR property



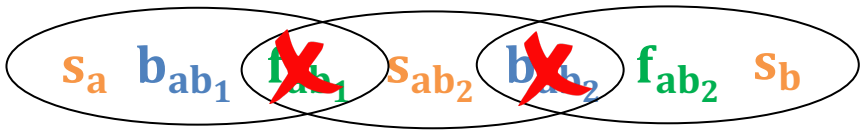
# Unified Approach to prove Hardness

**Key Property #1:** OR property


Pick at least a or b

VC = 2

Edge → IJP

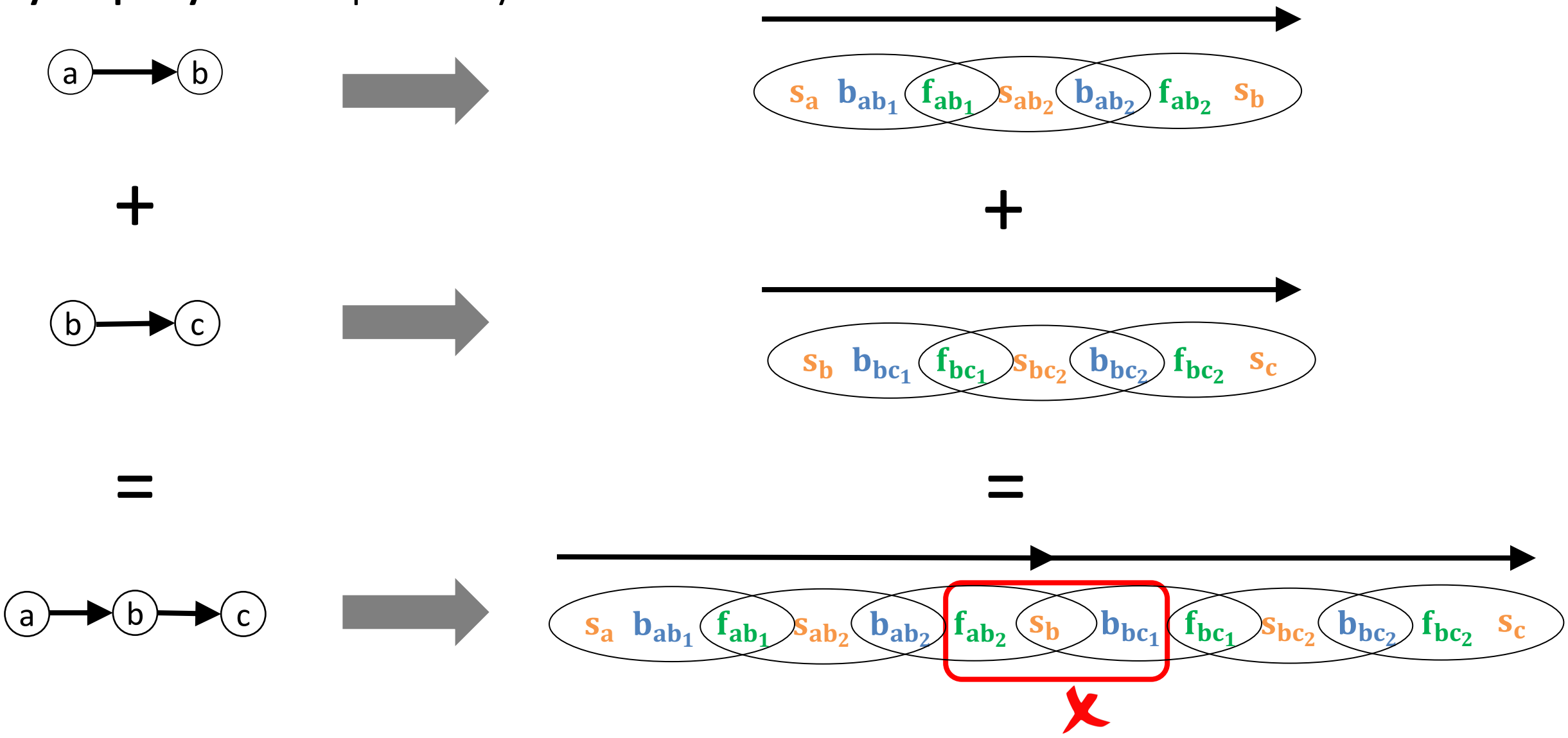



Do not delete at least S1 or S3

RES = 2

# Unified Approach to prove Hardness

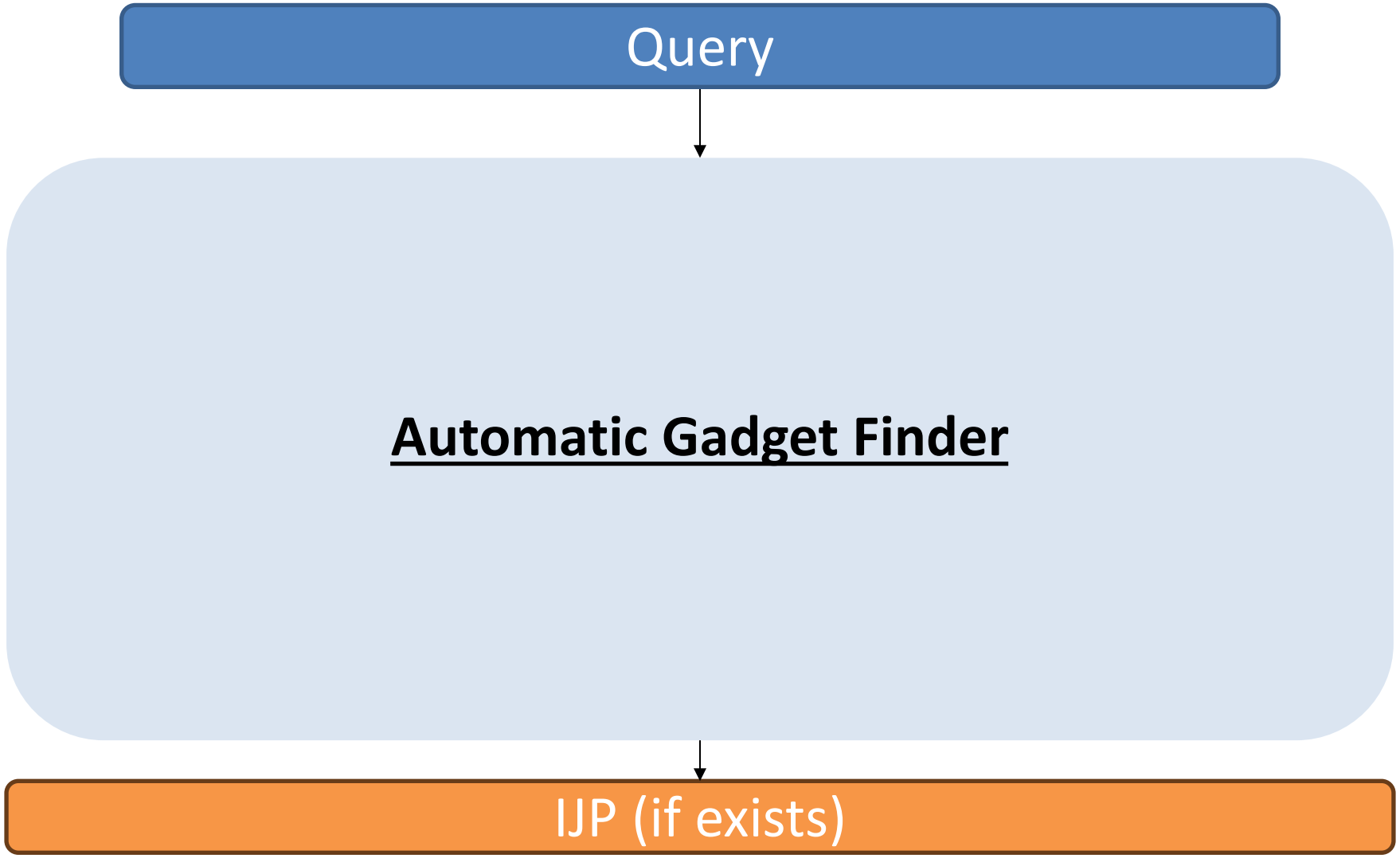
**Key Property #2:** Composability of IJPs




Composing two IJPs should not lead to additional witnesses

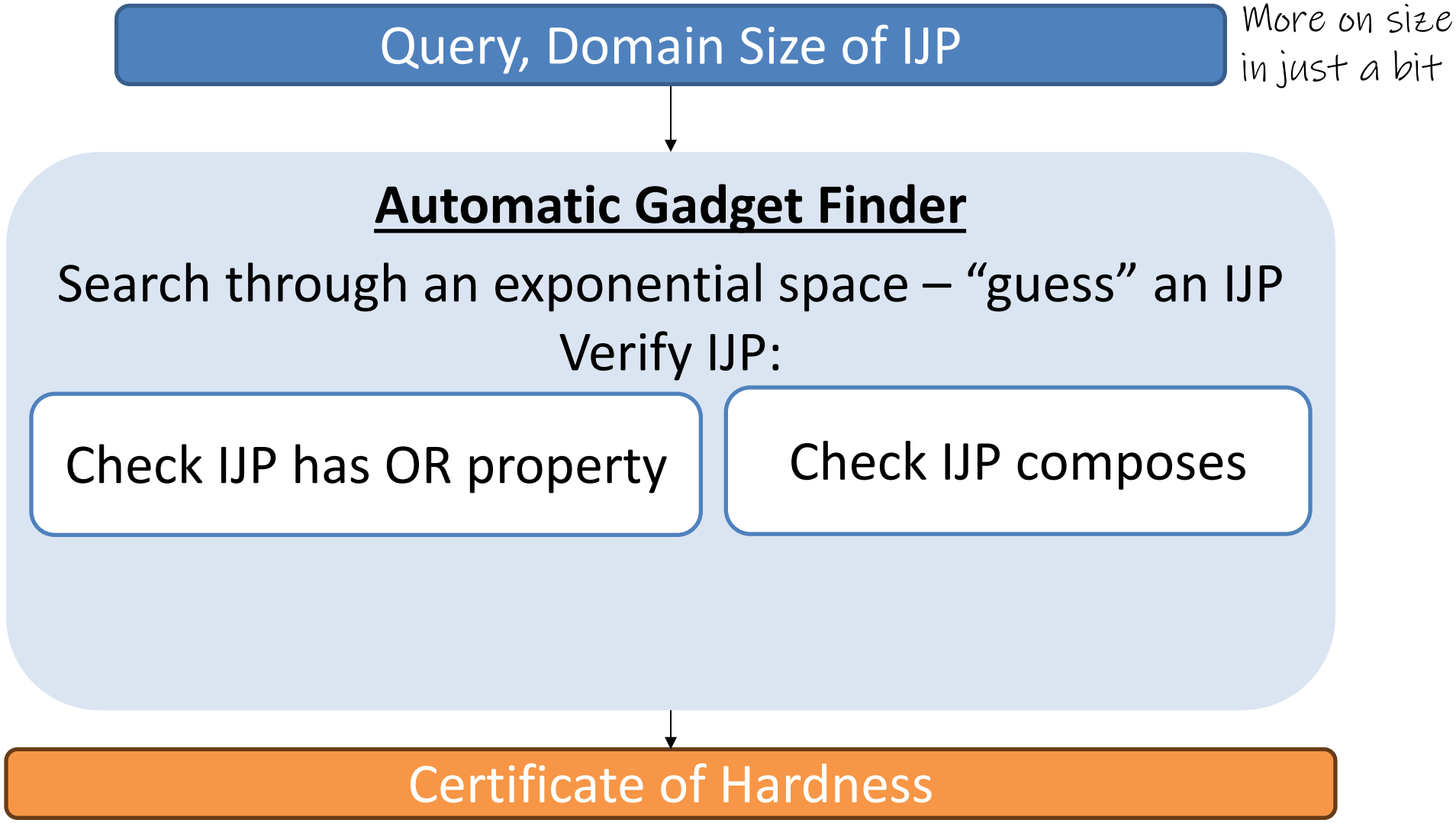
# Our Goal

Using the complete criterion for IJP, can we build a principled way to find IJPs?



# Unified Approach to prove Hardness

Using the complete criterion for IJP, we can build a principled way to find IJPs



# Unified Approach to prove Hardness

Using the complete criterion for IJP, we can build a principled way to find IJPs

Query, Domain Size of IJP

More on size in just a bit

**Automatic Gadget Finder**

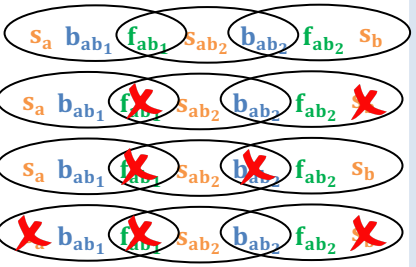
Search through an exponential space – "guess" an IJP

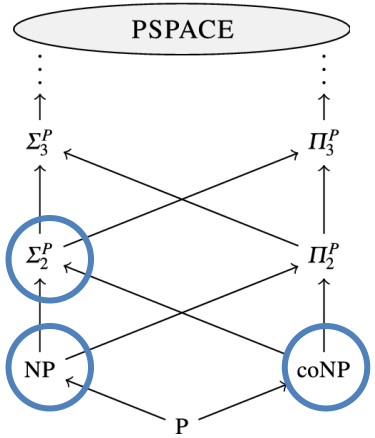
Verify IJP:

Check IJP has OR property

- Find size k resilience set ***(NP)***
- Find no size k-1 resilience set ***(Co-NP)***

Check IJP composes






Certificate of Hardness

# Unified Approach to prove Hardness

Using the complete criterion for IJP, we can build a principled way to find IJPs

Query, Domain Size of IJP

More on size in just a bit

**Automatic Gadget Finder**

Search through an exponential space – "guess" an IJP

Verify IJP:

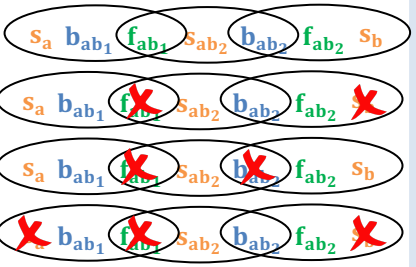
Check IJP has OR property

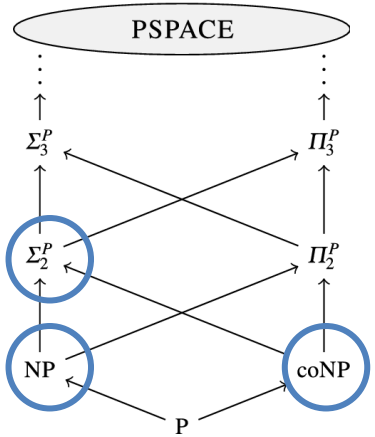
- Find size k resilience set ***(NP)***
- Find no size k-1 resilience set ***(Co-NP)***

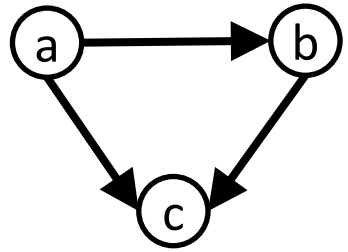
Check IJP composes

Theorem.

It suffices to check just 3 join paths compose




Certificate of Hardness