Updated 3/9/2022

Topic 2: Complexity of Query Evaluation
Unit 3: Provenance
Lecture 14

Wolfgang Gatterbauer

CS7240 Principles of scalable data management (sp22)

https://northeastern-datalab.github.io/cs7240/sp22/

3/4/2022

Topic 2: Complexity of Query Evaluation & Reverse Data Management

- **CONTINUED Lecture 11 (Tue 2/22):** 1 Conjunctive Queries
- **Lecture 12 (Fri 2/25):** Conjunctive Queries
- **Lecture 13 (Tue 3/1):** Beyond Conjunctive Queries
- **Lecture 14 (Fri 3/4):** Provenance
- **Lecture 15 (Tue 3/8):** Provenance, Reverse Data Management

Pointers to relevant concepts & supplementary material:

- **Unit 1. Conjunctive Queries**: Query evaluation of conjunctive queries (CQs), data vs. query complexity, homomorphisms, constraint satisfaction, query containment, query minimization, absorption: [Kolaitis, Vardi'00], [Vardi'00], [Kolaitis'16], [Koutris'19] L1 & L2
- **Unit 2. Beyond Conjunctive Queries**: unions of conjunctive queries, bag semantics, nested queries, tree pattern queries: [Kolaitis'16], [Tan+'14], [G.'11], [Martens'17]
- **Unit 3. Provenance**: [Buneman+02], [Green+07], [Cheney+09], [Green,Tannen'17], [Kepner+16], [Buneman, Tan'18]
- **Unit 4. Reverse Data Management**: update propagation, resilience: [Buneman+02], [Kimelfeld+12], [Freire+15]

Outline: T2-3/4: Provenance & Reverse Data Management

- T2-3: Provenance
  - Data Provenance
  - The Semiring Framework for Provenance
  - Algebra: Monoids and Semirings
  - Query-rewrite-insensitive provenance
- T2-4: Reverse Data Management
  - View Deletion Problem
  - Resilience & Causality

Data provenance.

Imagine a computational process that uses a complex input consisting of multiple items. The granularity and nature of "input item" can vary significantly. It can be a single tuple, a database table, or a whole database. It can a spreadsheet describing an experiment, a laboratory notebook entry, or another form of capturing annotation by humans in software. It can also be a file, or a storage system component. It can be a parameter used by a module in a scientific workflow. It can also be a configuration rule used in software-defined routing or in a complex network protocol. Or it can be a configuration decision made by a distributed computation scheduler (think map-reduce). *Provenance analysis* allows us to understand how these different input items affect the output of the computation. When done appropriately, such

Near-Term Challenges in II

II = Intelligent Infrastructure

- Error control for multiple decisions
- Systems that create markets
- Designing systems that can provide meaningful, calibrated notions of their uncertainty
- Achieving real-time performance goals
- Managing cloud-edge interactions
- Designing systems that can find abstractions quickly
- Provenance in systems that learn and predict
- Designing systems that can explain their decisions
- Finding causes and performing causal reasoning
- Systems that pursue long-term goals, and actively collect data in service of those goals
- Achieving fairness and diversity
- Robustness in the face of unexpected situations
- Robustness in the face of adversaries
- Sharing data among individuals and organizations
- Protecting privacy and issues of data ownership

Source: Michael I. Jordan: Machine Learning: Dynamical, Stochastic & Economic Perspectives, 2019: https://www.youtube.com/watch?v=-8yYFdV5SOc

Provenance: "Where Did this Data Come from?"

- Whenever data is shared (e.g., science, Web) natural questions appear:
  - How did I get this data?
  - What operations were used to create the data?
  - How much should I trust (believe) it?

- Provenance: describes the origins and history of data in its life cycle
- Two types of provenance
  - Provenance inside a database: that's our focus
  - Provenance outside databases: focus of ongoing research esp. in ML (causes, influence, fairness); less well-defined; there is a standard OPM (Open Provenance Model)
- There are also questions for our focus, provenance inside DBMS:
  - What is the "right data model" of provenance?
  - How do we query it? What operations should we support?

Based upon: Zachary Ives's talk "Querying Data Provenance", SIGMOD 2010. https://doi.org/10.1145/1807167.1807269

Example of data provenance

- A typical question:
  - For a given database D, a query Q, and a tuple t in the output of $Q(D)$, which parts of D "contribute" to output tuple t?

D

R

Emp	Dept
Alice	D01
Bob	D01
Charly	D02

S

Did	Mgr
D01	Dora
D02	Ema
D03	Felix

Q

```
SELECT  R.Emp, S.Mgr
FROM    R, S
WHERE   R.Dept=S.Did
```

Q

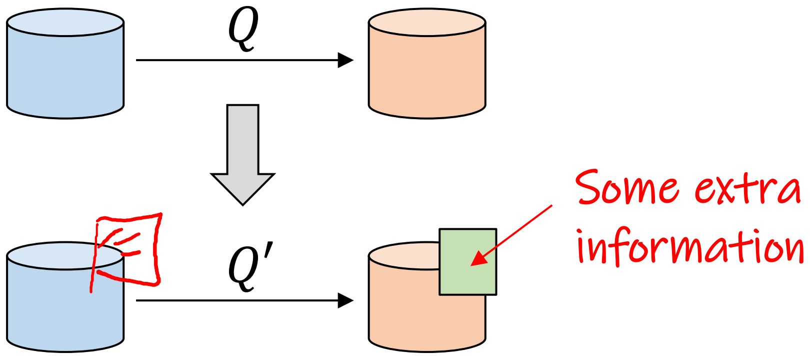
Emp	Mgr
Alice	Dora
Bob	Dora
Charly	Ema

t

  - The question can be applied to attribute values, tables, rows, etc.

Two approaches

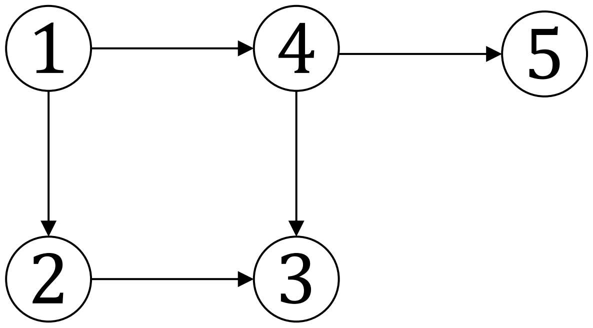
- Eager or annotation-based ("annotation propagation")
  - Changes the transformation from Q to Q' to carry extra information
  - Full source data not needed after transformation



- Lazy or non-annotation based
  - Q is unchanged
  - Recomputation and access to source required.
    - Good when extra storage is an issue.

Conceptual distinction from: Cheney, Chiticariu, Tan. "Provenance in databases -- why, how, and where", 2009. https://doi.org/10.1561/1900000006

Example graph problem, in 5 different variants



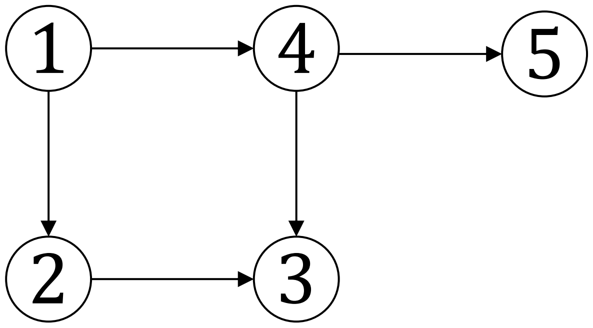
E

From	to
1	2
2	3
1	4
4	3
4	5

Q(z) :- E(1,y), E(y,z) → ?

Q: Points reachable in 2 hops, starting at node "1"

Example graph problem, in 5 different variants



E

1	2
2	3
1	4
4	3
4	5

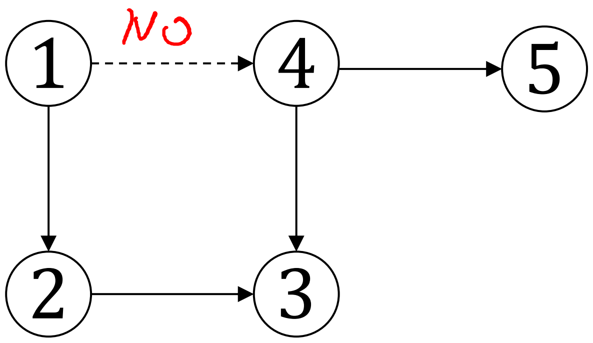
Q(z) :- E(1,y), E(y,z) →

Q

3
5

Q: Points reachable in 2 hops, starting at node "1"

Example variant 1



Now assume only certain edges are available (available yes/no or true/false). Which of the points remain reachable?

E

1	2	yes
2	3	yes
1	4	no
4	3	yes
4	5	yes

Q(z) :- E(1,y), E(y,z) →

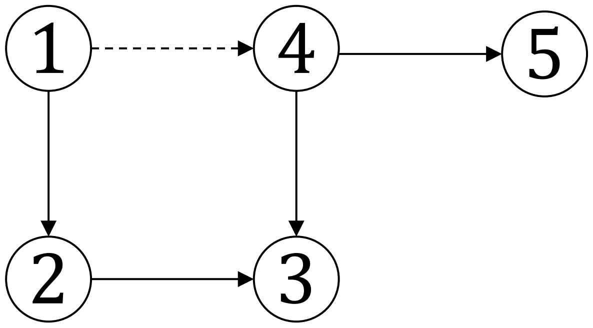
Q

3
5

?

Q: Points reachable in 2 hops, starting at node "1"

Example variant 1



Now assume only certain edges are available (available yes/no or true/false). Which of the points remain reachable?

E

1	2	yes
2	3	yes
1	4	no
4	3	yes
4	5	yes

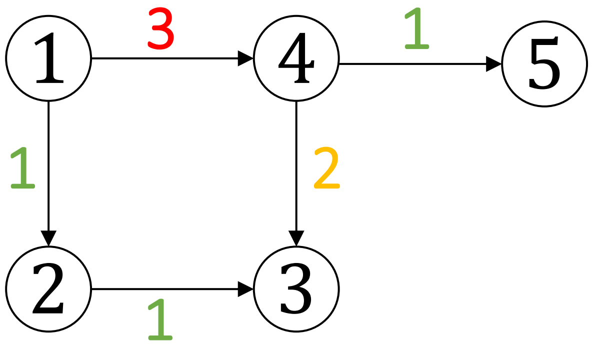
Q(z) :- E(1,y), E(y,z) →

Q: Points reachable in 2 hops, starting at node "1"

Q

3	yes
5	no

Example variant 2



Now assume passing along an edge needs a certain security clearance (1<2<3). What clearance do you need for reaching each point?

E

1	2	1
2	3	1
1	4	3
4	3	2
4	5	1

Q(z) :- E(1,y), E(y,z)

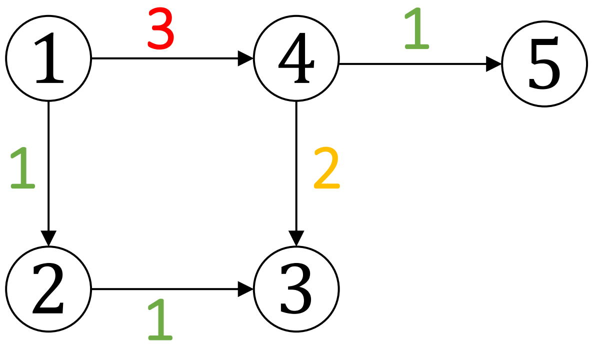
Q

3
5

?

Q: Points reachable in 2 hops, starting at node "1"

Example variant 2



Now assume passing along an edge needs a certain security clearance (1<2<3).
What clearance do you need for reaching each point?

E

1	2	1
2	3	1
1	4	3
4	3	2
4	5	1

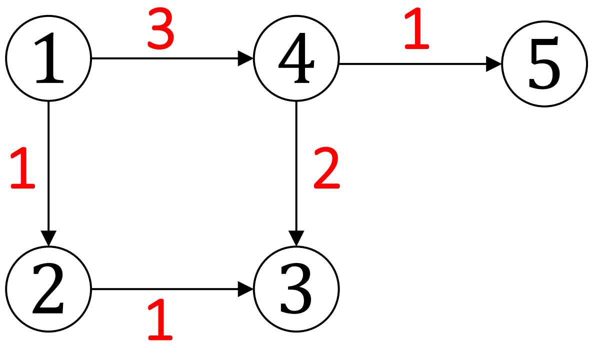
Q(z) :- E(1,y), E(y,z)

Q

3	1
5	3

Q: Points reachable in 2 hops, starting at node "1"

Example variant 3



Now assume each edge has a weight.
What is the shortest path to reach each point?

E

1	2	1
2	3	1
1	4	3
4	3	2
4	5	1

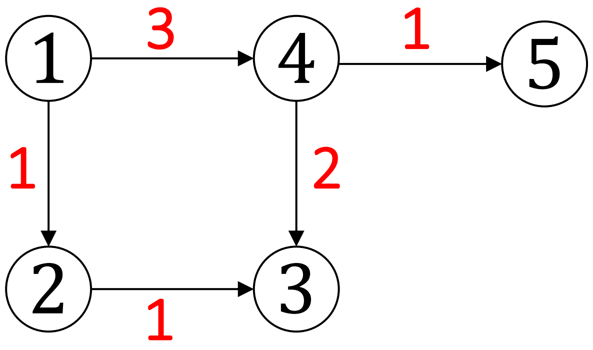
Q(z) :- E(1,y), E(y,z) → Q

Q
3
5

?

Q: Points reachable in 2 hops, starting at node "1"

Example variant 3



Now assume each edge has a weight.
What is the shortest path to reach each point?

E

1	2	1
2	3	1
1	4	3
4	3	2
4	5	1

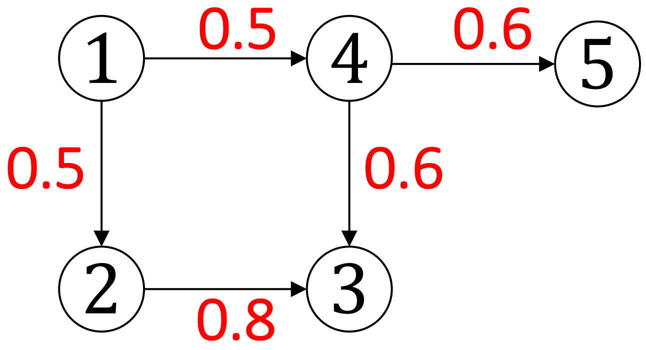
Q(z) :- E(1,y), E(y,z) →

Q

3	2
5	4

Q: Points reachable in 2 hops, starting at node "1"

Example variant 4



Now assume each edge has a confidence (probability of being available).
What is the probability of the most likely path?

E

1	2	0.5
2	3	0.8
1	4	0.5
4	3	0.6
4	5	0.6

Q(z) :- E(1,y), E(y,z) →

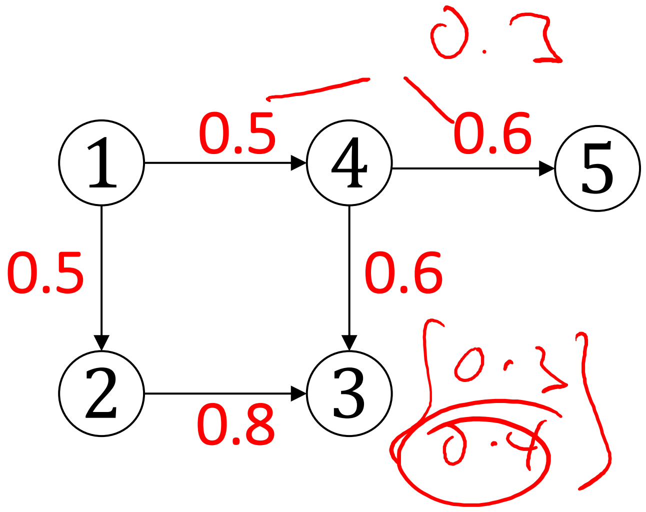
Q

3
5

?

Q: Points reachable in 2 hops, starting at node "1"

Example variant 4



Now assume each edge has a confidence (probability of being available).
What is the probability of the most likely path?

E

1	2	0.5
2	3	0.8
1	4	0.5
4	3	0.6
4	5	0.6

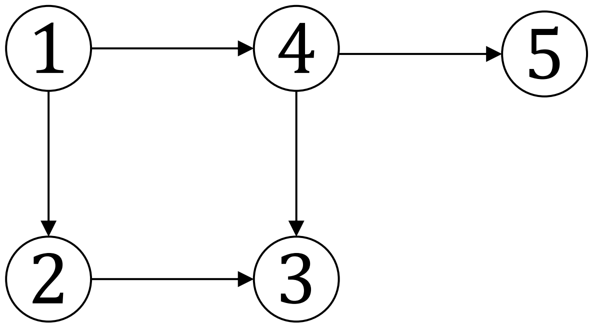
Q(z) :- E(1,y), E(y,z) →

Q

3	0.4
5	0.3

Q: Points reachable in 2 hops, starting at node "1"

Example variant 5



Finally assume we want to calculate the number of paths to a node. How many are there? What is even a reasonable way to calculate that in general?

E

1	2
2	3
1	4
4	3
4	5

Q(z) :- E(1,y), E(y,z) →

Q

3	2
5	1

Q: Points reachable in 2 hops, starting at node "1"

Outline: T2-3/4: Provenance & Reverse Data Management

- T2-3: Provenance
  - Data Provenance
  - The Semiring Framework for Provenance
  - Algebra: Monoids and Semirings
  - Query-rewrite-insensitive provenance
- T2-4: Reverse Data Management
  - View Deletion Problem
  - Resilience & Causality

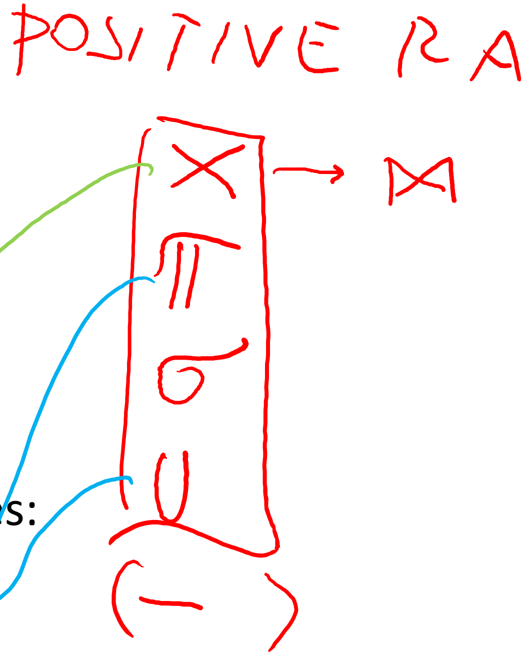
Do it once and use it repeatedly: provenance

Label (annotate) input items abstractly with **provenance tokens.**

Provenance tracking: propagate **expressions** (involving tokens)
(to annotate intermediate data and, finally, outputs)

Track two distinct ways of using data items by computation primitives:

- **jointly** (this alone is basically like keeping a log)
- **alternatively** (doing both is essential; think trust)



Input-output compositional; Modular (in the primitives)

Later, we want to **evaluate** the provenance expressions to obtain

binary trust, access control,
confidence scores, data prices, etc.

Algebraic interpretation for RDB

Set X of provenance tokens.

Space of annotations, provenance expressions Prov(X)

Prov(X)-relations:

every tuple is annotated with some element from Prov(X).

Binary operations on Prov(X):

- $\cdot$ corresponds to joint use (join, cartesian product),
- $+$ corresponds to alternative use (union and projection).

Special annotations:

"Absent" tuples are annotated with 0.

1 is a "neutral" annotation (data we do not track).

Source: Val Tannen. "The Semiring Framework for Database Provenance", PODS 2017 Test of Time Award talk : https://www.cis.upenn.edu/~val/15MayPODS.pdf
Wolfgang Gatterbauer. Principles of scalable data management: https://northeastern-datalab.github.io/cs7240/

K-Relational algebra

Algebraic laws of $(\text{Prov}(X), +, \cdot, 0,1)$? More generally, for annotations from a structure $(K, +, \cdot, 0,1)$?

K-relations. Generalize RA+ to (positive) **K-relational algebra.**

Desired optimization equivalences of K- relational algebra iff $(K, +, \cdot, 0,1)$ is a **commutative semiring.**

Generalizes SPJU or UCQ or non-rec. Datalog

set semantics $(\mathbb{B}, \vee, \wedge, \bot, \top)$ bag semantics $(\mathbb{N}, +, \cdot, 0, 1)$

c-table-semantics [IL84] $(\text{BoolExp}(X), \vee, \wedge, \bot, \top)$

event table semantics [FR97,Z97] $(\mathcal{P}(\Omega), \cup, \cap, \emptyset, \Omega)$

Source: Val Tannen. "The Semiring Framework for Database Provenance", PODS 2017 Test of Time Award talk : https://www.cis.upenn.edu/~val/15MayPODS.pdf
Wolfgang Gatterbauer. Principles of scalable data management: https://northeastern-datalab.github.io/cs7240/

$\{x, y\}$ $\{x+y, x \cdot y, x \cdot y \cdot y + x, \ldots\}$

What is a commutative semiring?

An algebraic structure $(K, +, \cdot, 0, 1)$ where:

- K is the domain
- $+$ is associative, commutative, with 0 identity
- $\cdot$ is associative, with 1 identity
- $\cdot$ distributes over $+$
- $a \cdot 0 = 0 \cdot a = 0$

} **semiring**

- $\cdot$ is also **commutative**

Unlike ring, no requirement for inverses to $+$

Source: Val Tannen. "The Semiring Framework for Database Provenance", PODS 2017 Test of Time Award talk : https://www.cis.upenn.edu/~val/15MayPODS.pdf
Wolfgang Gatterbauer. Principles of scalable data management: https://northeastern-datalab.github.io/cs7240/

Provenance polynomials

$\mathbb{N}[\{x, y\}] = \{xy, x + y, 2xy^2 + x, 2xy^2 + xy + x, \dots\}$

$(\mathbb{N}[X], +, \cdot, 0, 1)$ is the commutative semiring **freely generated** by X
(universality property involving homomorphisms)

Provenance polynomials are **PTIME**-computable (data complexity).
(query complexity depends on language and representation)
ORCHESTRA provenance (graph representation) about **30%** overhead

Monomials correspond to **logical derivations** (proof trees in non-rec. Datalog)

Provenance reading of polynomails:

output tuple has provenance $2r^2 + rs$

three derivations of the tuple
- two of them use r, twice,
- the third uses r and s, once each

Source: Val Tannen. "The Semiring Framework for Database Provenance", PODS 2017 Test of Time Award talk : https://www.cis.upenn.edu/~val/15MayPODS.pdf
Wolfgang Gatterbauer. Principles of scalable data management: https://northeastern-datalab.github.io/cs7240/

Two kinds of semirings in this framework

Provenance semirings, e.g.,

$(\mathbb{N}[X], +, \cdot, 0, 1)$ provenance polynomials [GKT07]

$(\mathrm{Why}(X), \cup, \Cup, \emptyset, \{\emptyset\})$ witness why-provenance [BKT01]

Application semirings, e.g.,

$(\mathbb{A}, \min, \max, 0, \mathrm{Pub})$ access control [FGT08]

$\mathbb{V} = ([0,1], \max, \cdot, 0, 1)$ Viterbi semiring (MPE) [GKIT07]

Provenance specialization relies on

- Provenance semirings are freely generated by provenance tokens
- Query commutation with semiring homomorphisms

Source: Val Tannen. "The Semiring Framework for Database Provenance", PODS 2017 Test of Time Award talk : https://www.cis.upenn.edu/~val/15MayPODS.pdf
Wolfgang Gatterbauer. Principles of scalable data management: https://northeastern-datalab.github.io/cs7240/

Some application semirings

$(\mathbb{B}, \wedge, \vee, \top, \bot)$ *binary trust*

$(\mathbb{N}, +, \cdot, 0, 1)$ *multiplicity (number of derivations)*

$(\mathbb{A}, \min, \max, 0, Pub)$ *access control*

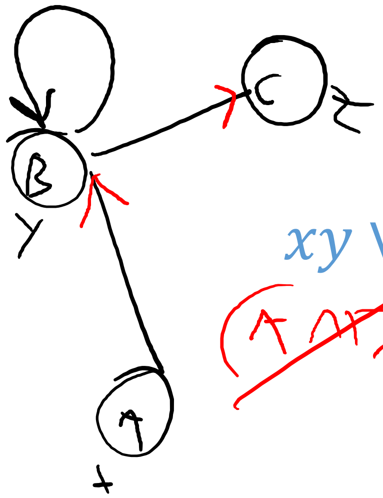
$\mathbb{V} = ([0,1], \max, \cdot, 0, 1)$ Viterbi semiring (MPE) *confidence scores*

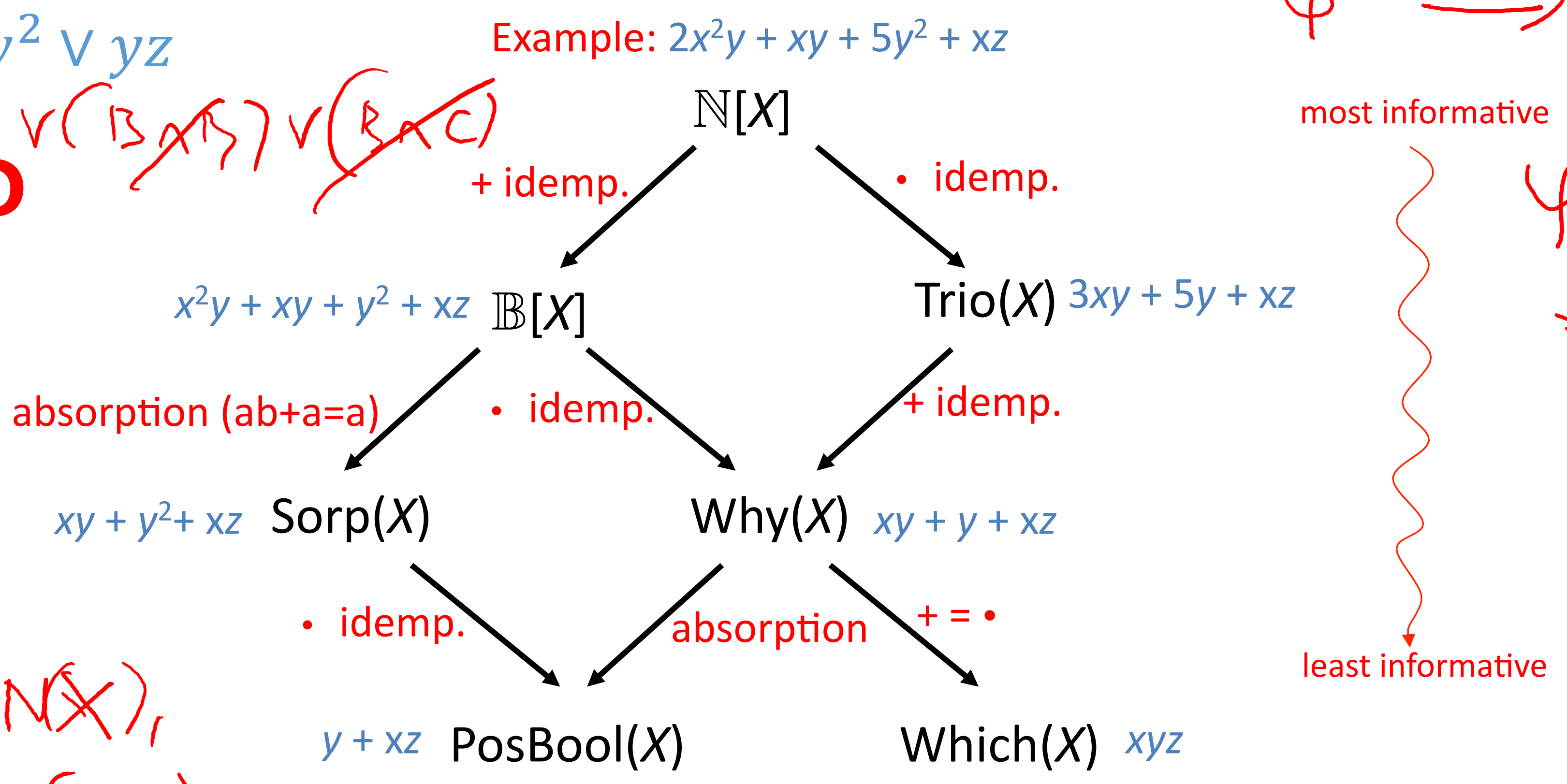
$\mathbb{T} = ([0, \infty], \min, +, \infty, 0)$
tropical semiring (shortest paths) *data pricing*

$\mathbb{F} = ([0,1], \max, \min, 0, 1)$ "fuzzy logic" semiring

Source: Val Tannen. "The Semiring Framework for Database Provenance", PODS 2017 Test of Time Award talk : https://www.cis.upenn.edu/~val/15MayPODS.pdf
Wolfgang Gatterbauer. Principles of scalable data management: https://northeastern-datalab.github.io/cs7240/

A Hierarchy of Provenance Semirings [G09, DMRT14]



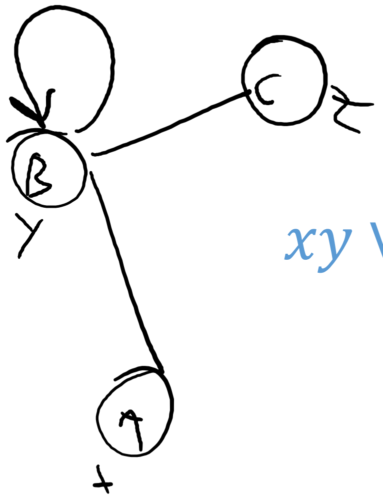


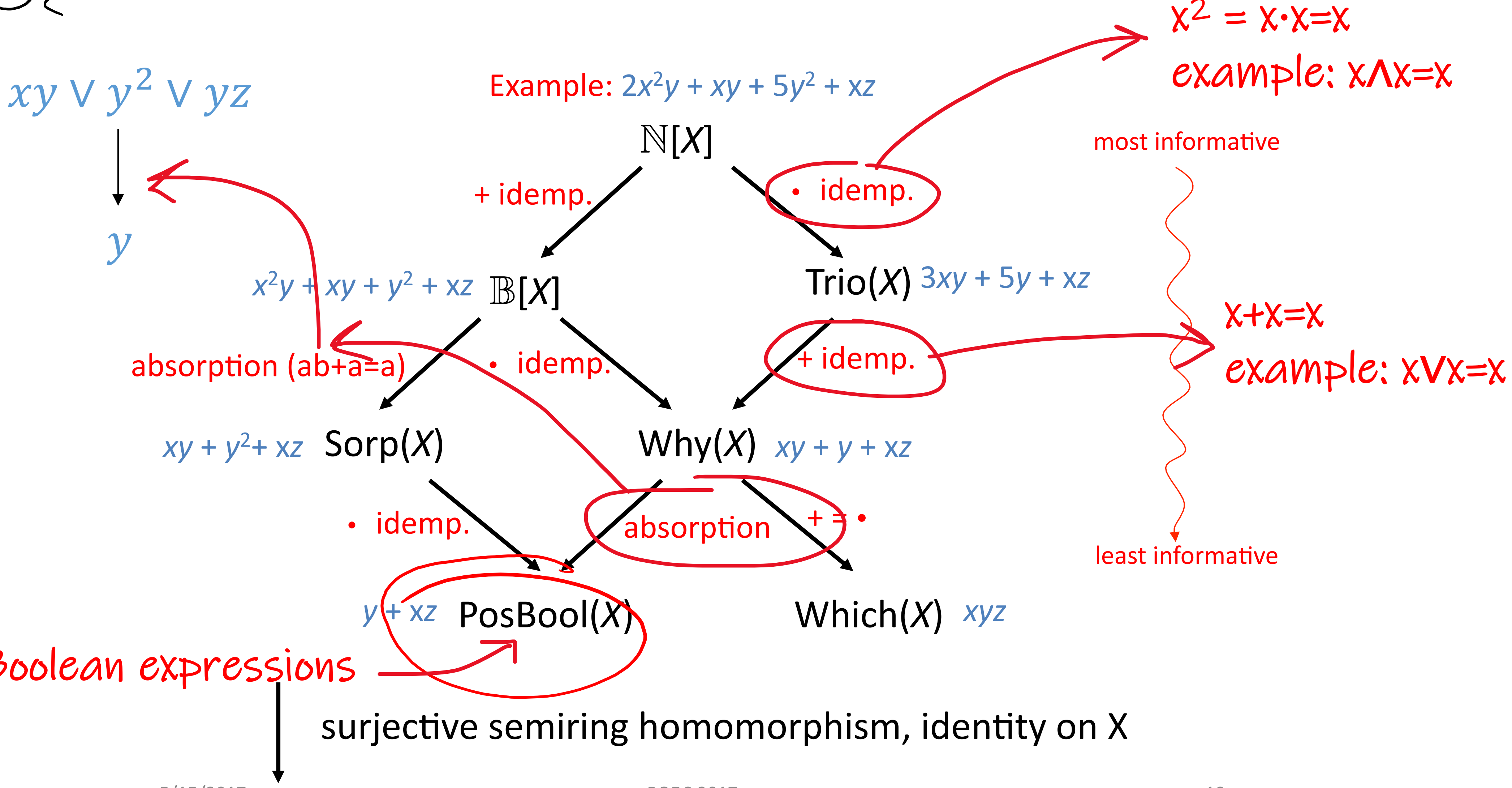
surjective semiring homomorphism, identity on X

Source: Val Tannen. "The Semiring Framework for Database Provenance", PODS 2017 Test of Time Award talk : https://www.cis.upenn.edu/~val/15MayPODS.pdf

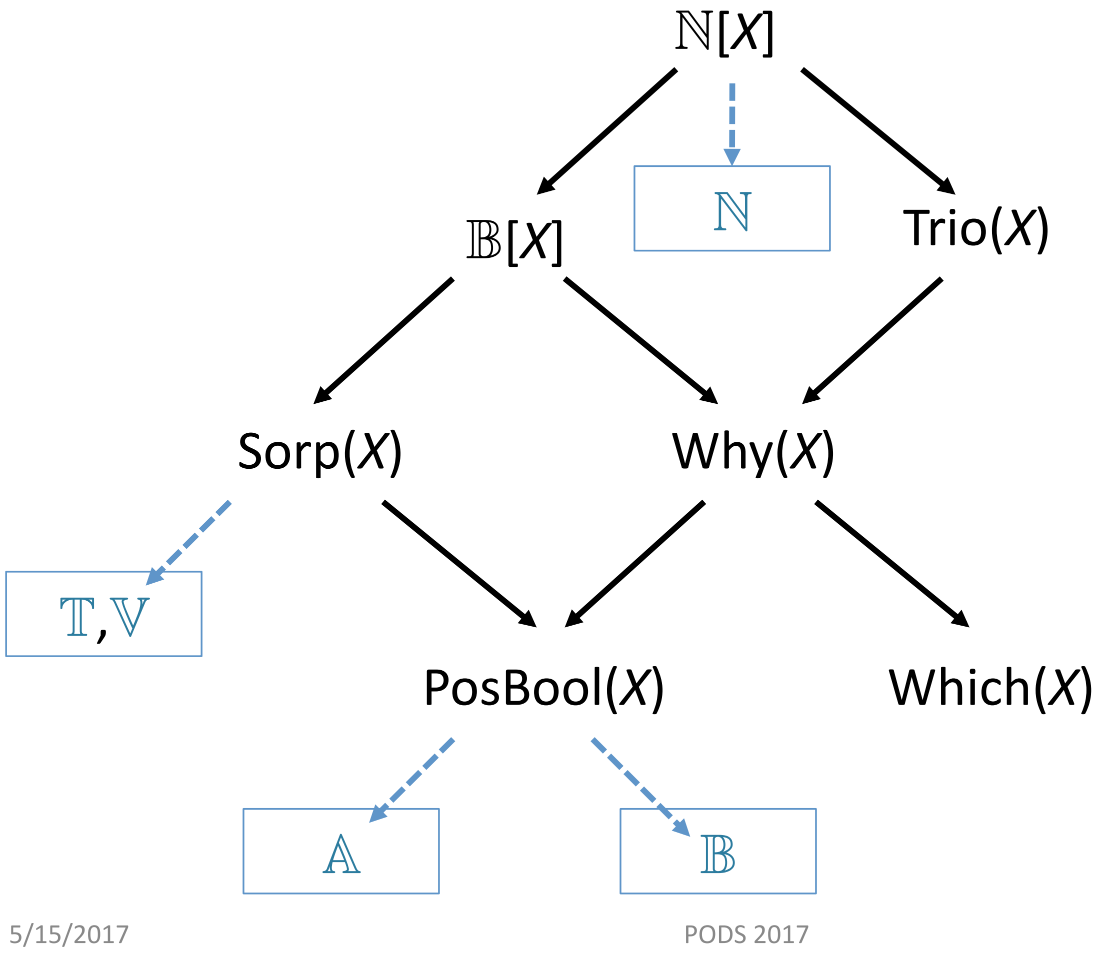
Wolfgang Gatterbauer. Principles of scalable data management: https://northeastern-datalab.github.io/cs7240/

A Hierarchy of Provenance Semirings [G09, DMRT14]





A Hierarchy of Provenance Semirings [G09, DMRT14]



A menagerie of provenance semirings

(Which(X), $\cup$, $\cup^*$, $\emptyset$, $\emptyset^*$) sets of contributing tuples "Lineage" (1) [CWW00]

(Why(X), $\cup$, $\Cup$, $\emptyset$, $\{\emptyset\}$) sets of sets of … Witness why-provenance [BKT01]

(PosBool(X), $\wedge$, $\vee$, $\top$, $\bot$) minimal sets of sets of… Minimal witness why-provenance [BKT01] also "Lineage" (2) used in probabilistic dbs [SORK11]

(Trio(X), +, $\cdot$, 0, 1) bags of sets of … "Lineage" (3) [BDHT08,G09]

($\mathbb{B}[X]$,+, $\cdot$, 0, 1) sets of bags of … Boolean coeff. polynomials [G09]

(Sorp(X),+, $\cdot$, 0, 1) minimal sets of bags of … absorptive polynomials [DMRT14]

($\mathbb{N}[X]$, +, $\cdot$, 0, 1) bags of bags of… universal provenance polynomials [GKT07]

Source: Val Tannen. "The Semiring Framework for Database Provenance", PODS 2017 Test of Time Award talk : https://www.cis.upenn.edu/~val/15MayPODS.pdf

Positive relational algebra: Join ⋈

⋈

R

A	B
1	1
2	1
2	2

S

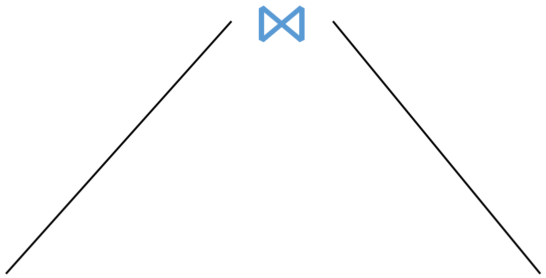
B	C
1	1
2	2
2	3

Q=R⋈S

A	B	C

?

Positive relational algebra: Join ⋈



R

A	B
1	1
2	1
2	2

S

B	C
1	1
2	2
2	3

Q=R⋈S

A	B	C
1	1	1
2	1	1
2	2	2
2	2	3

Positive relational algebra: Join ⋈

⋈

R

A	B	
1	1	r_1
2	1	r_2
2	2	r_3

S

B	C	
1	1	s_1
2	2	s_2
2	3	s_3

Q=R⋈S

A	B	C
1	1	1
2	1	1
2	2	2
2	2	3

?

Positive relational algebra: Join ⋈

The annotation "r · s" means <u>joint use</u> of data annotated by r and data annotated by s

⋈

R

A	B	
1	1	r_1
2	1	r_2
2	2	r_3

S

B	C	
1	1	s_1
2	2	s_2
2	3	s_3

Q=R⋈S

A	B	C	
1	1	1	$r_1 \cdot s_1$
2	1	1	$r_2 \cdot s_1$
2	2	2	$r_3 \cdot s_2$
2	2	3	$r_3 \cdot s_3$

Positive relational algebra: Projection π

$\pi_{\text{-B}}$

R

A	B	
1	1	r_1
1	2	r_2
2	1	r_3
2	2	r_4
2	3	r_5

$Q=\pi_{\text{-B}}R=\pi_{A}R$

A

?

Positive relational algebra: Projection π

$\pi_{\text{-B}}$

R

A	B	
1	1	r_1
1	2	r_2
2	1	r_3
2	2	r_4
2	3	r_5

$Q=\pi_{\text{-B}}R=\pi_A R$

A
1
2

?

Positive relational algebra: Projection π

The annotation "r + s" means <u>alternative use</u> of data

π_{-B}

R

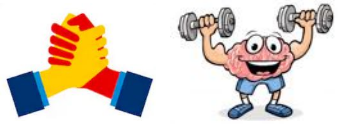
A	B	
1	1	r_1
1	2	r_2
2	1	r_3
2	2	r_4
2	3	r_5

$Q=\pi_{-B}R=\pi_A R$

A	
1	$r_1 + r_2$
2	$r_3 + r_4 + r_5$

$r_3 \vee r_4 \vee r_5$

Positive relational algebra: Union ∪



{(2,1), (2,1)} = (2,1) ↦ 2

∪

R

A	B	
1	1	r_1
2	1	r_2

S

A	B	
2	1	s_1
2	2	s_2

Q=R∪S

A	B

?

Positive relational algebra: Union ∪

The annotation "r + s" means <u>alternative use</u> of data

∪ (R, S)

R

A	B	
1	1	r_1
2	1	r_2

S

A	B	
2	1	s_1
2	2	s_2

Q=R∪S

A	B	
1	1	r_1
2	1	$r_2 + s_1$
2	2	s_2

$$R \cup S = \pi_{AB}(R \cup_{BAG} S)$$

Positive relational algebra: Selection σ

$\sigma_{A=1}$

R

A	B	
1	1	r_1
1	2	r_2
2	1	r_3
2	2	r_4
2	3	r_5

$Q=\sigma_{A=1}R$

A	B

?

Positive relational algebra: Selection σ

Two options for filtering:

1. Remove the tuples filtered out.

$\sigma_{A=1}$

|

R

A	B	
1	1	r_1
1	2	r_2
2	1	r_3
2	2	r_4
2	3	r_5

$Q = \sigma_{A=1} R$

A	B	
1	1	r_1
1	2	r_2

Positive relational algebra: Selection σ

Two options for filtering:

1. Remove the tuples filtered out.
2. Or keep them around ...

$\sigma_{A=1}$

R

A	B	
1	1	r_1
1	2	r_2
2	1	r_3
2	2	r_4
2	3	r_5

$Q=\sigma_{A=1}R$

A	B	
1	1	$r_1 \cdot 1$
1	2	$r_2 \cdot 1$
2	1	$r_3 \cdot 0$
2	2	$r_4 \cdot 0$
2	3	$r_5 \cdot 0$