Updated 3/16/2024

Topic 2: Complexity of Query Evaluation
Unit 3: Provenance
Lecture 16

Wolfgang Gatterbauer

CS7240 Principles of scalable data management (sp24)

https://northeastern-datalab.github.io/cs7240/sp24/

3/15/2024

Pre-class conversations

- Last class summary
- Projects: TUE 3/26 intermediate report
- Faculty candidate next week WED 3/20

- Today:
  - a comment on multitasking
  - provenance, semirings

A quizz

Which of the following lowers your measured IQ the most:

A. Smoking marijuana before taking test.

B. Responding to email/texting while taking test.

C. Losing a nights sleep before taking test.

A quizz

Which of the following lowers your measured IQ the most:

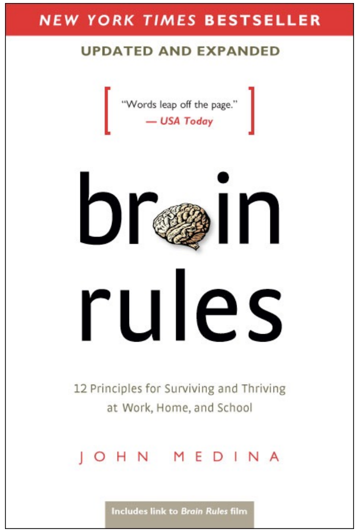
A. Smoking marijuana before taking test.
B. Responding to email/texting while taking test.
C. Losing a nights sleep before taking test.

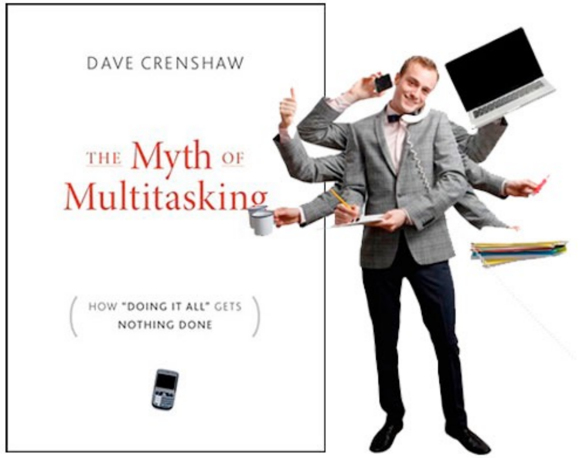
Answer: B

- You suck at multitasking!
- Everyone sucks at multitasking

Source: Courtesy of Michael D Smith (https://mds.heinz.cmu.edu/), http://news.bbc.co.uk/2/hi/uk_news/4471607.stm
(It is a bit of an over-simplification. Clarifications by the original author are here: http://www.drglennwilson.com/Infomania_experiment_for_HP.doc)

Multitasking





"Myth #3: Multitasking when it comes to paying attention, is a myth… studies show that a person who is interrupted takes 50% longer to accomplish a task. Not only that, he or she makes up to 50% more errors" -- John Medina (Brain rules)

"…multitasking is a lie. You're asking me to switch attention, and that makes me less productive." -- Dave Crenshaw (The myth of multitasking)

"multitasking adversely affects how you learn. Even if you learn while multitasking, that learning is less flexible and more specialized, so you cannot retrieve the information as easily." --Russell Poldrack, UCLA Psychology Professor

"Our research offers neurological evidence that the brain cannot effectively do two things at once." -- Rene Marois, Dept. of Psychology, Vanderbilt

"The brain is a lot like a computer. You may have several screens open on your desktop, but you're able to think about only one at a time." -- William Stixrud, Neuropsychologist

Topic 2: Complexity of Query Evaluation & Reverse Data Management

- **Lecture 14 (Fri 3/1):** T2-U1 Conjunctive Queries
- Spring break (Tue 3/5, Fri 3/8)
- **Lecture 15 (Tue 3/12):** T2-U1/2 Conjunctive & Beyond Conjunctive Queries
- **Lecture 16 (Fri 3/15):** T2-U1/2 Conjunctive & Beyond Conjunctive Queries
- **Lecture 17 (Tue 3/19):** T2-U3 Provenance
- **Lecture 18 (Fri 3/22):** T2-U3 Provenance
- **Lecture 20 (Tue 3/26):** T2-U4 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], [Gatterbauer'11], [Martens'17]
- **Unit 3. Provenance**: [Buneman+'02], [Green+'07], [Cheney+'09], [Green,Tannen'17], [Kepner+16], [Buneman, Tan'18], [Simons'23], [Dagstuhl'24]
- **Unit 4. Reverse Data Management**: update propagation, resilience: [Buneman+'02], [Kimelfeld+'12], [Freire+'15], [Makhija+'24]

Outline: T2-3: Provenance

- T2-3: Provenance
  - Data Provenance
  - The Semiring Framework for Provenance
  - Algebra: Monoids and Semirings
  - Query-rewrite-insensitive provenance

Data provenance.

~ Explanations

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?

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	Manager
D01	Dora
D02	Ema
D03	Felix

Q

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

Q

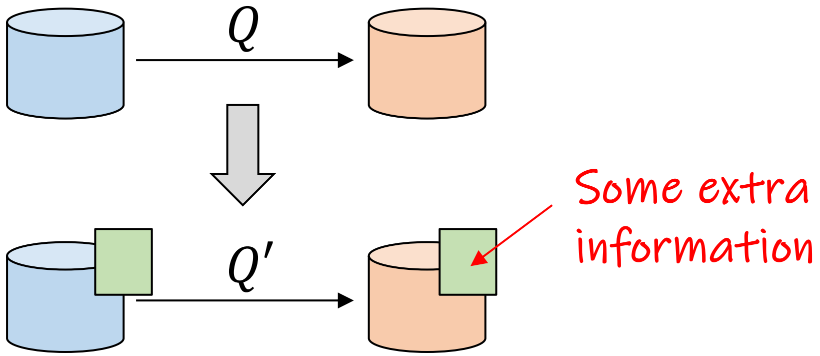
Emp	Manager
Alice	Dora
Bob	Dora
Charly	Ema

t

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

Two approaches

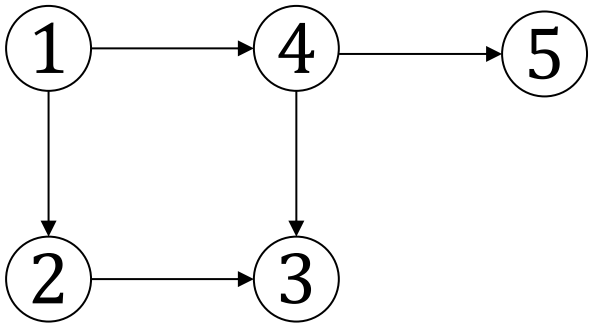
- Eager or annotation-based ("annotation propagation")
  - Changes the transformation from Q to Q' to carry extra information
  - Full source data not needed after transformation



- 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



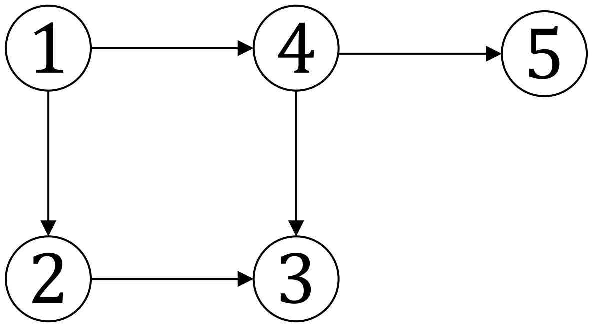
E

from	to
1	2
2	3
1	4
4	3
4	5

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

Q: ?

Example graph problem, in 5 different variants



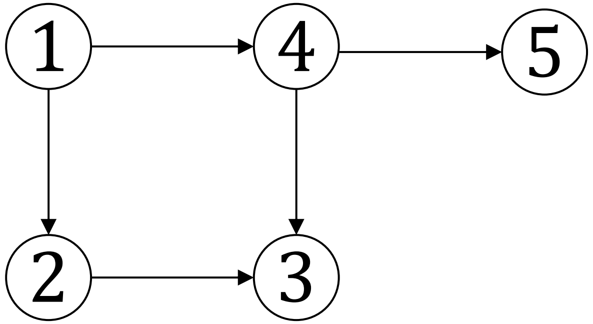
E

from	to
1	2
2	3
1	4
4	3
4	5

$\boxed{Q(z) \text{ :- } E(1,y), E(y,z)}$ $\longrightarrow$?

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

Example graph problem, in 5 different variants



E

from	to
1	2
2	3
1	4
4	3
4	5

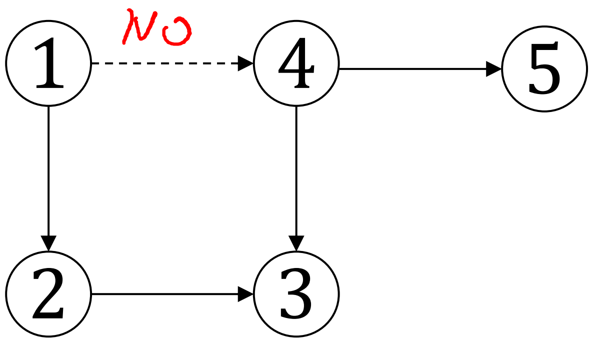
Q(z) :- E(1,y), E(y,z) →

Q

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

from	to	
1	2	yes
2	3	yes
1	4	no
4	3	yes
4	5	yes

Q(z) :- E(1,y), E(y,z) →

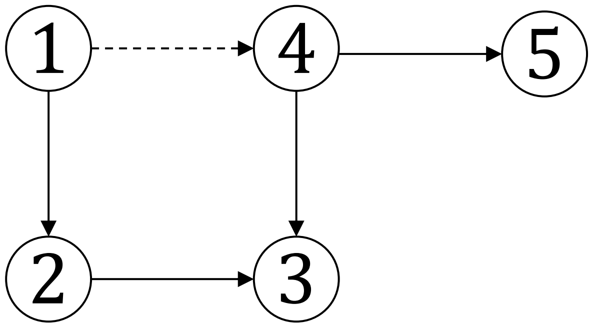
Q

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

from	to	
1	2	yes
2	3	yes
1	4	no
4	3	yes
4	5	yes

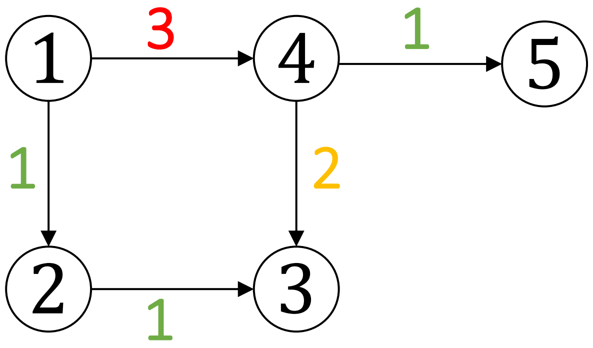
Q(z) :- E(1,y), E(y,z) →

Q

3	yes
5	no

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

from	to	
1	2	1
2	3	1
1	4	3
4	3	2
4	5	1

Q(z) :- E(1,y), E(y,z) $\longrightarrow$

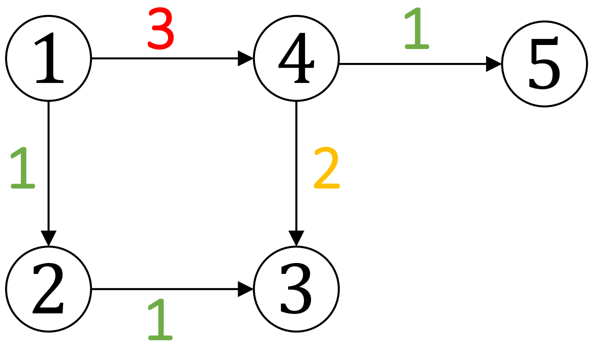
Q

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

from	to	
1	2	1
2	3	1
1	4	3
4	3	2
4	5	1

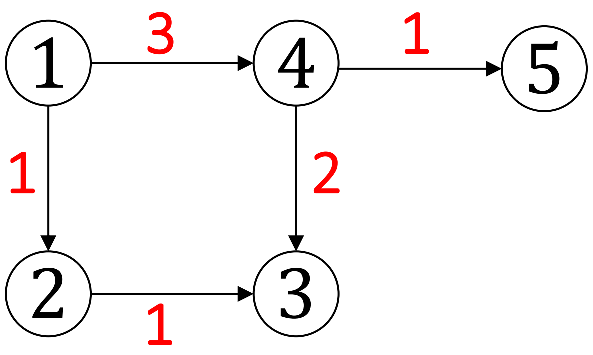
$Q(z)$:- $E(1,y)$, $E(y,z)$ $\longrightarrow$

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

from	to	
1	2	1
2	3	1
1	4	3
4	3	2
4	5	1

Q(z) :- E(1,y), E(y,z) →

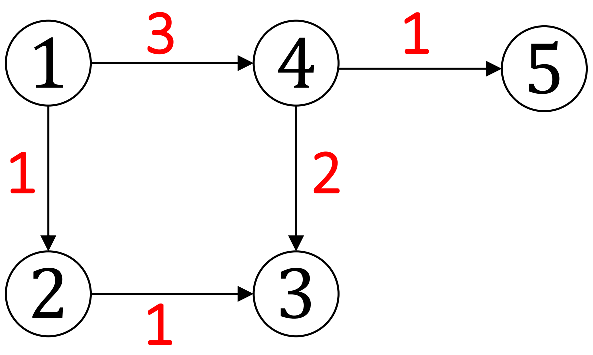
Q

3
5

?

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

Example variant 3



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

E

from	to	
1	2	1
2	3	1
1	4	3
4	3	2
4	5	1

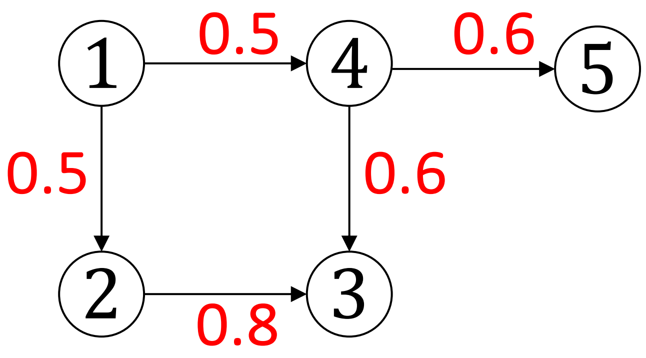
Q(z) :- E(1,y), E(y,z)

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

from	to	
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) →

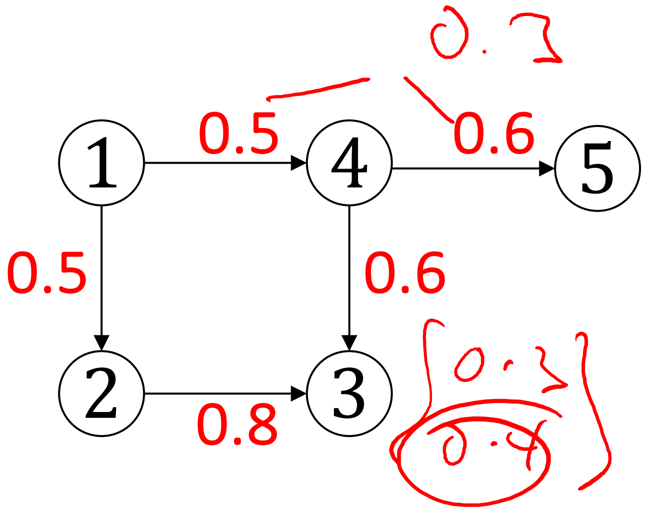
Q

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

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

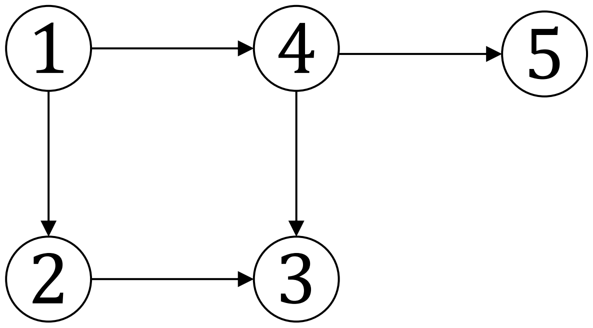
Q(z) :- E(1,y), E(y,z) →

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

from	to
1	2
2	3
1	4
4	3
4	5

Q(z) :- E(1,y), E(y,z) →

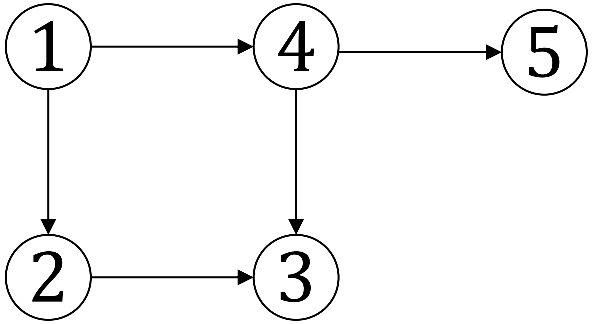
Q

3
5

?

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

Example variant 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

from	to
1	2
2	3
1	4
4	3
4	5

Q(z) :- E(1,y), E(y,z) →

Q

Q	
3	2
5	1

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

Outline: T2-3: Provenance

- T2-3: Provenance
  - Data Provenance
  - The Semiring Framework for Provenance
  - Algebra: Monoids and Semirings
  - Query-rewrite-insensitive provenance

Mainly slides by Val Tannen 2017

Positive relational algebra:

$\times$ $\bowtie$

σ

π

$\cup$

$(-)$

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)

REASONING

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

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

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

$X\ \{x, y, z\}$

Set X of provenance tokens.

Space of annotations, provenance expressions Prov(X) $\supset \{x \cdot y \cdot y + z,\ zy, \ldots\}$

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).

K-Relational algebra

Algebraic laws of $(\mathrm{Prov}(X), +, \cdot, 0,1)$? 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] $(\mathrm{BoolExp}(X), \vee, \wedge, \bot, \top)$

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

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

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

5/15/2017 PODS 2017 13

Two kinds of semirings in this framework

Provenance semirings, e.g.,

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

$(\mathrm{Why}(X), \cup, \uplus, \emptyset, \{\emptyset\})$ 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

Example 1: $(\mathbb{B}, \wedge, \vee, \top, \bot)$ *binary trust*

Example 5: $(\mathbb{N}, +, \cdot, 0, 1)$ *multiplicity (number of derivations)*

Example 2: $(\mathbb{A}, \min, \max, 0, \mathrm{Pub})$ *access control*

Example 4: $\mathbb{V} = ([0,1], \max, \cdot, 0, 1)$ Viterbi semiring (MPE) *confidence scores*

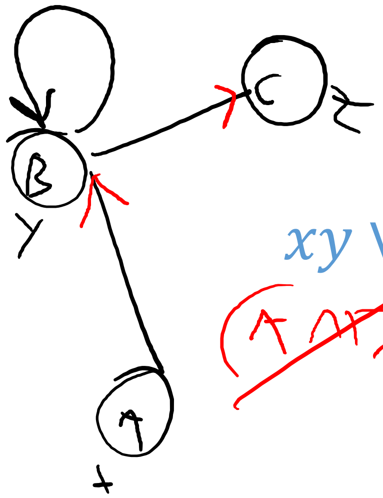
Example 3: $\mathbb{T} = ([0, \infty], \min, +, \infty, 0)$
tropical semiring (shortest paths) *data pricing*

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

$\underbrace{(A \wedge B) \vee (B)}_{\varphi} \Longrightarrow \underbrace{}_{\psi}$



= Y

$xy \vee y^2 \vee yz$

$\cancel{(A \wedge B)} \vee (\cancel{B \wedge B}) \vee (\cancel{B \wedge C})$?

Example: $2x^2y + xy + 5y^2 + xz$

$\mathbb{N}[X]$

+ idemp. · ~~idemp.~~

$x^2y + xy + y^2 + xz$ $\mathbb{B}[X]$ Trio(X) $3xy + 5y + xz$

absorption (ab+a=a) · ~~idemp.~~ + idemp.

$xy + y^2 + xz$ Sorp(X) Why(X) $xy + y + xz$

· ~~idemp.~~ absorption + = ·

$y + xz$ PosBool(X) Which(X) xyz

most informative

least informative

$\varphi \vee \psi = \psi$

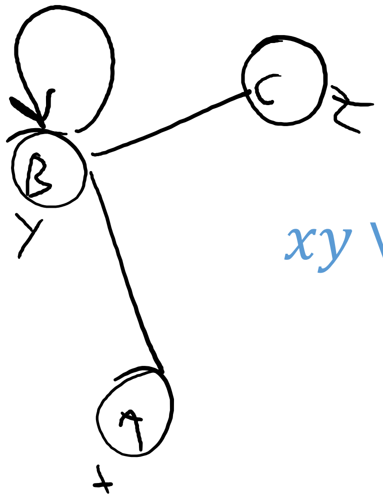
N	
A	X
B	Y
C	Z

Q :- N(X), E(X,Y), N(Y)

E	
A	B
B	B
B	

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]

$x^2 = x \cdot x = x$
example: $x \land x = x$
But not true! Take:
$(x+y)^2 = x + xy + y \mathrel{!=} x + y$

$xy \lor y^2 \lor yz$
↓
y

Example: $2x^2y + xy + 5y^2 + xz$

$\mathbb{N}[X]$

+ idemp.

~~• idemp.~~

just ignore exponents

$x^2y + xy + y^2 + xz$ $\mathbb{B}[X]$

Trio(X) $3xy + 5y + xz$

absorption (ab+a=a)

~~• idemp.~~

+ idemp.

$x + x = x$
example: $x \lor x = x$

most informative

least informative

$xy + y^2 + xz$ Sorp(X)

Why(X) $xy + y + xz$

~~• idemp.~~

absorption

$+ = \cdot$

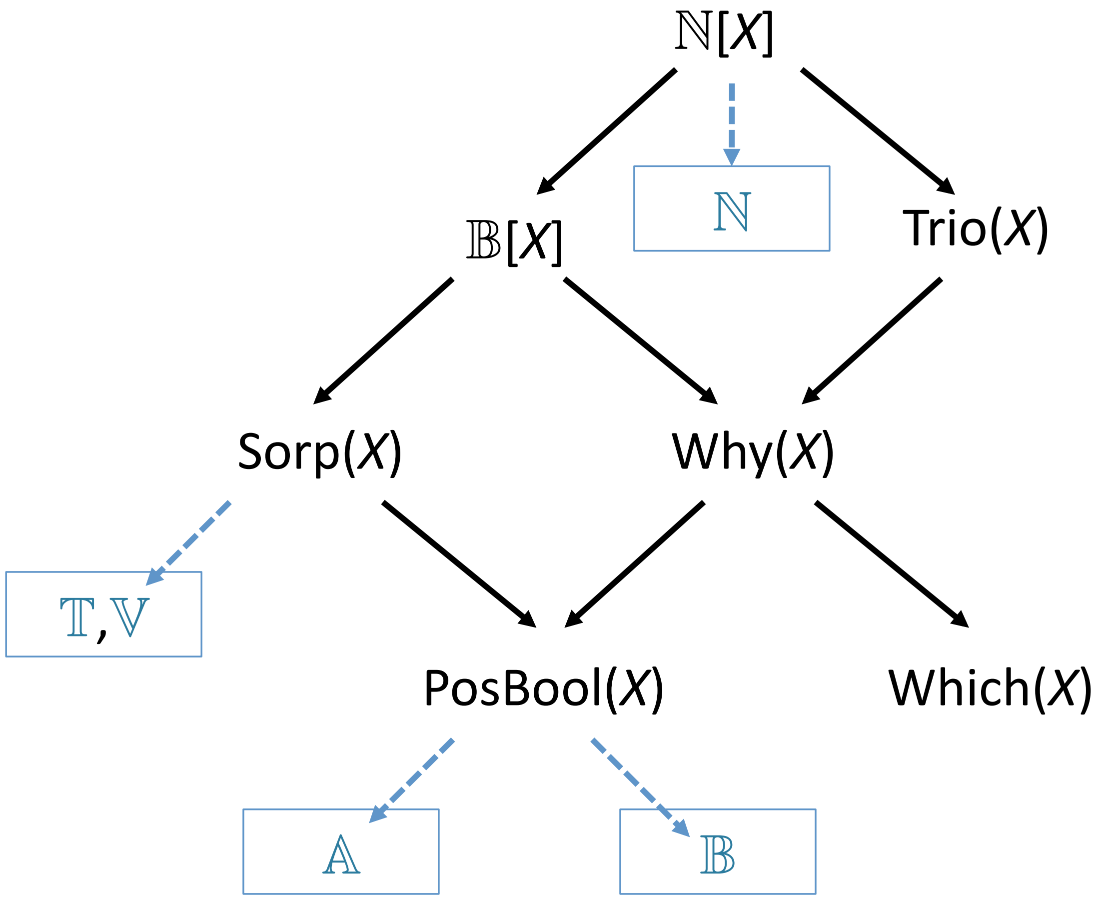
$y + xz$ PosBool(X)

Which(X) xyz

Positive Boolean expressions

↓ surjective semiring homomorphism, identity on X

A Hierarchy of Provenance Semirings [G09, DMRT14]



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

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]

Positive relational algebra: Join ⋈

$\bowtie$

R

A	B
1	1
2	1
2	2

S

B	C
1	1
2	2
2	3

$Q = R \bowtie S$

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 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
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

$Q = R \bowtie S$

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
alternative use 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 ∪

∪

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

A	B	
1	1	r_1
2	1	r_2

S

A	B	
2	1	s_1
2	2	s_2

$\{(2,1), (2,1)\} = (2,1) \mapsto 2$

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	$r_5 \cdot 0$

Boolean Query Provenance

Q :- R(x,y), S(y,z)

Calculate the provenance, operator-by-operator, with two algebraically equivalent query plans:

R

A	B	
1	1	r_1
2	2	r_2
3	2	r_3

S

B	C	
1	1	s_1
1	2	s_2
2	3	s_3

Query plan 1: $\pi_{\text{-}A,B,C}(R \bowtie S)$

?

Query plan 2: $\pi_{\text{-}B}(\pi_{\text{-}A}(R) \bowtie \pi_{\text{-}C}(S))$

?

Boolean Query Provenance

Q :- R(x,y), S(y,z)

R

A	B	
1	1	r_1
2	2	r_2
3	2	r_3

S

B	C	
1	1	s_1
1	2	s_2
2	3	s_3

Calculate the provenance, operator-by-operator, with two algebraically equivalent query plans:

Query plan 1: $\pi_{\text{-A,B,C}}(\underbrace{R \bowtie S})$

$R \bowtie S$

?

$\pi_{\text{-A,B,C}}(\ldots)$

?

Query plan 2: $\pi_{\text{-B}}(\pi_{\text{-A}}(R) \bowtie \pi_{\text{-C}}(S))$

?

Boolean Query Provenance

Q :- R(x,y), S(y,z)

Calculate the provenance, operator-by-operator, with two algebraically equivalent query plans:

R

A	B	
1	1	r_1
2	2	r_2
3	2	r_3

S

B	C	
1	1	s_1
1	2	s_2
2	3	s_3

Query plan 1: $\pi_{\text{-A,B,C}}(\underbrace{R \bowtie S})$

Query plan 2: $\pi_{\text{-B}}(\pi_{\text{-A}}(R) \bowtie \pi_{\text{-C}}(S))$

$R \bowtie S$

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

$\pi_{\text{-A,B,C}}(\ldots)$

?

?

Boolean Query Provenance

450

Q :- R(x,y), S(y,z)

Calculate the provenance, operator-by-operator, with two algebraically equivalent query plans:

R

A	B	
1	1	r_1
2	2	r_2
3	2	r_3

S

B	C	
1	1	s_1
1	2	s_2
2	3	s_3

Query plan 1: $\pi_{\text{-}A,B,C}(\underbrace{R \bowtie S})$

Query plan 2: $\pi_{\text{-}B}(\pi_{\text{-}A}(R) \bowtie \pi_{\text{-}C}(S))$

$R \bowtie S$

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

$\pi_{\text{-}A,B,C}(\ldots)$

```
SELECT EXISTS(
  SELECT *
  FROM R, S
  WHERE R.B = S.B)
```

	exists boolean
1	true

$r_1 \cdot s_1 + r_1 \cdot s_2 + r_2 \cdot s_3 + r_3 \cdot s_3$

Boolean Query Provenance

Q :- $R(x,y)$, $S(y,z)$

Calculate the provenance, operator-by-operator, with two algebraically equivalent query plans:

R

A	B	
1	1	r_1
2	2	r_2
3	2	r_3

S

B	C	
1	1	s_1
1	2	s_2
2	3	s_3

Query plan 1: $\pi_{\text{-}A,B,C}(R \bowtie S)$

$R \bowtie S$

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

$\pi_{\text{-}A,B,C}(\ldots)$

$r_1 \cdot s_1 + r_1 \cdot s_2 + r_2 \cdot s_3 + r_3 \cdot s_3$

Query plan 2: $\pi_{\text{-}B}(\pi_{\text{-}A}(R) \bowtie \pi_{\text{-}C}(S))$

?

Boolean Query Provenance

Q :- $R(x,y)$, $S(y,z)$

Calculate the provenance, operator-by-operator, with two algebraically equivalent query plans:

R

A	B	
1	1	r_1
2	2	r_2
3	2	r_3

S

B	C	
1	1	s_1
1	2	s_2
2	3	s_3

Query plan 1: $\pi_{\text{-}A,B,C}(\underbrace{R \bowtie S})$

$R \bowtie S$

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

$\pi_{\text{-}A,B,C}(\ldots)$

$r_1 \cdot s_1 + r_1 \cdot s_2 + r_2 \cdot s_3 + r_3 \cdot s_3$

Query plan 2: $\pi_{\text{-}B}(\underbrace{\pi_{\text{-}A}(R)} \bowtie \underbrace{\pi_{\text{-}C}(S)})$

$\pi_{\text{-}A}(R)$

?

$\pi_{\text{-}C}(S)$

?

$\pi_{\text{-}B}(R' \bowtie S')$

?

Boolean Query Provenance

Q :- R(x,y), S(y,z)

Calculate the provenance, operator-by-operator, with two algebraically equivalent query plans:

R

A	B	
1	1	r_1
2	2	r_2
3	2	r_3

S

B	C	
1	1	s_1
1	2	s_2
2	3	s_3

Query plan 1: $\pi_{\text{-}A,B,C}(\underbrace{R \bowtie S})$

$R \bowtie S$

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

$\pi_{\text{-}A,B,C}(\ldots)$

$r_1 \cdot s_1 + r_1 \cdot s_2 + r_2 \cdot s_3 + r_3 \cdot s_3$

Query plan 2: $\pi_{\text{-}B}(\underbrace{\pi_{\text{-}A}(R)} \bowtie \underbrace{\pi_{\text{-}C}(S)})$

$\pi_{\text{-}A}(R)$

B	
1	r_1
2	r_2+r_3

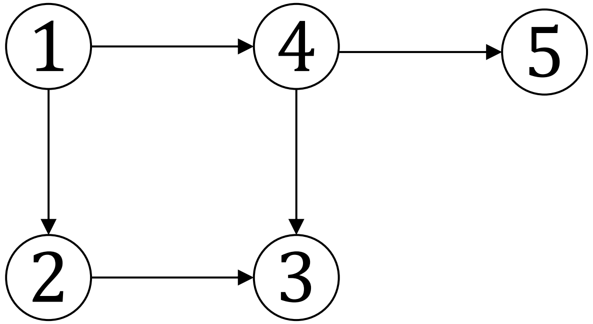
$\pi_{\text{-}C}(S)$

B	
1	s_1+s_2
2	s_3

$\pi_{\text{-}B}(R' \bowtie S')$

$r_1 \cdot (s_1+s_2) + (r_2+r_3) \cdot s_3$

Back to our Example: now with Semiring notation



Now assume we use semiring notation.
Idea: keep the tuple identifiers abstract.
Use provenance polynomials ($\mathbb{N}[X]$, +, ·, 0, 1)

E

1	2
2	3
1	4
4	3
4	5

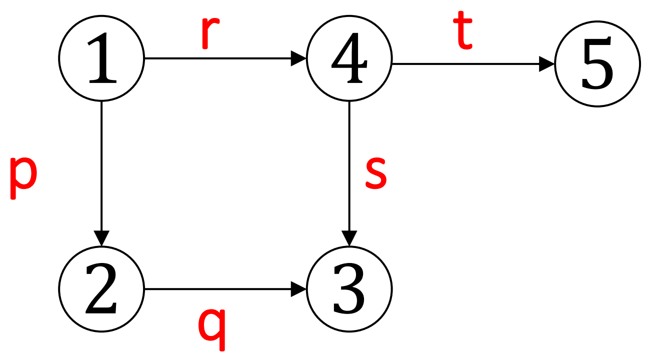
Q(z) :- E(1,y), E(y,z) ⟶

Q

3
5

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

Back to our Example: now with Semiring notation



Now assume we use semiring notation.

Idea: keep the tuple identifiers abstract.

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

$\oplus$ $\otimes$

E

1	2	p
2	3	q
1	4	r
4	3	s
4	5	t

$X = \{p, q, \ldots, t\}$

Q(z) :- E(1,y), E(y,z) $\longrightarrow$

Q

3	r·s+p·q
5	r·t

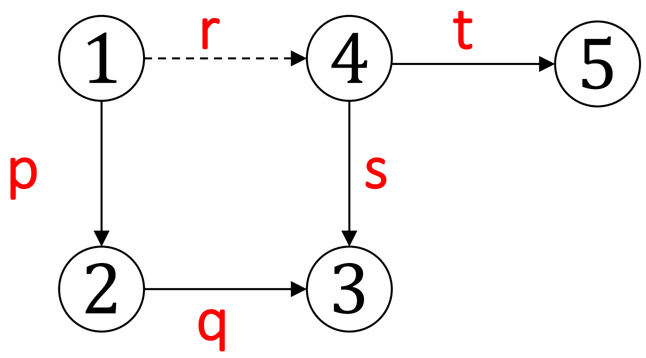
$\in \mathbb{N}(X)$

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

$\mathbb{N}[X]$=($\mathbb{N}[X]$, +, ·, 0, 1): Provenance polynomials

Example variant 1

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



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

E

1	2	$p = 1$
2	3	$q = 1$
~~1~~	~~4~~	~~$r = 0$~~
4	3	$s = 1$
4	5	$t = 1$

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

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

Q

3	$r{\cdot}s+p{\cdot}q$	$= 1$
5	$r{\cdot}t$	$= 0$

$(0 \wedge 1) \vee (1 \wedge 1) = 1$

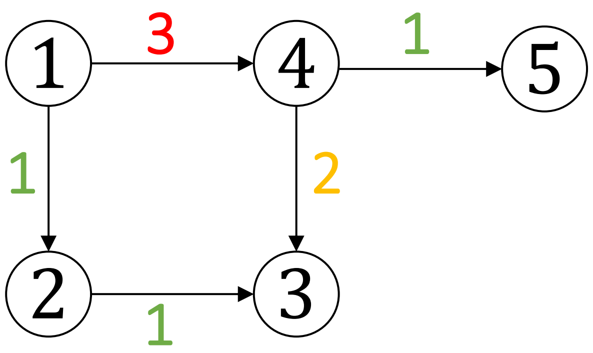
$(0 \wedge 1) = 0$

{0,1}

$\mathbb{B}=(\mathbb{B}, \vee, \wedge, 0, 1)$: Boolean algebra

Example variant 2

Provenance polynomials ($\mathbb{N}[X]$, +, ·, 0, 1)



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

infix ~> prefix

min[max[3,2], max[1,1]] = 1

E

1	2	p = 1
2	3	q = 1
1	4	r = 3
4	3	s = 2
4	5	t = 1

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

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

Q

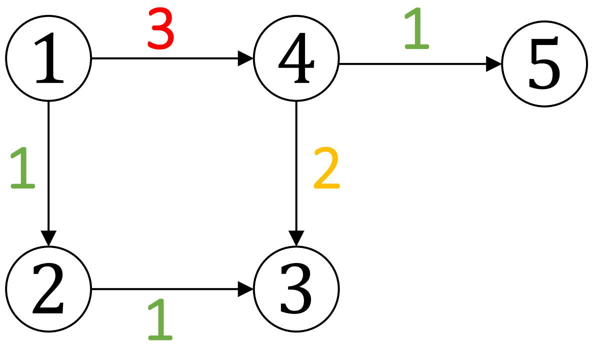
3	r·s+p·q	= 1
5	r·t	= 3

max[3,1] = 3

({1,2,3,∞}, min, max, ∞,1)

Example variant 3

Provenance polynomials ($\mathbb{N}[X]$, +, ·, 0, 1)



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

E

1	2	p = 1
2	3	q = 1
1	4	r = 3
4	3	s = 2
4	5	t = 1

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

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

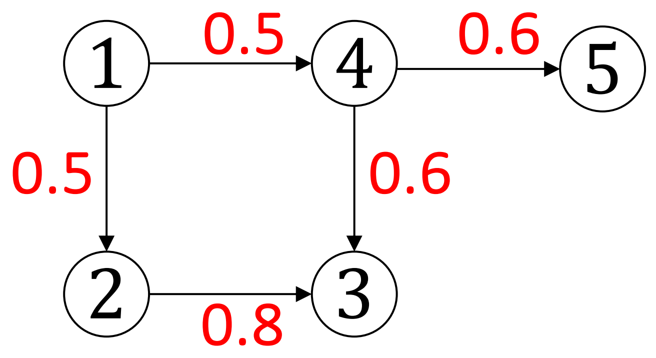
Q

3	r·s+p·q	= 2	min[3+2,1+1] = 2
5	r·t	= 4	3+1 = 4

$\mathbb{T}=(\mathbb{R}_+^\infty,\min,+,\infty,0)$: Tropical semiring

Example variant 4

Provenance polynomials ($\mathbb{N}[X]$, +, ·, 0, 1)



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

E

1	2
2	3
1	4
4	3
4	5

$p = 0.5$
$q = 0.8$
$r = 0.5$
$s = 0.6$
$t = 0.6$

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

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

Q

3
5

$r \cdot s + p \cdot q = 0.4$
$r \cdot t = 0.3$

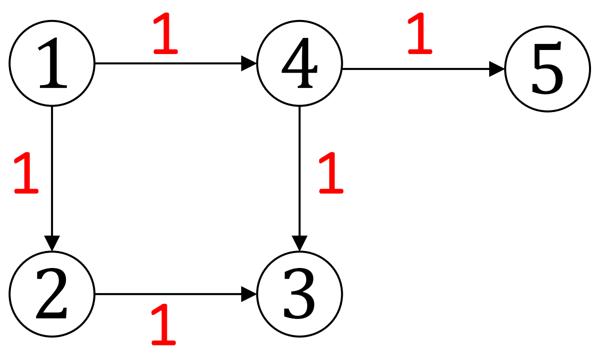
max[0.5·0.6,0.5·0.8] = 0.4

0.5·0.6 = 0.3

$\mathbb{V}$=([0,1],max, · ,0,1): Viterbi semiring (max likely sequence)

Example variant 5

Provenance polynomials ($\mathbb{N}[X]$, +, ·, 0, 1)



Finally assume we want to calculate the number of paths to a node. We start by annotating the tuples in the database with their duplicity (which is 1 to start with)

E

1	2	$p = 1$
2	3	$q = 1$
1	4	$r = 1$
4	3	$s = 1$
4	5	$t = 1$

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

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

Q

3	$r \cdot s + p \cdot q = 2$	$1 \cdot 1 + 1 \cdot 1 = 2$
5	$r \cdot t = 1$	$1 \cdot 1 = 1$

($\mathbb{N}$, +, ·, 0, 1): Counting derivations / bag semantics

Updated 3/19/2024

Topic 2: Complexity of Query Evaluation
Unit 3: Provenance
Lecture 17

Wolfgang Gatterbauer

CS7240 Principles of scalable data management (sp24)

https://northeastern-datalab.github.io/cs7240/sp24/

3/19/2024

Pre-class conversations

- Last class summary
- Projects: TUE 3/26 intermediate report
- Faculty candidate tomorrow WED 3/20

- Today:
  - provenance, semirings

PRELIMINARY

Topic 2: Complexity of Query Evaluation & Reverse Data Management

- **Lecture 14 (Fri 3/1):** T2-U1 Conjunctive Queries
- Spring break (Tue 3/5, Fri 3/8)
- **Lecture 15 (Tue 3/12):** T2-U1/2 Conjunctive & Beyond Conjunctive Queries
- **Lecture 16 (Fri 3/15):** T2-U1/2 Conjunctive & Beyond Conjunctive Queries
- **Lecture 17 (Tue 3/19):** T2-U3 Provenance
- **Lecture 18 (Fri 3/22):** T2-U3 Provenance
- **Lecture 20 (Tue 3/26):** T2-U4 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], [Gatterbauer'11], [Martens'17]
- **Unit 3. Provenance**: [Buneman+'02], [Green+'07], [Cheney+'09], [Green,Tannen'17], [Kepner+16], [Buneman, Tan'18], [Simons'23], [Dagstuhl'24]
- **Unit 4. Reverse Data Management**: update propagation, resilience: [Buneman+'02], [Kimelfeld+'12], [Freire+'15], [Makhija+'24]

A more complex example with exponents

$$Q(R) = \pi_{AC}(\pi_{AB}R \bowtie \pi_{BC}R \ \cup \ \pi_{AC}R \bowtie \pi_{BC}R)$$

R

A	B	C
a	b	c
d	b	e
f	g	e

$\pi_{AB}R \bowtie \pi_{BC}R$

A	B	C

?

$\pi_{AC}R \bowtie \pi_{BC}R$

A	B	C

?

$Q_1 \cup Q_2$

A	B	C

?

Q

A	C

?

Example from Section 2 of Green, Karvounarakis, Val Tannen. "Provenance Semirings", PODS 2007. https://doi.org/10.1145/1265530.1265535

A more complex example with exponents

$$Q(R) = \pi_{AC}(\pi_{AB}R \bowtie \pi_{BC}R \cup \pi_{AC}R \bowtie \pi_{BC}R)$$

R

A	B	C	
a	b	c	X
d	b	e	Y
f	g	e	Z

$\pi_{AB}R \bowtie \pi_{BC}R$

A	B	C
a	b	c
a	b	e
d	b	c
d	b	e
f	g	e

?

$\pi_{AC}R \bowtie \pi_{BC}R$

A	B	C
a	b	c
d	b	e
d	g	e
f	b	e
f	g	e

?

$Q_1 \cup Q_2$

A	B	C
a	b	c
a	b	e
d	b	c
d	b	e
d	g	e
f	b	e
f	g	e

?

Q

A	C
a	c
a	e
d	c
d	e
f	e

?

Example from Section 2 of Green, Karvounarakis, Val Tannen. "Provenance Semirings", PODS 2007. https://doi.org/10.1145/1265530.1265535

A more complex example with exponents

$$Q(R) = \pi_{AC}(\pi_{AB}R \bowtie \pi_{BC}R \ \cup \ \pi_{AC}R \bowtie \pi_{BC}R)$$

R

A	B	C	
a	b	c	X
d	b	e	Y
f	g	e	Z

$\pi_{AB}R \bowtie \pi_{BC}R$

A	B	C	
a	b	c	X^2
a	b	e	XY
d	b	c	XY
d	b	e	Y^2
f	g	e	Z^2

$\pi_{AC}R \bowtie \pi_{BC}R$

A	B	C
a	b	c
d	b	e
d	g	e
f	b	e
f	g	e

?

$Q_1 \cup Q_2$

A	B	C
a	b	c
a	b	e
d	b	c
d	b	e
d	g	e
f	b	e
f	g	e

?

Q

A	C
a	c
a	e
d	c
d	e
f	e

?

A more complex example with exponents

$$Q(R) = \pi_{AC}(\underbrace{\pi_{AB}R \bowtie \pi_{BC}R}_{} \cup \underbrace{\pi_{AC}R \bowtie \pi_{BC}R}_{})$$

R

A	B	C	
a	b	c	X
d	b	e	Y
f	g	e	Z

$\pi_{AB}R \bowtie \pi_{BC}R$

A	B	C	
a	b	c	X^2
a	b	e	XY
d	b	c	XY
d	b	e	Y^2
f	g	e	Z^2

$\pi_{AC}R \bowtie \pi_{BC}R$

A	B	C	
a	b	c	X^2
d	b	e	Y^2
d	g	e	YZ
f	b	e	YZ
f	g	e	Z^2

$Q_1 \cup Q_2$

A	B	C
a	b	c
a	b	e
d	b	c
d	b	e
d	g	e
f	b	e
f	g	e

?

Q

A	C
a	c
a	e
d	c
d	e
f	e

?

A more complex example with exponents

$$Q(R) = \pi_{AC}(\pi_{AB}R \bowtie \pi_{BC}R \;\cup\; \pi_{AC}R \bowtie \pi_{BC}R)$$

R

A	B	C	
a	b	c	X
d	b	e	Y
f	g	e	Z

$\pi_{AB}R \bowtie \pi_{BC}R$

A	B	C	
a	b	c	X^2
a	b	e	XY
d	b	c	XY
d	b	e	Y^2
f	g	e	Z^2

$\pi_{AC}R \bowtie \pi_{BC}R$

A	B	C	
a	b	c	X^2
d	b	e	Y^2
d	g	e	YZ
f	b	e	YZ
f	g	e	Z^2

$Q_1 \cup Q_2$

A	B	C	
a	b	c	$2X^2$
a	b	e	XY
d	b	c	XY
d	b	e	$2Y^2$
d	g	e	YZ
f	b	e	YZ
f	g	e	$2Z^2$

Q

A	C
a	c
a	e
d	c
d	e
f	e

?

A more complex example with exponents

$$Q(R) = \pi_{AC}(\pi_{AB}R \bowtie \pi_{BC}R \cup \pi_{AC}R \bowtie \pi_{BC}R)$$

R

A	B	C	
a	b	c	X
d	b	e	Y
f	g	e	Z

$\pi_{AB}R \bowtie \pi_{BC}R$

A	B	C	
a	b	c	X^2
a	b	e	XY
d	b	c	XY
d	b	e	Y^2
f	g	e	Z^2

$\pi_{AC}R \bowtie \pi_{BC}R$

A	B	C	
a	b	c	X^2
d	b	e	Y^2
d	g	e	YZ
f	b	e	YZ
f	g	e	Z^2

$Q_1 \cup Q_2$

A	B	C	
a	b	c	$2X^2$
a	b	e	XY
d	b	c	XY
d	b	e	$2Y^2$
d	g	e	YZ
f	b	e	YZ
f	g	e	$2Z^2$

Q

A	C	
a	c	$2X^2$
a	e	XY
d	c	XY
d	e	$2Y^2+YZ$
f	e	$YZ+2Z^2$

A more complex example with exponents

$$Q(R) = \pi_{AC}(\underbrace{\pi_{AB}R \bowtie \pi_{BC}R}_{} \cup \underbrace{\pi_{AC}R \bowtie \pi_{BC}R}_{})$$

R

A	B	C	
a	b	c	X =2
d	b	e	Y =5
f	g	e	Z =1

$\pi_{AB}R \bowtie \pi_{BC}R$

A	B	C	
a	b	c	X^2
a	b	e	XY
d	b	c	XY
d	b	e	Y^2
f	g	e	Z^2

$\pi_{AC}R \bowtie \pi_{BC}R$

A	B	C	
a	b	c	X^2
d	b	e	Y^2
d	g	e	YZ
f	b	e	YZ
f	g	e	Z^2

$Q_1 \cup Q_2$

A	B	C	
a	b	c	$2X^2$
a	b	e	XY
d	b	c	XY
d	b	e	$2Y^2$
d	g	e	YZ
f	b	e	YZ
f	g	e	$2Z^2$

Q

A	C	
a	c	$2X^2$
a	e	XY
d	c	XY
d	e	$2Y^2+YZ$
f	e	$YZ+2Z^2$

Let's assume bag semantics and duplicities in the input. How many output tuples do we get?

?

$(\mathbb{N}, +, \cdot, 0, 1)$: Counting derivations / bag semantics

Example from Section 2 of Green, Karvounarakis, Val Tannen. "Provenance Semirings", PODS 2007. https://doi.org/10.1145/1265530.1265535

A more complex example with exponents


$$Q(R) = \pi_{AC}(\underbrace{\pi_{AB}R \bowtie \pi_{BC}R}_{} \cup \underbrace{\pi_{AC}R \bowtie \pi_{BC}R}_{})$$

R

A	B	C	
a	b	c	X =2
d	b	e	Y =5
f	g	e	Z =1

$\pi_{AB}R \bowtie \pi_{BC}R$

A	B	C	
a	b	c	X^2
a	b	e	XY
d	b	c	XY
d	b	e	Y^2
f	g	e	Z^2

$\pi_{AC}R \bowtie \pi_{BC}R$

A	B	C	
a	b	c	X^2
d	b	e	Y^2
d	g	e	YZ
f	b	e	YZ
f	g	e	Z^2

$Q_1 \cup Q_2$

A	B	C	
a	b	c	$2X^2$
a	b	e	XY
d	b	c	XY
d	b	e	$2Y^2$
d	g	e	YZ
f	b	e	YZ
f	g	e	$2Z^2$

Q

A	C		
a	c	$2X^2$	=8
a	e	XY	=10
d	c	XY	=10
d	e	$2Y^2+YZ$	=55
f	e	$YZ+2Z^2$	=7

Let's assume bag semantics and duplicities in the input. How many output tuples do we get?

$(\mathbb{N}, +, \cdot, 0, 1)$: Counting derivations / bag semantics

A more complex example with exponents

617

$$Q(R) = \pi_{AC}(\underbrace{\pi_{AB}R \bowtie \pi_{BC}R}_{\pi_{R.A,R.B,R2.C}(R \bowtie_{R.B=R2.B} \rho_{R\to R2}R)} \cup \underbrace{\pi_{AC}R \bowtie \pi_{BC}R}_{\pi_{R.A,R2.B,R.C}(R \bowtie_{R.C=R2.C} \rho_{R\to R2}R)})$$

R

A	B	C	
a	b	c	X =2
d	b	e	Y =5
f	g	e	Z =1

```sql
SELECT A, C, COUNT(*)
FROM (
  SELECT R.A, R.B, R2.C
  FROM R, R R2
  WHERE R.B = R2.B
  UNION ALL
  SELECT R.A, R2.B, R.C
  FROM R, R R2
  WHERE R.C = R2.C) X
GROUP BY A, C
ORDER BY A, C
```

Q

A	C		
a	c	$2X^2$	=8
a	e	XY	=10
d	c	XY	=10
d	e	$2Y^2+YZ$	=55
f	e	$YZ+2Z^2$	=7

	a character varying	c character varying	count bigint
1	a	c	8
2	a	e	10
3	d	c	10
4	d	e	55
5	f	e	7

SQL example available at: https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql

Example from Section 2 of Green, Karvounarakis, Val Tannen. "Provenance Semirings", PODS 2007. https://doi.org/10.1145/1265530.1265535

Outline: T2-3: Provenance

- T2-3: Provenance
  - Data Provenance
  - The Semiring Framework for Provenance
  - Algebra: Monoids and Semirings
  - Query-rewrite-insensitive provenance

SIMONS INSTITUTE for the Theory of Computing

Logic and Algebra for Query Evaluation

Program Logic and Algorithms in Database Theory and AI

Date Monday, Nov. 13 – Friday, Nov. 17, 2023

About

The workshop will discuss semantics of logic programs over general semirings: constraints over semirings, query complexity with semiring semantics, termination conditions of logic programs over semirings. The connection between semirings and logic is a relatively new development in database theory (since 2007), and this area has high potential for major innovation. Some of the problems discussed at the workshop will be inspired by systems, others will be purely theoretical in nature, such as the quest for finding appropriate extensions of Pebble Games to semiring semantics.

Chairs/Organizers


Sudeepa Roy (Duke University; co-chair)


Dan Suciu (University of Washington; co-chair)


Wolfgang Gatterbauer (Northeastern University)


Val Tannen (University of Pennsylvania)

https://simons.berkeley.edu/workshops/logic-algebra-query-evaluation#simons-tabs

Why algebra? Think abstraction and generalization

- Abstraction:

- Generalization:

Why algebra? Think abstraction and generalization

- Abstraction: an emphasis on the idea and properties rather than the particulars (hiding irrelevant details)
  - main goal in "Abstract algebra"
  - e.g. groups in group theory
- Generalization:

For instance, consider the following three objects:

1. The set of functions A, B, C defined on the set $\{1, 2, 3\}$ by
   $A(1) = 1,\ A(2) = 2,\ A(3) = 3,$
   $B(1) = 2,\ B(2) = 3,\ B(3) = 1,$
   $C(1) = 3,\ C(2) = 1,\ C(3) = 2,$
2. The set of complex numbers $a = 1, b = e^{i2\pi/3}, c = e^{i4\pi/3}$.
3. The set of matrices $\alpha = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \beta = \begin{pmatrix} 0 & -1 \\ -1 & -1 \end{pmatrix}, \gamma = \begin{pmatrix} -1 & -1 \\ -1 & 0 \end{pmatrix}$

Consider these notations: AB means A(B(x)), ab is ordinary multiplication of complex numbers, and $\alpha\beta$ means ordinary matrix multiplication. Verify the following "multiplication" tables:

	A	B	C
A	A	B	C
B	B	C	A
C	C	A	B

	a	b	c
a	a	b	c
b	b	c	a
c	c	a	b

	α	β	γ
α	α	β	γ
β	β	γ	α
γ	γ	α	β

Notice that these tables are identical. Then let us by *abstraction* define an abstract object which is the set of three elements $\{e, g, g^{-1}\}$ paired with a binary operator $\cdot$ such that set acts on itself in the following manner with respect to the operator:

$\cdot$	e	g	g^{-1}
e	e	g	g^{-1}
g	g	g^{-1}	e
g^{-1}	g^{-1}	e	g

In Group Theory an object with such structure is called the cyclic group of order three. Then the examples above are representations of this abstract object. It is an abstract object because while we have now given it a definition, notice that it is itself a stand-in for a variety of objects that have the properties that it demonstrates. You might even consider the abstract object to be more of a set

Why algebra? Think abstraction and generalization

- Abstraction: an emphasis on the idea and properties rather than the particulars (hiding irrelevant details)
  - main goal in "Abstract algebra"
  - e.g. groups in group theory
- Generalization: a broadening of application to several objects with similar functions.
  - e.g. Algorithms: finding the shortest path not just in one graph but any graph

For instance, consider the following three objects:

1. The set of functions A, B, C defined on the set $\{1, 2, 3\}$ by
   $A(1) = 1,\ A(2) = 2,\ A(3) = 3,$
   $B(1) = 2,\ B(2) = 3,\ B(3) = 1,$
   $C(1) = 3,\ C(2) = 1,\ C(3) = 2,$
2. The set of complex numbers $a = 1, b = e^{i2\pi/3}, c = e^{i4\pi/3}$.
3. The set of matrices $\alpha = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \beta = \begin{pmatrix} 0 & -1 \\ -1 & -1 \end{pmatrix}, \gamma = \begin{pmatrix} -1 & -1 \\ -1 & 0 \end{pmatrix}$

Consider these notations: AB means A(B(x)), ab is ordinary multiplication of complex numbers, and $\alpha\beta$ means ordinary matrix multiplication. Verify the following "multiplication" tables:

	A	B	C
A	A	B	C
B	B	C	A
C	C	A	B

	a	b	c
a	a	b	c
b	b	c	a
c	c	a	b

	α	β	γ
α	α	β	γ
β	β	γ	α
γ	γ	α	β

Notice that these tables are identical. Then let us by *abstraction* define an abstract object which is the set of three elements $\{e, g, g^{-1}\}$ paired with a binary operator $\cdot$ such that set acts on itself in the following manner with respect to the operator:

$\cdot$	e	g	g^{-1}
e	e	g	g^{-1}
g	g	g^{-1}	e
g^{-1}	g^{-1}	e	g

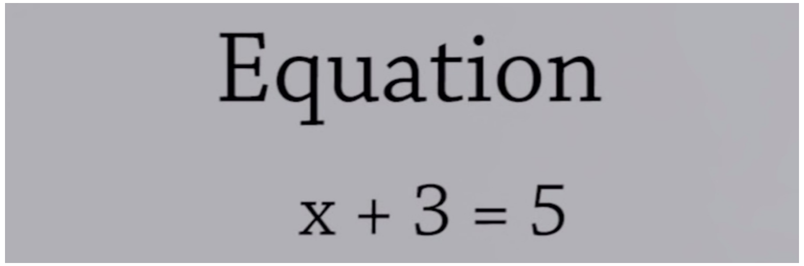
In Group Theory an object with such structure is called the cyclic group of order three. Then the examples above are representations of this abstract object. It is an abstract object because while we have now given it a definition, notice that it is itself a stand-in for a variety of objects that have the properties that it demonstrates. You might even consider the abstract object to be more of a set

Example on the right from: https://matheducators.stackexchange.com/questions/10949/what-is-abstraction-and-generalization/10957

Let's start with groups! Why groups?

- Groups are one of the most important structures studied in abstract algebra
- What is so special about groups?

Groups have the minimum properties needed to solve equations



Can you solve that ?

Groups have the minimum properties needed to solve equations

($\mathbb{Z}$,+, 0): Integers under addition

$(a+b)+c$

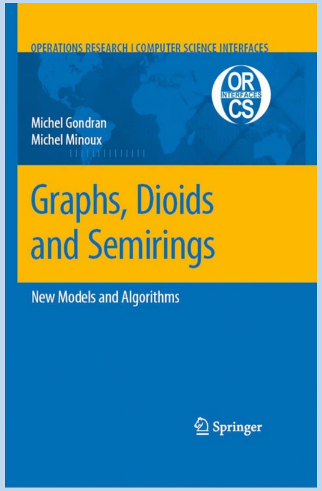
$a+(b+c)$

Equation	Properties
$x + 3 = 5$	✓ Integers under +
$[x + 3] + (-3) = 5 + (-3)$	✓ Inverses
$[x + 3] + (-3) = 2$	✓ Closed under +
$x + [3 + (-3)] = 2$	✓ Associativity
$x + 0 = 2$	✓ Identity
$x = 2$	

Screenshot from: Socratica: Abstract Algebra: Motivation for the definition of a group, https://www.youtube.com/watch?v=yHq_yzYZV6U

Why something weaker than groups?

- For some important computational problems like Dynamic Programming, we don't need to "solve equations".
  - Thus we don't need an inverse ("we don't need to go back")
- Let's look at weaker structures



Preface

During the last two or three centuries, most of the developments in science (in particular in Physics and Applied Mathematics) have been founded on the use of classical algebraic structures, namely groups, rings and fields. However many situations can be found for which those usual algebraic structures do not necessarily provide the most appropriate tools for modeling and problem solving. ...

Screenshot from: Gondran, Minoux. "Graphs, Dioids and Semirings: New Models and Algorithms", 2008. https://www.springer.com/gp/book/9780387754499

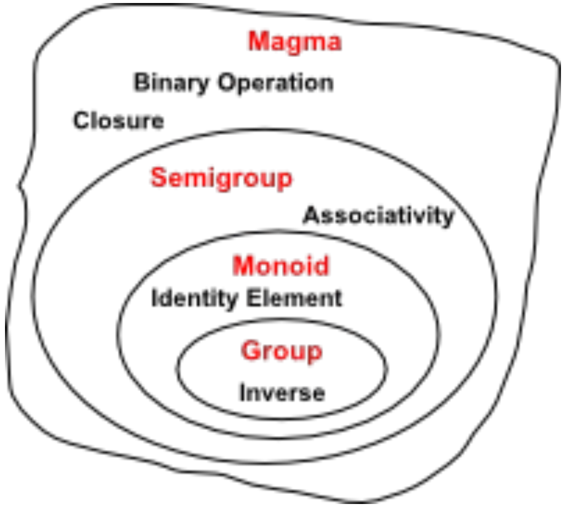
Group-like structures

Set S

(also "total")

+ 1. Closed binary operation ⊕:
If x,y∈S then the image (x⊕y)∈S

Magma (S,⊕)



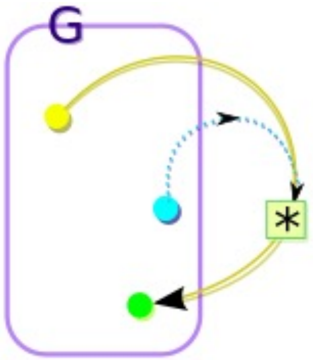


Figure credits: https://www.euclideanspace.com/maths/discrete/groups/monoid/index.htm , https://en.wikibooks.org/wiki/Abstract_Algebra/Group_Theory/Group/Definition_of_a_Group

Group-like structures

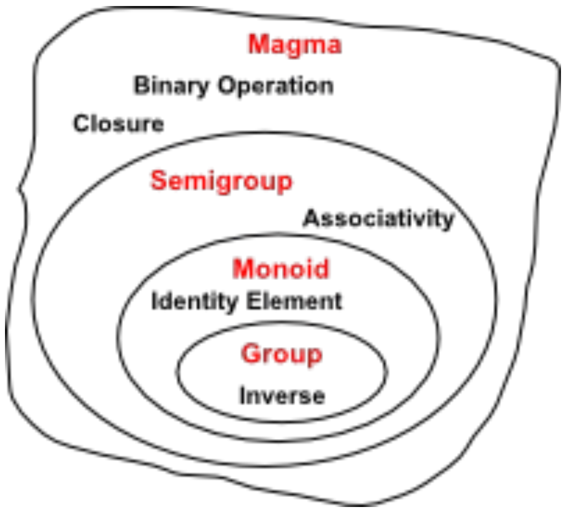
Set S

\+ 1. Closed binary operation $\oplus$:
If $x,y \in S$ then the image $(x \oplus y) \in S$

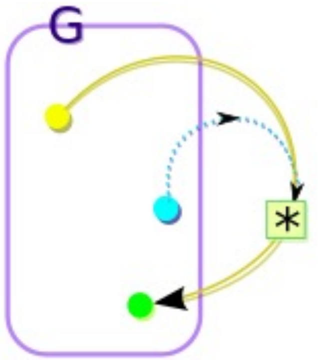
Magma $(S,\oplus)$

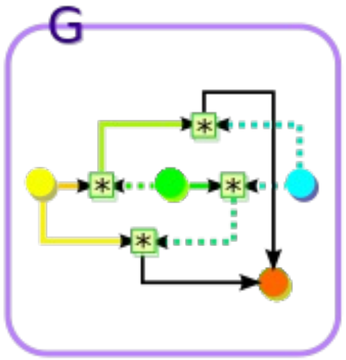
\+ 2. Associativity:
$x\oplus(y\oplus z) = (x\oplus y)\oplus z$

Semigroup $(S,\oplus)$



"In a category associativity is the condition that the two ways to use binary composition of morphisms to compose a sequence of three morphisms are equal"





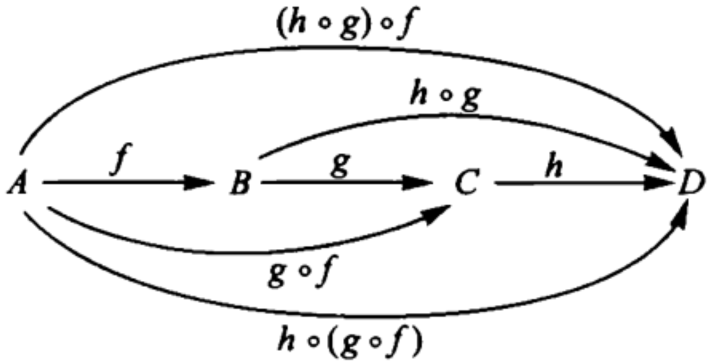


Figure credits: https://www.euclideanspace.com/maths/discrete/groups/monoid/index.htm , https://ncatlab.org/nlab/show/associativity , https://en.wikibooks.org/wiki/Abstract_Algebra/Group_Theory/Group/Definition_of_a_Group

Group-like structures

Set S

+ 1. Closed binary operation $\oplus$:
If $x,y \in S$ then the image $(x \oplus y) \in S$

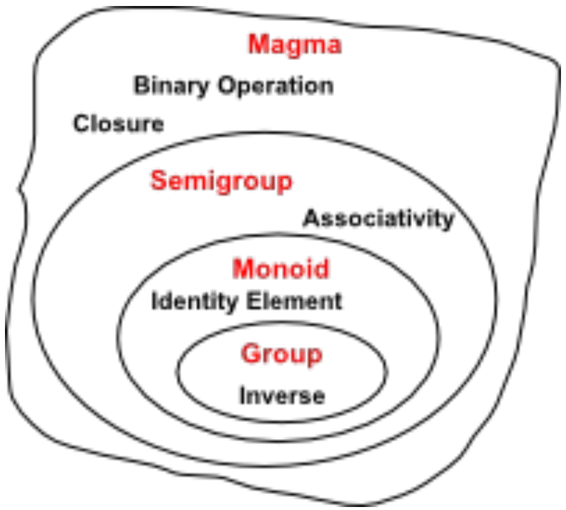
Magma $(S,\oplus)$

+ 2. Associativity:
$x \oplus (y \oplus z) = (x \oplus y) \oplus z$

Semigroup $(S,\oplus)$

+ 3. Identity element:
$\exists e \in S.\ \forall x \in S.\ [e \oplus x = x \oplus e = x]$

Monoid $(S,\oplus,e)$



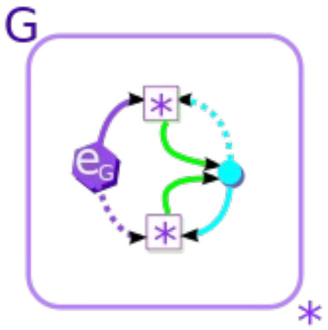


Figure credits: https://www.euclideanspace.com/maths/discrete/groups/monoid/index.htm ,
https://en.wikibooks.org/wiki/Abstract_Algebra/Group_Theory/Group/Definition_of_a_Group

Group-like structures

Set S

+ 1. Closed binary operation $\oplus$:
  If $x,y \in S$ then the image $(x \oplus y) \in S$

Magma $(S,\oplus)$

+ 2. Associativity:
  $x \oplus (y \oplus z) = (x \oplus y) \oplus z$

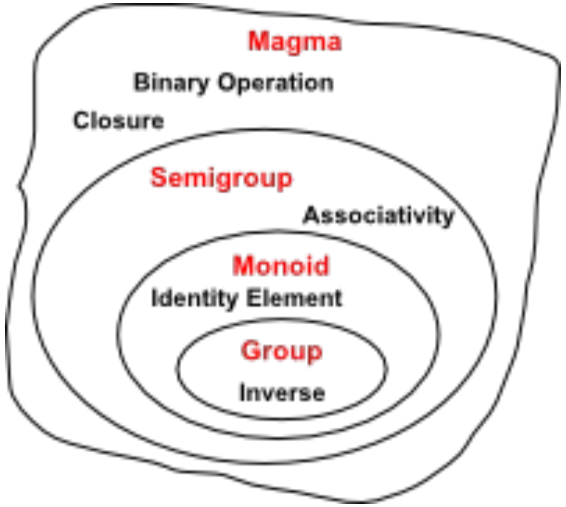
Semigroup $(S,\oplus)$

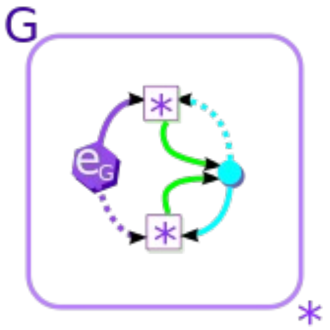
+ 3. Identity element:
  $\exists e \in S.\ \forall x \in S.\ [e \oplus x = x \oplus e = x]$

Monoid $(S,\oplus,e)$

+ 4. Inverse:
  $\forall x \in S.\ \exists x^{-1} \in S.\ [x^{-1} \oplus x = x \oplus x^{-1} = e]$

Group $(S,\oplus,e)$





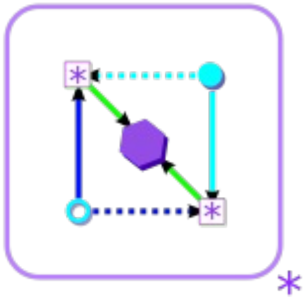


Figure credits: https://www.euclideanspace.com/maths/discrete/groups/monoid/index.htm

Group-like structures

Set S

+ 1. Closed binary operation $\oplus$:
If $x,y \in S$ then the image $(x \oplus y) \in S$

Magma $(S,\oplus)$

+ 2. Associativity:
$x \oplus (y \oplus z) = (x \oplus y) \oplus z$

Semigroup $(S,\oplus)$

+ 3. Identity element:
$\exists e \in S.\ \forall x \in S.\ [e \oplus x = x \oplus e = x]$

Monoid $(S,\oplus,e)$ → Commutative Monoid $(S,\oplus,e)$

+ 5. Commutativity: $x \oplus y = y \oplus x$

+ 4. Inverse:
$\forall x \in S.\ \exists x^{-1} \in S.\ [x^{-1} \oplus x = x \oplus x^{-1} = e]$

+ 4.

Group $(S,\oplus,e)$ → Abelian Group $(S,\oplus,e)$

+ 5.

What are intuitive examples for:

- a group ?
- monoids (that are not groups) ?
- semigroups (that are not monoids)? ?

Group-like structures

Set S

+ 1. Closed binary operation $\oplus$:
If $x,y \in S$ then the image $(x \oplus y) \in S$

Magma $(S,\oplus)$

+ 2. Associativity:
$x \oplus (y \oplus z) = (x \oplus y) \oplus z$

Semigroup $(S,\oplus)$

+ 3. Identity element:
$\exists e \in S.\ \forall x \in S.\ [e \oplus x = x \oplus e = x]$

Monoid $(S,\oplus,e)$ → Commutative Monoid $(S,\oplus,e)$

+ 5. Commutativity: $x \oplus y = y \oplus x$

+ 4. Inverse:
$\forall x \in S.\ \exists x^{-1} \in S.\ [x^{-1} \oplus x = x \oplus x^{-1} = e]$

+ 4.

Group $(S,\oplus,e)$ → Abelian Group $(S,\oplus,e)$

+ 5.

What are intuitive examples for:

- a group
  - $(\mathbb{Z},+,0)$: Integers under addition
- monoids (that are not groups)

?

- semigroups (that are not monoids)?

?

Group-like structures

Set S

+ 1. Closed binary operation ⊕:
If $x,y \in S$ then the image $(x \oplus y) \in S$

Magma (S,⊕)

+ 2. Associativity:
$x \oplus (y \oplus z) = (x \oplus y) \oplus z$

Semigroup (S,⊕)

+ 3. Identity element:
$\exists e \in S.\ \forall x \in S.\ [e \oplus x = x \oplus e = x]$

Monoid (S,⊕,e) → Commutative Monoid (S,⊕,e)

+ 5. Commutativity: $x \oplus y = y \oplus x$

+ 4. Inverse:
$\forall x \in S.\ \exists x^{-1} \in S.\ [x^{-1} \oplus x = x \oplus x^{-1} = e]$

+ 4.

Group (S,⊕,e) → Abelian Group (S,⊕,e)

+ 5.

What are intuitive examples for:

- a group
  - ($\mathbb{Z}$,+, 0): Integers under addition
- monoids (that are not groups)
  - ($\mathbb{N}$,+, 0): Natural numbers under add. {0, 1, ...}
  - ($\mathbb{R}$,min,∞): minimum has no inverse
  - String concatenation with null string ε
  - Square matrices under matrix multiplication
  - (P(S),∪): Power set under union
- semigroups (that are not monoids)?

?

Group-like structures

Set S

+ 1. Closed binary operation ⊕:
If x,y∈S then the image (x⊕y)∈S

Magma (S,⊕)

+ 2. Associativity:
x⊕(y⊕z) = (x⊕y)⊕z

Semigroup (S,⊕)

+ 3. Identity element:
∃e∈S. ∀x∈S. [e⊕x = x⊕e = x]

Monoid (S,⊕,e) → Commutative Monoid (S,⊕,e)
+ 5. Commutativity: x⊕y = y⊕x

+ 4. Inverse:
∀x∈S. ∃x^{-1}∈S. [x^{-1}⊕x = x⊕x^{-1} = e]

+ 4.

Group (S,⊕,e) → Abelian Group (S,⊕,e)
+ 5.

What are intuitive examples for:

- a group
  - (ℤ,+, 0): Integers under addition
- monoids (that are not groups)
  - (ℕ,+, 0): Natural numbers under add. {0, 1, ...}
  - (ℝ,min,∞): minimum has no inverse
  - String concatenation with null string ε
  - Square matrices under matrix multiplication
  - ($\mathcal{P}$(S),∪): Power set under union
- semigroups (that are not monoids)?
  - ($\mathbb{N}_1$,+): Positive integers under add. {1, 2, ...}
  - Even numbers under multiplication
  - String concatenation without null string

What do we exactly lose by not having an inverse?

- Let's take a quick detour and look at some examples to illustrate what we lose by having monoids instead of groups

Monoids vs. Groups: Examples

- Commutative group (with inverse)
  - $(\mathbb{R}, +, 0)$ e.g., $3 + 3^{-1} =$?

Monoids vs. Groups: Examples

- Commutative group (with ~~inverse~~)
  - $(\mathbb{R}, +, 0)$ e.g., $3 + 3^{-1} = 3 + (-3) = 0$ recall: inverse w.r.t. $(+, 0)$
  - $(\mathbb{R}\backslash\{0\}, \cdot\,, 1)$ e.g., $3 \cdot 3^{-1} =$?

Monoids vs. Groups: Examples

- Commutative group (with inverse)
  - $(\mathbb{R}, +, 0)$ e.g., $3 + 3^{-1} = 3 + (-3) = 0$

    recall: inverse w.r.t. (+, 0)
  - $(\mathbb{R}\backslash\{0\}, \cdot, 1)$ e.g., $3 \cdot 3^{-1} = 3 \cdot (1/3) = 1$

- Commutative monoid (w/o inverse)
  - $(\{0,1\},\wedge,1)$ ... logical conjunction
    - identity element 1: $x\wedge 1 = 1\wedge x = x$
    - What is the inverse 0^{-1} s.t. $0\wedge 0^{-1} = 1$

?

Monoids vs. Groups: Examples

- Commutative group (with inverse)
  - $(\mathbb{R}, +, 0)$ e.g., $3 + 3^{-1} = 3 + (-3) = 0$ recall: inverse w.r.t. $(+, 0)$
  - $(\mathbb{R}\backslash\{0\}, \cdot, 1)$ e.g., $3 \cdot 3^{-1} = 3 \cdot (1/3) = 1$

- Commutative monoid (w/o inverse)
  - $(\{0,1\}, \wedge, 1)$ ... logical conjunction
    - identity element 1: $x \wedge 1 = 1 \wedge x = x$
    - What is the inverse 0^{-1} s.t. $0 \wedge 0^{-1} = 1$ There is no such inverse ☹
  - $(\mathbb{R}^{\infty}, \min, \infty)$ $\mathbb{R} \cup \{\infty\}$
    - identity element ∞: $\min[x, \infty] = x$
    - What is the inverse 3^{-1} s.t. $\min[3, 3^{-1}] = \infty$?

Monoids vs. Groups: Examples

- Commutative group (with inverse)
  - $(\mathbb{R}, +, 0)$ e.g., $3 + 3^{-1} = 3 + (-3) = 0$ recall: inverse w.r.t. (+, 0)
  - $(\mathbb{R}\backslash\{0\}, \cdot, 1)$ e.g., $3 \cdot 3^{-1} = 3 \cdot (1/3) = 1$

- Commutative monoid (w/o inverse)
  - $(\{0,1\},\wedge,1)$ ... logical conjunction
    - identity element 1: $x\wedge 1 = 1\wedge x = x$
    - What is the inverse 0^{-1} s.t. $0\wedge 0^{-1} = 1$ There is no such inverse ☹
  - $(\mathbb{R}^{\infty},\min,\infty)$
    - identity element ∞: $\min[x,\infty] = x$
    - What is the inverse 3^{-1} s.t. $\min[3,3^{-1}] = \infty$ There is no such inverse ☹

$\mathbb{R} \cup \{\infty\}$

The power of groups (i.e. of having an inverse)

- Assume(x,y,z) s.t. $x \oplus y = z$
  - Given y and z (and knowing that z was calculated), deduce x
- $(\mathbb{R},+,0)$ and $(x,y,z)=(1,2,3)$
  - $x+2=3$

    What is x? ?

The power of groups (i.e. of having an inverse)

- Assume(x,y,z) s.t. $x \oplus y = z$
  - Given y and z (and knowing that z was calculated), deduce x
- $(\mathbb{R},+,0)$ and $(x,y,z)=(1,2,3)$
  - $x+2=3$

    What is x? $x=z+y^{-1}=3+(-2)=1$
- $(\{0,1\},\wedge,1)$ and $(x,y,z)=(1,0,0)$
  - $x \wedge 0=0$

    What is x? ?

The power of groups (i.e. of having an inverse)

- Assume(x,y,z) s.t. $x \oplus y = z$
  - Given y and z (and knowing that z was calculated), deduce x
- $(\mathbb{R},+,0)$ and $(x,y,z)=(1,2,3)$
  - $x+2=3$

    What is x? $x=z+y^{-1}=3+(-2)=1$
- $(\{0,1\},\wedge,1)$ and $(x,y,z)=(1,0,0)$
  - $x \wedge 0=0$ $\qquad x \wedge 0 = 1$

    What is x? x could be 0 or 1
- $(\mathbb{R}^{\infty},\min,\infty)$ and $(x,y,z)=(3,2,2)$
  - $x \min 2 = 2$

    What is x? ?

The power of groups (i.e. of having an inverse)

- Assume(x,y,z) s.t. $x \oplus y = z$
  - Given y and z (and knowing that z was calculated), deduce x
- $(\mathbb{R},+,0)$ and $(x,y,z)=(1,2,3)$
  - $x+2=3$

    What is x? $x=z+y^{-1}=3+(-2)=1$
- $(\{0,1\},\wedge,1)$ and $(x,y,z)=(1,0,0)$
  - $x \wedge 0=0$

    What is x? x could be 0 or 1
- $(\mathbb{R}^{\infty},\min,\infty)$ and $(x,y,z)=(3,2,2)$
  - $x \min 2 = 2$

    What is x? x can be anything in $[2,\infty]$

Rings and Semirings: what we get from two operators

- Groups and group-like structures consider a set and one binary operator (with various properties)
- Rings and ring-like structures consider a set and two operators (with various properties and "interactions" like the distributive law)

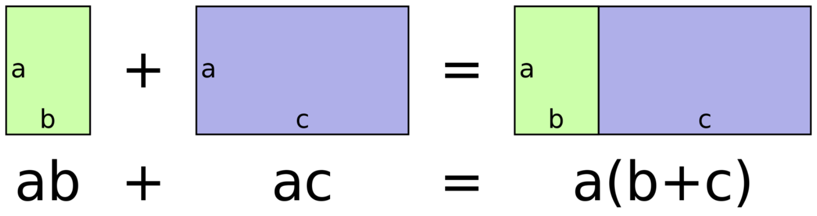
(Commutative) Semirings

two operators w/ neutral elements

- Semiring $(S,\oplus,\otimes,0,1)$
  1. $(S,\oplus,0)$ is commutative monoid
  2. $(S,\otimes,1)$ is (commutative) monoid
  3. $\otimes$ distributes over $\oplus$: $(x\oplus y) \otimes z = (x\otimes z) \oplus (y\otimes z)$
  4. 0 annihilates $\otimes$: $0 \otimes x = 0$

thus semirings are rings w/o the additive inverse (≈ GROUP)

Commutative semirings e.g.: matrix multiplication is not commutative



(Commutative) Semirings

two operators w/ neutral elements

thus semirings are rings w/o the additive inverse (≈ GROUP)

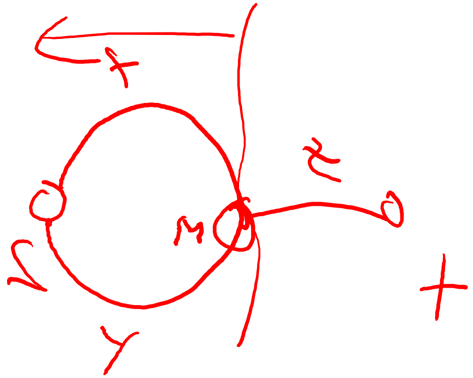
- Semiring $(S, \oplus, \otimes, 0, 1)$
  1. $(S, \oplus, 0)$ is commutative monoid
  2. $(S, \otimes, 1)$ is (commutative) monoid
  3. $\otimes$ distributes over $\oplus$: $(x \oplus y) \otimes z = (x \otimes z) \oplus (y \otimes z)$
  4. 0 annihilates $\otimes$: $0 \otimes x = 0$

Commutative semirings
e.g.: matrix multiplication is not commutative

- Examples

TROPICAL ADDITION MULTIPLICATION

  1. $\mathbb{T}=(\mathbb{R}_+^\infty, \min, +, \infty, 0)$ Shortest-distance: min[x,y] + z = min[(x+z),(y+z)]
     min-sum semiring, also called tropical semiring: sum distributes over min:
     min[x+y]+z = min[x+z,y+z]; e.g. min[3+4]+5 = min[3+5, 4+5] = 8
     not the other way: min[x+y,z] ≠ min[x,z] + min[y,z]; e.g. min[3+4,5] = 5 ≠ 7 =min[3,5] + min[4,5]
  2. $\mathbb{R}=(\mathbb{R}, +, \cdot, 0, 1)$ Ring of real numbers
  3. $\mathbb{B}=(\{0,1\}, \vee, \wedge, 0, 1)$ Boolean (set semantics)
  4. $\mathbb{N}=(\mathbb{N}, +, \cdot, 0, 1)$ Number of paths (bag semantics)
  5. $\mathbb{V}=([0,1], \max, \cdot, 0, 1)$ Probability of best derivation (Viterbi)



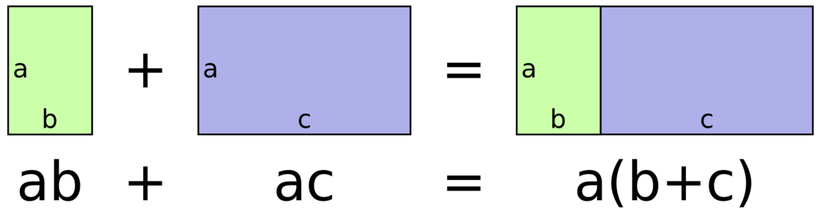


Figure credits: https://en.wikipedia.org/wiki/Distributive_property

Ring-like structures

annihilator,
missing on the left

Field	
Closure	Abelian Group (+)
Associativity	
Identity Element	
Inverse Element	
Commutativity	
Closure	Abelian Group (*)
Associativity	
Identity Element	
Inverse Element	
Commutativity	
Left Distributivity	Distributivity
Right Distributivity	

Ring	
Closure	Abelian Group (+)
Associativity	
Identity Element	
Inverse Element	
Commutativity	
Closure	Monoid (*)
Associativity	
Identity Element	
Inverse Element	
Commutativity	
Left Distributivity	Distributivity
Right Distributivity	

Commutative Ring	
Closure	Abelian Group (+)
Associativity	
Identity Element	
Inverse Element	
Commutativity	
Closure	Commutative Monoid (*)
Associativity	
Identity Element	
Inverse Element	
Commutativity	
Left Distributivity	Distributivity
Right Distributivity	

Pseudo-Ring	
Closure	Abelian Group (+)
Associativity	
Identity Element	
Inverse Element	
Commutativity	
Closure	Semi-Group (*)
Associativity	
Identity Element	
Inverse Element	
Commutativity	
Left Distributivity	Distributivity
Right Distributivity	

Semi-Ring	
Closure	Commutative Monoid (+)
Associativity	
Identity Element	
Inverse Element	
Commutativity	
Closure	~~Commutative Monoid (*)~~
Associativity	
Identity Element	
Inverse Element	
Commutativity	
Left Distributivity	Distributivity
Right Distributivity	

Commutative Semi-Ring	
Closure	Commutative Monoid (+)
Associativity	
Identity Element	
Inverse Element	
Commutativity	
Closure	Commutative Monoid (*)
Associativity	
Identity Element	
Inverse Element	
Commutativity	
Left Distributivity	Distributivity
Right Distributivity	

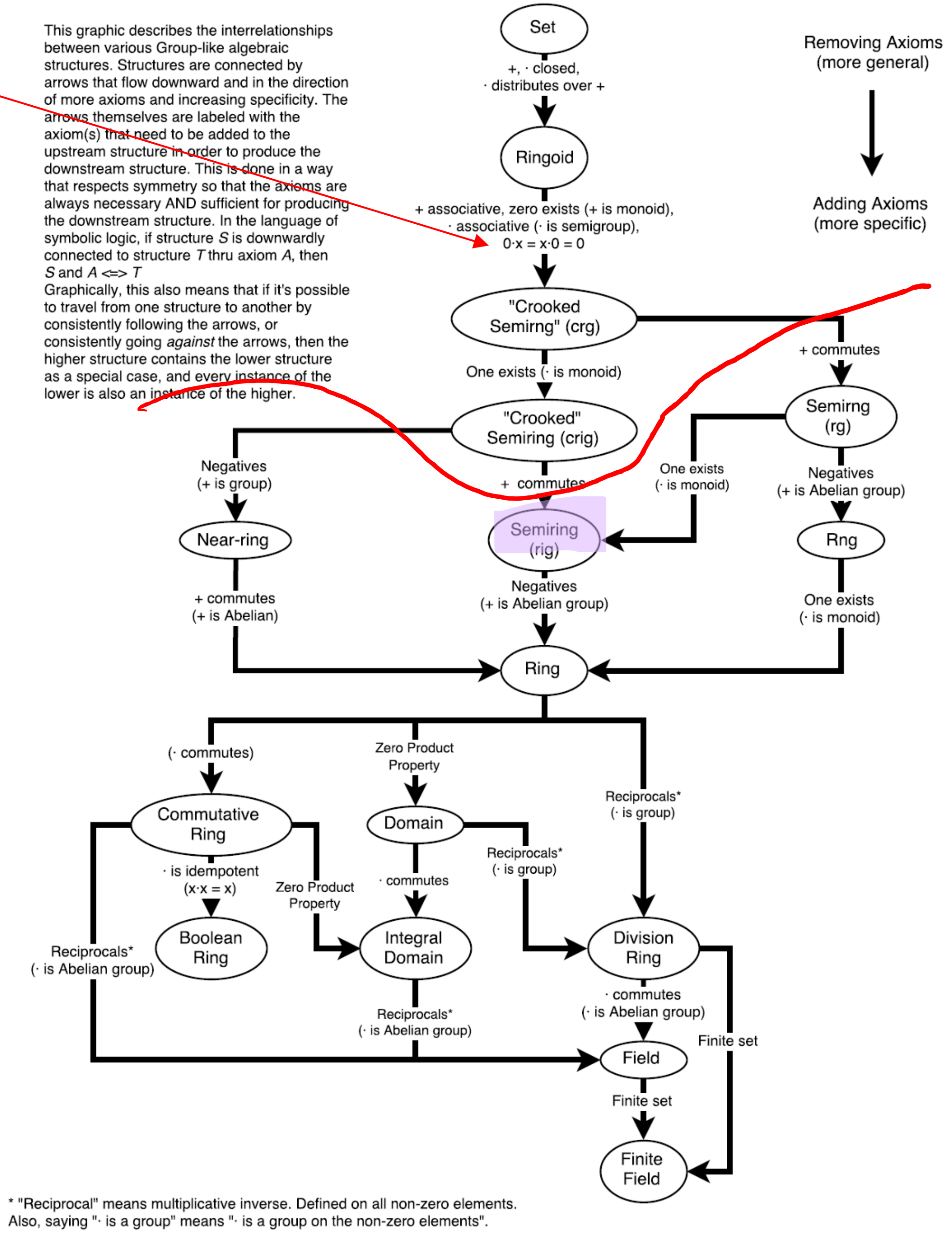
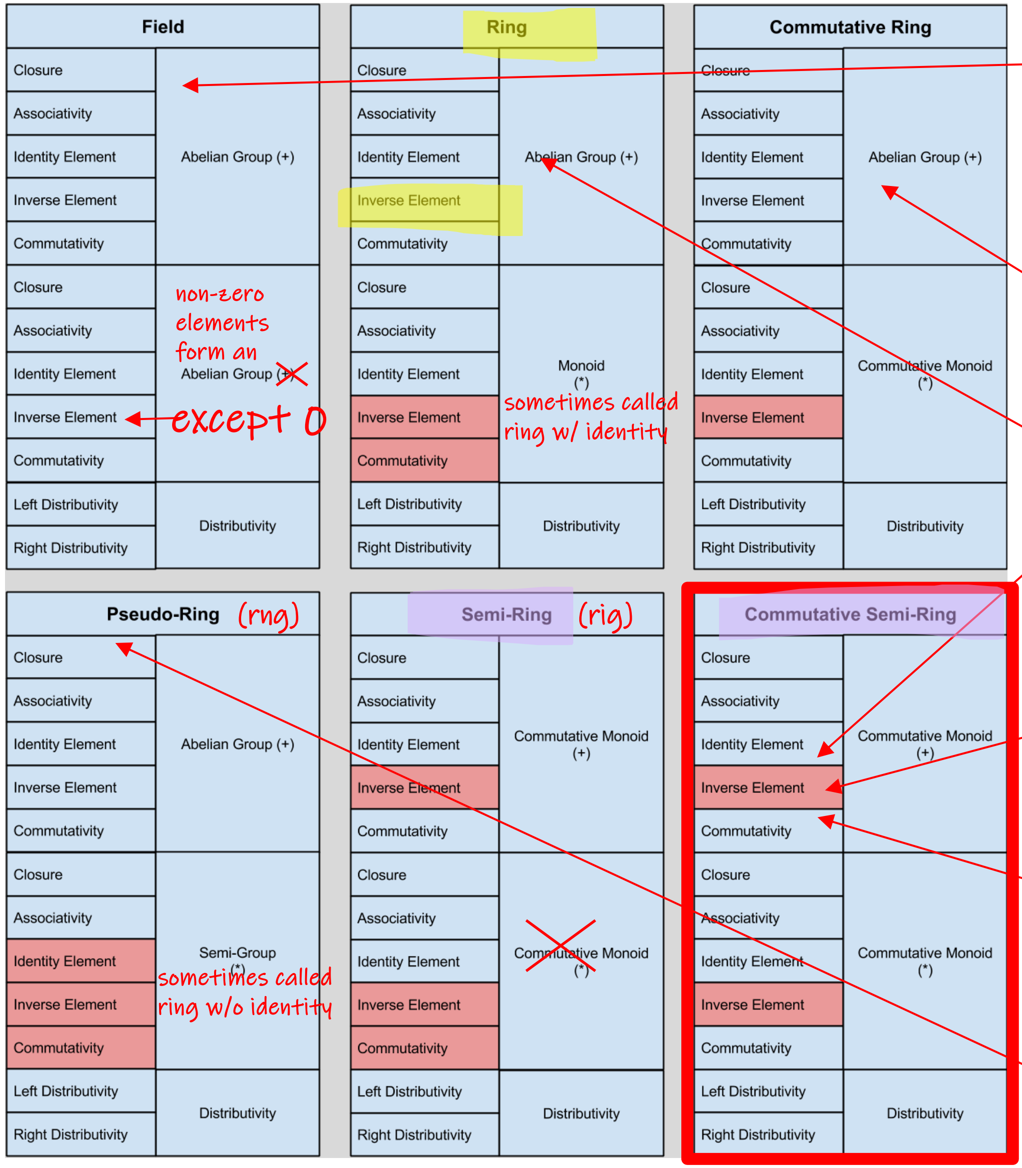


Figure credits: https://kevinbinz.com/2014/11/16/goodman-semiring-parsing/, https://math.stackexchange.com/questions/2361889/graphically-organizing-the-interrelationships-of-basic-algebraic-structures

Ring-like structures

annihilator, missing on the left



$\mathbb{Q}$ (rational numbers)
$\mathbb{Z}/5\mathbb{Z}$ (integers mod 5)
$\dfrac{f(x)}{g(x)}$ field of rational fcts

$\mathbb{R}$[x] real polynomials
$\mathbb{Z}/4\mathbb{Z}$ (integers mod 4)

$\left\{\begin{pmatrix} a & b \\ c & d \end{pmatrix} \middle| a, b, c, d \text{ are integers}\right\}$

$\mathbb{B}$=($\mathbb{B}$, ∨, ∧ , 0, 1): Boolean semiring
1 + 1 = 1, thus ∨ has no inverse

($\mathbb{N}$, +, · , 0, 1): Natural numbers
no inverses

Polynomials with semiring coefficients (e.g. $\mathbb{N}$[x])

2$\mathbb{Z}$: Even integers

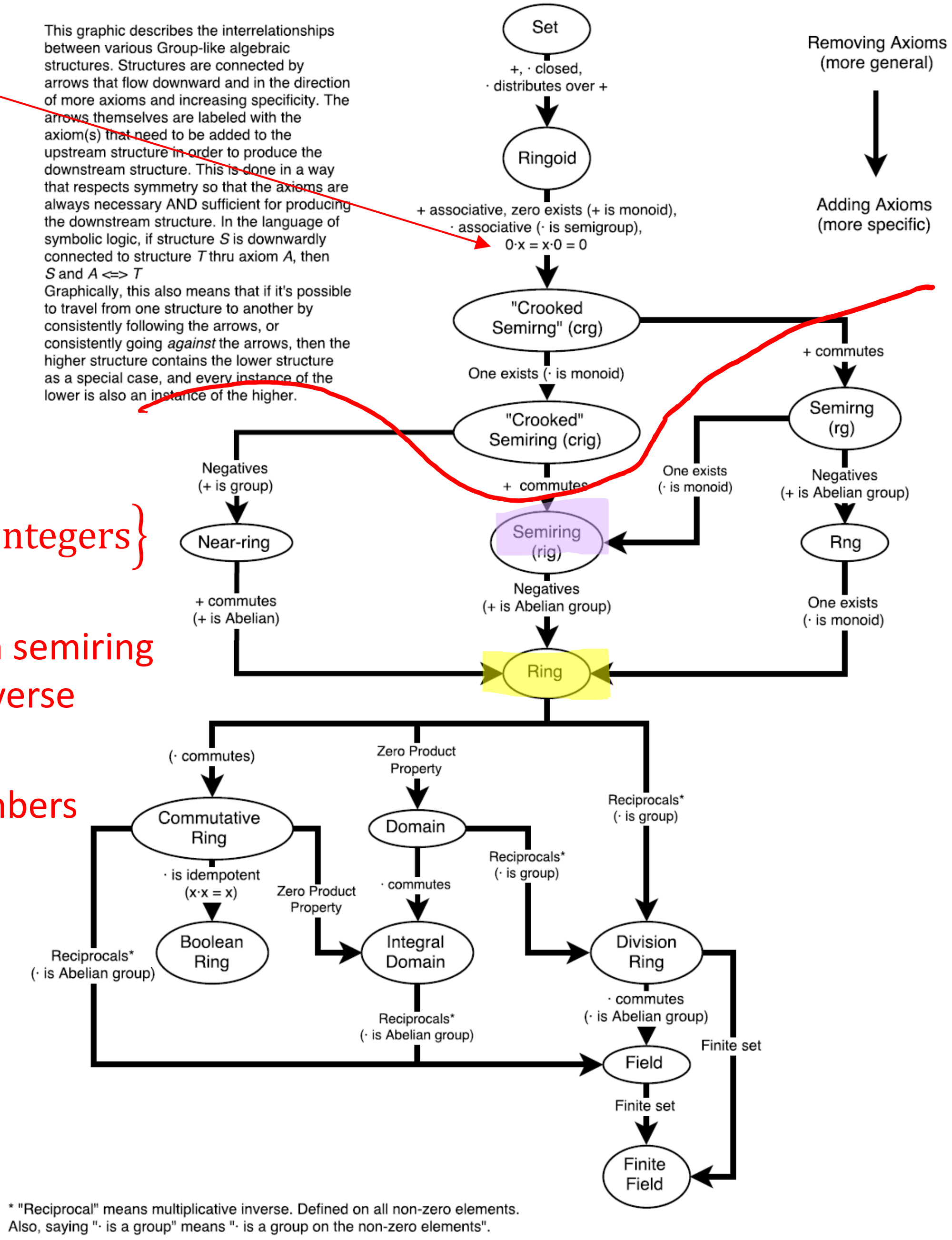


Figure credits: https://kevinbinz.com/2014/11/16/goodman-semiring-parsing/, https://math.stackexchange.com/questions/2361889/graphically-organizing-the-interrelationships-of-basic-algebraic-structures

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

Rings and Semiring homomorphisms

- We have seen homomorphisms for structures with 1 operator:
  - graphs
  - conjunctive queries
  - groups
  - general binary structures
- Semiring homomorphisms generalize this to two operators

RECALL Homomorphisms on Binary Structures

- **Definition (Binary algebraic structure):** A binary algebraic structure is a set together with a binary operation on it. This is denoted by an ordered pair $(S,\star)$ in which S is a set and $\star$ is a binary operation on S.

- **Definition (homomorphism of binary structures):** Let $(S,\star)$ and $(S',\circ)$ be binary structures. A homomorphism from $(S,\star)$ to $(S',\circ)$ is a map $h: S \longrightarrow S'$ that satisfies, for all x, y in S:

  $h(x \star y) = h(x) \circ h(y)$

- We can denote it by $h: (S,\star) \longrightarrow (S',\circ)$.

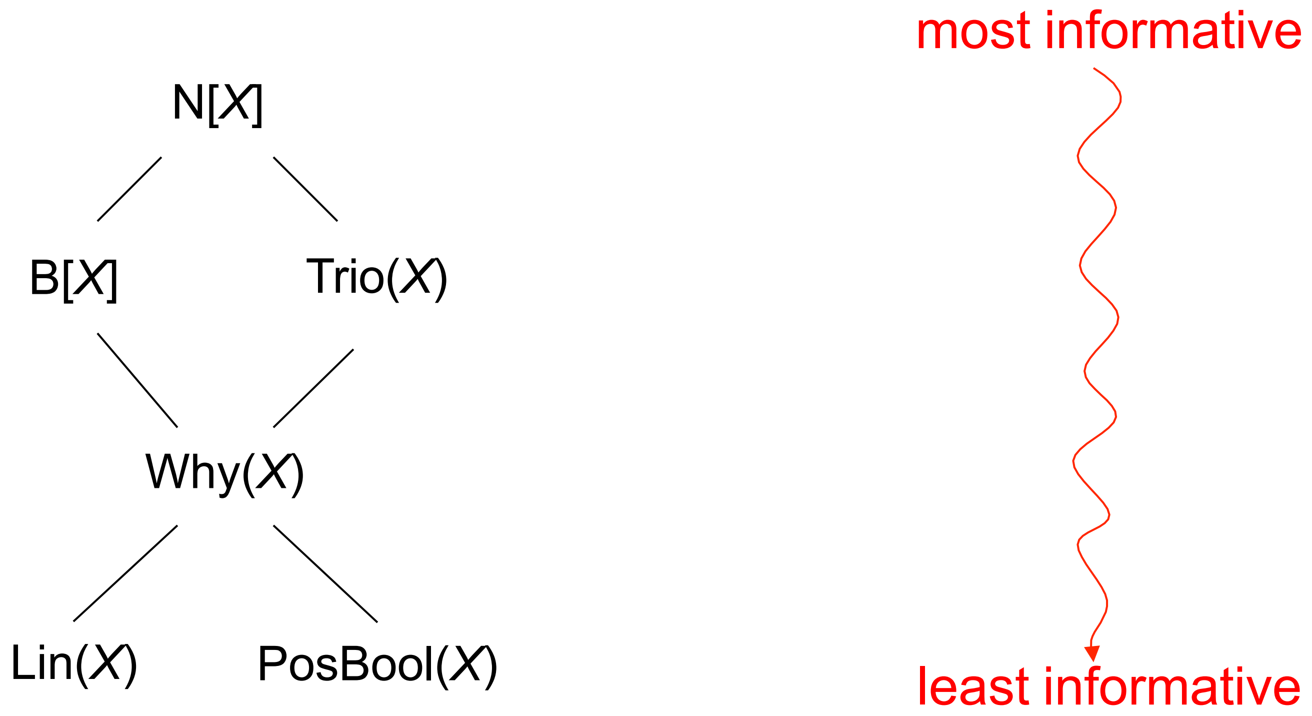
Homomorphisms now for ring-like structures

- A homomorphism between two semirings is a function between their underlying sets that preserves the two operations of addition and multiplication and also their identities.

$\oplus$ $\otimes$

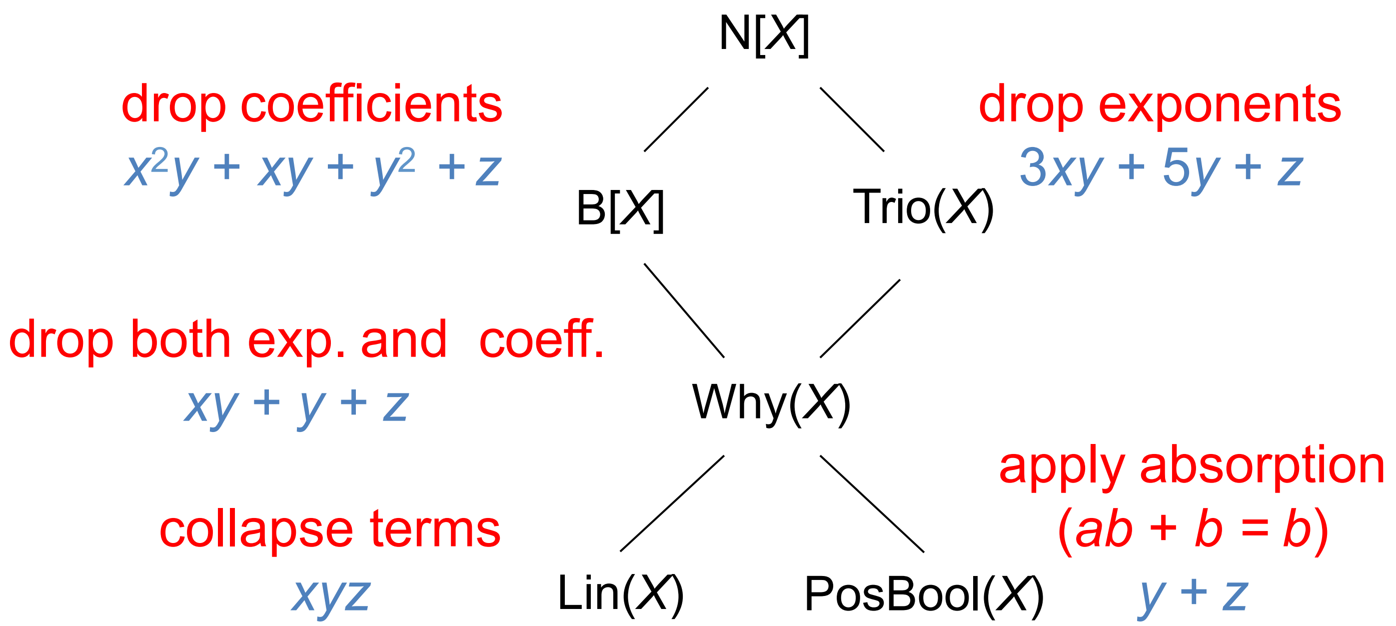
- **Definition (homomorphism between semirings):** Let (R,+,•) and (S,⋆,∘) be semirings. A homomorphism from (R,+,•) to (S,⋆,∘) is a map h: $S \longrightarrow S'$ that satisfies, for all x, y in S:
  - $h(x + y) = h(x) \star h(y)$ addition preserving
  - $h(x \bullet y) = h(x) \circ h(y)$ multiplication preserving
  - $h(1_R) = 1_S$ multiplicative identity preserving
  - $h(0_R) = 0_S$ additive identify preserving

A partial provenance hierarchy



Using homomorphisms to relate models

Example: $2x^2y + xy + 5y^2 + z$



A path downward from K_1 to K_2 indicates that there exists an **onto (surjective) semiring homomorphism** $h : K_1 \rightarrow K_2$
Furthermore, notice that for these homomorphisms $h(x) = x$

The power of Semirings is rediscovered again and again

- Semirings are not "as famous" as rings or groups in abstract algebra, but form the basis of efficient algorithms
  - we often don't need an inverse for the semiring addition
  - we calculate "forward" not backwards (we don't solve equations)
- Thus they are "rediscovered" again and again in various branches of computer science

Power of semirings is rediscovered again and again

1. Bistarelli, Montanari, Rossi. Semiring-Based Constraint Satisfaction and Optimization. JACM 1997 (cited > 800 times, 3/2020)

"We introduce a general framework for constraint satisfaction and optimization where classical CSPs, fuzzy CSPs, weighted CSPs, partial constraint satisfaction, and others can be easily cast. The framework is based on a semiring structure, where the set of the semiring specifies the values to be associated with each tuple of values of the variable domain, and the two semiring operations (1 and 3) model constraint projection and combination respectively. Local consistency algorithms, as usually used for classical CSPs, can be exploited in this general framework as well..."

Power of semirings is rediscovered again and again

2. Aji, McEliece: The generalized distributive law. IEEE Transactions on Information Theory 2000 (cited >950 times in 3/2020)

TABLE I
SOME COMMUTATIVE SEMIRINGS. HERE A DENOTES AN ARBITRARY COMMUTATIVE RING, S IS AN ARBITRARY FINITE SET, AND Λ DENOTES AN ARBITRARY DISTRIBUTIVE LATTICE

	K	"$(+,0)$"	"$(\cdot,1)$"	short name
1.	A	$(+,0)$	$(\cdot,1)$	
2.	$A[x]$	$(+,0)$	$(\cdot,1)$	
3.	$A[x,y,\ldots]$	$(+,0)$	$(\cdot,1)$	
4.	$[0,\infty)$	$(+,0)$	$(\cdot,1)$	sum-product
5.	$(0,\infty]$	$(\min,\infty)$	$(\cdot,1)$	min-product
6.	$[0,\infty)$	$(\max,0)$	$(\cdot,1)$	max-product
7.	$(-\infty,\infty]$	$(\min,\infty)$	$(+,0)$	min-sum
8.	$[-\infty,\infty)$	$(\max,-\infty)$	$(+,0)$	max-sum
9.	$\{0,1\}$	$(\mathrm{OR},0)$	$(\mathrm{AND},1)$	Boolean
10.	2^S	$(\cup,\emptyset)$	$(\cap,S)$	
11.	Λ	$(\vee,0)$	$(\wedge,1)$	
12.	Λ	$(\wedge,1)$	$(\vee,0)$.	

"... we discuss a general message passing algorithm, which we call the generalized distributive law (GDL). The GDL is a synthesis of the work of many authors in the information theory, digital communications, signal processing, statistics, and artificial intelligence communities. It includes as special cases ... Although this algorithm is guaranteed to give exact answers only in certain cases (the "junction tree" condition), ... much experimental evidence, and a few theorems, suggesting that it often works approximately even when it is not supposed to.

Power of semirings is rediscovered again and again

3. Mohri: Semiring frameworks and algorithms for shortest-distance problems. Journal of Automata, Languages and Combinatorics. 2002 (cited 290 times in 3/2020)

"We define general algebraic frameworks for shortest-distance problems based on the structure of semirings. We give a generic algorithm for finding single-source shortest distances in a weighted directed graph when the weights satisfy the conditions of our general semiring framework. ... Classical algorithms such as that of Bellman-Ford [4, 17] are specific instances of this generic algorithm ... The algorithm of Lawler [24] is a specific instance of this algorithm."

the system $(\mathbb{K}, \oplus, \otimes)$ is a semiring

Power of semirings is rediscovered again and again

4. Green, Karvounarakis, Tannen. Provenance semirings. PODS 2007. (PODS 2017 test-of-time award)

Conclusions and Further Work

General and versatile framework.

Dare I call it "semiring-annotated databases"?

Many apparent applications.

We clarified the hazy picture of multiple models for database provenance.

Essential component of the data sharing system Orchestra.

- Dealing with **negation** (progress: [Geerts&Poggi 08, GI&T ICDT 09])
- Dealing with **aggregates** (progress: [T ProvWorkshop 08])
- Dealing with **order** (speculations…)

Paper: Green, Karvounarakis, Tannen. Provenance semirings. PODS 2007. https://doi.org/10.1145/1265530.1265535 , Figure credit: Val Tannen's EDBT 2010 keynote.

Power of semirings is rediscovered again and again

5. Khamis, Ngo, Rudra. FAQ: Questions Asked Frequently. PODS 2016 (PODS 2016 best paper award)

"We define and study the Functional Aggregate Query (FAQ) problem, which encompasses many frequently asked questions in constraint satisfaction, databases, matrix operations, probabilistic graphical models and logic. This is our main conceptual contribution... We then present a simple algorithm called InsideOut to solve this general problem. InsideOut is a variation of the traditional dynamic programming approach for constraint programming based on variable elimination."

Problem	FAQ formulation	Previous Algo.	**Our Algo.**
#QCQ	$\sum_{(x_1,\ldots,x_f)} \bigoplus^{(f+1)}_{x_{f+1}} \cdots \bigoplus^{(n)}_{x_n} \prod_{S\in\mathcal{E}} \psi_S(\mathbf{x}_S)$ where $\bigoplus^{(i)} \in \{\max, \times\}$	No non-trivial algo	$\tilde{O}(N^{\text{faqw}(\varphi)} + \|\varphi\|)$
QCQ	$\bigoplus^{(f+1)}_{x_{f+1}} \cdots \bigoplus^{(n)}_{x_n} \prod_{S\in\mathcal{E}} \psi_S(\mathbf{x}_S)$ where $\bigoplus^{(i)} \in \{\max, \times\}$	$\tilde{O}(N^{\text{PW}(\mathcal{H})} + \|\varphi\|)$ [24]	$\tilde{O}(N^{\text{faqw}(\varphi)} + \|\varphi\|)$
#CQ	$\sum_{(x_1,\ldots,x_f)} \max_{x_{f+1}} \cdots \max_{x_n} \prod_{S\in\mathcal{E}} \psi_S(\mathbf{x}_S)$	$\tilde{O}(N^{\text{DM}(\mathcal{H})} + \|\varphi\|)$ [34]	$\tilde{O}(N^{\text{faqw}(\varphi)} + \|\varphi\|)$
Joins	$\bigcup_{\mathbf{x}} \bigcap_{S\in\mathcal{E}} \psi_S(\mathbf{x}_S)$	$\tilde{O}\left(N^{\text{fhtw}(\mathcal{H})} + \|\varphi\|\right)$ [46]	$\tilde{O}\left(N^{\text{fhtw}(\mathcal{H})} + \|\varphi\|\right)$
Marginal	$\sum_{(x_{f+1},\ldots,x_n)} \prod_{S\in\mathcal{E}} \psi_S(\mathbf{x}_S)$	$\tilde{O}(N^{\text{htw}(\varphi)} + \|\varphi\|)$ [54]	$\tilde{O}(N^{\text{faqw}(\varphi)} + \|\varphi\|)$
MAP	$\max_{(x_{f+1},\ldots,x_n)} \prod_{S\in\mathcal{E}} \psi_S(\mathbf{x}_S)$	$\tilde{O}(N^{\text{htw}(\varphi)} + \|\varphi\|)$ [54]	$\tilde{O}(N^{\text{faqw}(\varphi)} + \|\varphi\|)$
MCM	$\sum_{x_2,\ldots,x_n} \prod_{i=1}^{n} \psi_{i,i+1}(x_i, x_{i+1})$	DP bound [28]	DP bound
DFT	$\sum_{(y_0,\ldots,y_{m-1})\in\mathbb{Z}_p^m} b_y \cdot \prod_{0\le j+k<m} e^{i2\pi \frac{x_j \cdot y_k}{p^{m-j-k}}}$	$O(N \log_p N)$ [27]	$O(N \log_p N)$

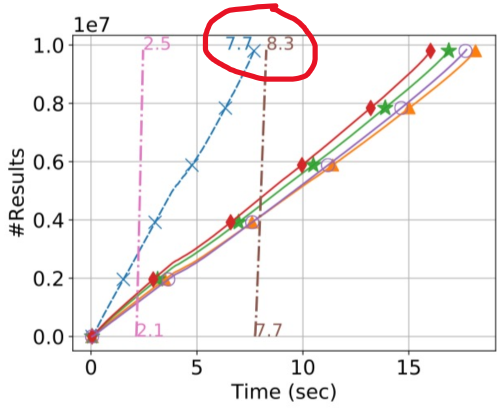
Power of semirings is rediscovered again and again

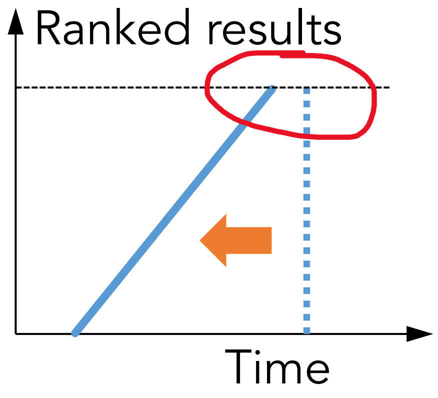
6. Tziavelis+. Optimal Algorithms for Ranked Enumeration of Answers to Full Conjunctive Queries. PVLDB 2020

ABSTRACT

We study ranked enumeration of join-query results according to very general orders defined by selective dioids. Our main contribution is a framework for ranked enumeration over a class of dynamic programming problems that generalizes seemingly different problems that had been studied in isolation. To this end, we extend classic algorithms that find the k-shortest paths in a weighted graph. For full conjunctive queries, including cyclic ones, our approach is optimal in terms of the time to return the top result and the delay between results. These optimality properties are de-

Generality. Our approach supports any selective dioid, including less obvious cases such as *lexicographic ordering* where two output tuples are first compared on their R_1 component, and if equal then on their R_2 component, and so on.





k-shortest paths. The literature is rich in algorithms for finding the k-shortest paths in general graphs [10, 17, 34, 35, 53, 56, 57, 59, 65, 68, 67, 93]. Many of the subtleties of the variants arise from issues caused by cyclic graphs whose structure is more general than the acyclic multi-stage graphs in our DP problems. Hoffman and Pavley [53] introduces the concept of "deviations" as a sufficient condition for finding the k^{th} shortest path. Building on that idea, Dreyfus [34] proposes an algorithm that can be seen as a modification to the procedure of Bellman and Kalaba [17]. The *Recursive Enumeration Algorithm* (REA) [57] uses the same set of equations as Dreyfus, but applies them in a top-down recursive manner. Our ANYK-REC builds upon REA. To the best of our knowledge, prior work has ignored the fact that this algorithm reuses computation in a way that can asymptotically outperform sorting in some cases. In another line of research, Lawler [65] generalizes an earlier algorithm of Murty [70] and applies it to k-shortest paths. Aside from k-shortest paths, the Lawler procedure has been widely used for a variety of problems in the database community [40]. Along with the Hoffman-Pavley deviations, they are one of the main ingredients of our ANYK-PART approach. Eppstein's algorithm [35, 56] achieves the best known asymptotical complexity, albeit with a complicated construction *whose practical performance is unknown.* His "basic" version of the algorithm has the same complexity as EAGER, while our TAKE2 algorithm matches the complexity of the "advanced" version for our problem setting where output tuples are materialized explicitly.

Power of semirings is rediscovered again and again

6. Tziavelis+. Optimal Algorithms for Ranked Enumeration of Answers to Full Conjunctive Queries. PVLDB 2020

2.2 Ranked Enumeration Problem

We want to order the results of a full CQ based on the weights of their corresponding witnesses. For maximal generality, we define ordering based on *selective dioids* [41], which are semirings with an ordering property:

DEFINITION 3 (SEMIRING). *A* monoid *is a 3-tuple* $(W, \oplus, \bar{0})$ *where* W *is a non-empty set,* $\oplus : W \times W \to W$ *is an associative operation, and* $\bar{0}$ *is the identity element, i.e.,* $\forall x \in W : x \oplus \bar{0} = \bar{0} \oplus x = x$*. In a commutative monoid,* $\oplus$ *is also commutative. A* semiring *is a 5-tuple* $(W, \oplus, \otimes, \bar{0}, \bar{1})$*, where* $(W, \oplus, \bar{0})$ *is a commutative monoid,* $(W, \otimes, \bar{1})$ *is a monoid,* $\otimes$ *distributes over* $\oplus$*, i.e.,* $\forall x, y, z \in W : (x \oplus y) \otimes z = (x \otimes z) \oplus (y \otimes z)$*, and* $\bar{0}$ *is absorbing for* $\otimes$*, i.e.,* $\forall a \in W : a \otimes \bar{0} = \bar{0} \otimes a = \bar{0}$.

DEFINITION 4 (SELECTIVE DIOID). *A* selective dioid *is a semiring for which* $\oplus$ *is* selective*, i.e., it always returns one of the inputs:* $\forall x, y \in W : (x \oplus y = x) \vee (x \oplus y = y)$.

Note that $\oplus$ being selective induces a total order on W by setting $x \leq y$ iff $x \oplus y = x$. We define result weight as an aggregate of input-tuple weights using $\otimes$:

Ranked enumeration. Both [26] and [90] provide any-k algorithms for *graph queries* instead of the more general CQs; they describe the ideas behind LAZY and ALL respectively. [60] gives an any-k algorithm for acyclic queries with polynomial delay. Similar algorithms have appeared for the equivalent Constraint Satisfaction Problem (CSP) [44, 50]. These algorithms fit into our family ANYK-PART, yet do not exploit common structure between sub-problems hence have weaker asymptotic guarantees for delay than any of the any-k algorithms discussed here. After we introduced the general idea of ranked enumeration over *cyclic* CQs based on multiple tree decompositions [91], an unpublished paper [33] on arXiv proposed an algorithm for it. Without realizing it, the authors reinvented the REA algorithm [57], which corresponds to RECURSIVE, for that specific context. We are the first to *guarantee optimal time-to-first result and optimal delay for both acyclic and cyclic* queries. For instance, we return the top-ranked result of a 4-cycle in $\mathcal{O}(n^{1.5})$, while [33] requires $\mathcal{O}(n^2)$. Furthermore, our work (1) addresses the more general problem of ranked enumeration for DP over a union of trees, (2) unifies several approaches that have appeared in the past, from graph-pattern search to k-shortest path, and shows that neither dominates all others, (3) provides a theoretical and experimental evaluation of trade-offs including algorithms that perform best for small k, and (4) is the first to prove that it is possible to achieve a time-to-last that asymptotically improves over batch processing by exploiting the stage-wise structure of the DP problem.

Power of semirings is rediscovered again and again

7. Atserias, Kolaitis. Structure and Complexity of Bag Consistency. PODS 2021, SIGMOD record 2022.

Consistency, Acyclicity, and Positive Semirings

Albert Atserias[1] and Phokion G. Kolaitis[2]

[1]Universitat Politècnica de Catalunya
[2]University of California Santa Cruz and IBM Research

September 20, 2020

Structure and Complexity of Bag Consistency

Albert Atserias
Universitat Politècnica de Catalunya
Barcelona, Catalonia
Spain
atserias@cs.upc.edu

Phokion G. Kolaitis
UC Santa Cruz and IBM Research
Santa Cruz, California
USA
kolaitis@ucsc.edu

It appears that Beeri et al. [9] were unaware of Vorob'ev work, but later on Vorob'ev's work was cited in a survey of database theory by Yannakakis [27]. In recent years, the interplay between local consistency and global consistency has been explored at great depth in the setting of quantum information by Abramsky and his collaborators (see, e.g., [3, 4, 5]). In that setting, the interest is in contextuality phenomena, which are situations where collections of measurements are locally consistent but globally inconsistent - Bell's celebrated theorem [10] is an instance of this. The similarities between these different settings (probability distributions, relational databases, and quantum mechanics) were pointed out explicitly by Abramsky [1, 2]. This also raised the question of developing a unifying framework in which, among other things, the results by Vorob'ev and the results by Beeri et al. are special cases of a single result. Using a relaxed notion of consistency, we established such a result for relations over semirings [6]. For the bag semiring, however, the relaxed notion of consistency that we studied in [6] is essentially equivalent to the consistency of probability distributions with rational values (and not to the consistency of bags). This left open the question of exploring the interplay between (the standard notions of) local consistency and global consistency for bags, which is what we set to do in the present paper.

Multiplying 2×2 matrices

$\oplus$ $\otimes$

MIN +

$$\begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$$

n n

$$\begin{aligned} C_{11} &= A_{11}B_{11} + A_{12}B_{21} \\ C_{12} &= A_{11}B_{12} + A_{12}B_{22} \\ C_{21} &= A_{21}B_{11} + A_{22}B_{21} \\ C_{22} &= A_{21}B_{12} + A_{22}B_{22} \end{aligned}$$

8 multiplications
4 additions

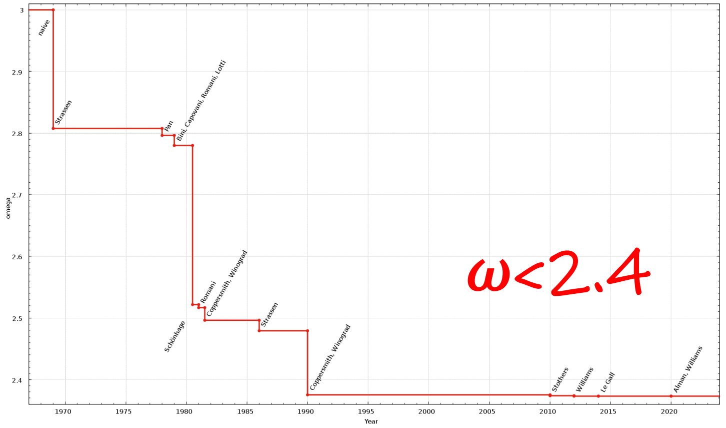
2^3 $O(n^3)$

Works over any semi-ring!

Strassen. Gaussian Elimination is not Optimal. Numerical Mathematics, 1969. https://doi.org/10.1007/BF02165411
https://en.wikipedia.org/wiki/Strassen_algorithm, https://en.wikipedia.org/wiki/Matrix_multiplication_algorithm
Wolfgang Gatterbauer. Principles of scalable data management: https://northeastern-datalab.github.io/cs7240/

Strassen's 2×2 algorithm

Matrix multiplication exponent ω



$\omega < 2.4$

"galactic"

$$C_{11} = A_{11}B_{11} + A_{12}B_{21}$$
$$C_{12} = A_{11}B_{12} + A_{12}B_{22}$$
$$C_{21} = A_{21}B_{11} + A_{22}B_{21}$$
$$C_{22} = A_{21}B_{12} + A_{22}B_{22}$$

$$M_1 = (A_{11} + A_{22})(B_{11} + B_{22})$$
$$M_2 = (A_{21} + A_{22})B_{11}$$
$$M_3 = A_{11}(B_{12} - B_{22})$$
$$M_4 = A_{22}(B_{21} - B_{11})$$
$$M_5 = (A_{11} + A_{12})B_{22}$$
$$M_6 = (A_{21} - A_{11})(B_{11} + B_{12})$$
$$M_7 = (A_{12} - A_{22})(B_{21} + B_{22})$$

Subtraction!

$$C_{11} = M_1 + M_4 - M_5 + M_7$$
$$C_{12} = M_3 + M_5$$
$$C_{21} = M_2 + M_4$$
$$C_{22} = M_1 - M_2 + M_3 + M_6$$

$O(n^\omega)$

7 multiplications

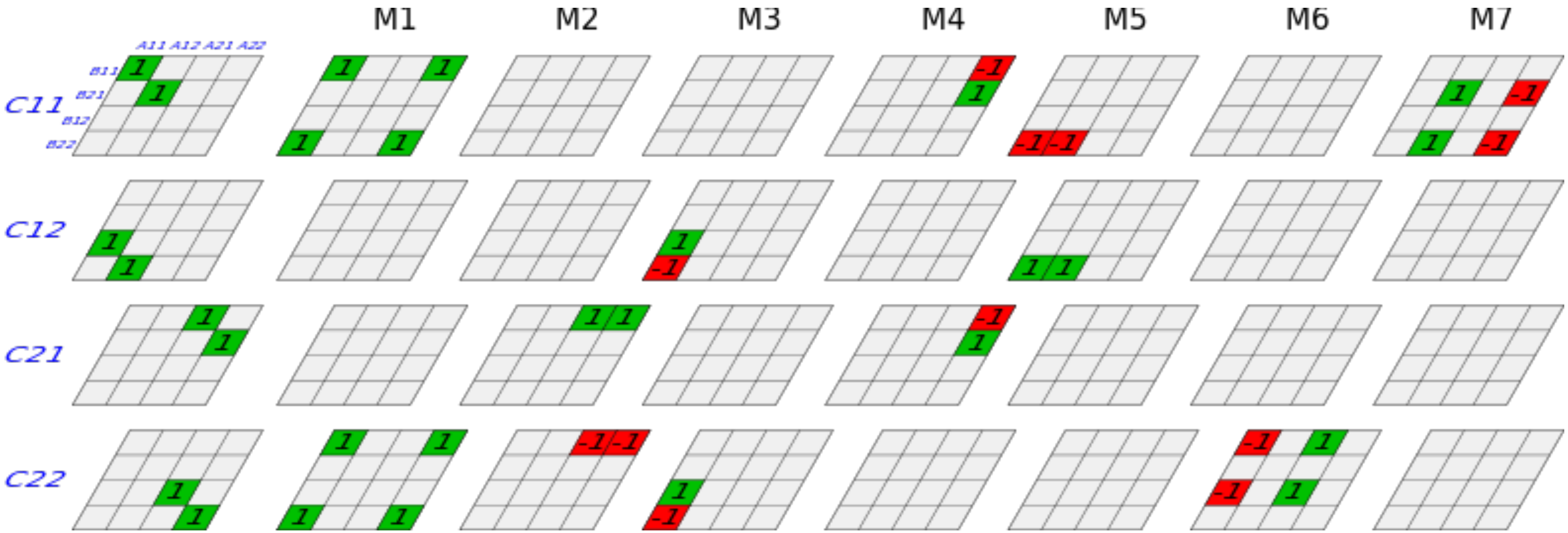
18 additions/subtractions

Works over any ring!

(requirees additive inverse, but does not assume multiplication to be commutative)

Strassen. Gaussian Elimination is not Optimal. Numerical Mathematics, 1969. https://doi.org/10.1007/BF02165411

https://en.wikipedia.org/wiki/Strassen_algorithm, https://en.wikipedia.org/wiki/Matrix_multiplication_algorithm



https://en.wikipedia.org/wiki/Strassen_algorithm

Table 1. Strassen's Algorithm

Phase 1	$T_1 = A_{11} + A_{22}$	$T_6 = B_{11} + B_{22}$
	$T_2 = A_{21} + A_{22}$	$T_7 = B_{12} - B_{22}$
	$T_3 = A_{11} + A_{12}$	$T_8 = B_{21} - B_{11}$
	$T_4 = A_{21} - A_{11}$	$T_9 = B_{11} + B_{12}$
	$T_5 = A_{12} - A_{22}$	$T_{10} = B_{21} + B_{22}$
Phase 2	$Q_1 = T_1 \times T_6$	$Q_5 = T_3 \times B_{22}$
	$Q_2 = T_2 \times B_{11}$	$Q_6 = T_4 \times T_9$
	$Q_3 = A_{11} \times T_7$	$Q_7 = T_5 \times T_{10}$
	$Q_4 = A_{22} \times T_8$	
Phase 3	$T_1 = Q_1 + Q_4$	$T_3 = Q_3 + Q_1$
	$T_2 = Q_5 - Q_7$	$T_4 = Q_2 - Q_6$
Phase 4	$C_{11} = T_1 - T_2$	$C_{12} = Q_3 + Q_5$
	$C_{21} = Q_2 + Q_4$	$C_{22} = T_3 - T_4$

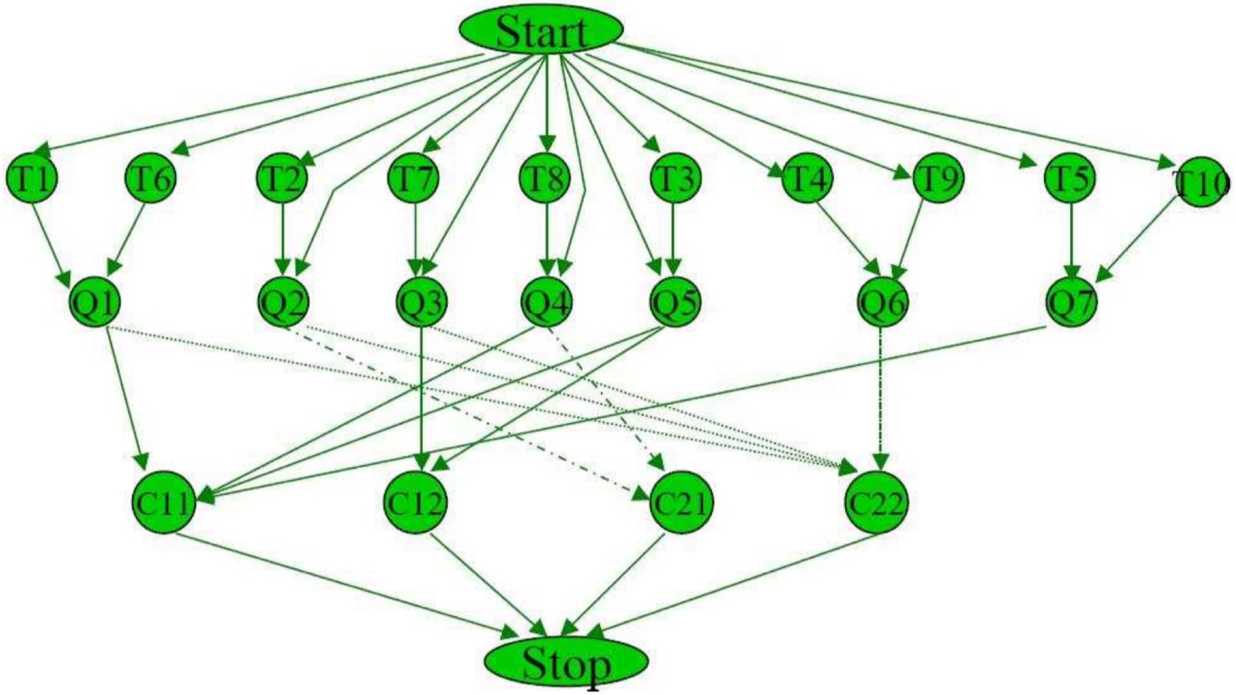
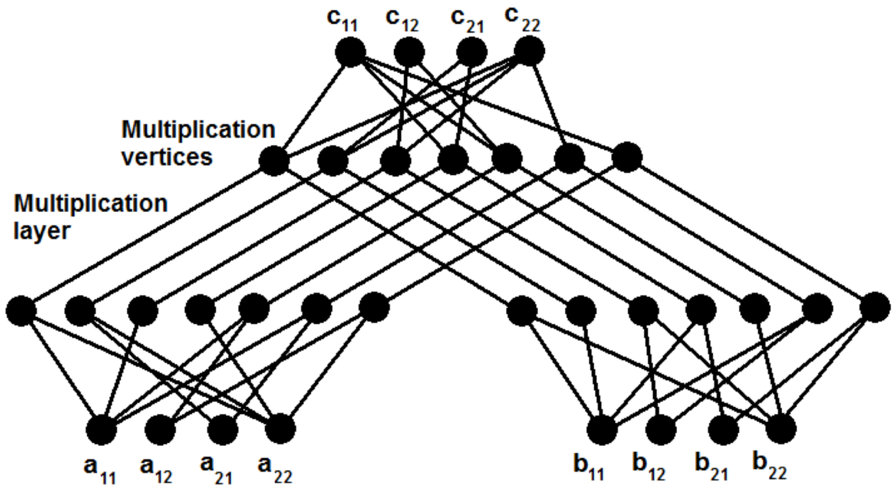
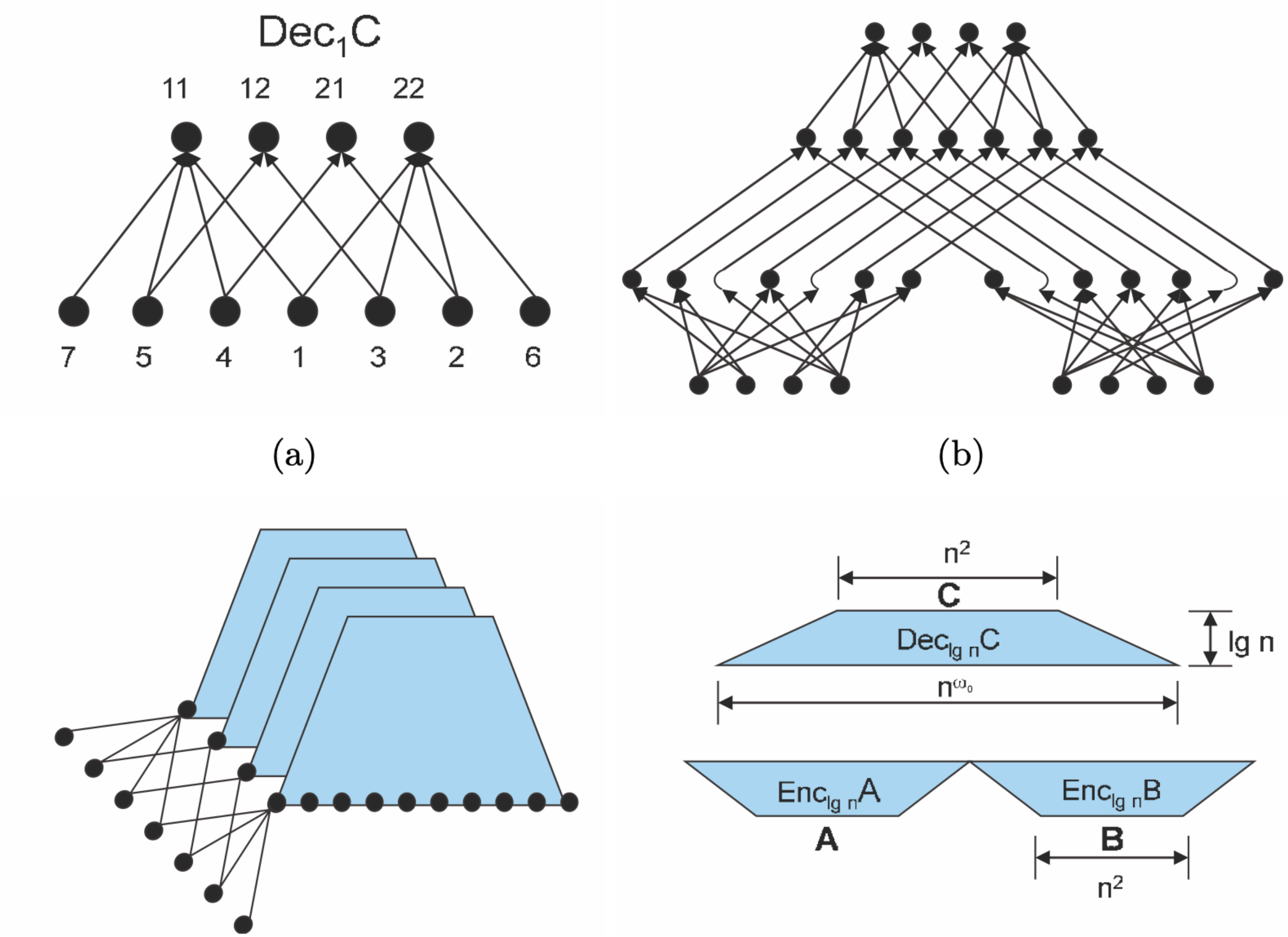


Figure 4. Task graph of Strassen's Algorithm.

Figure 1: The base graph G_1 of Strassen's algorithm for multiplying two 2×2 matrices A and B. Here $b = 7$.





(a) (b) (c) (d)

Figure 4.1. The computation graph of Strassen's algorithm (see Algorithm 4.1): (a) $\mathrm{Dec}_1\, C$, (b) H_1, (c) $\mathrm{Dec}_{\lg n}\, C$, (d) $H_{\lg n}$.

Updated 3/23/2024

Topic 2: Complexity of Query Evaluation
Unit 3: Provenance
Lecture 18

Wolfgang Gatterbauer

CS7240 Principles of scalable data management (sp24)

https://northeastern-datalab.github.io/cs7240/sp24/

3/22/2024

Pre-class conversations

- Last class summary
- Projects: TUE 3/26 intermediate report

- Today:
  - provenance at different granularities (cell level)
  - reverse data management

Outline: T2-3: Provenance

- T2-3: Provenance
  - Data Provenance
  - The Semiring Framework for Provenance
  - Algebra: Monoids and Semirings
  - Query-rewrite-insensitive provenance

Queries & provenance

Agencies

	name	based_in	phone
t_1:	BayTours	San Francisco	415-1200
t_2:	HarborCruz	Santa Cruz	831-3000

ExternalTours

	name	destination	type	price
t_3:	BayTours	San Francisco	cable car	\$50
t_4:	BayTours	Santa Cruz	bus	\$100
t_5:	BayTours	Santa Cruz	boat	\$250
t_6:	BayTours	Monterey	boat	\$400
t_7:	HarborCruz	Monterey	boat	\$200
t_8:	HarborCruz	Carmel	train	\$90

Q_1:

DISTINCT

SELECT a.name, a.phone
FROM Agencies a, ExternalTours e
WHERE a.name = e.name AND e.type='boat'

?

Queries & provenance

Agencies

	name	based_in	phone
t_1:	BayTours	San Francisco	415-1200
t_2:	HarborCruz	Santa Cruz	831-3000

ExternalTours

	name	destination	type	price
t_3:	BayTours	San Francisco	cable car	\$50
t_4:	BayTours	Santa Cruz	bus	\$100
t_5:	BayTours	Santa Cruz	boat	\$250
t_6:	BayTours	Monterey	boat	\$400
t_7:	HarborCruz	Monterey	boat	\$200
t_8:	HarborCruz	Carmel	train	\$90

Q_1:
SELECT a.name, a.phone
FROM Agencies a, ExternalTours e
WHERE a.name = e.name AND e.type='boat'

Result of Q_1:

name	phone
$\boxed{\text{BayTours}}$	$\boxed{\text{415-1200}}$
HarborCruz	831-3000

Lineage = ?

Definition Lineage:
Lineage for an output tuple t is a subset of the input tuples which are relevant to the output tuple

Queries & provenance

Agencies

	name	based_in	phone
t_1:	BayTours	San Francisco	415-1200
t_2:	HarborCruz	Santa Cruz	831-3000

ExternalTours

	name	destination	type	price
t_3:	BayTours	San Francisco	cable car	\$50
t_4:	BayTours	Santa Cruz	bus	\$100
t_5:	BayTours	Santa Cruz	boat	\$250
t_6:	BayTours	Monterey	boat	\$400
t_7:	HarborCruz	Monterey	boat	\$200
t_8:	HarborCruz	Carmel	train	\$90

Q_1:
SELECT a.name, a.phone
FROM Agencies a, ExternalTours e
WHERE a.name = e.name AND e.type='boat'

Result of Q_1:

name	phone
BayTours	415-1200
HarborCruz	831-3000

Lineage = $\{t_1, t_5, t_6\}$

Definition Lineage:
Lineage for an output tuple t is a subset of the input tuples which are relevant to the output tuple

Problem: Not very precise.
e.g., lineage above does not specify that t_5 and t_6 do not both need to exist.

"Why Provenance" & Witnesses

Agencies

	name	based_in	phone
t_1:	BayTours	San Francisco	415-1200
t_2:	HarborCruz	Santa Cruz	831-3000

ExternalTours

	name	destination	type	price
t_3:	BayTours	San Francisco	cable car	\$50
t_4:	BayTours	Santa Cruz	bus	\$100
t_5:	BayTours	Santa Cruz	boat	\$250
t_6:	BayTours	Monterey	boat	\$400
t_7:	HarborCruz	Monterey	boat	\$200
t_8:	HarborCruz	Carmel	train	\$90

Q_1:
SELECT a.name, a.phone
FROM Agencies a, ExternalTours e
WHERE a.name = e.name AND e.type='boat'

Result of Q_1:

name	phone
BayTours	415-1200
HarborCruz	831-3000

Lineage = $\{t_1, t_5, t_6\}$

{t1, t5} {t1, t6} {t1, t2, t6, t8}

Definition Witness of t:
Any subset of the database sufficient to reconstruct tuple t in the query result

{{t1, t5}, {t1, t6}}

Witness basis:
Leaves of the "proof tree" showing how result tuple t is generated

Example taken from "Provenance in databases: why, how, and where" by Cheney, Chiticariu, Tan, 2009. https://doi.org/10.1561/1900000006
Wolfgang Gatterbauer. Principles of scalable data management: https://northeastern-datalab.github.io/cs7240/

Minimality & query rewriting

Minimal witness basis:
Minimal witnesses in the witness basis

Instance I:

R

	A	B
t:	1	2
t':	1	3
t'':	4	2

Two equivalent queries:

$Q : Ans(x,y) :- R(x,y).$

$Q' : Ans(x,y) :- R(x,y), R(x,z).$

Output of $Q(I)$, $Q'(I)$:

A	B
1	2
1	3
4	2

Fig. 1.2 Example queries, input and output.

Instance I:

R

	A	B
t:	1	2
t':	1	3
t'':	4	2

Output of $Q(I)$

A	B	*why*
1	2	$\{\{t\}\}$
1	3	$\{\{t'\}\}$
4	2	$\{\{t''\}\}$

Output of $Q'(I)$

A	B	*why*
1	2	$\{\{t\},\{t,t'\}\}$
1	3	$\{\{t'\},\{t,t'\}\}$
4	2	$\{\{t''\}\}$

Fig. 1.3 Example showing that why-provenance is sensitive to query rewriting.

Instance I:

R

	A	B
t:	1	2
t':	1	3
t'':	4	2

Output of $Q(I)$

A	B	*how*
1	2	t
1	3	t'
4	2	t''

Output of $Q'(I)$

A	B	*how*
1	2	$t^2 + t \cdot t'$
1	3	$(t')^2 + t \cdot t'$
4	2	$(t'')^2$

Fig. 1.5 Example showing that how-provenance is sensitive to query rewriting.

Figures from Cheney, Chiticariu, Tan. Provenance in databases: why, how, and where. Foundations and trends in databases 2009. https://dl.acm.org/doi/abs/10.1561/1900000006

Fixing query-rewrite sensitivity for where provenance

data value
is stored y
single

Instance I:

R

	A	B
t:	1	2
t':	1	3
t'':	4	2

Two equivalent queries:

$Q : Ans(x,y) :\!- \; R(x,y).$

$Q' : Ans(x,y) :\!- \; R(x,y), R(x,z).$

Output of $Q(I)$, $Q'(I)$:

A	B
1	2
1	3
4	2

Fig. 1.2 Example queries, input and output.

Annotated instance I^a:

R

	A	B
t:	1^{a_1}	2^{a_2}
t':	1^{a_3}	3^{a_4}
t'':	4^{a_5}	2^{a_6}

Output of $Q(I^a)$ (DEFAULT propagation):

A	B
1^{a_1}	2^{a_2}
1^{a_3}	3^{a_4}
4^{a_5}	2^{a_6}

Output of $Q'(I^a)$ (DEFAULT propagation):

A	B
$1^{a_1,a_3}$	2^{a_2}
$1^{a_1,a_3}$	3^{a_4}
4^{a_5}	2^{a_6}

Output of $Q(I^a)$, $Q'(I^a)$ (DEFAULT-ALL propagation):

A	B
$1^{a_1,a_3}$	$2^{a_2,a_6}$
$1^{a_1,a_3}$	3^{a_4}
4^{a_5}	$2^{a_2,a_6}$

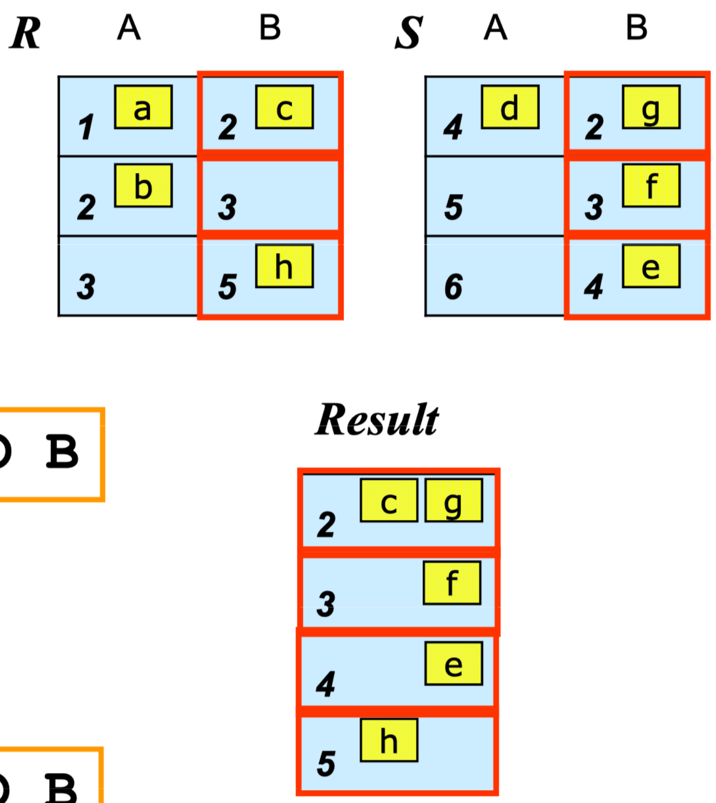
Fig. 1.6 Example showing that where-provenance is sensitive to query rewriting.

If a query Q propagates annotations under the *default-all* propagation scheme in DBNotes, then equivalent formulations of Q are guaranteed to produce identical annotated results. In the default-all scheme, annotations are propagated based on where data is copied from according to *all* equivalent queries of Q. Hence, this propagation scheme can be perceived as a "better" method for propagating annotations for Q. The

Default-all / Where provenance / Query rewriting

The DEFAULT Scheme

Propagate annotations according to **where data is copied from**




```
SELECT DISTINCT B
FROM R r
PROPAGATE DEFAULT r.B TO B

UNION

SELECT DISTINCT B
FROM S s
PROPAGATE DEFAULT s.B TO B
```

8

Natural semantics for tracing the provenance of data

Annotation Propagation under the DEFAULT Scheme

R

A	B
1	2 a

S

B	C
2 b	3

Q_1:

```
SELECT DISTINCT r.A, r.B, s.C
FROM R r, S s
WHERE r.B =a s.B
PROPAGATE DEFAULT
```

Output of Q_1

1	2 a	3

versus

equivalent queries, but different annotated output

Q_2:

```
SELECT DISTINCT *
FROM R NATURAL JOIN S
PROPAGATE DEFAULT
```

Output of Q_2

1	2 a b	3

9

The DEFAULT-ALL scheme

- Propagate annotations according to where data is copied from according to **all** equivalent formulations of the given query
- **User Query *Q*:**

```
SELECT DISTINCT r.A, s.B, s.C
FROM R r, S s
WHERE r.B = s.B
PROPAGATE DEFAULT-ALL
```

(*) the ***SQL*** query corresponding to ***Q***

- Compute the results of Q on a database D – **idea**:
  - $E(Q)$ denotes the set of all queries that are equivalent to Q (more precisely, *(*)*).
  - Execute each query in $E(Q)$ on the database D under the DEFAULT scheme, then combine the results under $\cup_a$.

10

Computing the results of a DEFAULT-ALL query

- **Question:**

  Given a *pSQL* query Q with **DEFAULT-ALL** propagation scheme and a database D, can we compute the result of $Q(D)$?

- **Problem:**

  There are infinitely many queries in $E(Q)$. It is therefore impossible to execute every query in $E(Q)$ in order to obtain the result of $Q(D)$.

- **Solution:** Compute a finite basis of $E(Q)$ first.

Source: Laura Chiticariu. "Systems for tracing the provenance of data". Talk at University of Washington, 2008.
Wolfgang Gatterbauer. Principles of scalable data management: https://northeastern-datalab.github.io/cs7240/

Version June 20, 2011

Default-all is dangerous!

Wolfgang Gatterbauer
Alexandra Meliou
Dan Suciu

3rd USENIX Workshop on the Theory and Praxis of Provenance (Tapp'11)

Database group
University of Washington

http://db.cs.washington.edu/causality/

Source: Gatterbauer, Meliou, Suciu. "Default-al is dangerous". Tapp 2011. https://arxiv.org/pdf/1105.4395

Overview Provenance Definitions

	Why? [illegible]	**Where?** [illegible]
Naive	Witness	"SQL interpretation"
Provenance definition	Why-provenance = witness basis (α_w) Buneman et al. [ICDT'01]	Where-provenance = propagation (α_p) Buneman et al. [PODS'02]
QRI definition (Query-Rewrite-Insensitive)	Minimal witness basis ($\alpha_w{}^m$) Buneman et al. [ICDT'01]	Default-all propagation ($\alpha_p{}^d$) Bhagwat et al. [VLDB'04]

We do not discuss here whether QRI is desirable (see also Glavic, Miller [Tapp'11]*), but merely point out that, if aiming for QRI, care has to be taken about the ramifications of the proposed semantics.*

Has problems if one interprets annotations on attribute values

Minimal propagation ($\alpha_p{}^m$)
Proposed in this paper!

Independent work presented at this WS

Overview Provenance Definitions

	Why?	**Where?**
Naive	Witness	"SQL interpretation"
Provenance definition	Why-provenance = witness basis (α_w) Buneman et al. [ICDT'01]	Where-provenance = propagation (α_p) Buneman et al. [PODS'02]
	($\alpha_w{}^m$) DT'01]	Default-all propagation ($\alpha_p{}^d$) Bhagwat et al. [VLDB'04]

Glavic, Miller [Tapp'11]

Semantics		Sound	Complete	Responsible	Insensitive (set)	Insensitive (bag)	Stable
Why	Wit	-	X	-	X	X	X
	Why	-	X	-	-	X	X
	IWhy	-	X	X	X	X	X
Where	Where	-	-	-	-	?	X
	IWhere	-	-	-	X	X	-
How		-	X	-	-	X	X
Lineage-based	Lineage	X	X	-	-	-	X
	PI-CS	X	X	-	-	-	X
	C-CS	X	-	-	-	-	X
Causality		-	X	X	X	X	X

Has problems if one interprets annotations on attribute values

Minimal propagation ($\alpha_p{}^m$)
Proposed in this paper!

Note that Minimal propagation is "stable", in contrast to Default-all

Source: Gatterbauer, Meliou, Suciu. "Default-al is dangerous". Tapp 2011. https://arxiv.org/pdf/1105.4395

Example 1: Query-Rewrite-Insensitivity (QRI)

Why

Why-provenance = witness basis (α_w)

Minimal witness basis (α_w^m)

Lineage (α_l)

Input
R

	A	B
t_1	1	2
t_2	1	3
t_3	2	2

Query 1
$Q_1(x,y)$:-$R(x,y)$

A	B	
1	2	$\{\{t_1\}\}$
1	3	$\{\{t_2\}\}$
2	2	$\{\{t_3\}\}$

Query 2 ≡ Query 1
$Q_2(x,y)$:-$R(x,y),R(\_,y)$

A	B	Why-provenance (α_w)	Minimal witness basis (α_w^m)	Lineage (α_l)
1	2	$\{\{t_1\},\{t_1,t_3\}\}$	$\{\{t_1\}\}$	$\{t_1,t_3\}$
1	3	$\{\{t_2\}\}$	$\{\{t_2\}\}$	$\{t_2\}$
2	2	$\{\{t_3\},\{t_1,t_3\}\}$	$\{\{t_3\}\}$	$\{t_1,t_3\}$

Where

Where-provenance = propagation (α_p)

Minimal propagation (α_p^m)

Default-all propagation (α_p^d)

Input
R^a

A	B
1^a	2^b
1^c	3^d
2^e	2^f

Query 1
$Q_1(x,y)$:-$R^a(x,y)$

A	B
1^a	2^b
1^c	3^d
2^e	2^f

Query 2 ≡ Query 1
$Q_2(x,y)$:-$R^a(x,y),R^a(\_,y)$

Where-provenance = propagation (α_p):

A	B
1^a	$2^{b,f}$
1^c	3^d
2^e	$2^{b,f}$

Default-all propagation (α_p^d):

A	B
$1^{a,c}$	$2^{b,f}$
$1^{a,c}$	3^d
2^e	$2^{b,f}$

Minimal propagation (α_p^m):

A	B
1^a	2^b
1^c	3^d
2^e	2^f

Example adapted from Cheney, Chiticariu, Tan. Provenance in databases: why, how, and where. Foundations and trends in databases 2009. https://dl.acm.org/doi/abs/10.1561/1900000006

Source: Gatterbauer, Meliou, Suciu. "Default-al is dangerous". Tapp 2011. https://arxiv.org/pdf/1105.4395

Real example: Why Default-all is dangerous

Hanako queries a community DB for contents of LF-milk*:

Community Database
R^a

Food	*Content*
LF Milk	Cesium-137^{b}
LF Milk	Calcium^{d}
SC Water	Cesium-137^{f}

b *Bob, March 18, 2011*
Don't drink, lots of Cesium!

f *Fuyumi, March 19, 2011*
No Cesium, save to drink!

Hanako's query
$Q(y)$:-R^a('LF Milk',y)

Content
Cesium-137$^{???}$
Calcium^{d}

Default-all propagation makes her drink the milk:

Default-all propagation (α_p^d)

Content
Cesium-137^{bf}
Calcium^{d}

b *Bob, March 18, 2011*
Don't drink, lots of Cesium!

f *Fuyumi, March 19, 2011*
No Cesium, save to drink!

"semantically irrelevant information": annotations leak over from SC Water tuple to LF Milk

Minimal propagation (α_p^m)

Content
Cesium-137^{b}
Calcium^{d}

b *Bob, March 18, 2011*
Don't drink, lots of Cesium!

"all relevant and only relevant"

* Note the one-to-one correspondence of this example with example 1 from previous page

Source: Gatterbauer, Meliou, Suciu. "Default-al is dangerous". Tapp 2011. https://arxiv.org/pdf/1105.4395

Definition Minimal propagation ($\alpha_p{}^m$)

$$\alpha_p^m(t,A,Q) := \bigcup_{\substack{t' \in \Cup \alpha_w^m(t,Q) \\ A' \in \text{attributes of } t' \text{ propagating to cell}(t,A)}} \alpha_p(t',A')$$

$\Cup$ transforms 'sets of sets' into 'sets', hence something like QRI lineage

Intuition:
Return the intersection between:
- query-specific where-provenanc (α_p)
- and QRI minimal witness basis ($\alpha_w{}^m$)

"all relevant ... and only relevant"

Example 1

Where provenance (α_p)

Minimal propagation ($\alpha_p{}^m$)

Input

R^a

	A	B
t_1	1^a	2^b
t_2	1^c	3^d
t_3	2^e	2^f

Query 2

$Q_2(x,y)$:-$R^a(x,y),R^a(\_,y)$

A	B		
1^a	$2^{b,f}$	$\{\{t_1\}\}$	$\{t_1\}$
1^c	3^d	$\{\{t_2\}\}$	$\{t_2\}$
2^e	$2^{b,f}$	$\{\{t_3\}\}$	$\{t_3\}$

$\Cup\, \alpha_w{}^m$

Minimal witness basis ($\alpha_w{}^m$)

	A	B
t_4	1^a	2^b
t_5	1^c	3^d
t_6	2^e	2^f

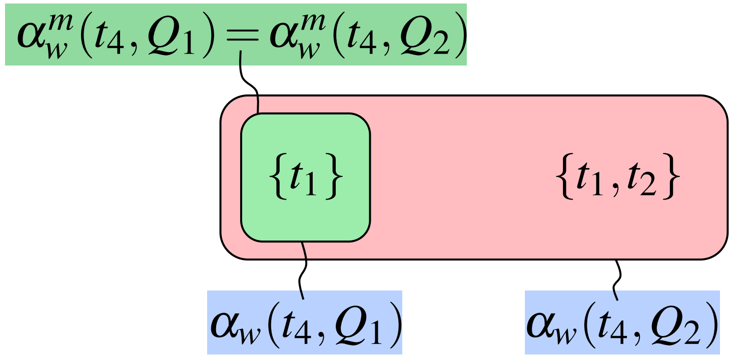
$$\alpha_p^m(t_4,B,Q_2) = \bigcup_{t' \in \{t_1\}, A'} \alpha_p(t',A') = \alpha_p(t_1,B) = \{b\}$$

Example 1: Illustration of "minimal" versus "all"

Why-provenance

Why-provenance (α_w)

Minimal witness basis ($\alpha_w{}^m$)

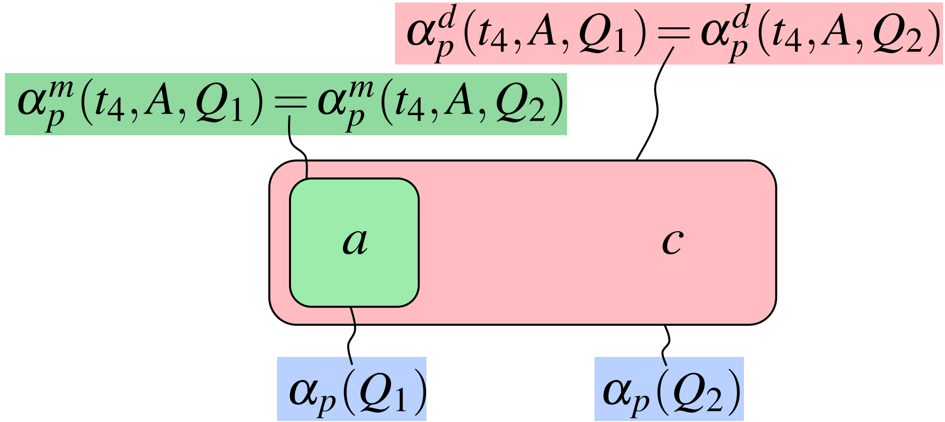


Where-provenance

Where-provenance (α_p)

Default-all propagation ($\alpha_p{}^d$)

Minimal propagation ($\alpha_p{}^m$)



Source: Gatterbauer, Meliou, Suciu. "Default-al is dangerous". Tapp 2011. https://arxiv.org/pdf/1105.4395

Interpretation of Annotations 1: Attribute Value*

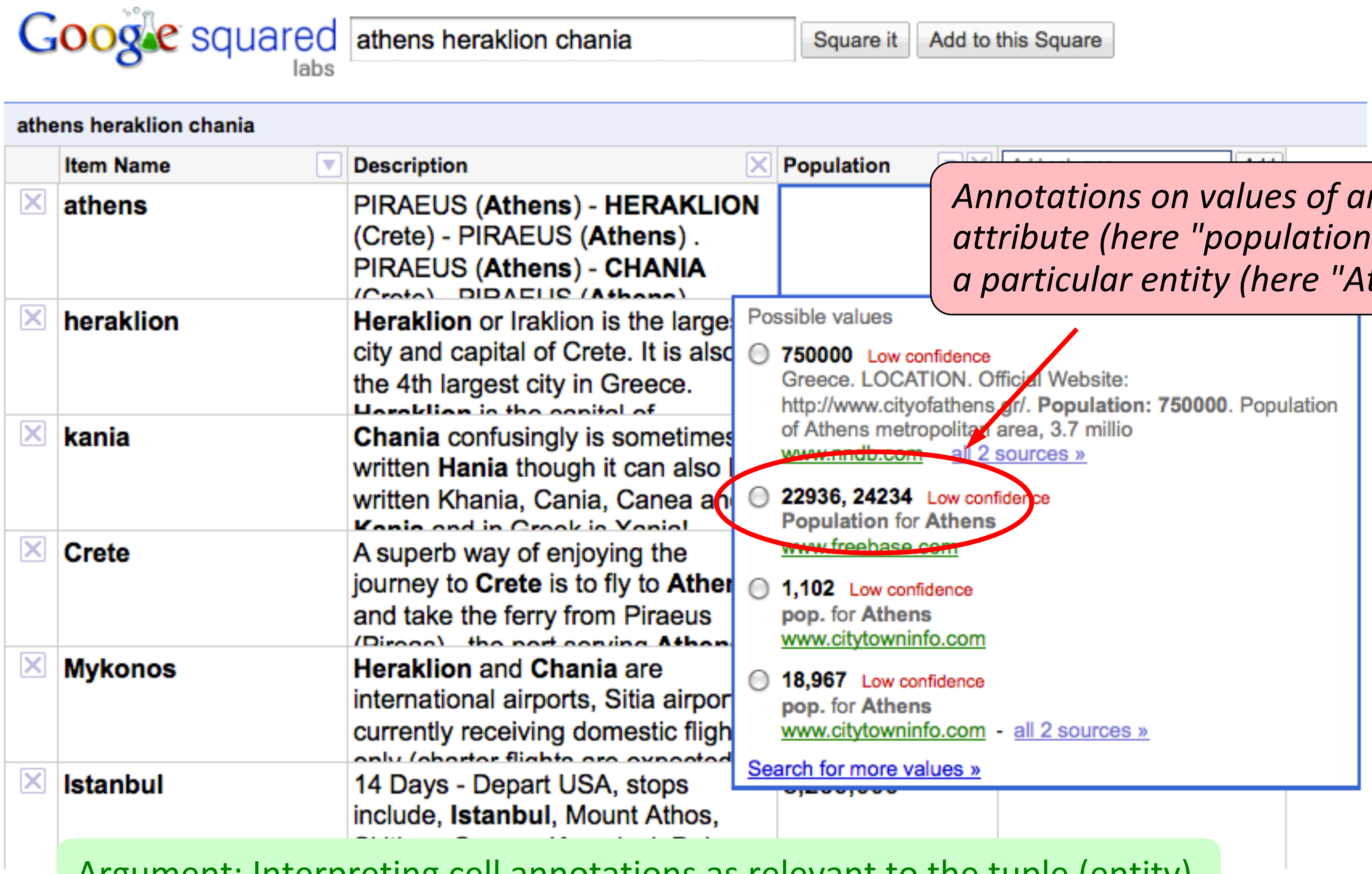
athens heraklion chania | Square it | Add to this Square

athens heraklion chania

	Item Name	Description	Population	Add columns
	athens	PIRAEUS (**Athens**) - **HERAKLION** (Crete) - PIRAEUS (**Athens**) . PIRAEUS (**Athens**) - **CHANIA**	4 possible values	
	heraklion	**Heraklion** or Iraklion is the largest city and capital of Crete. It is also the 4th largest city in Greece.	1 possible value	
	kania	**Chania** confusingly is sometimes written **Hania** though it can also be written Khania, Cania, Canea and	1 possible value	
	Crete	A superb way of enjoying the journey to **Crete** is to fly to **Athens** and take the ferry from Piraeus	623,666	
	Mykonos	**Heraklion** and **Chania** are international airports, Sitia airport is currently receiving domestic flights	9,320	
	Istanbul	14 Days - Depart USA, stops include, **Istanbul**, Mount Athos, Skithos, Samos, Kusadasi, Delos,	8,260,000	

* Interpretation of annotations on entity attribute values favored by us and underlying our model

Interpretation of Annotations 1: Attribute Value*



Annotations on values of an attribute (here "population") for a particular entity (here "Athens")

Argument: Interpreting cell annotations as relevant to the tuple (entity) adds something that is not trivially modeled with normalized tables.

* Interpretation of annotations on entity attribute values favored by us and underlying our model

Source: Gatterbauer, Meliou, Suciu. "Default-al is dangerous". Tapp 2011. https://arxiv.org/pdf/1105.4395

Interpretation of Annotations 2: Domain Value*

Domain value annotations*

Input R^a:

A	*B*
1^a	2^b
1^c	3^d
2^e	2^f

b *Bob, March 18, 2011*
This number is a prime number.

f *Fuyumi, March 19, 2011*
Two is not a prime number because it is even.

Input S^a:

…	*Date*
…	Dec 25
…	…
…	Dec 25

b *This is a holiday.*

f *This is a holiday too !!!*

Argument for default-all: If annotations are on domain values, then retrieving all annotations are relevant.

Alternative representation

Annotation table S^a:

B	*annotation*
2	b: *Bob, March 18, 2011* *This number is a prime number.*
2	f: *Fuyumi, March 19, 2011* *Two is not a prime number because it is even*

Annotation table S^a:

Date	*annotation*
Dec 25	This is a holiday.

Counter-Argument: But then these annotations can be modeled in a separate table as normalized tables.

* Alternative interpretation suggested by Wang-Chiew Tan (example created after conversation at Sigmod 2011)

Source: Gatterbauer, Meliou, Suciu. "Default-al is dangerous". Tapp 2011. https://arxiv.org/pdf/1105.4395

Backup: Detailed Example 2

R^a

	A	B
t_1	1^a	2^b
t_2	1^c	3^d
t_3	2^e	2^f
t_4	2^g	4^h

$Q_5(x,y)$:-$R^a(x,y),R^a(y,\_),R^a(x,\_)$

	A	B	Why-provenance (α_w)	Minimal witness basis ($\alpha_w{}^m$)	$\Cup\alpha_w{}^m$ (~QRI lineage)
t_5	$1^{a,c}$	$2^{b,e,g}$	$\{\{t_1,t_3\},\{t_1,t_2,t_3\},\{t_1,t_4\},\{t_1,t_2,t_4\}\}$	$\{\{t_1,t_3\}, \{t_1,t_4\}\}$	$\{t_1,t_3,t_4\}$
t_6	$2^{e,g}$	$2^{e,f,g}$	$\{\{t_3\},\{t_3,t_4\}\}$	$\{\{t_3\}\}$	$\{t_3\}$

Where-provenance (α_p)

Why-provenance (α_w)

$\Cup\alpha_w{}^m$ (~QRI lineage)

Minimal witness basis ($\alpha_w{}^m$)

Default-all propagation ($\alpha_p{}^d$)

A	B
$1^{a,c}$	$2^{b,e,f,g}$
$2^{e,g}$	$2^{b,e,f}$

$\alpha_p{}^d(t_4,B,Q_5) = \alpha_p(t_4,B,Q_6)$ with

$Q_6(x,y)$:-$R^a(x,y),R^a(y,\_),R^a(x,\_) ,S^a(\_,y)$

Minimal propagation ($\alpha_p{}^m$)

	A	B
t_4	1^a	$2^{b,e,g}$
t_5	2^e	$2^{e,f}$

$$\alpha_p^m(t_4,A,Q_5) = \bigcup_{t'\in\{t_1,t_3,t_4\},A'} \alpha_p(t',A') = \alpha_p(t_1,A) = \{a\}$$

$$\alpha_p^m(t_5,B,Q_5) = \bigcup_{t'\in\{t_3\},A'} \alpha_p(t',A') = \alpha_p(t_3,B)\cup\alpha_p(t_3,A) = \{e,f\}$$

Note minimal propagation is not equivalent to just evaluating the where-provenance for the query: $Q_7(x,y)$:-$R^a(x,y),R^a(y,\_)$. E.g. $\alpha_p(t_5,B,Q_7) = \{e,f,g\}$