Updated 3/17/2022

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

Wolfgang Gatterbauer

CS7240 Principles of scalable data management (sp22)

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

3/11/2022

Pre-class conversations

- Recapitulation of provenance semirings, including new exercise
- Projects & scribes: we are past halftime of the class
- Possible exercise: Provenance for relational division

- Today:
  - The algebra of provenance
  - a quick glimpse at reverse data management

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 ☹
  - $(\mathbb{R}^{\infty},\min,\infty)$
    - identity element ∞: $\min[x,\infty] = x$
    - What is the inverse 3^{-1} s.t. $\min[3, 3^{-1}] = \infty$?

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

$-\infty$

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)
  - $(\mathbb{R}\setminus\{0\}, \cdot, 1)$ 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 ☹

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

    What is x? ?

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

- Assume(x,y,z) s.t. x⨁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},∧,1) and (x,y,z)=(1,0,0)
  - x∧0=0

    What is x? x could be 0 or 1

    $x \wedge 0 = 1$
- ($\mathbb{R}^{\infty}$,min,∞) 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

- 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

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

(Commutative) Semirings

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

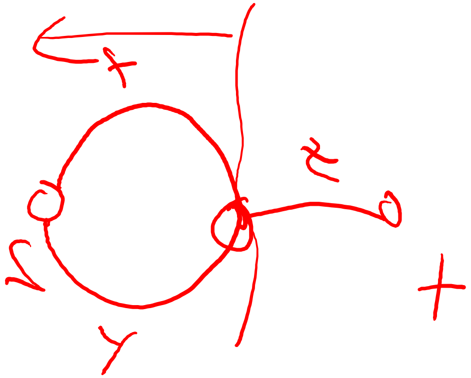
- Examples

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

     not the other way: $\min[x+y,z] \neq \min[x,z] + \min[y,z]$; e.g. $\min[3+4,5] = 5 \neq 7 = \min[3,5] + \min[4,5]$



  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)

Ring-like structures

Field	
Closure	Abelian Group (+)
Associativity	
Identity Element	
Inverse Element	
Commutativity	
Closure	non-zero elements form an Abelian Group (*)
Associativity	
Identity Element	
Inverse Element	except 0
Commutativity	
Left Distributivity	Distributivity
Right Distributivity	

Ring	
Closure	Abelian Group (+)
Associativity	
Identity Element	
Inverse Element	
Commutativity	
Closure	Monoid (*) sometimes called ring w identity
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 (rng)	
Closure	Abelian Group (+)
Associativity	
Identity Element	
Inverse Element	
Commutativity	
Closure	Semi-Group (*) sometimes called ring w/o identity
Associativity	
Identity Element	
Inverse Element	
Commutativity	
Left Distributivity	Distributivity
Right Distributivity	

Semi-Ring (rig)	
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	

$\mathbb{Q}$ (rational numbers)
$\mathbb{Z}/5\mathbb{Z}$ (integers mod 5)
$\frac{f(x)}{g(x)}$ 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

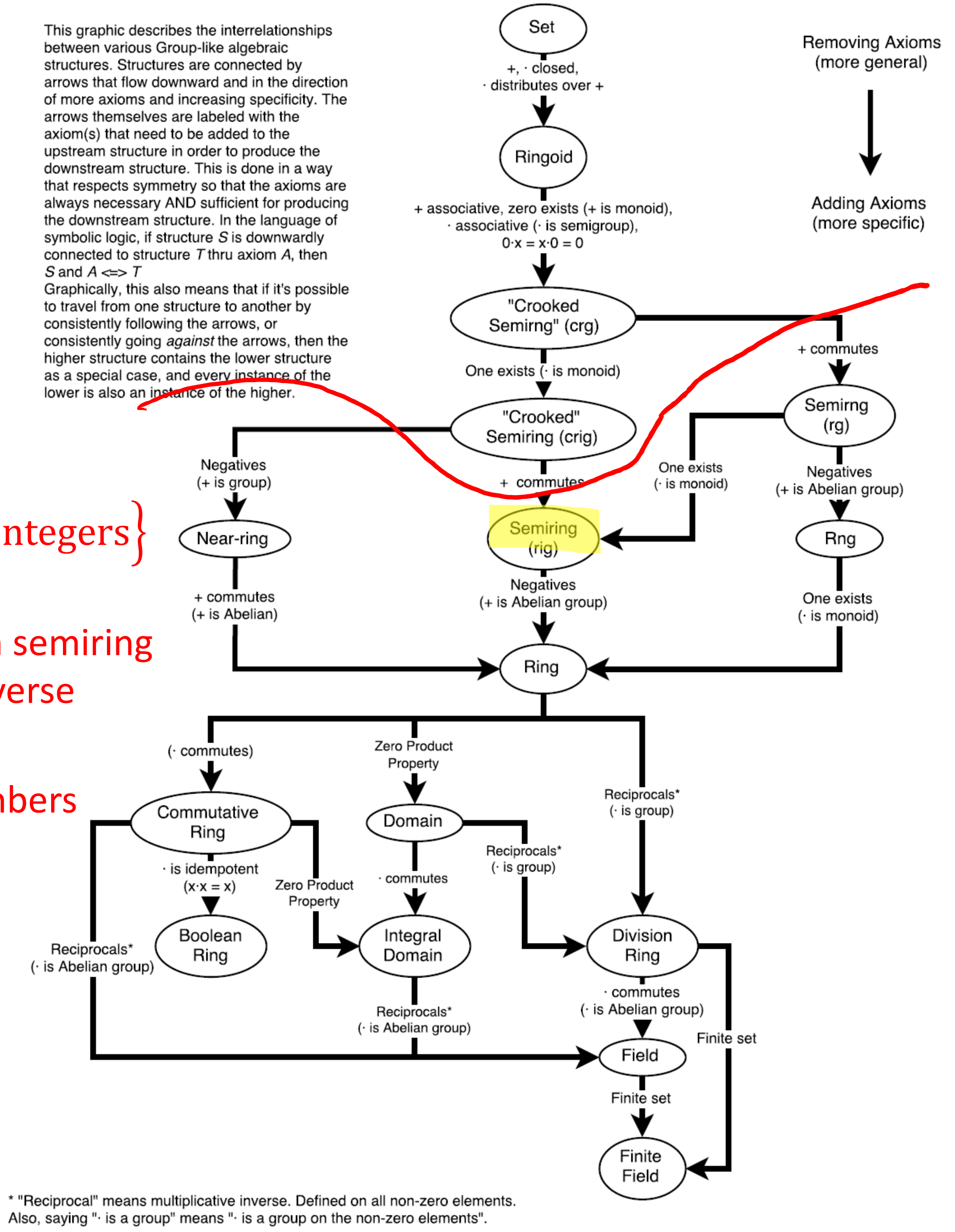


Figure credits: https://kevinbinz.com/2014/11/16/goodman-semiring-parsing/, https://math.stackexchange.com/questions/2361889/graphically-organizing-the-interrelationships-of-basic-algebraic-structures

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:

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

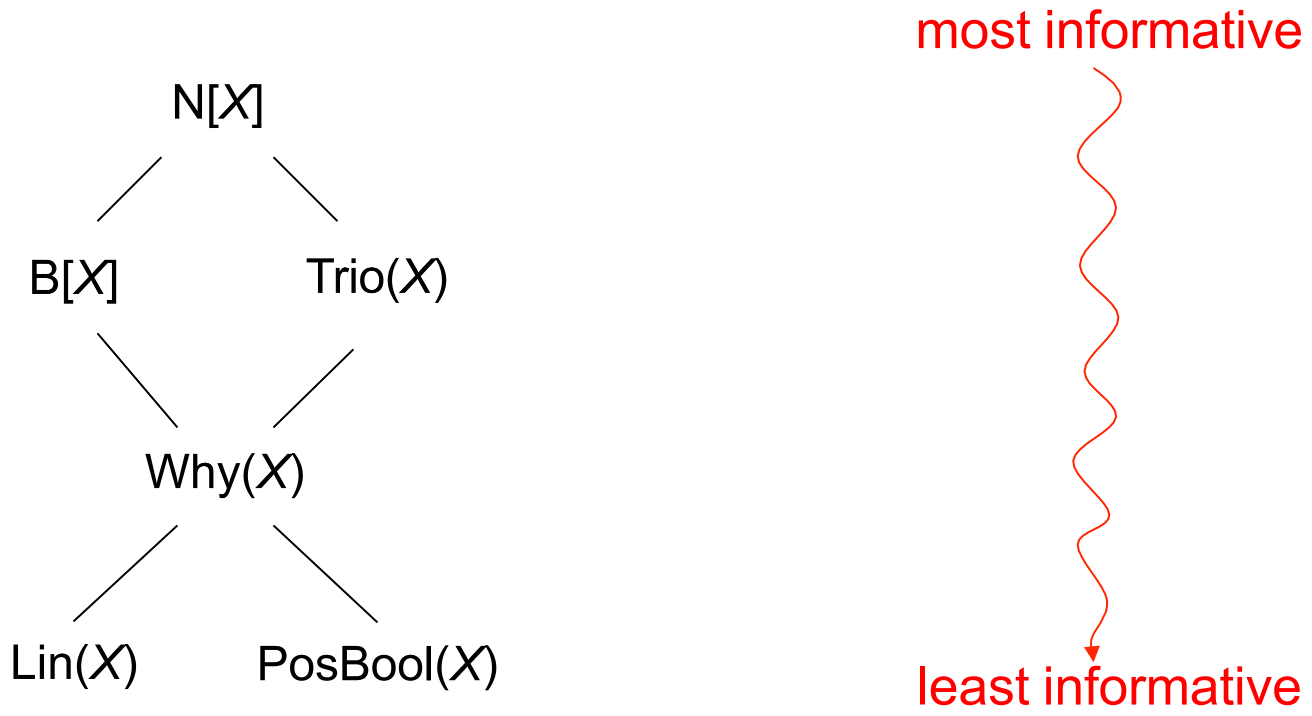
- We can denote it by $h: (S,\star) \longrightarrow (S',\circ)$.

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.

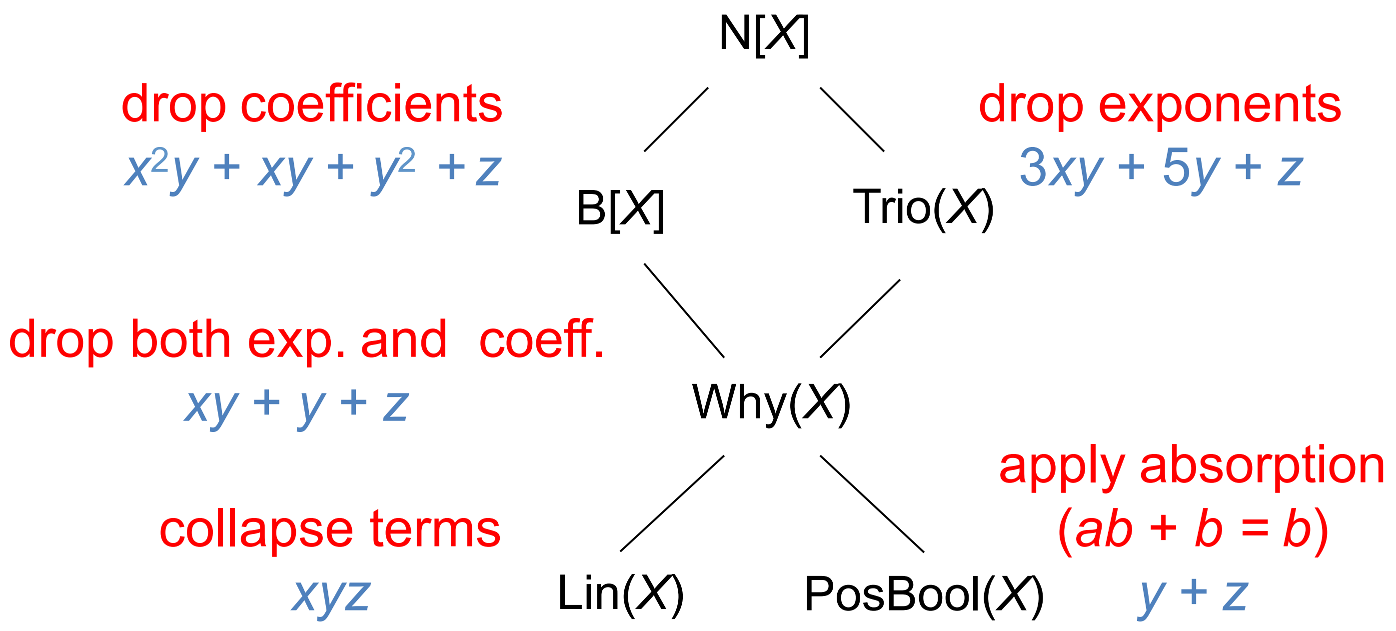
- **Definition (homomorphism between semirings):** Let $(R,+,\bullet)$ and $(S,\star,\circ)$ be semirings. A homomorphism from $(R,+,\bullet)$ to $(S,\star,\circ)$ is a map h: $S \longrightarrow S'$ that satisfies, for all x, y in S:
  - $h(x + y) = h(x) \star h(y)$ addition preserving
  - $h(x \bullet y) = h(x) \circ h(y)$ 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..."

Paper: Bistarelli, Montanari, Rossi. Semiring-Based Constraint Satisfaction and Optimization. JACM 1997. https://doi.org/10.1145/256303.256306

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.	$\{0,1\}$	$(\texttt{OR},0)$	$(\texttt{AND},1)$	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 <u>Functional Aggregate Query (FAQ)</u> 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

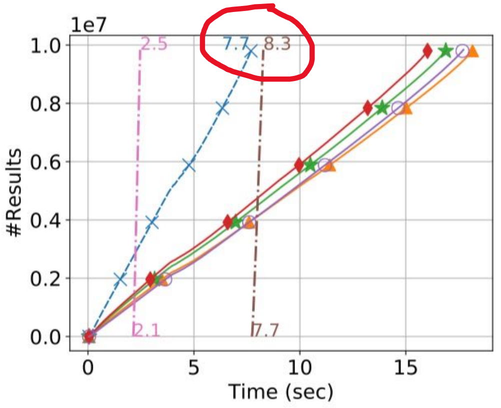
6. Tziavelis+. Optimal Algorithms for Ranked Enumeration of Answers to Full Conjunctive Queries. PVLDB 2020

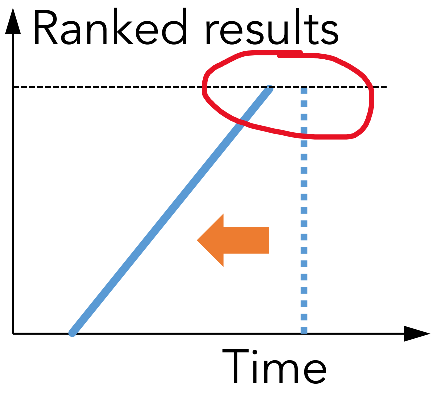
ABSTRACT

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-

$\bowtie$

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.

Multiplying 2×2 matrices

+ ×

MIN +

n

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

n

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

8 multiplications
4 additions

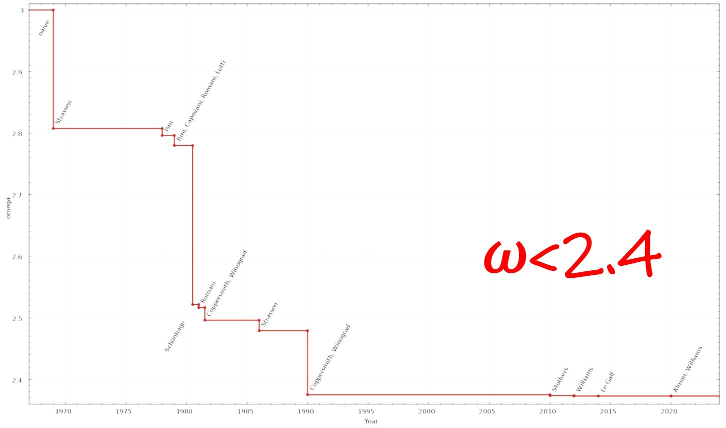
2^3 $O(n^3)$

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

Strassen's 2×2 algorithm

Matrix multiplication exponent ω



$\omega < 2.4$

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

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

Subtraction!

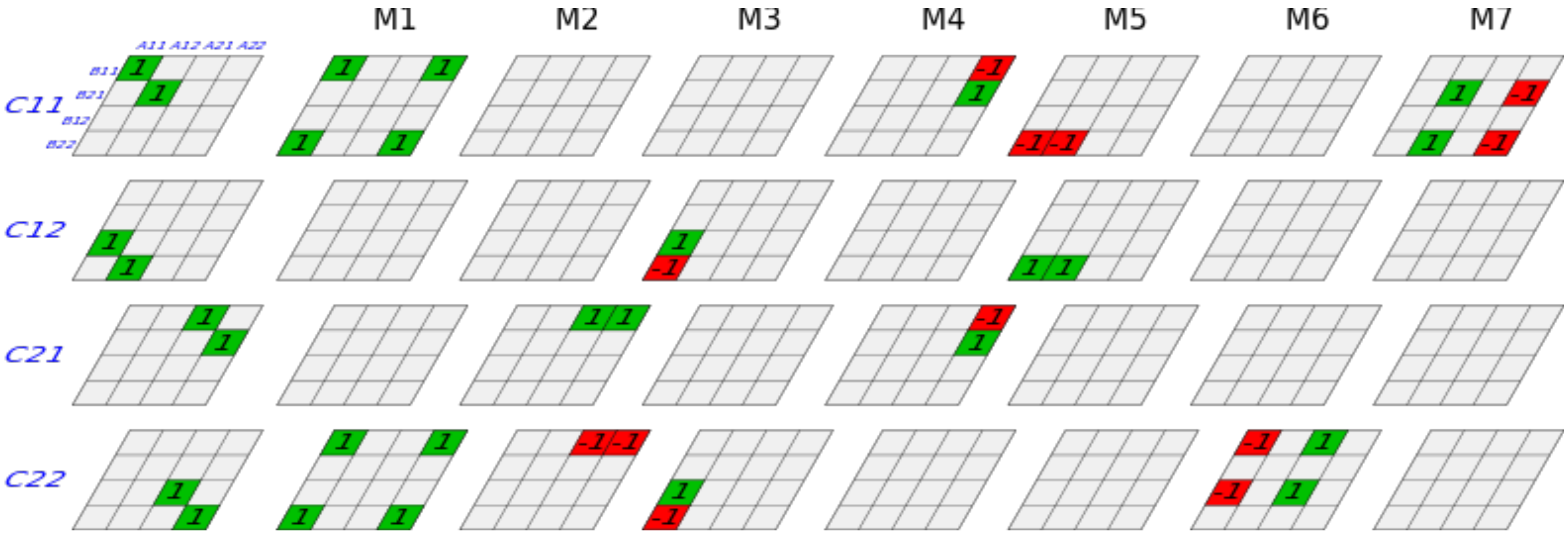
$$C_{11} = M_1 + M_4 - M_5 + M_7$$
$$C_{12} = M_3 + M_5$$
$$C_{21} = M_2 + M_4$$
$$C_{22} = M_1 - M_2 + M_3 + M_6$$

$O(n^\omega)$

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	$C_{11} = T_1 - T_2$	$C_{12} = Q_3 + Q_5$
	$C_{21} = Q_2 + Q_4$	$C_{22} = T_3 - T_4$

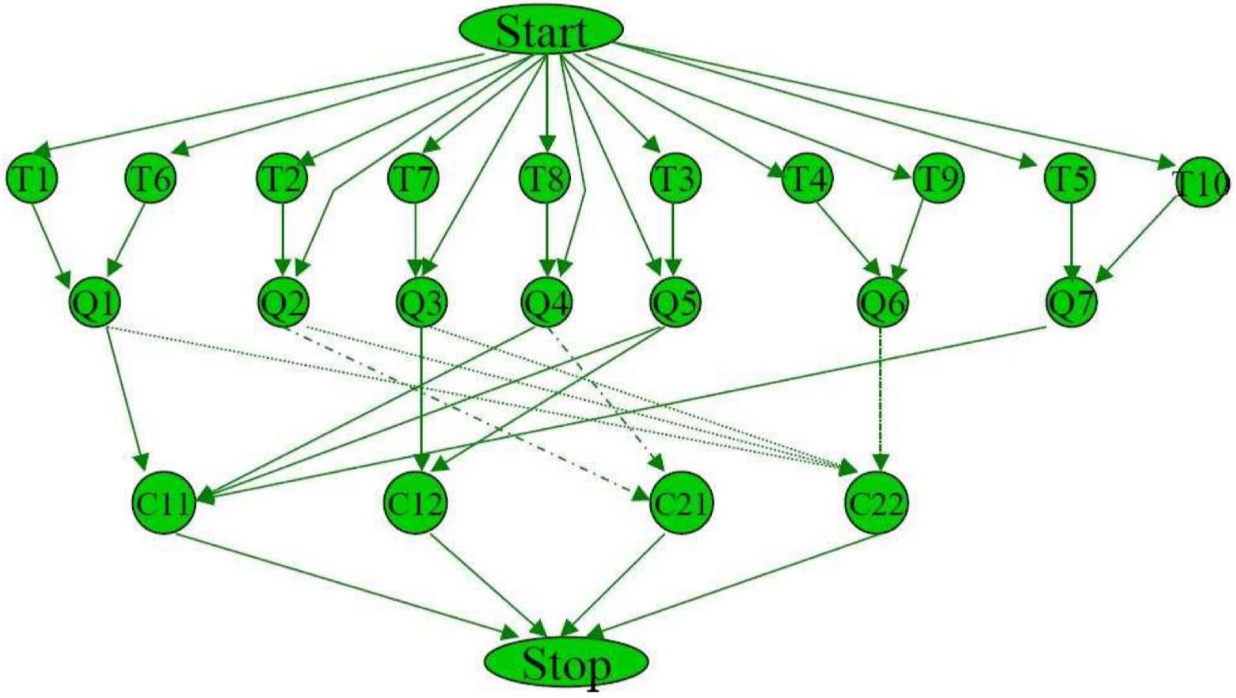
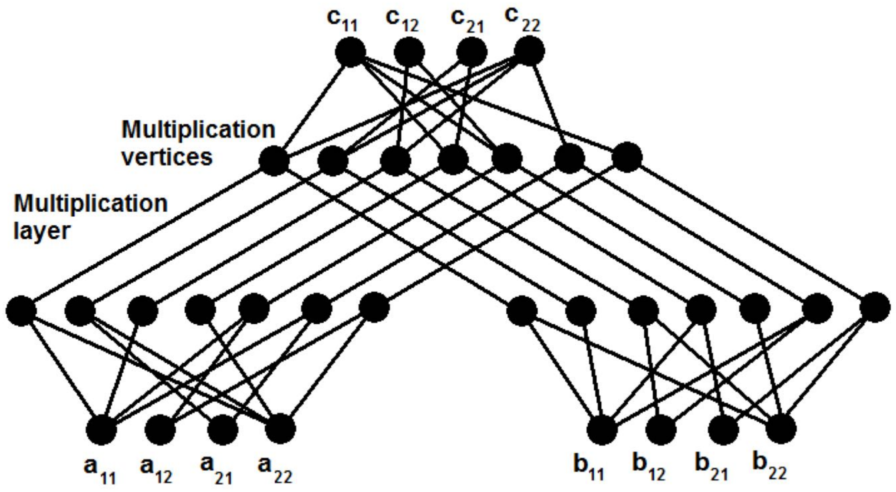


Figure 4. Task graph of Strassen's Algorithm.

Song, Dongarra, Moore. Experiments with Strassens' Algorithm: from sequential to parallel. PDCS 2006. https://scholar.google.com/scholar?cluster=11243079065050760755

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



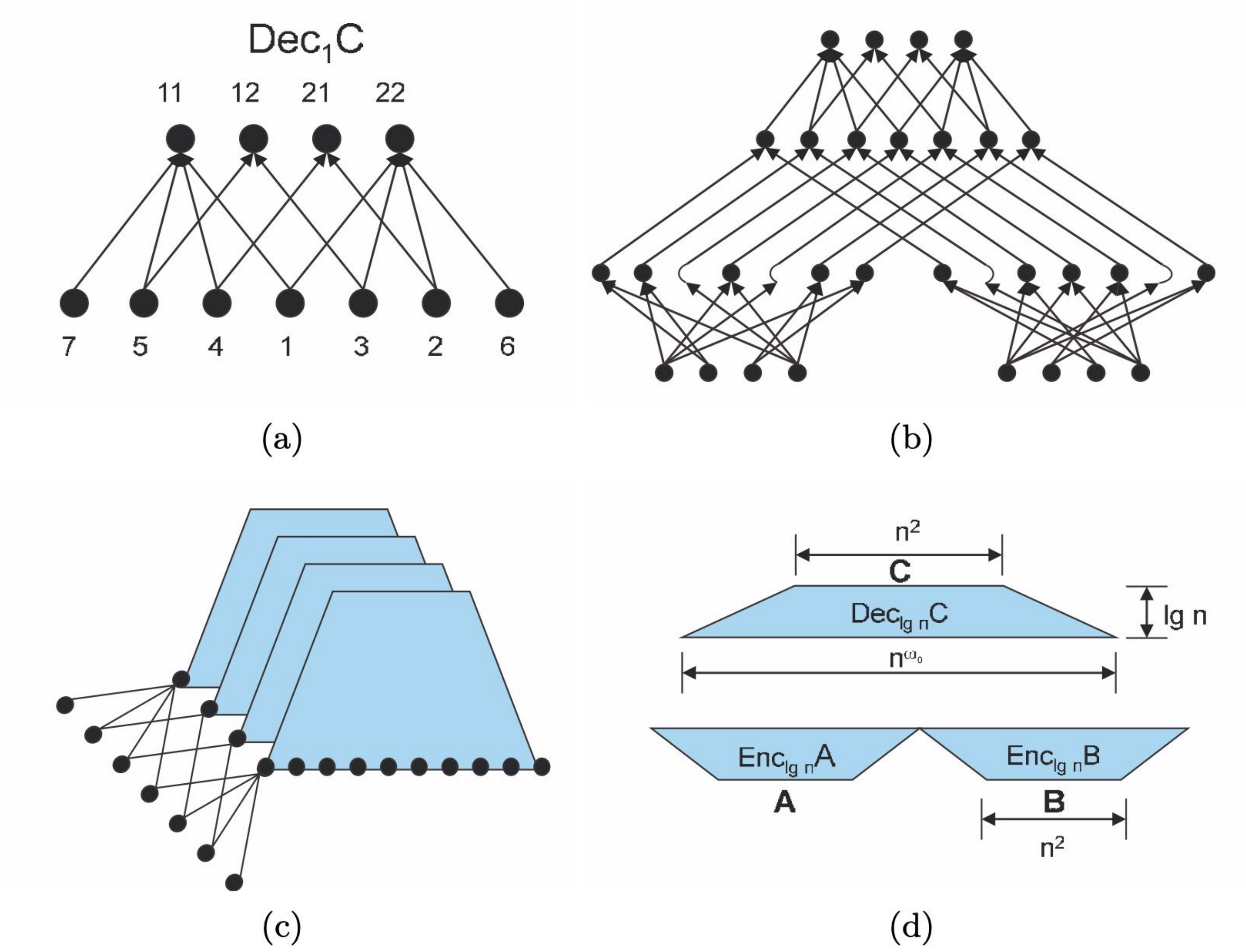


Figure 4.1. The computation graph of Strassen's algorithm (see Algorithm 4.1): (a) $\mathrm{Dec}_1\, C$, (b) H_1, (c) $\mathrm{Dec}_{\lg n}\, C$, (d) $H_{\lg n}$.

Ballard, Carson, Demmel, Hoemmen, Knight, Schwartz. "Communication lower bounds and optimal algorithms for numerical linear algebra." Acta numerica 2014. https://doi.org/10.1017/S0962492914000038
Ballard, Demmel, Holtz, Schwartz. "Graph Expansion and Communication Costs of Fast Matrix Multiplication." ACM 2012. https://doi.org/10.1145/2395116.2395121

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

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

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

DISTINCT

Q_1:
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
BayTours	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.

Example taken from "Provenance in databases: why, how, and where" by Cheney, Chiticariu, Tan, 2009. https://doi.org/10.1561/1900000006

"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

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.

Fixing query-rewrite sensitivity for where provenance

data value
instead of
tuple

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.

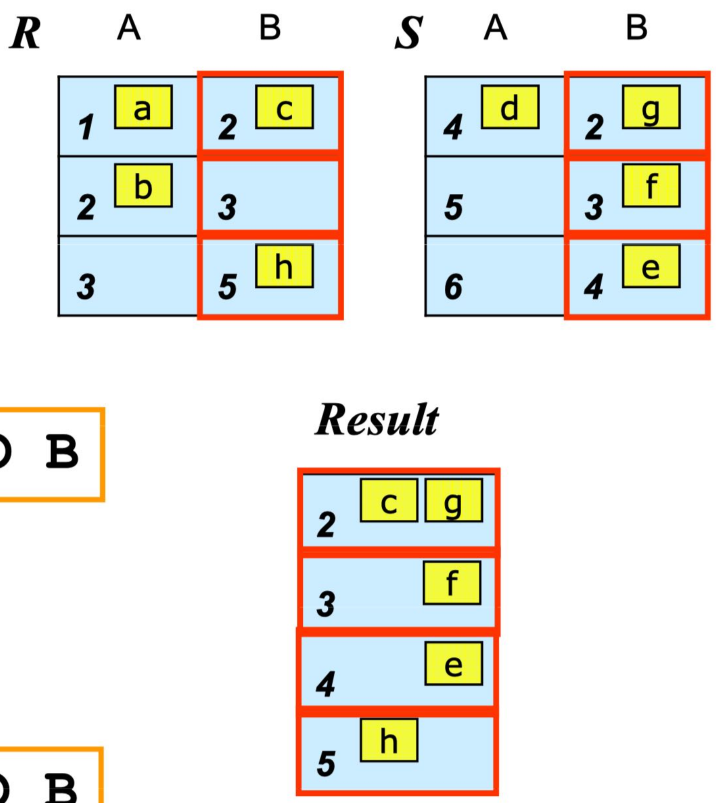
If a query Q propagates annotations under the *default-all* propagation scheme in DBNotes, then equivalent formulations of Q are guaranteed to produce identical annotated results. In the default-all scheme, annotations are propagated based on where data is copied from according to *all* equivalent queries 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**

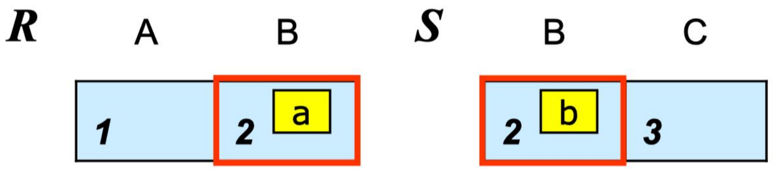
```
SELECT DISTINCT B
FROM R r
PROPAGATE DEFAULT r.B TO B

UNION

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



8

Natural semantics for tracing the provenance of data

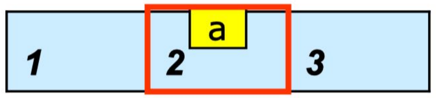
Annotation Propagation under the DEFAULT Scheme



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



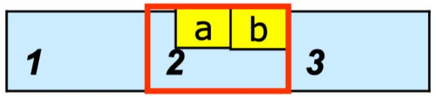
versus

equivalent queries, but different annotated output

Q_2:

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

Output of Q_2



Source: Laura Chiticariu. "Systems for tracing the provenance of data". Talk at University of Washington, 2008.

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 $\cup_a$.

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? deps	**Where?** values
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]

Has problems if one interprets annotations on attribute values

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

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.*

Independent work presented at this WS

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

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]
	(α_w^m) DT'01]	Default-all propagation (α_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 (α_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 ($\alpha_p{}^d$)

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 ($\alpha_p{}^m$)

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

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

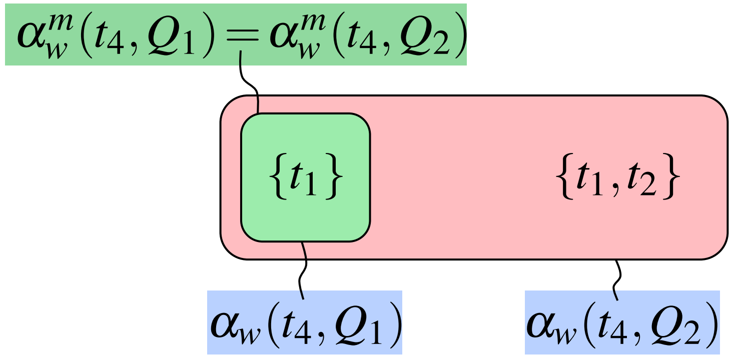
Source: Gatterbauer, Meliou, Suciu. "Default-al is dangerous". Tapp 2011. https://arxiv.org/pdf/1105.4395

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

Why-provenance

Why-provenance (α_w)

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

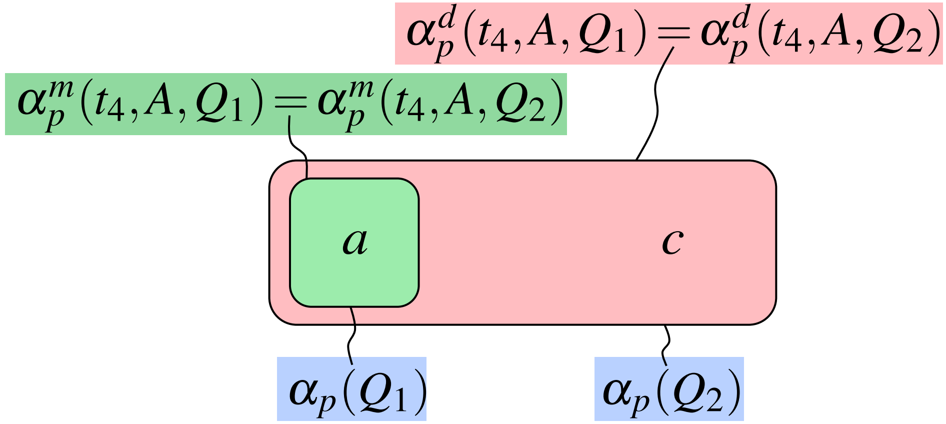


Where-provenance

Where-provenance (α_p)

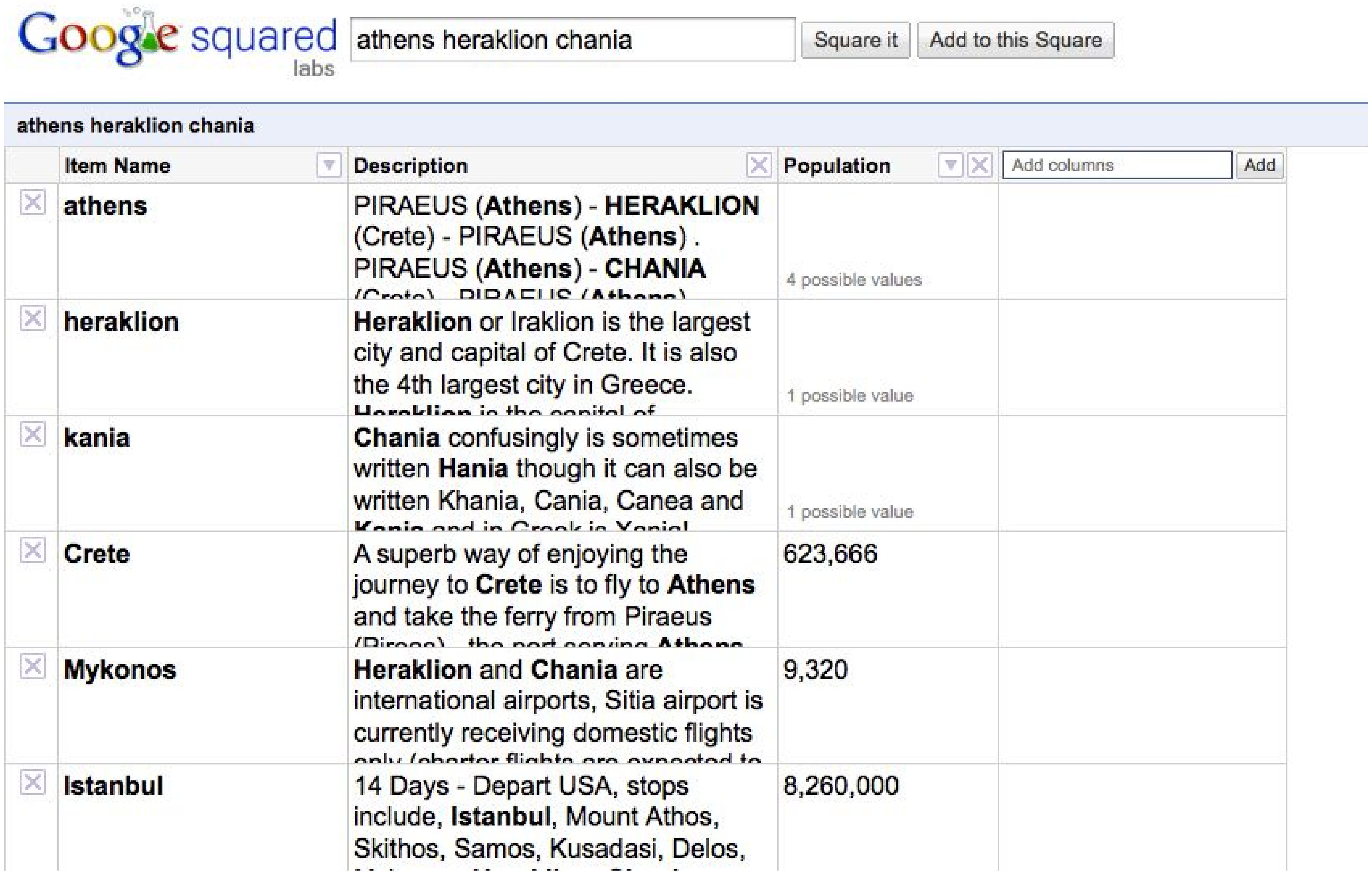
Default-all propagation ($\alpha_p{}^d$)

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*

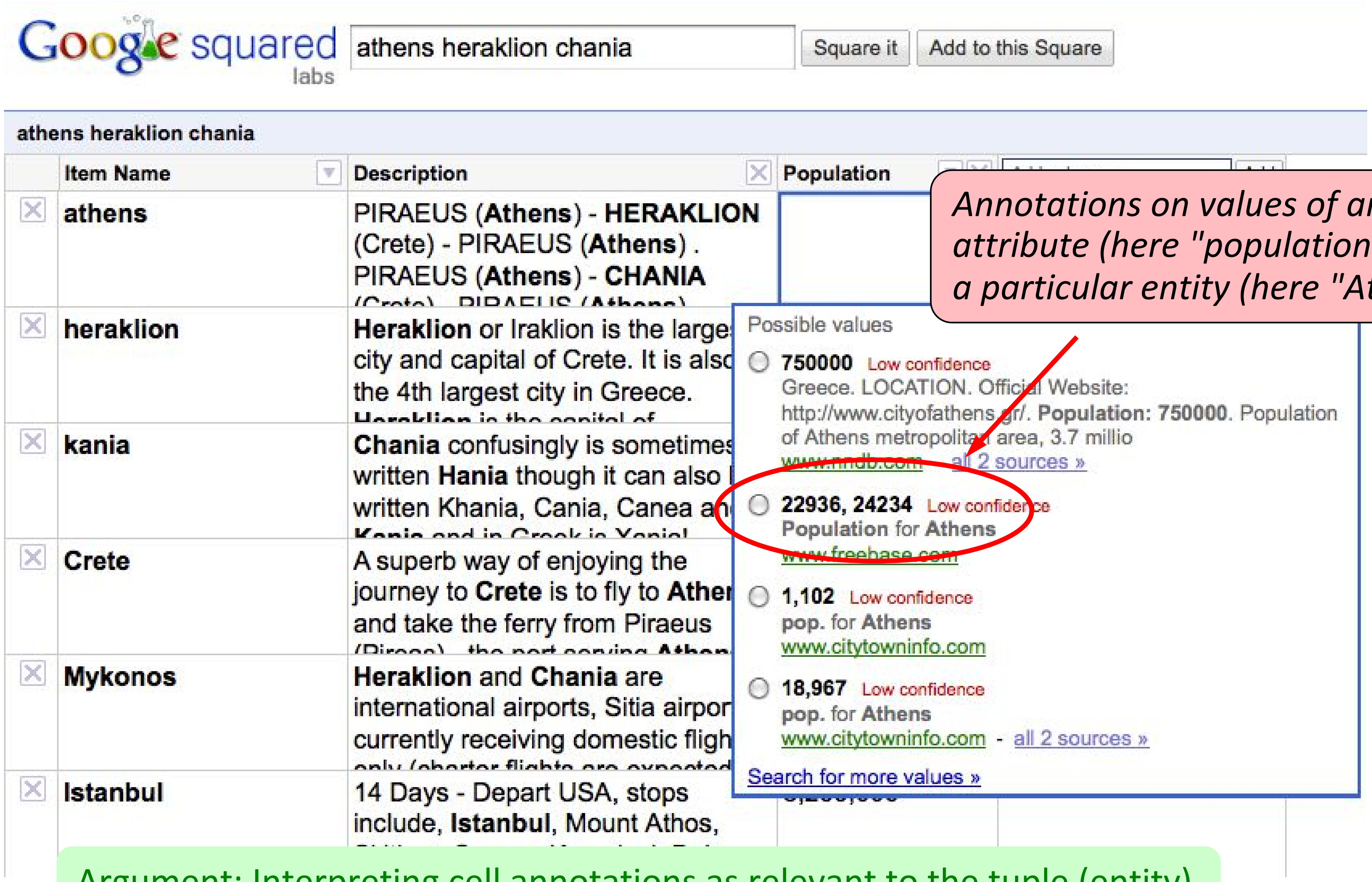


	Item Name	Description	Population	Add columns
	athens	PIRAEUS (**Athens**) - **HERAKLION** (Crete) - PIRAEUS (**Athens**) . PIRAEUS (**Athens**) - **CHANIA**	4 possible values	
	heraklion	**Heraklion** or Iraklion is the largest city and capital of Crete. It is also the 4th largest city in Greece.	1 possible 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

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

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

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

	A	B	Why-provenance (α_w)	Minimal witness basis (α_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)

Minimal witness basis (α_w^m)

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

Default-all propagation (α_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

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

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

Minimal propagation (α_p^m)

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

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

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

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

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

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

The view deletion problem

- *D* a database instance and *V*=*Q*(*D*) a view defined over *D*.
  - *Find a set of tuples ΔD to remove from D so that a specific tuple t is removed from the view*
  - *Minimize the number of side-effects in the view*
    - *View side-effect problem*
      - *Hard:queries with joins and projection or union*
      - *PTIME:the rest*
  - *Minimize the number of tuples deleted from D*
    - *Source side-effect problem*
      - *Same dichotomy*

VIEW

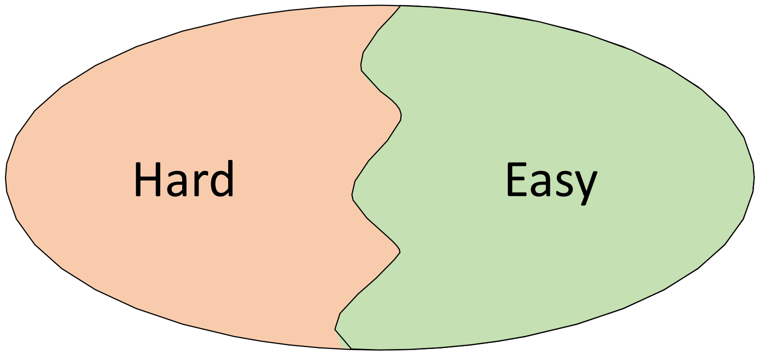
SOURCE

Source: Buneman, Khanna, Tan. On Propagation of Deletions and Annotations Through Views. PODS 2002. https://doi.org/10.1145/543613.543633

Dichotomy theorems

Dichotomy theorem

classifying every member of a family of problems as easy or hard.



In database context

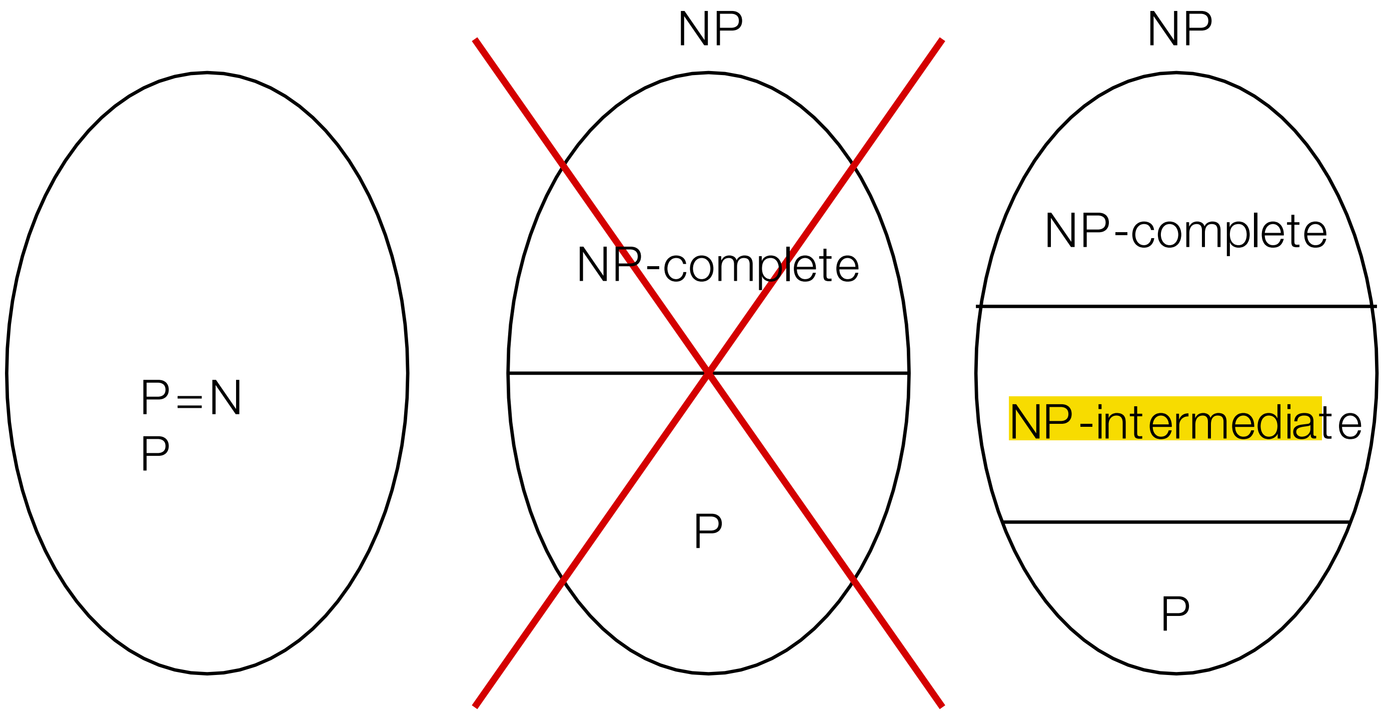
Given a certain problem and a query. Solving this problem for a query is either easy or hard.

Dichotomy theorems

Every problem is either in P or NP-complete.

Why are such theorems surprising?

Theorem [Ladner, 1973]
If $P \neq NP$, then there is a language $L \in NP \setminus P$ that is not NP-complete.



Source: Daniel Marx. Every graph is easy or hard: dichotomy theorems for graph problems, 2015.

Dichotomy theorems

- Dichotomy theorems give goods research programs: easy to formulate, but can be hard to complete.
- The search for dichotomy theorems may uncover algorithmic results that no one has thought of.
- Proving dichotomy theorems requires attacking the problem both from the algorithmic and the complexity side. Requires good command of both algorithmic and hardness proof techniques.
- Possible outcomes:
  - Everything is hard, except some trivial cases.
  - Everything is hard, except the famous known nontrivial positive cases.
  - Some unexpected easy cases are found.

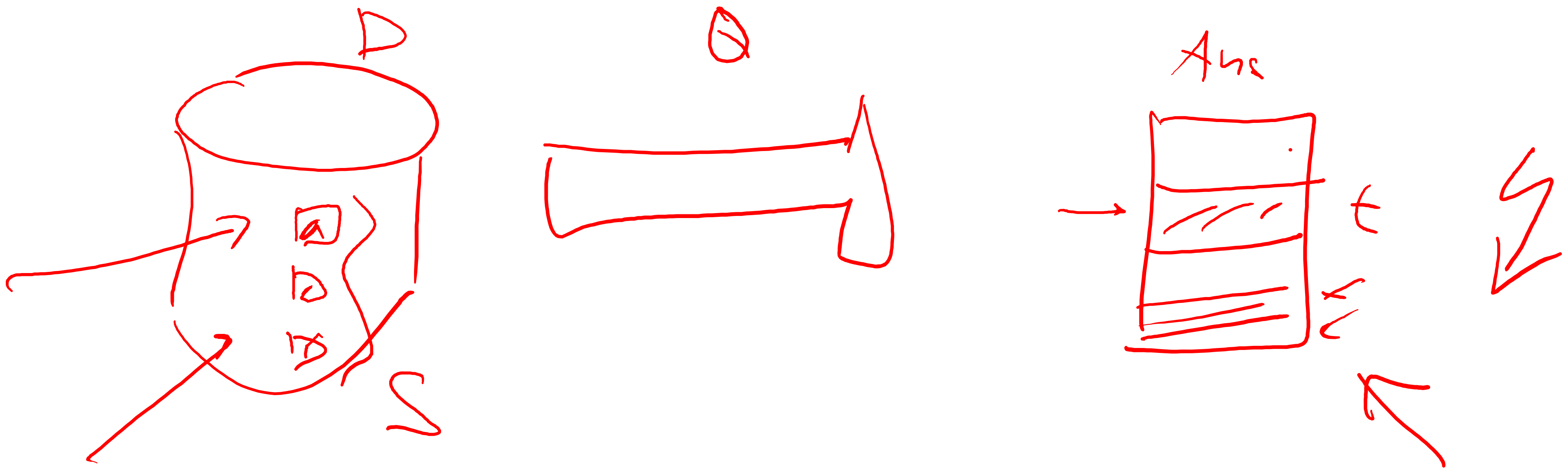
Source: Daniel Marx. Every graph is easy or hard: dichotomy theorems for graph problems, 2015.

Example dichotomy theorems in DB theory

- Probabilistic databases
  - Self-join (SJ) free: [Dalvi, Suciu, VLDB 2004]
  - SJ: [Dalvi, Suciu, JACM 2012]
- Resilience
  - SJ-free: [Freire+, VLDB 2015]
  - SJ: open (some progress in [Freire+, PODS 2020])
- View-side effect problem
  - SJ free with FDs [Kimelfeld, PODS 2012]
- Consistent query answering
  - SJ-free: [Koutris, Wijsen, PODS 2015]

Source: Dalvi, Suciu. "Efficient query evaluation on probabilistic databases", VLDB 2004. https://dl.acm.org/doi/abs/10.5555/1316689.1316764 , Dalvi, Suciu. "The dichotomy of probabilistic inference for unions of conjunctive queries", JACM 2012. https://doi.org/10.1145/2395116.2395119 , Freire, Gatterbauer, Immerman, Meliou, The complexity of resilience and responsibility for self-join-free conjunctive queries. PVLDB 2015. https://doi.org/10.14778/2850583.2850592 , Freire, Gatterbauer, Immerman, Meliou. New Results for the Complexity of Resilience for Binary Conjunctive Queries with Self-Joins. PODS 2020. https://doi.org/10.1145/3375395.3387647 , Kimelfeld. "A dichotomy in the complexity of deletion propagation with functional dependencies". PODS 2012. https://doi.org/10.1145/2213556.2213584 ,Koutris, Wijsen, "The Data Complexity of Consistent Query Answering for Self-Join-Free Conjunctive Queries Under Primary Key Constraints", PODS 2015, https://doi.org/10.1145/2745754.2745769

Reverse Data Management



Dichotomy theorems: Example: Resilience

Definition

Given a database and a query, what is the minimum set of tuples we must delete in order to change the query result?

Example

Consider Boolean query q:– $R(x,y)$, $S(y,z,w)$, $T(z,w)$.

R

	X	Y
r_1	1	3
r_2	1	4
r_3	2	5

S

	Y	Z	W
s_1	3	5	7
s_2	3	6	7
s_3	4	5	7

T

	Z	W
t_1	5	7

Tuples $\{r_1, s_1, t_1\}$ and $\{r_2, s_3, t_1\}$ join. Therefore q is true.

Delete set $\Gamma = \{r_2, s_1\}$. Γ is a contingency set.

$\Gamma_{\min} = \{t_1\}$.

Source: Freire, Gatterbauer, Immerman, Meliou, The complexity of resilience and responsibility for self-join-free conjunctive queries. PVLDB 2015. https://doi.org/10.14778/2850583.2850592

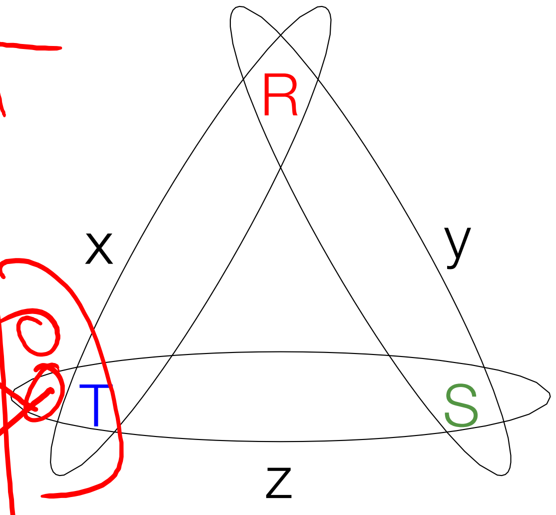
Dichotomy theorems: Example: Resilience

Reserach question

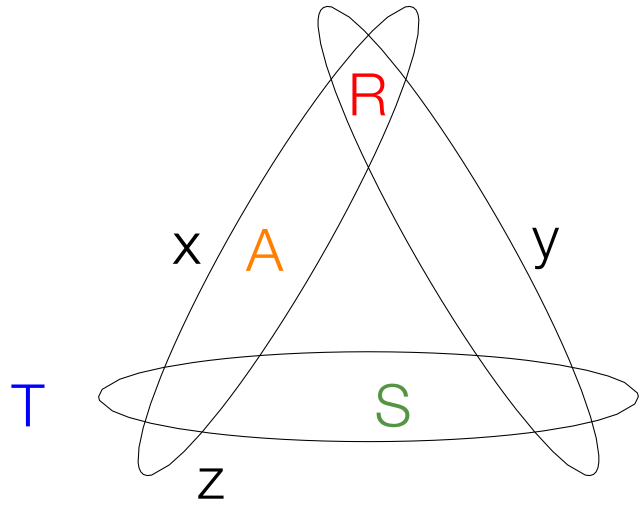
How difficult is it to find a minimal contingency set?

$q_{\triangle}\text{:-}R(x,y), S(y,z), T(z,x)$ $\qquad$ $q_{\text{rats}}\text{:-}A(x), R(x,y), S(y,z), T(z,x)$

R S T



Triangle query



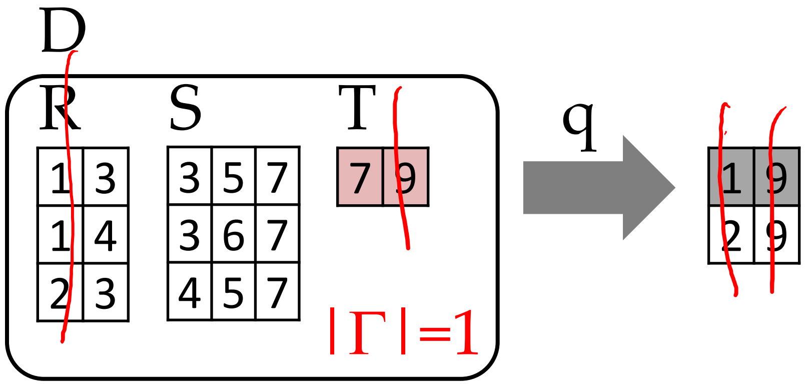
rats query

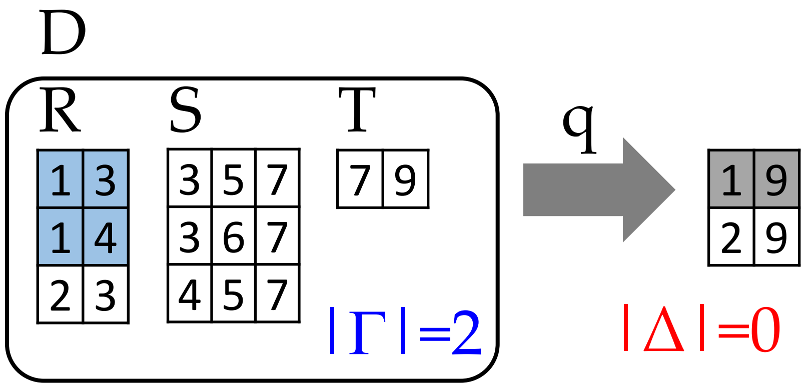
Lemma

RES(q_D) is NP-complete. But RES(q_{rats}) is in P!

Fig_SourceSideEffect

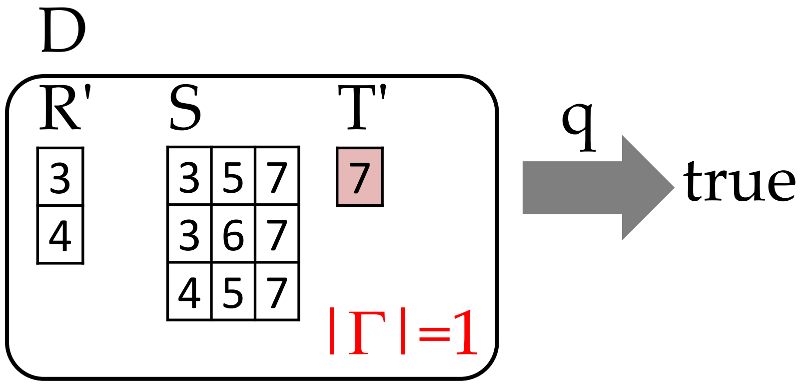
Fig_ViewSideEffect

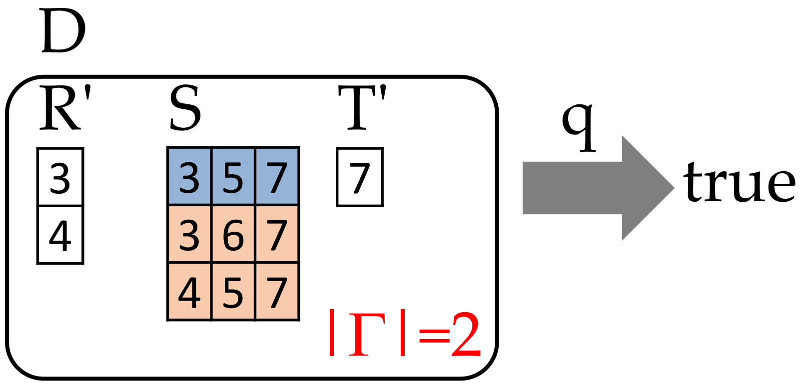




Fig_Resilience

Fig_Responsibility





2 appearances of R (“2-patterns” for this paper)

$\bigcup_i \mathtt{var}(R_i)$	name	position in linear order
2	permutation	$x \underset{R}{\overset{R}{\rightleftarrows}} y$
1	chain	$x \xrightarrow{R} y \xrightarrow{R} z$
	confluence	$x \xrightarrow{R} y \xleftarrow{R} z$
0	path	$x \xrightarrow{R} y \qquad z \xrightarrow{R} w$ $x \xrightarrow{R} y \qquad z \xleftarrow{R} w$ $\overset{A}{\circlearrowleft} x \qquad y \qquad \overset{A}{\circlearrowleft} z$