CSE599d: Advanced Query Processing

Lecture 2: Provenance, Semirings, and K-Relations

Dan Suciu

University of Washington

Provenance[1]

[1] Slides borrowed from [Tannen, 2023]

Binary Trust

Dan's notes (⋆)

Pred.	Prey
cat	mouse
cat	rat

Val's notes (⋆)

Animal	*Color*
mouse	gray
mouse	red
rat	gray

COMPUTATION ⇒

Sudeepa's data (⋆⋆)

Pred.	Food color
cat	gray
cat	red

(⋆) Dan and Val are noted zoologists

(⋆⋆) Sudeepa is a noted computational zoologist.

Binary Trust

D

Pred.	Prey	
cat	mouse	Y
cat	rat	Y

V

Animal	*Color*	
mouse	gray	N
mouse	red	N
rat	gray	Y

COMPUTATION $\Rightarrow$

$S(x,z)$:- $D(x,y) \wedge V(y,z)$

S

Pred.	Food color	
cat	gray	Y
cat	red	N

$Y = (Y \wedge N) \vee (Y \wedge Y)$

$N = Y \wedge N$

Access Control

D

Pred.	Prey	
cat	mouse	P
cat	rat	P

$S(x,z)$:- $D(x,y) \wedge V(y,z)$

COMPUTATION $\Rightarrow$

S

Pred.	Food color	
cat	gray	*C*
cat	red	*T*

V

Animal	*Color*	
mouse	gray	*T*
mouse	red	*T*
rat	gray	*C*

P=public < C=confidential < S=secret < T=top secret

Confidence Scores (non-binary trust)

D

Pred.	Prey	
cat	mouse	0.9
cat	rat	0.9

V

Animal	*Color*	
mouse	gray	0.6
mouse	red	0.1
rat	gray	0.8

COMPUTATION $\Rightarrow$

$S(x, z)$:- $D(x, y) \wedge V(y, z)$

S

Pred.	Food color	
cat	gray	0.72
cat	red	0.09

$0.72 = \max(0.9 \cdot 0.8,\ 0.9 \cdot 0.6)$

$0.09 = 0.9 \cdot 0.1$

Provenance

Sudeepa's data depends on Dan and Val's data. But how?

Provenance

Sudeepa's data depends on Dan and Val's data. But how?

Provenance is an abstract way of capturing this dependence. Use the previous examples as sanity checks.

Provenance

Sudeepa's data depends on Dan and Val's data. But how?

Provenance is an abstract way of capturing this dependence. Use the previous examples as sanity checks.

Useful: do provenance once and use it repeatedly for different applications.

Provenance

Sudeepa's data depends on Dan and Val's data. But how?

Provenance is an abstract way of capturing this dependence. Use the previous examples as sanity checks.

Useful: do provenance once and use it repeatedly for different applications.

Label (annotate) input items abstractly with provenance tokens.

Provenance

Sudeepa's data depends on Dan and Val's data. But how?

Provenance is an abstract way of capturing this dependence. Use the previous examples as sanity checks.

Useful: do provenance once and use it repeatedly for different applications.

Label (annotate) input items abstractly with provenance tokens.

Provenance tracking: propagate expressions involving tokens, to annotate intermediate data and, finally, outputs.

Provenance Expression Propagation

D

Pred.	Prey	
cat	mouse	p
cat	rat	q

$$S(x,z) \text{ :- } D(x,y) \wedge V(y,z)$$

V

Animal	*Color*	
mouse	gray	r
mouse	red	s
rat	gray	t

S

Pred.	Food color	
cat	gray	$p \cdot r + q \cdot t$
cat	red	$p \cdot s$

$p \cdot r + q \cdot t$ is the abstract version of $0.72 = \max(0.9 \cdot 0.8,\ 0.9 \cdot 0.6)$

End of slides borrowed from [Tannen, 2023]

Semirings

Monoids

Definition A monoid is a tuple $\boldsymbol{M} = (M, \circ, \mathbf{1})$, where:

- $\circ : M \times M \to M, \quad \mathbf{1} \in M.$
- $\circ$ is associative: $(x \circ y) \circ z = x \circ (y \circ z)$.
- $\mathbf{1}$ is a left and right identity: $\mathbf{1} \circ x = x \circ \mathbf{1} = x$.

The monoid is commutative if $x \circ y = y \circ x$.

Examples:

$(\mathbb{N}, +, 0)$, $([0, \infty], \min, \infty)$, $(\{0, 1\}, \vee, 0)$.

Semirings

Definition A semiring is a tuple $\boldsymbol{S} = (S, \oplus, \otimes, \mathbf{0}, \mathbf{1})$ where:

- $(\boldsymbol{S}, \oplus, \mathbf{0})$ commutative monoid; $(\boldsymbol{S}, \otimes, \mathbf{1})$ monoid.
- $\otimes$ distributes over $\oplus$:

$$x \otimes (y \oplus z) = (x \otimes y) \oplus (x \otimes z)$$
$$(y \oplus z) \otimes x = (y \otimes x) \oplus (z \otimes x)$$

- $\mathbf{0}$ is absorbing, also called annihilating: $x \otimes \mathbf{0} = \mathbf{0} \otimes x = \mathbf{0}$

$\boldsymbol{S}$ is a commutative semiring if $\otimes$ is commutative.

Examples

$\mathbb{B} = (\{0, 1\}, \vee, \wedge, 0, 1)$
Booleans

$V = ([0, 1], \max, *, 0, 1)$
Viterbi semiring
confidence scores

$(\mathbb{N}, +, \cdot, 0, 1)$
Natural numbers

$(\mathbb{N}[\boldsymbol{x}], +, \cdot, 0, 1)$
Multivariate polynomials in variables $\boldsymbol{x} = \{x_1, x_2, \ldots\}$

$\mathsf{Trop} = ([0, \infty], \min, +, \infty, 0)$
Tropical Semiring

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

K-Relations

Overview

A standard relation associates to each tuple a Boolean value: 0 or 1.

A K-relation associates to each tuple a value from a semiring K.

By choosing different semirings, we can support different applications.

K-Relations [Green et al., 2007]

Domain Dom, semiring $\boldsymbol{K} = (K, \oplus, \otimes, \mathbf{0}, \mathbf{1})$.

Definition

A K-relation is a function $R : \mathsf{Dom}^m \to K$ with "finite support":

$$\mathsf{Supp}(R) \stackrel{\text{def}}{=} \{t \in \mathsf{Dom}^m \mid R(t) \neq \mathbf{0}\} \text{ is finite.}$$

K-Relations [Green et al., 2007]

Domain Dom, semiring $\boldsymbol{K} = (K, \oplus, \otimes, \mathbf{0}, \mathbf{1})$.

Definition

A K-relation is a function $R : \mathsf{Dom}^m \to K$ with "finite support":

$$\mathsf{Supp}(R) \stackrel{\text{def}}{=} \{t \in \mathsf{Dom}^m \mid R(t) \neq \mathbf{0}\} \text{ is finite.}$$

A $\mathbb{B}$-relation:

Name	City	
Alice	SF	1
Alice	NYC	0
Bob	Seattle	1

Set semantics:

2 tuples

K-Relations [Green et al., 2007]

Domain Dom, semiring $\boldsymbol{K} = (K, \oplus, \otimes, \mathbf{0}, \mathbf{1})$.

Definition

A K-relation is a function $R : \mathsf{Dom}^m \to K$ with "finite support":

$$\mathsf{Supp}(R) \stackrel{\text{def}}{=} \{t \in \mathsf{Dom}^m \mid R(t) \neq \mathbf{0}\} \text{ is finite.}$$

A $\mathbb{B}$-relation:

Name	City	
Alice	SF	1
Alice	NYC	0
Bob	Seattle	1

Set semantics:

2 tuples

A $\mathbb{N}$-relation:

Name	City	
Alice	SF	5
Alice	NYC	0
Bob	Seattle	3

Bag semantics:

8 tuples

K-Relations [Green et al., 2007]

Domain Dom, semiring $\boldsymbol{K} = (K, \oplus, \otimes, \mathbf{0}, \mathbf{1})$.

Definition

A K-relation is a function $R : \mathsf{Dom}^m \to K$ with "finite support":

$$\mathsf{Supp}(R) \stackrel{\text{def}}{=} \{t \in \mathsf{Dom}^m \mid R(t) \neq \mathbf{0}\} \text{ is finite.}$$

A $\mathbb{B}$-relation:

Name	City	
Alice	SF	1
Alice	NYC	0
Bob	Seattle	1

Set semantics:
2 tuples

A $\mathbb{N}$-relation:

Name	City	
Alice	SF	5
Alice	NYC	0
Bob	Seattle	3

Bag semantics:
8 tuples

An $\mathbb{R}$-relation:

Name	City	
Alice	SF	-0.5
Alice	NYC	0.1
Bob	Seattle	3.4

A sparse 2×3 matrix

Discussion

- In [Green et al., 2007] queries have standard notation, e.g. $E(X,Y) \wedge E(Y,Z)$, but compute both the standard answer and the provenance annotation.

- We take a more radical view: the query computes the semiring value. New notation: sum-product query, e.g. $E[X,Y] \otimes E[Y,Z]$

Sum-product Queries

Sum-Product Queries, SPQ

Conjunctive Query: $Q(\boldsymbol{X}) = \exists \boldsymbol{Y}\,(R_1(\boldsymbol{Z}_1) \wedge R_2(\boldsymbol{Z}_2) \wedge \cdots)$

or, simply: $Q(\boldsymbol{X}) = R_1(\boldsymbol{Z}_1) \wedge R_2(\boldsymbol{Z}_2) \wedge \cdots$

$$Q(\boldsymbol{D}) \stackrel{\text{def}}{=} \{\boldsymbol{x} \in \mathsf{Dom}^{\boldsymbol{X}} \mid \exists h : Q \to D, h(\boldsymbol{X}) = \boldsymbol{x}\}$$

Sum-Product Queries, SPQ

Conjunctive Query: $Q(\boldsymbol{X}) = \exists \boldsymbol{Y}\,(R_1(\boldsymbol{Z}_1) \wedge R_2(\boldsymbol{Z}_2) \wedge \cdots)$
or, simply: $Q(\boldsymbol{X}) = R_1(\boldsymbol{Z}_1) \wedge R_2(\boldsymbol{Z}_2) \wedge \cdots$

$$Q(\boldsymbol{D}) \stackrel{\text{def}}{=} \{\boldsymbol{x} \in \mathsf{Dom}^{\boldsymbol{X}} \mid \exists h : Q \to D, h(\boldsymbol{X}) = \boldsymbol{x}\}$$

Sum-Product Query: $Q(\boldsymbol{X}) = \bigoplus_{\boldsymbol{Y}} (R_1[\boldsymbol{Z}_1] \otimes R_2[\boldsymbol{Z}_2] \otimes \cdots)$
or, simply: $Q(\boldsymbol{X}) = R_1[\boldsymbol{Z}_1] \otimes R_2[\boldsymbol{Z}_2] \otimes \cdots$

Sum-Product Queries, SPQ

Conjunctive Query: $Q(\boldsymbol{X}) = \exists \boldsymbol{Y}\,(R_1(\boldsymbol{Z}_1) \wedge R_2(\boldsymbol{Z}_2) \wedge \cdots)$
or, simply: $Q(\boldsymbol{X}) = R_1(\boldsymbol{Z}_1) \wedge R_2(\boldsymbol{Z}_2) \wedge \cdots$

$$Q(\boldsymbol{D}) \stackrel{\text{def}}{=} \{\boldsymbol{x} \in \mathsf{Dom}^{\boldsymbol{X}} \mid \exists h : Q \to D, h(\boldsymbol{X}) = \boldsymbol{x}\}$$

Sum-Product Query: $Q(\boldsymbol{X}) = \bigoplus_{\boldsymbol{Y}} (R_1[\boldsymbol{Z}_1] \otimes R_2[\boldsymbol{Z}_2] \otimes \cdots)$
or, simply: $Q(\boldsymbol{X}) = R_1[\boldsymbol{Z}_1] \otimes R_2[\boldsymbol{Z}_2] \otimes \cdots$

$$Q(\boldsymbol{D})[\boldsymbol{x}] \stackrel{\text{def}}{=} \bigoplus_{h:Q\to\boldsymbol{D}:h(\boldsymbol{X})=\boldsymbol{x}} R_1[h(\boldsymbol{Z}_1)] \otimes R_2[h(\boldsymbol{Z}_2)] \otimes \cdots$$

Sum-Product Queries, SPQ

In a nutshell, the semantics over K-relations is:

Replace $\vee$, $\wedge$, $\exists x$ with $\oplus$, $\otimes$, $\bigoplus_x$

Example: Sparse Tensors

$\mathbb{R}$-relations are logically equivalent to sparse tensors.

Example: Sparse Tensors

$\mathbb{R}$-relations are logically equivalent to sparse tensors.

A sparse matrix:

$$A = \begin{pmatrix} 9 & 0 & 0 \\ 0 & 0 & 7 \\ 1.1 & -5 & 0 \end{pmatrix}$$

Representation as an $\mathbb{R}$-relation:

i	j	
1	1	9
2	3	7
3	1	1.1
3	2	−5

Example: Sparse Tensors

$\mathbb{R}$-relations are logically equivalent to sparse tensors.

A sparse matrix:

$$A = \begin{pmatrix} 9 & 0 & 0 \\ 0 & 0 & 7 \\ 1.1 & -5 & 0 \end{pmatrix}$$

Representation as an $\mathbb{R}$-relation:

i	j	
1	1	9
2	3	7
3	1	1.1
3	2	−5

A CQ:

$$Q(X,Z) = \exists Y (A(X,Y) \wedge B(Y,Z))$$

A SPQ:

$$Q[i,k] = \sum_j A[i,j] \cdot B[j,k]$$

Einsums[2]: Drop Quantifier $\bigoplus$

MM: $Q[i,k] = A[i,j] \cdot B[j,k]$ `torch.einsum('ij,jk->ik', [a,b])`

[2] https://rockt.github.io/2018/04/30/einsum

Einsums[2]: Drop Quantifier $\bigoplus$

MM: $Q[i,k] = A[i,j] \cdot B[j,k]$ `torch.einsum('ij,jk->ik', [a,b])`

Transpose: $B[i,j] = A[i,j]$ `torch.einsum('ij->ji', [a])`

[2] https://rockt.github.io/2018/04/30/einsum

Einsums[2]: Drop Quantifier $\bigoplus$

MM: $Q[i,k] = A[i,j] \cdot B[j,k]$ `torch.einsum('ij,jk->ik', [a,b])`

Transpose: $B[i,j] = A[i,j]$ `torch.einsum('ij->ji', [a])`

Summation: $S = A[i,j]$ `torch.einsum('ij->', [a])`

[2] https://rockt.github.io/2018/04/30/einsum

Einsums[2]: Drop Quantifier $\bigoplus$

MM: $Q[i,k] = A[i,j] \cdot B[j,k]$ `torch.einsum('ij,jk->ik', [a,b])`

Transpose: $B[i,j] = A[i,j]$ `torch.einsum('ij->ji', [a])`

Summation: $S = A[i,j]$ `torch.einsum('ij->', [a])`

Row sum: $R[i] = A[i,j]$ `torch.einsum('ij->i', [a])`

[2] https://rockt.github.io/2018/04/30/einsum

Einsums[2]: Drop Quantifier $\bigoplus$

MM: $Q[i,k] = A[i,j] \cdot B[j,k]$ `torch.einsum('ij,jk->ik', [a,b])`

Transpose: $B[i,j] = A[i,j]$ `torch.einsum('ij->ji', [a])`

Summation: $S = A[i,j]$ `torch.einsum('ij->', [a])`

Row sum: $R[i] = A[i,j]$ `torch.einsum('ij->i', [a])`

Dot product: $P = A[i] \cdot B[i]$ `torch.einsum('i,i->', [a, b])`

[2] https://rockt.github.io/2018/04/30/einsum

Einsums[2]: Drop Quantifier $\bigoplus$

MM: $Q[i,k] = A[i,j] \cdot B[j,k]$ `torch.einsum('ij,jk->ik', [a,b])`

Transpose: $B[i,j] = A[i,j]$ `torch.einsum('ij->ji', [a])`

Summation: $S = A[i,j]$ `torch.einsum('ij->', [a])`

Row sum: $R[i] = A[i,j]$ `torch.einsum('ij->i', [a])`

Dot product: $P = A[i] \cdot B[i]$ `torch.einsum('i,i->', [a, b])`

Outer product $T[i,j] = A[i] \cdot B[j]$ `torch.einsum('i,j->ij', [a, b])`

[2] https://rockt.github.io/2018/04/30/einsum

Einsums[2]: Drop Quantifier $\bigoplus$

MM: $Q[i,k] = A[i,j] \cdot B[j,k]$ `torch.einsum('ij,jk->ik', [a,b])`

Transpose: $B[i,j] = A[i,j]$ `torch.einsum('ij->ji', [a])`

Summation: $S = A[i,j]$ `torch.einsum('ij->', [a])`

Row sum: $R[i] = A[i,j]$ `torch.einsum('ij->i', [a])`

Dot product: $P = A[i] \cdot B[i]$ `torch.einsum('i,i->', [a, b])`

Outer product $T[i,j] = A[i] \cdot B[j]$ `torch.einsum('i,j->ij', [a, b])`

Batch MM: $C[i,k,m] = A[i,j,m] \cdot B[j,k,m]$

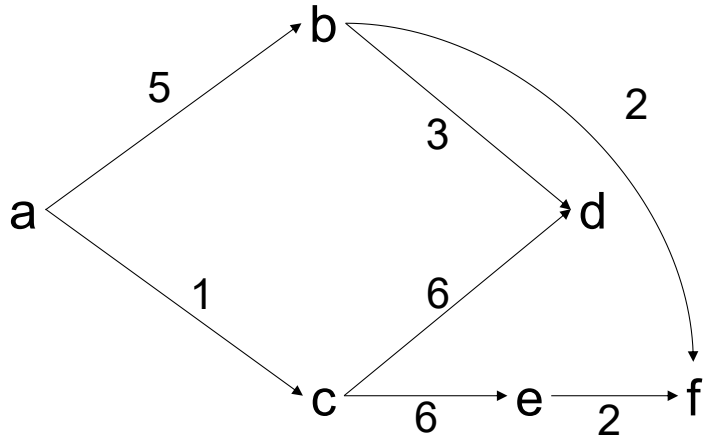
`torch.einsum('ijk,ikl->ijl',[a,b])`

[2] `https://rockt.github.io/2018/04/30/einsum`

Shortest Path

Labeled graph $G = (V, E)$

Trop $= ([0, \infty], \min, +, \infty, 0)$



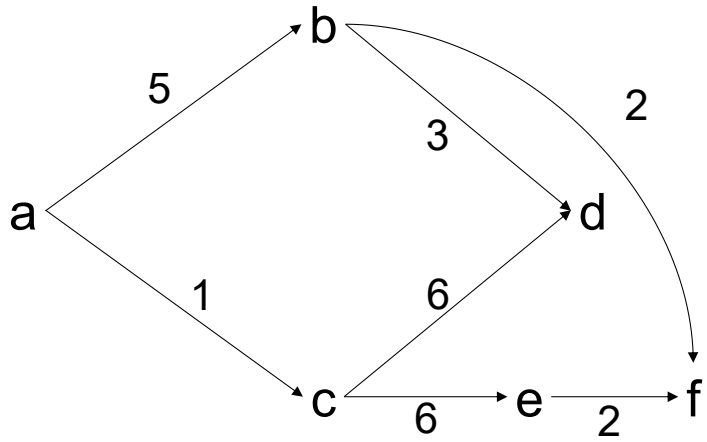
What does this query compute?

$$Q(X, Y) = E[X, Y] \oplus \bigoplus_Z (E[X, Z] \otimes E[Z, Y])$$

Shortest Path

Labeled graph $G = (V, E)$

$\text{Trop} = ([0, \infty], \min, +, \infty, 0)$



What does this query compute?

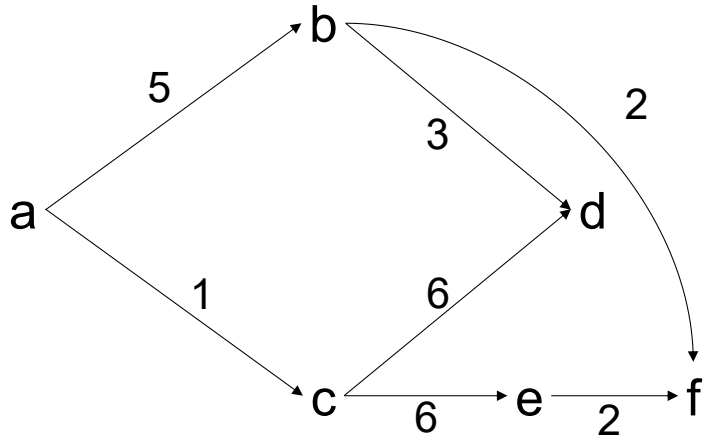
$$Q(X, Y) = E[X, Y] \oplus \bigoplus_Z (E[X, Z] \otimes E[Z, Y])$$

a	b	?
a	c	?
a	d	?
a	e	...
a	f	...
b	d	...
...	...	...

Shortest Path

Labeled graph $G = (V, E)$

$\text{Trop} = ([0, \infty], \min, +, \infty, 0)$



What does this query compute?

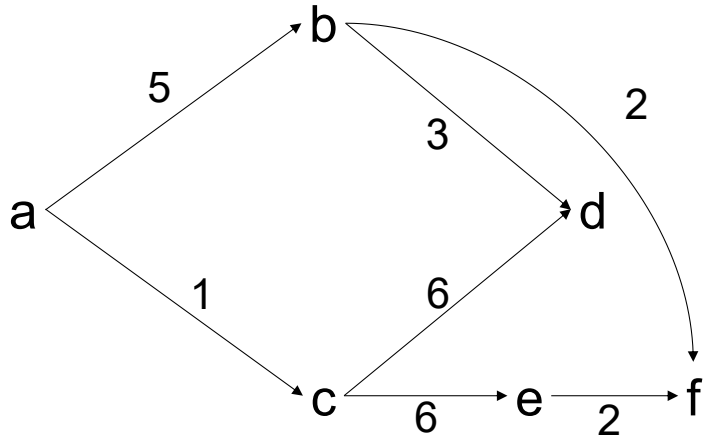
$$Q(X, Y) = E[X, Y] \oplus \bigoplus_Z (E[X, Z] \otimes E[Z, Y])$$

a	b	5
a	c	1
a	d	?
a	e	...
a	f	...
b	d	...
...	...	...

Shortest Path

Labeled graph $G = (V, E)$

$\text{Trop} = ([0, \infty], \min, +, \infty, 0)$



What does this query compute?

$$Q(X, Y) = E[X, Y] \oplus \bigoplus_Z (E[X, Z] \otimes E[Z, Y])$$

a	b	5
a	c	1
a	d	7
a	e	...
a	f	...
b	d	...
...	...	...

. . . and Set Semantics!

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

$\mathbb{B}$-relations are standard relations under set semantics.

Evaluating a SPQ in the $\mathbb{B}$-semiring is standard evaluation of a CQ.

Mandatory Access Control

The access control semiring: $(\mathbb{A}, \min, \max, 0, P)$

$\mathbb{A} = \{\text{Public} < \text{Confidential} < \text{Secret} < \text{Top-secret} < 0\}$

0 means "No Such Thing"

Pics

PID	
p1	S
p2	T

Occ

PID	DID	
p1	d1	P
p2	d1	P
p2	d2	P

Docs

DID	
d1	C
d2	0

Mandatory Access Control

The access control semiring: $(\mathbb{A}, \min, \max, 0, P)$

$\mathbb{A} = \{\text{Public} < \text{Confidential} < \text{Secret} < \text{Top-secret} < 0\}$

0 means "No Such Thing"

Pics

PID	
p1	S
p2	T

Occ

PID	DID	
p1	d1	P
p2	d1	P
p2	d2	P

Docs

DID	
d1	C
d2	0

$$Q(p) = \text{Pics}(p) \wedge \text{Occ}(p, d) \wedge \text{Docs}(d)$$

Mandatory Access Control

The access control semiring: $(\mathbb{A}, \min, \max, 0, P)$

$\mathbb{A} = \{\text{Public} < \text{Confidential} < \text{Secret} < \text{Top-secret} < 0\}$

0 means "No Such Thing"

Pics

PID	
p1	S
p2	T

Occ

PID	DID	
p1	d1	P
p2	d1	P
p2	d2	P

Docs

DID	
d1	C
d2	0

Answer

PID	
p1	?
p2	?

$$Q(p) = \text{Pics}(p) \wedge \text{Occ}(p, d) \wedge \text{Docs}(d)$$

Mandatory Access Control

The access control semiring: $(\mathbb{A}, \min, \max, 0, P)$

$\mathbb{A} = \{\text{Public} < \text{Confidential} < \text{Secret} < \text{Top-secret} < 0\}$

0 means "No Such Thing"

Pics

PID	
p1	S
p2	T

Occ

PID	DID	
p1	d1	P
p2	d1	P
p2	d2	P

Docs

DID	
d1	C
d2	0

Answer

PID	
p1	S
p2	T

$$Q(p) = \text{Pics}(p) \wedge \text{Occ}(p, d) \wedge \text{Docs}(d)$$

Discussion

- K-Relations: powerful abstraction that allows us to apply concepts from the relational model to other domains
- Einsum notation popular in ML: numpy, TensorFlow, pytorch
- The original motivation of K-relations in [Green et al., 2007] was to model provenance. Will discuss next.

Provenance Polynomials

Provenance Polynomials

Input tuple: annotated with provenance token $\boldsymbol{x} = \{x_1, x_2, \ldots\}$
Output tuple: annotated with provenance polynomial $\mathbb{N}[\boldsymbol{x}]$

$R =$

A	B	C	
a	b	c	x
d	b	e	y
f	g	e	z

$Q =$

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

$$Q(A, C) =$$
$$\exists A_1 B_1 C_1 (R(A, B_1, C_1) \wedge R(A_1, B_1, C))$$
$$\vee \exists A_1 B_1 B_2 (R(A, B_1, C) \wedge R(A_1, B_2, C))$$

[Green et al., 2007]

Provenance Polynomials

Input tuple: annotated with provenance token $\boldsymbol{x} = \{x_1, x_2, \ldots\}$
Output tuple: annotated with provenance polynomial $\mathbb{N}[\boldsymbol{x}]$

$R =$

A	B	C	
a	b	c	x
d	b	e	y
f	g	e	z

$Q =$

A	C	
a	c	$2x^2$
a	e	xy
d	c	xy
d	e	$2y^2 + yz$
f	e	$2z^2 + yz$

$$Q(A, C) =$$
$$\exists A_1 B_1 C_1 (R(A, B_1, C_1) \wedge R(A_1, B_1, C))$$
$$\vee \exists A_1 B_1 B_2 (R(A, B_1, C) \wedge R(A_1, B_2, C))$$

[Green et al., 2007]

Provenance Polynomials

Input tuple: annotated with provenance token $\boldsymbol{x} = \{x_1, x_2, \ldots\}$
Output tuple: annotated with provenance polynomial $\mathbb{N}[\boldsymbol{x}]$

$R =$

A	B	C	
a	b	c	x
d	b	e	y
f	g	e	z

$Q =$

A	C	
a	c	$2x^2$
a	e	xy
d	c	xy
d	e	$2y^2 + yz$
f	e	$2z^2 + yz$

$Q(A, C) =$
$\exists A_1 B_1 C_1 (R(A, B_1, C_1) \wedge R(A_1, B_1, C))$
$\vee \exists A_1 B_1 B_2 (R(A, B_1, C) \wedge R(A_1, B_2, C))$

(a, e) is derived from x and y.

[Green et al., 2007]

Provenance Polynomials

Input tuple: annotated with provenance token $\boldsymbol{x} = \{x_1, x_2, \ldots\}$
Output tuple: annotated with provenance polynomial $\mathbb{N}[\boldsymbol{x}]$

$R =$

A	B	C	
a	b	c	x
d	b	e	y
f	g	e	z

$Q =$

A	C	
a	c	$2x^2$
a	e	xy
d	c	xy
d	e	$2y^2 + yz$
f	e	$2z^2 + yz$

$Q(A, C) =$
$\exists A_1 B_1 C_1 (R(A, B_1, C_1) \wedge R(A_1, B_1, C))$
$\vee \exists A_1 B_1 B_2 (R(A, B_1, C) \wedge R(A_1, B_2, C))$

[Green et al., 2007]

(a, e) is derived from x and y.

(a, c) is derived in two ways:
using x twice, and using x twice again.

Provenance Polynomials

Input tuple: annotated with provenance token $\boldsymbol{x} = \{x_1, x_2, \ldots\}$
Output tuple: annotated with provenance polynomial $\mathbb{N}[\boldsymbol{x}]$

$R =$

A	B	C	
a	b	c	x
d	b	e	y
f	g	e	z

$Q(A, C) =$

$\exists A_1 B_1 C_1 (R(A, B_1, C_1) \wedge R(A_1, B_1, C))$

$\vee \exists A_1 B_1 B_2 (R(A, B_1, C) \wedge R(A_1, B_2, C))$

[Green et al., 2007]

$Q =$

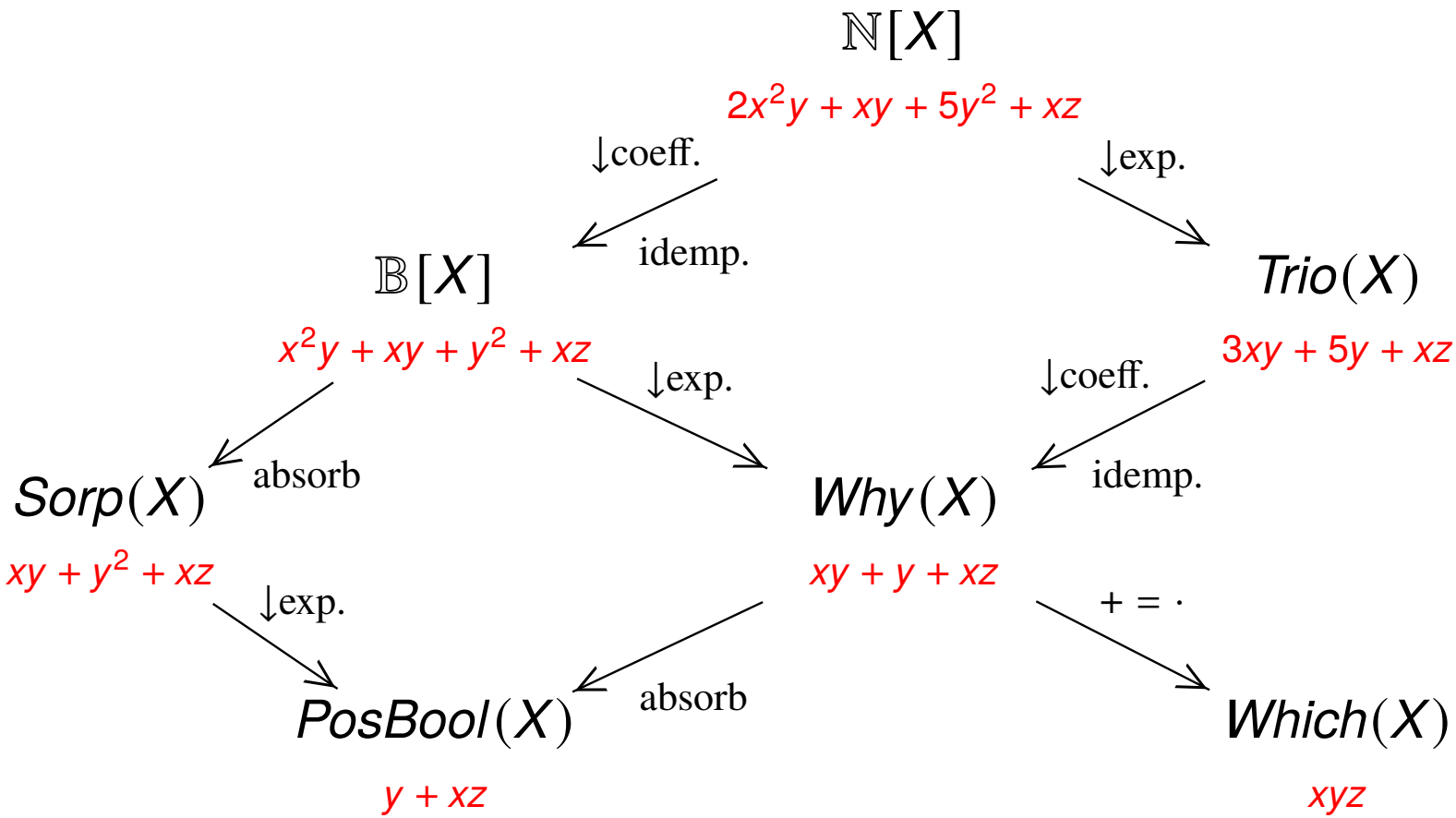
A	C	
a	c	$2x^2$
a	e	xy
d	c	xy
d	e	$2y^2 + yz$
f	e	$2z^2 + yz$

(a, e) is derived from x and y.

(a, c) is derived in two ways:
using x twice, and using x twice again.

(d, e) is derived . . .

Landscape of Provenance Formalisms



Query Equivalence

Query Equivalence

Last lecture:

$$\boxed{Q_1 \equiv Q_2 \text{ if } Q_1(\boldsymbol{D}) = Q_2(\boldsymbol{D}) \text{ for all } \boldsymbol{D}, \text{ using set semantics.}}$$

Now:

$$\boxed{Q_1 \equiv Q_2 \text{ if } Q_1(\boldsymbol{D}) = Q_2(\boldsymbol{D}) \text{ for all } \boldsymbol{D} \text{ and all semirings } K.}$$

It turns out that this is the same as equivalence under bag semantics!

Semiring Homomorphisms, and Universality Property

A homomorphism $f : (S, \oplus, \otimes, \mathbf{0}, \mathbf{1}) \to (K, +, \cdot, 0, 1)$ is a function $f : S \to K$ such that:

$$f(\mathbf{0}) = 0 \qquad f(\mathbf{1}) = 1$$

$$f(x \oplus y) = f(x) + f(y) \qquad f(x \otimes y) = f(x) \cdot f(y)$$

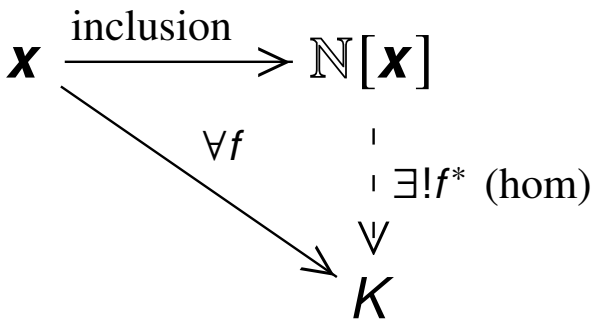
Semiring Homomorphisms, and Universality Property

A homomorphism $f : (S, \oplus, \otimes, \mathbf{0}, \mathbf{1}) \to (K, +, \cdot, 0, 1)$ is a function $f : S \to K$ such that:

$$f(\mathbf{0}) = 0 \qquad f(\mathbf{1}) = 1$$

$$f(x \oplus y) = f(x) + f(y) \qquad f(x \otimes y) = f(x) \cdot f(y)$$

Theorem $(\mathbb{N}[\boldsymbol{x}], +, \cdot, 0, 1)$ is the freely generated commutative semiring



Applications to Query Equivalence

Fix a semiring identity $E_1 = E_2$. The following are equivalent:

1. $E_1 = E_2$ holds in $(\mathbb{N}, +, \cdot, 0, 1)$.
2. $E_1 = E_2$ holds in $(\mathbb{N}[\boldsymbol{x}], +, \cdot, 0, 1)$.
3. $E_1 = E_2$ holds in all commutative semirings.

Proof (in class) Item 1 $\Rightarrow$ Item 2 $\Rightarrow$ Item 3 $\Rightarrow$ Item 1

Applications to Query Equivalence

Fix a semiring identity $E_1 = E_2$. The following are equivalent:

1. $E_1 = E_2$ holds in $(\mathbb{N}, +, \cdot, 0, 1)$.
2. $E_1 = E_2$ holds in $(\mathbb{N}[\boldsymbol{x}], +, \cdot, 0, 1)$.
3. $E_1 = E_2$ holds in all commutative semirings.

Proof (in class) Item 1 $\Rightarrow$ Item 2 $\Rightarrow$ Item 3 $\Rightarrow$ Item 1

This identity in $(\mathbb{N}[x, y, z], +, *, 0, 1)$:

$$(x+y)(x+z)(y+z) = xy(x+y) + xz(x+z) + yz(y+z) + 2xyz$$

implies this identity in $\mathsf{Trop} = ([0, \infty], \min, +, \infty, 0)$:

$$\min(a,b) + \min(a,c) + \min(b,c) = \min(a+b+\min(a,b),\ a+c+\min(a,c),\ b+c+\min(b,c),\ a+b+c)$$

Conclusion

- An identity of SPQs $E_1 \equiv E_2$ holds in all semirings iff it holds under bag semantics, i.e. $(\mathbb{N}, +, *, 0, 1)$.
- In particular, $R \bowtie R \equiv R$ does NOT hold in all semirings; it only holds in idempotent semirings, where $x \otimes x = x$.
- Similarly for $R \cup R \equiv R$.
- Cost-based query optimizers designed for SQL apply, out of the box, to any other domains, e.g. Einsum kernels.

Green, T. J., Karvounarakis, G., and Tannen, V. (2007).

Provenance semirings.

In Libkin, L., editor, Proceedings of the Twenty-Sixth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 11-13, 2007, Beijing, China, pages 31–40. ACM.

Tannen, V. (2023).

Algebraic Structures in Logic and Query Evaluation 1.

https://simons.berkeley.edu/talks/val-tannen-university-pennsylvania-2023-08-23.

Workshop on Logic and Algorithms in Database Theory and AI Boot Camp.