Provenance Semirings

Todd Green   Grigoris Karvounarakis   Val Tannen

presented by Clemens Ley

“place of origin”

Provenance Semirings

Todd Green   Grigoris Karvounarakis   Val Tannen

presented by Clemens Ley

“place of origin”

algebraic structure, e.g $(\mathbb{N},*,+,0,1)$

Provenance Semirings

Todd Green Grigoris Karvounarakis Val Tannen

presented by Clemens Ley

Outline

- Data provenance by example
- Relational algebra for data provenance
- Datalog for data provenance

Data Provenance

Data Provenance

Data provenance aims to explain how a particular query result was obtained.

Data Provenance

R:

A	B	C
	...	
a	b	c
	...	

S:

D	B	E
	...	
d	b	e
	...	

join on **B**

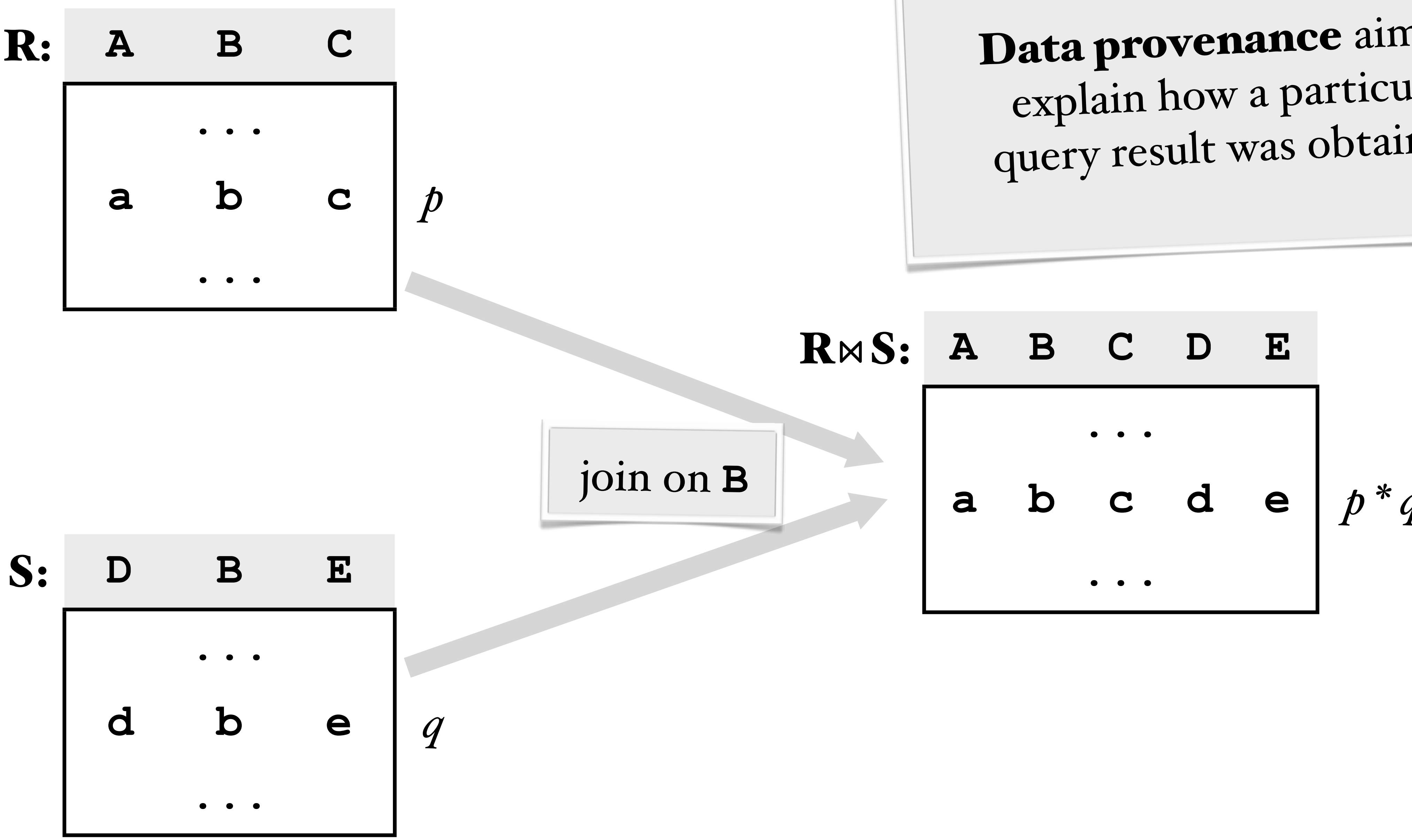
R⋈S:

A	B	C	D	E
		...		
a	b	c	d	e
		...		

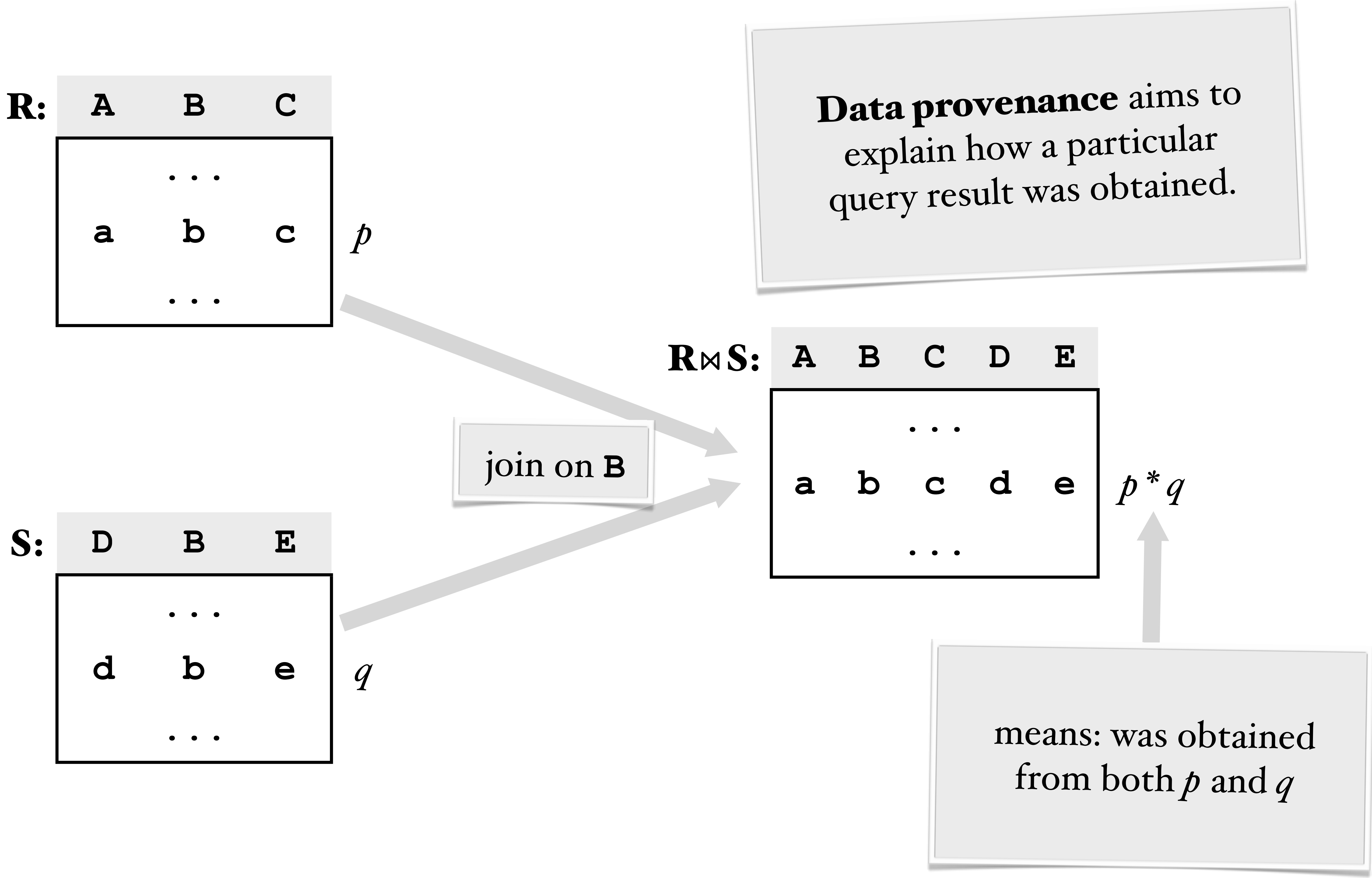
Data provenance aims to explain how a particular query result was obtained.

Data Provenance

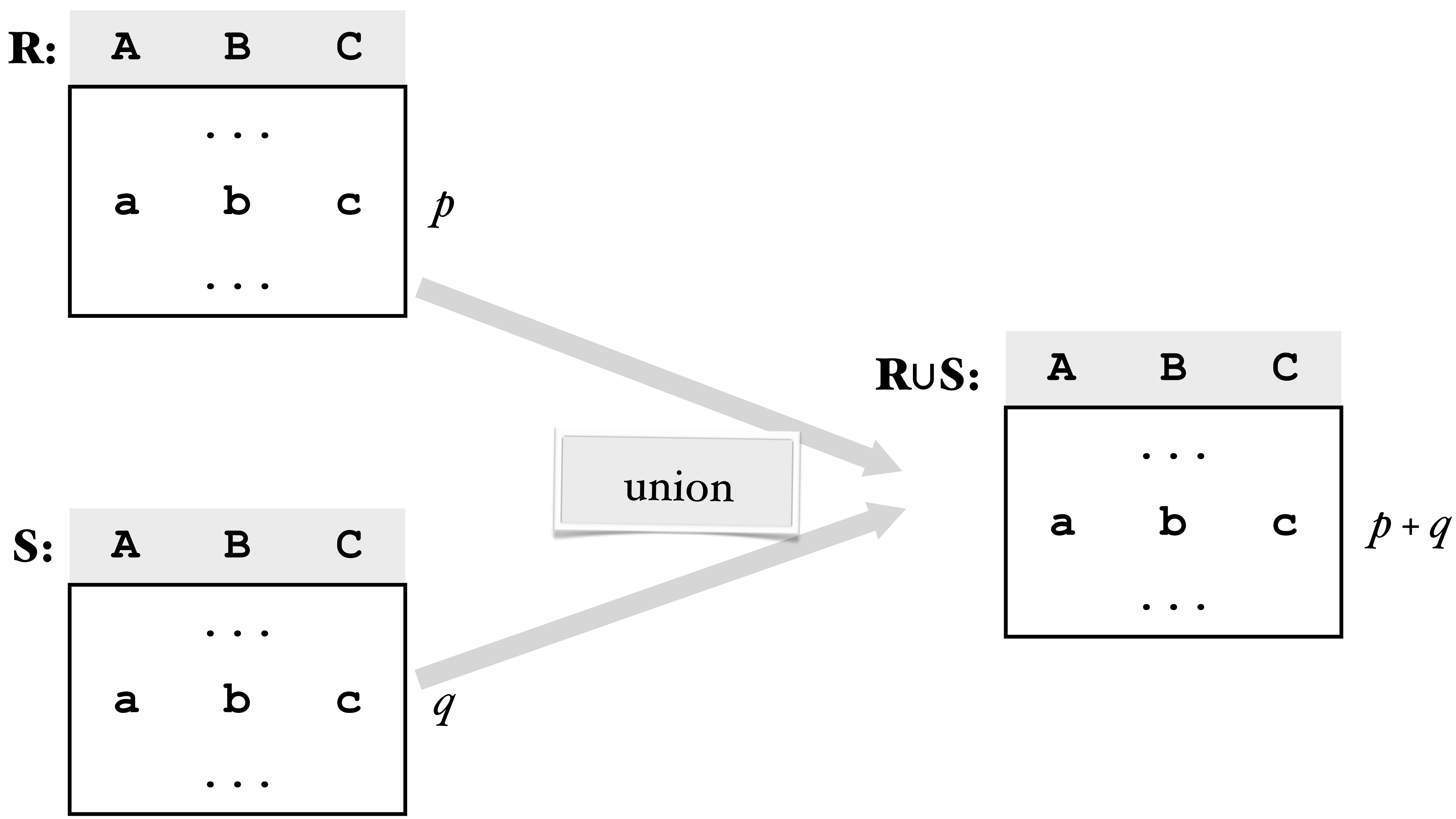
Data provenance aims to explain how a particular query result was obtained.



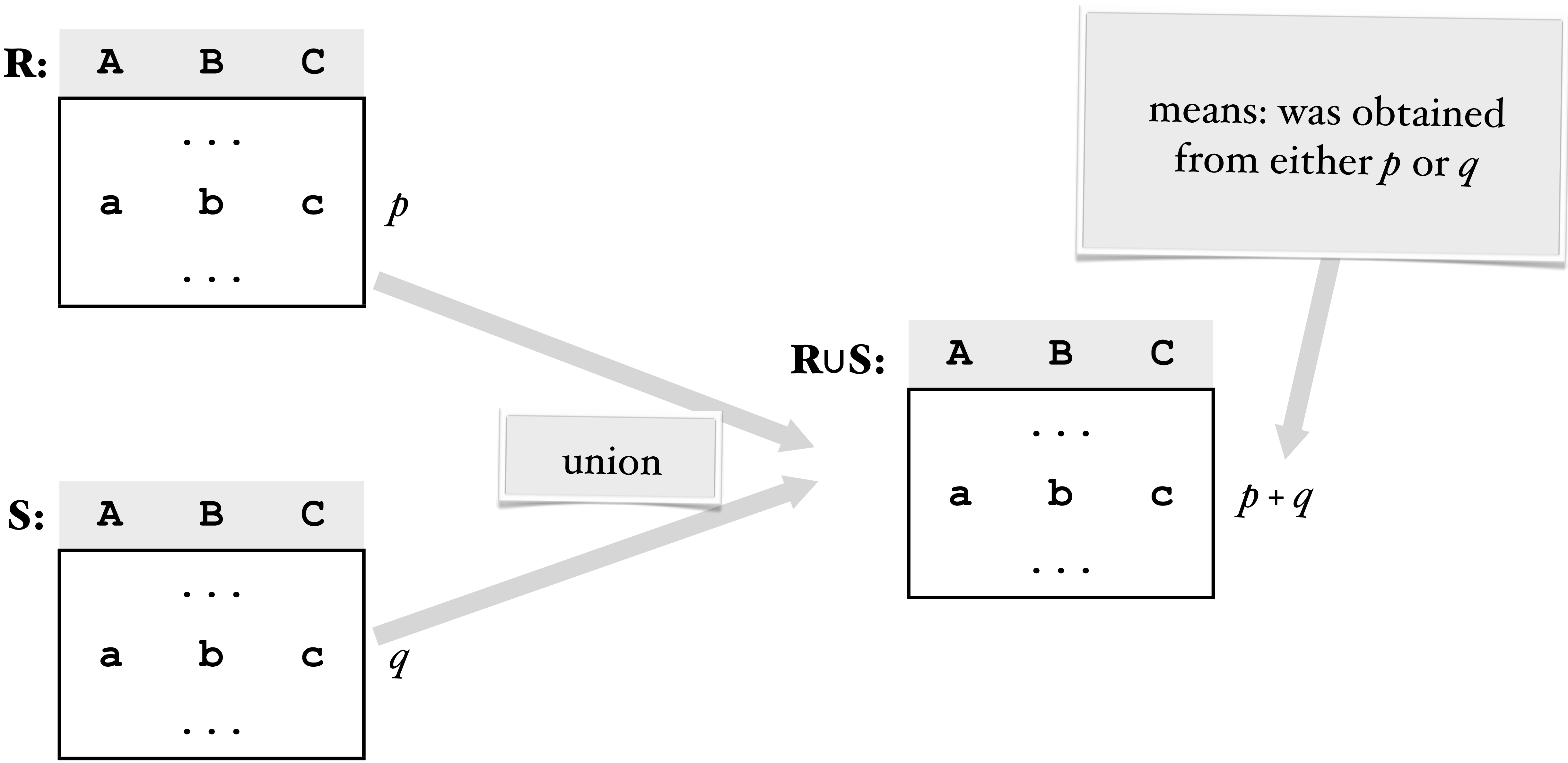
Data Provenance



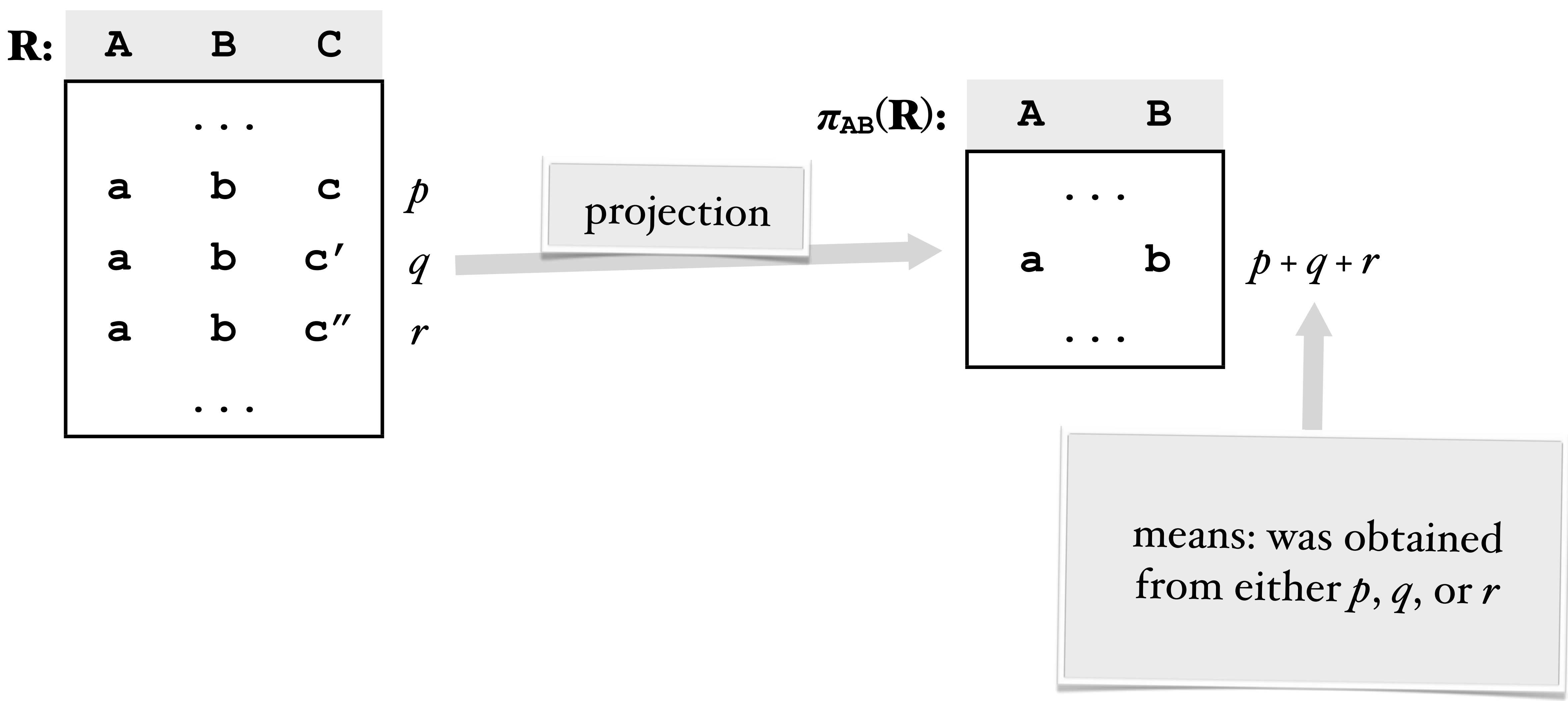
Data Provenance (2)



Data Provenance (2)



Data Provenance (3)



Data Provenance (4)

$$\mathbf{Q} = \sigma_{C=e}\, \pi_{AC}(\, \pi_{AC}\mathbf{R} \bowtie \pi_{BC}\mathbf{R} \cup \pi_{AB}\mathbf{R} \bowtie \pi_{BC}\mathbf{R})$$

R:

A	B	C	
a	b	c	*p*
d	b	e	*r*
f	g	e	*s*

Q:

A	C	
a	c	$(p^2 + p^2) * 0$
a	e	$(pr) * 1$
d	c	$(rp) * 0$
d	e	$(r^2 + rs + r^2) * 1$
f	e	$(s^2 + rs + s^2) * 1$

Data Provenance (4)

$$\mathbf{Q} = \sigma_{C=e}\, \pi_{AC}(\, \pi_{AC}\mathbf{R} \bowtie \pi_{BC}\mathbf{R} \cup \pi_{AB}\mathbf{R} \bowtie \pi_{BC}\mathbf{R})$$

R:

A	B	C	
a	b	c	*p*
d	b	e	*r*
f	g	e	*s*

Q:

A	C	
a	c	$(p^2 + p^2) * 0$
a	e	$(pr) * 1$
d	c	$(rp) * 0$
d	e	$(r^2 + rs + r^2) * 1$
f	e	$(s^2 + rs + s^2) * 1$

for selection, multiply by *1* or *0*.

Why would this be useful?

Q:

A	C	
a	c	$(p^2 + p^2) * 0$
a	e	$(pr) * 1$
d	c	$(rp) * 0$
d	e	$(r^2 + rs + r^2) * 1$
f	e	$(s^2 + rs + s^2) * 1$

$$\mathbf{Q} = \sigma_{C=e}\, \pi_{AC}(\, \pi_{AC}\mathbf{R} \bowtie \pi_{BC}\mathbf{R} \cup \pi_{AB}\mathbf{R} \bowtie \pi_{BC}\mathbf{R})$$

R:

A	B	C	
a	b	c	*p*
d	b	e	*r*
f	g	e	*s*

for selection, multiply by *1* or *0*.

Why would this be useful?

$\mathbf{Q} = \sigma_{C=e}\, \pi_{AC}(\, \pi_{AC}\mathbf{R} \bowtie \pi_{BC}\mathbf{R} \cup \pi_{AB}\mathbf{R} \bowtie \pi_{BC}\mathbf{R})$

Q:

A	C	
a	c	$(p^2 + p^2) * 0 = 0$
a	e	$(pr) * 1 = 10$
d	c	$(rp) * 0 = 0$
d	e	$(r^2 + rs + r^2) * 1 = 55$
f	e	$(s^2 + rs + s^2) * 1 = 7$

R:

A	B	C	
a	b	c	$p := 2$
d	b	e	$r := 5$
f	g	e	$s := 1$

for **bag semantics**, consider annotations as multiplicities

Why would this be useful?

Q:

A	C	
a	c	$(p^2 + p^2) * 0 = ((b_1 \wedge b_1) \vee (b_1 \wedge b_1)) \wedge \text{false}$
a	e	$(pr) * 1 = (b_1 \wedge b_2) \wedge \text{true}$
d	c	$(rp) * 0 = (b_2 \wedge b_1) \wedge \text{false}$
d	e	$(r^2 + rs + r^2) * 1 = (b_2 \vee (b_2 \wedge b_3) \vee b_2) \wedge \text{true}$
f	e	$(s^2 + rs + s^2) * 1 = (b_3 \vee (b_2 \wedge b_3) \vee b_3) \wedge \text{true}$

R:

A	B	C	
a	b	c	$p := b_1$
d	b	e	$r := b_2$
f	g	e	$s := b_3$

for **incomplete databases**, consider annotations as boolean values,
* as $\wedge$, + as $\vee$, *1* as true, and *0* as false

Data Structure

- Relations are mappings from tuples to annotations in K; we require that $R(t) \neq 0$ for only finitely many tuples t.
- intuitively, "+" means "alternative use" corresponds to union
- "*" means "joint use" and corresponds to join
- "0" and "1" are special annotations

- But what is a query languages for such relations?

Data Structure

- Relations are **mappings from tuples to annotations in *K***; we require that $R(t) \neq 0$ for only finitely many tuples t.
- intuitively, "+" means "alternative use" corresponds to union
- "*" means "joint use" and corresponds to join
- "0" and "1" are special annotations

- But what is $(K,+,*,0,1)$ and how are annotations computed?

Positive Algebra

Positive Algebra

DEFINITION 3.2. *Let* $(K, +, \cdot, 0, 1)$ *be an algebraic structure with two binary operations and two distinguished elements. The operations of the* **positive algebra** *are defined as follows:*

empty relation *For any set of attributes* U*, there is* $\emptyset : U\text{-}\mathsf{Tup} \to K$ *such that* $\emptyset(t) = 0$*.*

union *If* $R_1, R_2 : U\text{-}\mathsf{Tup} \to K$ *then* $R_1 \cup R_2 : U\text{-}\mathsf{Tup} \to K$ *is defined by*

$$(R_1 \cup R_2)(t) \stackrel{\mathtt{def}}{=} R_1(t) + R_2(t)$$

projection *If* $R : U\text{-}\mathsf{Tup} \to K$ *and* $V \subseteq U$ *then* $\pi_V R : V\text{-}\mathsf{Tup} \to K$ *is defined by*

$$(\pi_V R)(t) \stackrel{\mathtt{def}}{=} \sum_{t=t' \text{ on } V \text{ and } R(t') \neq 0} R(t')$$

(here $t = t'$ *on* V *means* t' *is a* U*-tuple whose restriction to* V *is the same as the* V*-tuple* t*; note also that the sum is finite since* R *has finite support)*

Positive Algebra (2)

selection *If* $R : U\text{-}\mathsf{Tup} \to K$ *and the selection predicate* $\mathbf{P}$ *maps each* U*-tuple to either* 0 *or* 1 *then* $\sigma_{\mathbf{P}} R : U\text{-}\mathsf{Tup} \to K$ *is defined by*

$$(\sigma_{\mathbf{P}} R)(t) \stackrel{\text{def}}{=} R(t) \cdot \mathbf{P}(t)$$

Which $\{0, 1\}$*-valued functions are used as selection predicates is left unspecified, except that we assume that* false*—the constantly* 0 *predicate, and* true*—the constantly* 1 *predicate, are always available.*

natural join *If* $R_i : U_i\text{-}\mathsf{Tup} \to K \quad i = 1, 2$ *then* $R_1 \bowtie R_2$ *is the* K*-relation over* $U_1 \cup U_2$ *defined by*

$$(R_1 \bowtie R_2)(t) \stackrel{\text{def}}{=} R_1(t_1) \cdot R_2(t_2)$$

where $t_1 = t$ on U_1 *and* $t_2 = t$ on U_2 *(recall that* t *is a* $U_1 \cup U_2$*-tuple).*

renaming *If* $R : U\text{-}\mathsf{Tup} \to K$ *and* $\beta : U \to U'$ *is a bijection then* $\rho_\beta R$ *is a* K*-relation over* U' *defined by*

$$(\rho_\beta R)(t) \stackrel{\text{def}}{=} R(t \circ \beta)$$

What is K?

PROPOSITION 3.4. *The following $\mathcal{RA}$ identities:*

- *union is associative, commutative and has identity* $\emptyset$*;*
- *join is associative, commutative and distributive over union;*
- *projections and selections commute with each other as well as with unions and joins (when applicable);*
- $\sigma_{\mathsf{false}}(R) = \emptyset$ *and* $\sigma_{\mathsf{true}}(R) = R$.

hold for the positive algebra on K*-relations if and only if* $(K, +, \cdot, 0, 1)$ *is a commutative semiring.*

What is K?

Proposition 3.4. *The following $\mathcal{RA}$ identities:*

- *union is associative, commutative and has identity $\emptyset$;*
- *join is associative, commutative and distributive over union;*
- *projections and selections commute with each other as well as with unions and joins (when applicable);*
- $\sigma_{\mathsf{false}}(R) = \emptyset$ *and* $\sigma_{\mathsf{true}}(R) = R$.

hold for the positive algebra on K-relations if and only if $(K, +, \cdot, 0, 1)$ is a commutative semiring.

Note that the list does not contain idempotence of union and self join, as these fail for set semantics

What is K?

Proposition 3.4. *The following $\mathcal{RA}$ identities:*

- *union is associative, commutative and has identity $\emptyset$;*
- *join is associative, commutative and distributive over union;*
- *projections and selections commute with each other as well as with unions and joins (when applicable);*
- $\sigma_{\mathsf{false}}(R) = \emptyset$ *and* $\sigma_{\mathsf{true}}(R) = R$.

hold for the positive algebra on K-relations if and only if $(K, +, \cdot, 0, 1)$ is a commutative semiring.

Def. A **commutative semiri**

- + is commutative, associati
- * is associative with identity
- * distributes over +
- $a * 0 = 0 * a = 0$

Note that the list does not contain idempotence of union and self join, as these fail for set semantics

What is K?

PROPOSITION 3.4. *The following $\mathcal{RA}$ identities:*

- *union is associative, commutative and has identity $\emptyset$;*
- *join is associative, commutative and distributive over union;*
- *projections and selections commute with each other as well as with unions and joins (when applicable);*
- $\sigma_{\mathsf{false}}(R) = \emptyset$ *and* $\sigma_{\mathsf{true}}(R) = R$.

hold for the positive algebra on K-relations if and only if $(K, +, \cdot, 0, 1)$ is a commutative semiring.

Def. A **commutative semiring** is a structure (K,+,*,0,1) where

- + is commutative, associative, with identity 0
- * is associative with identity 1
- * distributes over +
- $a * 0 = 0 * a = 0$

What is K?

Def. A **commutative semiring** is a structure (K,+,*,0,1) where

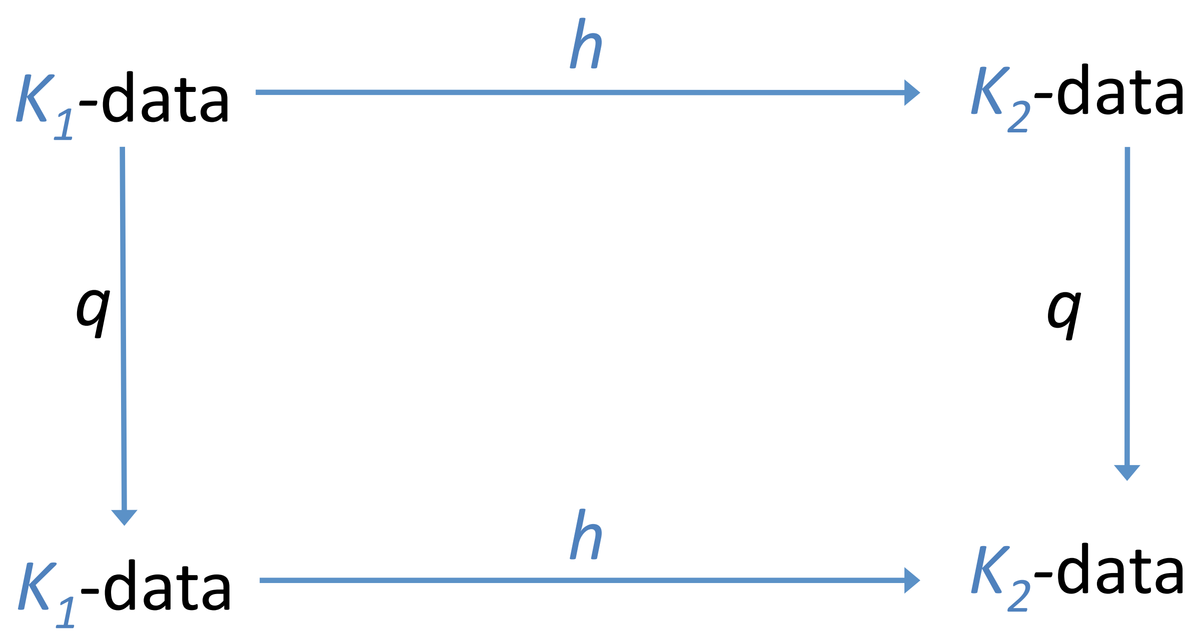
- + is commutative, associative, with identity 0
- * is associative with identity 1
- * distributes over +
- $a * 0 = 0 * a = 0$

Examples:

- the natural numbers: $(\mathbb{N}, +, *, 0, 1)$
- the booleans: $(\mathbb{B}, \wedge, \vee, \mathit{true}, \mathit{false})$
- subsets of a set: $(\mathscr{P}(\Omega), \cup, \cap, \emptyset, \Omega)$
- the naturals with infinity: $(\mathbb{N}^{\infty}, +, *, 0, 1)$
- polynomials in X: $(\mathbb{N}[X], +, *, 0, 1)$

The fundamental property of RA

For every query q and every homomorphism of commutative semirings $h : K_1 \rightarrow K_2$ the following "commutes":



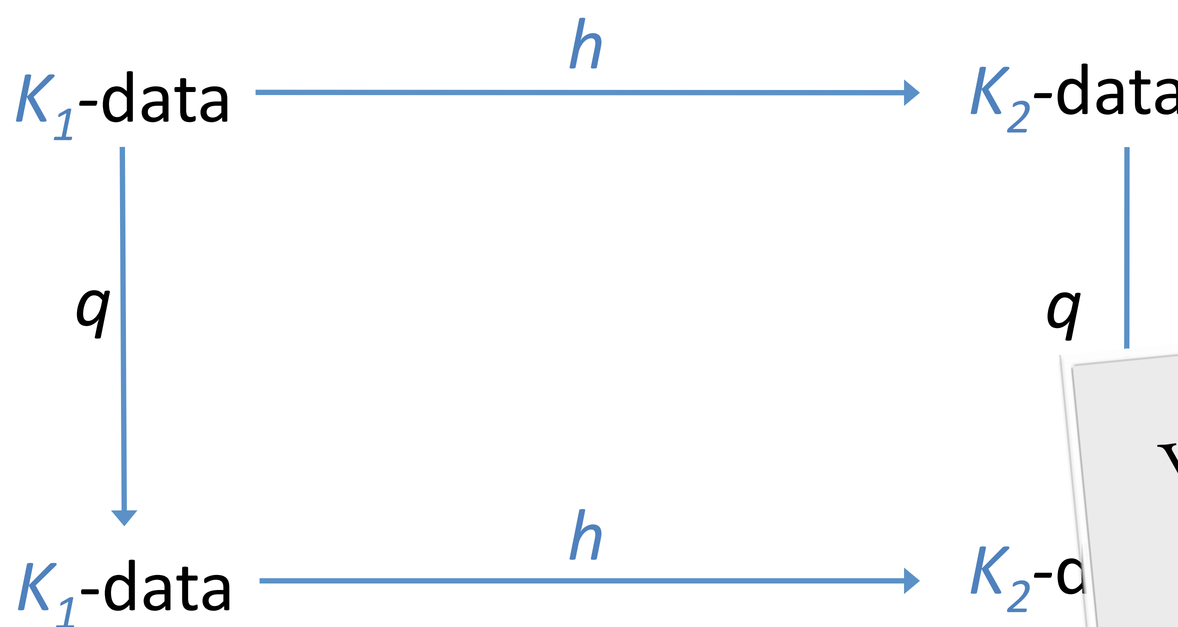
Recall, semiring homomorphism is mapping h:$K_1 \rightarrow K_2$ such that

$h(1_{K_1}) = 1_{K_2}$ $\quad$ $h(0_{K_1}) = 0_{K_2}$

$h(a +_{K_1} b) = h(a) +_{K_2} h(b)$ $\quad$ $h(a *_{K_1} b) = h(a) *_{K_2} h(b)$

The fundamental property of RA

For every query q and every homomorphism of commutative semirings $h : K_1 \rightarrow K_2$ the following "commutes":



Works only if q in RA⁺.
Does not generalize
e.g. to negation.

Recall, semiring homomorphism is mapping $h{:}K_1 \rightarrow K_2$ such that

$h(1_{K_1}) = 1_{K_2}$ $\quad$ $h(0_{K_1}) = 0_{K_2}$

$h(a +_{K_1} b) = h(a) +_{K_2} h(b)$ $\quad$ $h(a *_{K_1} b) = h(a) *_{K_2} h(b)$

Which semiring do we choose?

Definition 4.1. *Let X be the set of tuple ids of a (usual) database instance I. The* **positive algebra provenance semiring** *for I is the semiring of polynomials with variables (a.k.a. indeterminates) from X and coefficients from $\mathbb{N}$, with the operations defined as usual*[4]*:* $(\mathbb{N}[X], +, \cdot, 0, 1)$.

Which semiring do we choose?

DEFINITION 4.1. *Let X be the set of tuple ids of a (usual) database instance I. The* **positive algebra provenance semiring** *for I is the semiring of polynomials with variables (a.k.a. indeterminates) from X and coefficients from $\mathbb{N}$, with the operations defined as usual*[4]*:* $(\mathbb{N}[X], +, \cdot, 0, 1)$.

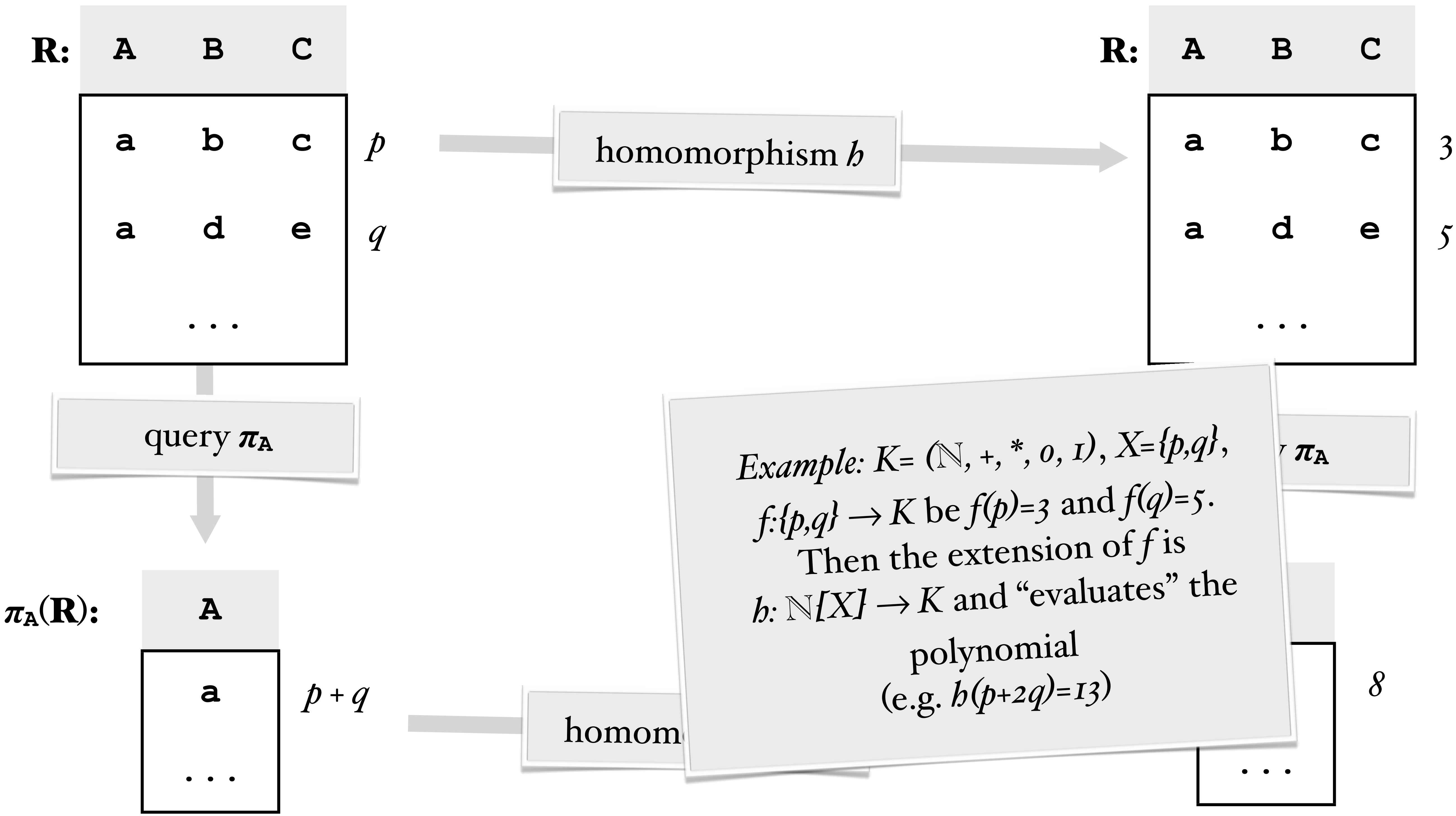
But why?

A nice property of $\mathbb{N}[X]$

If K is a commutative semiring, then any function on tokens, $f : X \to K$ extends uniquely to a homomorphism $h : \mathbb{N}[X] \to K$.

Example: $K = (\mathbb{N}, +, *, 0, 1)$, $X = \{p,q\}$, $f:\{p,q\} \to K$ be $f(p)=3$ and $f(q)=5$. Then the extension of f is $h: \mathbb{N}[X] \to K$ and "evaluates" the polynomial (e.g. $h(p+2q)=13$)

Nice + Fundamental



Nice + Fundamental

R:

A	B	C	
a	b	c	p
a	d	e	q
	. . .		

homomorphism h →

R:

A	B	C	
a	b	c	*3*
a	d	e	*5*
	. . .		

Example: $K = (\mathbb{N}, +, *, 0, 1)$, $X = \{p,q\}$,
$f: \{p,q\} \rightarrow K$ be $f(p)=3$ and $f(q)=5$.
Then the extension of f is
$h: \mathbb{N}[X] \rightarrow K$ and "evaluates" the polynomial
(e.g. $h(p+2q)=13$)

query π_A ↓

π_A(**R**):

A	
a	$p + q$
. . .	

homomorphism h →

π_A(**R**):

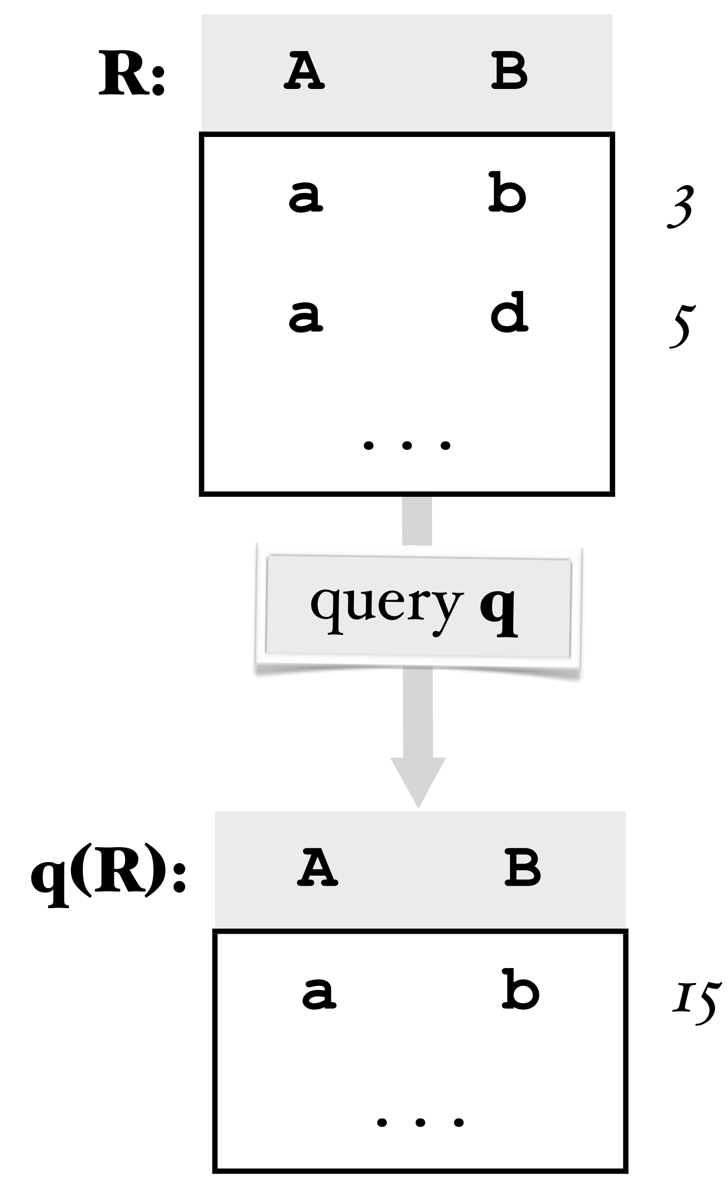
A	
a	*8*
. . .	

Free the semiring!

"Nice" implies: For every commutative semiring K, and every K-relation **R**, there is abstractly tagged N[X]-relation $\overline{\mathbf{R}}$ and a homomorphism $Eval_v$ from $\overline{\mathbf{R}}$ to **R**.

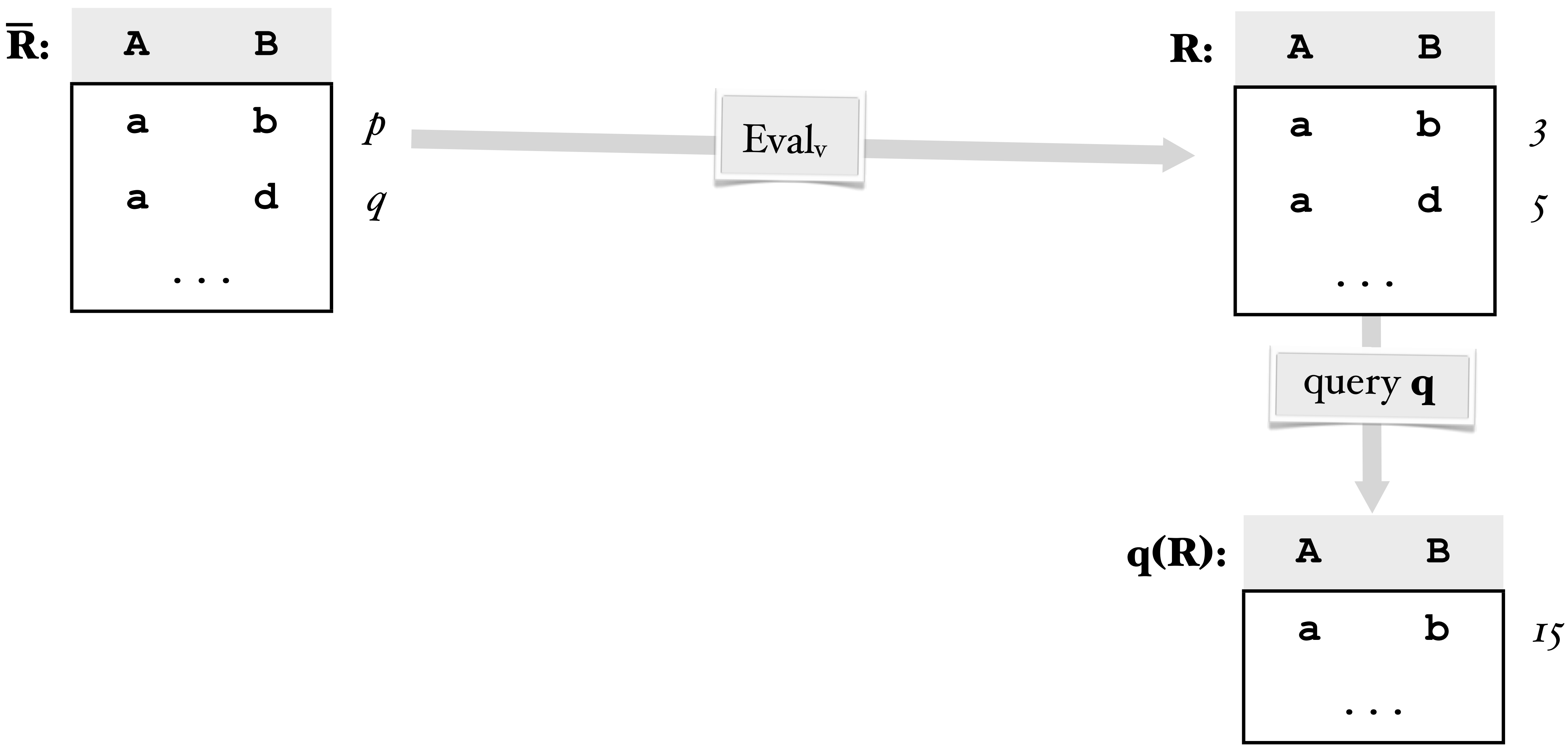
Free the semiring!

"Nice" implies: For every commutative semiring K, and every K-relation **R**, there is abstractly tagged N[X]-relation $\bar{\mathbf{R}}$ and a homomorphism $Eval_v$ from $\bar{\mathbf{R}}$ to **R**.



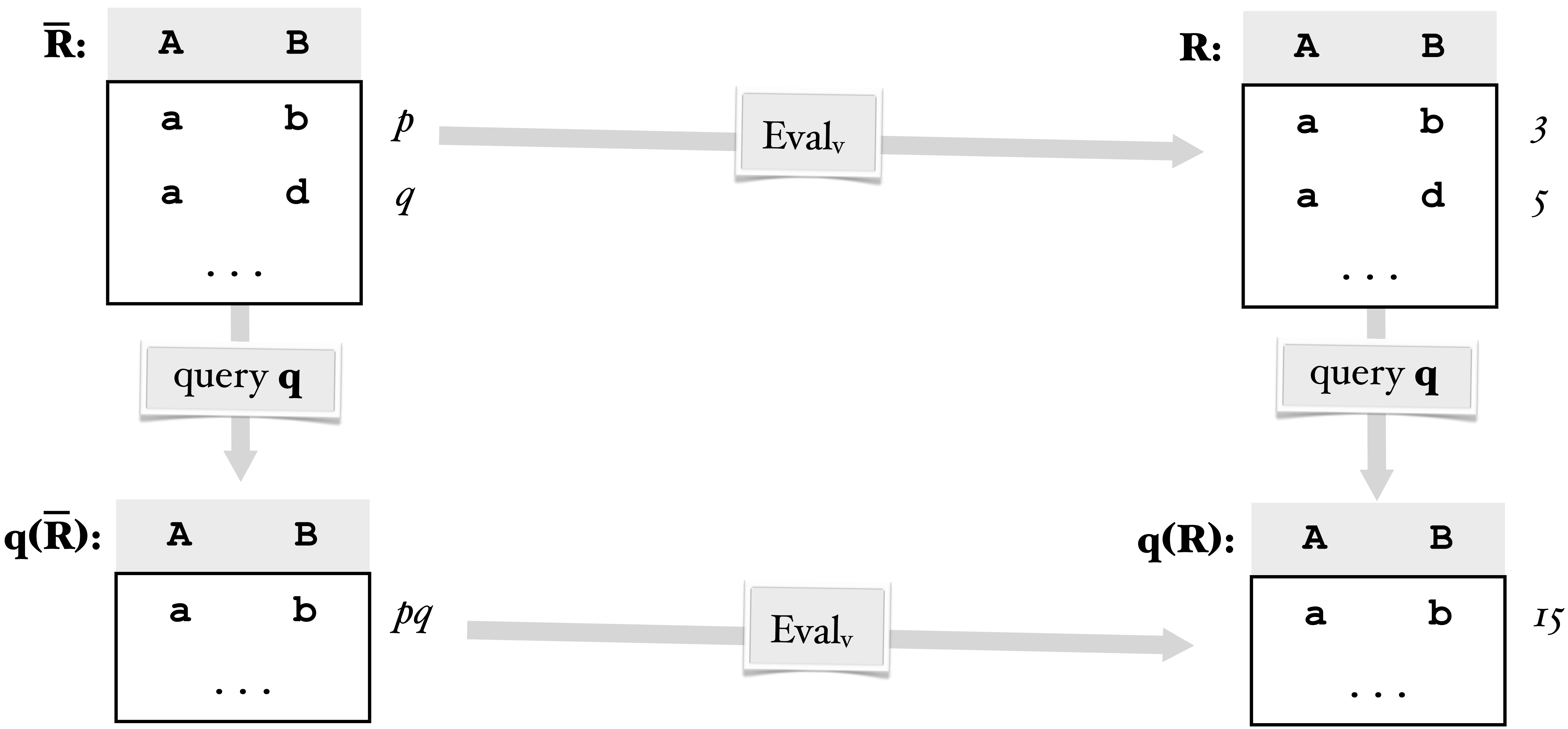
Free the semiring!

"Nice" implies: For every commutative semiring K, and every K-relation **R**, there is abstractly tagged N[X]-relation $\overline{\mathbf{R}}$ and a homomorphism Eval_v from $\overline{\mathbf{R}}$ to **R**.



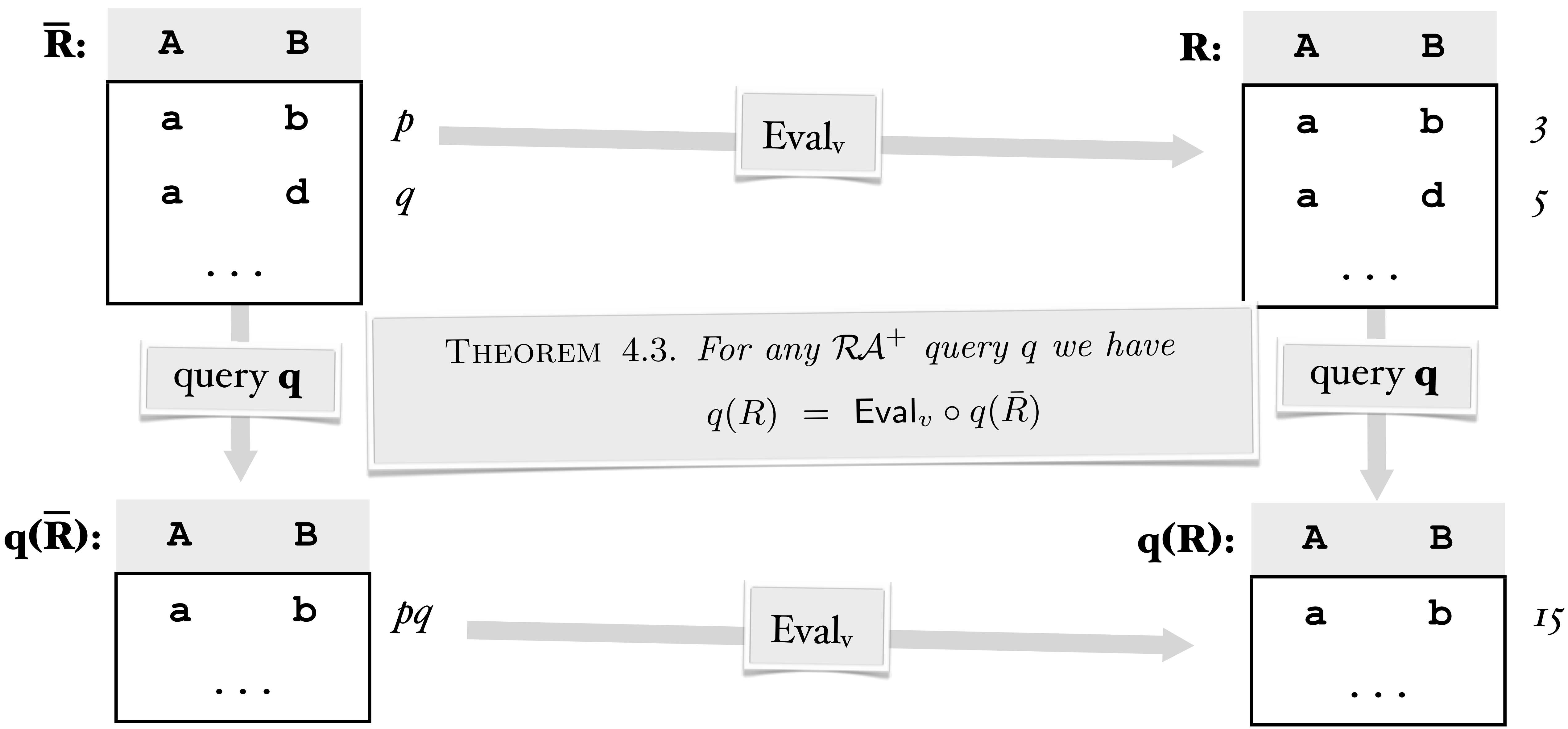
Free the semiring!

“Nice” implies: For every commutative semiring K, and every K-relation **R**, there is abstractly tagged N[X]-relation $\bar{\mathbf{R}}$ and a homomorphism Eval_v from $\bar{\mathbf{R}}$ to **R**.



Free the semiring!

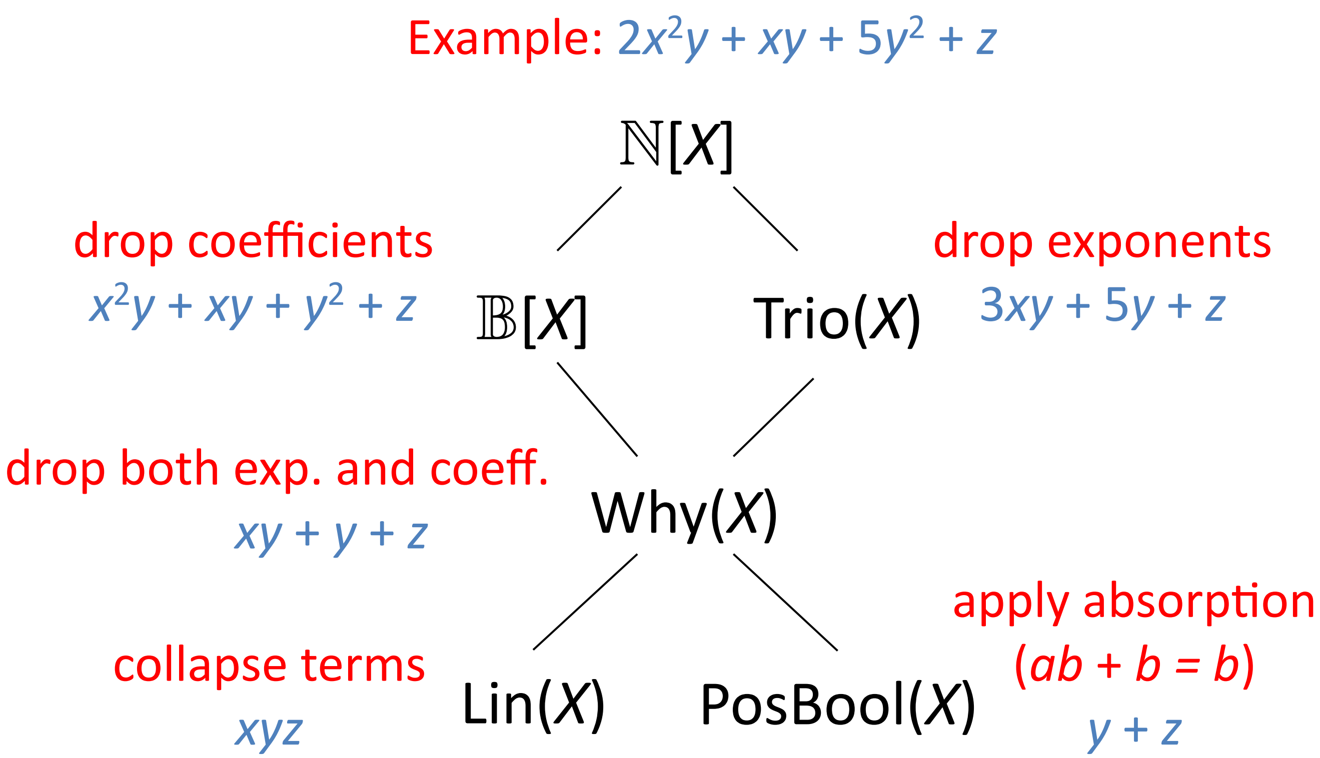
"Nice" implies: For every commutative semiring K, and every K-relation **R**, there is abstractly tagged N[X]-relation $\bar{\mathbf{R}}$ and a homomorphism Eval_v from $\bar{\mathbf{R}}$ to **R**.



Instantiation of Positive Algebra

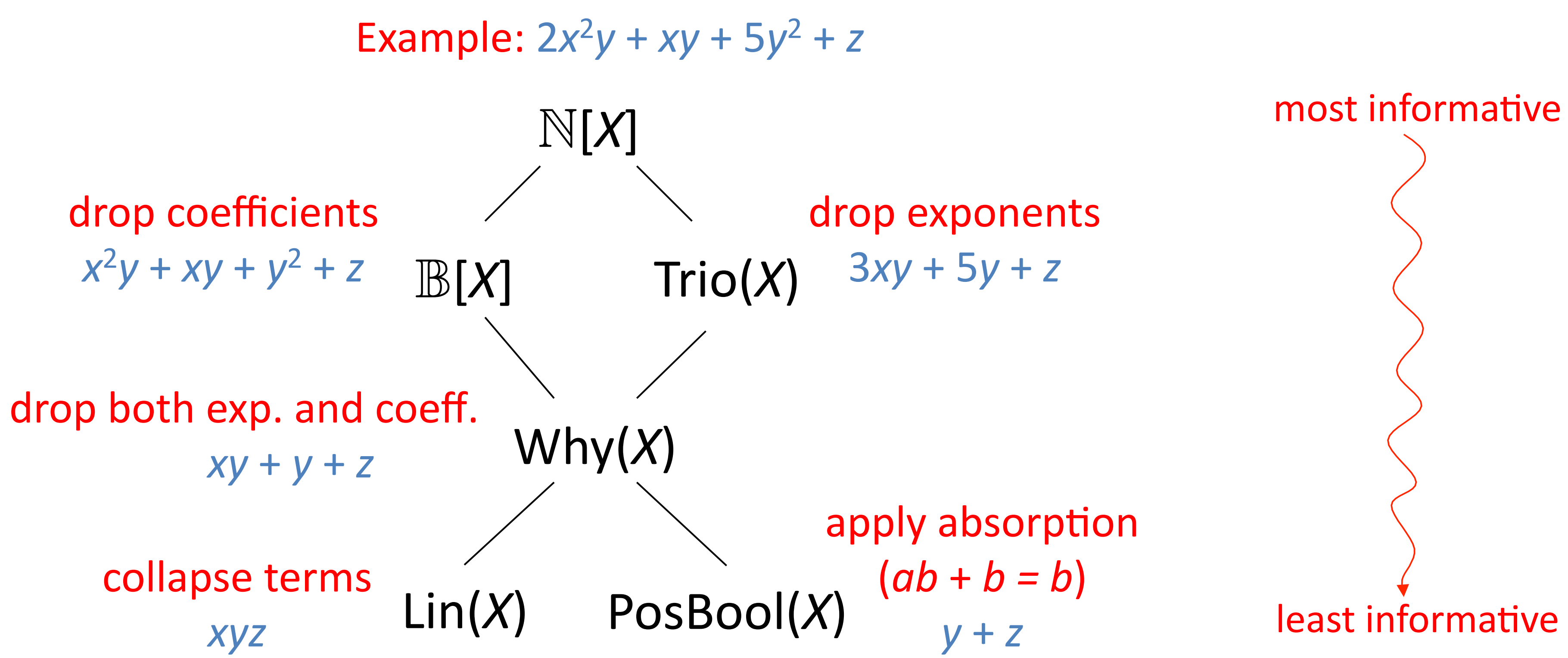
$(\mathbb{B}, \wedge, \vee, \text{true}, \text{false})$	Set semantics
$(\mathbb{N}, +, *, 0, 1)$	Bag semantics
$(\mathscr{P}(\Omega), \cup, \cap, \emptyset, \Omega)$	Probabilistic events
$(\text{BoolExp}(P), \vee, \wedge, \text{true}, \text{false})$	Conditional tables
$(A, \min, \max, 0, P)$ where $A = \mathbb{P} < \mathbb{C} < \mathbb{S} < \mathbb{T} < \mathbb{0}$	Access control levels

More nice…



A path downward from K_1 to K_2 indicates that there exists an **onto (surjective) semiring homomorphism** $h : K_1 \rightarrow K_2$

More nice...



A path downward from K_1 to K_2 indicates that there exists an **onto (surjective) semiring homomorphism** $h : K_1 \rightarrow K_2$

Datalog

Syntax and Semantics

$$Q(x, y) \text{ :- } R(x, z), R(z, y)$$

edb

R:

a	a
a	b
b	b

Q:

a	a
a	b
b	b

idb

Syntax and Semantics

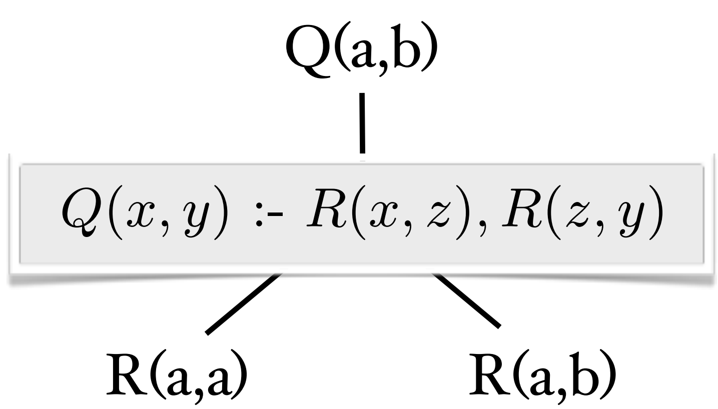
$$Q(x, y) \text{ :- } R(x, z), R(z, y)$$

edb → R:

a	a
a	b
b	b

Q: ← idb

a	a
a	b
b	b



Syntax and Semantics

$$Q(x,y) \text{ :- } R(x,z), R(z,y)$$

edb

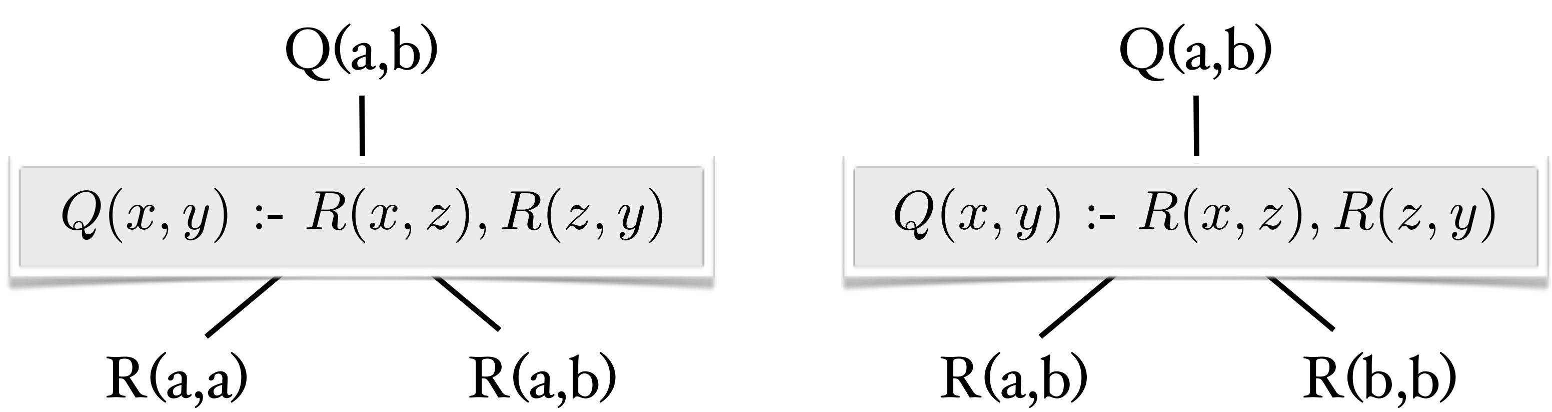
R:

a	a
a	b
b	b

Q:

a	a
a	b
b	b

idb



Syntax and Semantics

$$Q(x, y) \text{ :- } R(x, z), R(z, y)$$

edb

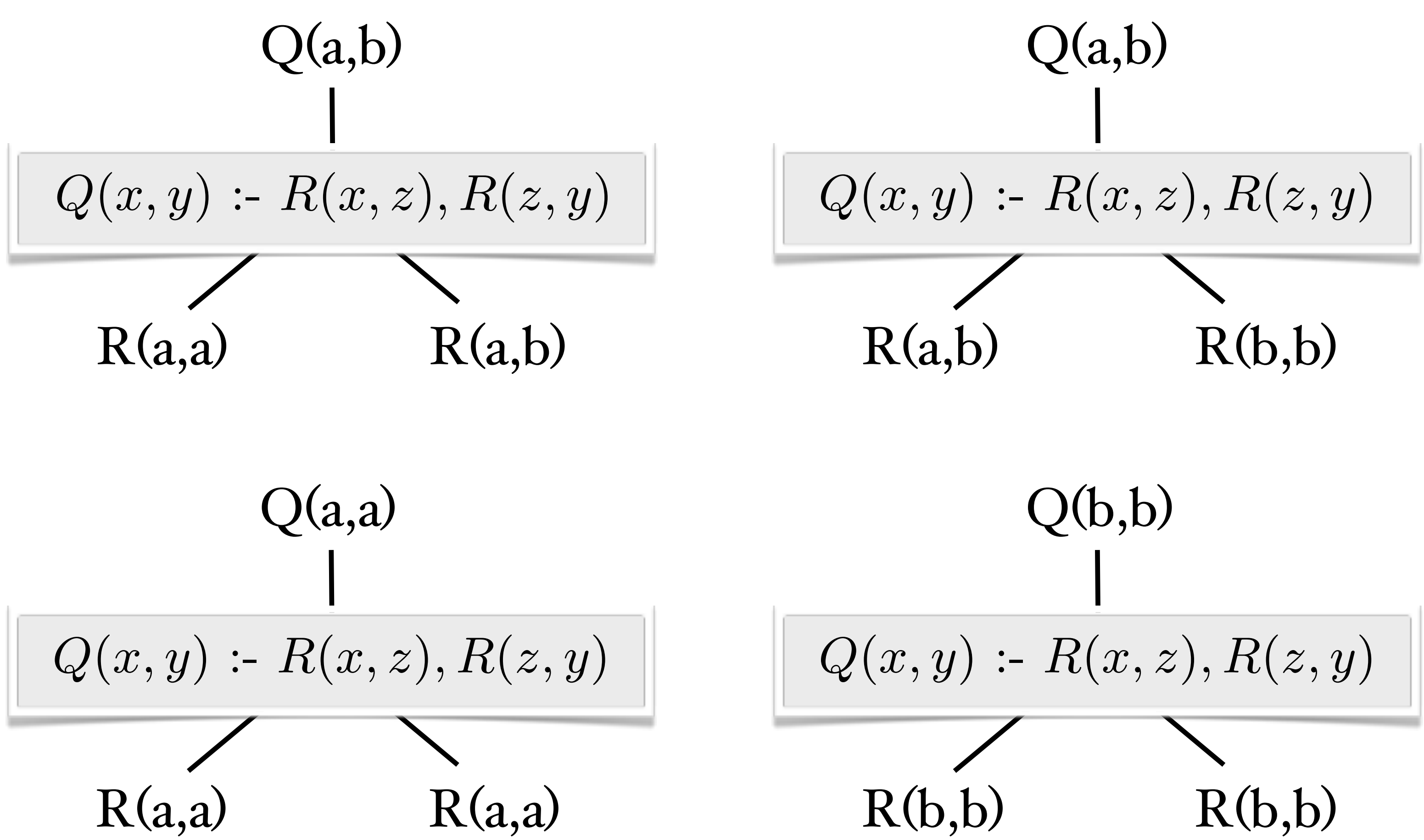
R:

a	a
a	b
b	b

Q:

a	a
a	b
b	b

idb



Datalog with Bag Semantics

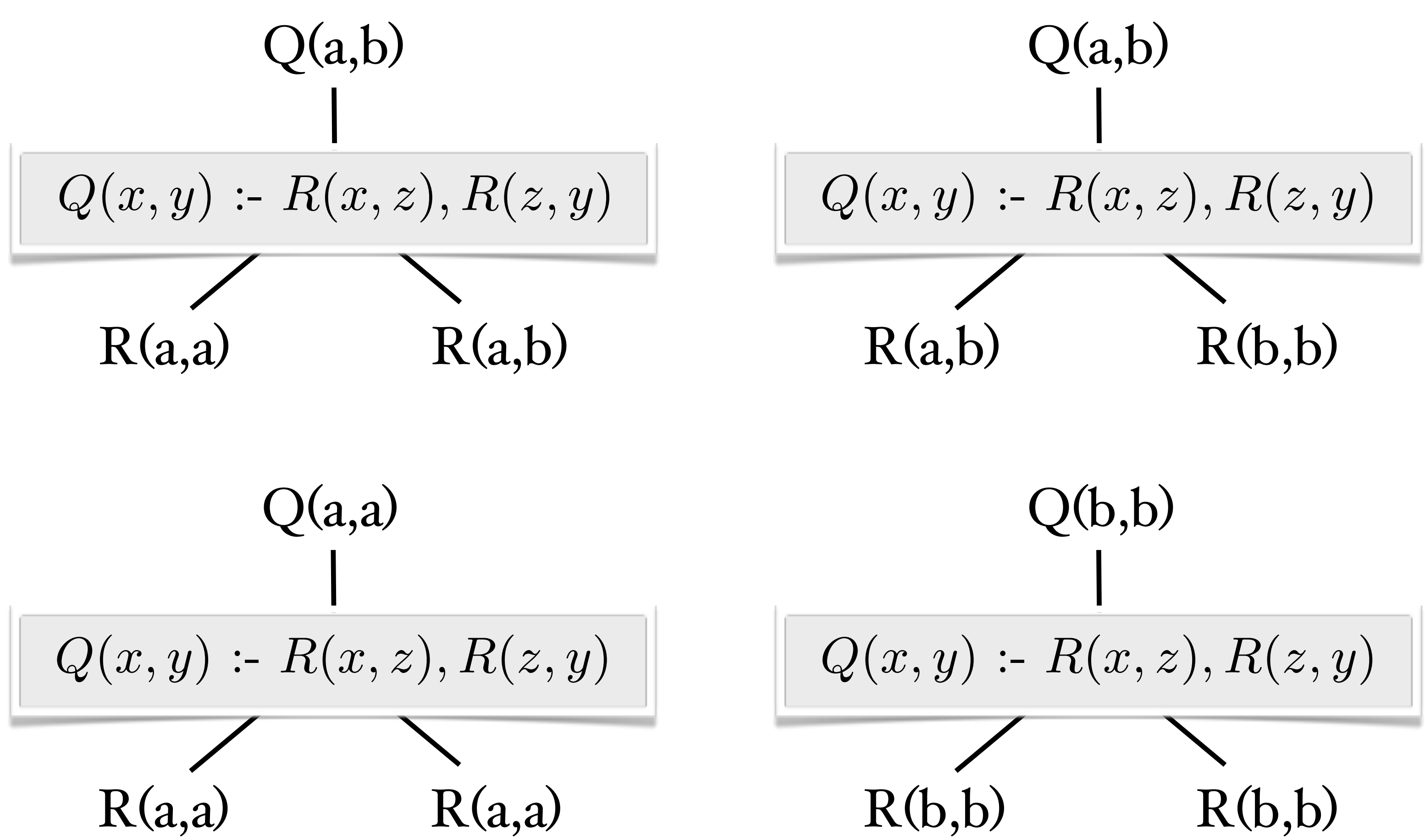
$$Q(x, y) \text{ :- } R(x, z), R(z, y)$$

R:

a	a
a	b
b	b

Q:

a	a
a	b
b	b



Datalog with Bag Semantics

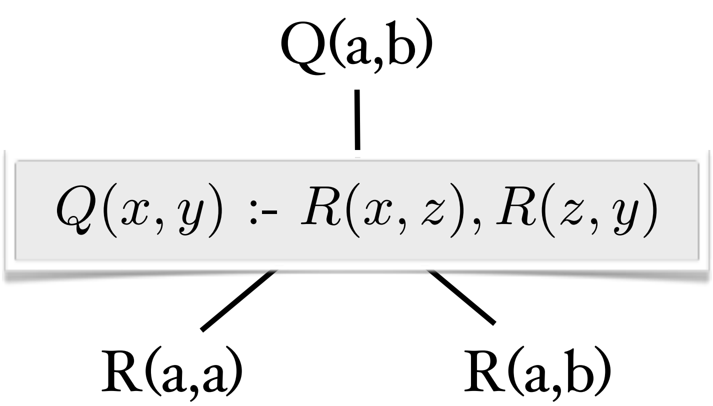
$$Q(x,y) \text{ :- } R(x,z), R(z,y)$$

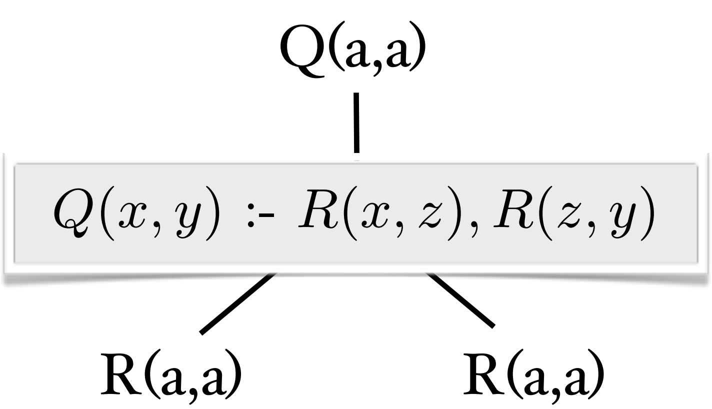
R:

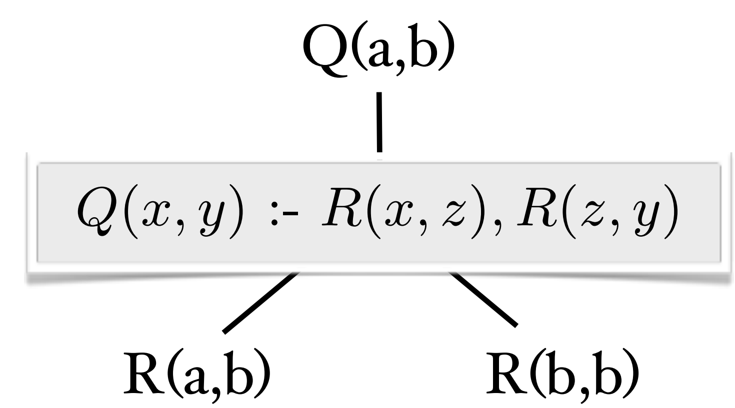
$a\ a$	2
$a\ b$	3
$b\ b$	4

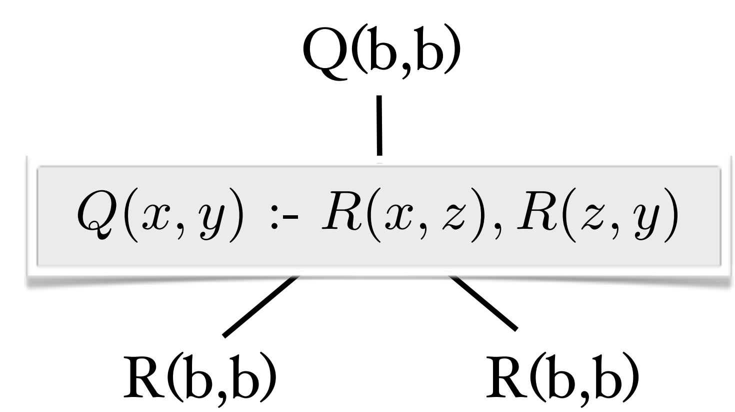
Q:

$a\ a$
$a\ b$
$b\ b$









Datalog with Bag Semantics

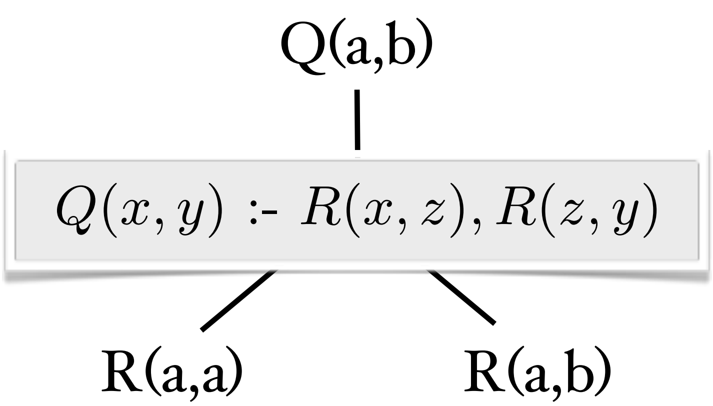
$$Q(x,y) \text{ :- } R(x,z), R(z,y)$$

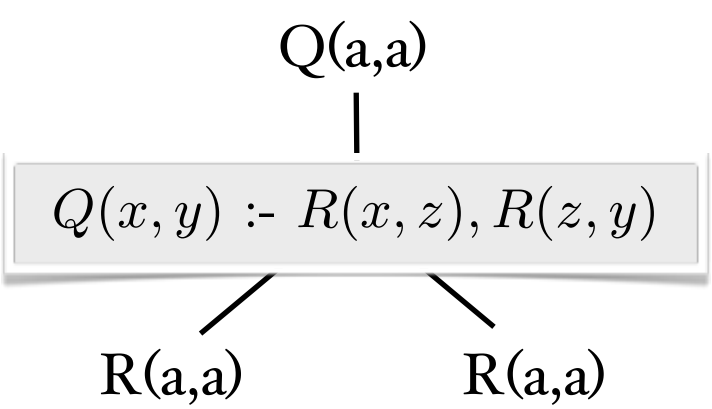
R:

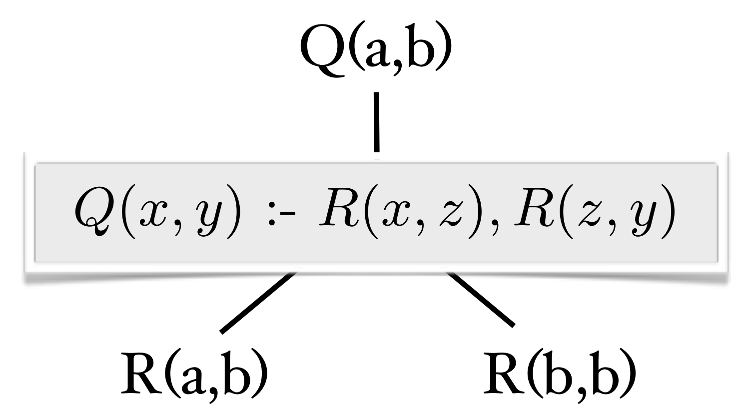
$a\ a$	2
$a\ b$	3
$b\ b$	4

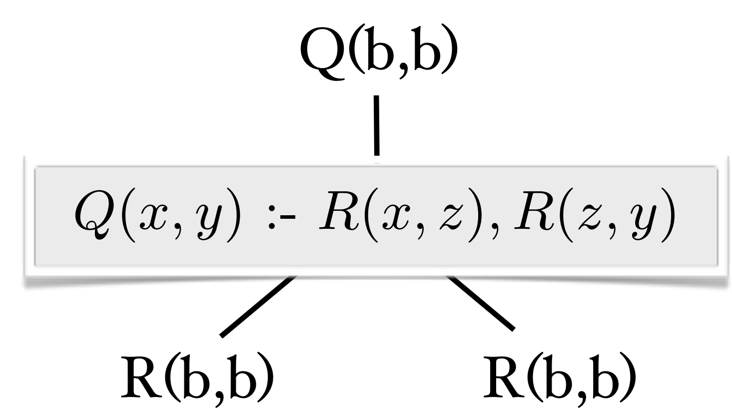
Q:

$a\ a$	$2 \cdot 2 = 4$
$a\ b$	$2 \cdot 3 + 3 \cdot 4 = 18$
$b\ b$	$4 \cdot 4 = 16$









What annotations do we need?

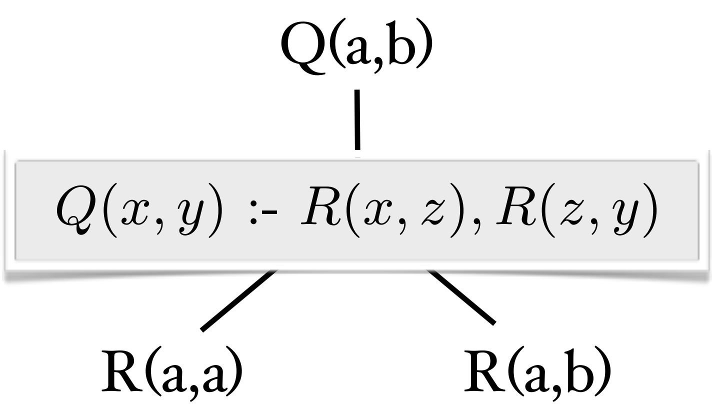
$$Q(x, y) \text{ :- } R(x, z), R(z, y)$$

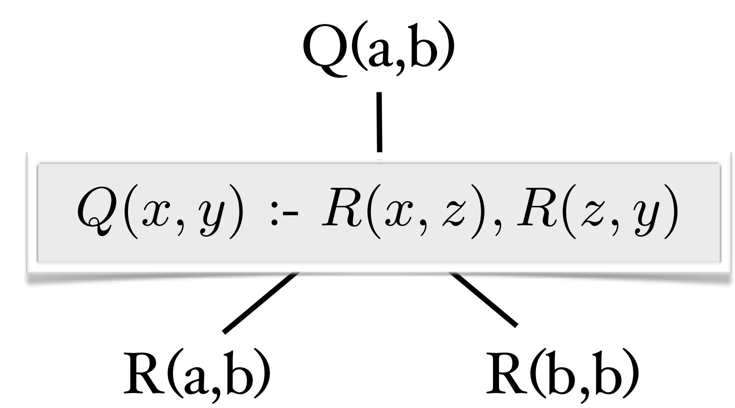
R:

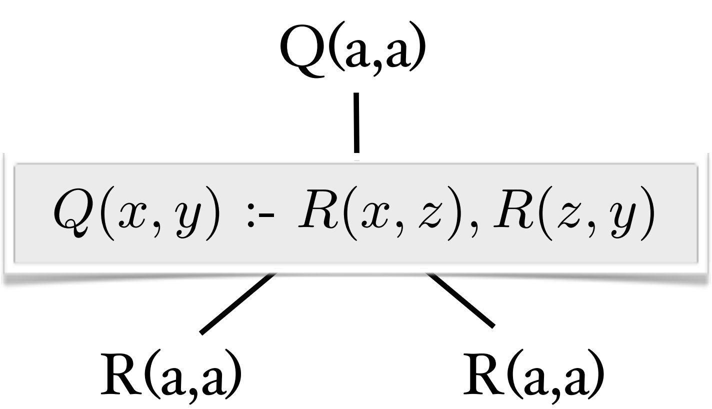
$a\ a$	2
$a\ b$	3
$b\ b$	4

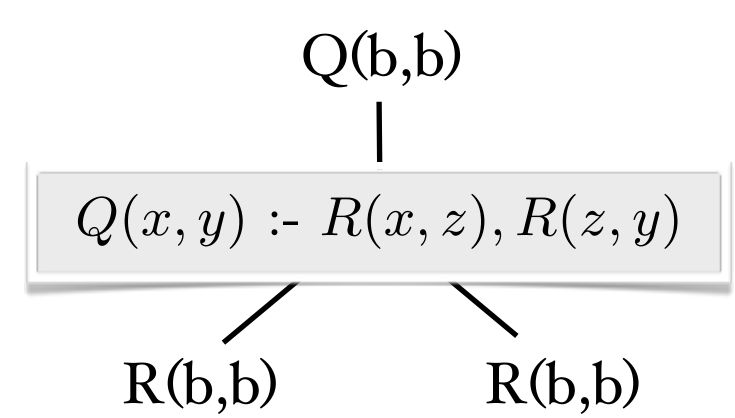
Q:

$a\ a$	$2 \cdot 2 = 4$
$a\ b$	$2 \cdot 3 + 3 \cdot 4 = 18$
$b\ b$	$4 \cdot 4 = 16$









What annotations do we need?

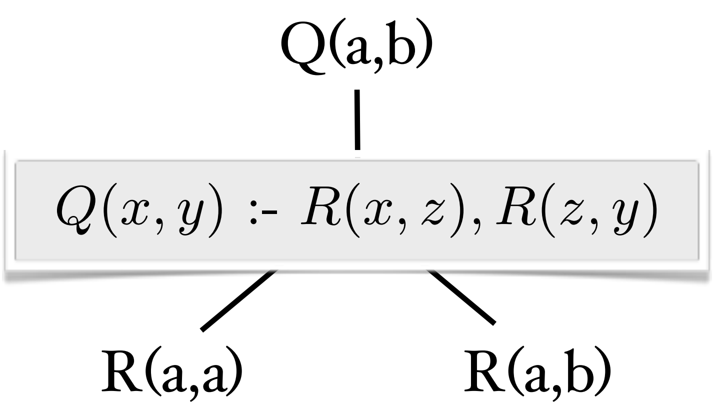
$$Q(x,y) \text{ :- } R(x,z), R(z,y)$$

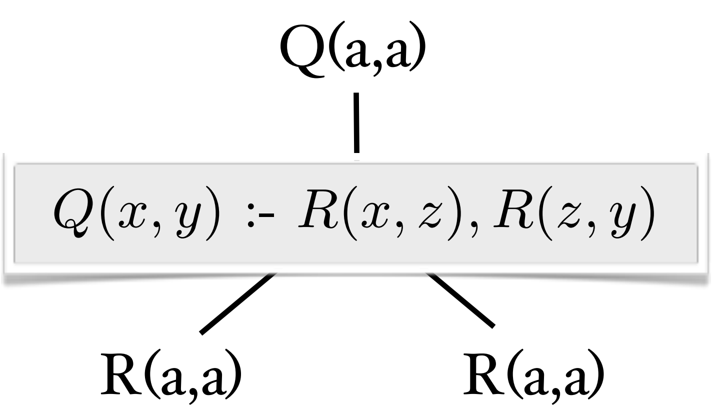
R:

$a\ a$	2
$a\ b$	3
$b\ b$	4

Q:

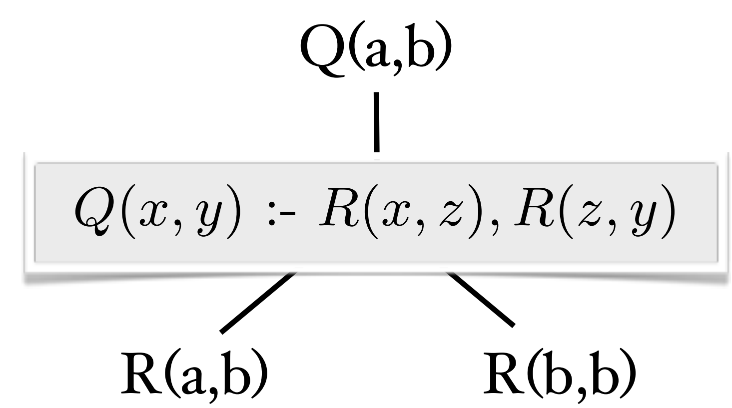
$a\ a$	$2 \cdot 2 = 4$
$a\ b$	$2 \cdot 3 + 3 \cdot 4 = 18$
$b\ b$	$4 \cdot 4 = 16$

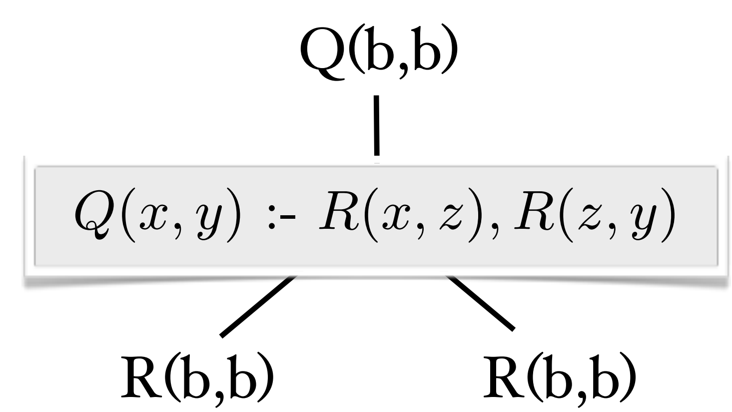




How about: the tag of an answer tuple is the sum over all derivation trees and the product of the tags of each leaf.

$$q(R)(t) = \sum_{\tau \ yields \ t} \left(\prod_{t' \in leaves(\tau)} R(t') \right)$$





What annotations do we need?

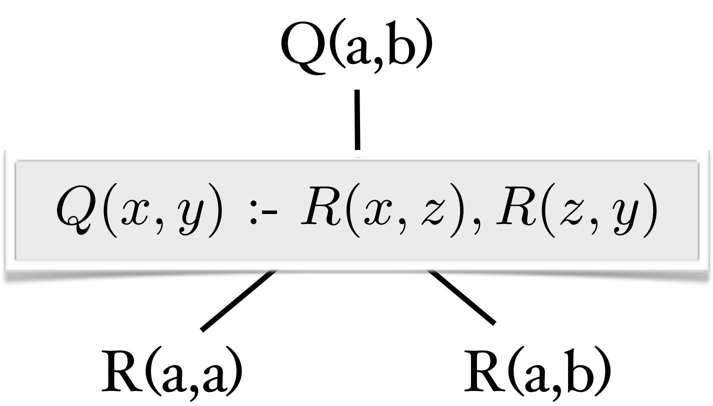
$$Q(x,y) \text{ :- } R(x,z), R(z,y)$$

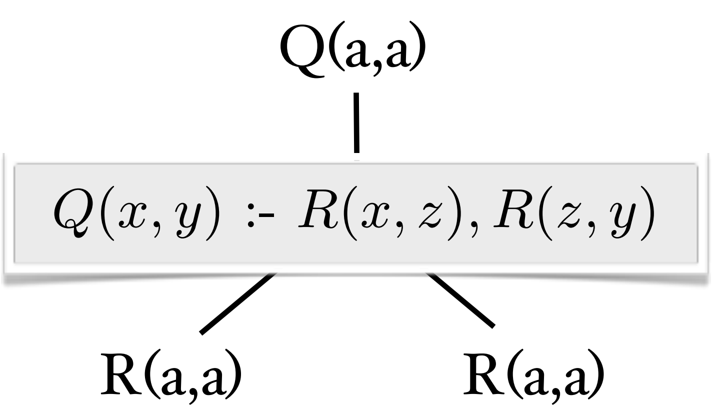
R:

$a\ a$	2
$a\ b$	3
$b\ b$	4

Q:

$a\ a$	$2 \cdot 2 = 4$
$a\ b$	$2 \cdot 3 + 3 \cdot 4 = 18$
$b\ b$	$4 \cdot 4 = 16$

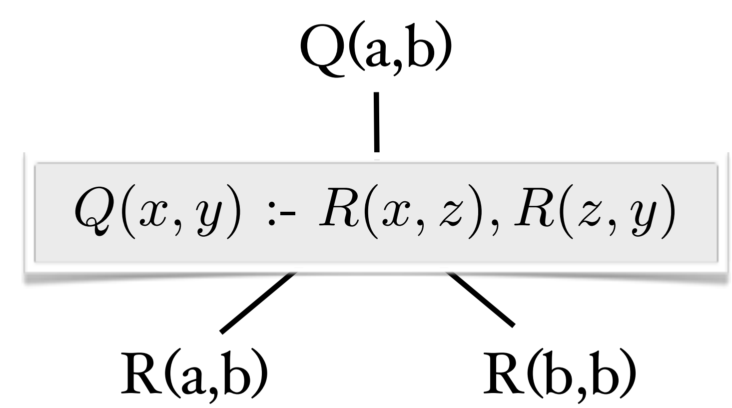


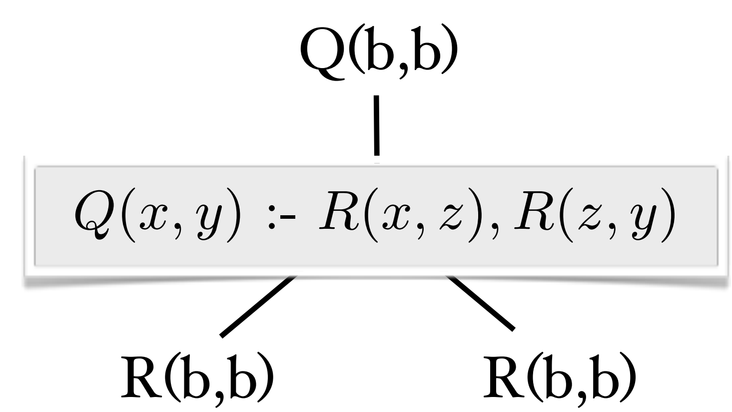


How about: the tag of an answer tuple is the sum over all derivation trees and the product of the tags of each leaf.

$$q(R)(t) = \sum_{\tau \ yields \ t} \left(\prod_{t' \in leaves(\tau)} R(t') \right)$$

Problem: A tuple may have infinitely many derivation trees. Hence we need to work in semirings in which infinite sums are defined.





ω-continuos semirings

Def. ***(Natural preorder)*** $x \leq y$ iff there is z such that $x+z=y$.

Def. ***(Naturally ordered semiring)*** if the natural pre-order is an order.

Def. ***(ω-complete)*** when $x_1 \leq x_2 \leq x_2 \leq \ldots$ have suprema.

In naturally ordered semirings, we can make sense of infinite sums:

$$\sum_{n \in \mathbb{N}} a_n \stackrel{\text{def}}{=} \sup_{m \in \mathbb{N}} (\sum_{i=0}^{m} a_i)$$

Def. ***(ω-continous)*** when * and + preserve suprema.
(e.g. $\sup(a_i + b_i) = \sup(a_i) + b_i$).

Lemma. Over ω-continuos semirings, functions defined by polynomials have least fixed points.

ω-continuos semirings

Def. ***(Natural preorder)*** $x \leq y$ iff there is z such that $x+z=y$.

Def. ***(Naturally ordered semiring)***
an order.

Preorder: reflexive and transitive.
Not necessarily anti-symmetric
($x \leq y$ and $y \leq x$ implies $x=y$)

Def. ***(ω-complete)*** when $x_1 \leq x_2 \leq x_2 \leq \ldots$

In naturally ordered semirings, we can make sense of infinite sums:

$$\sum_{n \in \mathbb{N}} a_n \stackrel{\text{def}}{=} \sup_{m \in \mathbb{N}} \left(\sum_{i=0}^{m} a_i \right)$$

Def. ***(ω-continous)*** when * and + preserve suprema.
(e.g. $\sup(a_i + b_i) = \sup(a_i) + b_i$).

Lemma. Over ω-continuos semirings, functions defined by polynomials have least fixed points.

ω-continuos semirings

Def. ***(Natural preorder)*** $x \leq y$ iff there is z such that $x+z=y$.

Def. ***(Naturally ordered semiring)*** if the natural pre-order is an order.

E.g. $\mathbb{Z}$ is not naturally ordered because $-5 \leq 5$, $5 \leq -5$, but $-5 \neq 5$

when $x_1 \leq x_2 \leq x_2 \leq \ldots$ have suprema.

d semirings, we can make sense of infinite sums:

$$\sum_{n \in \mathbb{N}} a_n \stackrel{\text{def}}{=} \sup_{m \in \mathbb{N}} (\sum_{i=0}^{m} a_i)$$

Def. ***(ω-continous)*** when * and + preserve suprema.
(e.g. $\sup(a_i + b_i) = \sup(a_i) + b_i$).

Lemma. Over ω-continuos semirings, functions defined by polynomials have least fixed points.

ω-continuos semirings

Def. ***(Natural preorder)*** $x \leq y$ iff there is z such that $x+z=y$.

Def. ***(Naturally ordered semiring)*** if the natural pre-order is an order.

Def. ***(ω-complete)*** when $x_1 \leq x_2 \leq x_2 \leq \ldots$ have suprema.

In naturally ordered semirings, we can make sense of infinite sums:

$$\sum_{n\in\mathbb{N}} a_n \stackrel{\text{def}}{=} \sup_{m\in\mathbb{N}} (\sum_{i=0}^{m} a_i)$$

Def. ***(ω-continous)*** when * and + preserve suprema.
(e.g. $\sup(a_i + b_i) = \sup(a_i) + b_i$).

Lemma. Over ω-continuos semirings, functions defined by polynomials have least fixed points.

Semantics of annotated Datalog

Definition 5.1. *Let* $(K, +, \cdot, 0, 1)$ *be a commutative* ω*-continuous semiring. To keep notation simple let* q *be a datalog query with one argument (it is easy to generalize to multiple arguments). For any* K*-relation* R *define*

$$q(R)(t) = \sum_{\tau \text{ yields } t} \Big(\prod_{t' \in leaves(\tau)} R(t') \Big)$$

where τ *ranges over all* q*-derivation trees for* t *and* t' *ranges over all the leaves of* τ*.*

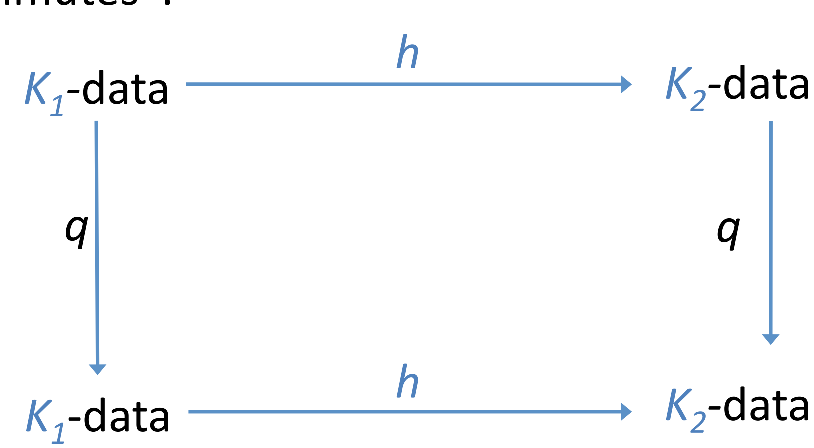
Semantics of annotated Datalog

DEFINITION 5.1. *Let* $(K, +, \cdot, 0, 1)$ *be a commutative* ω*-continuous semiring. To keep notation simple let* q *be a datalog query with one argument (it is easy to generalize to multiple arguments). For any* K*-relation* R *define*

$$q(R)(t) = \sum_{\tau \text{ yields } t} \left(\prod_{t' \in leaves(\tau)} R(t') \right)$$

where τ *ranges over all* q*-derivation trees for* t *and* t' *ranges over all the leaves of* τ*.*

For every query q and every homomorphism of commutative semirings $h : K_1 \to K_2$ the following "commutes":



Semantics of annotated Datalog

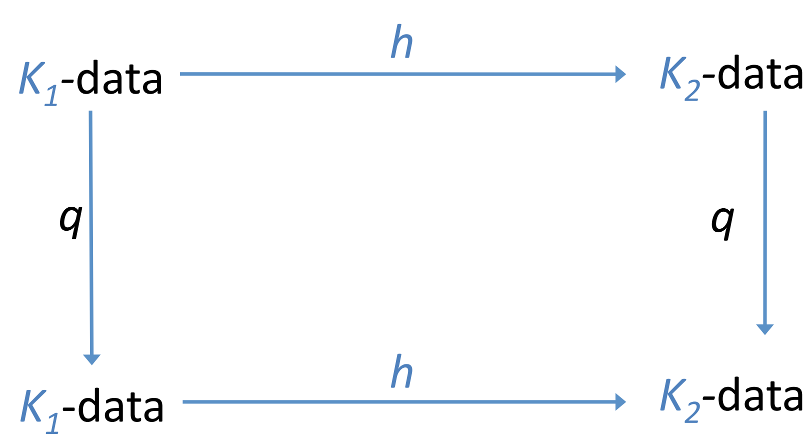
DEFINITION 5.1. *Let* $(K, +, \cdot, 0, 1)$ *be a commutative* ω*-continuous semiring. To keep notation simple let* q *be a datalog query with one argument (it is easy to generalize to multiple arguments). For any* K*-relation* R *define*

$$q(R)(t) = \sum_{\tau \ yields \ t} \left(\prod_{t' \in leaves(\tau)} R(t') \right)$$

where τ *ranges over all* q*-derivation trees for* t *and* t' *ranges over all the leaves of* τ.

Datalog ω-continuos

For every query q and every homomorphism of commutative semirings $h : K_1 \to K_2$ the following "commutes":



The Datalog provenance Semiring

Problem: there can be infinitely many derivation trees for one tuple

☞ infinite sums in annotations

In particular two kinds of infinite summations

- infinitely many copies of the same monomial → coefficients in $\mathbb{N}^{\infty} = \mathbb{N} \cup \{\infty\}$
- infinitely many copies of different monomials → formal power series K[[X]]

The Datalog provenance Semiring

Problem: there can be infinitely many derivation trees for one tuple

☞ infinite sums in annotations

In particular two kinds of infinite summations

- infinitely many copies of the same monomial → coefficients in $\mathbb{N}^\infty = \mathbb{N} \cup \{\infty\}$
- infinitely many copies of different monomials → formal power series K[[X]]

Formal power series:
basically polynomials with
infinite summation

The Datalog provenance Semiring

Problem: there can be infinitely many derivation trees for one tuple

☞ infinite sums in annotations

In particular two kinds of infinite summations

- infinitely many copies of the same monomial → coefficients in $\mathbb{N}^\infty = \mathbb{N} \cup \{\infty\}$
- infinitely many copies of different monomials → formal power series K[[X]]

Definition 6.1. *Let X be the set of tuple ids of a database instance I. The* **datalog provenance semiring** *for I is the commutative ω-continuous semiring of formal power series $\mathbb{N}^\infty[[X]]$.*

Fixed Point Semantics

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

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

(a)

R:

$a\ b$	2
$a\ c$	3
$c\ b$	2
$b\ d$	1
$d\ d$	1

(b)

Q(R):

$a\ b$	8
$a\ c$	3
$c\ b$	2
$b\ d$	∞
$d\ d$	∞
$a\ d$	∞

(c)

$\bar{R}$:

$a\ b$	m
$a\ c$	n
$c\ b$	p
$b\ d$	r
$d\ d$	s

(d)

Q($\bar{R}$):

$a\ b$	$\mathbf{x}$
$a\ c$	$\mathbf{y}$
$c\ b$	$\mathbf{z}$
$b\ d$	$\mathbf{u}$
$d\ d$	$\mathbf{v}$
$a\ d$	$\mathbf{w}$

(e)

$$\begin{aligned}
\mathbf{x} &= m + \mathbf{yz} \\
\mathbf{y} &= n \\
\mathbf{z} &= p \\
\mathbf{u} &= r + \mathbf{uv} \\
\mathbf{v} &= s + \mathbf{v}^2 \\
\mathbf{w} &= \mathbf{xu} + \mathbf{wv}
\end{aligned}$$

(f)

- Transform immediate consequence operator of Q into a union of conjunctive queries; here R ∪ (Q $\bowtie_{2=1}$ Q)
- Apply this RA query to $\bar{R}$ and $\bar{Q}$.
- Equate!

This leads to system of equations of polynomials in

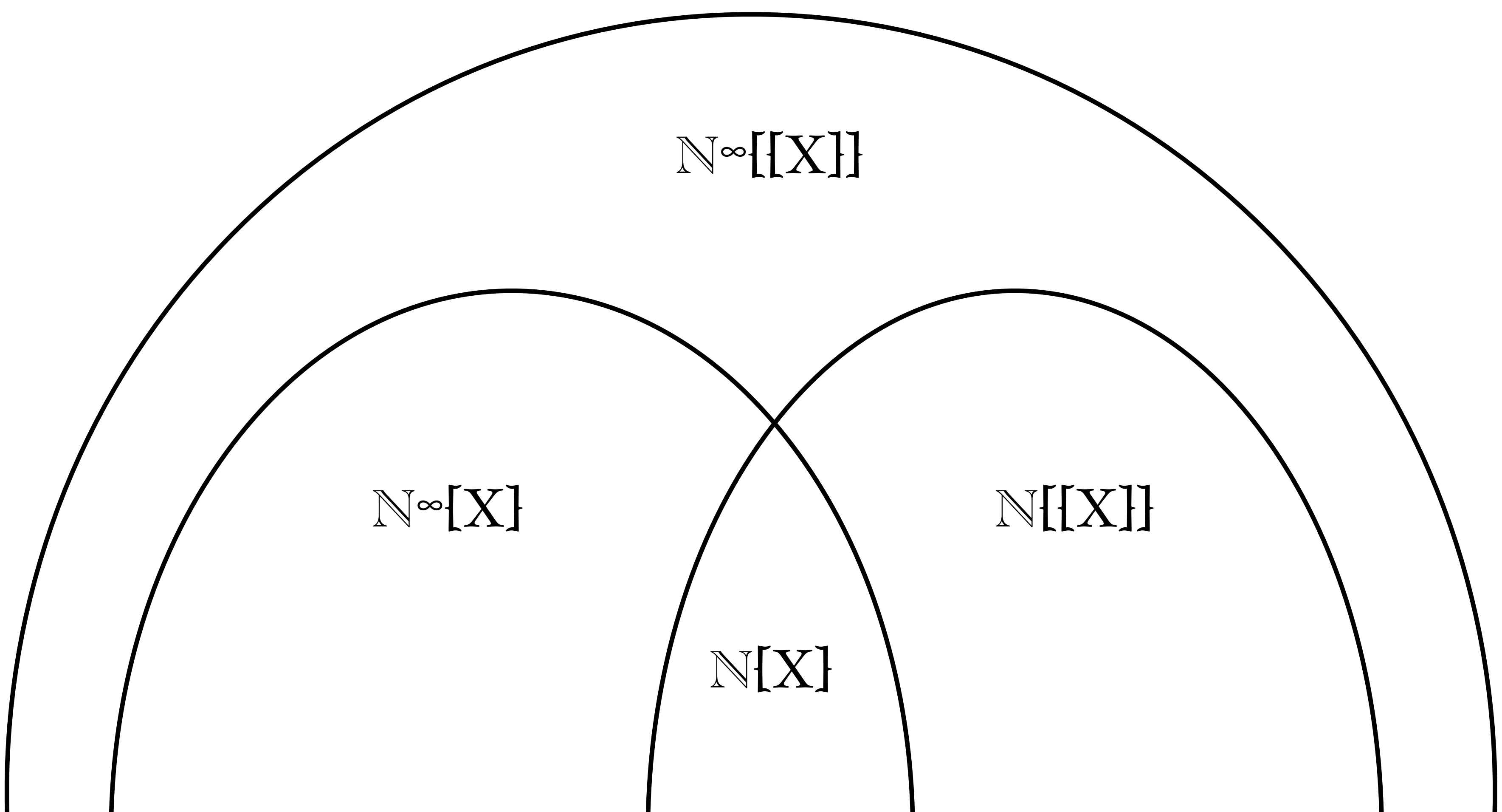
$$\mathbb{N}^\infty[[m,n,p,r,s]][\mathbf{x},\mathbf{y},\mathbf{z},\mathbf{u},\mathbf{v},\mathbf{w}]$$

As $\mathbb{N}^\infty[[m,n,p,r,s]]$ is omega continuos, these equations have least fixed points that can be computed.

Decidability

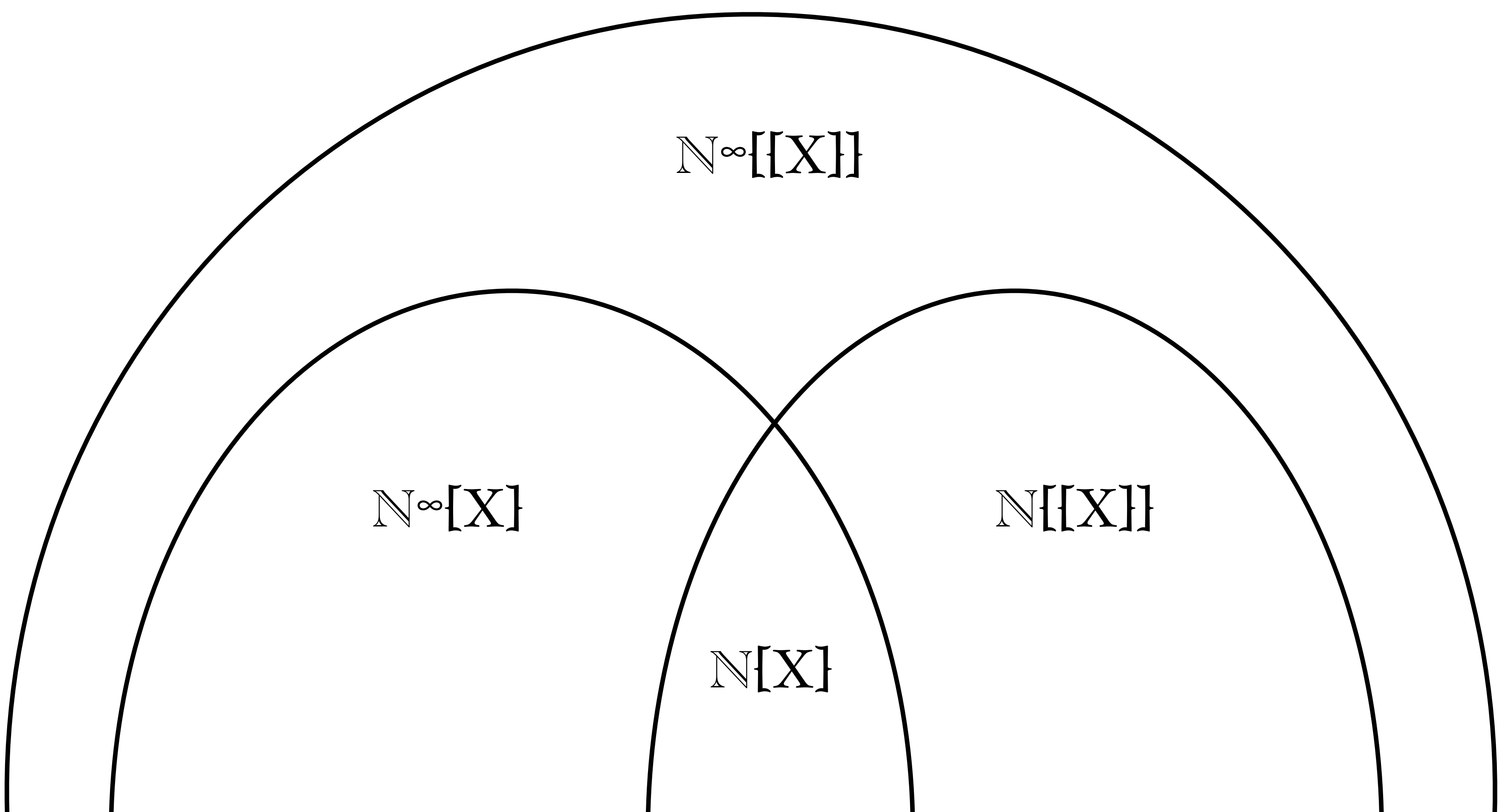
A tuple can have an annotations in any of the classes below.

It is decidable in which class the annotation of a tuple is.



Decidability: Case $\mathbb{N}[X]$

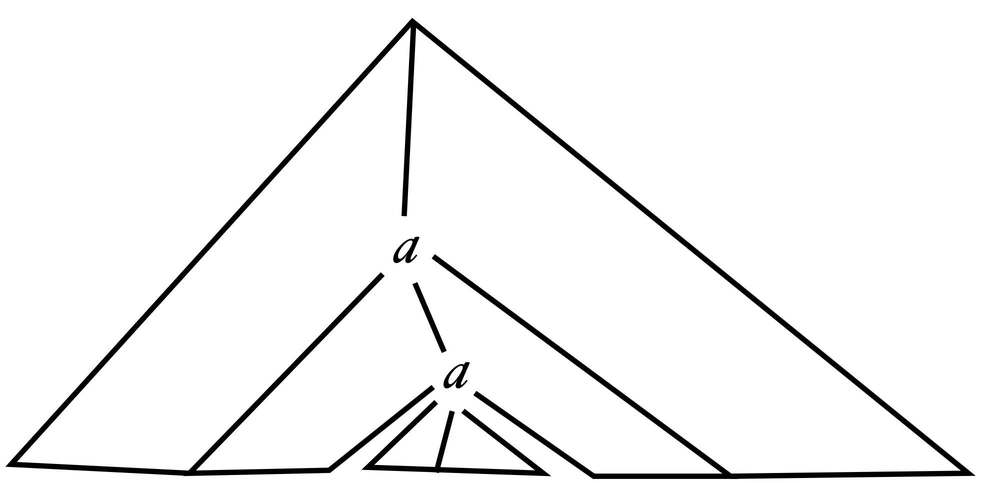
Claim: Let Q be a Datalog program, D a database, and R an relation in the intensional schema of P.
R(t) $\notin \mathbb{N}[X]$ iff t has a derivation tree T w.r.t. Q and D of height less than (# of atoms +2) that has a path with two occurrences of the same atom *a*.



Decidability: Case $\mathbb{N}[X]$

Claim: Let Q be a Datalog program, D a database, and R an relation in the intensional schema of P.
R(t) $\notin \mathbb{N}[X]$ iff t has a derivation tree T w.r.t. Q and D of height less than (# of atoms +2) that has a path with two occurrences of the same atom *a*.

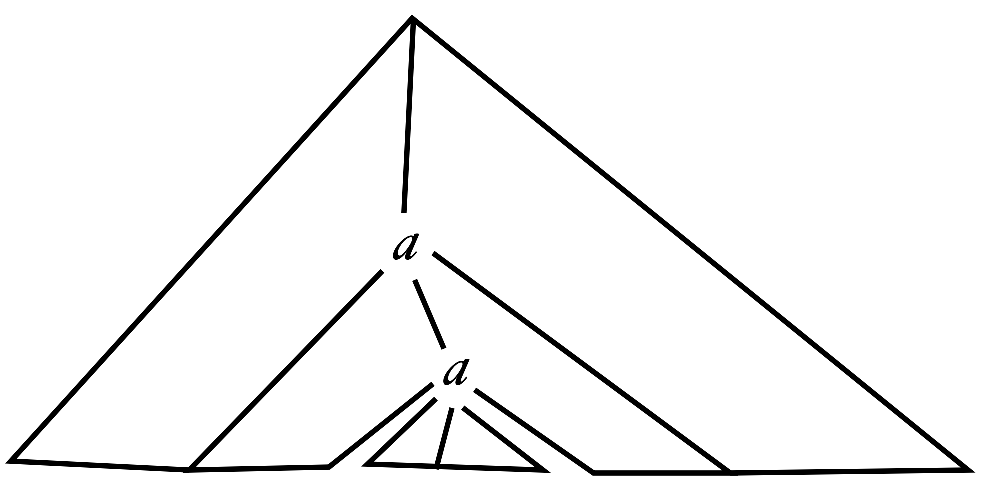
Proof: "⇐" Assume such a tree exists:



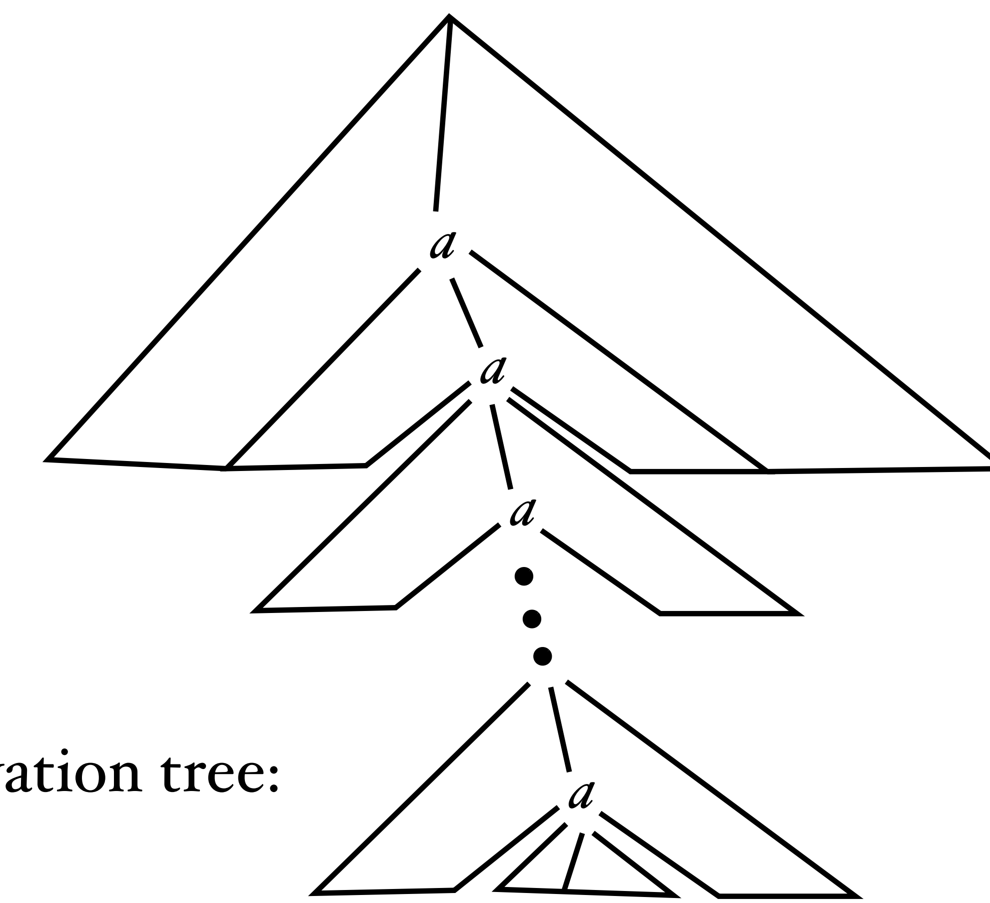
Decidability: Case $\mathbb{N}[X]$

Claim: Let Q be a Datalog program, D a database, and R an relation in the intensional schema of P.
R(t) $\notin \mathbb{N}[X]$ iff t has a derivation tree T w.r.t. Q and D of height less than (# of atoms +2) that has a path with two occurrences of the same atom *a*.

Proof: "⇐" Assume such a tree exists:



Then this is also a derivation tree:



Decidability: Case $\mathbb{N}[X]$

Claim: Let Q be a Datalog program, D a database, and R an relation in the intensional schema of P.
R(t) $\notin \mathbb{N}[X]$ iff t has a derivation tree T w.r.t. Q and D of height less than (# of atoms +2) that has a path with two occurrences of the same atom *a*.

Proof: "⇒" In particular, trees of height (# of atoms +1) have no path with two repeated atoms.

Decidability: Case $\mathbb{N}[X]$

Claim: Let Q be a Datalog program, D a database, and R an relation in the intensional schema of P.
R(t) $\notin \mathbb{N}[X]$ iff t has a derivation tree T w.r.t. Q and D of height less than (# of atoms +2) that has a path with two occurrences of the same atom *a*.

Proof: "⇒" In particular, trees of height (# of atoms +1) have no path with two repeated atoms.

Hence, by the pigeon hole principle, there are no derivation trees of height equal to (# of atoms +1).

Decidability: Case $\mathbb{N}[X]$

Claim: Let Q be a Datalog program, D a database, and R an relation in the intensional schema of P.
R(t) $\notin \mathbb{N}[X]$ iff t has a derivation tree T w.r.t. Q and D of height less than (# of atoms +2) that has a path with two occurrences of the same atom *a*.

Proof: "⇒" In particular, trees of height (# of atoms +1) have no path with two repeated atoms.

Hence, by the pigeon hole principle, there are no derivation trees of height ~~equal to~~ (# of atoms +1).

Decidability: Case $\mathbb{N}[X]$

Claim: Let Q be a Datalog program, D a database, and R an relation in the intensional schema of P.
R(t) $\notin \mathbb{N}[X]$ iff t has a derivation tree T w.r.t. Q and D of height less than (# of atoms +2) that has a path with two occurrences of the same atom *a*.

Proof: "⇒" In particular, trees of height (# of atoms +1) have no path with two repeated atoms.

Hence, by the pigeon hole principle, there are no derivation trees of height ~~equal to~~ (# of atoms +1).

greater than or equal

Decidability: Case $\mathbb{N}[X]$

Claim: Let Q be a Datalog program, D a database, and R an relation in the intensional schema of P.
$R(t) \notin \mathbb{N}[X]$ iff t has a derivation tree T w.r.t. Q and D of height less than (# of atoms +2) that has a path with two occurrences of the same atom *a*.

Proof: "⇒" In particular, trees of height (# of atoms +1) have no path with two repeated atoms.

Hence, by the pigeon hole principle, there are no derivation trees of height ~~equal to~~ (# of atoms +1).

greater than
or equal

Thus there are only finitely many derivation trees.

Also decidable:

- given $t \in q(I)$, and a monomial μ, the coefficient of μ in the power series that is the provenance of t is computable (including when it is ∞).

- testing whether all coefficients are $\neq \infty$.

Not decidable:

- testing whether all coefficients are 1.

Conclusion

- A versatile framework for provenance computation.
- Specializes to many known systems for provenance.
- In a sense most general within frameworks that use Semirings.
- Provides semantics for positive datalog under rich semantics (e.g. bag semantics).

Thank You!