Provenance in relational databases A Promising Formal Framework For Data Cleaning?

Romuald Thion

LIRIS, UMR 5205 CNRS Université Claude Bernard, Lyon 1







April 11, 2016

Provenance

Structures

**Query Evaluation** 

Access control

Conclusion

MedClean

#### First year research axis

- Collecte et Annotation des masses de données scientifiques
- Architecture de nettoyage de données scientifiques
- Utilisation des données probabilistes/incertaines

This talk : semiring-annotated data (a.k.a, relational provenance)

A formal framework which extends (traditional) relational algebra to annotated tuples [Green *et al.*, PODS'07]

#### Provenance

Structures

**Query Evaluation** 

Access control

Conclusion

MedClean

# The SPJRU Algebra

R	Α	В		5	Α	С
t <sub>0</sub>	а	1	ť	' )	а	2
t <sub>1</sub>	а	2	ť	; 	b	1

 $\phi := \operatorname{Rel} \mid \emptyset \mid \sigma_{\operatorname{P}}(\phi) \mid \pi_{\operatorname{V}}(\phi) \mid \phi \bowtie \psi \mid \rho_{\beta}(\phi) \mid \phi \cup \psi$ 

		A	B/C								
U		а	1					A	В	С	
							<i>j</i> o	а	1		
<i>u</i> <sub>1</sub> ·	ĺ		2 2				<i>j</i> 1	а	2		
U											
	A			$ ho_{\mathcal{C}/\mathcal{B}}(\mathcal{S})$	A	В				A	В
				r <sub>o</sub>			_			а	
 $p_0$	a			ľ1			_		50	a	~

#### MedClean

# The SPJRU Algebra

R	Α	В	S	' /	4	С
t <sub>0</sub>	а	1	$t'_0$	ě	а	2
<i>t</i> <sub>1</sub>	а	2	$t_1^{\check{\prime}}$	l	Ь	1

 $\phi := \operatorname{\textit{Rel}} \mid \emptyset \mid \sigma_{\operatorname{\textit{P}}}(\phi) \mid \pi_{\operatorname{\textit{V}}}(\phi) \mid \phi \bowtie \psi \mid \rho_{\beta}(\phi) \mid \phi \cup \psi$ 



#### MedClean

## The SPJRU Algebra

R	Α	В	S	Α	С
t <sub>0</sub>	а	1	$t'_0$	а	2
t <sub>1</sub>	а	2	$t_1^{\prime}$	b	1

 $\phi := \operatorname{\textit{Rel}} \mid \emptyset \mid \sigma_{\operatorname{\textit{P}}}(\phi) \mid \pi_{\operatorname{\textit{V}}}(\phi) \mid \phi \bowtie \psi \mid \rho_{\beta}(\phi) \mid \phi \cup \psi$ 

	$R \cup S$	Α	B/C								
	u <sub>0</sub>	а	1	-		_	$R \bowtie S$	Α	В	С	
	<i>u</i> <sub>1</sub> {	а	2				<i>j</i> o <i>j</i> 1	а	1	2	
	$u_1$	а	2				<i>j</i> 1	а	2	2	
	<i>U</i> <sub>2</sub>	b	1	_		-					
$\pi$	$_{A}(R)$ A		_	$\rho_{C/B}(S)$				σρ	(R)	A	B
	$p_0$ a			<i>r</i> <sub>0</sub>	а	2	_	<i>∨ D=</i> 4		a	
	P0 4	. <u> </u>	_	r <sub>1</sub>	b	1			00	u	

MedClean

Same language, more general semantics: set-based SPJRU operations are lifted to  $\langle \mathbb{K}, \oplus, \otimes, 0, 1 \rangle$ 

RAB			S	;	A	С	
t <sub>0</sub> a 1 t <sub>1</sub> a 2 			t' <sub>0</sub> t' <sub>1</sub>			2 1 	
$R \cup S$ $A$ $B/C$							
<i>u</i> <sub>0</sub> <i>a</i> 1		_ <i>R</i> ×	S	Α	В	С	
$u_1 \left\{ \begin{array}{cc} a & 2 \\ a & 2 \end{array} \right\}$							
u <sub>2</sub> b 1	$\beta_1$						
$\pi_{\mathcal{A}}(\boldsymbol{R})$ $\boldsymbol{A}$		$\sigma_{B=2}(I)$	<b>?</b> )	A	В		
$p_0$ <b>a</b> $\alpha_0$ $\oplus$			<i>S</i> 0	а	2		1 = $\alpha_1$

MedClean

Same language, more general semantics: set-based SPJRU operations are lifted to  $\langle \mathbb{K}, \oplus, \otimes, 0, 1 \rangle$ 

R A B			5	5	A	С	
t <sub>0</sub> a 1			$t_0'$	)	a	2	$\beta_0$
t <sub>1</sub> a 2			$t'_1$		b	1	
···· ···	0		····				0
$R \cup S$ $A$ $B/C$							
<i>u</i> <sub>0</sub> <i>a</i> 1	$\alpha_0$	<i>R</i> ×	1 <i>S</i>	Α	В	С	
$u_1 \left\{ egin{array}{cc} a & 2 \ a & 2 \end{array}  ight.$	$\alpha_1 \oplus \beta_0$		<i>j</i> o				$lpha_{f 0}\otimeseta_{f 0}$
			<i>j</i> 1	а	2	2	$\alpha_1\otimes \beta_0$
u <sub>2</sub> b 1	$\beta_1$						
$\pi_A(R)$ A		$\sigma_{B=2}(I)$	R)	Α	В		
$p_0 a \alpha_0 \oplus$	$\alpha_1$		<i>s</i> 0	а	2	$\alpha_1$ (	$\otimes 1 = \alpha_1$

MedClean

#### Example

Α	В		S	Α	С
а	1	$\alpha_0$	$t'_0$		
	2		$t'_1$		

$$\boldsymbol{Q} := \pi_{\boldsymbol{A}\boldsymbol{C}}(\boldsymbol{R} \bowtie \boldsymbol{S}) \cup \sigma_{\boldsymbol{B}=1}(\boldsymbol{R})$$

MedClean

#### Example



$$\boldsymbol{Q} := \pi_{\boldsymbol{A}\boldsymbol{C}}(\boldsymbol{R} \bowtie \boldsymbol{S}) \cup \sigma_{\boldsymbol{B}=1}(\boldsymbol{R})$$

MedClean

#### Example

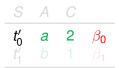


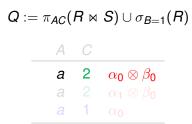
$$\boldsymbol{Q} := \pi_{\boldsymbol{A}\boldsymbol{C}}(\boldsymbol{R} \bowtie \boldsymbol{S}) \cup \sigma_{\boldsymbol{B}=1}(\boldsymbol{R})$$

MedClean

#### Example

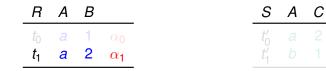


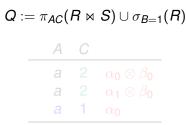




MedClean

### Example

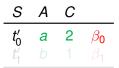


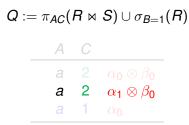


MedClean

### Example



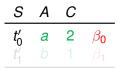


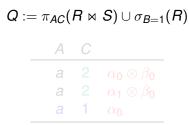


MedClean

### Example





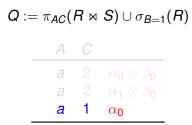


MedClean

#### Example







MedClean

### Example



$$Q := \pi_{AC}(R \bowtie S) \cup \sigma_{B=1}(R)$$

$$A C$$

$$a 2 \alpha_0 \otimes \beta_0$$

$$a 2 \alpha_1 \otimes \beta_0$$

$$a 1 \alpha_0$$

MedClean

### Example



$$Q := \pi_{AC}(R \bowtie S) \cup \sigma_{B=1}(R)$$
$$\frac{A \quad C}{a \quad 2 \quad (\alpha_0 \oplus \alpha_1) \otimes \beta_0}$$
$$a \quad 1 \quad \alpha_0$$

MedClean

The specific case of the *extended* SPJRU Algebra in which  $\langle \mathbb{K}, \oplus, \otimes, 0, 1 \rangle$  is instanciated to  $\langle \{0, 1\}, \lor, \land, 0, 1 \rangle$  is the classical SPJRU Algebra (with set semantics).

Key result [Green et. al., PODS'07]

The *extended* SPJRU algebra *"behaves well"*<sup>1</sup>, when the structure of annotations

 $\langle \mathbb{K}, \oplus, \otimes, 0, 1 \rangle$ 

is a commutative semiring

<sup>1</sup>morphisms between annotations commute with query evaluation MedClean *Provenance (for data cleaning)* 

Provenance

#### Structures

**Query Evaluation** 

Access control

Conclusion

MedClean

# Commutative semiring

### $\langle \mathbb{K}, \oplus, \otimes, 0, 1 \rangle$ formal definition

- K, underlying set
- $\blacktriangleright \ \langle \mathbb{K}, \oplus, \mathbf{0} \rangle$  a commutative (a.k.a., Abelian) monoid
  - (associative)  $(a \oplus b) \oplus c = a \oplus (b \oplus c)$
  - (unit)  $a \oplus 0 = a = 0 \oplus a$
  - (commutative)  $a \oplus b = b \oplus a$
- $\langle \mathbb{K}, \otimes, 1 \rangle$  a *commutative* monoid
- The two sub-monoids are linked together
  - ► (distribution law)  $a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c)$
  - ► (distribution law)<sup>2</sup>  $(b \oplus c) \otimes a = (b \otimes a) \oplus (c \otimes a)$
  - (absorption law)<sup>3</sup>  $0 \otimes a = 0 = a \otimes 0$

<sup>2</sup>Theorem when  $\langle \mathbb{K}, \otimes, 1 \rangle$  is commutative, but needed otherwise

<sup>3</sup>Theorem when  $\langle \mathbb{K}, \oplus, \otimes, 0, 1 \rangle$  is a ring, but needed otherwise

MedClean

### Some commutative semirings

- Boolean  $\mathbb{B} = \langle \{0, 1\}, \lor, \land, 0, 1 \rangle$
- ► Multiplicity (N, +, ×, 0, 1) i.e., bag semantics for relational algebra
- ► Security (L, min, max, O, P) with L = P < C < S < TS < O</p>
- {Where, How, Why} provenance: containers of containers<sup>4</sup> of tuple identifiers.
- Uncertainty  $\langle \mathcal{P}(\Omega), \cup, \cap, \emptyset, \Omega \rangle$
- Trust scores  $\langle \mathbb{R}^{\infty}_+, \textit{min}, +, \infty, \mathbf{0} \rangle$

### One semiring to rule them all: $\mathbb{N}[X]$

The set of all multivariate polynomials over a set X with integer coefficients is the *free*<sup>5</sup> *commutative semiring* over X.

<sup>&</sup>lt;sup>4</sup>e.g., *sets* of *bags*, *sets* of *sets*, etc.

<sup>&</sup>lt;sup>5</sup>intuitively, the syntactic algebra quotiented with the laws of a semiring MedClean *Provenance (for data cleaning)* 

# Combining heterogeneous semirings

### How to combine semirings?

- $\oplus$  and  $\otimes$  merge homogeneous annotations
- However, there may be different kinds of annotations in a data integration setting

### New semirings from old ones

- Given  $\langle \mathbb{K}_i, \oplus_i, \otimes_i, \mathbf{0}_i, \mathbf{1}_i \rangle$  an indexed family of semirings
- Construct de product semiring:
  - $\blacktriangleright \mathbb{K} = \mathbb{K}_0 \times \ldots \times \mathbb{K}_n$
  - extend  $\oplus_i$  and  $\otimes_i$  component wise
  - define  $0 = \langle 0_0, \dots, 0_n \rangle$  and  $1 = \langle 1_0, \dots, 1_n \rangle$
- ▶ Injectors: a  $\mathbb{K}_i$  should be read as a specific  $\mathbb{K}$  annotation
  - $\blacktriangleright \iota_i: \mathbb{K}_i \to \mathbb{K}$
  - ▶  $\iota_i(k) = (1_0, \ldots, 1_{i-1}, k, 1_{i+1}, \ldots, 1_n)$

#### MedClean

# Combining heterogeneous semirings

### How to combine semirings?

- $\oplus$  and  $\otimes$  merge homogeneous annotations
- However, there may be different kinds of annotations in a data integration setting

### New semirings from old ones

- Given  $\langle \mathbb{K}_i, \oplus_i, \otimes_i, \mathbf{0}_i, \mathbf{1}_i \rangle$  an indexed family of semirings
- Construct de product semiring:
  - $\mathbb{K} = \mathbb{K}_0 \times \ldots \times \mathbb{K}_n$
  - extend  $\oplus_i$  and  $\otimes_i$  component wise
  - define  $0 = \langle 0_0, \dots, 0_n \rangle$  and  $1 = \langle 1_0, \dots, 1_n \rangle$
- ► Injectors: a  $\mathbb{K}_i$  should be read as a specific  $\mathbb{K}$  annotation

$$\iota_i : \mathbb{K}_i \to \mathbb{K}$$

• 
$$\iota_i(k) = (\mathbf{1}_0, \ldots, \mathbf{1}_{i-1}, k, \mathbf{1}_{i+1}, \ldots, \mathbf{1}_n)$$

MedClean

Provenance

Structures

**Query Evaluation** 

Access control

Conclusion

MedClean

# TBAC – Query Evaluation VS Morphisms

### Homomorphism of semirings

An homomorphism between  $\langle \mathbb{K}_0, \oplus_0, \otimes_0, 0_0, 1_0 \rangle$  and  $\langle \mathbb{K}_1, \oplus_1, \otimes_1, 0_1, 1_1 \rangle$  is a structure-preserving function  $f : \mathbb{K}_0 \to \mathbb{K}_1$  between underyling sets:

- $f(0_0) = 0_1$
- $f(1_0) = 1_1$

$$\bullet f(a \oplus_0 b) = f(a) \oplus_1 f(b)$$

 $\bullet f(a \otimes_0 b) = f(a) \otimes_1 f(b)$ 

### Morphism into the boolean semiring

Let  $\langle \mathcal{P}(\mathcal{U}), \cup, \cap, \emptyset, \mathcal{U} \rangle$ , the following function  $f_c$  is an homomorphism from  $\mathcal{P}(\mathcal{U})$  to  $\langle \{0, 1\}, \vee, \wedge, 0, 1 \rangle$  for each  $c \in \mathcal{U}$ :

$$f_c(X) = c \in X$$

MedClean

TBAC – Query Evaluation VS Morphisms With  $Q = R \bowtie \rho_{B'/B}(R)$  and  $f = f_B$ Α В 1  $\alpha_0 = AB$ а **2**  $\alpha_1 = AC$ а

MedClean

Provenance (for data cleaning)

TBAC – Query Evaluation VS Morphisms With  $Q = R \bowtie \rho_{B'/B}(R)$  and  $f = f_B$ Α В Α В  $f(\alpha_0) = 1$ 1  $\alpha_0 = AB$ f a 1 а **2**  $\alpha_1 = AC$ а

MedClean

Provenance (for data cleaning)

TBAC – Query Evaluation VS Morphisms With  $Q = R \bowtie \rho_{B'/B}(R)$  and  $f = f_B$ Α В Α В f  $f(\alpha_0) = 1$ 1  $\alpha_0 = AB$ a 1 а **2**  $\alpha_1 = AC$ а  $\downarrow Q$ B В Α а 1 1

MedClean

Provenance (for data cleaning)

TBAC – Query Evaluation VS Morphisms With  $Q = R \bowtie \rho_{B'/B}(R)$  and  $f = f_B$ Α В 1  $\alpha_0 = AB$ а **2**  $\alpha_1 = AC$ а

MedClean

Provenance (for data cleaning)

TBAC – Query Evaluation VS Morphisms With  $Q = R \bowtie \rho_{B'/B}(R)$  and  $f = f_B$ Α В 1  $\alpha_0 = AB$ а 2  $\alpha_1 = AC$ а  $\downarrow Q$ В B Α а 1 1  $\alpha_0 \cap \alpha_0 = AB$ 1 2  $\alpha_0 \cap \alpha_1 = A$ а а 2 1  $\alpha_1 \cap \alpha_0 = A$ 2 2  $\alpha_1 \cap \alpha_1 = AC$ а

MedClean

TBAC – Query Evaluation VS Morphisms With  $Q = R \bowtie \rho_{B'/B}(R)$  and  $f = f_B$ Α В 1  $\alpha_0 = AB$ а 2  $\alpha_1 = AC$ а  $\downarrow Q$ В B Α а 1 1  $\alpha_0 \cap \alpha_0 = AB$ B Α В 2 f а 1  $\alpha_0 \cap \alpha_1 = A$ а 1 1 а 2 1  $\alpha_1 \cap \alpha_0 = A$ 2 2  $\alpha_1 \cap \alpha_1 = AC$ а

MedClean

TBAC – Query Evaluation VS Morphisms With  $Q = R \bowtie \rho_{B'/B}(R)$  and  $f = f_B$ Α В Α В f $f(\alpha_0) = 1$ 1  $\alpha_0 = AB$ a 1 а 2  $\alpha_1 = AC$ а The square commutes!  $\downarrow Q$  $\downarrow Q$ В B Α а 1 1  $\alpha_0 \cap \alpha_0 = AB$ B Α В 2 fа 1  $\alpha_0 \cap \alpha_1 = A$ а 1 1 а 2 1  $\alpha_1 \cap \alpha_0 = A$ 2 2  $\alpha_1 \cap \alpha_1 = AC$ а

MedClean

# Query Evaluation VS Morphisms

### Query Evaluation VS Morphisms With $Q(\mathcal{I})$ the evaluation of the query Q on an instance $\mathcal{I}$

 $eval_{AfterQ}(\mathcal{I})(Q) = \{(t: f(k)) \mid (t: k) \in Q(\mathcal{I})\}$ 

 $eval_{Before Q}(\mathcal{I})(Q) = Q(\{(t: f(k)) \mid (t: k) \in \mathcal{I}\})$ 

Evaluation commutes<sup>6</sup> with morphisms

 $eval_{AfterQ}(\mathcal{I})(Q) = eval_{BeforeQ}(\mathcal{I})(Q)$ 

16/23

<sup>6</sup>because extended SPJRU algebra "behaves well w.r.t. morphisms" MedClean Provenance (for data cleaning)

# Query Evaluation VS Morphisms

### Query Evaluation VS Morphisms With $Q(\mathcal{I})$ the evaluation of the query Q on an instance $\mathcal{I}$

 $eval_{AfterQ}(\mathcal{I})(Q) = \{(t : f(k)) \mid (t : k) \in Q(\mathcal{I})\}$ 

 $\mathsf{eval}_{\mathrm{Before}\,\mathcal{Q}}(\mathcal{I})(\mathcal{Q}) = \mathcal{Q}(\{(t:f(k)) \mid (t:k) \in \mathcal{I}\})$ 

Evaluation commutes<sup>6</sup> with morphisms

 $eval_{AfterQ}(\mathcal{I})(Q) = eval_{BeforeQ}(\mathcal{I})(Q)$ 

<sup>6</sup>because extended SPJRU algebra "behaves well w.r.t. morphisms" MedClean Provenance (for data cleaning)

## Query Evaluation VS Morphisms

## Query Evaluation VS Morphisms With $Q(\mathcal{I})$ the evaluation of the query Q on an instance $\mathcal{I}$

 $eval_{AfterQ}(\mathcal{I})(Q) = \{(t: f(k)) \mid (t: k) \in Q(\mathcal{I})\}$ 

## $\mathsf{eval}_{\mathrm{Before}\,\mathcal{Q}}(\mathcal{I})(\mathcal{Q}) = \mathcal{Q}(\{(t:f(k)) \mid (t:k) \in \mathcal{I}\})$

Evaluation commutes<sup>6</sup> with morphisms

 $eval_{AfterQ}(\mathcal{I})(Q) = eval_{BeforeQ}(\mathcal{I})(Q)$ 

16/23

<sup>6</sup>because extended SPJRU algebra "behaves well w.r.t. morphisms" MedClean Provenance (for data cleaning)

## Query Evaluation VS Morphisms

## Query Evaluation VS Morphisms With $Q(\mathcal{I})$ the evaluation of the query Q on an instance $\mathcal{I}$

 $eval_{AfterQ}(\mathcal{I})(Q) = \{(t : f(k)) \mid (t : k) \in Q(\mathcal{I})\}$ 

 $eval_{Before Q}(\mathcal{I})(Q) = Q(\{(t: f(k)) \mid (t: k) \in \mathcal{I}\})$ 

Evaluation commutes<sup>6</sup> with morphisms

 $eval_{AfterQ}(\mathcal{I})(Q) = eval_{BeforeQ}(\mathcal{I})(Q)$ 

<sup>6</sup>because extended SPJRU algebra *"behaves well w.r.t. morphisms"* MedClean *Provenance (for data cleaning)* 

#### Context

Provenance

Structures

**Query Evaluation** 

Access control

Conclusion

MedClean

## Tuple-Based Access Control (TBAC) (1/2)

## Key insight

## $\langle \mathbb{K},\oplus,\otimes,0,1\rangle$

### is the domain of authorizations

## Intuitive authorization semantics

0 and 1 are extremal policies of type  $\mathbb K$ 

- ▶  $0 \in \mathbb{K}$  : deny all
- ▶  $1 \in \mathbb{K}$  : authorize all

### $\oplus$ and $\otimes~$ are policy combinators of type $\mathbb{K}\times\mathbb{K}\to\mathbb{K}$

- ► ⊕: addition (disjunction)
- ▶ ⊗: multiplication (conjunction)

#### MedClean

## Tuple-Based Access Control (TBAC) (1/2)

Key insight

## $\langle \mathbb{K},\oplus,\otimes,0,1\rangle$

### is the domain of authorizations

## Intuitive authorization semantics

0 and 1 are extremal policies of type  $\mathbb K$ 

- ▶ 0 ∈ K : deny all
- I ∈ K : authorize all

### $\oplus \text{ and } \otimes \text{ are policy combinators of type } \mathbb{K} \times \mathbb{K} \to \mathbb{K}$

- ⊕: addition (disjunction)
- ▶ ⊗: multiplication (conjunction)

MedClean

## Tuple-Based Access Control (TBAC) (2/2)

Example:  $\langle \mathbb{K}, \oplus, \otimes, 0, 1 \rangle$  is  $\langle \mathcal{P}(\mathcal{U}), \cup, \cap, \emptyset, \mathcal{U} \rangle$ 

• 
$$\alpha_0 = \{ Alice, Bob \} = AB$$

•  $\alpha_1 = \{ Alice, Charlie \} = AC$ 

► 
$$\beta_0 = \{\mathsf{Bob}\} = B$$

$$\mathbb{K}$$
 $\mathcal{P}(\mathcal{U})$  $a$ 2 $(\alpha_0 \oplus \alpha_1) \otimes \beta_0$  $B = (AB \cup AC) \cap B$  $a$ 1 $\alpha_0$  $AB$ 

 $Q := \pi_{AC}(R \bowtie \rho_{C/B}(S)) \cup \sigma_{B=1}(R)$ 

MedClean

## Tuple-Based Access Control (TBAC) (2/2)

Example:  $\langle \mathbb{K}, \oplus, \otimes, 0, 1 \rangle$  is  $\langle \mathcal{P}(\mathcal{U}), \cup, \cap, \emptyset, \mathcal{U} \rangle$ 

• 
$$\alpha_0 = \{\text{Alice, Bob}\} = AB$$
  
•  $\alpha_1 = \{\text{Alice, Charlie}\} = AC$   
•  $\beta_0 = \{\text{Bob}\} = B$   
 $\mathbb{K} \qquad \mathcal{P}(\mathcal{U})$ 

а	2	$(\alpha_{0}\oplus \alpha_{1})\otimes \beta_{0}$	$B = (AB \cup AC) \cap B$
а	1	$\alpha_0$	AB

$$\boldsymbol{Q} := \pi_{\boldsymbol{A}\boldsymbol{C}}(\boldsymbol{R} \bowtie \rho_{\boldsymbol{C}/\boldsymbol{B}}(\boldsymbol{S})) \cup \sigma_{\boldsymbol{B}=1}(\boldsymbol{R})$$

MedClean

# TBAC – Filtering (1/2)

### Filtering function $f : \mathbb{C} \to \mathbb{K} \to \mathbb{B}$

With  $\mathbb{C}$  the set of credentials, associated to subjects:

- $f(c)(k) = \top$  if k allows c to read (t:k)
- $f(c)(k) = \bot$  if k denies c to read (t : k)

### Example: $\langle \mathcal{P}(\mathcal{U}), \cup, \cap, \emptyset, \mathcal{U} \rangle$ with $f(c)(k) = c \in k$

Α	С		f(Alice)	f(Bob)	f(Charlie)
а		В			
а	1	AB			

Filtering *Q* with *X*'s identity is to compute f(X)(k)

MedClean



# TBAC – Filtering (1/2)

### Filtering function $f : \mathbb{C} \to \mathbb{K} \to \mathbb{B}$

With  $\mathbb{C}$  the set of credentials, associated to subjects:

- $f(c)(k) = \top$  if k allows c to read (t : k)
- $f(c)(k) = \bot$  if k denies c to read (t : k)

Example:  $\langle \mathcal{P}(\mathcal{U}), \cup, \cap, \emptyset, \mathcal{U} \rangle$  with  $f(c)(k) = c \in k$ 

Α	С		f(Alice)	f(Bob)	f(Charlie)
а	2	В	$\perp$	Т	$\perp$
а	1	AB	Т	Т	$\perp$

Filtering *Q* with *X*'s identity is to compute f(X)(k)

MedClean

"one may read a tuple if he/she is has access to the source tuples which contribute to it"

- f(c)(0) = ⊥ nobody can read (t : 0)
- f(c)(1) = ⊤ anybody can read (t : 1)
- f(c)(a ⊕ b) = f(c)(a) ∨ f(c)(b) one can read (t : a ⊕ b) if he/she can read either a or b
- f(c)(a ⊗ b) = f(x)(a) ∧ f(x)(b) one can read (t : a ⊗ b) if he/she can read both a and b

f(c) is a morphism from  $\langle K, \oplus, \otimes, 0, 1 \rangle$  into  $\langle \mathbb{B}, \vee, \wedge, \bot, \top \rangle$ 

#### MedClean

> "one may read a tuple if he/she is has access to the source tuples which contribute to it"

- *f*(*c*)(0) = ⊥
   *nobody* can read (*t* : 0)
- *f*(*c*)(1) = ⊤
   *anybody* can read (*t* : 1)
- *f*(*c*)(*a* ⊕ *b*) = *f*(*c*)(*a*) ∨ *f*(*c*)(*b*)
   one can read (*t* : *a* ⊕ *b*) if he/she can read either a or b
- f(c)(a ⊗ b) = f(x)(a) ∧ f(x)(b)
   one can read (t : a ⊗ b) if he/she can read both a and b

f(c) is a morphism from  $\langle K, \oplus, \otimes, 0, 1 \rangle$  into  $\langle \mathbb{B}, \lor, \land, \bot, \top \rangle$ 

MedClean

> "one may read a tuple if he/she is has access to the source tuples which contribute to it"

- *f*(*c*)(0) = ⊥
   *nobody* can read (*t* : 0)
- *f*(*c*)(1) = ⊤
   *anybody* can read (*t* : 1)
- ►  $f(c)(a \oplus b) = f(c)(a) \lor f(c)(b)$ one can read  $(t : a \oplus b)$  if he/she can read either a or b
- f(c)(a ⊗ b) = f(x)(a) ∧ f(x)(b)
   one can read (t : a ⊗ b) if he/she can read both a and b

f(c) is a morphism from  $\langle K, \oplus, \otimes, 0, 1 \rangle$  into  $\langle \mathbb{B}, \vee, \wedge, \bot, \top \rangle$ 

MedClean

> "one may read a tuple if he/she is has access to the source tuples which contribute to it"

- *f*(*c*)(0) = ⊥
   *nobody* can read (*t* : 0)
- ►  $f(c)(1) = \top$ anybody can read (t : 1)
- *f*(*c*)(*a* ⊕ *b*) = *f*(*c*)(*a*) ∨ *f*(*c*)(*b*) one can read (*t* : *a* ⊕ *b*) if he/she can read either a or b
- *f*(*c*)(*a* ⊗ *b*) = *f*(*x*)(*a*) ∧ *f*(*x*)(*b*) one can read (*t* : *a* ⊗ *b*) if he/she can read both a and b

f(c) is a morphism from  $\langle K, \oplus, \otimes, 0, 1 \rangle$  into  $\langle \mathbb{B}, \vee, \wedge, \bot, \top \rangle$ 

MedClean

> "one may read a tuple if he/she is has access to the source tuples which contribute to it"

- *f*(*c*)(0) = ⊥
   *nobody* can read (*t* : 0)
- ►  $f(c)(1) = \top$ anybody can read (t : 1)
- *f*(*c*)(*a* ⊕ *b*) = *f*(*c*)(*a*) ∨ *f*(*c*)(*b*)
   one can read (*t* : *a* ⊕ *b*) if he/she can read either a or b
- *f*(*c*)(*a* ⊗ *b*) = *f*(*x*)(*a*) ∧ *f*(*x*)(*b*) one can read (*t* : *a* ⊗ *b*) if he/she can read both a and b

f(c) is a morphism from  $\langle K, \oplus, \otimes, 0, 1 \rangle$  into  $\langle \mathbb{B}, \lor, \land, \bot, \top \rangle$ 

MedClean

> "one may read a tuple if he/she is has access to the source tuples which contribute to it"

- *f*(*c*)(0) = ⊥
   *nobody* can read (*t* : 0)
- ►  $f(c)(1) = \top$ anybody can read (t : 1)
- *f*(*c*)(*a* ⊕ *b*) = *f*(*c*)(*a*) ∨ *f*(*c*)(*b*)
   one can read (*t* : *a* ⊕ *b*) if he/she can read either *a* or *b*
- *f*(*c*)(*a* ⊗ *b*) = *f*(*x*)(*a*) ∧ *f*(*x*)(*b*) one can read (*t* : *a* ⊗ *b*) if he/she can read both *a* and *b*

f(c) is a morphism from  $\langle K, \oplus, \otimes, 0, 1 \rangle$  into  $\langle \mathbb{B}, \lor, \land, \bot, \top \rangle$ 

MedClean

#### Context

Provenance

Structures

**Query Evaluation** 

Access control

Conclusion

MedClean



### Conclusion

- generic and algebraic approach
- "behaves well" w.r.t. query evaluation
- can be extended to Datalog (with proper limit condition)

## Extensions

- $\blacktriangleright$  Set-inspired negation: partial order and difference on  $\mathbb K$
- ► Aggregation operators: K-semimodules

### Drawbacks

- Implementations!
- Hard to divide between data and metadata?

## Thank you for your attention!

MedClean

