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

Context

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]

The SPJRU Algebra

R	A	B
t_0	a	1
t_1	a	2

S	A	C
t'_0	a	2
t'_1	b	1

$$\phi := Rel \mid \emptyset \mid \sigma_P(\phi) \mid \pi_V(\phi) \mid \phi \bowtie \psi \mid \rho_\beta(\phi) \mid \phi \cup \psi$$

$R \cup S$	A	B/C
u_0	a	1
u_1	a	2
	a	2
u_2	b	1

$R \bowtie S$	A	B	C
j_0	a	1	2
j_1	a	2	2

$\pi_A(R)$	A
p_0	a

$\rho_{C/B}(S)$	A	B
r_0	a	2
r_1	b	1

$\sigma_{B=2}(R)$	A	B
s_0	a	2

The SPJRU Algebra

R	A	B
t_0	a	1
t_1	a	2

S	A	C
t'_0	a	2
t'_1	b	1

$$\phi := Rel \mid \emptyset \mid \sigma_P(\phi) \mid \pi_V(\phi) \mid \phi \bowtie \psi \mid \rho_\beta(\phi) \mid \phi \cup \psi$$

$R \cup S$	A	B/C
u_0	a	1
u_1	a	2
	a	2
u_2	b	1

$R \bowtie S$	A	B	C
j_0	a	1	2
j_1	a	2	2

$\pi_A(R)$	A
p_0	a

$\rho_{C/B}(S)$	A	B
r_0	a	2
r_1	b	1

$\sigma_{B=2}(R)$	A	B
s_0	a	2

The SPJRU Algebra

R	A	B
t_0	a	1
t_1	a	2

S	A	C
t'_0	a	2
t'_1	b	1

$$\phi := Rel \mid \emptyset \mid \sigma_P(\phi) \mid \pi_V(\phi) \mid \phi \bowtie \psi \mid \rho_\beta(\phi) \mid \phi \cup \psi$$

$R \cup S$	A	B/C
u_0	a	1
u_1	a	2
	a	2
u_2	b	1

$R \bowtie S$	A	B	C
j_0	a	1	2
j_1	a	2	2

$\pi_A(R)$	A
p_0	a

$\rho_{C/B}(S)$	A	B
r_0	a	2
r_1	b	1

$\sigma_{B=2}(R)$	A	B
s_0	a	2

The *extended* SPJRU Algebra

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

R	A	B	
t_0	a	1	α_0
t_1	a	2	α_1
...	...	...	0

S	A	C	
t'_0	a	2	β_0
t'_1	b	1	β_1
...	...	...	0

$R \cup S$	A	B/C	
u_0	a	1	α_0
u_1	a	2	$\alpha_1 \oplus \beta_0$
	a	2	
u_2	b	1	β_1

$R \bowtie S$	A	B	C	
j_0	a	1	2	$\alpha_0 \otimes \beta_0$
j_1	a	2	2	$\alpha_1 \otimes \beta_0$

$\pi_A(R)$	A	
p_0	a	$\alpha_0 \oplus \alpha_1$

$\sigma_{B=2}(R)$	A	B	
s_0	a	2	$\alpha_1 \otimes 1 = \alpha_1$

The *extended* SPJRU Algebra

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

R	A	B	
t_0	a	1	α_0
t_1	a	2	α_1
...	...	...	0

S	A	C	
t'_0	a	2	β_0
t'_1	b	1	β_1
...	...	...	0

$R \cup S$	A	B/C	
u_0	a	1	α_0
u_1	a	2	$\alpha_1 \oplus \beta_0$
	a	2	
u_2	b	1	β_1

$R \bowtie S$	A	B	C	
j_0	a	1	2	$\alpha_0 \otimes \beta_0$
j_1	a	2	2	$\alpha_1 \otimes \beta_0$

$\pi_A(R)$	A	
p_0	a	$\alpha_0 \oplus \alpha_1$

$\sigma_{B=2}(R)$	A	B	
s_0	a	2	$\alpha_1 \otimes 1 = \alpha_1$

The *extended* SPJRU Algebra

Example

R	A	B	
t_0	a	1	α_0
t_1	a	2	α_1

S	A	C	
t'_0	a	2	β_0
t'_1	b	1	β_1

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

The *extended* SPJRU Algebra

Example

R	A	B	
t_0	a	1	α_0
t_1	a	2	α_1

S	A	C	
t'_0	a	2	β_0
t'_1	b	1	β_1

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

The *extended* SPJRU Algebra

Example

R	A	B	
t_0	a	1	α_0
t_1	a	2	α_1

S	A	C	
t'_0	a	2	β_0
t'_1	b	1	β_1

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

The *extended* SPJRU Algebra

Example

R	A	B	
t_0	a	1	α_0
t_1	a	2	α_1

S	A	C	
t'_0	a	2	β_0
t'_1	b	1	β_1

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

A	C	
a	2	$\alpha_0 \otimes \beta_0$
a	2	$\alpha_1 \otimes \beta_0$
a	1	α_0

The *extended* SPJRU Algebra

Example

R	A	B	
t_0	a	1	α_0
t_1	a	2	α_1

S	A	C	
t'_0	a	2	β_0
t'_1	b	1	β_1

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

A	C	
a	2	$\alpha_0 \otimes \beta_0$
a	2	$\alpha_1 \otimes \beta_0$
a	1	α_0

The *extended* SPJRU Algebra

Example

R	A	B	
t_0	a	1	α_0
t_1	a	2	α_1

S	A	C	
t'_0	a	2	β_0
t'_1	b	1	β_1

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

A	C	
a	2	$\alpha_0 \otimes \beta_0$
a	2	$\alpha_1 \otimes \beta_0$
a	1	α_0

The *extended* SPJRU Algebra

Example

R	A	B	
t_0	a	1	α_0
t_1	a	2	α_1

S	A	C	
t'_0	a	2	β_0
t'_1	b	1	β_1

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

A	C	
a	2	$\alpha_0 \otimes \beta_0$
a	2	$\alpha_1 \otimes \beta_0$
a	1	α_0

The *extended* SPJRU Algebra

Example

R	A	B	
t_0	a	1	α_0
t_1	a	2	α_1

S	A	C	
t'_0	a	2	β_0
t'_1	b	1	β_1

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

A	C	
a	2	$\alpha_0 \otimes \beta_0$
a	2	$\alpha_1 \otimes \beta_0$
a	1	α_0

The *extended* SPJRU Algebra

Example

R	A	B	
t_0	a	1	α_0
t_1	a	2	α_1

S	A	C	
t'_0	a	2	β_0
t'_1	b	1	β_1

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

A	C		
a	2	$\alpha_0 \otimes \beta_0$	}
a	2	$\alpha_1 \otimes \beta_0$	
a	1	α_0	

The *extended* SPJRU Algebra

Example

R	A	B	
t_0	a	1	α_0
t_1	a	2	α_1

S	A	C	
t'_0	a	2	β_0
t'_1	b	1	β_1

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

A	C	
a	2	$(\alpha_0 \oplus \alpha_1) \otimes \beta_0$
a	1	α_0

The *extended* SPJRU Algebra

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

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

The *extended* SPJRU algebra *"behaves well"* [1], when the structure of annotations

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

is a *commutative semiring*

[1] morphisms between annotations commute with query evaluation

Commutative semiring

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

- $\mathbb{K}$, underlying set
- $\langle \mathbb{K}, \oplus, 0 \rangle$ 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)[2] $(b \oplus c) \otimes a = (b \otimes a) \oplus (c \otimes a)$
  - (absorption law)[3] $0 \otimes a = 0 = a \otimes 0$

[2] Theorem when $\langle \mathbb{K}, \otimes, 1 \rangle$ is commutative, but needed otherwise

[3] Theorem when $\langle \mathbb{K}, \oplus, \otimes, 0, 1 \rangle$ is a ring, but needed otherwise

Some commutative semirings

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

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*[5] *commutative semiring* over X.

---

[4] e.g., *sets* of *bags*, *sets* of *sets*, etc.

[5] intuitively, the syntactic algebra quotiented with the laws of a semiring

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 $\langle \mathbb{K}_i, \oplus_i, \otimes_i, 0_i, 1_i \rangle$ 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 $0 = \langle 0_0, \ldots, 0_n \rangle$ and $1 = \langle 1_0, \ldots, 1_n \rangle$
- Injectors: a $\mathbb{K}_i$ should be read as a specific $\mathbb{K}$ annotation
  - $\iota_i : \mathbb{K}_i \to \mathbb{K}$
  - $\iota_i(k) = (1_0, \ldots, 1_{i-1}, k, 1_{i+1}, \ldots, 1_n)$

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 $\langle \mathbb{K}_i, \oplus_i, \otimes_i, 0_i, 1_i \rangle$ 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 $0 = \langle 0_0, \ldots, 0_n \rangle$ and $1 = \langle 1_0, \ldots, 1_n \rangle$
- Injectors: a $\mathbb{K}_i$ should be read as a specific $\mathbb{K}$ annotation
  - $\iota_i : \mathbb{K}_i \to \mathbb{K}$
  - $\iota_i(k) = (1_0, \ldots, 1_{i-1}, k, 1_{i+1}, \ldots, 1_n)$

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:

- $f(0_0) = 0_1$
- $f(1_0) = 1_1$
- $f(a \oplus_0 b) = f(a) \oplus_1 f(b)$
- $f(a \otimes_0 b) = f(a) \otimes_1 f(b)$

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 $c \in \mathcal{U}$:

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

TBAC – Query Evaluation VS Morphisms

With $Q = R \bowtie \rho_{B'/B}(R)$ and $f = f_B$

A	B	
a	1	$\alpha_0 = AB$
a	2	$\alpha_1 = AC$

$\xrightarrow{f}$

A	B	
a	1	$f(\alpha_0) = 1$

$\downarrow Q$ $\downarrow Q$

A	B	B'	
a	1	1	$\alpha_0 \cap \alpha_0 = AB$
a	1	2	$\alpha_0 \cap \alpha_1 = A$
a	2	1	$\alpha_1 \cap \alpha_0 = A$
a	2	2	$\alpha_1 \cap \alpha_1 = AC$

$\xrightarrow{f}$

A	B	B'	
a	1	1	1

TBAC – Query Evaluation VS Morphisms

With $Q = R \bowtie \rho_{B'/B}(R)$ and $f = f_B$

A	B	
a	1	$\alpha_0 = AB$
a	2	$\alpha_1 = AC$

$\xrightarrow{f}$

A	B	
a	1	$f(\alpha_0) = 1$

$\downarrow Q$ $\downarrow Q$

A	B	B'	
a	1	1	$\alpha_0 \cap \alpha_0 = AB$
a	1	2	$\alpha_0 \cap \alpha_1 = A$
a	2	1	$\alpha_1 \cap \alpha_0 = A$
a	2	2	$\alpha_1 \cap \alpha_1 = AC$

$\xrightarrow{f}$

A	B	B'	
a	1	1	1

TBAC – Query Evaluation VS Morphisms

With $Q = R \bowtie \rho_{B'/B}(R)$ and $f = f_B$

A	B	
a	1	$\alpha_0 = AB$
a	2	$\alpha_1 = AC$

$\xrightarrow{f}$

A	B	
a	1	$f(\alpha_0) = 1$

$\downarrow Q$

$\downarrow Q$

A	B	B'	
a	1	1	$\alpha_0 \cap \alpha_0 = AB$
a	1	2	$\alpha_0 \cap \alpha_1 = A$
a	2	1	$\alpha_1 \cap \alpha_0 = A$
a	2	2	$\alpha_1 \cap \alpha_1 = AC$

$\xrightarrow{f}$

A	B	B'	
a	1	1	1

TBAC – Query Evaluation VS Morphisms

With $Q = R \bowtie \rho_{B'/B}(R)$ and $f = f_B$

A	B	
a	1	$\alpha_0 = AB$
a	2	$\alpha_1 = AC$

$\xrightarrow{f}$

A	B	
a	1	$f(\alpha_0) = 1$

$\downarrow Q$ $\downarrow Q$

A	B	B'	
a	1	1	$\alpha_0 \cap \alpha_0 = AB$
a	1	2	$\alpha_0 \cap \alpha_1 = A$
a	2	1	$\alpha_1 \cap \alpha_0 = A$
a	2	2	$\alpha_1 \cap \alpha_1 = AC$

$\xrightarrow{f}$

A	B	B'	
a	1	1	1

TBAC – Query Evaluation VS Morphisms

With $Q = R \bowtie \rho_{B'/B}(R)$ and $f = f_B$

A	B	
a	1	$\alpha_0 = AB$
a	2	$\alpha_1 = AC$

$\xrightarrow{f}$

A	B	
a	1	$f(\alpha_0) = 1$

$\downarrow Q$ $\qquad$ $\downarrow Q$

A	B	B'	
a	1	1	$\alpha_0 \cap \alpha_0 = AB$
a	1	2	$\alpha_0 \cap \alpha_1 = A$
a	2	1	$\alpha_1 \cap \alpha_0 = A$
a	2	2	$\alpha_1 \cap \alpha_1 = AC$

$\xrightarrow{f}$

A	B	B'	
a	1	1	1

TBAC – Query Evaluation VS Morphisms

With $Q = R \bowtie \rho_{B'/B}(R)$ and $f = f_B$

A	B	
a	1	$\alpha_0 = AB$
a	2	$\alpha_1 = AC$

$\xrightarrow{f}$

A	B	
a	1	$f(\alpha_0) = 1$

$\downarrow Q$

$\downarrow Q$

A	B	B′	
a	1	1	$\alpha_0 \cap \alpha_0 = AB$
a	1	2	$\alpha_0 \cap \alpha_1 = A$
a	2	1	$\alpha_1 \cap \alpha_0 = A$
a	2	2	$\alpha_1 \cap \alpha_1 = AC$

$\xrightarrow{f}$

A	B	B′	
a	1	1	1

TBAC – Query Evaluation VS Morphisms

With $Q = R \bowtie \rho_{B'/B}(R)$ and $f = f_B$

A	B	
a	1	$\alpha_0 = AB$
a	2	$\alpha_1 = AC$

$\xrightarrow{f}$

A	B	
a	1	$f(\alpha_0) = 1$

$\downarrow Q$ The square commutes! $\downarrow Q$

A	B	B'	
a	1	1	$\alpha_0 \cap \alpha_0 = AB$
a	1	2	$\alpha_0 \cap \alpha_1 = A$
a	2	1	$\alpha_1 \cap \alpha_0 = A$
a	2	2	$\alpha_1 \cap \alpha_1 = AC$

$\xrightarrow{f}$

A	B	B'	
a	1	1	1

Query Evaluation VS Morphisms

Query Evaluation VS Morphisms

With $Q(\mathcal{I})$ the evaluation of the query Q on an instance $\mathcal{I}$

$$\text{eval}_{\text{After}Q}(\mathcal{I})(Q) = \{(t : f(k)) \mid (t : k) \in Q(\mathcal{I})\}$$

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

Evaluation commutes[6] with morphisms

$$\text{eval}_{\text{After}Q}(\mathcal{I})(Q) = \text{eval}_{\text{Before}Q}(\mathcal{I})(Q)$$

---

[6] because extended SPJRU algebra *"behaves well w.r.t. morphisms"*

Query Evaluation VS Morphisms

Query Evaluation VS Morphisms

With $Q(\mathcal{I})$ the evaluation of the query Q on an instance $\mathcal{I}$

$$\text{eval}_{\text{After}Q}(\mathcal{I})(Q) = \{(t : f(k)) \mid (t : k) \in Q(\mathcal{I})\}$$

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

Evaluation commutes[6] with morphisms

$$\text{eval}_{\text{After}Q}(\mathcal{I})(Q) = \text{eval}_{\text{Before}Q}(\mathcal{I})(Q)$$

---

[6] because extended SPJRU algebra *"behaves well w.r.t. morphisms"*

Query Evaluation VS Morphisms

Query Evaluation VS Morphisms

With $Q(\mathcal{I})$ the evaluation of the query Q on an instance $\mathcal{I}$

$$\text{eval}_{\text{After}Q}(\mathcal{I})(Q) = \{(t : f(k)) \mid (t : k) \in Q(\mathcal{I})\}$$

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

Evaluation commutes[6] with morphisms

$$\text{eval}_{\text{After}Q}(\mathcal{I})(Q) = \text{eval}_{\text{Before}Q}(\mathcal{I})(Q)$$

[6] because extended SPJRU algebra *"behaves well w.r.t. morphisms"*

Query Evaluation VS Morphisms

Query Evaluation VS Morphisms

With $Q(\mathcal{I})$ the evaluation of the query Q on an instance $\mathcal{I}$

$$\text{eval}_{\text{After}Q}(\mathcal{I})(Q) = \{(t : f(k)) \mid (t : k) \in Q(\mathcal{I})\}$$

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

Evaluation commutes[6] with morphisms

$$\text{eval}_{\text{After}Q}(\mathcal{I})(Q) = \text{eval}_{\text{Before}Q}(\mathcal{I})(Q)$$

---

[6] because extended SPJRU algebra *"behaves well w.r.t. morphisms"*

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 $\mathbb{K} \times \mathbb{K} \to \mathbb{K}$

- $\oplus$: *addition* (*disjunction*)
- $\otimes$: *multiplication* (*conjunction*)

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 $\mathbb{K} \times \mathbb{K} \to \mathbb{K}$

- $\oplus$: *addition* (*disjunction*)
- $\otimes$: *multiplication* (*conjunction*)

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

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

- $\alpha_0 = \{\text{Alice}, \text{Bob}\} = AB$
- $\alpha_1 = \{\text{Alice}, \text{Charlie}\} = AC$
- $\beta_0 = \{\text{Bob}\} = B$

		$\mathbb{K}$	$\mathcal{P}(\mathcal{U})$
a	2	$(\alpha_0 \oplus \alpha_1) \otimes \beta_0$	$B = (AB \cup AC) \cap B$
a	1	α_0	AB

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

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

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

- $\alpha_0 = \{\text{Alice}, \text{Bob}\} = AB$
- $\alpha_1 = \{\text{Alice}, \text{Charlie}\} = AC$
- $\beta_0 = \{\text{Bob}\} = B$

		$\mathbb{K}$	$\mathcal{P}(\mathcal{U})$
a	2	$(\alpha_0 \oplus \alpha_1) \otimes \beta_0$	$B = (AB \cup AC) \cap B$
a	1	α_0	AB

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

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 $f(c)(k) = c \in k$

A	C		f(Alice)	f(Bob)	f(Charlie)
a	2	B	$\bot$	$\top$	$\bot$
a	1	AB	$\top$	$\top$	$\bot$

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

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 $f(c)(k) = c \in k$

A	C		f(Alice)	f(Bob)	f(Charlie)
a	2	B	$\bot$	$\top$	$\bot$
a	1	AB	$\top$	$\top$	$\bot$

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

TBAC – Filtering (2/2)

Flow policy

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

- $f(c)(0) = \bot$
  nobody can read $(t : 0)$
- $f(c)(1) = \top$
  anybody can read $(t : 1)$
- $f(c)(a \oplus b) = f(c)(a) \vee f(c)(b)$
  one can read $(t : a \oplus b)$ *if he/she can read either a or b*
- $f(c)(a \otimes b) = f(x)(a) \wedge f(x)(b)$
  one can read $(t : a \otimes b)$ *if he/she can read both a and b*

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

TBAC – Filtering (2/2)

Flow policy

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

- $f(c)(0) = \bot$
  nobody can read $(t : 0)$
- $f(c)(1) = \top$
  anybody can read $(t : 1)$
- $f(c)(a \oplus b) = f(c)(a) \vee f(c)(b)$
  one can read $(t : a \oplus b)$ *if he/she can read either a or b*
- $f(c)(a \otimes b) = f(x)(a) \wedge f(x)(b)$
  one can read $(t : a \otimes b)$ *if he/she can read both a and b*

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

TBAC – Filtering (2/2)

Flow policy

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

- $f(c)(0) = \bot$
  nobody can read $(t : 0)$
- $f(c)(1) = \top$
  anybody can read $(t : 1)$
- $f(c)(a \oplus b) = f(c)(a) \vee f(c)(b)$
  one can read $(t : a \oplus b)$ *if he/she can read either a or b*
- $f(c)(a \otimes b) = f(x)(a) \wedge f(x)(b)$
  one can read $(t : a \otimes b)$ *if he/she can read both a and b*

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

TBAC – Filtering (2/2)

Flow policy

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

- $f(c)(0) = \bot$
  nobody can read $(t : 0)$
- $f(c)(1) = \top$
  anybody can read $(t : 1)$
- $f(c)(a \oplus b) = f(c)(a) \vee f(c)(b)$
  one can read $(t : a \oplus b)$ *if he/she can read either a or b*
- $f(c)(a \otimes b) = f(x)(a) \wedge f(x)(b)$
  one can read $(t : a \otimes b)$ *if he/she can read both a and b*

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

TBAC – Filtering (2/2)

Flow policy

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

- $f(c)(0) = \bot$
  nobody can read $(t : 0)$
- $f(c)(1) = \top$
  anybody can read $(t : 1)$
- $f(c)(a \oplus b) = f(c)(a) \vee f(c)(b)$
  one can read $(t : a \oplus b)$ *if he/she can read either a or b*
- $f(c)(a \otimes b) = f(x)(a) \wedge f(x)(b)$
  one can read $(t : a \otimes b)$ *if he/she can read both a and b*

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

TBAC – Filtering (2/2)

Flow policy

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

- $f(c)(0) = \bot$
  nobody can read $(t : 0)$
- $f(c)(1) = \top$
  anybody can read $(t : 1)$
- $f(c)(a \oplus b) = f(c)(a) \vee f(c)(b)$
  one can read $(t : a \oplus b)$ *if he/she can read either a or b*
- $f(c)(a \otimes b) = f(x)(a) \wedge f(x)(b)$
  one can read $(t : a \otimes b)$ *if he/she can read both a and b*

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

Conclusion

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

Extensions

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

Drawbacks

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

Thank you for your attention!