Relational Similarity-Based Databases II

Vilem Vychodil (Palacky University, Olomouc)

DATA ANALYSIS AND MODELING LAB
Palacky University, Olomouc, Czech Republic

INVESTMENTS IN EDUCATION DEVELOPMENT

Preliminaries

utilized notions (part I):

- attributes, relation scheme
- tuple, cartesian product
- domain, similarity
- ranked data table (over domains with similarities)

in case you missed it: goto DAMOL webpage

Belohlavek, Vychodil: *Relational Similarity-Based Databases I*

new notion: object constants = *names for individual values that can appear in a database*

- $\mathfrak{C}$: denumerable set of all object constants
- object constants denoted by $\mathfrak{c}, \mathfrak{d}, \mathfrak{c}_1, \mathfrak{c}_2, \ldots$
- note the difference: name (syntactic notion) $\neq$ value (semantic notion)

Type Declarations and Types

motivation: The need to distinguish between different *types* of attributes (to avoid mixing apples and pears) on the syntactic level.

Definition (types, type declarations)

A **type declaration** for a set Y of attributes and a set $\mathfrak{C}$ of object constants is a structure $\mathbf{\Lambda} = \langle \Lambda, \lambda \rangle$, where Λ is a nonempty set of elements called **types**, and λ is a map $\lambda\colon Y \cup \mathfrak{C} \to \Lambda$ such that for each $\tau \in \Lambda$ there is an infinite $Y' \subseteq Y$ such that $\lambda(Y') = \{\tau\}$.

notes:

- λ in $\mathbf{\Lambda}$ assigns to each attribute and object constant z its type $\lambda(z)$
- $\lambda(y) = \lambda(z)$ means: *y and z have the same type* (under $\mathbf{\Lambda} = \langle \Lambda, \lambda \rangle$)
- for each type there are infinitely many attributes of that type (Λ is thus at most a denumerable set)

can be extended: hierarchy of types (not discussed here)

Types vs. Domains

domains = sets of all possible values of attributes

- D_y is domain of attribute $y \in Y$;
- nonempty (at most) denumerable set of all possible values y

type $\times$ domain

- type = syntactic notion (a name),
- domain = semantic notion (a set of permissible values)

our approach:

"Domain is an interpretation of a type."

notes: different approaches to domains vs. types:

- E. F. Codd: notion of a domain
- C. J. Date: domains = types (type is a synonym for a domain)

Consistency Between Types and Domains

motivation:

- The distinction between types and domains has its benefits
  (various issues can be resolved syntactically based on types and their mutual relationship without considering concrete sets of values)
- there is a need to keep "types and domains in sync"

Definition (domains with similarities respecting type declaration)

Set $\{\langle D_y, \approx_y \rangle \mid y \in Y\}$ of domains with similarities **respects** type declaration $\mathbf{\Lambda} = \langle \Lambda, \lambda \rangle$ if for all $y_1, y_2 \in Y$, the following condition is satisfied: If $\lambda(y_1) = \lambda(y_2)$, then $\langle D_{y_1}, \approx_{y_1} \rangle$ coincides with $\langle D_{y_2}, \approx_{y_2} \rangle$.

note:

- for $\lambda(y_1) \neq \lambda(y_2)$, we can have $D_{y_1} \cap D_{y_2} \neq \emptyset$

Database Scheme

motivation:

- relational queries involve (sets of) RDTs denoted by names;
- in order to formalize the concept, we introduce database schemes

Definition (database scheme, relation symbols)

A **database scheme on** Y is any couple $\langle \mathbb{R}, \varrho \rangle$ where $\mathbb{R}$ is a finite nonempty set of **relation symbols** and ϱ is a map $\varrho\colon \mathbb{R} \to 2^Y$ assigning to each relation symbol $\mathbb{r} \in \mathbb{R}$ a relation scheme $\varrho(\mathbb{r}) \subseteq Y$.

In terminology of Date, relation symbols correspond to RELVARs (relation variables):

- relation symbol is a *name* that can be *bound* to an RDT
- $\varrho(\mathbb{r})$ prescribes a relation scheme of an RDT that can be bound to $\mathbb{r}$
- RDTs are bound to relation symbols in *database instances*, ...

Database Instance

Definition (database instance)

A **database instance $\mathcal{D}$** (shortly, an **instance**) of a database scheme $\langle \mathbb{R}, \varrho \rangle$ and object constants $\mathfrak{C}$ over domains $\{\langle D_y, \approx_y \rangle \mid y \in Y\}$ with similarities is a triplet $\mathcal{D} = \langle U^{\mathcal{D}}, \mathbb{R}^{\mathcal{D}}, \mathfrak{C}^{\mathcal{D}} \rangle$ where

1. $U^{\mathcal{D}} = \{\langle D_y, \approx_y \rangle \mid y \in Y\}$ is a set of domains with similarities;
2. $\mathbb{R}^{\mathcal{D}}$ is a set of RDTs such that for each relation symbol $\mathbb{r} \in \mathbb{R}$ there is an RDT $\mathbb{r}^{\mathcal{D}} \in \mathbb{R}^{\mathcal{D}}$ on $\varrho(\mathbb{r})$ over $U^{\mathcal{D}}$;
3. $\mathfrak{C}^{\mathcal{D}} = \{\mathfrak{c}^{\mathcal{D}} \in \bigcup_{y \in Y} D_y \mid \mathfrak{c} \in \mathfrak{C}\}$ is a set of values interpreting object constants.

If $\mathfrak{C}$ is clear from the context, we say that $\mathcal{D}$ is an instance of $\langle \mathbb{R}, \varrho \rangle$. Moreover, we say that $\mathcal{D} = \langle U^{\mathcal{D}}, \mathbb{R}^{\mathcal{D}}, \mathfrak{C}^{\mathcal{D}} \rangle$ **respects type declaration** $\mathbf{\Lambda} = \langle \Lambda, \lambda \rangle$ if $U^{\mathcal{D}}$ respects $\mathbf{\Lambda}$ and if for each $\mathfrak{c} \in \mathfrak{C}$, $\mathfrak{c}^{\mathcal{D}} \in D_y$ whenever $\lambda(\mathfrak{c}) = \lambda(y)$.

notes on database schemes $\times$ database instances:

- $\langle \mathbb{R}, \varrho \rangle$ and $\mathfrak{C}$ determine a *predicate language* (syntactic notions)
- $\mathcal{D} = \langle U^{\mathcal{D}}, \mathbb{R}^{\mathcal{D}}, \mathfrak{C}^{\mathcal{D}} \rangle$ is a first-order *predicate structure* (semantic notion)

A General View on Relational Queries

general approach to queries:

Definition (query)

A *query* on database scheme $\langle \mathbb{R}, \varrho \rangle$ and object constants $\mathfrak{C}$ is any partial recursive function from the set of all database instances of $\langle \mathbb{R}, \varrho \rangle$ and $\mathfrak{C}$ to the set of all RDTs.

typically:

- queries are induced by evaluating expressions or formulas in database instances

two (basic) types of query systems in our model:

1. queries are induces by evaluating *relation algebra expressions* ...
2. queries are induces by evaluating *formulas of domain relational calculus* ...

in database instances

Relation Algebra

Definition (Relation Algebra Expressions, RA-expressions)

Let $\langle \mathbb{R}, \varrho \rangle$ be a database scheme on Y and let $\mathbf{\Lambda} = \langle \Lambda, \lambda \rangle$ be a type declaration for Y and $\mathfrak{C}$. Then, **relational algebra expressions over** $\langle \mathbb{R}, \varrho \rangle$ (shortly, **RA-expressions**) are defined as follows:

1. If $\mathbb{r} \in \mathbb{R}$, then $\mathbb{r}$ is RA-expression on $\varrho(\mathbb{r})$;
2. if $a \in L$, then $\overline{a}_{\emptyset}$ is RA-expression on $\emptyset$;
3. if $\mathfrak{c} \in \mathfrak{C}$, $y \in Y$, and $\lambda(\mathfrak{c}) = \lambda(y)$, then $[y{:}\mathfrak{c}]$ is RA-expression on $\{y\}$;
4. if E_1 and E_2 are RA-expressions on R,
   then $(E_1 \cap E_2)$ and $(E_1 \cup E_2)$ are RA-expressions on R,
5. if E_1, E_2, and E_3 are RA-expressions on R, then $(E_1 \twoheadrightarrow^{E_3} E_2)$ is RA-expression on R;
6. if E_1 is RA-expression on R_1 and E_2 is RA-expression on R_2,
   then $(E_1 \bowtie E_2)$ is RA-expression on $R_1 \cup R_2$;
   $\vdots$

Relation Algebra

Definition (... continuation)

⋮

7. if E is RA-expression on T and $R \subseteq T$, then $\pi_R(E)$ is RA-expression on R;
8. if E_1 is RA-expression on R, E_2 is RA-expression on $S \subseteq R$, and E_3 is RA-expression on $T = R \setminus S$, then $(E_1 \div^{E_3} E_2)$ is RA-expression on T;
9. if E is RA-expression on R, $y \in R$, and $z \in R \cup \mathfrak{C}$ such that $\lambda(y) = \lambda(z)$, then $\sigma_{y \approx z}(E)$ is RA-expression on R;
10. if E is RA-expression on R, then ΔE and ∇E are RA-expressions on R;
11. if E is RA-expression on R, and $h\colon R \to Y$ is an injective map such that for each $y \in R$ we have $\lambda(h(y)) = \lambda(y)$, then $\rho_h(E)$ is RA-expression on $h(R)$.

All RA-expressions result by application of 1–11. In addition, if E is RA-expression on R, we call R the **relation scheme of** E and denote it by $\mathrm{sch}(E)$.

Definition (Results of RA-expressions in Database Instances)

Let $\mathcal{D} = \langle U^{\mathcal{D}}, \mathbb{R}^{\mathcal{D}}, \mathfrak{C}^{\mathcal{D}} \rangle$ be an instance of database scheme $\langle \mathbb{R}, \varrho \rangle$ which respects a type declaration $\mathbf{\Lambda} = \langle \Lambda, \lambda \rangle$ for Y and $\mathfrak{C}$. Then, for each RA-expression E over $\langle \mathbb{R}, \varrho \rangle$, we define an RDT $E^{\mathcal{D}}$, called the **result of E in $\mathcal{D}$** as follows:

1. If E is $\mathbb{r} \in \mathbb{R}$, then $E^{\mathcal{D}} = \mathbb{r}^{\mathcal{D}}$;
2. if E is $\overline{a}_{\emptyset}$, then $E^{\mathcal{D}} = a_{\emptyset}$;
3. if E is $[y{:}\mathfrak{c}]$, then $E^{\mathcal{D}}$ is $[y{:}\mathfrak{c}^{\mathcal{D}}]$;
4. if E is $E_1 \cap E_2$, then $E^{\mathcal{D}} = E_1^{\mathcal{D}} \cap E_2^{\mathcal{D}}$; if E is $E_1 \cup E_2$, then $E^{\mathcal{D}} = E_1^{\mathcal{D}} \cup E_2^{\mathcal{D}}$;
5. if E is $E_1 \twoheadrightarrow^{E_3} E_2$, then $E^{\mathcal{D}} = E_1^{\mathcal{D}} \twoheadrightarrow^{E_3^{\mathcal{D}}} E_2^{\mathcal{D}}$;
6. if E is $E_1 \bowtie E_2$, then $E^{\mathcal{D}} = E_1^{\mathcal{D}} \bowtie E_2^{\mathcal{D}}$;
7. if E is $\pi_R(F)$, then $E^{\mathcal{D}} = \pi_R(F^{\mathcal{D}})$;
8. if E is $E_1 \div^{E_3} E_2$, then $E^{\mathcal{D}} = E_1^{\mathcal{D}} \div^{E_3^{\mathcal{D}}} E_2^{\mathcal{D}}$;
9. if E is $\sigma_{y \approx z}(F)$ and $z \in Y$, then $E^{\mathcal{D}} = \sigma_{y \approx z}(F^{\mathcal{D}})$; if $z \in \mathfrak{C}$, then $E^{\mathcal{D}} = \sigma_{y \approx z^{\mathcal{D}}}(F^{\mathcal{D}})$;
10. if E is ΔF, then $E^{\mathcal{D}} = \Delta F^{\mathcal{D}}$; if E is ∇F, then $E^{\mathcal{D}} = \nabla F^{\mathcal{D}}$;
11. if E is $\rho_h(F)$, then $E^{\mathcal{D}} = \rho_h(F^{\mathcal{D}})$.

Example (Relation Algebra Expressions)

$\mathbb{r}$

$\sigma_{y \approx_{\mathfrak{c}}}(\mathbb{r})$

$0.5_{\emptyset} \bowtie \sigma_{y \approx_{\mathfrak{c}}}(\mathbb{r})$

$\pi_{\{x,y\}}(0.5_{\emptyset} \bowtie \sigma_{y \approx_{\mathfrak{c}}}(\mathbb{r}))$

$\pi_{\{x,y\}}(0.5_{\emptyset} \bowtie \sigma_{y \approx_{\mathfrak{c}}}(\mathbb{r})) \bowtie \mathbb{s}$

$\pi_{\{x,y\}}(0.5_{\emptyset} \bowtie \sigma_{y \approx_{\mathfrak{c}}}(\mathbb{r})) \bowtie \Delta\mathbb{s}$

$\sigma_{x \approx z}(\pi_{\{x,y\}}(0.5_{\emptyset} \bowtie \sigma_{y \approx_{\mathfrak{c}}}(\mathbb{r})) \bowtie \Delta\mathbb{s})$

$\vdots$

Notes on Integrity Constraints

classic RM:

- $E \subseteq F$ (with E, F being RA-expressions)
- functional dependencies, keys, referential integrity constraints ...

similarity-based integrity constrains:

- threshold-based: $S(E, F) \geq a$
  - $S(E, F) \geq a$ is **valid** in $\mathcal{D}$ if $S(E^{\mathcal{D}}, F^{\mathcal{D}}) \geq a$
- graded approach $S(E, F)$
  - $S(E^{\mathcal{D}}, F^{\mathcal{D}})$ is a **degree of validity** of $S(E, F)$ in $\mathcal{D}$

Example (similarity-based (soft) constraints)

For Q (high-quality wines), E (expensive wines),

$$S\big(\pi_{\{\mathit{wine\text{-}id}\}}(Q), \pi_{\{\mathit{wine\text{-}id}\}}(E)\big) \geq 0.95$$

means "high-quality wines are expensive at least to degree 0.95".

Domain Relational Calculus: The Basic Idea

object variables = *names for (possibly changing) values in database instances*

- X: denumerable set of all object variables
- (object) variables from X are denoted by $\mathbb{x}, \mathbb{x}', \mathbb{x}_1, \mathbb{x}_2, \ldots$
- object constrant (has a fixed value in $\boldsymbol{\mathcal{D}}$) $\times$
  object variable (values not determined solely by $\boldsymbol{\mathcal{D}}$)

from formulas to queries:

- take a suitable first-order formula φ, and evaluate φ in $\boldsymbol{\mathcal{D}}$
- note: the value of φ in $\boldsymbol{\mathcal{D}}$ depends on values of *free variables*
- if the free variables are bound to y-values of a tuple t, then the value of φ in $\boldsymbol{\mathcal{D}}$ (under the valuation of free variables) is the *degree to which φ is true in $\boldsymbol{\mathcal{D}}$* are can be seen as a rank of t.
- collect all tuples with nonzero ranks $\Longmapsto$ result
- issues: domain independence

Range Component, Range Declaration

requirement: finiteness of query results

Definition (range component, range declaration)

A **range component** (over $\langle \mathbb{R}, \varrho \rangle$ and $\mathfrak{C}$) is (i) any object constant from $\mathfrak{C}$ and (ii) any expression $\mathfrak{r}(y)$, where $\mathfrak{r} \in \mathbb{R}$, and $y \in \varrho(\mathfrak{r})$. A **range declaration** (over $\langle \mathbb{R}, \varrho \rangle$ and $\mathfrak{C}$) is any finite set R of range components.

Definition (type compatibility of range declarations)

A range declaration R is **compatible** with type declaration $\mathbf{\Lambda} = \langle \Lambda, \lambda \rangle$ if all attributes and constants appearing in R have the same type $\tau \in \Lambda$ which is called the **type of** R and denoted $\lambda(\mathsf{R})$. The set of all range declarations over $\langle \mathbb{R}, \varrho \rangle$ and $\mathfrak{C}$ compatible with $\mathbf{\Lambda}$ is denoted by $\mathrm{Rd}(\mathbb{R}, \varrho, \mathfrak{C}, \mathbf{\Lambda})$. Any map $rd\colon X \to \mathrm{Rd}(\mathbb{R}, \varrho, \mathfrak{C}, \mathbf{\Lambda})$ is called a **range declaration for variables from X.**

note: value $rd(\mathfrak{x})$ is called a range declaration for $\mathfrak{x}$

Values of Range Declarations in Database Instances

syntax $\times$ semantics:

- range declaration (syntactic notion),
- value of range declaration (semantic notion) introduced as follows:

Definition (value of range declaration)

Let $\mathcal{D} = \langle U^{\mathcal{D}}, \mathbb{R}^{\mathcal{D}}, \mathfrak{C}^{\mathcal{D}} \rangle$ be an instance of database scheme $\langle \mathbb{R}, \varrho \rangle$ and let $\mathsf{R} \in \mathrm{Rd}(\mathbb{R}, \varrho, \mathfrak{C}, \mathbf{\Lambda})$ be a range declaration. The following set

$$\mathsf{R}^{\mathcal{D}} = \{\mathfrak{c}^{\mathcal{D}} \mid \mathfrak{c} \in \mathsf{R}\} \cup \{d \in D_y \mid \text{if } \mathbb{r}(y) \in \mathsf{R} \text{ and } \pi_{\{y\}}(\mathbb{r}^{\mathcal{D}})(\{\langle y, d \rangle\}) > 0\}$$

is called the **value of** R **in database instance** $\mathcal{D}$**.**

meaning:

- in $\mathcal{D}$, free variable $\mathbb{x}$ can take values only from $rd(\mathbb{x})^{\mathcal{D}}$
- ranges of bound variables are specified in quantifiers ...

Formulas

Definition (DRC-RD formulas)

Let $\langle \mathbb{R}, \varrho \rangle$ be a database scheme on Y, let $\mathbf{\Lambda}$ be a type declaration for Y and $\mathfrak{C}$, and let X be a set of object variables. Then, **formulas of the domain relational calculus with range declarations** (shortly, **DRC-RD formulas** or **formulas**) **over $\langle \mathbb{R}, \varrho \rangle$, $\mathfrak{C}$, and $\mathbf{L}$** are defined as follows:

1. If $\mathbb{r} \in \mathbb{R}$ is a relation symbol such that $\varrho(\mathbb{r}) = \{y_1, \ldots, y_n\}$ and $\mathbb{x}_1, \ldots, \mathbb{x}_n \in X$ are arbitrary variables, then $\mathbb{r}(y_1{:}\mathbb{x}_1, \ldots, y_n{:}\mathbb{x}_n)$ is formula;
2. if $a \in L$, then $\overline{a}$ is formula;
3. if $\mathbb{x} \in X$ and $z \in X \cup \mathfrak{C}$, then $\mathbb{x} \approx z$ is formula;
4. if φ and ψ are formulas then $(\varphi \otimes \psi)$, $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $(\varphi \Rightarrow \psi)$ are formulas;
5. if φ is formula, then $\triangle\varphi$ is formula;
6. if φ is formula, $\mathbb{x} \in X$, and $\mathrm{R} \in \mathrm{Rd}(\mathbb{R}, \varrho, \mathfrak{C}, \mathbf{\Lambda})$, then $(\forall \mathbb{x} {\in} \mathrm{R})\varphi$ and $(\exists \mathbb{x} {\in} \mathrm{R})\varphi$ are formulas and φ is called the scope of the quantifiers $(\forall \mathbb{x} {\in} \mathrm{R})$ and $(\exists \mathbb{x} {\in} \mathrm{R})$, respectively.

All formulas result by application of 1–6.

Notes on Formulas

unordered nature of attributes:

- $\mathbb{r}(y_1{:}\mathbb{x}_1, y_2{:}\mathbb{x}_2)$ is (syntactically) equivalent to $\mathbb{r}(y_2{:}\mathbb{x}_2, y_1{:}\mathbb{x}_1)$

we adopt the usual notions: *subformula*, *free* and *bound* occurrences of variables

Example (natural reading of formulas)

Formulas are read the usual way.

For instance,

- $\mathbb{r}(y_1{:}\mathbb{x}, y_2{:}\mathbb{z}) \otimes \mathbb{s}(y_1{:}\mathbb{x}, y_2{:}\mathbb{z})$ reads "the value of y_1 denoted by $\mathbb{x}$ and the value of y_2 denoted by $\mathbb{z}$ are related according to $\mathbb{r}$ *and* the value of y_1 denoted by $\mathbb{x}$ and the value of y_2 denoted by $\mathbb{z}$ are related according to $\mathbb{s}$".
- $(\exists \mathbb{x} \in \mathrm{R})\mathbb{r}(y_1{:}\mathbb{x}, y_2{:}\mathbb{z})$ reads "*there is* a value of y_1 denoted by $\mathbb{x}$ (which ranges over R) which is according to $\mathbb{r}$ related to the value of y_2 denoted by $\mathbb{z}$", ...

Formulas and Type Safety

motivation:

- define formulas which together with range declarations for (free) variables "make sense" from the point of view of the type system; (e.g., $\mathbb{r}(y{:}\mathbb{x})$ makes sense only if the type of y and the range declaration for $\mathbb{x}$ is the same)

Definition (type safety)

Formula φ is called **safe with respect to** $rd\colon X \to \mathrm{Rd}(\mathbb{R}, \varrho, \mathfrak{C}, \mathbf{\Lambda})$ if the following conditions hold for any variable $\mathbb{x}$ that occurs in φ:

1. If the occurrence of $\mathbb{x}$ in a subformula ψ of φ is free in φ and if
   - ψ is $\mathbb{r}(y{:}\mathbb{x}, \dots)$, then $\lambda(y) = \lambda(rd(\mathbb{x}))$; or if
   - ψ is $\mathbb{x} \approx \mathfrak{c}$, then $\lambda(\mathfrak{c}) = \lambda(rd(\mathbb{x}))$; or if
   - ψ is $\mathbb{x} \approx \mathbb{y}$ and this occurrence of $\mathbb{y}$ is free in φ, then $\lambda(rd(\mathbb{x})) = \lambda(rd(\mathbb{y}))$.
2. For any subformula $(Q\mathbb{x}{\in}\mathsf{R})\psi$ of φ with Q being $\forall$ or $\exists$, the formula ψ is safe with respect to $rd'\colon X \to \mathrm{Rd}(\mathbb{R}, \varrho, \mathfrak{C}, \mathbf{\Lambda})$, where $rd'(\mathbb{y}) = rd(\mathbb{y})$ for each $\mathbb{y} \neq \mathbb{x}$, and $rd'(\mathbb{x}) = \mathsf{R}$.

Valuation of Object Variables

values of objects variables $\times$ objects constants:

- *object constants* are assigned values by database instances,
- (*free*) *object variables* are assigned values by valuations (as in predicate logic),
- specific feature: we only allow values from values of range declarations

Definition (valuation, values of object variables)

Let $\boldsymbol{\mathcal{D}} = \langle U^{\boldsymbol{\mathcal{D}}}, \mathbb{R}^{\boldsymbol{\mathcal{D}}}, \mathfrak{C}^{\boldsymbol{\mathcal{D}}} \rangle$ be a database instance of $\langle \mathbb{R}, \varrho \rangle$ where $U^{\boldsymbol{\mathcal{D}}} = \{\langle D_y, \approx_y \rangle \mid y \in Y\}$ and let $rd\colon X \to \mathrm{Rd}(\mathbb{R}, \varrho, \mathfrak{C}, \boldsymbol{\Lambda})$ be a range declaration for variables. A map $\boldsymbol{v}\colon X \to \bigcup_{y \in Y} D_y$ such that $\boldsymbol{v}(\mathbb{x}) \in rd(\mathbb{x})^{\boldsymbol{\mathcal{D}}}$ is called a **$\boldsymbol{\mathcal{D}}$-valuation** (of variables from X with respect to rd) and $\boldsymbol{v}(\mathbb{x})$ is called the **value of $\mathbb{x}$ under $\boldsymbol{v}$.**

notation:

- for two $\boldsymbol{\mathcal{D}}$-valuations $\boldsymbol{w}$ and $\boldsymbol{v}$ we write $\boldsymbol{w} =_{\mathbb{x}} \boldsymbol{v}$ whenever $\boldsymbol{w}(\mathbb{x}') = \boldsymbol{v}(\mathbb{x}')$ for each $\mathbb{x}' \in X$ different from $\mathbb{x}$

Interpretation of Formulas

Definition (interpretation of formulas)

Let $\boldsymbol{\mathcal{D}} = \langle U^{\boldsymbol{\mathcal{D}}}, \mathbb{R}^{\boldsymbol{\mathcal{D}}}, \mathfrak{C}^{\boldsymbol{\mathcal{D}}} \rangle$ be an instance of database scheme $\langle \mathbb{R}, \varrho \rangle$ which respects a type declaration $\boldsymbol{\Lambda} = \langle \Lambda, \lambda \rangle$ for Y and $\mathfrak{C}$, let $\boldsymbol{v}$ be a $\boldsymbol{\mathcal{D}}$-valuation, and let φ be a formula which is safe with respect to range declaration $rd\colon X \to \mathrm{Rd}(\mathbb{R}, \varrho, \mathfrak{C}, \boldsymbol{\Lambda})$ for variables from X. Then, we define a degree $||\varphi||^{rd}_{\boldsymbol{\mathcal{D}},\boldsymbol{v}} \in L$, called the **degree to which φ is true in $\mathcal{D}$ under $\boldsymbol{v}$ and rd** as follows:

1. If φ is $\mathbb{r}(y_1{:}\mathbb{x}_1, \ldots, y_n{:}\mathbb{x}_n)$, then $||\varphi||^{rd}_{\boldsymbol{\mathcal{D}},\boldsymbol{v}} = \mathbb{r}^{\boldsymbol{\mathcal{D}}}(t)$, where t is a tuple on $\varrho(\mathbb{r})$ such that $t(y_i) = \boldsymbol{v}(\mathbb{x}_i)$ for all $i = 1, \ldots, n$;
2. if φ is $\overline{a}$, then $||\varphi||^{rd}_{\boldsymbol{\mathcal{D}},\boldsymbol{v}} = a$;
3. if φ is $\mathbb{x} \approx \mathbb{z}$, then $||\varphi||^{rd}_{\boldsymbol{\mathcal{D}},\boldsymbol{v}} = \boldsymbol{v}(\mathbb{x}) \approx_y \boldsymbol{v}(\mathbb{z})$ where $\lambda(y) = \lambda(rd(\mathbb{x}))$;
   if φ is $\mathbb{x} \approx \mathfrak{c}$, then $||\varphi||^{rd}_{\boldsymbol{\mathcal{D}},\boldsymbol{v}} = \boldsymbol{v}(\mathbb{x}) \approx_y \mathfrak{c}^{\boldsymbol{\mathcal{D}}}$ where $\lambda(y) = \lambda(rd(\mathbb{x}))$;
   $\vdots$

Interpretation of Formulas

Definition (... continuation)

$\vdots$

4. if φ is $\psi \otimes \chi$, then $||\varphi||^{rd}_{\mathcal{D},\boldsymbol{v}} = ||\psi||^{rd}_{\mathcal{D},\boldsymbol{v}} \otimes ||\chi||^{rd}_{\mathcal{D},\boldsymbol{v}}$;
   if φ is $\psi \wedge \chi$, then $||\varphi||^{rd}_{\mathcal{D},\boldsymbol{v}} = ||\psi||^{rd}_{\mathcal{D},\boldsymbol{v}} \wedge ||\chi||^{rd}_{\mathcal{D},\boldsymbol{v}}$;
   if φ is $\psi \vee \chi$, then $||\varphi||^{rd}_{\mathcal{D},\boldsymbol{v}} = ||\psi||^{rd}_{\mathcal{D},\boldsymbol{v}} \vee ||\chi||^{rd}_{\mathcal{D},\boldsymbol{v}}$;
   if φ is $\psi \Rightarrow \chi$, then $||\varphi||^{rd}_{\mathcal{D},\boldsymbol{v}} = ||\psi||^{rd}_{\mathcal{D},\boldsymbol{v}} \to ||\chi||^{rd}_{\mathcal{D},\boldsymbol{v}}$;
5. if φ is $\triangle\psi$, then $||\varphi||^{rd}_{\mathcal{D},\boldsymbol{v}} = \begin{cases} 1, & ||\psi||^{rd}_{\mathcal{D},\boldsymbol{v}} = 1, \\ 0, & \text{otherwise;} \end{cases}$
6. if φ is $(\forall \mathbb{x} \in \mathrm{R})\psi$ then $||\varphi||^{rd}_{\mathcal{D},\boldsymbol{v}} = \bigwedge\{||\psi||^{rd'}_{\mathcal{D},\boldsymbol{w}} \mid \boldsymbol{w} =_{\mathbb{x}} \boldsymbol{v} \text{ and } \boldsymbol{w}(\mathbb{x}) \in \mathrm{R}^{\mathcal{D}}\}$;
   if φ is $(\exists \mathbb{x} \in \mathrm{R})\psi$ then $||\varphi||^{rd}_{\mathcal{D},\boldsymbol{v}} = \bigvee\{||\psi||^{rd'}_{\mathcal{D},\boldsymbol{w}} \mid \boldsymbol{w} =_{\mathbb{x}} \boldsymbol{v} \text{ and } \boldsymbol{w}(\mathbb{x}) \in \mathrm{R}^{\mathcal{D}}\}$;
   where in both cases $rd' : X \to \mathrm{Rd}(\mathbb{R}, \varrho, \mathfrak{C}, \boldsymbol{\Lambda})$ is a range declaration for variables such that $rd'(\mathbb{y}) = rd(\mathbb{y})$ for each $\mathbb{y} \neq \mathbb{x}$, and $rd'(\mathbb{x}) = \mathrm{R}$.

Results of Queries in DRC-RD

Definition (target component, target set, query result)

Let $\boldsymbol{\mathcal{D}}$ be an instance of $\langle \mathbb{R}, \varrho \rangle$ which respects type declaration $\boldsymbol{\Lambda} = \langle \Lambda, \lambda \rangle$ for Y and $\mathfrak{C}$ and let φ be a formula which is safe with respect to range declaration $rd\colon X \to \mathrm{Rd}(\mathbb{R}, \varrho, \mathfrak{C}, \boldsymbol{\Lambda})$ for variables from X.

A **target component** (based on rd) is any expression $y{:}\mathbb{x}$ such that $y \in Y$, $\mathbb{x} \in X$, and $\lambda(y) = \lambda(rd(\mathbb{x}))$. A **target set** (based on rd)

$$\mathcal{T} = \{y_1{:}\mathbb{x}_1, \ldots, y_n{:}\mathbb{x}_n\}$$

is any finite set of target components (all $y_1, \ldots, y_n$ and $\mathbb{x}_1, \ldots, \mathbb{x}_n$ pairwise distinct).

For $\mathcal{T}$ and φ, we define an RDT $\mathcal{T}\varphi^{rd,\boldsymbol{\mathcal{D}}}$ on the relation scheme $R = \{y_1, \ldots, y_n\}$, called the **result of query $\mathcal{T}\varphi$ in $\boldsymbol{\mathcal{D}}$ under rd** as follows:

$$\left(\mathcal{T}\varphi^{rd,\boldsymbol{\mathcal{D}}}\right)(r) = \begin{cases} \bigvee\{||\varphi||^{rd}_{\boldsymbol{\mathcal{D}},\boldsymbol{v}} \mid \text{for each } i\colon \boldsymbol{v}(\mathbb{x}_i) = r(y_i)\}, & \text{if } r \in \prod_{y_i \in R} rd(\mathbb{x}_i)^{\boldsymbol{\mathcal{D}}}, \\ 0, & \text{otherwise.} \end{cases}$$

Moreover, if $\boldsymbol{v}(\mathbb{x}_i) = r(y_i)$ for all $i = 1, \ldots, n$, we say that $\boldsymbol{v}$ is *induced by r and $\mathcal{T}$*.

Theorem

The following are true for each safe formula φ:

1. *If for each $\mathbb{x}$ which is free in φ, we have $\boldsymbol{v}(\mathbb{x}) = \boldsymbol{w}(\mathbb{x})$, then $||\varphi||^{rd}_{\mathcal{D},\boldsymbol{v}} = ||\varphi||^{rd}_{\mathcal{D},\boldsymbol{w}}$;*
2. *If for each $\mathbb{x}$ which is free in φ, we have $rd(\mathbb{x}) = rd'(\mathbb{x})$, then for any $\boldsymbol{v}$ which is a $\mathcal{D}$-valuation with respect to both rd and rd', we have $||\varphi||^{rd}_{\mathcal{D},\boldsymbol{v}} = ||\varphi||^{rd'}_{\mathcal{D},\boldsymbol{v}}$;*
3. *For each target set $\mathcal{T}$, there is safe formula ψ so that the variables in $\mathcal{T}$ are exactly the free variables in ψ and for each $\mathcal{D}$, we have $\mathcal{T}\varphi^{rd,\mathcal{D}} = \mathcal{T}\psi^{rd,\mathcal{D}}$ and*

$$\left(\mathcal{T}\psi^{rd,\mathcal{D}}\right)(r) = ||\varphi||^{rd}_{\mathcal{D},\boldsymbol{v}}$$

for all $\mathcal{D}$-valuations $\boldsymbol{v}$ induced by r and $\mathcal{T}$.

consequence: *simplified notation of queries*:

$$\{y_1{:}\mathbb{x}_1{\in}\mathsf{R}_1, \ldots, y_n{:}\mathbb{x}_n{\in}\mathsf{R}_n\}\varphi \ ,$$

meaning we consider $\{y_1{:}\mathbb{x}_1, \ldots, y_n{:}\mathbb{x}_n\}\varphi$ under a range declaration of variables $rd\colon X \to \mathrm{Rd}(\mathbb{R}, \varrho, \mathfrak{C}, \boldsymbol{\Lambda})$ such that $rd(\mathbb{x}_i) = \mathsf{R}_i$ for all $i = 1, \ldots, n$ and all free variables in φ are exactly $\mathbb{x}_1, \ldots, \mathbb{x}_n$.

Generality of Similarity-Based Databases

Theorem (on generalization of the ordinary DRC-RD)

If $\mathbf{L}$ *is a two-element Boolean algebra then each* $\mathfrak{T}\varphi^{\boldsymbol{\mathcal{D}}}$ *can be obtained as a result of a query in the ordinary DRC-RD calculus provided that all similarity relations in* $\boldsymbol{\mathcal{D}}$ *are identities.* □

Theorem (crisp case of DRC-RD)

If $\mathbf{L}$ *is arbitrary complete residuated lattice, all* $\mathbb{r}^{\boldsymbol{\mathcal{D}}}$ *in* $\boldsymbol{\mathcal{D}}$ *are nonranked, all similarities in* $\boldsymbol{\mathcal{D}}$ *are identities, and the only constants of truth degrees appearing in* φ *are* $\overline{0}$ *or* $\overline{1}$*, then* $\mathfrak{T}\varphi^{\boldsymbol{\mathcal{D}}}$ *is nonranked and a corresponding ordinary relation can be obtained as a result of a query in the ordinary DRC-RD calculus.* □

meaning: the similarity-based model (and its DRC-RD) is a (nontrivial) generalization of the RM (and DRC-RD)

Relational Completeness

Theorem (RA is more general than DRC-RD)

Let φ be a safe formula with respect to $rd\colon X \to \mathrm{Rd}(\mathbb{R}, \varrho, \mathfrak{C}, \mathbf{\Lambda})$ and let $\mathcal{T}$ be a target set based on rd. Then, there is an RA-expression E_φ over $\langle \mathbb{R}, \varrho \rangle$ such that $\mathcal{T}\varphi^{rd,\mathcal{D}} = E_\varphi^{\mathcal{D}}$ for any instance $\mathcal{D}$ of $\langle \mathbb{R}, \varrho \rangle$ which respects $\mathbf{\Lambda}$. □

Theorem (DRC-RD is more general than RA)

Let E be an RA-expression over $\langle \mathbb{R}, \varrho \rangle$. Then, there are a range declaration $rd_E\colon X \to \mathrm{Rd}(\mathbb{R}, \varrho, \mathfrak{C}, \mathbf{\Lambda})$, a formula φ_E which is safe with respect to rd_E, and a target set $\mathcal{T}_E$ based on rd_E such that $E^{\mathcal{D}} = \mathcal{T}_E\varphi_E^{rd_E,\mathcal{D}}$ holds true for any instance $\mathcal{D}$ of $\langle \mathbb{R}, \varrho \rangle$ which respects $\mathbf{\Lambda}$. □

details in:

Belohlavek, R. and Vychodil, V.

Relational Similarity-Based Databases, Part I: Query Systems (in preparation)

Derived Operations: Similarity-Based θ-restrictions

idea: for any RA-expression E and safe formula θ whose free variables are in a correspondence with the attributes from $\mathrm{sch}(E)$, we can introduce a (**similarity-based**) **θ-restriction** $\sigma_\theta(E^{\mathcal{D}})$ of RDT $E^{\mathcal{D}}$ by θ as follows:

$$\sigma_\theta(E^{\mathcal{D}}) = \mathfrak{T}_E(\varphi_E \otimes \theta)^{rd_E, \mathcal{D}},$$

where φ_E, rd_E, and $\mathfrak{T}_E$ correspond to E

generalization of similarity-based restrictions: take $\mathfrak{y} \approx \mathfrak{c}$ for θ

Example (threshold similarity-based restriction)

Example of a derived restriction: take $\overline{a} \Rightarrow \mathfrak{y} \approx \mathfrak{c}$ for θ, i.e.

$$\big(\sigma_{\overline{a} \Rightarrow \mathfrak{y} \approx \mathfrak{c}}(\mathcal{D})\big)(r) = \mathcal{D}(r) \otimes (a \to r(y) \approx_y \mathfrak{c}^{\mathcal{D}})$$

for all $r \in \mathrm{Tupl}(R)$.

Derived Operations: Similarity-Based Semijoins and Closures

tuple similarity: for any tuples $r_1, r_2 \in \mathrm{Tupl}(R)$, we put

$$r_1 \approx_R r_2 = \bigwedge_{y \in R} r_1(y) \approx_y r_2(y)$$

and call $r_1 \approx_R r_2$ the **degree to which r_1 and r_2 are similar**

similarity-based semijoins: for any RDTs $\mathcal{D}_1$ and $\mathcal{D}_2$ on $R \cup S$ and $S \cup T$ such that $R \cap S = R \cap T = S \cap T = \emptyset$, define a (**natural**) **similarity-based semijoin** $\mathcal{D}_1 \ltimes^{\approx} \mathcal{D}_2$ of $\mathcal{D}_1$ and $\mathcal{D}_2$ (in this order) by putting

$$\big(\mathcal{D}_1 \ltimes^{\approx} \mathcal{D}_2\big)(rs) = \mathcal{D}_1(rs) \otimes \bigvee_{t \in \mathrm{Tupl}(T)} \bigvee_{s' \in \mathrm{Tupl}(S)} \big(\mathcal{D}_2(s't) \otimes s' \approx_S s\big)$$

for all $r \in \mathrm{Tupl}(R)$ and $s \in \mathrm{Tupl}(S)$

similarity-based closures: particular case for $R = T = \emptyset$, non-ranked $\mathcal{D}_1$, and $\mathcal{D}_2 \subseteq \mathcal{D}_1$

$$\big(\mathcal{D}_1 \ltimes^{\approx} \mathcal{D}_2\big)(s) = \mathcal{D}_1(s) \otimes \bigvee_{s' \in \mathrm{Tupl}(S)} \big(\mathcal{D}_2(s') \otimes s' \approx_S s\big)$$

Derived Operations: Further Join-like Operations

similarity-based θ-join: using cross joins and θ-restrictions, for $\mathcal{D}_1$ and $\mathcal{D}_2$ on disjoint relation schemes, define **θ-join** by

$$\mathcal{D}_1 \bowtie_\theta \mathcal{D}_2 = \sigma_\theta(\mathcal{D}_1 \bowtie \mathcal{D}_2).$$

similarity-based join: for $\mathcal{D}_1$ and $\mathcal{D}_2$ on $R \cup S$ and $S \cup T$ such that $R \cap S = R \cap T = S \cap T = \emptyset$ and $\mathcal{D}_3$ on S, define a (**natural**) **similarity-based join** $\mathcal{D}_1 \bowtie^{\approx}_{\mathcal{D}_3} \mathcal{D}_2$ of $\mathcal{D}_1$ and $\mathcal{D}_2$ by putting

$$\big(\mathcal{D}_1 \bowtie^{\approx}_{\mathcal{D}_3} \mathcal{D}_2\big)(rst) = \mathcal{D}_3(s) \otimes \bigvee_{s',s''} \big(\mathcal{D}_1(rs') \otimes s' \approx_S s \otimes s \approx_S s'' \otimes \mathcal{D}_2(s''t)\big),$$

where s', s'' range over $\mathrm{Tupl}(S)$.

Similarity-Based Functional Dependencies

Definition (tuple similarity)

For an RDT $\mathcal{D}$ on R over $\{\langle D_y, \approx_y\rangle \mid y \in R\}$, tuples $r_1, r_2 \in \mathrm{Tupl}(R)$ and $C \in L^R$, we introduce a **degree $r_1(C) \approx_{\mathcal{D}} r_2(C)$ to which r_1 and r_2 from $\mathcal{D}$ have similar values on attributes from C** by

$$r_1(C) \approx_{\mathcal{D}} r_2(C) = (\mathcal{D}(r_1) \otimes \mathcal{D}(r_2)) \to \bigwedge_{y \in R}\big(C(y) \to r_1(y) \approx_y r_2(y)\big).$$

similarity-based functional dependencies: $A \Rightarrow B$ $(A, B \in L^R)$

Definition (semantics of SBFDs)

Let $\mathcal{D}$ be an RDT over $\{\langle D_y, \approx_y\rangle \mid y \in R\}$ and let $A, B \in L^R$. A **degree $||A \Rightarrow B||_{\mathcal{D}}$ to which $A \Rightarrow B$ is true in $\mathcal{D}$** is defined by

$$||A \Rightarrow B||_{\mathcal{D}} = \bigwedge_{r_1, r_2 \in \mathrm{Tupl}(R)}\big((r_1(A) \approx_{\mathcal{D}} r_2(A))^* \to r_1(B) \approx_{\mathcal{D}} r_2(B)\big).$$

For $A \Rightarrow B$, there is RA-expression E such that $||A \Rightarrow B||_{\mathbb{T}^{\boldsymbol{\mathcal{D}}}} = E^{\boldsymbol{\mathcal{D}}}(\emptyset)$ for any $\boldsymbol{\mathcal{D}}$.

Further Extensions

general comparators:

- other comparators than similarity
- *domains with general comparators* $\langle D_y, \approx_y, \lesssim_y, \ldots \rangle$

additional connectives:

- linguistic hedges (nontrivial for general $\mathbf{L}$, important for dependencies)
- difference-like operations (and adjoint) non-idempotent disjunctions

  $a \ominus 0 = a$, $(a \ominus b) \ominus c = (a \ominus c) \ominus b$, $\bigvee_{i \in I} a_i \ominus b = \bigvee_{i \in I} (a_i \ominus b)$

  $a \oplus b = \bigvee \{c \in L \,|\, c \ominus a \leq b\}$

additional quantifiers:

- can be done following Hájek's approach (with some adjustments)
- graded quantifiers like "many" (cf. approach by Badia)

References


Badia, A. 2009.
Quantifiers in Action. Advances in Database Systems, vol. 37.
Springer.


Belohlavek, R. 2002.
Fuzzy Relational Systems: Foundations and Principles.
Kluwer Academic Publishers, Norwell, MA, USA.


Hájek, P. 1998.
Metamathematics of Fuzzy Logic.
Kluwer Academic Publishers, Dordrecht, The Netherlands.


Lacroix, M. and Pirotte, A. 1977.
Domain-oriented relational languages.
In *VLDB*. IEEE Computer Society, 370–378.


Maier, D. 1983.
Theory of Relational Databases.
Computer Science Pr, Rockville, MD, USA.

Concluding Remarks

consequences of RA = DRC-RD:

- formal background for:
  - new derived (relational) operations
  - integrity constraints (similarity-based functional dependencies and more)
  - various types of data decomposition

further results/directions:

- dependency theory
- efficiency issues
  - algorithms for query execution
  - algorithms for reasoning wiht dependencies
- design of (implementable) query languages

details in:

Belohlavek, R. and Vychodil, V.

Relational Similarity-Based Databases, Part I: Query Systems
Relational Similarity-Based Databases, Part II: Dependencies in Data