Data dependency theory made generic — by calculation

J.N. Oliveira

Dept. Informática,
Universidade do Minho
Braga, Portugal

IFIP WG2.1 meeting #62
Dec. 2006
Namur, Belgium

Motivation

- Computer science theories are (usually) pointwise.
- What do we gain by *replaying* them in the (relational) pointfree style?

Motivation

Significant gains are known in some CS theories, eg.

- Program calculation — esp. functional, recursive programs, recall *(cata,ana,hylo,...)* -morphisms etc
- Abstract interpretation, polymorphism, unification etc

What about theories which *"everybody has heard of"*?

- Automata and transition systems
- Databases
- Parsing, compiling etc
- ...

Motivation

Significant gains are known in some CS theories, eg.

- Program calculation — esp. functional, recursive programs, recall *(cata,ana,hylo,...)* -morphisms etc
- Abstract interpretation, polymorphism, unification etc

What about theories which *"everybody has heard of"*?

- Automata and transition systems
- Databases
- Parsing, compiling etc
- ...

In this talk

- We will pick one such widespread body of knowledge

  Relational database theory [1]

  and will start *refactoring* it in a *"let the symbols do the work"* calculation style.
- Is this concern for *theory refactoring* a new one?
  No — it has a long tradition in mathematics and engineering:

[1] In fact, the data dependency part of it, as far as the talk is concerned

In this talk

- We will pick one such widespread body of knowledge

  Relational database theory [1]

  and will start *refactoring* it in a *"let the symbols do the work"* calculation style.
- Is this concern for *theory refactoring* a new one?
  No — it has a long tradition in mathematics and engineering:

[1] In fact, the data dependency part of it, as far as the talk is concerned

A "notation problem"

Mathematical modelling

requires *descriptive* notations, therefore:

- intuitive
- domain-specific
- often graphical, geometrical

Reasoning

requires *elegant* notations, therefore:

- simple and compact
- generic
- cryptic, otherwise clumsy to manipulate

A "notation problem"

Mathematical modelling

requires *descriptive* notations, therefore:

- intuitive
- domain-specific
- often graphical, geometrical

Reasoning

requires *elegant* notations, therefore:

- simple and compact
- generic
- cryptic, otherwise clumsy to manipulate

Modelling? Reasoning?

Our civilization has a long tradition in ("al-djabr") equational reasoning:

- Examples of "al-djabr" rules: in arithmetics

$$x - z \leq y \quad \equiv \quad x \leq y + z$$

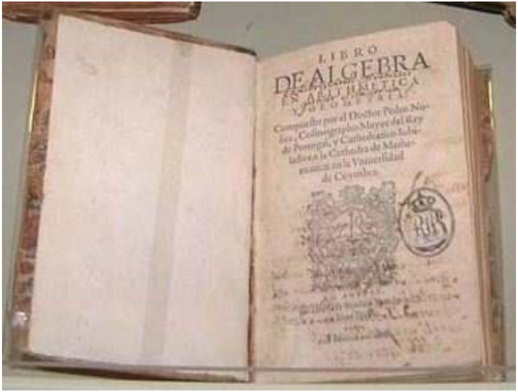
- in set theory

$$A - B \subseteq C \quad \equiv \quad A \subseteq C \cup B$$

"Al-djabr" rules are known since the 9c. (They are nowadays referred to as **Galois connections**.)

By the way

"Al-djabr" reasoning rediscovered in Nunes' *Libro de Algebra en Arithmetica y Geometria (1567)*



(...) the inventor of this art was a Moorish mathematician, whose name was Gebre, & in some libraries there is a small arabic treaty which contains chapters that we use
(fol. a ij r)

Reference to *On the calculus of al-gabr and al-muqâbala* by Abû Al-Huwârizmî, a famous 9c Persian mathematician.

A problem in CS teaching

CS students faced with a contradiction:

- at middle school they are trained in "al-djabr" reasoning (linear equations, polynomials, etc)
- at high-school they are faced with *modus ponens* — massive use of "implication-first" logic (if any)

Shouldn't we all be concerned about this?

How does one bring "al-djabr" reasoning in?

Tradition (again) points to **"math-space" transforms**, eg.

t-space

s-space

Given problem

$$y'' + 4y' + 3y = 0$$
$$y(0) = 3$$
$$y'(0) = 1$$

Subsidiary equation

$$s^2 Y + 4sY + 3Y = 3s + 13$$

Solution of subs. equation

$$Y = \frac{-2}{s+3} + \frac{5}{s+1}$$

Solution of given problem

$$y(t) = -2e^{-3t} + 5e^{-t}$$

How does one bring "al-djabr" reasoning in?

Tradition (again) points to **"math-space" transforms**, eg.

t-space

Given problem

$$y'' + 4y' + 3y = 0$$
$$y(0) = 3$$
$$y'(0) = 1$$

s-space

Subsidiary equation

$$s^2 Y + 4sY + 3Y = 3s + 13$$

Solution of subs. equation

$$Y = \frac{-2}{s+3} + \frac{5}{s+1}$$

Solution of given problem

$$y(t) = -2e^{-3t} + 5e^{-t}$$

Integration? Quantification?

An integral transform:

$$(\mathcal{L}\ f)s = \int_0^\infty e^{-st} f(t)dt$$

$f(t)$	$\mathcal{L}(f)$
1	$\frac{1}{s}$
t	$\frac{1}{s^2}$
t^n	$\frac{n!}{s^{n+1}}$
e^{at}	$\frac{1}{s-a}$
etc	

A parallel:

$$\langle \int\ x\ :\ 0 \leq x \leq 10 :\ x^2 - x \rangle$$

$$\langle \forall\ x\ :\ 0 \leq x \leq 10 :\ x^2 \geq x \rangle$$

The pointfree (PF) transform

An "*s*-space analog" for logical quantification

ϕ	$PF\ \phi$
$\langle \exists\ a\ ::\ b\ R\ a \wedge a\ S\ c \rangle$	$b(R \cdot S)c$
$\langle \forall\ a, b\ ::\ b\ R\ a \Rightarrow b\ S\ a \rangle$	$R \subseteq S$
$\langle \forall\ a\ ::\ a\ R\ a \rangle$	$id \subseteq R$
$\langle \forall\ x\ ::\ x\ R\ b \Rightarrow x\ S\ a \rangle$	$b(R \setminus S)a$
$\langle \forall\ c\ ::\ b\ R\ c \Rightarrow a\ S\ c \rangle$	$a(S\ /\ R)b$
$b\ R\ a \wedge c\ S\ a$	$(b, c)\langle R, S \rangle a$
$b\ R\ a \wedge d\ S\ c$	$(b, d)(R \times S)(a, c)$
$b\ R\ a \wedge b\ S\ a$	$b\ (R \cap S)\ a$
$b\ R\ a \vee b\ S\ a$	$b\ (R \cup S)\ a$
$(f\ b)\ R\ (g\ a)$	$b(f^\circ \cdot R \cdot g)a$
$\textsc{True}$	$b\ \top\ a$
$\textsc{False}$	$b\ \bot\ a$

Road map in theory “PF-refactoring”

- Start with **coreflexive** models of the existing theory
- Generalize coreflexives to **arbitrary** binary relations “as much as possible”
- Add to the theory by restricting to functions and “seeing what happens”

Predicates PF-transformed

- **Binary** predicates :

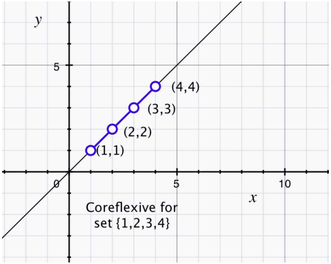
$$R = [\![b]\!] \quad \equiv \quad (y\ R\ x \equiv b(y,x))$$

- **Unary** predicates become fragments of *id* (coreflexives) :

$$R = [\![p]\!] \quad \equiv \quad (y\ R\ x \equiv (p\ x) \wedge x = y)$$

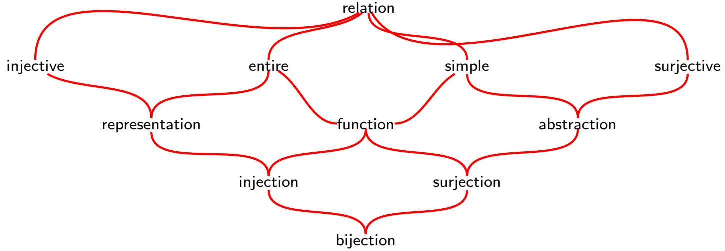
eg.

$$[\![1 \leq x \leq 4]\!] =$$



Some definitions

The whole picture:



where

	Reflexive	*Coreflexive*
ker R	**entire** R	**injective** R
img R	**surjective** R	**simple** R

$$\begin{aligned} \ker R &= R^\circ \cdot R \\ \operatorname{img} R &= R \cdot R^\circ \end{aligned}$$

Data dependency theory

Recall

- Data bases — collections of (large) sets on n-ary **tuples** ("tables")
- **Attributes** — names for indices in n-tuples

Data dependency theory:

- A data **factorization** ("fission") theory — large sets of (long) tuples are split into less redundant structures of smaller sets of (shorter) tuples
- **No loss** of data if particular data dependencies hold
- Data dependencies can be functional (FDs) or multi-valued (MVDs)

FDs — Maier (1983) etc

Given subsets $x, y \subseteq S$ of the relation scheme S of a relation R, this relation is said to satisfy functional dependency $x \rightarrow y$ iff all pairs of tuples $t, t' \in R$ which "agree" on x also "agree" on y:

$R =$

...	x	...	y	...
...	...	...	...	...
t	a	...	c	...
t'	a	...	c	...
...	...	...	...	...

$$\langle \forall\ t, t' \ :\ t, t' \in R : \quad t[x] = t'[x] \quad \Rightarrow \quad t[y] = t'[y] \ \rangle \qquad (1)$$

□

(Notation $t[x]$ means "the values in t of the attributes in x")

MVD definition — Maier (1983)

Given subsets $x, y \subseteq S$ of the relation scheme S of n-ary relation R, this relation is said to satisfy *multi-valued* dependency (MVD) $x \twoheadrightarrow y$ iff, for any two tuples $t, t' \in R$ which "agree" on x there exists a tuple $t'' \in R$ which "agrees" with t on xy and "agrees" with t' on $z = S - xy$:

	x	y	z
t	a	c	b
t''	a	c	b'
t'	a	c'	b'

$$\langle \forall\ t, t' \ :\ t, t' \in R : \quad t[x] = t'[x] \quad \rangle \qquad (2)$$
$$\Downarrow$$
$$\langle \exists\ t'' \ :\ t'' \in R : \quad \begin{array}{l} t[xy] = t''[xy] \wedge \\ t''[z] = t'[z] \end{array} \rangle$$

holds. $\square$

MVD definition — Beeri, Fagin & Howard (1977)

Given subsets $x, y \subseteq S$ of the relation scheme S of an n-ary relation R, let $z = S - xy$. R is said to satisfy the *multi-valued* dependency (MVD) $x \twoheadrightarrow y$ iff, for every xz-value ab that appears in R, one has $Y(ab) = Y(a)$, where for every $k \subseteq S$ and k-value c, function Y is defined as follows:

	x	y	z
t	a	c	b
t'''	a	c'	b
t''	a	c	b'
t'	a	c'	b'

$$Y(c) \;=\; \{v \mid \langle \exists\ t\ :\ t \in R :\ t[k] = c \wedge t[y] = v \rangle\}$$

$\square$

Putting everything together, $x \overset{R}{\twoheadrightarrow} Y$ means:

$$\langle \forall\ a, b\ :\ \langle \exists\ t\ :\ t \in R :\ t[xz] = ab \rangle :\ Y_{R,x}(a) = Y_{R,xz}(ab) \rangle \quad (3)$$

Standard FD theory

Inference rules for FD reasoning based on

- *Armstrong axioms* for computing closures of sets of FDs

However,

- base formulæ too complex
- no explicit proof of

$$\text{Maier} \quad \equiv \quad \text{Beeri, Fagin \& Howard (?)}$$

Who has checked

$$\begin{array}{lcl} \text{Maier} & \Rightarrow & \text{Beeri, Fagin \& Howard?} \\ \text{Maier} & \Leftarrow & \text{Beeri, Fagin \& Howard?} \end{array}$$

We want to write less maths and... "let the symbols do the work"

The role of functions

From **Database Systems: The Complete Book** by Garcia-Molina, Ullman and Widom (2002), p. 87:

> **What Is "Functional" About Functional Dependencies?**
>
> $A_1 A_2 \cdots A_n \rightarrow B$ is called a "functional dependency" because in principle there is a function that takes a list of values [...] and produces a unique value (or no value at all) for B [...] However, this function is not the usual sort of function that we meet in mathematics, because there is no way to compute it from first principles. [...] Rather, the function is only computed by lookup in the relation [...]

In fact, (partial) functions are everywhere in FD theory:

- as attributes
- as the FDs themselves

However,

- No advantage is taken of the rich calculus of functions

Functions in one slide

- A function f is a binary relation such that

Pointwise	Pointfree	
"Left" Uniqueness		
$b\ f\ a \wedge b'\ f\ a \quad \Rightarrow \quad b = b'$	$\mathrm{img}\, f \quad \subseteq \quad id$	(f is simple)
Leibniz principle		
$a = a' \quad \Rightarrow \quad f\ a = f\ a'$	$id \quad \subseteq \quad \ker f$	(f is entire)

- Useful "al-djabr" rules (GCs):

$$f \cdot R \subseteq S \quad \equiv R \subseteq f^{\circ} \cdot S \tag{4}$$

$$R \cdot f^{\circ} \subseteq S \quad \equiv R \subseteq S \cdot f \tag{5}$$

$$\tag{6}$$

- Equality:

$$f \subseteq g \equiv f = g \equiv f \supseteq g$$

Functions in one slide

- A function f is a binary relation such that

Pointwise	Pointfree	
"Left" Uniqueness		
$b\ f\ a \wedge b'\ f\ a \quad \Rightarrow \quad b = b'$	$\mathsf{img}\ f \quad \subseteq \quad id$	(f is simple)
Leibniz principle		
$a = a' \quad \Rightarrow \quad f\ a = f\ a'$	$id \quad \subseteq \quad \ker f$	(f is entire)

- Useful "al-djabr" rules (GCs):

$$f \cdot R \subseteq S \quad \equiv R \subseteq f^\circ \cdot S \tag{4}$$

$$R \cdot f^\circ \subseteq S \quad \equiv R \subseteq S \cdot f \tag{5}$$

(6)

- Equality:

$$f \subseteq g \equiv f = g \equiv f \supseteq g$$

Simple relations in one slide

- "Al-djabr" rules for simple R:

$$R \cdot R \subseteq T \quad \equiv \quad (\delta\, R) \cdot R \quad \subseteq \quad R^\circ \cdot T \tag{7}$$

$$R \cdot R^\circ \subseteq T \quad \equiv \quad R \cdot \delta\, R \quad \subseteq \quad T \cdot R \tag{8}$$

where $\delta\, R$ (=domain of R) is the coreflexive part of $\ker R$ ($\delta\, R \;=\; \ker R \cap id$).

- **Equality**

$$R = S \quad \equiv \quad R \subseteq S \wedge \delta\, S \subseteq \delta\, R \tag{9}$$

follows from (7, 8).

Simple relations in one slide

- "Al-djabr" rules for simple R:

$$R \cdot R \subseteq T \quad \equiv \quad (\delta\, R) \cdot R \quad \subseteq \quad R^\circ \cdot T \tag{7}$$

$$R \cdot R^\circ \subseteq T \quad \equiv \quad R \cdot \delta\, R \quad \subseteq \quad T \cdot R \tag{8}$$

  where $\delta\, R$ (=domain of R) is the coreflexive part of $\ker R$ ($\delta\, R \quad = \quad \ker R \cap id$).

- **Equality**

$$R = S \quad \equiv \quad R \subseteq S \wedge \delta\, S \subseteq \delta\, R \tag{9}$$

  follows from (7, 8).

FDs PF transformed (1)

Pointwise

$$\langle \forall\ t, t' \ :\ t, t' \in R :\quad t[x] = t'[x] \quad \Rightarrow \quad t[y] = t'[y]\ \rangle$$

Pointfree:

$$
\begin{array}{cl}
& R \cdot (x^\circ \cdot x) \cdot R \quad \subseteq \quad y^\circ \cdot y \\
\equiv & \qquad \{\ \text{shunting}\ \} \\
& (y \cdot R \cdot x^\circ) \cdot (x \cdot R \cdot y^\circ) \quad \subseteq \quad id \\
\equiv & \qquad \{\ R \text{ is coreflexive}\ \} \\
& (y \cdot R \cdot x^\circ) \cdot (y \cdot R \cdot x^\circ)^\circ \quad \subseteq \quad id \\
\equiv & \qquad \{\ \text{define projection } \pi_{g,f} = g \cdot R \cdot f^\circ\ \} \\
& \pi_{y,x} R \text{ is simple}
\end{array}
$$

FD generalization

We let R be **any** binary relation and f, g **arbitrary** functions in

$$\pi_{g,f} R \stackrel{\mathrm{def}}{=} g \cdot R \cdot f^{\circ} \qquad \begin{array}{ccc} A & \xleftarrow{R} & B \\ {\scriptstyle g}\downarrow & & \downarrow{\scriptstyle f} \\ C & \xleftarrow[\pi_{g,f} R]{} & D \end{array} \tag{10}$$

and define:

$$f \xrightarrow{R} g \quad \equiv \quad \text{projection } \pi_{g,f} R \text{ is simple}$$

Our aim :

- Calculate the standard **Armstrong** axioms from this PF definition

FDs PF-transformed (2): injectivity

Pointwise

$$\langle \forall\ t, t' \ :\ t, t' \in R : \quad t[x] = t'[x] \quad \Rightarrow \quad t[y] = t'[y]\ \rangle$$

Pointfree:

$$
\begin{array}{ll}
& R \cdot (x^\circ \cdot x) \cdot R \subseteq y^\circ \cdot y \\
\equiv & \qquad \{ \ \text{converses} \ ; \ R \text{ is coreflexive} \ \} \\
& (R \cdot x^\circ) \cdot (x \cdot R^\circ)^\circ \subseteq y^\circ \cdot y \\
\equiv & \qquad \{ \ \ker R = R^\circ \cdot R \ \} \\
& \ker (x \cdot R^\circ) \subseteq \ker y \\
\equiv & \qquad \{ \ y \text{ is } \textit{less injective} \text{ than } x \text{ ``inside } R\text{''} \ \} \\
& y \leq x \cdot R^\circ
\end{array}
$$

Injectivity preorder

- Definition

$$R \leq S \quad \stackrel{\text{def}}{=} \quad \ker S \subseteq \ker R \tag{11}$$

($R \leq S \equiv$ *"R is less injective than S"*)

- "Al-djabr" rules, eg:

$$R \cdot g \leq S \quad \equiv \quad R \leq S \cdot g^{\circ} \tag{12}$$

— the "injectivity derivative" of the corresponding "at most" rule (5).

PF-transformed FD: injectivity

We let R be *any* binary relation and f, g *arbitrary* functions in

$$f \xrightarrow{R} g \quad \equiv \quad g \leq f \cdot R^{\circ} \qquad \begin{tikzcd} A \arrow[d, "g"'] \arrow[dr, "f \cdot R^{\circ}"] & B \arrow[l, "R"'] \arrow[d, "f"] \\ C & D \end{tikzcd} \tag{13}$$

This PF-version is

- simple and elegant
- particularly agile in calculations

Example of reasoning

The following fact — FD composition — is absent from the standard theory:

$$f \xrightarrow{S \cdot R} h \quad \Leftarrow \quad f \xrightarrow{R} g \quad \wedge \quad g \xrightarrow{S} h \tag{14}$$

Calculation:

$$
\begin{array}{ll}
& f \xrightarrow{R} g \quad \wedge \quad g \xrightarrow{S} h \\
\equiv & \quad \{ \text{ (13) twice } \} \\
& g \leq f \cdot R^{\circ} \quad \wedge \quad h \leq g \cdot S^{\circ} \\
\Rightarrow & \quad \{ \ \leq\text{-monotonicity of } (\ \cdot S^{\circ}) \text{ ; converses } \} \\
& g \cdot S^{\circ} \leq f \cdot (S \cdot R)^{\circ} \quad \wedge \quad h \leq g \cdot S^{\circ} \\
\Rightarrow & \quad \{ \ \leq\text{-transitivity } \} \\
& h \leq f \cdot (S \cdot R)^{\circ} \\
\equiv & \quad \{ \text{ (13) again } \} \\
& f \xrightarrow{S \cdot R} h
\end{array}
$$

Rôle of injectivity

1. After all, what matters about f and g in (13) is their "degree of injectivity" — as measured by $\ker f$ and $\ker g$ — in opposite directions:
   - more injective f
   - less injective g

   will strengthen a given FD $f \xrightarrow{R} g$.
2. Limit cases (for all f, g):
   - "Most injective" antecendent

   $$id \xrightarrow{R} g \tag{15}$$

   - "Least injective" consequent

   $$f \xrightarrow{R} ! \tag{16}$$

Rôle of definedness

Kernel $\ker R$ also measures *definedness* (otherwise $\delta\, R = \ker R \cap id$ would be a contradiction). Then, for all f, g

$$f \xrightarrow{\perp} g$$

holds (where $\perp$ denotes the empty relation) and — of course —

$$f \xrightarrow{id} f \tag{17}$$

Side topic: (17) and (14) together set up a **category** whose objects are functions f, g, etc. and whose arrows $f \xrightarrow{\quad R \quad} g$ are relations satisfying $f \xrightarrow{R} g$.

Sets of attributes

In the standard theory, x and y in (1) are **sets** of observable attributes, as in eg. the following Armstrong axioms:

- F3. **Additivity** (or **Union**):

$$x \xrightarrow{T} y \wedge x \xrightarrow{T} z \quad \Rightarrow \quad x \xrightarrow{T} yz \tag{18}$$

- F4. **Projectivity**:

$$x \xrightarrow{T} yz \quad \Rightarrow \quad x \xrightarrow{T} y \wedge x \xrightarrow{T} z \tag{19}$$

Our generic theory interprets "set" yz as function $\langle y, z \rangle$, where

$$(a, b)\langle R, S \rangle c \quad \equiv \quad a\ R\ c \wedge b\ S\ c \tag{20}$$

Relational splits

Below we calculate $F3, F4$ in one go, for arbitrary (suitably typed) R, f, g, h:

$$f \xrightarrow{R} gh \quad \equiv \quad f \xrightarrow{R} g \quad \wedge \quad f \xrightarrow{R} h \tag{21}$$

Calculation:

$$
\begin{array}{cl}
& f \xrightarrow{R} gh \\
\equiv & \{ \ (13) \ ; \text{expansion of shorthand } gh \ \} \\
& \langle g, h \rangle \leq f \cdot R^{\circ} \\
\equiv & \{ \ \text{split is } \mathit{lub} \ (22) \text{ — see next slide} \ \} \\
& g \leq f \cdot R^{\circ} \wedge h \leq f \cdot R^{\circ} \\
\equiv & \{ \ (13) \text{ twice} \ \} \\
& f \xrightarrow{R} g \quad \wedge \quad f \xrightarrow{R} h
\end{array}
$$

Split injectivity (little) theory

Relevance of GC

$$\langle R, S\rangle \leq T \quad \equiv \quad R \leq T \wedge S \leq T \tag{22}$$

which is the ker-derivative of

$$T \subseteq R \cap S \quad \equiv \quad T \subseteq R \wedge T \subseteq S \tag{23}$$

Thus we can rely on cancellation laws

$$R \leq \langle R, S\rangle \quad \text{and} \quad S \leq \langle R, S\rangle \tag{24}$$

(compare with set inclusion).

Abbreviation

To keep up with the standard theory, we will write fg instead of $\langle f, g\rangle$.

Generic Armstrong axioms

Thanks to the $\leq$-ordering, our PF-calculations show that

- Checking the axioms is almost not work at all
- Four of these axioms generalize to arbitrary binary relations
- Alternative versions of some axioms are no longer equivalent in the general case
- Co-transitivity ($R \subseteq R \cdot R$) emerges as interesting property
- Coreflexives (sets) generalize to *pers* ("sets with axioms")

(Details in [4])

MVDs

Recall Maier's definition:

$$\begin{array}{l}\langle \forall\ t, t' \ :\ t, t' \in R : \qquad t[x] = t'[x] \qquad \rangle \\ \qquad\qquad\qquad\qquad\qquad\qquad \Downarrow \\ \qquad\qquad \langle \exists\ t'' \ :\ t'' \in R : \ \begin{array}{l} t[xy] = t''[xy] \wedge \\ t''[z] = t'[z] \end{array} \rangle \end{array}$$

This PF-transforms to

$$x \overset{R}{\twoheadrightarrow} y \quad = \quad R \cdot (\ker x) \cdot R \subseteq (\ker xy) \cdot R \cdot \ker z \tag{25}$$

where z is the projection function associated to the attributes in $S - xy$.

"Al-djabr"ing MVDs

$$
\begin{array}{lcll}
x \stackrel{R}{\twoheadrightarrow} y & \equiv & R \cdot (\ker x) \cdot R \subseteq (\ker xy) \cdot R \cdot \ker z & (26) \\
& \equiv & \qquad \{ \text{ kernels ; (4 and 5) } \} & \\
& & (xy \cdot R \cdot x^\circ) \cdot (x \cdot R \cdot z^\circ) \;\subseteq\; xy \cdot R \cdot z^\circ & (27) \\
& \equiv & \qquad \{ \text{ (10) three times } \} & \\
& & (\pi_{xy,x} R) \cdot (\pi_{x,z} R) \;\subseteq\; \pi_{xy,z} R & (28)
\end{array}
$$

cf.

$$
\begin{tikzcd}
A \arrow[dr, "x"'] \arrow[dd, "xy"'] & & A \arrow[ll, "R"'] \arrow[dl, "x"] \arrow[dd, "z"] \\
& X \arrow[dl, "\pi_{xy,x} R"'] \arrow[d, phantom, "\subseteq"] & \\
X \times Y & {} & Z \arrow[ul, "\pi_{x,z} R"'] \arrow[ll, "\pi_{xy,z} R"]
\end{tikzcd}
$$

MVD "meaning"

PF version

$$(\pi_{xy,x} R) \cdot (\pi_{x,z} R) \quad \subseteq \quad \pi_{xy,z} R$$

requires R to be an endo-relation and provides a simple meaning for MVDs: $x \overset{R}{\twoheadrightarrow} y$ *holds iff projection* $\pi_{xy,z} R$ *"factorizes" through* x, for instance:

$$\left(\begin{array}{c|c|c|c} & x & y & x \\ \hline\hline t & a & c & a \\ \hline t' & a & c' & a \\ \hline \end{array} \right) \cdot \left(\begin{array}{c|c|c} & x & z \\ \hline\hline t & a & b \\ \hline t' & a & b' \\ \hline \end{array} \right) \quad \subseteq \quad \begin{array}{c|c|c|c} & x & y & z \\ \hline\hline t & a & c & b \\ \hline t''' & a & c' & b \\ \hline t'' & a & c & b' \\ \hline t' & a & c' & b' \\ \hline \end{array}$$

Lossless decomposition

We are pretty close to one of the main results in RDB theory, the theorem of **lossless decomposition** of MVDs: $x \overset{R}{\twoheadrightarrow} y$ holds *iff* R decomposes losslessly into two relations with schemata xy and xz, respectively:

$$x \overset{R}{\twoheadrightarrow} y \quad \equiv \quad (\pi_{y,x} R) \bowtie (\pi_{z,x} R) = \pi_{yz,x} R$$

Maier [3] proves this in "implication-first" logic style, in two parts — *if* + *only if* — involving existential and universal quantifications over no less than six tuple variables $t, t_1, t_2, t_1', t_2', t_3$:

Lossless decomposition (Maier)

Theorem 7.1 Let r be a relation on scheme R, and let X, Y, and Z be subsets of R such that $Z = R - (X\,Y)$. Relation r satisfies the MVD $X \twoheadrightarrow Y$ if and only if r decomposes losslessly onto the relation schemes $R_1 = X\,Y$ and $R_2 = X\,Z$.

Proof: Suppose the MVD holds. Let $r_1 = \pi_{R_1}(r)$ and $r_2 = \pi_{R_2}(r)$. Let t be a tuple in $r_1 \bowtie r_2$. There must be a tuple $t_1 \in r_1$ and a tuple $t_2 \in r_2$ such that $t(X) = t_1(X) = t_2(X)$, $t(Y) = t_1(Y)$, and $t(Z) = t_2(Z)$. Since r_1 and r_2 are projections of r, there must be tuples t_1' and t_2' in r with $t_1(X\,Y) = t_1'(X\,Y)$ and $t_2(X\,Z) = t_2'(X\,Z)$. The MVD $X \twoheadrightarrow Y$ implies that t must be in r, since r must contain a tuple t_3 with $t_3(X) = t_1'(X)$, $t_3(Y) = t_1'(Y)$, and $t_3(Z) = t_2'(Z)$, which is a description of t.

Suppose now that r decomposes losslessly onto R_1 and R_2. Let t_1 and t_2 be tuples in r such that $t_1(X) = t_2(X)$. Let r_1 and r_2 be defined as before. Relation r_1 contains a tuple $t_1' = t_1(X\,Y)$ and relation r_2 contains a tuple $t_2' = t_2(X\,Z)$. Since $r = r_1 \bowtie r_2$, r contains a tuple t such that $t(X\,Y) = t_1(X\,Y)$ and $t(X\,Z) = t_2(X\,Z)$. Tuple t is the result of joining t_1' and t_2'. Hence t_1 and t_2 cannot be used in a counterexample to $X \twoheadrightarrow Y$, hence r satisfies $X \twoheadrightarrow Y$.

Alternative PF calculation

Sequence of equivalences based on the following facts:

- joining two projections which share the same antecedent function, say x, is nothing but binary relation *split* (20):

$$(\pi_{y,x} R) \bowtie (\pi_{z,x} R) \stackrel{\text{def}}{=} \langle y \cdot R \cdot x^\circ, z \cdot R \cdot x^\circ \rangle \qquad (29)$$

- lossless decomposition can be expressed parametrically *wrt* consequent functions y and z,

$$\pi_{yz,x} R = (\pi_{y,x} R) \bowtie (\pi_{z,x} R)$$

that is

$$\langle y, z \rangle \cdot R \cdot x^\circ = \langle y \cdot R \cdot x^\circ, z \cdot R \cdot x^\circ \rangle$$

By the way

The following special case of *lossless* decomposition is known to every AoP practitioner:

$$\langle y, z \rangle \cdot f \quad = \quad \langle y \cdot f, z \cdot f \rangle \tag{30}$$

— *split-fusion* — a consequence of isomorphism

$$(A \times B)^C \quad \cong \quad (A^C) \times (B^C)$$

(functions yielding pairs "decompose *losslessly*" into pairs of functions)

Alternative PF calculation

$$(\pi_{y,x} R) \bowtie (\pi_{z,x} R) = \pi_{yz,x} R$$

$\equiv$ { (29) ; (10) three times }

$$\langle y \cdot R \cdot x^\circ, z \cdot R \cdot x^\circ \rangle \quad = \quad yz \cdot R \cdot x^\circ$$

$\equiv$ { since $\langle X, Y \rangle \cdot Z \subseteq \langle X \cdot Z, Y \cdot Z \rangle$ holds by monotonicity }

$$\langle y \cdot R \cdot x^\circ, z \cdot R \cdot x^\circ \rangle \quad \subseteq \quad yz \cdot R \cdot x^\circ$$

$\equiv$ { "split twist" rule (31) — twice ; converses }

$$\langle y \cdot R \cdot x^\circ, id \rangle \cdot x \cdot R^\circ \cdot z^\circ \quad \subseteq \quad \langle y, x \cdot R^\circ \rangle \cdot z^\circ$$

$\equiv$ { instances of split-fusion: (32) and (34) }

$$\langle y \cdot R \cdot x^\circ, x \cdot x^\circ \rangle \cdot x \cdot R \cdot z^\circ \quad \subseteq \quad \langle y, x \rangle \cdot R \cdot z^\circ$$

$\equiv$ { instances of split-fusion: (33) and (34) }

$$(\langle y, x \rangle \cdot R \cdot x^\circ) \cdot (x \cdot R \cdot z^\circ) \quad \subseteq \quad \langle y, x \rangle \cdot R \cdot z^\circ$$

$\equiv$ { (27) }

$$x \overset{R}{\twoheadrightarrow} y$$

PF calculation details

"Split twist" rule

$$\langle R, S\rangle \cdot T \subseteq \langle U, V\rangle \cdot X \quad \equiv \quad \langle R, T^\circ\rangle \cdot S^\circ \subseteq \langle U, X^\circ\rangle \cdot V^\circ \tag{31}$$

Instances of (relational) split-fusion

- For simple (thus difunctional) S:

$$\langle R, T\rangle \cdot S \quad = \quad \langle R, T \cdot S \cdot S^\circ\rangle \cdot S \tag{32}$$

$$\langle R, S\rangle \cdot S^\circ \quad = \quad \langle R \cdot S^\circ, S \cdot S^\circ\rangle \tag{33}$$

- Split pre-conditioning rule:

$$\langle R, S\rangle \cdot \Phi = \langle R, S \cdot \Phi\rangle \quad \equiv \quad \Phi \text{ is coreflexive} \tag{34}$$

Checking Beeri, Fagin & Howard's definition

(First step in the calculation is based on the fact that y and z are interchangeable in MVDs, see [4] for details):

$$
\begin{array}{lll}
\text{Maier's def.} & \equiv & xy \cdot R \cdot x^\circ \cdot x \cdot R \cdot z^\circ \subseteq xy \cdot R \cdot z^\circ \\
& \equiv & \{ \text{ swap } y \text{ and } z \text{ and take converses } \} \\
& & y \cdot R \cdot x^\circ \cdot x \cdot R \cdot xz^\circ \subseteq y \cdot R \cdot xz^\circ \\
& \equiv & \{ \ R = R \cdot R^\circ \text{ since } R \text{ is coreflexive } \} \\
& & y \cdot R \cdot x^\circ \cdot \pi_1 \cdot xz \cdot R \cdot R^\circ \cdot xz^\circ \subseteq y \cdot R \cdot xz^\circ \\
& \equiv & \{ \text{ please turn over } \}
\end{array}
$$

MVDs PF-transformed

$$
\begin{array}{ll}
& y \cdot R \cdot x^\circ \cdot \pi_1 \cdot xz \cdot R \cdot R^\circ \cdot xz^\circ \quad \subseteq \quad y \cdot R \cdot xz^\circ \\
\equiv & \qquad \{ \text{ introduce image and the power-transpose } \} \\
& \Lambda(y \cdot R \cdot x^\circ \cdot \pi_1) \cdot \mathrm{img}\,(xz \cdot R) \quad \subseteq \quad \Lambda(y \cdot R \cdot xz^\circ) \\
\equiv & \qquad \{ \text{ define } \gamma_{f,g} R = \Lambda(f \cdot R \cdot g^\circ) \text{ ; “al-djabr” (shunting) } \} \\
& \mathrm{img}\,(xz \cdot R) \quad \subseteq \quad (\gamma_{y,x} R \cdot \pi_1)^\circ \cdot (\gamma_{y,xz} R)
\end{array}
$$

Finally, we go back to points (third step of a typical PF-transform argument):

MVDs PF-transformed

$$
\begin{array}{ll}
& \mathrm{img}\,(xz \cdot R) \quad \subseteq \quad (\gamma_{y,x} R \cdot \pi_1)^{\circ} \cdot \gamma_{y,xz} R \\
\equiv & \qquad \{ \text{ reverse PF-transform (for } R \text{ coreflexive, } xz \cdot R \text{ is simple) } \} \\
& \langle \forall\ k \ :\ k\ \mathrm{img}\,(xz \cdot R)\ k :\ (\gamma_{y,x} R \cdot \pi_1) k = (\gamma_{y,xz} R) k \rangle \\
\equiv & \qquad \{ \text{ reverse PF-transform of the image of } xz \cdot R \ \} \\
& \langle \forall\ k \ :\ \langle \exists\ t \ :\ t \in R :\ xz(t) = k \rangle :\ (\gamma_{y,x} R \cdot \pi_1) k = (\gamma_{y,xz} R) k \rangle \\
\equiv & \qquad \{ \text{ rename } k := (b, a) \text{ and simplify } \} \\
& \left\langle \begin{array}{c} \forall\ a, b \ : \\ \langle \exists\ t \ :\ t \in R :\ (x\ \ t) = a \wedge (z\ \ t) = b \rangle : \\ (\gamma_{y,x} R)\ a = (\gamma_{y,xz} R)(a, b) \end{array} \right\rangle \\
\equiv & \qquad \{ \text{ recognize } (\gamma_{y,x} R) a \text{ as } Y(a) \ \} \\
& \text{Beeri, Fagin \& Howard definition}
\end{array}
$$

Difficulties

Some MVD rules are hard to PF-transform, eg.

- M5. **Transitivity**:

$$x \overset{R}{\twoheadrightarrow} y \wedge y \overset{R}{\twoheadrightarrow} z \quad \Rightarrow \quad x \overset{R}{\twoheadrightarrow} (z - y) \tag{35}$$

- M6. **Pseudotransitivity**:

$$x \overset{R}{\twoheadrightarrow} y \wedge yw \overset{R}{\twoheadrightarrow} z \quad \Rightarrow \quad xw \overset{R}{\twoheadrightarrow} (z - yw) \tag{36}$$

Question

Given two functions f, g, what is the generic meaning of "$f - g$" ?

Richer theory

Promoting attributes to functions brings about richer results such as eg.

$$x \xrightarrow{R} y \quad \equiv \quad f \cdot x \xrightarrow{R} f \cdot y \qquad \Leftarrow \quad f \text{ is injective}$$

eg. structural FDs:

$$x \xrightarrow{R} y \quad \equiv \quad \mathsf{F}x \xrightarrow{\mathsf{F}R} \mathsf{F}y$$

eg. specific results on functional dependences on "the functions themselves",

$$f \xrightarrow{g} id \quad \equiv \quad f \xrightarrow{id} g$$

etc.

Current work

- Basic: analyse the impact of a richer definition of kernel (by Jeremy)

$$\ker R \quad = \quad (R \setminus R) \cap (R \setminus R)^{\circ}$$

  on the injectivity preorder. (Both coincide on functions).
- Extension: `NULL` values (!)
- Applied: replay Mark Jones' **Type Classes with Functional Dependencies** [2] in our approach — the most well-known (non-trivial) application of FDs outside the database domain. This is likely to benefit from our generalization (interplay with extra ingredients such substitutions and unification).
- Generic: synergies with other disciplines

Current work

Relationship between function *divisibility* and the injectivity preorder: two preorders on functions

- "Left divisibility" — $g \sqsubseteq f$ iff exists k such that

$$f = g \cdot k \tag{37}$$

- "Right divisibility" — $g \preceq f$ iff exists k such that

$$f = k \cdot g \tag{38}$$

Clearly, $\preceq$ is the converse of the injectivity preorder, restricted to functions (next slide)

Current work

$$
\begin{array}{ll}
 & f \leq g \\
\equiv & \quad \{ \text{ FDs on functions: } f \leq g \equiv g \xrightarrow{id} f \text{ ; projections [4] } \} \\
 & f \cdot g^\circ \text{ is simple} \\
\equiv & \quad \{ \text{ simple relations are fragments of functions (and vice versa) } \} \\
 & \langle \exists\, k \;::\; f \cdot g^\circ \subseteq k \rangle \\
\equiv & \quad \{ \text{ ``al-djabr'' (shunting) } \} \\
 & \langle \exists\, k \;::\; f \subseteq k \cdot g \rangle \\
\equiv & \quad \{ \text{ function equality } \} \\
 & g \preceq f
\end{array}
$$

Synergies with other CS diciplines

- **Bisimulations** — FD $d \overset{R}{\rightarrow} c$ holds wherever R is a simple bisimulation from coalgebra d to coalgebra c. In other words: c can be less injective than d as far as "allowed by" R. So (implementation) d is allowed to distinguish states which (specification) c does not.

$$\begin{tikzcd} \mathsf{F}\,B & B \arrow[l, "c"'] \\ \mathsf{F}\,A \arrow[u, "\mathsf{F}\,R"] & A \arrow[l, "d"] \arrow[u, "R"'] \end{tikzcd}$$

- **Algebra of Programming** — possible impact in reasoning about specifications. Example: from the *sorting* spec in [1]

$$Sort \quad = \quad [\![ordered]\!] \cdot \ker bagify$$

infer FD $bagify \overset{Sort}{\rightarrow} bagify$, etc

Conclusions

- *"Ut faciant opus signa"* is great
- How could "they" survive for so long only at point-level?
- PF-refactoring of existing theories is useful
- It develops the PF-transform (Algebra of Programming) itself

Rôle of generic **pointfree patterns** in the reasoning:

- Projection:

$$f \cdot R \cdot g^{\circ}$$

- Selection (Greek letters denote coreflexives):

$$\Psi \cdot R \cdot \Phi$$

and so on

Epilogue

"Algebra (...) is thing causing admiration"

> *(...) "Mainly because we see often a great Mathematician unable to resolve a question by Geometrical means, and solve it by Algebra, being that same Algebra taken from Geometry, which is thing causing admiration."*

[**Pedro Nunes** (1502-1578) in **Libro de Algebra en Arithmetica y Geometria**, 1567, fol. 270.]

— my (literal, not literary) translation of:

> *(...) Principalmente que vemos algumas vezes, no poder vn gran Mathematico resoluer vna question por medios Geometricos, y resolverla por Algebra, siendo la misma Algebra sacada de la Geometria, q̃ es cosa de admiraciõ.*

Epilogue

"Algebra (...) is thing causing admiration"

> *(...) "Mainly because we see often a great Mathematician unable to resolve a question by Geometrical means, and solve it by Algebra, being that same Algebra taken from Geometry, which is thing causing admiration."*

[**Pedro Nunes** (1502-1578) in **Libro de Algebra en Arithmetica y Geometria**, 1567, fol. 270.]

— my (literal, not literary) translation of:

> *(...) Principalmente que vemos algumas vezes, no poder vn gran Mathematico resoluer vna question por medios Geometricos, y resolverla por Algebra, siendo la misma Algebra sacada de la Geometria, q̃ es cosa de admiraciõ.*

Verdict

(...) De manera, que
quien sabe por Algebra,
sabe scientificamente.

((...) In this way, who knows by Algebra knows scientifically)

R. Bird and O. de Moor.
Algebra of Programming.
Series in Computer Science. Prentice-Hall International, 1997.
C.A.R. Hoare, series editor.

Mark P. Jones.
Type classes with functional dependencies.
In Gert Smolka, editor, *Programming Languages and Systems, 9th European Symposium on Programming, ESOP 2000, Held as Part of the European Joint Conferences on the Theory and Practice of Software, ETAPS 2000, Berlin, Germany, March 25 - April 2, 2000, Proceedings*, volume 1782 of *LNCS*, pages 230–244. Springer, 2000.

D. Maier.
The Theory of Relational Databases.
Computer Science Press, 1983.

J.N. Oliveira.
Pointfree foundations for lossless decomposition, 2006.

Draft of paper in preparation.