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 \((\text{cata,ana,hylo,} \ldots)\) -morphisms etc
- Abstract interpretation, polymorphism, unification etc

What about theories which \textit{“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

  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:

---

1In 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
  \[ x - z \leq y \equiv x \leq y + z \]
- in set theory
  \[ A - B \subseteq C \equiv A \subseteq C \cup B \]

"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.

Given problem

\[ y'' + 4y' + 3y = 0 \]
\[ y(0) = 3 \]
\[ y'(0) = 1 \]

Subsidiary equation

\[ s^2Y + 4sY + 3Y = 3s + 13 \]

Solution of given problem

\[ y(t) = -2e^{-3t} + 5e^{-t} \]

Solution of subs. equation

\[ Y = \frac{-2}{s+3} + \frac{5}{s+1} \]
How does one bring “al-djabr” reasoning in?

Tradition (again) points to “math-space” transforms, eg.

Given problem

\[ y'' + 4y' + 3y = 0 \]
\[ y(0) = 3 \]
\[ y'(0) = 1 \]

Subsidiary equation

\[ s^2 Y + 4sY + 3Y = 3s + 13 \]

Solution of given problem

\[ y(t) = -2e^{-3t} + 5e^{-t} \]

Solution of subs. equation

\[ Y = \frac{-2}{s+3} + \frac{5}{s+1} \]
Integration? Quantification?

An integral transform:

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

<table>
<thead>
<tr>
<th>(f(t))</th>
<th>(\mathcal{L}(f))</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(\frac{1}{s})</td>
</tr>
<tr>
<td>(t)</td>
<td>(\frac{1}{s^2})</td>
</tr>
<tr>
<td>(t^n)</td>
<td>(\frac{n!}{s^{n+1}})</td>
</tr>
<tr>
<td>(e^{at})</td>
<td>(\frac{1}{s-a})</td>
</tr>
<tr>
<td>etc</td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>$\phi$</th>
<th>$\text{PF } \phi$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\langle \exists a :: b \ R \ a \wedge a \ S \ c \rangle$</td>
<td>$b(R \cdot S)c$</td>
</tr>
<tr>
<td>$\langle \forall a, b :: b \ R \ a \Rightarrow b \ S \ a \rangle$</td>
<td>$R \subseteq S$</td>
</tr>
<tr>
<td>$\langle \forall a :: a \ R \ a \rangle$</td>
<td>$id \subseteq R$</td>
</tr>
<tr>
<td>$\langle \forall x :: x \ R \ b \Rightarrow x \ S \ a \rangle$</td>
<td>$b(R \setminus S)a$</td>
</tr>
<tr>
<td>$\langle \forall c :: b \ R \ c \Rightarrow a \ S \ c \rangle$</td>
<td>$a(S / R)b$</td>
</tr>
<tr>
<td>$b \ R \ a \wedge c \ S \ a$</td>
<td>$(b, c)\langle R, S\rangle a$</td>
</tr>
<tr>
<td>$b \ R \ a \wedge d \ S \ c$</td>
<td>$(b, d)(R \times S)(a, c)$</td>
</tr>
<tr>
<td>$b \ R \ a \wedge b \ S \ a$</td>
<td>$b \ (R \cap S) \ a$</td>
</tr>
<tr>
<td>$b \ R \ a \vee b \ S \ a$</td>
<td>$b \ (R \cup S) \ a$</td>
</tr>
<tr>
<td>$(f \ b) \ R \ (g \ a)$</td>
<td>$b(f^\circ \cdot R \cdot g)a$</td>
</tr>
<tr>
<td>$\text{TRUE}$</td>
<td>$b \top a$</td>
</tr>
<tr>
<td>$\text{FALSE}$</td>
<td>$b \perp a$</td>
</tr>
</tbody>
</table>
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 :
  \[ R = [b] \equiv (y \, R \, x \equiv b(y, x)) \]

- **Unary** predicates become fragments of \( id \) (coreflexives) :
  \[ R = [p] \equiv (y \, R \, x \equiv (p \, x) \land x = y) \]

eg.

\[ [1 \leq x \leq 4] = \]
Some definitions

The whole picture:

<table>
<thead>
<tr>
<th></th>
<th>Reflexive</th>
<th>Coreflexive</th>
</tr>
</thead>
<tbody>
<tr>
<td>ker R</td>
<td>entire R</td>
<td>injective R</td>
</tr>
<tr>
<td>img R</td>
<td>surjective R</td>
<td>simple R</td>
</tr>
</tbody>
</table>

where

\[
\text{ker } R = R^\circ \cdot R
\]

\[
\text{img } R = R \cdot R^\circ
\]
Recall

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

Data dependency theory:

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

\[
\langle \forall t, t' : t, t' \in R : t[x] = t'[x] \implies t[y] = t'[y] \rangle \tag{1}
\]

(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) $x \rightarrow y$ 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 $z = S - xy$:

$$\langle \forall t, t' : t, t' \in R : \quad t[x] = t'[x] \quad \downarrow \quad \exists t'' : t'' \in R : \quad t[xy] = t''[xy] \wedge \quad t''[z] = t'[z] \rangle$$ (2) holds. □
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) $x \rightarrow y$ 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:

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

Putting everything together, $x \rightarrow y^R$ means:

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

Inference rules for FD reasoning based on

- Armstrong axioms for computing closures of sets of FDs

However,

- base formulae too complex
- no explicit proof of

\[
\text{Maier } \equiv \text{ Beeri, Fagin & Howard (?)}
\]

Who has checked

\[
\text{Maier } \Rightarrow \text{ Beeri, Fagin & Howard?}
\]

\[
\text{Maier } \Leftarrow \text{ Beeri, Fagin & Howard?}
\]

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?

$A_1A_2\cdots A_n \rightarrow B$ 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

\[
\begin{array}{|c|c|}
\hline
\text{Pointwise} & \text{Pointfree} \\
\hline
\text{"Left" Uniqueness} & (f \text{ is simple}) \\
\hline
b \; f \; a \land b' \; f \; a & \Rightarrow \; b = b' \\
\hline
\text{Leibniz principle} & (f \text{ is entire}) \\
a = a' & \Rightarrow \; f \; a = f \; a' \\
\hline
\end{array}
\]

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

\[
\begin{align*}
& f \cdot R \subseteq S \equiv R \subseteq f^\circ \cdot S \\
& R \cdot f^\circ \subseteq S \equiv R \subseteq S \cdot f
\end{align*}
\]

- Equality:

\[
f \subseteq g \equiv f = g \equiv f \supseteq g
\]
**Functions in one slide**

- A function \( f \) is a binary relation such that

<table>
<thead>
<tr>
<th>Pointwise</th>
<th>Pointfree</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ b f a \land b' f a \Rightarrow b = b' ]</td>
<td>( \text{img } f \subseteq \text{id} ) (( f ) is simple)</td>
</tr>
<tr>
<td>[ a = a' \Rightarrow f a = f a' ]</td>
<td>( \text{id} \subseteq \ker f ) (( f ) is entire)</td>
</tr>
</tbody>
</table>

- Useful “al-djabr” rules (GCs):

\[
\begin{align*}
    f \cdot R \subseteq S & \equiv R \subseteq f^\circ \cdot S \quad (4) \\
    R \cdot f^\circ \subseteq S & \equiv R \subseteq S \cdot f \quad (5) \\
    R \cdot f^\circ & \subseteq S \quad (6)
\end{align*}
\]

- Equality:

\[ f \subseteq g \equiv f = g \equiv f \supseteq g \]
Simple relations in one slide

• “Al-djabr” rules for simple $R$:

$$R \cdot R \subseteq T \equiv (\delta R) \cdot R \subseteq R^\circ \cdot T \quad (7)$$

$$R \cdot R^\circ \subseteq T \equiv R \cdot \delta R \subseteq T \cdot R \quad (8)$$

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

• Equality

$$R = S \equiv R \subseteq S \land \delta S \subseteq \delta R \quad (9)$$

follows from (7, 8).
Simple relations in one slide

- “Al-djabr” rules for simple $R$:

\[
R \cdot R \subseteq T \iff (\delta R) \cdot R \subseteq R^\circ \cdot T \quad (7)
\]

\[
R \cdot R^\circ \subseteq T \iff R \cdot \delta R \subseteq T \cdot R \quad (8)
\]

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

- Equality

\[
R = S \iff R \subseteq S \land \delta S \subseteq \delta R \quad (9)
\]

follows from (7, 8).
FDs PF transformed (1)

Pointwise

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

Pointfree:

\[
R \cdot (x^\circ \cdot x) \cdot R \subseteq y^\circ \cdot y
\]

\[
\equiv \{\text{shunting}\}
\]

\[
(y \cdot R \cdot x^\circ) \cdot (x \cdot R \cdot y^\circ) \subseteq id
\]

\[
\equiv \{ R \text{ is coreflexive}\}
\]

\[
(y \cdot R \cdot x^\circ) \cdot (y \cdot R \cdot x^\circ)^\circ \subseteq id
\]

\[
\equiv \{\text{define projection } \pi_{g,f} = g \cdot R \cdot f^\circ\}
\]

\[\pi_{y,x} R \text{ is simple}\]
FD generalization

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

$$\pi_{g,f} R \overset{\text{def}}{=} g \cdot R \cdot f^\circ$$

and define:

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

Our aim:

- Calculate the standard Armstrong axioms from this PF definition
FDs PF-transformed (2): injectivity

Pointwise

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

Pointfree:

\[ R \cdot (x^\circ \cdot x) \cdot R \subseteq y^\circ \cdot y \]

\[ \equiv \{ \text{converses ; } R \text{ is coreflexive} \} \]

\[ (R \cdot x^\circ) \cdot (x \cdot R^\circ) \subseteq y^\circ \cdot y \]

\[ \equiv \{ \ker R = R^\circ \cdot R \} \]

\[ \ker (x \cdot R^\circ) \subseteq \ker y \]

\[ \equiv \{ \ y \text{ is less injective than } x \text{ “inside } R” \} \]

\[ y \leq x \cdot R^\circ \]
Injectivity preorder

• Definition

\[ R \leq S \quad \text{def} \quad \ker S \subseteq \ker R \quad (11) \]

\( (R \leq S \equiv \text{“}R \text{ is less injective than } S\text{”} ) \)

• “Al-djabr” rules, eg:

\[ R \cdot g \leq S \equiv R \leq S \cdot g^\circ \quad (12) \]

— the “injectivity derivative” of the corresponding “at most” rule (5).
We let $R$ be any binary relation and $f, g$ arbitrary functions in

$$f \xrightarrow{R} g \equiv g \leq f \cdot R^\circ$$

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:

\[ f^{S \cdot R} \rightarrow h \iff f \rightarrow g \land g^{S} \rightarrow h \]  

(14)

Calculation:

\[ f \rightarrow g \land g^{S} \rightarrow h \]

\[ \equiv \{ (13) \text{ twice} \} \]

\[ g \leq f \cdot R^{\circ} \land h \leq g \cdot S^{\circ} \]

\[ \Rightarrow \{ \leq \text{-monotonicity of } ( \cdot S^{\circ}) ; \text{converses} \} \]

\[ g \cdot S^{\circ} \leq f \cdot (S \cdot R)^{\circ} \land h \leq g \cdot S^{\circ} \]

\[ \Rightarrow \{ \leq \text{-transitivity} \} \]

\[ h \leq f \cdot (S \cdot R)^{\circ} \]

\[ \equiv \{ (13) \text{ again} \} \]

\[ f^{S \cdot R} \rightarrow h \]
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” antecedent
     \[
     id \xrightarrow{R} g
     \]  \hspace{1cm} (15)
   - “Least injective” consequent
     \[
     f \xrightarrow{R} !
     \]  \hspace{1cm} (16)
Rôle of definedness

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

\[
f \not\rightarrow g
\]

holds (where \( \not\rightarrow \) denotes the empty relation) and — of course —

\[
f \xrightarrow{\text{id}} f
\]

(17)

Side topic: (17) and (14) together set up a \textit{category} whose objects are functions \( f, g, \) etc. and whose arrows \( f \xrightarrow{R} g \) 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**):
  \[
  x \xrightarrow{T} y \land x \xrightarrow{T} z \implies x \xrightarrow{T} yz
  \] (18)

- **F4. Projectivity**:
  \[
  x \xrightarrow{T} yz \implies x \xrightarrow{T} y \land x \xrightarrow{T} z
  \] (19)

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

\[
(a, b)\langle R, S \rangle c \equiv a \ R \ c \land b \ S \ c
\] (20)
Relational splits

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

$$f ^R \rightarrow gh \equiv f ^R \rightarrow g \land f ^R \rightarrow h \quad (21)$$

Calculation:

$$f ^R \rightarrow gh$$

$$\equiv \{ (13) ; \text{expansion of shorthand } gh \}$$

$$\langle g, h \rangle \leq f \cdot R^\circ$$

$$\equiv \{ \text{split is lub (22) — see next slide } \}$$

$$g \leq f \cdot R^\circ \land h \leq f \cdot R^\circ$$

$$\equiv \{ (13) \text{ twice } \}$$

$$f ^R \rightarrow g \land f ^R \rightarrow h$$
Split injectivity (little) theory

Relevance of GC

\[ \langle R, S \rangle \leq T \equiv R \leq T \land S \leq T \] (22)

which is the ker-derivative of

\[ T \subseteq R \cap S \equiv T \subseteq R \land T \subseteq S \] (23)

Thus we can rely on cancellation laws

\[ R \leq \langle R, S \rangle \quad \text{and} \quad S \leq \langle R, S \rangle \] (24)

(compare with set inclusion).

Abbreviation

To keep up with the standard theory, we will write \( fg \) instead of \( \langle f, g \rangle \).
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])
Recall Maier’s definition:
\[
\langle \forall t, t' : t, t' \in R : \quad t[x] = t'[x] \rangle \\
\Downarrow
\langle \exists t'' : t'' \in R : \quad t[xy] = t''[xy] \land t''[z] = t'[z] \rangle
\]

This PF-transforms to
\[
x \overset{R}{\longrightarrow} y = R \cdot (\ker x) \cdot R \subseteq (\ker xy) \cdot R \cdot \ker z
\]

where \(z\) is the projection function associated to the attributes in \(S - xy\).
“Al-djabr”ing MVDs

\[
\begin{align*}
R \cdot (\ker x) \cdot R & \subseteq (\ker xy) \cdot R \cdot \ker z \quad (26) \\
& \equiv \{ \text{kernels ; (4 and 5)} \} \\
(xy \cdot R \cdot x^o) \cdot (x \cdot R \cdot z^o) & \subseteq xy \cdot R \cdot z^o \quad (27) \\
& \equiv \{ (10) \text{ three times} \} \\
(\pi_{xy,x} R) \cdot (\pi_{x,z} R) & \subseteq \pi_{xy,z} R \quad (28)
\end{align*}
\]

cf.
MVD “meaning”

PF version

\[(\pi_{xy,x} R) \cdot (\pi_{x,z} R) \subseteq \pi_{xy,z} R\]

requires \( R \) to be an endo-relation and provides a simple meaning for MVDs: \( x \rightarrow y \) holds iff projection \( \pi_{xy,z} R \) “factorizes” through \( x \), for instance:

\[
\begin{pmatrix}
  x & y & x \\
  t & a & c & a \\
  t' & a & c' & a
\end{pmatrix}
\cdot
\begin{pmatrix}
  x & z \\
  t & a & b \\
  t' & a & b'
\end{pmatrix}
\subseteq
\begin{array}{ccc}
  x & y & z \\
  t & a & c & b \\
  t'' & a & c' & b \\
  t''' & a & c & b' \\
  t' & a & c' & b'
\end{array}
\]
We are pretty close to one of the main results in RDB theory, the theorem of **lossless decomposition** of MVDs: $x \rightarrow_{R} y$ holds *iff* $R$ decomposes losslessly into two relations with schemata $xy$ and $xz$, respectively:

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

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 $t, t_1, t_2, t'_1, t'_2, t_3$: 
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 $X \rightarrow\rightarrow Y$ if the 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 $r_1 \Join r_2$. 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 $X \rightarrow\rightarrow Y$ 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 $r = r_1 \Join r_2$, $r$ 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 $X \rightarrow\rightarrow Y$, hence $r$ satisfies $X \rightarrow\rightarrow Y$. 
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 $\text{split}$ (20):
  \[
  (\pi_{y,x} R) \Join (\pi_{z,x} R) \overset{\text{def}}{=} \langle y \cdot R \cdot x^\circ, z \cdot R \cdot x^\circ \rangle \quad (29)
  \]

- lossless decomposition can be expressed parametrically $\text{wrt}$ consequent functions $y$ and $z$,
  \[
  \pi_{yz,x} R = (\pi_{y,x} R) \Join (\pi_{z,x} R)
  \]
  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:

\[
\langle y, z \rangle \cdot f = \langle y \cdot f, z \cdot f \rangle
\]  

— *split-fusion* — a consequence of isomorphism

\[
(A \times B)^C \cong (A^C) \times (B^C)
\]

(functions yielding pairs “decompose losslessly” into pairs of functions)
Alternative PF calculation

\[(\pi_y, x R) \Join (\pi_z, x R) = \pi_{yz, x R}\]

\[\equiv \{ (29) ; (10) \text{ three times } \} \]

\[\langle y \cdot R \cdot x^\circ, z \cdot R \cdot x^\circ \rangle = yz \cdot R \cdot x^\circ\]

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

\[\langle y \cdot R \cdot x^\circ, z \cdot R \cdot x^\circ \rangle \subseteq yz \cdot R \cdot x^\circ\]

\[\equiv \{ \text{ “split twist” rule (31) — twice ; converses } \} \]

\[\langle y \cdot R \cdot x^\circ, id \rangle \cdot x \cdot R^\circ \cdot z^\circ \subseteq \langle y, x \cdot R^\circ \rangle \cdot z^\circ\]

\[\equiv \{ \text{ instances of split-fusion: (32) and (34) } \} \]

\[\langle y \cdot R \cdot x^\circ, x \cdot x^\circ \rangle \cdot x \cdot R \cdot z^\circ \subseteq \langle y, x \rangle \cdot R \cdot z^\circ\]

\[\equiv \{ \text{ instances of split-fusion: (33) and (34) } \} \]

\[\langle \langle y, x \rangle \cdot R \cdot x^\circ \rangle \cdot (x \cdot R \cdot z^\circ \rangle \subseteq \langle y, x \rangle \cdot R \cdot z^\circ\]

\[\equiv \{ (27) \} \]
PF calculation details

“Split twist” rule

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

Instances of (relational) split-fusion

- For simple (thus difunctional) \( S \):
  \[
  \langle R, T \rangle \cdot S = \langle R, T \cdot S \cdot S^\circ \rangle \cdot S \tag{32}
  \]
  \[
  \langle R, S \rangle \cdot S^\circ = \langle R \cdot S^\circ, S \cdot S^\circ \rangle \tag{33}
  \]

- Split pre-conditioning rule:
  \[
  \langle R, S \rangle \cdot \Phi = \langle R, S \cdot \Phi \rangle \equiv \Phi \text{ is coreflexive} \tag{34}
  \]
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):

Maier’s def. \[\equiv xy \cdot R \cdot x^\circ \cdot x \cdot R \cdot z^\circ \subseteq xy \cdot R \cdot z^\circ\]
\[\equiv \{ \text{swap } y \text{ and } z \text{ and take converses} \} \]
\[y \cdot R \cdot x^\circ \cdot x \cdot R \cdot xz^\circ \subseteq y \cdot R \cdot xz^\circ\]
\[\equiv \{ \text{ } R = R \cdot R^\circ \text{ since } R \text{ is coreflexive} \} \]
\[y \cdot R \cdot x^\circ \cdot \pi_1 \cdot xz \cdot R \cdot R^\circ \cdot xz^\circ \subseteq y \cdot R \cdot xz^\circ\]
\[\equiv \{ \text{ please turn over } \} \]
Finally, we go back to points (third step of a typical PF-transform argument):
MVDs PF-transformed

\[ \text{img}(xz \cdot R) \subseteq (\gamma_{y,x}R \cdot \pi_1)^{\circ} \cdot \gamma_{y,xz}R \]

\[ \equiv \{ \text{reverse PF-transform (for } R \text{ coreflexive, } xz \cdot R \text{ is simple) } \} \]

\[ \langle \forall \ k : k \ \text{img}(xz \cdot R)k : (\gamma_{y,x}R \cdot \pi_1)k = (\gamma_{y,xz}R)k \rangle \]

\[ \equiv \{ \text{reverse PF-transform of the image of } xz \cdot R \} \]

\[ \langle \forall \ k : \langle \exists \ t : t \in R : xz(t) = k \rangle : (\gamma_{y,x}R \cdot \pi_1)k = (\gamma_{y,xz}R)k \rangle \]

\[ \equiv \{ \text{rename } k := (b, a) \text{ and simplify} \} \]

\[ \forall \ a, b : \]

\[ \langle \exists \ t : t \in R : (x \ t) = a \land (z \ t) = b \rangle : \]

\[ (\gamma_{y,x}R) \ a = (\gamma_{y,xz}R)(a, b) \]

\[ \equiv \{ \text{recognize } (\gamma_{y,x}R)a \text{ as } Y(a) \} \]

Beeri, Fagin & Howard definition
Some MVD rules are hard to PF-transform, eg.

- **M5. Transitivity:**

  \[ R \quad x \rightarrow y \land y \rightarrow z \Rightarrow x \rightarrow (z - y) \quad (35) \]

- **M6. Pseudotransitivity:**

  \[ R \quad x \rightarrow y \land yw \rightarrow z \Rightarrow xw \rightarrow (z - yw) \quad (36) \]

**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.

\[ x R \rightarrow y \equiv f \cdot x R \rightarrow f \cdot y \leftarrow f \text{ is injective} \]

eg. structural FDs:

\[ x R \rightarrow y \equiv Fx FR \rightarrow Fy \]

eg. specific results on functional dependences on “the functions themselves”,

\[ f \rightarrow id \equiv f id \rightarrow g \]

etc.
Current work

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

\[ \ker R = (R \setminus R) \cap (R \setminus R)^o \]

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
Relationship between function \textit{divisibility} and the injectivity preorder: two preorders on functions

- “Left divisibility” — $g \sqsubseteq f$ iff exists $k$ such that
  \[ f = g \cdot k \]  
  \[(37)\]

- “Right divisibility” — $g \preceq f$ iff exists $k$ such that
  \[ f = k \cdot g \]  
  \[(38)\]

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

\[ f \leq g \]
\[
\equiv \quad \{ \text{FDs on functions: } f \leq g \equiv g^{id} \rightarrow f \text{; projections [4]} \}
\]
\[ f \cdot g^{\circ} \text{ is simple} \]
\[
\equiv \quad \{ \text{simple relations are fragments of functions (and vice versa)} \}
\]
\[ \langle \exists \ k \quad : \quad f \cdot g^{\circ} \subseteq k \rangle \]
\[
\equiv \quad \{ \text{“al-djabr” (shunting)} \}
\]
\[ \langle \exists \ k \quad : \quad f \subseteq k \cdot g \rangle \]
\[
\equiv \quad \{ \text{function equality} \}
\]
\[ g \leq f \]
Synergies with other CS disciplines

- **Bisimulations** — FD $d \xrightarrow{R} c$ 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.

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

  $$\text{Sort} = \left[\text{ordered}\right] \cdot \ker \text{bagify}$$

infer FD $\text{bagify} \xrightarrow{\text{Sort}} \text{bagify}$, 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
“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 resolver vna question por medios Geometricos, y resolverla por Algebra, siendo la misma Algebra sacada de la Geometria, ñ es cosa de admiraciõ.
“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, ñ es cosa de admiraciœ.
(...) De manera, que quien sabe por Algebra, sabe científicamente.

((...) In this way, who knows by Algebra knows scientifically)
R. Bird and O. de Moor.

*Algebra of Programming.*
C.A.R. Hoare, series editor.

Mark P. Jones.

Type classes with functional dependencies.

D. Maier.

*The Theory of Relational Databases.*

J.N. Oliveira.

<table>
<thead>
<tr>
<th>Context</th>
<th>Notation</th>
<th>PF transform</th>
<th>FDs</th>
<th>MVDs</th>
<th>Difficulties</th>
<th>Synergies</th>
<th>Conclusions</th>
<th>Epilogue</th>
</tr>
</thead>
</table>

Draft of paper in preparation.