Pointfree Foundations for (Generic) Lossless Decomposition

J.N. Oliveira
jno@di.uminho.pt

Techn. Report TR-HASLab:03:2011
Nov. 2011 [1]

HASLab - High-Assurance Software Laboratory
Universidade do Minho
Campus de Gualtar – Braga – Portugal
http://haslab.di.uminho.pt

[1] Document history: first version - Dec. 2009; updated: Nov. 2011

TR-HASLab:03:2011
Pointfree Foundations for (Generic) Lossless Decomposition
by J.N. Oliveira

Abstract

This report presents a typed, "pointfree" generalization of relational data dependency theory expressed not in the standard set-theoretic way, "à la Codd", but in the calculus of *binary* relations which, initiated by De Morgan in the 1860s, is the core of modern *algebra of programming*.

Contrary to the intuition that a binary relation is *just* a particular case of an n-ary relation, this report shows the effectiveness of the former in "explaining" and reasoning about the latter. Data dependency theory, which is central to relational database design, is addressed in pointfree calculational style instead of reasoning about (sets of) tuples in conventional "implication-first" logic style.

It turns out that the theory becomes more general, more structured and simpler. Elegant expressions replace lengthy formulæ and easy-to-follow calculations replace pointwise proofs with lots of "$\cdots$" notation, case analysis and natural language explanations for "obvious" steps. In particular, attributes are generalized to arbitrary (observation) functions and the principle of lossless decomposition is established for arbitrary such functions.

The report concludes by showing how the proposed generalization of data dependency theory paves the way to interesting synergies with other branches of computer science, namely formal modeling and transition systems theory. A number of open topics for research in the field are proposed as future work.

Pointfree Foundations for (Generic) Lossless Decomposition

J.N. Oliveira
jno@di.uminho.pt

Nov. 2011 †

Abstract

This report presents a typed, "pointfree" generalization of relational data dependency theory expressed not in the standard set-theoretic way, "à la Codd", but in the calculus of *binary* relations which, initiated by De Morgan in the 1860s, is the core of modern *algebra of programming*.

Contrary to the intuition that a binary relation is *just* a particular case of an n-ary relation, this report shows the effectiveness of the former in "explaining" and reasoning about the latter. Data dependency theory, which is central to relational database design, is addressed in pointfree calculational style instead of reasoning about (sets of) tuples in conventional "implication-first" logic style.

It turns out that the theory becomes more general, more structured and simpler. Elegant expressions replace lengthy formulæ and easy-to-follow calculations replace pointwise proofs with lots of "$\cdots$" notation, case analysis and natural language explanations for "obvious" steps. In particular, attributes are generalized to arbitrary (observation) functions and the principle of lossless decomposition is established for arbitrary such functions.

The report concludes by showing how the proposed generalization of data dependency theory paves the way to interesting synergies with other branches of computer science, namely formal modeling and transition systems theory. A number of open topics for research in the field are proposed as future work.

1 Introduction

In a paper addressing the influence of Alfred Tarski (1901-1983) in computer science, Feferman (2006) quotes the following sentence by his colleague John Etchemendy:

> *You see those big shiny Oracle towers on Highway 101? They would never have been built without Tarski's work on the recursive definitions of satisfaction and truth.*

†Document history: first version - Dec. 2009; updated: Nov. 2011

The 'big shiny Oracle towers' are nothing but the headquarters of Oracle Corporation, the giant database software provider sited in the San Francisco Peninsula.

Already in 2001 Bussche (2001) had shown that many of Tarski's ideas find application in database theory. Among these, he mentions cylindric algebras and relation algebras. Further back in time, Kanellakis (1990) had already commented on the relationship between Codd's theory and the former. Concerning the latter, Tarski *"single-handedly revived and advanced the 19th century work on binary relations by Peirce and Schröder"* (Feferman, 2006). This was part of his life-long pursuit in developing methods for elimination of quantifiers from logic expressions, an effort which ultimately lead to his *formalization of set theory without variables* (Tarski and Givant, 1987; Givant, 2006).

Further to (Kanellakis, 1990), both Bussche (2001) and Feferman (2006) try and find connections between Tarski's research and Codd's pioneering work on the thereafter called *relational data model* theory (Codd, 1970). Recall that, in standard relational data processing "à la Codd", real life objects or entities are recorded by assigning values to their observable properties or *attributes*. A database file is a collection of such attribute assignments, one per object, such that all values of a particular attribute (say i) are of the same type (say A_i). For n such attributes, a *relational database file* R can be regarded as a set of n-tuples, that is, $R \subseteq A_1 \times \ldots \times A_n$. A *relational database* is a collection of several such n-ary relations.

According to Kanamori (2003), it was Quine, in his 1932 Ph.D. dissertation, who showed how to develop the theory of n-ary relations for all n simultaneously, by defining ordered n-tuples in terms of the ordered pair. (Norbert Wiener is apparently the first mathematician to publicly identify, in the 1910s, n-ary relations with subsets of n-tuples.) Since the 1970s, the information system community is indebted to Codd for his pioneering work on the foundations of the *relational data model* theory (Codd, 1970).

Codd discovered and publicized procedures for constructing a set of simple n-ary relations which can support a set of given data and constructed an extension of the calculus of binary relations capable of handling most typical data retrieval problems. Since then, relational database theory has been thoroughly studied and found a distinguished place in computing curricula, supported by several textbooks among which (Maier, 1983; Ullman, 1988; Codd, 1990; O'Neil and O'Neil, 2001; Garcia-Molina et al., 2002) are widespread.

When software designers refer to the *relational calculus*, by default what is understood is the calculus of n-ary relations studied in logics and database theory (Maier, 1983), and not the above mentioned calculus of *binary* relations which was initiated by De Morgan in the 19c, was axiomatized by Tarski and others (Tarski and Givant, 1987; Givant, 2006) and eventually became the core of the *algebra of programming* (Aarts et al., 1992; Bird and de Moor, 1997; Backhouse, 2004) [1].

[1] The idea of encoding predicates in terms of relations was initiated by De Morgan in the 1860s and followed by Peirce who, in the 1870s, found interesting equational laws of the calculus of binary relations (Maddux, 1991; Pratt, 1992). The pointfree nature of the notation which emerged from this embryonic work was later further exploited by Tarski and his students (Tarski and Givant, 1987; Givant, 2006). In the 1980's, Freyd and Scedrov (1990) developed the notion of an *allegory* (a category whose morphisms are partially ordered) which finally accommodates the binary relation calculus as we understand it today. Interestingly enough, Freyd and Scedrov (1990) develop (in the first part of the book) a categorial theory of tables and relations based on *monic n-tuples* which bears a strong relationship to the

The common understanding is that binary relations are just n-ary relations, for $n = 2$, and so there seems to be little point in explaining n-ary relational theory in terms of binary relations. As a matter of fact, when Codd (1970) talks about the binary relation representation of an n-ary relation, one has the feeling that there are more disadvantages than advantages in such a representation. Moreover, further attempts to impose a binary relation data model such as in eg. MDV (Jones et al., 1985) and AW/1 (Welsh, 1989) have not gained widespread adoption.

Contrary to such common understanding, this report aims at providing evidence that refactoring n-ary relational theory in terms of binary relations is worthwhile. This is nothing but following the *exercise* proposed by Bussche (2001):

> *We conclude that Tarksi produced two alternatives for Codd's relational algebra: cylindric set algebra, and relational algebra with pairing* [...] *For example, we can represent the ternary relation* $\{(a, b, c), (d, e, f)\}$ *as* $\{(a, (b, c)), (d, (e, f))\}$. *Using such representations, we leave it as an exercise to the reader to simulate Codd's relational algebra in RA*$^+$ [relational algebra with pairing].

In this report we carry out this exercise with respect to *data dependency* theory. We will show that Tarskian binary relational algebra is directly applicable to calculating with data dependencies in a rather advantageous way.

Outside the database context, functional data dependencies have been of help in solving type system ambiguities in modern functional programming languages such as Haskell (Jones, 2000). The current report provides for another cross-breeding between the areas of database programming and functional programming: our approach to reasoning about data dependencies is based on the same calculus which — in a so-called *pointfree* style — is used to reason (relationally) about functional programs (Bird and de Moor, 1997).

Despite its 19th century ancestry and visibility since John Backus' Turing Award Lecture (1978), pointfree reasoning is not yet widespread. As most theories in computing, classical relational database theory is pointwise. This leads to lengthy formulæ and proofs with lots of "..." notation, case analyzes and natural language explanations for "obvious" steps. We show that the adoption of the (pointfree) binary relation calculus is beneficial in several respects. First, the fact that pointfree notation abstracts from "points" or variables makes the reasoning more compact and effective. Second, proofs are performed by easy-to-follow calculations. Third, one is able to generalize the original theory, as will happen with our generalization of attributes to arbitrary (suitably typed) functions in functional dependencies and multi-valued dependencies.

1.1 Paper structure

This report is laid out as follows. The section which follows provides some motivation for the *pointfree transform*. Then we introduce the standard notions of *functional dependency* (FD) and *multi-valued dependency* (MVD) and revise the pointfree theory of functions and binary relations. Both worlds are combined in section 6, where FDs are presented in the pointfree style. (The pointfree transform of MVDs is deferred to section 10.) Section 7 shows that injectivity is *what matters* in FD reasoning. Sections 8 to 12 provide calculational proofs for the foundations of data dependency theory, including the Armstrong-axioms and the theorem of lossless

approach put forward in the current report.

decomposition. The remainder of the report presents conclusions and prospect for future work. The reader is referred to the appendices for some useful (but not mainstream) results.

2 On the pointfree transform

Science is about understanding how things work and technology is about ensuring that some desirable things happen reliably. Properties of real-world entities are identified which, once expressed by mathematical formulæ, become abstract models which can be queried and reasoned about. This universal problem-solving strategy often raises a kind of *notation* conflict between *descriptiveness* (ie., adequacy to describe domain-specific objects and properties, inc. diagrams or other graphical objects) and *compactness* (as required by algebraic reasoning and solution calculation).

Database design is paradigmatic in this respect. The complex structure of the objects and entities to be modeled demands much on descriptiveness, and thus the need for graphical notations (eg. entity-relationship diagrams (Chen, 1976), UML (Booch et al., 1999) etc.) and verbose programming notations such as Cobol (1974) and SQL (1992). When it comes to reasoning about the semantics of such diagrams or notations, predicate/temporal logics and naïve set theory are the most common formal resources.

However, such *pointwise* notations involving operators as well as variable symbols, logical connectives, quantifiers, etc. are not agile enough. This kind of notational problem is not new in engineering mathematics. Elsewhere in physics and several branches of engineering, people have learned to overcome it by changing the "mathematical space", for instance by moving (temporarily) from the t-space (t for time) to the s-space in the *Laplace transformation*. Quoting (Kreyszig, 1988), p.242:

> *The Laplace transformation is a method for solving differential equations (...) The process of solution consists of three main steps:*
>
> **1st step.** *The given "hard" problem is transformed into a "simple" equation (*subsidiary equation*).*
>
> **2nd step.** *The subsidiary equation is solved by* ***purely algebraic*** *manipulations.*
>
> **3rd step.** *The solution of the subsidiary equation is transformed back to obtain the solution of the given problem.*
>
> *In this way the Laplace transformation reduces the problem of solving a differential equation to an* ***algebraic problem****.*

The *pointfree transform* (*PF-transform* for short) adopted in this report is at the heels of this old reasoning technique. Standard set-theory-formulated database concepts are regarded as "hard" problems to be transformed into "simple", *subsidiary equations* dispensing with points and involving only binary relation concepts. As in the Laplace transformation, these are solved by *purely algebraic* manipulations and the outcome is mapped back to the original (descriptive) mathematical space wherever required.

Note the advantages of this two-tiered approach: intuitive, domain-specific descriptive formulæ are used wherever the model is to be "felt" by people. Such formulæ are transformed into a more *elegant*, simple and compact — but also more cryptic — algebraic notation whose single purpose is easy manipulation.

3 What is a data dependency?

In an n-ary relation, attribute *names* are more expressive than natural number indices as attribute identifiers. The enumeration of all attribute names in a database relation, for instance

$$S \quad = \quad \{\text{PILOT}, \text{FLIGHT}, \text{DATE}, \text{DEPARTS}\} \tag{1}$$

concerning an airline scheduling system [2], is a finite set called the relation's *scheme*. This scheme captures the *syntax* of the data. What about *semantics*?

Even non-experts in airline scheduling will accept attaching the following "business" rule to schema (1): *A single pilot is assigned to a given flight, on a given date*. This restriction is an example of a so-called *functional dependency* (FD) among attributes, which can be stated more formally as follows: *attribute* PILOT *is functionally dependent on* FLIGHT *and* DATE. In the standard practice, this will be abbreviated by writing

$$\text{FLIGHT DATE} \rightarrow \text{PILOT}$$

which has the following, alternative reading: FLIGHT *and* DATE *functionally determine* PILOT. Another FD in this example is

$$\text{FLIGHT} \rightarrow \text{DEPARTS} \tag{2}$$

which captures the fact that a given flight always departs at the same time.

The addition of functional dependencies to a relational schema is comparable to the addition of axioms to an algebraic signature, eg. axioms such as $pop(push(a, s)) = s$ adding semantics to the syntax of a stack datatype involving operators $push$ and pop. How does one reason about such functional dependency-based semantics of data?

Functional dependencies (FD) are central to standard relational database theory, where they addressed in an axiomatic way based on the definition of FD-satisfiability which follows (Maier, 1983).

Definition 1 *Given subsets $x, y \subseteq S$ of the relation scheme S of a n-ary 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, that is,*

$$\langle \forall\, t, t' \;:\; t, t' \in R : \quad t[x] = t'[x] \quad \Rightarrow \quad t[y] = t'[y] \;\rangle \tag{3}$$

(Notation $t[x]$ means "the values in t of the attributes in x" and will be scrutinized in the sequel.)
□

[2] This well-known example is taken from (Maier, 1983).

Formula (3), with its logical implication inside a two-dimensional universal quantification, is not particularly agile. Designs involving many FDs at the same time would be hard to reason about if based on (3) alone. This situation gets worse when the more general (and useful) concept of a *multi-valued* dependency (MVD) is addressed. This is defined by Maier (1983) as follows:

Definition 2 *Given subsets $x, y \subseteq S$ of the relation scheme S of an n-ary relation R, this relation is said to satisfy the* 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 $z = S - xy$, that is,*

$$\begin{array}{lll} \langle \forall\, t, t' \;:\; t, t' \in R : & t[x] = t'[x] & \rangle \\ & \Downarrow & \\ \langle \exists\, t'' \;:\; t'' \in R : & t[xy] = t''[xy] \wedge & \rangle \\ & t''[z] = t'[z] & \end{array} \tag{4}$$

holds. □

Beeri et al. (1977) give the alternative definition which follows:

Definition 3 *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:*

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

□

Notation is overly simplified in this definition. In fact, function Y should be equipped with two extra parameters, attribute k and relation R itself. So, the overall definition should be

$$\langle \forall\, a, b \;:\; \langle \exists\, t \;:\; t \in R :\; t[xz] = ab \rangle \;:\; Y_{R,x}(a) = Y_{R,xz}(ab) \rangle \tag{5}$$

as illustrated in the following picture:

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

(6)

Despite its complexity, the MVD concept is central to one of the main ingredients of relational data refinement — that of *loss-less decomposition* (Maier, 1983). Its complexity has lead database theorists to develop FD/MVD-theory in an axiomatic style, based on the so-called *Armstrong axioms*, which can be used as inference rules for such dependencies. Equivalent axioms have been found which make FD/MVD checking more efficient. However, most database practitioners use this theory while ignoring its foundations. Even textbooks such as (Ullman, 1988) and (Garcia-Molina et al., 2002) don't go very deep into the subject. Can this be accepted?

To show that FD/MVD-theory can be re-factored into a generic, elegant and easy-to-follow body of knowledge is the main motivation of this report. The work stems from the author's own experience and method for relational data modeling (Oliveira, 1992; Alves et al., 2005; Cunha et al., 2006; Oliveira, 2008), itself a way to dispense with such a theory provided designs are expressed in model-oriented formal specification notations such as, eg., VDM-SL (ISO/IEC 13817-1) (International Organization for Standardization, 1996).

Both in the current report and in (Oliveira, 2008) the approach is the same — the avoidance of complex pointwise formulæ and proofs via the pointfree algebra of programming (Bird and de Moor, 1997). It turns out that the theory becomes more general and considerably simpler, thanks to the calculus of *simplicity* and *coreflexivity*. (Details about this terminology will be presented shortly.)

We will start by reviewing some basic principles. Qualifier "functional" in "functional dependency" stems from "function", of course. (Garcia-Molina et al., 2002) write:

> ***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 order to prepare the reader for our own answer to the question above, our first effort goes into making sure we have a clear idea of "what a function is".

4 What is a function? — the "Leibniz view"

A function f is a special case of binary relation satisfying two main properties [3]:

- *"Left" uniqueness*

$$b\ f\ a\ \wedge\ b'\ f\ a \quad \Rightarrow \quad b = b' \tag{7}$$

- *Leibniz principle*

$$a = a' \quad \Rightarrow \quad f\ a = f\ a' \tag{8}$$

It can be shown (see section 5) that establishing properties (7, 8) is the same as saying that functions are *simple* and *entire* relations [4], respectively:

- f is simple:

$$img\ f \quad \subseteq \quad id \tag{9}$$

[3] Following a widespread convention, functions will be denoted by lowercase characters (eg. f, g, ϕ) or identifiers starting with lowercase characters, and function application will be denoted by juxtaposition, eg. $f\ a$ instead of $f(a)$.

[4] This terminology is borrowed from (Freyd and Scedrov, 1990). Being simple means being "functional"; being entire means being total.

- f is entire:

$$id \quad \subseteq \quad ker\, f \tag{10}$$

Formulæ (9,10) are examples of *pointfree* notation in which *points* — eg. a, a', b, b' in (7,8) — disappear. For instance, instead of writing $a = a'$, one resorts to the identity relation id which relates a and a' if and only if they are the same.

In order to parse such *compressed* formulæ we need to understand the meaning of expressions such as *ker* f (read: "*kernel* of f") and *img* f (read: "*image* of f"),

$$ker\, R \quad = \quad R^\circ \cdot R \tag{11}$$
$$img\, R \quad = \quad R \cdot R^\circ \tag{12}$$

whose definitions involve two standard binary relation combinators (Bird and de Moor, 1997): *converse* (R°) and *composition* ($R \cdot S$). The former converts a relation R into R° such that $a(R^\circ)b$ holds iff bRa holds. (Following the standard practice, we write bRa to mean that pair (b, a) is in R. Wherever R is a function f, bfa means the same as $b = f\ a$.) The latter *(composition)* is defined in the usual way: given binary relations

$$B \xleftarrow{R} A \xleftarrow{S} C \tag{13}$$

assertion $b(R \cdot S)c$ holds wherever there exist one or more mediating $a \in A$ such that conjunction $bRa\ \wedge\ aSc$ holds.

Converse commutes with composition in a contravariant way,

$$(R \cdot S)^\circ = S^\circ \cdot R^\circ \tag{14}$$

and so image and kernel commute via converse:

$$ker\,(R^\circ) \quad = \quad img\, R \tag{15}$$
$$img\,(R^\circ) \quad = \quad ker\, R \tag{16}$$

As in (9,10), the underlying partial order on relations is written $R \subseteq S$, meaning

$$R \subseteq S \quad \Leftrightarrow \quad \langle \forall\, b, a \ ::\ bRa \Rightarrow bSa \rangle \tag{17}$$

for all suitably typed a and b. All relational combinators presented so far are monotonic with respect to $\subseteq$.

The *simple* and *entire* classes of relation mentioned above are part of a wider binary relation taxonomy, depicted in figure 1, whose four top-level classification criteria are captured by table

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

(18)

where R is said to be *reflexive* iff it is at least the identity ($id \subseteq R$) and it is said to be *coreflexive* (or a *partial identity*) iff it is at most the identity ($R \subseteq id$).

Coreflexive relations are fragments of the identity relation which can be used to model predicates or sets. The meaning of a predicate ϕ is the coreflexive relation

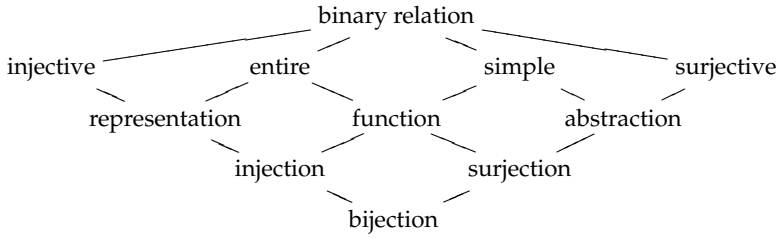


Figure 1: Binary relation taxonomy

$[\![\phi]\!]$ such that $b[\![\phi]\!]a \Leftrightarrow (b = a) \wedge (\phi\, a)$. We often denote $[\![\phi]\!]$ by uppercase Φ. This is the simple relation that maps every a which satisfies ϕ onto itself and is undefined otherwise. The meaning of a set S is the meaning of its *characteristic* predicate $[\![\lambda a.a \in S]\!]$, that is,

$$b[\![S]\!]a \quad \Leftrightarrow \quad (b = a) \ \wedge \ a \in S \tag{19}$$

Wherever clear from the context, we will drop brackets $[\![\ ]\!]$.

Standard set theory can thus be expressed in terms of coreflexives, the algebra of which is therefore very rich in useful properties, see eg. (Bird and de Moor, 1997; Backhouse, 2004). For instance, every coreflexive Φ is symmetric ($\Phi^\circ = \Phi$) and pre/post-conditioning are *closure* operators (Backhouse, 2004),

$$R \cdot \Phi \subseteq S \quad \Leftrightarrow \quad R \cdot \Phi \subseteq S \cdot \Phi \tag{20}$$

$$\Phi \cdot R \subseteq S \quad \Leftrightarrow \quad \Phi \cdot R \subseteq \Phi \cdot S \tag{21}$$

for arbitrary R, S and coreflexive Φ.

Set union, intersection etc are thus modeled by their obvious counterparts at coreflexive level. A more interesting example is the relational equivalent to the *cross-product* of two sets S and T,

$$[\![S]\!] \otimes [\![T]\!] \quad = \quad \{(b, a) \mid b \in S \ \wedge \ a \in T\}$$

where

$$R \otimes S \quad \stackrel{\text{def}}{=} \quad R \cdot \top \cdot S \tag{22}$$

Above, $\top$ is the largest relation of its type, that is, the relation which is such that $b \top a$ holds for every (suitably typed) a and b.

Before embarking on converting (3,4) and (5) into pointfree notation, let us see an alternative view of functions better suited for calculation.

5 What is a function? — the "Galois view"

To say *"f is a function"* is *equivalent* to stating any of the two Galois connections which follow [5]:

$$f \cdot R \subseteq S \quad \Leftrightarrow \quad R \subseteq f^\circ \cdot S \tag{23}$$

[5] These equivalences are popularly known as the "shunting rules" (Bird and de Moor, 1997).

$$R \cdot f^\circ \subseteq S \quad \Leftrightarrow \quad R \subseteq S \cdot f \tag{24}$$

As a warming-up exercise, let us check one of these, say (23). (The whole picture can be found in eg. (Hoogendijk, 1997; Bird and de Moor, 1997; Backhouse, 2004).) That f being simple and entire (9, 10) *implies* equivalence (23) can be proved by circular implication:

$$\begin{array}{ll} & f \cdot R \quad \subseteq \quad S \\ \Rightarrow & \quad \{ \text{ monotonicity of composition } \} \\ & f^\circ \cdot f \cdot R \quad \subseteq \quad f^\circ \cdot S \\ \Rightarrow & \quad \{ \ f \text{ is entire (10) } \} \\ & R \quad \subseteq \quad f^\circ \cdot S \\ \Rightarrow & \quad \{ \text{ monotonicity of composition } \} \\ & f \cdot R \quad \subseteq \quad f \cdot f^\circ \cdot S \\ \Rightarrow & \quad \{ \ f \text{ is simple (9) } \} \\ & f \cdot R \quad \subseteq \quad S \end{array}$$

That (23) *implies* that f is entire and simple can be checked via instantiation $R, S := id, f$ (left-cancellation) and $S, R := id, f^\circ$ (right-cancellation), respectively.

The following are easy-to-show outcomes of Galois connections (23, 24) relevant to this report. First of all, functions are *difunctional* [6]:

$$f \cdot f^\circ \cdot f \quad = \quad f \tag{25}$$

Secondly, kernels of functions are equivalence relations. That is, besides reflexivity (10), one has

$$(\mathit{ker}\ f) \cdot \mathit{ker}\ f = \mathit{ker}\ f \tag{26}$$

$$(\mathit{ker}\ f)^\circ = \mathit{ker}\ f \tag{27}$$

Finally, pre/post-composition with functional kernels are *closure* operations:

$$S \cdot \mathit{ker}\ f \subseteq R \cdot \mathit{ker}\ f \quad \Leftrightarrow \quad S \subseteq R \cdot \mathit{ker}\ f \tag{28}$$

$$(\mathit{ker}\ f) \cdot S \subseteq (\mathit{ker}\ f) \cdot R \quad \Leftrightarrow \quad S \subseteq (\mathit{ker}\ f) \cdot R \tag{29}$$

(See section C.2 in the appendix.)

Function converses enjoy a number of properties of which the following is singled out because of its rôle in pointwise-pointfree conversion,

$$b(f^\circ \cdot R \cdot g)a \quad \Leftrightarrow \quad (f\ b)R(g\ a) \tag{30}$$

where f and g are functions [7]. The interested reader may wish to resort to (30) in checking the equivalence between (7, 8) and (9, 10), respectively.

The following precedence order on prefix or infix relational operators

$$\_^\circ > \{\mathit{ker}\,, \mathit{img}\} > (\cdot) > \cap > \cup$$

is assumed in order to save parentheses in relational expressions. Assuming this, expression $R \cdot \mathit{ker}\ S^\circ \cap T \cup V$ will abbreviate $((R \cdot (\mathit{ker}\ (S^\circ))) \cap T) \cup V$, for instance.

[6] See appendix C.2 for a generalization of this concept.

[7] See eg. (Backhouse and Backhouse, 2004).

6 FD-satisfiability in pointfree style

6.1 Attributes are functions

Let R be a n-ary relation with schema S, t be a tuple in R and a be an attribute in S. Notation $t[a]$ was adopted in (3) to mean "the value exhibited by attribute a in tuple t". Tuples can be regarded as inhabitants of n-dimensional Cartesian products, either in the standard format (eg. $A_1 \times \cdots \times A_n$) or in "rich syntax format" equipped with tuple constructors and selector (field) names, one per attribute. For instance, relational scheme (1) can be modeled in the Haskell type system by declaring (assuming types `Pilot`, `Flight`, `Date`, `Departs` declared elsewhere)

```
data S = S {
         pilot :: Pilot,
         flight :: Flight,
         date :: Date,
         departs :: Departs
        }
```

or simply by typing

```
data S = S ( Pilot , Flight , Date , Departs )
```

which is Cartesian product `Pilot` $\times$ `Flight` $\times$ `Date` $\times$ `Departs` in Haskell-speak.

From our perspective, it doesn't matter which of these alternatives is adopted, since in *both cases* attributes are modeled by *functions*. For instance, function `pilot :: S -> Pilot` gives access to the values of attribute PILOT in the first model of `S`, in the same way projection function $\pi_1(a, b) \stackrel{\text{def}}{=} a$ extracts the same data from the Cartesian tuples of the second model. In summary, *attributes are* (projection) *functions* [8]. Since this view extends smoothly to a collection x of attributes (please wait until section 7.3 for some technical details), we can convert (3) into

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

Assuming the universal quantification implicit, we reason:

$$\begin{array}{ll} & t \in R \ \wedge\ t' \in R \ \wedge\ (x\ t) = (x\ t') \quad \Rightarrow \quad (y\ t) = (y\ t') \\ \Leftrightarrow & \quad \{\ \text{let } f, R, g := x, id, x \text{ in (30) — twice}\ \} \\ & t \in R \ \wedge\ t' \in R \ \wedge\ t(x^\circ \cdot x\ )t' \quad \Rightarrow \quad t(y^\circ \cdot y\ )t' \\ \Leftrightarrow & \quad \{\ (19) \text{ twice}\ \} \\ & t = u \ \wedge\ t[\![R]\!]u \ \wedge\ t' = u' \ \wedge\ t'[\![R]\!]u' \ \wedge\ t(x^\circ \cdot x\ )t' \quad \Rightarrow \quad t(y^\circ \cdot y\ )t' \\ \Leftrightarrow & \quad \{\ \wedge \text{ is commutative; substitution of equals for equals; converse}\ \} \\ & t[\![R]\!]u \ \wedge\ u(x^\circ \cdot x\ )u' \ \wedge\ u'[\![R]\!]^\circ t' \quad \Rightarrow \quad t(y^\circ \cdot y\ )t' \\ \Leftrightarrow & \quad \{\ \text{going pointfree via composition and relation inclusion (17)}\ \} \end{array}$$

[8] Attributes in this view correspond to *columns* in the monic n-tuple categorial approach of Freyd and Scedrov (1990).

$$
\begin{array}{ll}
 & [\![R]\!] \cdot (x^{\circ} \cdot x) \cdot [\![R]\!]^{\circ} \subseteq y^{\circ} \cdot y \\
\Leftrightarrow & \quad \{ \text{ rules (23) and (24) } \} \\
 & y \cdot [\![R]\!] \cdot x^{\circ} \cdot x \cdot [\![R]\!]^{\circ} \cdot y^{\circ} \subseteq id \\
\Leftrightarrow & \quad \{ \text{ converse versus composition (14) followed by (12) } \} \\
 & img\,(y \cdot [\![R]\!] \cdot x^{\circ}) \subseteq id
\end{array}
$$

In summary: a *n-ary* relation R as in definition 1 satisfies functional dependency $x \rightarrow y$ iff *binary* relation

$$y \cdot [\![R]\!] \cdot x^{\circ} \tag{31}$$

is simple, cf. (9) [9].

6.2 Functional dependencies in general

Our approach to the FD concept starts from the observation that coreflexive relation $[\![R]\!]$ and projection functions x and y in (31) can be generalized to arbitrary binary relations and functions. This leads to the more general definition which follows. (The use of "$\rightharpoonup$" instead of "$\rightarrow$" is intentional: it is an indication that we are moving from the restricted to the generic notion.)

Definition 4 *Binary relation* $B \xleftarrow{R} A$ *is said to satisfy the "*$f \rightharpoonup g$*" functional dependency — written* $f \overset{R}{\rightharpoonup} g$ *— iff* $g \cdot R \cdot f^{\circ}$ *in*

$$
\begin{array}{ccc}
B & \xleftarrow{R} & A \\
{\scriptstyle g}\downarrow & & \downarrow{\scriptstyle f} \\
C & \xleftarrow[g \cdot R \cdot f^{\circ}]{} & D
\end{array}
$$

is simple (9). Equivalent definitions are

$$f \overset{R}{\rightharpoonup} g \quad \Leftrightarrow \quad R \cdot (\ker f) \cdot R^{\circ} \subseteq \ker g \tag{32}$$

and

$$f \overset{R}{\rightharpoonup} g \quad \Leftrightarrow \quad \ker (f \cdot R^{\circ}) \subseteq \ker g \tag{33}$$

thanks to (11, 14, 23) and (24).

Function f (resp. g) will be mentioned as the left side *or* antecedent *(resp.* right side *or* consequent*) of FD* $f \overset{R}{\rightharpoonup} g$. □

[9] It can be observed that our reasoning above instantiates rule (141) of the *PF-transform* (given in appendix A) which generalizes binary relation inclusion (17).

6.3 Trivial properties and examples

In contrast with (3), equations (32, 33) are easy to reason about, as the reader may check by proving the following, elementary properties, which hold for all R, f, g of appropriate type:

$$f \stackrel{\perp}{\leftarrow} g$$

(where $\perp$ denotes the empty relation)

$$f \stackrel{R}{\leftarrow} ! \tag{34}$$

(where $1 \xleftarrow{!} B$ denotes the unique, constant function of its type and 1 denotes the singleton type)

$$id \stackrel{R}{\leftarrow} id \quad \Leftrightarrow \quad R \text{ is simple} \tag{35}$$

$$f \stackrel{R}{\leftarrow} f \quad \Leftarrow \quad R \subseteq id \tag{36}$$

An immediate consequence of (36) is

$$f \stackrel{id}{\leftarrow} f \tag{37}$$

Back to pointwise notation, (32) and (33) expand to the following generalization of (3):

$$\langle \forall\, b, b' \,:\, \langle \exists\, a, a' \,::\, b\, R\, a \,\wedge\, b'\, R\, a' \,\wedge\, f\, a = f\, a' \rangle :\, g\, b = g\, b' \rangle \tag{38}$$

Even inepter than (3) for calculation purposes, formula (38) is interesting for its appeal to intuition, as the reader may feel by checking that $f \stackrel{R}{\leftarrow} g$ holds for R any of the relations tabulated by the a and b columns of

b	a	$f\ a = a^2$	$g\ b = rem\ b\ 3$
2	-1	1	2
5	-1	1	2
17	1	1	2
10	-2	4	1
4	-2	4	1
1	2	4	1

and

b	a	$f = id$	$g = \pi_1$
$(1,2)$	-1	-1	1
$(1,10)$	-1	-1	1
$(0,0)$	1	1	0
$(5,6)$	-2	-2	5
$(5,0)$	-2	-2	5
$(1,2)$	2	2	1

In the sequel, we proceed to another, more elegant pointfree statement of FD-satisfiability which takes advantage of the rich mathematics underlying *injectivity*, one of the classification criteria of the taxonomy of figure 1.

7 The role of injectivity

7.1 Ordering relations by injectivity

It can be observed that what matters about f and g in (32) is their "degree of injectivity" as measured by *ker* f and *ker* g — recall table (18) — in opposite directions:

more injective f and less injective g will strengthen a given FD $f \stackrel{R}{\leftarrow} g$. An extreme case is $f = id$ and $g = !$, since functional dependency $id \stackrel{R}{\leftarrow} !$ will always hold for any R, cf. (34).

In order to *measure* injectivity in general we define the *injectivity preorder* on relations as follows:

$$R \leq S \quad \Leftrightarrow \quad ker\, S \subseteq ker\, R \tag{39}$$

that is, $R \leq S$ means R is *less* injective than S [10]. To be more precise, we should write *"less injective or more defined"* since *ker* measures both properties, cf. (18) and

$$R \subseteq S \quad \Rightarrow \quad S \leq R \tag{40}$$

(see appendix D). In case of functions, $f \leq g$ unambiguously means that f is less injective than g [11].

Limit cases of injectivity (or the lack of it) are apparent from

$$! \quad \leq \quad R \quad \leq \quad \bot$$

since the kernel of function ! is the top (ie. largest) relation of its type,

$$!^\circ \cdot ! \quad = \quad \top \tag{41}$$

and that of the empty relation is empty ($\bot^\circ \cdot \bot = \bot$).

The fact pre-composition respects the injectivity preorder,

$$R \leq S \quad \Rightarrow \quad R \cdot T \leq S \cdot T \tag{42}$$

is easy to prove [12]:

$$\begin{array}{ll} & R \leq S \\ \Leftrightarrow & \quad \{ \text{ (39) and (11) } \} \\ & S^\circ \cdot S \quad \subseteq \quad R^\circ \cdot R \\ \Rightarrow & \quad \{ \text{ monotonicity of } (T^\circ \cdot\ ) \text{ and } (\ \cdot T) \ \} \\ & T^\circ \cdot S^\circ \cdot S \cdot T \quad \subseteq \quad T^\circ \cdot R^\circ \cdot R \cdot T \\ \Leftrightarrow & \quad \{ \text{ (14) twice, followed by (11) and (39) } \} \\ & R \cdot T \quad \leq \quad S \cdot T \end{array}$$

The following property involving two functions ordered by injectivity

$$(ker\, f) \cap (S \cdot ker\, g) = (ker\, f \cap S) \cdot ker\, g \quad \Leftarrow \quad f \leq g \tag{43}$$

will be useful in the sequel. It instantiates the more general fact (148) proved in appendix B.

[10] Note that R and S must have the same source type but don't need to share the same target datatype.

[11] This restriction of $\leq$ to functions is referred to as the *collapsing order* in (Matsuda et al., 2007).

[12] This proof instantiates a more general construction presented in appendix D.

7.2 FD defined via the injectivity preorder

The close relationship between FDs and injectivity of observations is well captured by the following re-statement of (33) in terms of (39):

$$f \stackrel{R}{\rightharpoonup} g \quad \Leftrightarrow \quad g \;\leq\; f \cdot R^{\circ} \tag{44}$$

For its conciseness, this definition of FD is very amenable to calculation, as is illustrated below by proving two facts which will be useful in the sequel:

$$f \stackrel{S \cdot R}{\rightharpoonup} h \quad \Leftarrow \quad f \stackrel{R}{\rightharpoonup} g \quad \wedge \quad g \stackrel{S}{\rightharpoonup} h \tag{45}$$

$$f \stackrel{R}{\rightharpoonup} g \quad \Leftarrow \quad f \stackrel{S}{\rightharpoonup} g \quad \wedge \quad R \subseteq S \tag{46}$$

The first shows how two FDs with matching antecedent / consequent functions yield a composite FD, cf.

$$\begin{array}{ll} & f \stackrel{R}{\rightharpoonup} g \quad \wedge \quad g \stackrel{S}{\rightharpoonup} h \\ \Leftrightarrow & \quad \{ \text{ (44) twice } \} \\ & g \leq f \cdot R^{\circ} \quad \wedge \quad h \leq g \cdot S^{\circ} \\ \Rightarrow & \quad \{ \text{ } \leq\text{-monotonicity of } (\ \cdot S^{\circ}) \text{ (42) followed by (14) } \} \\ & g \cdot S^{\circ} \leq f \cdot (S \cdot R)^{\circ} \quad \wedge \quad h \leq g \cdot S^{\circ} \\ \Rightarrow & \quad \{ \text{ } \leq\text{-transitivity } \} \\ & h \leq f \cdot (S \cdot R)^{\circ} \\ \Leftrightarrow & \quad \{ \text{ (44) again } \} \\ & f \stackrel{S \cdot R}{\rightharpoonup} h \end{array}$$

Fact (46) states that FD-satisfiability is downward-closed, cf.

$$\begin{array}{ll} & R \subseteq S \;\wedge\; f \stackrel{S}{\rightharpoonup} g \\ \Leftrightarrow & \quad \{ \text{ converses and (44) } \} \\ & R^{\circ} \subseteq S^{\circ} \;\wedge\; g \leq f \cdot S^{\circ} \\ \Rightarrow & \quad \{ \text{ monotonicity of } (f \cdot\ ) \text{ and (40) } \} \\ & f \cdot S^{\circ} \leq f \cdot R^{\circ} \;\wedge\; g \leq f \cdot S^{\circ} \\ \Rightarrow & \quad \{ \text{ } \leq\text{-transitivity } \} \\ & g \leq f \cdot R^{\circ} \\ \Leftrightarrow & \quad \{ \text{ (44) } \} \\ & f \stackrel{R}{\rightharpoonup} g \end{array}$$

Note in passing that (37) and (45) together suggest that we can build a category whose objects are functions f, g, etc. and whose arrows $f \xrightarrow{R} g$ are relations which satisfy $f \stackrel{R}{\rightharpoonup} g$.

7.3 Simultaneous observations

In the same way x and y in (3) may involve more that one observable attribute, we would like f and g in (32) to involve more than one *observation* function. Multiple observations add more detail and so are likely to be more injective. The relational *split* combinator $\langle\ ,\ \rangle$ — also termed *fork* [13] and defined by

$$(a,b)\langle R,S\rangle c \quad \Leftrightarrow \quad a\ R\ c\ \wedge\ b\ S\ c \tag{47}$$

— captures this effect, and facts

$$R \leq \langle R,S\rangle \quad \text{and} \quad S \leq \langle R,S\rangle \tag{48}$$

are easy to check by recalling

$$ker\ \langle R,S\rangle \quad = \quad (ker\ R) \cap (ker\ S) \tag{49}$$

which stems from equality

$$\langle R,S\rangle^{\circ} \cdot \langle X,Y\rangle \quad = \quad (R^{\circ} \cdot X) \cap (S^{\circ} \cdot Y)$$

proved by Bird and de Moor (1997). Moreover, (48) is nothing but left-cancellation of Galois connection

$$\langle R,S\rangle \leq T \quad \Leftrightarrow \quad R \leq T\ \wedge\ S \leq T \tag{50}$$

which stems from the one underlying $\cap$ (see appendix D) and can be used to state other facts, eg.

$$\langle R,!\rangle \leq R \tag{51}$$

The anti-symmetric closure of $\leq$ yields an equivalence relation

$$R \simeq S \Leftrightarrow ker\ R = ker\ S$$

which is such that, for instance, $! \simeq \top$ holds. We also have

$$S \leq R \quad \Leftrightarrow \quad \langle S,R\rangle \simeq R \tag{52}$$

The following equivalences will be relevant in the sequel, for suitably typed R, S and T:

$$R \quad \simeq \quad \langle R,R\rangle \tag{53}$$

$$\langle R,S\rangle \quad \simeq \quad \langle S,R\rangle \tag{54}$$

$$\langle T,\langle R,S\rangle\rangle \quad \simeq \quad \langle\langle T,R\rangle,S\rangle \tag{55}$$

The *product* of two relations,

$$(a,b)(R \times S)(c,d) \quad \Leftrightarrow \quad a\ R\ c\ \wedge\ b\ S\ d \tag{56}$$

is a special case of *split*. Thanks to these two constructs, we can elaborate on antecedents and consequents of FDs and write, for instance, $\langle f,h\rangle \stackrel{R}{\rightarrow} g$ or $f \times h \stackrel{R}{\rightarrow} g$, both expressing two simultaneous observations on the antecedent (similarly for the consequent) of a FD.

[13] This is the *pairing operation* which Tarski found missing in relation algebra (Tarski and Givant, 1987; Bussche, 2001). Its addition has lead to the study of so-called *fork algebras* (Veloso and Haeberer, 1991).

7.4 FDs on functions

Since attributes in the n-ary relational database model are (projection) functions, we will be particularly interested in comparing functions for their injectivity. Recall that the kernel of a function is always reflexive (10). So, restricted to functions, the $\leq$ ordering is such that, for all f,

$$! \quad \leq \quad f \quad \leq \quad id \tag{57}$$

and

$$f \simeq id \quad \Leftrightarrow \quad f \text{ is an injection (figure 1)}$$

Given function f, any function $\overline{f}$ such that

$$id \leq \langle f, \overline{f} \rangle \tag{58}$$

holds is referred to as a (view) *complement* of f (Matsuda et al., 2007). The pair of functions $(f, \overline{f})$ is also said to be a *monic pair* (Freyd and Scedrov, 1990) or *jointly monic* (Bird and de Moor, 1997). For instance, the two projections

$$\pi_1(a, b) \stackrel{\text{def}}{=} a \quad , \quad \pi_2(a, b) \stackrel{\text{def}}{=} b$$

are each other's complement, since (by reflection) $\langle \pi_1, \pi_2 \rangle = id$ and therefore (58) holds. Clearly, id complements any function, including itself.

From (57) and (42) we obtain $f \cdot R \leq R$ which, in words, means that $(f \cdot)$ always lowers injectivity. From (9) we draw $id \leq f^\circ$ and thus $S \leq f^\circ \cdot S$, thanks to (42). Moreover, Galois connection

$$R \cdot g \leq S \quad \Leftrightarrow \quad R \leq S \cdot g^\circ \tag{59}$$

holds — see proof in appendix D — which can be regarded as the "injectivity counterpart" of "shunting" rule (24).

As special cases of relations, functions may also satisfy functional dependencies. For instance, it will be easy to show that $bagify \stackrel{setify}{\rightharpoonup} id$ holds, where $bagify$ (resp. $setify$) is the function which extracts, from a finite list, the bag (resp. set) of all its elements [14]. From (59) we draw:

$$g \cdot h \leq f \quad \Leftrightarrow \quad f \stackrel{h}{\rightharpoonup} g \quad \Leftrightarrow \quad g \leq f \cdot h^\circ \tag{60}$$

Thus the equivalences

$$g \leq f \quad \Leftrightarrow \quad f \stackrel{id}{\rightharpoonup} g \tag{61}$$

$$h \leq f \quad \Leftrightarrow \quad f \stackrel{h}{\rightharpoonup} id \tag{62}$$

and a more general pattern of FD chaining

$$f \stackrel{S \cdot R}{\rightharpoonup} h \quad \Leftarrow \quad f \stackrel{R}{\rightharpoonup} g \quad \wedge \quad g \leq j \quad \wedge \quad j \stackrel{S}{\rightharpoonup} h \tag{63}$$

which extends (45) via (61).

[14] The $bagify$ / $setify$ terminology is taken from (Bird and de Moor, 1997).

It can be observed that — restricted to functions — the $\leq$ ordering is nothing but the converse of functional "right divisibility":

$$
\begin{array}{ll}
& f \leq g \\
\Leftrightarrow & \quad \{ \text{ (61) ; (31) } \} \\
& f \cdot g^\circ \text{ is simple} \\
\Leftrightarrow & \quad \{ \text{ simple relations are fragments of functions ; "at most simple is simple" } \} \\
& \langle \exists\, k \;::\; f \cdot g^\circ \subseteq k \rangle \\
\Leftrightarrow & \quad \{ \text{ shunting (24) } \} \\
& \langle \exists\, k \;::\; f \subseteq k \cdot g \rangle \\
\Leftrightarrow & \quad \{ \text{ function equality } \} \\
& \langle \exists\, k \;::\; f = k \cdot g \rangle \\
\Leftrightarrow & \quad \{ \text{ right-divisibility } \} \\
& g \text{ (right) divides } f
\end{array}
$$

From (50) we know that *split* is the *lub* of the injectivity preorder. From above we can also regard it as the *glb* of right divisibility, whose universal bounds are also established by (57), in opposite order.

There is another interesting connection between FDs on functions and discrete mathematics. This can be easily grasped by reverting the particular FD $f \stackrel{h}{\rightarrow} f$ to points, while making explicit the equivalence relation which the kernel of function f always is, below denoted by $\sim_f$:

$$
\begin{array}{ll}
& f \stackrel{h}{\rightarrow} f \\
\Leftrightarrow & \quad \{ \text{ (60) } \} \\
& \sim_f \;\; \subseteq \;\; h^\circ \cdot \sim_f \cdot h \\
\Leftrightarrow & \quad \{ \text{ go pointwise, for all suitably typed } a, b \text{ (30) } \} \\
& a \sim_f b \;\; \Rightarrow \;\; (h\, a) \sim_f (h\, b)
\end{array} \tag{64}
$$

The line just above can be recognized as the statement that h is *compatible* with $\sim_f$, that is, that $\sim_f$ is a *congruence* with respect to h. Clearly, this generalizes to n-ary functions, eg. to binary h in FD

$$f \times f \stackrel{h}{\rightarrow} f$$

which involves the product of f by itself (56) and expands to

$$a \sim_f b \;\wedge\; c \sim_f d \;\; \Rightarrow \;\; h(a,c) \sim_f h(b,d) \tag{65}$$

for all (suitably typed) a, b, c, d. This furthermore generalizes to heterogeneous compatibility, that is, to multiple congruences associated to the different functions involved in functional dependencies of shape

$$f \times g \stackrel{h}{\rightarrow} k$$

etc.

7.5 On $\simeq$-equivalence and simplified "split" notation

Discriminating functions beyond $\simeq$-equivalence is unnecessary in the context of FD reasoning. Since order and repetition in "splits" are $\simeq$-irrelevant — recall (53, 54) and (55) — we will abbreviate $\langle f, g\rangle$ by fg, or by gf, wherever this notation shorthand is welcome and makes sense [15]. Such is the case in fact

$$f \stackrel{R}{\leftarrow} gh \quad \Leftrightarrow \quad f \stackrel{R}{\leftarrow} g \quad \wedge \quad f \stackrel{R}{\leftarrow} h \tag{66}$$

which will be particularly helpful in the sequel, despite its straightforward proof:

$$\begin{array}{ll} & f \stackrel{R}{\leftarrow} gh \\ \Leftrightarrow & \quad \{ \text{ (44) ; expansion of shorthand } gh \text{ } \} \\ & \langle g, h\rangle \leq f \cdot R^\circ \\ \Leftrightarrow & \quad \{ \text{ (50) } \} \\ & g \leq f \cdot R^\circ \ \wedge \ h \leq f \cdot R^\circ \\ \Leftrightarrow & \quad \{ \text{ (44) twice } \} \\ & f \stackrel{R}{\leftarrow} g \quad \wedge \quad f \stackrel{R}{\leftarrow} h \end{array}$$

7.6 FD strengthening

The comment above about the contra-variant behavior (concerning injectivity) of the antecedent and consequent functions of an FD is now made precise,

$$h \stackrel{R}{\leftarrow} k \quad \Leftarrow \quad h \geq f \quad \wedge \quad f \stackrel{R}{\leftarrow} g \quad \wedge \quad g \geq k \tag{67}$$

and justified:

$$\begin{array}{ll} & h \geq f \quad \wedge \quad f \stackrel{R}{\leftarrow} g \quad \wedge \quad g \geq k \\ \Leftrightarrow & \quad \{ \text{ (61) twice } \} \\ & h \stackrel{id}{\leftarrow} f \quad \wedge \quad f \stackrel{R}{\leftarrow} g \quad \wedge \quad g \stackrel{id}{\leftarrow} k \\ \Rightarrow & \quad \{ \text{ (45) twice; identity of composition } \} \\ & h \stackrel{R}{\leftarrow} k \end{array}$$

The following are corollaries of (67), since $fh \geq f$:

$$fh \stackrel{R}{\leftarrow} g \quad \Leftarrow \quad f \stackrel{R}{\leftarrow} g \tag{68}$$

$$f \stackrel{R}{\leftarrow} g \quad \Leftarrow \quad f \stackrel{R}{\leftarrow} gh \tag{69}$$

[15] This is inspired by a similar shorthand popular in the standard notation of relational database theory: attribute set union, eg. $X \cup Y$, is denoted by simple juxtaposition, eg. XY (Maier, 1983). Under this correspondence, equivalence (52) becomes the expected $X \subseteq Y \ \Leftrightarrow \ X \cup Y = Y$. In the terminology of Freyd and Scedrov (1990), $f \leq g \ \Leftrightarrow \ fg \simeq g$ corresponds to saying that f is a *short column*.

By $\Leftarrow$-transitivity, we see that it is always possible to move observations in a FD from the consequent ("dependent") side to the antecedent ("independent") one:

$$f h \stackrel{R}{\rightarrow} g \quad \Leftarrow \quad f \stackrel{R}{\rightarrow} g h$$

Moving the "very last" one also makes sense, since

$$f h g \stackrel{R}{\rightarrow} ! \quad \Leftarrow \quad f h \stackrel{R}{\rightarrow} g$$

holds.

7.7 Keys

Every function x such that $x \stackrel{R}{\rightarrow} id$ holds is called a *superkey* for R. *Keys* are minimal superkeys, that is, they are attributes (functions) x such that, for all $y \leq x$ such that $y \not\simeq x$, $y \stackrel{R}{\rightarrow} id$ does not hold. In symbols:

$$x \text{ is a key of } R \quad \Leftrightarrow \quad x \stackrel{R}{\rightarrow} id \ \wedge \ \langle \forall\, y \ : \ y \stackrel{R}{\rightarrow} id : \ y \simeq x \vee y \not\leq x \rangle$$

From (35) and (57) we draw that id is always a (maximal) superkey for simple relations, coreflexives included.

In the Cartesian model of tuples (recall section 6) we can resort to the $\times$-reflection property $\langle \pi_1, \ldots, \pi_n \rangle = id$ to infer that all attributes together are maximal superkeys: $\pi_1 \cdots \pi_n \simeq id$. (A similar inference takes place in the "rich syntax format", however less generic since it is dependent on the chosen selectors.) In fact, any permutation of this split is an isomorphism (eg. $swap = \pi_2 \pi_1$, for $n = 2$) and therefore a maximal superkey. Wherever f is an arbitrary *split* of attributes, we denote by $\overline{f}$ the *split* of the remaining attributes, in any order — a notation convention consistent with the fact that f and $\overline{f}$ are each other complements (58).

8 The Armstrong-axioms

In this section we prove the correctness of the Armstrong-axioms (Maier, 1983), which are the standard inference rules for FDs underlying relational database theory. We show that this subset of FD theory is an immediate consequence of the pointfree formalization presented so far.

In the standard formulation, these axioms involve sets of attributes of a relational schema S ordered by inclusion, eg. $X \subseteq Y \subseteq S$. Unions of such attribute sets are written by juxtaposition, eg. XY instead of $X \cup Y$. Since attributes $X$and Y "are" (projection) functions, XY will mean the *split* of such projections in our setting. Moreover, we generalize these to arbitrary functions ordered by injectivity. Let us, for notation economy, use the same symbols X and Y to denote *both* the attribute symbols and the associated projection functions. Then $X \subseteq Y$ — which is equivalent to the equality of attribute sets $X \cup Y$ and Y — PF-transforms to $X \leq Y$ — which is equivalent to the $\simeq$-equivalence between the *split* of projections XY and projection Y.

The whole schema S corresponds to a maximal observation. In our setting, this is captured by the identity function id, since — by product reflection — the *split* of all projections in a finite product is the identity (recall section 7.7).

As seen already in section 6, n-ary relational database tables are sets of tuples which PF-transform to coreflexive relations. For instance, a table

$$T \quad \subseteq \quad A \times B \times C$$

with three attributes will be modeled by coreflexive

$$A \times B \times C \xleftarrow{[\![T]\!]} A \times B \times C$$

such that $t[\![T]\!]t' \quad \Leftrightarrow \quad t = t' \ \wedge \ t \in T$ (19). As a rule, we will abbreviate $[\![T]\!]$ by T for economy of notation.

Calculational proofs of the Armstrong-axioms follow:

- F1. **Reflexivity**:

$$x \xrightarrow{T} x \tag{70}$$

  This is (36), since T is coreflexive. An equivalent way to put it is

$$yz \xrightarrow{T} y \tag{71}$$

  That (70) entails (71) is an instance of (68). Conversely, (71) instantiates to (70) for $z := !$ — recall (51). Yet another way to express (70) is

$$y \leq x \quad \Rightarrow \quad x \xrightarrow{T} y \tag{72}$$

  see eg. (Beeri et al., 1977) and (O'Neil and O'Neil, 2001), where it is called the *inclusion rule*. That (70) entails (72) follows from $x \xrightarrow{T} x \ \wedge \ x \geq y$, via (67). Conversely, (70) instantiates (72) for $y := x$, given that $\leq$ is a preorder.

- F2. **Augmentation**: Maier's statement of this axiom (1983)

$$x \xrightarrow{T} y \Rightarrow xz \xrightarrow{T} y \tag{73}$$

  is an instance of our fact (68). Another version of this axiom (O'Neil and O'Neil, 2001) is

$$x \xrightarrow{T} y \quad \Rightarrow \quad xz \xrightarrow{T} yz \tag{74}$$

  which is easily shown to be equivalent to (73):

$$\begin{array}{ll} & xz \xrightarrow{T} yz \\ \Leftrightarrow & \quad \{ \ (66) \ \} \\ & xz \xrightarrow{T} y \ \wedge \ xz \xrightarrow{T} z \\ \Leftrightarrow & \quad \{ \ (71), \text{since } T \text{ is coreflexive} \ \} \\ & xz \xrightarrow{T} y \end{array}$$

  Beeri et al. (1977) give yet another (equivalent) version of this axiom:

$$v \leq w \ \wedge \ x \xrightarrow{T} y \quad \Rightarrow \quad xw \xrightarrow{T} yv \tag{75}$$

Clearly, (75) instantiates to (74) for $v = w$. That (74) entails (75) is also easy to show:

$$\begin{array}{ll} & v \leq w \ \wedge \ x \xrightarrow{T} y \\ \Leftrightarrow & \quad \{ \ (61) \ \} \\ & w \xrightarrow{id} v \ \wedge \ x \xrightarrow{T} y \\ \Rightarrow & \quad \{ \ (74) \text{ twice } \} \\ & xw \xrightarrow{id} xv \ \wedge \ xv \xrightarrow{T} yv \\ \Rightarrow & \quad \{ \ (63) \ \} \\ & xw \xrightarrow{T} yv \end{array}$$

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

$$x \xrightarrow{T} y \ \wedge \ x \xrightarrow{T} z \quad \Rightarrow \quad x \xrightarrow{T} yz \tag{76}$$

  This is the $\Leftarrow$-part of equivalence (66).

- F4. **Projectivity**:

$$x \xrightarrow{T} yz \quad \Rightarrow \quad x \xrightarrow{T} y \ \wedge \ x \xrightarrow{T} z \tag{77}$$

  This is the $\Rightarrow$-part of equivalence (66).

- F5. **Transitivity**:

$$x \xrightarrow{T} y \ \wedge \ y \xrightarrow{T} z \quad \Rightarrow \quad x \xrightarrow{T} z \tag{78}$$

  This stems from (45) for S and R the same coreflexive T, thanks to $T \cdot T = T$.

- F6. **Pseudo-transitivity**:

$$x \xrightarrow{T} y \ \wedge \ wy \xrightarrow{T} z \quad \Rightarrow \quad xw \xrightarrow{T} z \tag{79}$$

  As in the standard theory, this stems from F2 and F5:

$$\begin{array}{ll} & x \xrightarrow{T} y \ \wedge \ wy \xrightarrow{T} z \\ \Rightarrow & \quad \{ \ \text{augmentation (74)} \ \} \\ & xw \xrightarrow{T} yw \ \wedge \ wy \xrightarrow{T} z \\ \Rightarrow & \quad \{ \ \text{transitivity (77)} \ \} \\ & xw \xrightarrow{T} z \end{array}$$

This completes the six *inference axioms* which are presented and proved by Maier (1983) either directly — using (3) — or indirectly, using tuple counting and properties of two standard n-ary relation operators: *select* and *project*. Our proofs are substantially simpler thanks to the economy of (44) and derived results.

To complete the set, we present below two consequences of the standard axioms which are often adopted for FD reasoning efficiency:

- **Decomposition** :

$$x \stackrel{T}{\rightharpoonup} y \ \wedge \ z \leq y \quad \Rightarrow \quad x \stackrel{T}{\rightharpoonup} z \tag{80}$$

This is (67) for $z = k$.

- **Accumulation** (O'Neil and O'Neil, 2001):

$$x \stackrel{T}{\rightharpoonup} yz \ \wedge \ z \stackrel{T}{\rightharpoonup} wv \quad \Rightarrow \quad x \stackrel{T}{\rightharpoonup} yzv \tag{81}$$

In fact:

$$\begin{array}{ll} & x \stackrel{T}{\rightharpoonup} yz \ \wedge \ z \stackrel{T}{\rightharpoonup} wv \\ \Rightarrow & \quad \{ \ (74) \ \} \\ & x \stackrel{T}{\rightharpoonup} yz \ \wedge \ yz \stackrel{T}{\rightharpoonup} ywv \\ \Rightarrow & \quad \{ \ (78) \ \} \\ & x \stackrel{T}{\rightharpoonup} yz \ \wedge \ x \stackrel{T}{\rightharpoonup} ywv \\ \Leftrightarrow & \quad \{ \ (66) \ \} \\ & x \stackrel{T}{\rightharpoonup} yzwv \\ \Rightarrow & \quad \{ \ (69) \ \} \\ & x \stackrel{T}{\rightharpoonup} yzv \end{array}$$

8.1 Generalization

In the same way the PF-definition of a FD (44) generalizes the standard set-theoretic definition (3), it is to be expected that the Armstrong axioms (may) extend to relations other than coreflexives. Let R be an arbitrary binary relation. In the discussion of the generalization of the Armstrong axioms which follows the reader should recall the terminology of figure 1 and analyze that of figure 2, noting that a relation R is transitive iff $R \cdot R \subseteq R$ holds, cotransitive iff $R \subseteq R \cdot R$ holds [16], anti-symmetric iff $R \cap R^\circ \subseteq id$ holds and connected iff $R \cup R^\circ = \top$ holds.

- F1. **Reflexivity**: (70) — ie. (36) — generalizes to

$$x \stackrel{R}{\rightharpoonup} x \quad \Leftarrow \quad R \subseteq \mathit{ker}\, x \tag{82}$$

thanks to (46) and trivial FD $x \stackrel{\mathit{ker}\, x}{\rightharpoonup} x$, which is a direct consequence of the difunctionality of x (25). Thus R has to be an endo-relation, but not necessarily coreflexive. Since the equivalence between (70) and (71) is independent of T, the latter also generalizes to

$$yz \stackrel{R}{\rightharpoonup} y \quad \Leftarrow \quad R \subseteq \mathit{ker}\, y \tag{83}$$

[16] We follow the terminology of (Kahl, 2006).

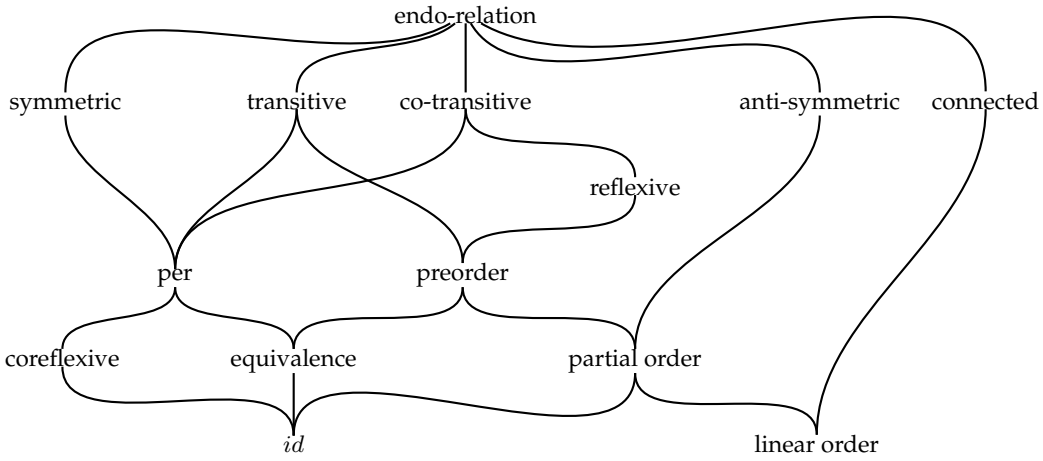


Figure 2: Endo-relation taxonomy, where per stands for *partial equivalence relation*.

- F2. **Augmentation**: Coreflexive T in (73) can be generalized to any arbitrary binary relation R, recall (68). Thanks to (83), (74) generalizes to

$$x \stackrel{R}{\leftarrow} y \ \wedge \ R \subseteq \ker z \quad \Rightarrow \quad xz \stackrel{R}{\leftarrow} yz \tag{84}$$

  and (75) generalizes to

$$v \leq w \ \wedge \ x \stackrel{R}{\leftarrow} y \ \wedge \ R \subseteq \ker v \quad \Rightarrow \quad xw \stackrel{R}{\leftarrow} yv$$

- F3. **Additivity** (or **Union**): As part of (66), (76) holds for arbitrary relations.
- F4. **Projectivity**: As part of (66), (76) holds for arbitrary relations.
- F5. **Transitivity** : Thanks to (45, 46) together, coreflexive T in (78) generalizes to any relation R satisfying the cotransitivity condition $R \subseteq R \cdot R$. Reflexive relations and partial equivalence relations (which include coreflexives, cf. *pers* in figure 2) qualify for this property.
- F6. **Pseudo-transitivity**: Thanks to the above generalizations of F5 and F2, coreflexive T in (79) generalizes to any cotransitive sub-relation of $\ker w$, that is, to any R such that $R \subseteq R \cdot R \cap \ker w$ holds.
- **Decomposition**: As an instance of (67), T in (80) extends to arbitrary relations.
- **Accumulation**: Similarly to F6, T in (81) generalizes to any cotransitive sub-relation of $\ker y$.

All in all, three main conclusions arise from the generalization:

- augmentation (F2 in version (72)), additivity (F3), projectivity (F4) and decomposition scale up to arbitrary binary relations;
- equivalent statements of axiom F2 (73, 74) in the standard theory get split throughout the generalization, cf. (84);

- cotransitivity (figure 2) emerges as an interesting property of endo-relations, which encompasses a quite broad set of sub-classes, eg. *per*s, partial and linear orders, etc.

We proceed to the pointfree formulation and generalization of other standard concepts in relational database theory which are required by our final aim in the report: to establish the principle of *lossless decomposition* (Maier, 1983) by pointfree calculation.

9 Generic relational projections

Relational database theory incorporates a concept which is central to the principle of relational data decomposition — that of a relational *projection*. Given n-ary relation T with schema S and $x \subseteq S$, Maier (1983) defines the x-projection of T as set-comprehension

$$\pi_x T \quad = \quad \{x[t] \mid t \in T\} \tag{85}$$

Above we have shown how to express concepts involving a given n-ary relation T — which in the binary relation calculus is modeled by coreflexive $[\![T]\!]$ — in terms of binary relations. A similar construction can be provided for the *projection* operator (85), as follows: given a binary relation R and functions f and g such as in definition 4, the g, f*-projection* of R is defined as binary relation

$$\pi_{g,f} R \quad \stackrel{\text{def}}{=} \quad g \cdot R \cdot f^\circ \tag{86}$$

So, $f \stackrel{R}{\perp} g$ can be rephrased by saying that *projection* $\pi_{g,f} R$ *is simple*.

The set-theoretical meaning of $\pi_{g,f} R$ can be grasped by noting that, while putting together the lower adjoints of shunting rules (23, 24), $\pi_{g,f}$ is itself a lower adjoint:

$$\pi_{g,f} R \subseteq S \quad \Leftrightarrow \quad R \subseteq g^\circ \cdot S \cdot f \tag{87}$$

This means that $\pi_{g,f} R$ is the *smallest* relation which, wherever b is R-related to a, relates $(g\ b)$ to $(f\ a)$ — recall (30). Regarding relations as sets of pairs, we have

$$\pi_{g,f} R \quad \stackrel{\text{def}}{=} \quad \{(g\ b, f\ a) \mid (b, a) \in R\} \tag{88}$$

The close relationship between FDs and (binary) relational projection is captured by the following equivalence

$$h \stackrel{\pi_{g,f} R}{\perp} k \quad \Leftrightarrow \quad (h \cdot f) \stackrel{R}{\perp} (k \cdot g) \tag{89}$$

which enables observer function "trading" between a projection and a (composite) FD. The calculation of (89) is yet another display of agility of (44) and associated theory:

$$\begin{array}{ll} & h \stackrel{\pi_{g,f} R}{\perp} k \\ \Leftrightarrow & \qquad \{\ \text{(44), (86) and (14)}\ \} \end{array}$$

$$
\begin{array}{ll}
& k \leq h \cdot f \cdot R^\circ \cdot g^\circ \\
\Leftrightarrow & \quad \{\ (59)\ \} \\
& k \cdot g \leq h \cdot f \cdot R^\circ \\
\Leftrightarrow & \quad \{\ (44)\ \} \\
& (h \cdot f) \stackrel{R}{\leftarrow} (k \cdot g)
\end{array}
$$

A rather interesting view of (87) is

$$\pi_{g,f} R \subseteq S \quad \Leftrightarrow \quad g(S \leftarrow R)f \tag{90}$$

where $S \leftarrow R$ is Reynolds "arrow combinator"

$$g(S \leftarrow R)f \quad \Leftrightarrow \quad g \cdot R \subseteq S \cdot f \tag{91}$$

which is extensively studied by Backhouse and Backhouse (2004). So, a (f, g-parametric) projection between two relations (R and S) can be equated as a (R, S-parametric) relation on the projection functions f and g themselves.

Besides monotonicity and $\cup$-preservation, ensured by lower-adjointedness (Aarts et al., 1992), binary relation projection obeys to a number of useful properties, namely

$$
\begin{array}{rcl}
\pi_{id,id} & = & id \\
\pi_{f,g} \cdot \pi_{h,k} & = & \pi_{f \cdot h, g \cdot k} \\
(\pi_{f,g} R)^\circ & = & \pi_{g,f}(R^\circ)
\end{array}
$$

and

$$\pi_{f,f} R \subseteq id \quad \Leftrightarrow \quad R \subseteq \mathit{ker}\, f \tag{92}$$

which is related to (82, 83). It follows that the $\pi_{f,f}$ projection of a coreflexive is also coreflexive, in fact it own domain:

$$\pi_{f,f} R = \delta\,(\pi_{f,f} R) \quad \Leftarrow \quad R \text{ is coreflexive} \tag{93}$$

Thus (86) extends (85), as shown by equality

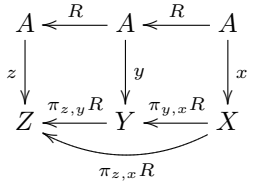
$$[\![\pi_x T]\!] \quad = \quad \pi_{x,x}[\![T]\!]$$

which holds for any n-ary relation T thanks to (88, 92). Note the use of the same symbol π to denote both the standard set-theoretic projection operator, on the left hand side of the equality, and the pointfree one, on the right hand side.

Also useful in the sequel is fact

$$\pi_{z,x} R \subseteq (\pi_{z,y} R) \cdot (\pi_{y,x} R) \quad \Leftarrow \quad R \subseteq \mathit{ker}\, y \tag{94}$$

cf. diagram



easy to infer by monotonicity of composition and the fact that $R \subseteq R \cdot R^\circ \cdot R$ holds for every R (see section C.2).

10 Lossless decomposition

Arbitrary FDs are, in general, hard to maintain because they constrain the CRUD (create, update, and delete) operations on database files, and waste space. Therefore, instead of allowing for some n-ary relation T to satisfy an arbitrary FD, it is preferable to "extract" such a dependency by decomposing T in two parts — the FD itself, eg. with schema

$$S_1 \quad = \quad \{\text{FLIGHT}, \text{DEPARTS}\}$$

recalling FD (2) in our introductory example, and the "rest" of T, with schema

$$S_2 \quad = \quad \{\text{PILOT}, \text{FLIGHT}, \text{DATE}\}$$

in the same example. Such components are nothing but projections $\pi_{S_1} T$ and $\pi_{S_2} T$ of T, respectively — recall (85).

In this example, the fact that FLIGHT — the antecedent of the selected FD — is kept in schema S_2 has to do with the principle of *lossless decomposition*: once T is decomposed into projections $\pi_{S_1} T$ and $\pi_{S_2} T$, by "joining" them one should be able to recover the original relation [17]:

$$(\pi_{S_1} T) \bowtie (\pi_{S_2} T) \quad = \quad T$$

Lossless decomposition is a representation technique which is central to relational database implementation. Of course, not every pair of projections is lossless. A kernel topic of the process of database *design by decomposition* is precisely that of finding conditions for safe decomposition. Such is the case of extracting functional dependencies, as illustrated above, thanks to a couple of theorems which will be dealt with in the sequel.

The first of these — exercise 6.4 in (Maier, 1983) — is as follows: *given relation schemes Y and Z such that $Y \cap Z = X$ and a relation T with schema YZ satisfying FD $X \rightarrow Y$, then lossless decomposition*

$$T \quad = \quad (\pi_Y T) \bowtie (\pi_Z T)$$

holds.

Our proof of this result boils down to almost *no-work-at-all* thanks to the binary relation extension of the projection operator given by (86). Recall that (86) expresses the standard semantics of relational projection, the only difference being in requiring two projection functions — antecedent f and consequent g — instead of one. This pair leads to a straightforward definition of *join*: joining two projections which share the same antecedent function, say x, is nothing but binary relation *split* (47, 149):

$$(\pi_{y,x} R) \bowtie (\pi_{z,x} R) \quad \stackrel{\text{def}}{=} \quad \langle y \cdot R \cdot x^\circ, z \cdot R \cdot x^\circ \rangle \tag{95}$$

[17] The standard, set-theoretic semantics of the n-ary relation join operator $\bowtie$ is as follows (Maier, 1983): given relations T, T' with schemes S, S', respectively, $T \bowtie T'$ is the relation with schema SS' defined by

$$T \bowtie T' \quad = \quad \{t'' \mid \langle \exists\, t, t' \,:\, t \in T \,\wedge\, t' \in T' :\, t = t''[S] \,\wedge\, t' = t''[S'] \rangle\}$$

And lossless decomposition can be expressed parametrically with respect to consequent functions y and z,

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

that is,

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

It is well-known that such unconditioned $\times$-fusion doesn't hold in relation algebra, in general. Theorem 12.30(b) in (Aarts et al., 1992) adds a side-condition for such a fusion to take place, where R, S, T are suitably typed binary relations [18]:

$$\langle R, S \rangle \cdot T = \langle R \cdot T, S \cdot T \rangle \quad \Leftarrow \quad R \cdot (img\, T) \subseteq R \vee S \cdot (img\, T) \subseteq S \qquad (96)$$

The instance of (96) which suits our needs is $(R, S := y, z)$

$$\langle y, z \rangle \cdot T = \langle y \cdot T, z \cdot T \rangle \quad \Leftarrow \quad y \leq T^\circ \vee z \leq T^\circ$$

— recall (15) and (39) — whereby further instantiating $T := R \cdot x^\circ$ we obtain

$$\langle y, z \rangle \cdot (R \cdot x^\circ) = \langle y \cdot R \cdot x^\circ, z \cdot R \cdot x^\circ \rangle \quad \Leftarrow \quad x \stackrel{R}{\leftarrow} y \vee x \stackrel{R}{\leftarrow} z$$

that is,

$$\pi_{yz,x}R = (\pi_{y,x}R) \bowtie (\pi_{z,x}R) \quad \Leftarrow \quad x \stackrel{R}{\leftarrow} y \vee x \stackrel{R}{\leftarrow} z \qquad (97)$$

In summary, lossless decomposition via FD extraction is, in our framework, a corollary of (conditioned) $\times$-fusion (96). The question arises: are there side-conditions weaker than that of (97) for lossless decomposition to take place?

It turns out that FD existence is a sufficient but not necessary condition for safe decomposition to take place: the more general (but less intuitive) concept of a *multi-valued* dependency — already introduced in section 3, recall (4) — is what is actually required.

10.1 Multi-valued dependencies

Recall from section 3 that we have two alternative definitions of a multi-valued dependency, as captured by logical formulæ (4) and (5).

The task of calculating the pointfree transform of (4) will be considerably softened by rule (142) of the appendix, which generalizes relational composition (13). We remind the reader that x, y and $z = S - xy$ are relational attributes which will be regarded as projection functions in the PF-transformation of (4) which follows. We address the existential quantification of same formula first:

$$\begin{array}{ll} & \langle \exists\, t'' \;:\; t'' \in T :\; t[xy] = t''[xy] \;\wedge\; t''[z] = t'[z] \rangle \\ \Leftrightarrow & \qquad \{ \text{ (142) for } \phi := (\in T), \text{ and so on } \} \\ & t(ker\, xy \cdot [\![T]\!] \cdot ker\, z)t' \end{array}$$

[18] This result can be regarded as an instance of property (145) in the appendices.

Then we insert this in the overall formula and proceed:

$$\begin{array}{ll} & \langle \forall\, t, t' \;:\; t, t' \in T : \quad t[x] = t'[x] \quad \Rightarrow \quad t(\ker x y \cdot [\![T]\!] \cdot \ker z) t' \;\rangle \\ \Leftrightarrow & \qquad \{ \text{ rule (141) for } \phi = \psi = (\in T) \;\} \\ & [\![T]\!] \cdot (\ker x) \cdot [\![T]\!] \quad \subseteq \quad (\ker x y) \cdot [\![T]\!] \cdot \ker z \end{array}$$

Thus we reach pointfree definition (98) below, in which we generalize $[\![T]\!]$ to an arbitrary endo-relation $A \xleftarrow{R} A$ and introduce notation $x \overset{R}{\rightharpoonup\!\!\!\!\rightharpoonup} y$ (read: *x multi-determines y in R*) in the spirit of notation $x \overset{R}{\rightharpoonup} y$ already adopted for FDs:

$$x \overset{R}{\rightharpoonup\!\!\!\!\rightharpoonup} y \quad \overset{\text{def}}{=} \quad R \cdot (\ker x) \cdot R \subseteq (\ker x y) \cdot R \cdot \ker z \tag{98}$$

where z is the projection function associated to the attributes in $S - xy$. For symmetric R (ie. such that $R = R^\circ$) this can be rewritten into the form

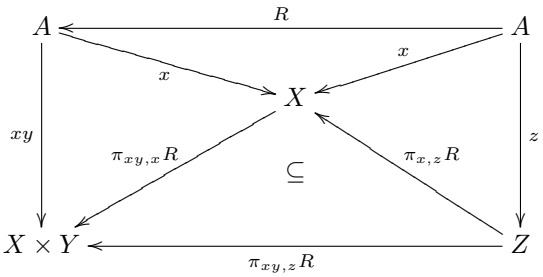
$$x \overset{R}{\rightharpoonup\!\!\!\!\rightharpoonup} y \quad \overset{\text{def}}{=} \quad \ker (x \cdot R^\circ) \subseteq (\ker x y) \cdot R \cdot \ker z \tag{99}$$

which bears closer resemblance with definition (33) of a FD.

That definition (98) requires R to have the same source and target type can be checked by expanding its right hand-side and "shunting" wherever possible:

$$\begin{array}{ll} & R \cdot (\ker x) \cdot R \subseteq (\ker x y) \cdot R \cdot \ker z & (100) \\ \Leftrightarrow & \qquad \{ \text{ (11) ; (23 and 24) } \} \\ & (x y \cdot R \cdot x^\circ) \cdot (x \cdot R \cdot z^\circ) \quad \subseteq \quad x y \cdot R \cdot z^\circ & (101) \\ \Leftrightarrow & \qquad \{ \text{ (86) three times } \} \\ & (\pi_{xy,x} R) \cdot (\pi_{x,z} R) \quad \subseteq \quad \pi_{xy,z} R & (102) \end{array}$$

In this way we obtain diagram



(103)

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

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

recalling (6).

It is easy to see that condition $R \cdot (\mathit{ker}\, x) \cdot R \;\subseteq\; R$ is sufficient for (99) to hold, since function kernels are reflexive (10). Thus, both $\bot$ and $\top$ (resp. the least and the greatest endo-relations over A) satisfy *any* MVD.

The special case $x \overset{R}{\twoheadrightarrow} x$ is also easy to calculate:

$$
\begin{array}{ll}
 & x \overset{R}{\twoheadrightarrow} x \\
\Leftrightarrow & \quad \{\ \text{(99) for } x := y \text{ ; } xx \simeq x \text{ (53)}\ \} \\
 & R \cdot (\mathit{ker}\, x) \cdot R \subseteq (\mathit{ker}\, x) \cdot R \cdot \mathit{ker}\, z \\
\Leftrightarrow & \quad \{\ \text{(102)}\ \} \\
 & (\pi_{x,x} R) \cdot (\pi_{x,z} R) \;\subseteq\; \pi_{x,z} R \\
\Leftarrow & \quad \{\ \text{monotonicity}\ \} \\
 & \pi_{x,x} R \;\subseteq\; id \\
\Leftrightarrow & \quad \{\ \text{(92)}\ \} \\
 & R \subseteq \mathit{ker}\, x
\end{array}
$$

Thus we have established the MVD counterpart of (82):

$$x \overset{R}{\twoheadrightarrow} x \;\Leftarrow\; R \subseteq \mathit{ker}\, x \tag{104}$$

As it happens with FDs, the standard axiomatic theory of MVDs (Maier, 1983) assumes R to be "a set of tuples". As earlier on, we model such a set by a coreflexive relation and use capital letter T to mark this assumption (R will be written otherwise). Clearly, (102) becomes equality for $R := T$ (coreflexive),

$$(\pi_{xy,x} T) \cdot (\pi_{x,z} T) \;=\; \pi_{xy,z} T \tag{105}$$

since the converse inclusion always holds thanks to (94). In this situation, there is still another alternative to (105) which will be useful later on:

$$(\pi_{z,x} T) \cdot \pi_1 \cdot \delta\,(\pi_{z,xy} T) \;=\; \pi_{z,xy} T \tag{106}$$

The fact that T is coreflexive is central to the equivalence between (105) and (106):

$$
\begin{array}{ll}
 & (\pi_{xy,x} T) \cdot (\pi_{x,z} T) \;=\; \pi_{xy,z} T \\
\Leftrightarrow & \quad \{\ \text{converses (92) ; } T \text{ coreflexive}\ \} \\
 & (\pi_{z,x} T) \cdot (\pi_{x,xy} T) \;=\; \pi_{z,xy} T \\
\Leftrightarrow & \quad \{\ \times\text{-cancellation}\ \} \\
 & (\pi_{z,x} T) \cdot \pi_1 \cdot (\pi_{xy,xy} T) \;=\; \pi_{z,xy} T \\
\Leftrightarrow & \quad \{\ \text{(93)}\ \} \\
 & (\pi_{z,x} T) \cdot \pi_1 \cdot \delta\,(\pi_{xy,xy} T) \;=\; \pi_{z,xy} T \\
\Leftrightarrow & \quad \{\ \text{domain of composition; } z \text{ and } xy \text{ are entire}\ \} \\
 & (\pi_{z,x} T) \cdot \pi_1 \cdot \delta\,(\pi_{z,xy} T) \;=\; \pi_{z,xy} T
\end{array}
$$

The fact that FDs are special cases of MVDs is known from the standard theory and is captured by Maier (1983)'s *replication axiom* C1:

$$x \xrightarrow{T} y \quad \Rightarrow \quad x \overset{T}{\twoheadrightarrow} y \tag{107}$$

In our (more general) setting, we state with care: *within coreflexive relations*, MVDs are more general than FDs [19], as the following calculation shows for arbitrary coreflexive T:

$$\begin{array}{ll} & x \xrightarrow{T} y \\ \Leftrightarrow & \quad \{ \text{ (36) and (66) } \} \\ & x \xrightarrow{T} xy \\ \Leftrightarrow & \quad \{ \text{ definition (33) } \} \\ & \mathit{ker}\,(x \cdot T^\circ) \quad \subseteq \quad \mathit{ker}\,xy \\ \Leftrightarrow & \quad \{ \text{ expansion of } \mathit{ker}\,(x \cdot T^\circ), \text{ symmetry of } T \text{ and (20) } \} \\ & T \cdot (\mathit{ker}\,x) \cdot T \quad \subseteq \quad (\mathit{ker}\,xy) \cdot T \\ \Rightarrow & \quad \{ \text{ } \mathit{ker}\,z \text{ is reflexive (10) } \} \\ & T \cdot (\mathit{ker}\,x) \cdot T \quad \subseteq \quad (\mathit{ker}\,xy) \cdot T \cdot \mathit{ker}\,z \\ \Leftrightarrow & \quad \{ \text{ definition (99) } \} \\ & x \overset{T}{\twoheadrightarrow} y \end{array} \tag{108, 109, 110}$$

10.2 Generic lossless decomposition theorem

Maier (1983) gives the replication axiom (107) as a corollary of the theorem of *lossless decomposition* of MVDs. This theorem [20] states that fact $x \overset{T}{\twoheadrightarrow} y$ holds *if and only if* T decomposes losslessly into two relations with schemata yx and zx (for $z = S - yx$), respectively:

$$x \overset{T}{\twoheadrightarrow} y \quad \Leftrightarrow \quad (\pi_{y,x}T) \bowtie (\pi_{z,x}T) = \pi_{yz,x}T \tag{111}$$

The pointwise proof of this result given by Maier (1983) follows the "implication-first" logic style, in two parts — the *if* side followed by the *only if* side of the equivalence. Being performed as they are directly over formula (4), these proofs aren't easy to follow with their existential and universal quantifications over no less than six tuple variables $t, t_1, t_2, t'_1, t'_2, t_3$. By contrast, our proof is a sequence of PF-equivalences:

$$\begin{array}{ll} & (\pi_{y,x}T) \bowtie (\pi_{z,x}T) = \pi_{yz,x}T \\ \Leftrightarrow & \quad \{ \text{ (95) ; (86) three times } \} \end{array}$$

[19] This statement will be further generalized in section 11.4 to a subclass of cotransitive, symmetric relations.

[20] Theorem 7.1 in (Maier, 1983).

$$
\begin{array}{ll}
& \langle y \cdot T \cdot x^\circ, z \cdot T \cdot x^\circ \rangle \quad = \quad yz \cdot T \cdot x^\circ \\
\Leftrightarrow & \quad \{ \text{ since } \langle R, S \rangle \cdot T \subseteq \langle R \cdot T, S \cdot T \rangle \text{ holds by monotonicity } \} \\
& \langle y \cdot T \cdot x^\circ, z \cdot T \cdot x^\circ \rangle \quad \subseteq \quad yz \cdot T \cdot x^\circ \\
\Leftrightarrow & \quad \{ \text{ “split twist” rule (154) — twice ; converses } \} \\
& \langle y \cdot T \cdot x^\circ, id \rangle \cdot x \cdot T^\circ \cdot z^\circ \quad \subseteq \quad \langle y, x \cdot T^\circ \rangle \cdot z^\circ \\
\Leftrightarrow & \quad \{ \text{ on the lower side, (162) and } T^\circ = T \text{ ; (153) on the upper side, } \Phi := T^\circ = T \} \\
& \langle y \cdot T \cdot x^\circ, x \cdot x^\circ \rangle \cdot x \cdot T \cdot z^\circ \quad \subseteq \quad \langle y, x \rangle \cdot T \cdot z^\circ \\
\Leftrightarrow & \quad \{ \text{ (152) for } S := x \text{, followed by (153) for } \Phi := T \ \} \\
& (\langle y, x \rangle \cdot T \cdot x^\circ) \cdot (x \cdot T \cdot z^\circ) \quad \subseteq \quad \langle y, x \rangle \cdot T \cdot z^\circ \\
\Leftrightarrow & \quad \{ \text{ (101) } \} \\
& x \overset{T}{\twoheadrightarrow} y
\end{array}
$$

It should be noted that the assumption that z is the projection associated to all attributes other than xy in (99) has played no rôle whatsoever in the calculations of (104, 107, 111) given above. This means that z can be regarded as yet another arbitrary (but suitably typed) observer function. Altogether, we are lead to the following, more general relational definition of a multi-valued dependency:

Definition 5 *Given endo-relation* $A \xleftarrow{R} A$ *and three functions* $X \xleftarrow{x} A$, $Y \xleftarrow{y} A$ *and* $Z \xleftarrow{z} A$ *, multivalued dependency* $x \overset{R}{\twoheadrightarrow}_z y$ *holds (note the subscript* z*) if and only if*

$$R \cdot (\ker x) \cdot R \subseteq (\ker xy) \cdot R \cdot \ker z \tag{112}$$

holds, which equivales

$$xy \cdot R \cdot x^\circ \cdot x \cdot R \cdot z^\circ \quad \subseteq \quad xy \cdot R \cdot z^\circ \tag{113}$$

itself the same as

$$(\pi_{xy,x} R) \cdot (\pi_{x,z} R) \quad \subseteq \quad \pi_{xy,z} R \tag{114}$$

For cotransitive R*, (113,114) strengthen to equality,*

$$(\pi_{xy,x} R) \cdot (\pi_{x,z} R) \quad = \quad \pi_{xy,z} R \tag{115}$$

since $R \subseteq R \cdot x^\circ \cdot x \cdot R$ *holds for such relations. For symmetric* R*, (112) can be further re-written into*

$$\ker (x \cdot R^\circ) \subseteq (\ker xy) \cdot R \cdot \ker z \tag{116}$$

As with FDs, x *(resp.* y*) will be referred to as the* antecedent *(resp.* consequent*) of MVD* $x \overset{R}{\twoheadrightarrow}_z y$*. Function* z *will be mentioned as the* context [21].
□

[21] The context of the standard definition (Maier, 1983) is fixed to $z = S - yx$.

Hereupon we shall write $x \overset{R}{\rightarrowtail} y$ as an abbreviation of $\langle \forall\ z\ ::\ x \overset{R}{\rightarrowtail}_z y \rangle$, meaning that the MVD holds for *every* context z. As we will see in the sequel, there are MVD rules which are context-independent and rules which are context-dependent [22]. A simple example of context-independence is the following generic statement of MVD *reflexivity*

$$y \leq x \quad \Rightarrow \quad x \overset{R}{\rightarrowtail} y \Leftrightarrow x \overset{R}{\rightarrowtail} x \tag{117}$$

which generalizes rule MVD1 of (Beeri et al., 1977)

$$y \leq x \quad \Rightarrow \quad x \overset{T}{\rightarrowtail} y \tag{118}$$

to an *arbitrary* endo-relation R. (Rule (118) stems from (117) thanks to (104), for R instantiated to coreflexive T.) The calculation of (117) follows from the definitions:

$$\begin{array}{ll} & y \leq x \\ \Leftrightarrow & \quad \{\ (52) \text{ and } (53)\ \} \\ & \ker x y = \ker x x \\ \Rightarrow & \quad \{\ \text{Leibniz}\ \} \\ & (\ker x y) \cdot R \cdot \ker z = (\ker x x) \cdot R \cdot \ker z \\ \Rightarrow & \quad \{\ (112)\ \} \\ & x \overset{R}{\rightarrowtail} y \Leftrightarrow x \overset{R}{\rightarrowtail} x \end{array}$$

We close this section by stating the theorem of lossless decomposition in the generic context of definition 5.

Theorem 1 *Given coreflexive relation* $A \overset{T}{\longleftarrow} A$ *and projection functions* x, y *and* z *such as in definition 5, multivalued dependence* $x \overset{T}{\rightarrowtail}_z y$ *— as defined by (112,113,114) — equivales lossless decomposition of projection* $\pi_{yz,x}T$ *into projections* $\pi_{y,x}T$ *and* $\pi_{y,x}T$. *That is,*

$$x \overset{T}{\rightarrowtail}_z y \quad \Leftrightarrow \quad \pi_{yz,x}T = (\pi_{y,x}T) \bowtie (\pi_{z,x}T)$$

holds. Proof: it suffices to regard z *as an arbitrary context in the calculation of (111) given above.* □

Our first comment about theorem 1 goes to the fact that it is stated and proved free of the restrictions on x, y and z which are found in the literature, see eg. proposition 2 in (Beeri et al., 1977) and theorem 7.1 in (Maier, 1983). Thus lossless decomposition is far more general than it has been understood thus far.

In its standard, restricted format, Beeri et al. (1977) regard this result as *"probably the most important single property of multi-valued dependencies"*. The PF-restatement of this theorem and proof fulfills the main target of this report, as purported by its

[22] Fagin (1977) was among the first to identify the relevance of context-dependency in MVD reasoning.

title: we wanted to produce evidence that lossless decomposition is more general that it has been regarded so far, in particular with respect to its extension to functions and binary relations and to the freedom enjoyed by context z in definition 5.

Such a broader view of this classical result can be regarded as a kind of follow-up of similar generalizations which occurred in the past. For instance, it is only after section 8 that Fagin (1977) relaxes antecedent x and consequent y from being disjoint.

Conversely, it is to be expected that not all standard MVD inference rules will survive such a generalization, in particular those relying on attribute (set) difference. We will devote the following sections to (generic) MVD reasoning and inference-rules *not involving* difference [23].

11 Calculating with (generic) MVDs

11.1 Limit cases

Let us start by calculating limit cases $x \overset{R}{\rightarrowtail} !$ and $! \overset{R}{\rightarrowtail} y$, which correspond to MVDs $x \twoheadrightarrow \emptyset$ and $\emptyset \twoheadrightarrow y$ in the standard theory (Maier, 1983). The former is the MVD counterpart to (34):

$$
\begin{array}{ll}
 & x \overset{R}{\rightarrowtail} ! \\
\Leftrightarrow & \quad \{\ ! \leq x \text{ in (117)}\ \} \\
 & x \overset{R}{\rightarrowtail} x \\
\Leftrightarrow & \quad \{\ (104)\ \} \\
 & R \subseteq \mathit{ker}\, x
\end{array}
$$

Thus, for coreflexive T, $x \overset{T}{\rightarrowtail} !$ always holds, since function kernels are reflexive.

Concerning $! \overset{T}{\rightarrowtail}_z y$, Maier (1983) shows (somewhat short-circuitously) that any T satisfying this MVD *must be the cross product of projections $\pi_y T$ and $\pi_z T$*. The detailed calculation given below shows this to be an equivalence (and not just an implication) and that T can be any endo-relation R:

$$
\begin{array}{ll}
 & ! \overset{R}{\rightarrowtail}_z y \\
\Leftrightarrow & \quad \{\ \text{(112) together with (41) and } !y \simeq y \text{ ; shunting}\ \} \\
 & y \cdot R \cdot \top \cdot R \cdot z^\circ \quad \subseteq \quad y \cdot R \cdot z^\circ \\
\Leftrightarrow & \quad \{\ \text{equality ensured by circular inclusion, since } R \subseteq R \cdot \top \cdot R \text{ for all } R\ \} \\
 & y \cdot R \cdot \top \cdot R \cdot z^\circ \quad = \quad y \cdot R \cdot z^\circ \\
\Leftrightarrow & \quad \{\ \text{(41) and fusion-law } ! \cdot f = ! \text{ for all } f\ \}
\end{array}
$$

[23] How to suitably generalize attribute (set) difference to the FD and MVD definitions given in this report is subject of ongoing research, see section 14.

$$
\begin{array}{ll}
& y \cdot R \cdot y^\circ \cdot \top \cdot z \cdot R \cdot z^\circ \quad = \quad y \cdot R \cdot z^\circ \\
\Leftrightarrow & \qquad \{ \text{ (86) and definition of cross-product (22) } \} \\
& \pi_{yy} R \otimes \pi_{zz} R \quad = \quad \pi_{yz} R
\end{array}
$$

Mapped back to the pointwise level, for R the coreflexive representing a n-ary relation T, the last line of the reasoning above yields the expected

$$\{t[y] \mid t \in T\} \times \{t[z] \mid t \in T\} = \{(t[y], t[z]) \mid t \in T\}$$

11.2 Monotonicity

From (112) we draw the MVD counterpart of (67) which follows

$$x \leq x' \leq xy \ \wedge \ x \overset{R}{\rightarrow\!\!\!\!\!\cdot}_z y \ \wedge \ y' \leq xy \ \wedge \ z' \leq z \quad \Rightarrow \quad x' \overset{R}{\rightarrow\!\!\!\!\!\cdot}_{z'} y' \tag{119}$$

simply by solving a system of equations

$$\left\{ \begin{array}{rcl} ker\, x' & \subseteq & ker\, x \\ ker\, xy & \subseteq & ker\, x'y' \\ ker\, z & \subseteq & ker\, z' \end{array} \right. \quad \Leftrightarrow \quad \left\{ \begin{array}{c} x \leq x' \leq xy \\ y' \leq xy \\ z' \leq z \end{array} \right.$$

which altogether ensure $x' \overset{R}{\rightarrow\!\!\!\!\!\cdot}_{z'} y'$ from $x \overset{R}{\rightarrow\!\!\!\!\!\cdot}_z y$, thanks to monotonicity of composition.

From this we immediately generalize Proposition 1 in (Beeri et al., 1977) in a way which is context free and dispenses with attribute difference and complementation:

$$x \overset{R}{\rightarrow\!\!\!\!\!\cdot} yv \ \Leftrightarrow \ x \overset{R}{\rightarrow\!\!\!\!\!\cdot} y \quad \Leftarrow \quad v \leq x \tag{120}$$

The equivalence is proved by circular implication:

$$
\begin{array}{ll}
& x \overset{R}{\rightarrow\!\!\!\!\!\cdot} y \\
\Rightarrow & \qquad \{ \ yv \leq xy \text{ in (119) since } v \leq x \text{, keeping antecedent and context } \} \\
& x \overset{R}{\rightarrow\!\!\!\!\!\cdot} yv \\
\Rightarrow & \qquad \{ \ y \leq xyv \text{ in (119), still keeping antecedent and context } \} \\
& x \overset{R}{\rightarrow\!\!\!\!\!\cdot} y
\end{array}
$$

Another immediate consequence of (119) is that context can always "shrink" in MVDs:

$$x \overset{R}{\rightarrow\!\!\!\!\!\cdot}_z y \ \wedge \ z' \leq z \quad \Rightarrow \quad x \overset{R}{\rightarrow\!\!\!\!\!\cdot}_{z'} y \tag{121}$$

From (67) we know that consequents can always "shrink" in FDs. Does the same happen with MVDs? Below we show that for this to hold in an MVD the relation involved must be coreflexive,

$$T \text{ is coreflexive } \wedge \ x \overset{T}{\rightarrow\!\!\!\!\!\cdot}_z y \ \wedge \ y' \leq y \quad \Rightarrow \quad x \overset{T}{\rightarrow\!\!\!\!\!\cdot}_z y' \tag{122}$$

a fact which stems from an auxiliary result which ensures that, for $R := T$ coreflexive, context z and consequent y are interchangeable in definition 5:

$$x \overset{T}{\rightharpoonup}_z y \quad \Leftrightarrow \quad x \overset{T}{\rightharpoonup}_y z \quad \Leftarrow \quad T \text{ is coreflexive} \tag{123}$$

The proof of (123) goes as follows:

$$\begin{array}{ll}
& x \overset{T}{\rightharpoonup}_z y \\
\Leftrightarrow & \quad \{ \text{ (116) ; expansion of } xy \text{ after shunting } \} \\
& \langle x, y\rangle \cdot (\mathit{ker}\,(x \cdot T^\circ)) \cdot z^\circ \quad \subseteq \quad \langle x, y\rangle \cdot T \cdot z^\circ \\
\Leftrightarrow & \quad \{ \text{ ``split twist'' rule (154) ; converses } \} \\
& \langle x, z \cdot \mathit{ker}\,(x \cdot T^\circ)\rangle \cdot y^\circ \quad \subseteq \quad \langle x, z \cdot T^\circ\rangle \cdot y^\circ \\
\Leftrightarrow & \quad \{ \text{ (161,153) since } T \text{ is coreflexive and thus } x \cdot T^\circ \text{ is simple } \} \\
& \langle x, z\rangle \cdot \mathit{ker}\,(x \cdot T^\circ) \cdot y^\circ \quad \subseteq \quad \langle x, z\rangle \cdot T^\circ \cdot y^\circ \\
\Leftrightarrow & \quad \{ \text{ shunting and (116) } \} \\
& x \overset{T}{\rightharpoonup}_y z
\end{array}$$

We are now in position to check (122) just by putting (123), (121) and (119) together:

$$\begin{array}{ll}
& x \overset{T}{\rightharpoonup}_z y \ \wedge \ y' \leq y \\
\Leftrightarrow & \quad \{ \text{ (123) } \} \\
& x \overset{T}{\rightharpoonup}_y z \ \wedge \ y' \leq y \\
\Rightarrow & \quad \{ \text{ (119) } \} \\
& x \overset{T}{\rightharpoonup}_{y'} z \\
\Rightarrow & \quad \{ \text{ (121) } \} \\
& x \overset{T}{\rightharpoonup}_z y'
\end{array}$$

It should be noted that (121) and (122) generalize Theorem 5 in (Fagin, 1977) and that (123) can also be regarded as a generalization of rule MVD0 *(Complementation)* of (Beeri et al., 1977) to arbitrary x, y and z. (In the formulation of (Beeri et al., 1977), constraint $xyz \simeq id$ is implicit).

11.3 Handling context

Definition 5 imposes no constraints whatsoever to functions x, y and z apart from well-typing according to diagram (103). So one may wonder about what happens for particular instances of context z relative to antecedent x and consequent y. Below we look at the special cases where z is as injective as x, y, ! and id.

Before we deal with these special cases, let us recall our observation in section 7.5 that discriminating functions beyond $\simeq$-equivalence is irrelevant in reasoning

about data dependences. This happens because antecedent, consequent and context functions participate in (112) as arguments of operator *ker* . This means that such functions can always be *substituted* by other $\simeq$-equivalent functions, cf. eg.

$$z \simeq z' \quad \Rightarrow \quad (x \overset{R}{\rightharpoonup}_z y \Leftrightarrow x \overset{R}{\rightharpoonup}_{z'} y) \tag{124}$$

For its triviality, we consider case $z \simeq !$ in the first place:

$$\begin{array}{ll} & x \overset{R}{\rightharpoonup}_! y \\ \Leftrightarrow & \quad \{ \text{ (112) and (113) } \} \\ & xy \cdot R \cdot (\mathit{ker}\, x) \cdot R \cdot !^\circ \subseteq xy \cdot R \cdot !^\circ \\ \Leftarrow & \quad \{ \text{ monotonicity of composition } \} \\ & (\mathit{ker}\, x) \cdot R \cdot !^\circ \subseteq !^\circ \\ \Leftrightarrow & \quad \{ \text{ shunting ; (41) } \} \\ & (\mathit{ker}\, x) \cdot R \subseteq \top \\ \Leftrightarrow & \quad \{ \text{ every relation is below top } \} \\ & \textsc{True} \end{array}$$

Case $z \simeq x$ is also easy to handle:

$$\begin{array}{ll} & x \overset{R}{\rightharpoonup}_x y \\ \Leftrightarrow & \quad \{ \text{ (114) for } z \simeq x \} \\ & (\pi_{xy,x} R) \cdot (\pi_{x,x} R) \quad \subseteq \quad \pi_{xy,x} R \\ \Leftarrow & \quad \{ \text{ monotonicity of composition } \} \\ & \pi_{x,x} R \quad \subseteq \quad id \\ \Leftrightarrow & \quad \{ \text{ (92) } \} \\ & R \subseteq \mathit{ker}\, x \end{array}$$

From this we draw that, for coreflexives, all MVDs with antecedent as injective as context hold trivially.

Finally, we show that the remaining cases ($z \simeq y$ and $z \simeq id$) boil down to functional dependency $x \overset{T}{\rightharpoonup} y$, for T coreflexive. Concerning $z \simeq y$, we recall (107), which abbreviates

$$x \overset{T}{\rightharpoonup} y \quad \Rightarrow \quad \langle \forall\, z \;::\; x \overset{T}{\rightharpoonup}_z y \rangle$$

for T coreflexive. So, in particular,

$$x \overset{T}{\rightharpoonup} y \quad \Rightarrow \quad x \overset{T}{\rightharpoonup}_y y \tag{125}$$

holds. Conversely,

$$x \overset{T}{\rightharpoonup}_y y \quad \Rightarrow \quad x \overset{T}{\rightharpoonup} y \tag{126}$$

holds, cf.

$$
\begin{array}{ll}
 & x \overset{T}{\rightarrow\!\!\!\!\!\rightarrow}_{y} y \\
\Leftrightarrow & \quad \{ \text{ (116) } \} \\
 & \mathit{ker}\,(x \cdot T^{\circ}) \subseteq (\mathit{ker}\,xy) \cdot T \cdot \mathit{ker}\,y \\
\Rightarrow & \quad \{ \; y \leq xy \; ; \, T \subseteq \mathit{ker}\,y \; ; \text{ monotonicity } \} \\
 & \mathit{ker}\,(x \cdot T^{\circ}) \subseteq \mathit{ker}\,y \cdot \mathit{ker}\,y \cdot \mathit{ker}\,y \\
\Leftrightarrow & \quad \{ \text{ (26) twice } \} \\
 & \mathit{ker}\,(x \cdot T^{\circ}) \subseteq \mathit{ker}\,y \\
\Leftrightarrow & \quad \{ \text{ (32) } \} \\
 & x \overset{T}{\rightarrow} y
\end{array}
$$

(Noting that $T \cdot \mathit{ker}\,y \subseteq \mathit{ker}\,y$ is equivalent to $T \subseteq \mathit{ker}\,y$ (28), we can say that the reasoning above expands to symmetric relations at most the kernel of the consequent function y.) Putting (125, 126) together, we draw the equivalence

$$x \overset{T}{\rightarrow\!\!\!\!\!\rightarrow}_{y} y \quad \Leftrightarrow \quad x \overset{T}{\rightarrow} y \tag{127}$$

which means that, wherever context is as injective as consequent in MVDs over coreflexives, these degenerate to the corresponding FDs.

Finally, we deal with case $z \simeq id$ (ie. z is injective) in MVDs over coreflexives:

$$
\begin{array}{ll}
 & x \overset{T}{\rightarrow\!\!\!\!\!\rightarrow}_{id} y \\
\Leftrightarrow & \quad \{ \; \mathit{ker}\,id = id \; ; \text{ coreflexive } T \text{ is symmetric (116) } \} \\
 & \mathit{ker}\,(x \cdot T^{\circ}) \subseteq (\mathit{ker}\,xy) \cdot T \\
\Leftrightarrow & \quad \{ \text{ closure property (20) } \} \\
 & \mathit{ker}\,(x \cdot T^{\circ}) \subseteq \mathit{ker}\,xy \\
\Leftrightarrow & \quad \{ \text{ (33) } \} \\
 & x \overset{T}{\rightarrow} xy \\
\Leftrightarrow & \quad \{ \text{ (76) and (70) } \} \\
 & x \overset{T}{\rightarrow} y
\end{array}
$$

Therefore, equivalence

$$x \overset{T}{\rightarrow\!\!\!\!\!\rightarrow}_{id} y \quad \Leftrightarrow \quad x \overset{T}{\rightarrow} y$$

holds.

11.4 (Generic) inference rules for MVDs

We close the study of MVD inference rules in this report by discussing the generalization of the standard MVD axioms entailed by definition 5. As mentioned earlier

on, we do not consider rules which make difference of observations explicit — a price to pay so far for the generalization, see section 14. We adopt Maier (1983)'s enumeration and terminology:

- M1. **Reflexivity**: This has already been dealt with and generalized in (104), (117) and (118).
- M2. **Augmentation**: Our (generic) version of this rule

$$x \overset{T}{\rightharpoonup}_{zw} y \quad \Rightarrow \quad xw \overset{T}{\rightharpoonup}_{z} y \tag{128}$$

makes it explicit that the observation w which augments antecedent x cannot be arbitrary: it must be "taken out" from the context (as happens in the standard, set-theoretic approach). The calculation which underlies (128) assumes T coreflexive:

$$\begin{array}{ll} & x \overset{T}{\rightharpoonup}_{zw} y \\ \Leftrightarrow & \quad \{ \text{ (112) } \} \\ & T \cdot (\mathit{ker}\, x) \cdot T \subseteq (\mathit{ker}\, xy) \cdot T \cdot \mathit{ker}\, zw \\ \Rightarrow & \quad \{ \ x \leq xw \ \} \\ & T \cdot (\mathit{ker}\, xw) \cdot T \subseteq (\mathit{ker}\, xy) \cdot T \cdot \mathit{ker}\, zw \\ \Leftrightarrow & \quad \{ \ T \cdot (\mathit{ker}\, xw) \cdot T \subseteq \mathit{ker}\, w \text{ since } w \leq xw \text{ and } T \text{ is coreflexive } \} \\ & T \cdot (\mathit{ker}\, xw) \cdot T \subseteq (\mathit{ker}\, w) \cap ((\mathit{ker}\, xy) \cdot T \cdot \mathit{ker}\, zw) \\ \Leftrightarrow & \quad \{ \text{ (43) since } w \leq zw \ \} \\ & T \cdot (\mathit{ker}\, xw) \cdot T \subseteq (\mathit{ker}\, w \cap (\mathit{ker}\, xy) \cdot T) \cdot \mathit{ker}\, zw \\ \Leftrightarrow & \quad \{ \text{ (148) since } T \text{ is coreflexive } \} \\ & T \cdot (\mathit{ker}\, xw) \cdot T \subseteq (\mathit{ker}\, xwy) \cdot T \cdot \mathit{ker}\, zw \\ \Leftrightarrow & \quad \{ \text{ (112) } \} \\ & xw \overset{T}{\rightharpoonup}_{zw} y \\ \Rightarrow & \quad \{ \text{ (119) } \} \\ & xw \overset{T}{\rightharpoonup}_{z} y \end{array}$$

The version MVD2 of this rule given by Beeri et al. (1977) is written as follows in our notation

$$x \overset{T}{\rightharpoonup}_{zw} y \ \wedge \ v \leq w \quad \Rightarrow \quad xw \overset{T}{\rightharpoonup}_{z} yv \tag{129}$$

and is easily shown to stem from (128):

$$\begin{array}{ll} & xw \overset{T}{\rightharpoonup}_{z} yv \\ \Leftrightarrow & \quad \{ \text{ (119) under the assumption } v \leq w \ \} \end{array}$$

$$
\begin{array}{ll}
& xw \overset{T}{\rightharpoonup}_z y \\
\Leftarrow & \quad \{ \text{ (128) } \} \\
& x \overset{T}{\rightharpoonup}_{zw} y
\end{array}
$$

Is augmentation extensible to an arbitrary endo-relation R? Rule

$$w \leq xy \ \wedge \ x \overset{R}{\rightharpoonup}_z y \quad \Rightarrow \quad xw \overset{R}{\rightharpoonup}_z y$$

follows from (119) in the same way (68) follows from (67). Clearly, the cost of this generalization is a quite strong constraint on w.

- M3. **Additivity**: The interplay between consequents and contexts, which in the standard presentation is "frozen" to context being the complement of the union of antecedent and consequent, is particularly apparent from the generic layout of this inference rule, which Beeri et al. (1977) term *union*:

$$x \overset{T}{\rightharpoonup}_z y \ \wedge \ x \overset{T}{\rightharpoonup}_{zy} w \quad \Rightarrow \quad x \overset{T}{\rightharpoonup}_z yw \tag{130}$$

The calculation of (130) is based on auxiliary result (131) to follow shortly:

$$
\begin{array}{ll}
& x \overset{T}{\rightharpoonup}_z y \ \wedge \ x \overset{T}{\rightharpoonup}_{zy} w \\
\Rightarrow & \quad \{ \text{ augmentation (128) } \} \\
& x \overset{T}{\rightharpoonup}_z y \ \wedge \ xy \overset{T}{\rightharpoonup}_z w \\
\Rightarrow & \quad \{ \text{ (119) since } yw \leq xyw \ \} \\
& x \overset{T}{\rightharpoonup}_z y \ \wedge \ xy \overset{T}{\rightharpoonup}_z yw \\
\Rightarrow & \quad \{ \text{ (131) below, since coreflexive } T \text{ is cotransitive } \} \\
& x \overset{T}{\rightharpoonup}_z yw
\end{array}
$$

Finally, the auxiliary result assumed above

$$x \overset{R}{\rightharpoonup}_z y \ \wedge \ xy \overset{R}{\rightharpoonup}_z v \quad \Rightarrow \quad x \overset{R}{\rightharpoonup}_z v \tag{131}$$

is an inference rule valid for every cotransitive relation R (ie. such that $R \subseteq R \cdot R$). To show this we first derive the following consequence of $xy \overset{R}{\rightharpoonup}_z v$:

$$
\begin{array}{ll}
& xy \overset{R}{\rightharpoonup}_z v \\
\Leftrightarrow & \quad \{ \text{ (112) } \} \\
& R \cdot (\mathit{ker}\, xy) \cdot R \subseteq (\mathit{ker}\, xyv) \cdot R \cdot \mathit{ker}\, z \\
\Rightarrow & \quad \{ \ xv \leq xyv \ \} \\
& R \cdot (\mathit{ker}\, xy) \cdot R \subseteq (\mathit{ker}\, xv) \cdot R \cdot \mathit{ker}\, z \\
\Rightarrow & \quad \{ \text{ monotonicity of } (\cdot \mathit{ker}\, z) \text{ followed by (26) } \} \\
& R \cdot (\mathit{ker}\, xy) \cdot R \cdot \mathit{ker}\, z \subseteq (\mathit{ker}\, xv) \cdot R \cdot \mathit{ker}\, z
\end{array} \tag{132}
$$

Then we take (132) into account in showing that $x \overset{R}{\rightharpoonup}_z y$ entails $x \overset{R}{\rightharpoonup}_z v$:

$$
\begin{array}{ll}
 & x \overset{R}{\rightharpoonup}_z y \\
\Leftrightarrow & \quad \{ \text{ (112) } \} \\
 & R \cdot (\mathit{ker}\, x) \cdot R \subseteq (\mathit{ker}\, xy) \cdot R \cdot \mathit{ker}\, z \\
\Rightarrow & \quad \{ \text{ monotonicity of } (R \cdot) \ \} \\
 & R \cdot R \cdot (\mathit{ker}\, x) \cdot R \subseteq R \cdot (\mathit{ker}\, xy) \cdot R \cdot \mathit{ker}\, z \\
\Rightarrow & \quad \{ \ R \text{ assumed cotransitive } \} \\
 & R \cdot (\mathit{ker}\, x) \cdot R \subseteq R \cdot (\mathit{ker}\, xy) \cdot R \cdot \mathit{ker}\, z \\
\Rightarrow & \quad \{ \text{ (132) } \} \\
 & R \cdot (\mathit{ker}\, x) \cdot R \subseteq (\mathit{ker}\, xv) \cdot R \cdot \mathit{ker}\, z \\
\Leftrightarrow & \quad \{ \text{ (112) } \} \\
 & x \overset{R}{\rightharpoonup}_z v
\end{array}
$$

- M4. **Projectivity**: Putting (119) and (123) together, we obtain, for T coreflexive:

$$x \overset{T}{\rightharpoonup}_z yw \quad \Rightarrow \quad x \overset{T}{\rightharpoonup}_z y \ \wedge \ x \overset{T}{\rightharpoonup}_z w \tag{133}$$

cf.

$$
\begin{array}{ll}
 & x \overset{T}{\rightharpoonup}_z yw \\
\Leftrightarrow & \quad \{ \text{ (123) } \} \\
 & x \overset{T}{\rightharpoonup}_{yw} z \\
\Rightarrow & \quad \{ \text{ (119) twice, since } y \leq yw \text{ and } w \leq yw \ \} \\
 & x \overset{T}{\rightharpoonup}_y z \ \wedge \ x \overset{T}{\rightharpoonup}_w z \\
\Leftrightarrow & \quad \{ \text{ (123) twice } \} \\
 & x \overset{T}{\rightharpoonup}_z y \ \wedge \ x \overset{T}{\rightharpoonup}_z w
\end{array}
$$

Rule (133) compares favorably to its standard formulation (Maier, 1983) involving intersection and difference of y and z (regarded as "sets" of attributes). It clearly shows the advantage of handling context z explicitly in MVD reasoning.

- M5. **Transitivity**: See section 14.
- M6. **Pseudo-transitivity**: See section 14.
- M7. **Complementation**: Maier (1983) gives the *complementation axiom*

$$x \overset{T}{\rightharpoonup} y \quad \Rightarrow \quad x \overset{T}{\rightharpoonup} \overline{y} \tag{134}$$

as example of MVD axiom which has no FD counterpart. Clearly, (134) can be recognized as an instance of our generic *interchangeability rule* (123) for $z := \overline{y}$. This rule, however, is an *equivalence* and not just an *implication*. Although the rules of attribute complementation in (Maier, 1983) allow for the inference of the equivalence by mutual implication, our reasoning establishes the general equivalence *in one go* thanks to the power of pointfree equational reasoning when compared with traditional *implication-first* logic.

- C1. **Replication**: This has already been dealt with in (107). It can be observed that in our calculation of (107), all steps but one (110) are equivalences. Clearly, step (108) still holds for non-coreflexive relations provided these are at most $\ker x$, cf. (104). And step (109) still works for symmetric, cotransitive relations, although the equivalence gives room to an implication [24].

  All in all, we may say that the replication axiom extends to all symmetric, cotransitive sub-relations of $\ker x$. This singles out all *pers* at most $\ker x$ (an equivalence relation) as a class of relations in which both FDs and MVDs can be discussed as in the standard theory.

- C2. **Coalescence**: Our (generic) version of this inference rule

$$x \overset{T}{\rightarrow\!\!\!\!\!\cdot}_z y \;\wedge\; v \overset{T}{\rightarrow} w \;\wedge\; w \leq y \;\wedge\; v \leq z \quad \Rightarrow \quad x \overset{T}{\rightarrow} w$$

  is calculated as follows, for T coreflexive:

$$\begin{array}{ll} & x \overset{T}{\rightarrow\!\!\!\!\!\cdot}_z y \\ \Rightarrow & \quad \{ \text{ (121) since } v \leq z \text{ } \} \\ & x \overset{T}{\rightarrow\!\!\!\!\!\cdot}_v y \\ \Rightarrow & \quad \{ \text{ (135) below is applicable since } v \overset{T}{\rightarrow} w \text{ holds } \} \\ & x \overset{T}{\rightarrow\!\!\!\!\!\cdot}_w y \\ \Rightarrow & \quad \{ \text{ (122) since } w \leq y \text{ } \} \\ & x \overset{T}{\rightarrow\!\!\!\!\!\cdot}_w w \\ \Leftrightarrow & \quad \{ \text{ (127) } \} \\ & x \overset{T}{\rightarrow} w \end{array}$$

We are left with calculating the following rule, which is valid for R an arbitrary *per* (partial equivalence relation), and therefore for any coreflexive T (recall figure 2):

$$x \overset{R}{\rightarrow\!\!\!\!\!\cdot}_z y \;\wedge\; z \overset{R}{\rightarrow} w \quad \Rightarrow \quad x \overset{R}{\rightarrow\!\!\!\!\!\cdot}_w y \tag{135}$$

[24] This is due to (20), which holds as an implication when Φ is generalized to a cotransitive relation.

We reason:

$$
\begin{array}{ll}
& x \overset{R}{\twoheadrightarrow}_z y \ \wedge \ z \overset{R}{\twoheadrightarrow} w \\
\Leftrightarrow & \quad \{ \text{ (112) and (32) } \} \\
& R \cdot (\ker x) \cdot R \subseteq (\ker xy) \cdot R \cdot \ker z \ \ \wedge \ \ R \cdot (\ker z) \cdot R^\circ \subseteq \ker w \\
\Rightarrow & \quad \{ \text{ composition is monotonic (twice) } \} \\
& R \cdot (\ker x) \cdot R \cdot R^\circ \subseteq (\ker xy) \cdot R \cdot \ker z \cdot R^\circ \ \ \wedge \\
& (\ker xy) \cdot R \cdot R \cdot (\ker z) \cdot R^\circ \subseteq (\ker xy) \cdot R \cdot \ker w \\
\Leftrightarrow & \quad \{ \ R = R \cdot R = R \cdot R^\circ \text{ since } R \text{ is a } \textit{per} \ \} \\
& R \cdot (\ker x) \cdot R \subseteq (\ker xy) \cdot R \cdot \ker z \cdot R^\circ \ \ \wedge \\
& (\ker xy) \cdot R \cdot (\ker z) \cdot R^\circ \subseteq (\ker xy) \cdot R \cdot \ker w \\
\Rightarrow & \quad \{ \ \subseteq\text{-transitivity } \} \\
& R \cdot (\ker x) \cdot R \subseteq (\ker xy) \cdot R \cdot \ker w \\
\Leftrightarrow & \quad \{ \ (112) \ \} \\
& x \overset{R}{\twoheadrightarrow}_w y
\end{array}
$$

12 Epilogue

In its original set-theoretic setting, the equivalence between lossless decomposition and MVDs has been known for thirty years. So, what has been gained with its (pointfree) re-statement presented in the current report, after all?

If we compare our calculations with earlier expressions of the same results — see eg. (Fagin, 1977) and (Beeri et al., 1977) — it is clear that a sheer amount of detail was overlooked in their short-circuitous, almost telegram-like proofs, which were trusted on the basis of an almost informal common understanding of naïve set theory. This includes the use of two, seemingly equivalent, definitions for MVD, one universally quantified over pairs of tuples (4) and the other universally quantified over data values (5) and based on a set-valued function. While the latter is typical of earlier publications in the field (eg. (Fagin, 1977; Beeri et al., 1977)), the former is favored in textbooks such as Maier's (1983).

No dedicated proof has been produced — to the best of the author's knowledge — of the equivalence between these definitions. We close this report by calculating this equivalence as an exercise in relational *transposition*, a device commonly used in the PF-relational calculus. The main ingredient behind transposition is the fact that every binary relation R can be converted into a (set-valued) function ΛR via the *power-transpose* isomorphism (Bird and de Moor, 1997) defined by universal property

$$f = \Lambda\, R \quad \Leftrightarrow \quad (bRa \quad \Leftrightarrow \quad b \in f\ a) \tag{136}$$

This means that *any* set-valued function (eg. Y in definition 3) can be regarded as the *power-transpose* of some binary relation. Substitution $f := \Lambda\ R$ in (136) yields the so-called Λ-cancellation law $b \in (\Lambda R)a \quad \Leftrightarrow \quad bRa$, that is,

$$\in \cdot (\Lambda R) \quad = \quad R \tag{137}$$

which means that $(\Lambda R)a$ yields exactly the set of all b which R relates to a.

The theory behind relation transposition can be found in eg. (Bird and de Moor, 1997; Oliveira and Rodrigues, 2004). For our purposes below, it is enough to record the following property of the power-transpose [25]:

$$R \cdot \delta\, S = S \quad \Leftrightarrow \quad \Lambda R \cdot \delta\, S \subseteq \Lambda S \tag{138}$$

Our proof below of the equivalence between the two MVD definitions is, for coreflexive relations, a calculation which re-writes Maier's definition (1983) into (Beeri et al., 1977)'s:

$$\begin{array}{ll}
& x \overset{T}{\twoheadrightarrow}_z y \\
\Leftrightarrow & \quad \{\ \text{(123) since } T \text{ is coreflexive}\ \} \\
& x \overset{T}{\twoheadrightarrow}_y z \\
\Leftrightarrow & \quad \{\ (106)\ \} \\
& (\pi_{y,x}T) \cdot \pi_1 \cdot \delta\,(\pi_{y,xz}T) = \pi_{y,xz}T \qquad (139) \\
\Leftrightarrow & \quad \{\ (138)\ \} \\
& \Lambda(\pi_{y,x}T \cdot \pi_1) \cdot \delta\,(\pi_{y,xz}T) \subseteq \Lambda(\pi_{y,xz}T) \\
\Leftrightarrow & \quad \{\ \Lambda(R \cdot f) = (\Lambda R) \cdot f \text{ ; then introduce } \gamma_{g,f}T = \Lambda(\pi_{g,f}T)\ \} \\
& (\ \gamma_{y,x}T) \cdot \pi_1 \cdot \delta\,(\pi_{y,xz}T) \subseteq \gamma_{y,xz}T \\
\Leftrightarrow & \quad \{\ \text{shunting (23), since } \gamma_{y,x}T \cdot \pi_1 \text{ is a function ; } \delta\,(\pi_{y,xz}T) = img(xz \cdot T)\ \} \\
& img(xz \cdot T) \quad \subseteq \quad (\gamma_{y,x}T \cdot \pi_1)^\circ \cdot \gamma_{y,xz}T \\
\Leftrightarrow & \quad \{\ \text{go pointwise noting that, for } T \text{ coreflexive, } xz \cdot T \text{ is simple ; (30)}\ \} \\
& \langle \forall\ k\ :\ k \mathit{\ img\ } (xz \cdot T)\ k :\ (\gamma_{y,x}T \cdot \pi_1)k = (\gamma_{y,xz}T)k \rangle \\
\Leftrightarrow & \quad \{\ (143)\ \} \\
& \langle \forall\ k\ :\ \langle \exists\ t\ :\ t \in T :\ (xz\ t) = k \rangle :\ (\gamma_{y,x}T \cdot \pi_1)k = (\gamma_{y,xz}T)k \rangle \\
\Leftrightarrow & \quad \{\ \text{rename } k := (b,a) \text{ and simplify}\ \} \\
& \langle \forall\ a,b\ :\ \langle \exists\ t\ :\ t \in T :\ xz\ t = (a,b) \rangle :\ (\gamma_{y,x}T)\ a = (\gamma_{y,xz}T)(a,b) \rangle
\end{array}$$

We thus reach (5), the only difference being that function Y is generalized to

$$\gamma_{g,f} = \Lambda \cdot \pi_{g,f} \tag{140}$$

which is nothing but the power-transpose of a projection (86). So, while Y groups y-values only, γ's two parameters cater for any such groups of values:

$$(\gamma_{g,f}R)a = \{g\ b \mid \langle \exists\ a\ ::\ b\ R\ a\ \wedge\ c = f\ a \rangle\}$$

[25] This follows from exercise 4.48 in (Bird and de Moor, 1997).

13 Conclusions

This report puts forward a generalization of data dependency theory, the kernel of relational database design "à la Codd". Contrary to the intuition that a binary relation is just a particular case of n-ary relation, the report shows the effectiveness of the former in "explaining" and reasoning about the latter. It turns out that the theory becomes more general, better layered and more algebraic.

The adoption of the pointfree binary relation calculus is beneficial in several respects. First, pointfree notation abstracts from "points" or variables and makes the reasoning more compact and effective. Elegant formulæ such as eg. (44) — when compared with eg. (3) — come in support of this claim. Second, proofs are performed by easy-to-follow calculations in a *"let the symbols do the work"* reasoning style, reminiscent of school algebra.

It is the author's belief that this change in reasoning style is essential to data dependency theory "refactoring" as a whole, so as to meet current standards on proof by calculation (Bird and de Moor, 1997; Boute, 2003; Backhouse, 2004; Oliveira, 2008) and the original research aims of its pioneers, as expressed by Beeri et al. (1977) three decades ago: *a general theory that ties together dependencies, relations and operations on relations is still lacking*. Surely, the whole theory has advanced enormously in the thirty years which separate us today from the times of such highly innovative work. But the expected general theory has not yet become available because its kernel concepts have been kept too specific.

In the current report, one gets closer to such research aims by generalizing the original FD/MVD theory in two main directions: sets of tuples become binary relations and attributes generalize to arbitrary (suitably typed) functions. Awareness of function injectivity as *being what matters* in FD reasoning is another outcome of the generalization.

In retrospect, the use of *coreflexive* relations to model sets of tuples and predicates as binary relations is perhaps the main ingredient of the simplification and subsequent generalization. (The role of coreflexives as inspiring devices for pointfree transforming standard results in set theory can be observed elsewhere, eg. in (Oliveira and Rodrigues, 2004) concerning pointfree models of *hash tables* and most notably in (Doornbos et al., 1997) concerning well-foundedness and induction.)

To the best of our knowledge, this report presents the first comprehensive study of PF-transformed data dependency theory. Our main conclusion is that this theory would have been built perhaps rather differently should it be based on such a transformation in the first place. It is the complexity of formulæ (3) and (4) that led database theorists to invest solely into an axiomatic theory based on inference rules, closures of sets of dependencies and so on, instead of *calculating directly* from the definitions themselves, as we have shown it can be done once the latter are PF-transformed. We hope this launches a new approach in the field where new intuitions can be gathered simply by looking at PF-formulæ and their patterns.

But — in a sense — our contribution is still more qualitative than quantitative, as much work remains to be done. In the section which follows we browse a number of topics for research which we feel pointfree data dependency theory brings about. The lemma is to capitalize on the genericity and algebraic flavor of the approach and find synergies with other branches of computer science.

14 Related and future work

Foundational work FD-generalization opens paths for fresh developments, as is the case of results of functional programming theory which can be related to data dependency theory. For instance, injective functions are easily shown to be left-cancellable in a FD,

$$x \stackrel{R}{\preceq} y \quad \Leftrightarrow \quad f \cdot x \stackrel{R}{\preceq} f \cdot y \quad \Leftarrow \quad f \text{ is injective}$$

and the monotonicity of a parametric type F (relator) (Backhouse et al., 1992) with respect to the injectivity preorder (see equation (169) in the appendix) leads to structural FDs:

$$x \stackrel{R}{\preceq} y \quad \Leftrightarrow \quad \mathsf{F}x \stackrel{\mathsf{F}R}{\preceq} \mathsf{F}y$$

Moreover, positive impact of such a generic FD theory can be expected on the foundations of the current use of the functional dependency concept in *type level* functional programming (Hallgren, 2001; Duck et al., 2004). For instance, FDs are used by Silva and Visser (2006) *both* at the meta level (as a mechanism underlying multiple parameter type classes in the Haskell type system) and at the target level, where a strongly typed database model is defined.

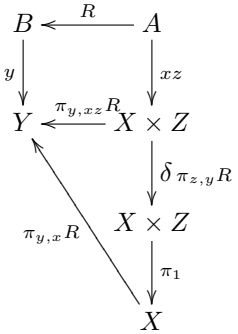
At the level of the PF-transform itself, our notion of *kernel* of a binary relation may look simplistic when compared to that adopted by Gibbons (2003), which applies to arbitrary relations the *greatest extension* of a *per* (partial equivalence relation) studied by Voermans (1999). This extension ensures kernels as equivalence relations (thus entire and reflexive) but is less agile and sacrifices some of the conceptual economy of our approach. Advantages of going in this direction need to be properly evaluated. Anyway, both approaches coincide on functions.

Another interesting topic at foundational level is the connection between binary relation projection and Reynolds "relation on functions" expressed by (90). This is worth studying in more detail, taking into consideration the corresponding point-free theory already developed by Backhouse and Backhouse (2004).

Last but not least, our approach should be related to the τ-category theory of relations given by Freyd and Scedrov (1990) based on *monic n-tuples*. Concepts such as *table*, *column*, *short column* etc. fit into the spirit of pointfree data dependency (and database) theory and should be carefully studied in this context.

MVD generalization Our experiments in generalizing data-dependency theory via the PF-transform show that, while FD theory broadens scope in a rather smooth way, MVD theory is not as straightforward to generalize. This is because it is not obvious how to extend attribute set intersection and difference to the generic view where such attributes become *arbitrary* functions. This is why two axioms were left out in section 11.4 which explicitly involve attribute set difference. Although related to the concept of function *complement* (58), a suitable notion of *difference* $f - g$ of two functions f and g with respect to the injectivity preorder (39) is not easy to define (Oliveira, 2006).

An alternative way to generalization could be taking the PF-version of the MVD definition given by Beeri et al. (1977), as captured by (139), as starting point. In fact, the type diagram of (139) displayed aside allows for a general



binary relation R when compared to the endorelation of (103).

Summing up, (generic) MVD theory requires further research and a new evaluation as a whole, once more definite (generic) results are obtained and the issues of soundness and completeness are thoroughly re-evaluated. This should include generalizing other kinds of data dependency — eg. *difunctional* dependencies (Jaoua et al., 2004) — which have not been dealt with at all in the current report.

Incomplete information Besides pointfree coverage of normal forms (Maier, 1983), *null-values* and partial information present another challenge, whereby antecedent and consequent functions of FD/MVDs become partial (ie. pure functions give place to *simple* relations). At first sight, transposition (136) has potential to map this new situation back to the one already dealt with in the current report, but semantically things are not that straightforward, as Maier (1983) takes some time to explain. The definition of *functional dependence with nulls* given by Lien (1982) PF-translates to

$$R \cdot ker\, X \cdot R^\circ \quad \subseteq \quad ker\, (\Lambda Y)$$

where antecedent X and consequent Y are demoted to simple relations. Levene and Loizou (1998) provide an interesting update on FDs with nulls by letting two (dual) approaches to partial information coexist: *strong* FDs and *weak* FDs. Both are defined in a modal style relying on all possible completions ("worlds") of a given incomplete relation R. In our PF approach this means keeping R and completing the antecedent and consequent of the given (partial) FD:

- Strong semantics:

$$\langle \forall\, f, g \;:\; X \subseteq f \;\wedge\; Y \subseteq g :\; f \stackrel{R}{\leftarrow} g \rangle$$

- Weak semantics:

$$\langle \exists\, f, g \;:\; X \subseteq f \;\wedge\; Y \subseteq g :\; f \stackrel{R}{\leftarrow} g \rangle$$

XML Functional dependencies have also been defined for XML documents. Vincent et al. (2007) address the equivalence between FDs in XML and relational FDs using the so called *closest node* approach. Another way of relating both kinds of FD relies on *tree tuples*, which are "flattened" representations of XML documents mapping paths allowed by the relevant type definition (DTD) to nodes in the corresponding XML tree. Both approaches are quite complex notation-wise and can benefit from the generality of the PF-transform taking into account treatment of similar nested structures such as hierarchical file systems as given in (Oliveira, 2009a).

Synergies with other theories Last but not least, we find that the more generic data dependency theory becomes, the more synergies will be found between classical database theory and computer science in general [26], hopefully blending everything into a unified framework. For instance, the upper-adjoint of (87) is of

[26] And discrete maths, recall (64,65) in section 7.4.

interest to reference (Barbosa et al., 2008) in developing a PF approach to bisimulations and invariants, in a coalgebraic setting. This shows a connection between the (apparently remote) fields of database design and automata (transition systems) which includes FDs holding for special classes of bisimulation, for instance.

Another synergy arises when comparing PF-transformed FDs and PF-transformed proof obligations capturing (extended) type-checking (ESC) as studied in (Oliveira, 2009a), where arrow notation $\Phi \xrightarrow{f} \Psi$ is adopted to mean that a given function f ensures post-condition Ψ once pre-conditioned by Φ. The analogy with FDs is apparent from fact

$$\Phi \xrightarrow{f} \Psi \quad \Leftrightarrow \quad \Psi \cdot f \leq \Phi$$

cf. (44). So functions and coreflexives swap places when compared with ESC arrows. The parallel between the FD-calculus and the ESC-calculus is well apparent by putting the rules of both calculi side by side. For instance, the *weakening/strengthening* rule of the latter (Oliveira, 2009a)

$$\Phi \xrightarrow{f} \Psi \quad \Leftarrow \quad \Phi \subseteq \Upsilon \ \wedge \ \Upsilon \xrightarrow{f} \Theta \ \wedge \ \Theta \subseteq \Psi$$

has the same "shape" of *decomposition rule* (67) given in this report. Our conclusion is that functional dependencies form a *type system* (Naik and Palsberg, 2005) for relational databases, a view which, barely implicit in the standard theory, is detailed in (Oliveira, 2009b).

Our current interests include yet another such synergy: the use of FDs to record and reason about software model properties in formal modeling. For instance, the invariant on the explosives store controller model of (Fitzgerald and Larsen, 1998) imposing that stores have unique names within sites is of course a FD, and many other examples abound in the literature of similar FDs embedded in formal models. Quite often, FDs arise as invariants in data refinement. For example, simple relations of type $B \leftarrow A$ are often refined into association lists of type $(B \times A)^\star$ subject to an invariant preventing duplication of 'keys' of type A. This invariant can be easily shown to be an FD, once every such list of pairs l is represented by simple relation $B \times A \xleftarrow{L} \mathbb{N}$ telling which pair takes which position in list. Relation $\pi_1 \cdot \rho\, L \cdot \pi_2^\circ$ relates all A and B recorded in L. Checking the simplicity of this relation easily yields FD $\pi_2 \overset{\rho\, L}{\rightarrowtail} \pi_1$.

Also interesting would be to study invariants such as that implicit in the universal *expression tree* datatype (see eg. (Oliveira, 2004)) imposing that at every node of a given expression tree, the number of sub-trees is functionally dependent on the associated operator symbol arity, for instance. Recursive invariants of this kind can be regarded as *inductive FDs*, a concept which, only hinted at by Necco and Oliveira (2002), deserves attention.

Our final hint for future research has to do with the Algebra of Programming itself, where we started from: Bird and de Moor (1997) present the following specification of sorting

$$Sort \quad \stackrel{\text{def}}{=} \quad [\![ordered]\!] \cdot ker\ bagify$$

where $bagify$ is the function mentioned in section 7.4 and $ordered$ is the predicate which tests whether a finite list is ordered wrt. some pre-defined (usually linear) order. Clearly, $Sort$ is below $ker\ bagify$ and therefore satisfies FD-reflexivity

$bagify \stackrel{Sort}{\leq} bagify$, thanks to (82). This example drives attention to specs which are *at most the kernel of a given function*, a pattern of practical interest which we have seen emerging from a number of inference rules in this report. In particular, such specs can be partial equivalence relations (*pers*). The fact perceived in the current report that most inference rules of standard data dependency theory extend from coreflexives (sets of tuples) to *pers* on arbitrary data domains is interesting: it tallies with the view expressed by Voermans (1999) that datatypes are better viewed as pers (data sets "quotiented" by sets of axioms which create equalities among their elements) than as coreflexives (data sets obeying particular data type invariants). Studying this extension in detail can be a promising way of freeing standard data dependency theory from its original relational database context.

Acknowledgements

The work reported in this report has been carried out in the context of the PURE Project *(Program Understanding and Re-engineering: Calculi and Applications)* funded by FCT (the Portuguese Science and Technology Foundation) under contract `POSI/ICHS/44304/2002`. It has been the subject of several presentations, the first one at the CAFEPURE seminar series (Oliveira, 2005) and the last at the IFIP WG 2.1 #62 Meeting in Namur, Belgium (Oliveira, 2006). Subsequent exchange of ideas with Jan van den Bussche, Zhenjiang Hu, Jeremy Gibbons, Joost Visser, Claudia Necco and Alexandra Silva are gratefully acknowledged.

References

C. Aarts, R.C. Backhouse, P. Hoogendijk, E.Voermans, and J. van der Woude. A relational theory of datatypes, December 1992. Available from `www.cs.nott.ac.uk/~rcb`.

T.L. Alves, P.F. Silva, J. Visser, and J.N. Oliveira. *Strategic term rewriting and its application to a VDM-SL to SQL conversion*. In *FM'05*, volume 3582 of *LNCS*, pages 399–414. Springer-Verlag, 2005.

American National Standards Institute and International Organization for Standardization. *American National Standard programming language COBOL, approved May 10, 1974: ANSI X3.23-1974: revision of X3.23-1968*, volume 21-1 of *Federal Information Processing Standards publication*. American National Standards Institute, 1430 Broadway, New York, NY 10018, USA, 1974.

K. Backhouse and R.C. Backhouse. Safety of abstract interpretations for free, via logical relations and Galois connections. *SCP*, 15(1–2):153–196, 2004.

R.C. Backhouse. *Mathematics of Program Construction*. Univ. of Nottingham, 2004. Draft of book in preparation. 608 pages.

R.C. Backhouse, P. de Bruin, P. Hoogendijk, G. Malcolm, T.S. Voermans, and J. van der Woude. Polynomial relators. In *AMAST'91*, pages 303–362. Springer, 1992.

J. Backus. Can programming be liberated from the von Neumann style? a functional style and its algebra of programs. *CACM*, 21(8):613–639, August 1978.

L.S. Barbosa, J.N. Oliveira, and A.M. Silva. *Calculating Invariants as Coreflexive Bisimulations*. In *AMAST'08*, volume 5140 of *LNCS*, pages 83–99. Springer-Verlag, 2008.

C. Beeri, R. Fagin, and J.H. Howard. A complete axiomatization for functional and multivalued dependencies in database relations. In D.C.P. Smith, editor, *Proc. 1977 ACM SIGMOD, Toronto*, pages 47–61, New York, NY, USA, 1977. ACM.

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

G. Booch, J. Rumbaugh, and I. Jacobson. *The Unified Modeling Language User Guide*. Addison Wesley Longman, Inc., 1999. ISBN 0-201-57168-4.

R. Boute. Concrete generic functionals: Principles, design and applications. In Jeremy Gibbons and Johan Jeuring, editors, *Generic Programming*, pages 89–119. Kluwer, 2003.

J. Bussche. Applications of Alfred Tarski's ideas in database theory. In *CSL '01: Proceedings of the 15th International Workshop on Computer Science Logic*, pages 20–37, London, UK, 2001. Springer-Verlag. ISBN 3-540-42554-3.

P.P. Chen. The entity-relationship model: toward a unified view of data. *ACM Trans. Database Syst.*, 1(1):9–36, 1976. ISSN 0362-5915. doi: http://doi.acm.org/10.1145/320434.320440.

E.F. Codd. A relational model of data for large shared data banks. *CACM*, 13(6): 377–387, June 1970.

E.F. Codd. *The Relational Model for Database Management: Version 2*. Addison Wesley Publishing Company, 2nd edition, 1990. ISBN 0-201-14192-2.

A. Cunha, J.N. Oliveira, and J. Visser. *Type-safe two-level data transformation*. In *FM'06* , volume 4085 of *LNCS*, pages 284–299. Springer-Verlag, Aug. 2006.

H. Doornbos, R. Backhouse, and J. van der Woude. A calculational approach to mathematical induction. *Theoretical Computer Science*, 179(1–2):103–135, 1997. ISSN 0304-3975. doi: 10.1016/S0304-3975(96)00154-5.

G.J. Duck, S.L. Peyton Jones, P.J. Stuckey, and M. Sulzmann. Sound and decidable type inference for functional dependencies. In David A. Schmidt, editor, *ESOP*, volume 2986 of *LNCS*, pages 49–63. Springer, 2004. ISBN 3-540-21313-9.

R. Fagin. Multivalued dependencies and a new normal form for relational databases. *ACM Trans. Database Syst.*, 2(3):262–278, 1977. ISSN 0362-5915. doi: http://doi.acm.org/10.1145/320557.320571.

S. Feferman. Tarski's influence on computer science. *Logical Methods in Computer Science*, 2:1–1–13, 2006. doi: 10.2168/LMCS-2(3:6)2006.

J. Fitzgerald and P.G. Larsen. *Modelling Systems: Practical Tools and Techniques for Software Development* . Cambridge University Press, 1st edition, 1998.

P.J. Freyd and A. Scedrov. *Categories, Allegories*, volume 39 of *Mathematical Library*. North-Holland, 1990.

H. Garcia-Molina, J.D. Ullman, and J.D. Widom. *Database Systems: The Complete Book*. Prentice Hall, 2002. ISBN: 0-13-031995-3.

J. Gibbons. When is a function a fold or an unfold?, 2003. Working document 833 FAV-12 available from the website of IFIP WG 2.1, 57th meeting, New York City, USA.

S. Givant. The calculus of relations as a foundation for mathematics. *J. Autom. Reasoning*, 37(4):277–322, 2006. ISSN 0168-7433. doi: http://dx.doi.org/10.1007/s10817-006-9062-x.

T. Hallgren. Fun with data dependencies. In *Proc. of the Joint CS/CE Winter Meeting*, pages 135–145. DCS of Chalmers Götebord University, 2001.

P. Hoogendijk. *A Generic Theory of Data Types*. PhD thesis, Univ. of Eindhoven, The Netherlands, 1997.

International Organization for Standardization. *ISO/IEC 13817-1:1996: Information technology — Programming languages, their environments and system software interfaces — Vienna Development Method — Specification Language — Part 1: Base language*. International Organization for Standardization, Geneva, Switzerland, 1996.

ISO. Information technology – database languages – SQL, Nov. 1992. Reference number ISO/IEC 9075:1992(E).

A. Jaoua, S. Elloumi, A. Hasnah, J. Jaam, and I. Nafkha. Discovering Regularities in Databases Using Canonical Decomposition of Binary Relations. *JoRMiCS*, 1: 217–234, 2004.

C.B. Jones, T.N. Nipkow, and M. Wolczko. MDB: A graph-like persistent database. In *Data Types and Persistence (Appin), Informal Proceedings*, pages 25–34, 1985.

M.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. ISBN 3-540-67262-1.

W. Kahl. Semigroupoid interfaces for relation-algebraic programming in Haskell. In *RelMiCS'06*, volume 4136 of *Lecture Notes in Computer Science*, pages 235–250. Springer, 2006.

Akihiro Kanamori. The empty set, the singleton, and the ordered pair. *The Bulletin of Symbolic Logic*, 9(3):273–298, 2003.

P.C. Kanellakis. Elements of relational database theory. *Handbook of theoretical computer science: formal models and semantics*, B:1073–1156, 1990.

E. Kreyszig. *Advanced Engineering Mathematics*. J. Wiley & Sons, 6th edition, 1988.

M. Levene and G. Loizou. Axiomatisation of functional dependencies in incomplete relations. *Theor. Comput. Sci.*, 206(1-2):283–300, 1998.

Y.E. Lien. On the equivalence of database models. *J. ACM*, 29(2):333–362, 1982. ISSN 0004-5411. doi: http://doi.acm.org/10.1145/322307.322311.

R.D. Maddux. The origin of relation algebras in the development and axiomatization of the calculus of relations. *Studia Logica*, 50:421–455, 1991.

D. Maier. *The Theory of Relational Databases*. Computer Science Press, 1983. ISBN 0-914894-42-0.

K. Matsuda, Z. Hu, K. Nakano, M. Hamana, and M. Takeichi. Bidirectionalization transformation based on automatic derivation of view complement functions, 2007. 12th ACM SIGPLAN International Conference on Functional Programming (ICFP 2007), Freiburg, Germany, October 1-3.

S.-C. Mu and R.S. Bird. Inverting functions as folds. In Eerke Boiten and Bernhard Möller, editors, *Sixth International Conference on Mathematics of Program Construction*, number 2386 in LNCS, pages 209–232. Springer-Verlag, July 2002. URL `citeseer.ist.psu.edu/mu02inverting.html`.

M. Naik and J. Palsberg. A type system equivalent to a model checker. In *ESOP*, volume 3444 of *Lecture Notes in Computer Science*, pages 374–388. Springer, 2005. ISBN 3-540-25435-8.

C. Necco and J.N. Oliveira. Generic data processing: A normalization exercise. In *CACIC'02* , pages 751–761, October 2002. 8th Argentinian Computer Science Congress, Univ. Buenos Aires, 15-18th October.

J.N. Oliveira. Invited paper: *Software Reification using the SETS Calculus*. In Tim Denvir, Cliff B. Jones, and Roger C. Shaw, editors, *Proc. of the BCS FACS 5th Refinement Workshop, Theory and Practice of Formal Software Development, London, UK*, pages 140–171. Springer-Verlag, 8–10 January 1992.

J.N. Oliveira. *Calculate databases with 'simplicity'*, September 2004. Presentation at the *IFIP WG 2.1 #59 Meeting*, Nottingham, UK. (Slides available from the author's website.).

J.N. Oliveira. Functional dependency theory made 'simpler'. Technical Report DI-PURe-05.01.01, DI/CCTC, University of Minho, Gualtar Campus, Braga, 2005. PUReCafé, 2005.01.18 [talk]; available from `http://wiki.di.uminho.pt/twiki/bin/view/Research/PURe/PUReCafe`.

J.N. Oliveira. *Data dependency theory made generic — by calculation*, December 2006. Presentation at the *IFIP WG 2.1 #62 Meeting*, Namur, Belgium. (Joint work with A. Silva and L.S. Barbosa. Slides available from the author's website).

J.N. Oliveira. *Transforming Data by Calculation*. In *GTTSE'07*, volume 5235 of *LNCS*, pages 134–195. Springer, 2008.

J.N. Oliveira. *Extended Static Checking by Calculation using the Pointfree Transform* . In A. Bove et al., editor, *LerNet ALFA Summer School 2008*, volume 5520 of *LNCS*, pages 195–251. Springer-Verlag, 2009a.

J.N. Oliveira. Functional dependencies: the under-appreciated type system of relational databases, 2009b. (In preparation).

J.N. Oliveira and C.J. Rodrigues. Transposing relations: from *Maybe* functions to hash tables. In *MPC'04*, volume 3125 of *LNCS*, pages 334–356. Springer, 2004.

P. O'Neil and E. O'Neil. *Database (2nd ed.): principles, programming, and performance*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2001. ISBN 1-55860-438-3.

V. Pratt. Origins of the calculus of binary relations. In *Proc. of the 7th Annual IEEE Symp. on Logic in Computer Science*, pages 248–254, Santa Cruz, CA, 1992. IEEE Comp. Soc.

A. Silva and J. Visser. Functional pearl: Strong types for relational databases. In *Proc. of the ACM SIGPLAN 2006 Haskell Workshop*, pages 25–37. ACM, September 2006. ISBN: 1-59593-489-8.

A. Tarski and S. Givant. *A Formalization of Set Theory without Variables*. American Mathematical Society, 1987. ISBN 0821810413. AMS Colloquium Publications, volume 41, Providence, Rhode Island.

J.D. Ullman. *Principles of Database and Knowledge-Base Systems*. Computer Science Press, 1988.

P.A.S. Veloso and A.M. Haeberer. A finitary relational algebra for classical first-order logic. *Bulletin of the Section of Logic*, 20:52–62, 1991.

M.W. Vincent, J. Liu, and M.K. Mohania. On the equivalence between FDs in XML and FDs in relations. *Acta Inf.*, 44(3-4):207–247, 2007.

T.S. Voermans. *Inductive Datatypes with Laws and Subtyping — A Relational Model*. PhD thesis, University of Eindhoven, The Netherlands, 1999.

A. Welsh. Formal methods for database language design and constraint handling. *IEEE Software Engineering Journal*, 4(1):15–24, January 1989.

APPENDIX

A Guarded inclusion and composition

The following two rules of the *pointfree transform* generalize standard relational inclusion and composition by addition of coreflexive relations which internalize constraints or logical guards:

Given two binary relations $B \xleftarrow{R,S} A$ *and two predicates* $2 \xleftarrow{\psi} A$ *and* $2 \xleftarrow{\phi} B$ *(coreflexively denoted by* Ψ *and* Φ*, respectively), then*

$$\langle \forall\, b, a \;:\; (\phi\; b) \;\wedge\; (\psi\; a) :\; b\; R\; a \Rightarrow b\; S\; a \rangle \quad \Leftrightarrow \quad \Phi \cdot R \cdot \Psi \subseteq S \tag{141}$$

□

Clearly, (17) instantiates (141) for $\Phi = \Psi = id$. Concerning (guarded) composition, we have:

Given two binary relations $B \xleftarrow{R} A$ *and* $A \xleftarrow{S} C$ *and predicate* ϕ *(coreflexively denoted by* $A \xleftarrow{\Phi} A$ *), we have that, for all* b, c

$$\langle \exists\, a \;:\; \phi\, a :\; b\, R\, a \;\wedge\; a\, S c \rangle \quad \Leftrightarrow \quad b(R \cdot \Phi \cdot S)c \tag{142}$$

holds and extends relational composition (for $\Phi = id$ *we are back to* $R \cdot S$*).*
□

For R a function f in (142) and S its converse, one obtains the image of pre-conditioned $f \cdot \Phi$ (a coreflexive):

$$\begin{array}{lcll} b(f \cdot \Phi \cdot f^\circ)c & \Leftrightarrow & b(img\,(f \cdot \Phi))c & \\ & \Leftrightarrow & b = c \;\wedge\; \langle \exists\, a \;:\; \phi\, a :\; b = f\, a \rangle & (143) \end{array}$$

B Some properties of relational *meet*

Backhouse (2004) infers the following distribution properties

$$R \cdot (S \cap T) = (R \cdot S) \cap (R \cdot T) \quad \Leftarrow \quad (ker\, R) \cdot S \subseteq S \vee (ker\, R) \cdot T \subseteq T \tag{144}$$
$$(T \cap S) \cdot U = (T \cdot U) \cap (S \cdot U) \quad \Leftarrow \quad T \cdot img\, U \subseteq T \vee S \cdot img\, U \subseteq S \tag{145}$$

from the so-called *modular identity* rule

$$R \cap (S \cdot T) \quad \subseteq \quad ((R \cdot T^\circ) \cap S) \cdot T \tag{146}$$

also known as the *Dedekind rule*. Among other consequences of (146) we find

$$\langle \forall\, S, T \;::\; R \cdot S \cap T = R \cdot (S \cap T) \rangle \quad \Leftrightarrow \quad R \subseteq id \tag{147}$$

in the same reference.

Our rule (43) is an instance of

$$R \cap (S \cdot T) = (R \cap S) \cdot T \quad \Leftarrow \quad T \subseteq R \text{ and } R \text{ symmetric and transitive} \tag{148}$$

which is another, obvious consequence of (146), easy to prove by mutual inclusion:

$$\begin{array}{ll} & R \cap (S \cdot T) \\ \subseteq & \quad \{\ (146)\ \} \\ & ((R \cdot T^\circ) \cap S) \cdot T \\ \subseteq & \quad \{\ T \subseteq R \text{ is assumed}\ \} \\ & ((R \cdot R^\circ) \cap S) \cdot T \\ \subseteq & \quad \{\ R \text{ assumed symmetric and transitive}\ \} \\ & (R \cap S) \cdot T \\ \subseteq & \quad \{\ \text{monotonicity of composition}\ \} \end{array}$$

$$
\begin{array}{ll}
& R \cdot T \cap S \cdot T \\
\subseteq & \quad \{\ T \subseteq R \text{ again ; monotonicity of composition } \} \\
& R \cdot R \cap S \cdot T \\
\subseteq & \quad \{\ R \text{ assumed transitive } \} \\
& R \cap (S \cdot T)
\end{array}
$$

C Some properties of relational "split"

Relational *split* is a special case of *meet*, as can be checked by PF-transforming (47):

$$\langle R, S \rangle \quad = \quad \pi_1^\circ \cdot R \cap \pi_2^\circ \cdot S \tag{149}$$

Together with (145), (149) leads straight to (guarded) $\times$-fusion (96) and, in turn, to the following corollaries of $\times$-fusion (96): fusion takes place wherever T is simple,

$$\langle R, S \rangle \cdot T = \langle R \cdot T, S \cdot T \rangle \quad \Leftarrow \quad T \text{ is simple} \tag{150}$$

and wherever R (or S) is simple and T is its converse,

$$\langle R, S \rangle \cdot R^\circ = \langle img\ R, S \cdot R^\circ \rangle \quad \Leftarrow \quad R \text{ is simple} \tag{151}$$

$$\langle R, S \rangle \cdot S^\circ = \langle R \cdot S^\circ, img\ S \rangle \quad \Leftarrow \quad S \text{ is simple} \tag{152}$$

Together with (147), (149) leads straight to the following "split" pre-conditioning rule

$$\langle R, S \rangle \cdot \Phi = \langle R, S \cdot \Phi \rangle \tag{153}$$

for Φ coreflexive, which is required in a step of the proof of lossless decomposition (111) and is easy to justify:

$$
\begin{array}{ll}
& \langle R, S \rangle \cdot \Phi = \langle R, S \cdot \Phi \rangle \\
\Leftrightarrow & \quad \{\ (149)\ \} \\
& (\pi_1^\circ \cdot R \cap \pi_2^\circ \cdot S) \cdot \Phi = \pi_1^\circ \cdot R \cap \pi_2^\circ \cdot S \cdot \Phi \\
\Leftrightarrow & \quad \{\ \text{converses and commutativity } \} \\
& \Phi \cdot (S^\circ \cdot \pi_2 \cap R^\circ \cdot \pi_1) = (\Phi \cdot S^\circ \cdot \pi_2) \cap (R^\circ \cdot \pi_1) \\
\Leftarrow & \quad \{\ (147)\ \} \\
& \Phi \subseteq id
\end{array}
$$

C.1 The "split twist" rule

Another step of the proof of lossless decomposition (111) is based on the following equivalence,

$$\langle R, S \rangle \subseteq \langle U, V \rangle \cdot X \quad \Leftrightarrow \quad \langle R, id \rangle \cdot S^\circ \subseteq \langle U, X^\circ \rangle \cdot V^\circ \tag{154}$$

itself a consequence of

$$\langle R, S \rangle \cdot T \subseteq \langle U, V \rangle \cdot X \quad \Leftrightarrow \quad \langle R, T^\circ \rangle \cdot S^\circ \subseteq \langle U, X^\circ \rangle \cdot V^\circ \tag{155}$$

for $T := id$. In order to prove (155), we reason in terms of universally quantified points x, y and z:

$$\begin{array}{ll} & (y,z)\ (\langle R,S\rangle \cdot T)\ x \\ \Leftrightarrow & \quad \{\ \text{composition and split (47)}\ \} \\ & \langle \exists\, u\ ::\ y\ R\ u\ \wedge\ z\ S\ u\ \wedge\ u\ T\ x\rangle \\ \Leftrightarrow & \quad \{\ \text{converses}\ \} \\ & \langle \exists\, u\ ::\ y\ R\ u\ \wedge\ x\ T^{\circ}\ u\ \wedge\ u\ S^{\circ}\ z\rangle \\ \Leftrightarrow & \quad \{\ \text{split and composition again}\ \} \\ & (y,x)\ (\langle R,T^{\circ}\rangle \cdot S^{\circ})\ z \end{array}$$

Similarly,

$$(y,z)\ \langle U,V\rangle \cdot X\ x \quad \Leftrightarrow \quad (y,x)\ \langle U,X^{\circ}\rangle \cdot V^{\circ}\ z$$

Therefore:

$$\begin{array}{ll} & (y,z)\ (\langle R,S\rangle \cdot T)\ x \Rightarrow (y,z)\ \langle U,V\rangle \cdot X\ x \\ \Leftrightarrow & \quad \{\ \text{logical implication is a congruence}\ \} \\ & (y,x)\ (\langle R,T^{\circ}\rangle \cdot S^{\circ})\ z \Rightarrow (y,x)\ \langle U,X^{\circ}\rangle \cdot V^{\circ}\ z \end{array}$$

holds, which shrinks to (155) once points are dropped.

C.2 "Splits" involving difunctional relations

A relation S is difunctional iff $S \cdot S^{\circ} \cdot S = S$ holds (Bird and de Moor, 1997; Jaoua et al., 2004). This is equivalent to $S \cdot S^{\circ} \cdot S \subseteq S$ since $S \subseteq S \cdot S^{\circ} \cdot S$ holds for every S (Bird and de Moor, 1997; Backhouse, 2004).

Two extreme instances of difunctional relations are $\top$ and $\bot$. That every *simple* relation is difunctional is easy to check: it only requires relaxing function f in the *shunting* rules (23, 24) to a simple relation S, leading to equivalences (Mu and Bird, 2002)

$$S \cdot R \subseteq T \quad \Leftrightarrow \quad (\delta\, S) \cdot R \quad \subseteq \quad S^{\circ} \cdot T \tag{156}$$

$$R \cdot S^{\circ} \subseteq T \quad \Leftrightarrow \quad R \cdot \delta\, S \quad \subseteq \quad T \cdot S \tag{157}$$

which, albeit similar to (23, 24), are not Galois connections. These rules involve the δ (domain) operator, which satisfies properties

$$\delta\, R \quad = \quad \mathit{ker}\, R \cap id \tag{158}$$

$$R \cdot \delta\, R \quad = \quad R \tag{159}$$

among many others not used in this report [27]. The calculation of $S \cdot S^{\circ} \cdot S \subseteq S$ for simple S follows:

$$S \cdot S^{\circ} \cdot S \subseteq S$$

[27] See eg. (Bird and de Moor, 1997; Backhouse, 2004) for details.

$$
\begin{array}{ll}
\Leftrightarrow & \quad \{ \text{ converses } \} \\
& S^\circ \cdot S \cdot S^\circ \subseteq S^\circ \\
\Leftrightarrow & \quad \{ \text{ (156) } \} \\
& S^\circ \cdot S \cdot \delta\, S \subseteq S^\circ \cdot S \\
\Leftrightarrow & \quad \{ \text{ (159) } \} \\
& S^\circ \cdot S \subseteq S^\circ \cdot S \\
\Leftrightarrow & \quad \{ \text{ trivial } \} \\
& \text{TRUE}
\end{array}
$$

Therefore, functions are (as expected) difunctional, as are coreflexives, which are also simple.

A trivial monotonicity argument will show that symmetry and transitivity entail difunctionality. So partial equivalence relations (*pers* in figure 2) also belong to the class of difunctional relations. It is also easy to show that kernels of difunctional relations are transitive and cotransitive at the same time,

$$
ker\, S \cdot ker\, S = ker\, S \quad \Leftarrow \quad S \text{ is difunctional} \tag{160}
$$

In the case of functions, difunctionality combines with shunting (23, 24) and leads to (28, 29). The calculation of (28) is as follows (that of (29) can be obtained by taking converses):

$$
\begin{array}{ll}
& S \subseteq (ker\, f) \cdot R \\
\Leftrightarrow & \quad \{ \text{ (23) } \} \\
& f \cdot S \subseteq f \cdot R \\
\Leftrightarrow & \quad \{ \text{ (25) } \} \\
& f \cdot f^\circ \cdot f \cdot S \subseteq f \cdot R \\
\Leftrightarrow & \quad \{ \text{ (23) } \} \\
& f^\circ \cdot f \cdot S \subseteq f^\circ \cdot f \cdot R \\
\Leftrightarrow & \quad \{ \text{ (11) } \} \\
& (ker\, f) \cdot S \subseteq (ker\, f) \cdot R
\end{array}
$$

The following equality holds for difunctional S

$$
\langle S, R \cdot ker\, S \rangle \quad = \quad \langle S, R \rangle \cdot ker\, S \tag{161}
$$

cf.

$$
\begin{array}{ll}
& \langle S, R \rangle \cdot S^\circ \cdot S \\
= & \quad \{ \text{ (96) since } S \text{ is difunctional } (S = S \cdot S^\circ \cdot S) \} \\
& \langle S \cdot S^\circ \cdot S, R \cdot S^\circ \cdot S \rangle \\
= & \quad \{ \ S \text{ is difunctional } (S = S \cdot S^\circ \cdot S) \ \} \\
& \langle S, R \cdot S^\circ \cdot S \rangle
\end{array}
$$

The following equality holds for simple (thus difunctional) S

$$\langle R, T\rangle \cdot S \quad = \quad \langle R, T \cdot img\, S\rangle \cdot S \tag{162}$$

cf.

$$\begin{array}{ll} & \langle R, T\rangle \cdot S \\ = & \quad \{\ S \text{ is difunctional } \} \\ & \langle R, T\rangle \cdot S \cdot S^\circ \cdot S \\ = & \quad \{\ (153) \text{ since } S \text{ is simple } \} \\ & \langle R, T \cdot S \cdot S^\circ\rangle \cdot S \end{array}$$

D A little preorder construction

The concept of a *preorder* — ie. that of a *reflexive* and *transitive* endo-relation (figure 2) — is central to the mathematics of computing. It paves the way to Galois connections and other interesting topics (eg. lexicographic orders, etc.). In this annex we concentrate on a particular preorder construction which is used extensively in this report. For more about preorders see eg. (Aarts et al., 1992) and (Bird and de Moor, 1997).

D.1 The construction

Let $A \xleftarrow{\sqsubseteq} A$ be a preorder. Given a function $A \xleftarrow{h} B$, the relation $B \xleftarrow{\preceq} B$ defined by

$$\preceq \quad \stackrel{\text{def}}{=} \quad h^\circ \cdot \sqsubseteq \cdot h \tag{163}$$

is also a preorder: it is reflexive,

$$\begin{array}{ll} & id \quad \subseteq \quad \preceq \\ \Leftrightarrow & \quad \{\ (163) \text{ and shunting (23) } \} \\ & h \quad \subseteq \quad \sqsubseteq \cdot h \\ \Leftarrow & \quad \{\ (\cdot h) \text{ is monotonic } \} \\ & id \quad \subseteq \quad \sqsubseteq \\ \Leftrightarrow & \quad \{\ \sqsubseteq \text{ is a preorder, thus reflexive } \} \\ & \text{TRUE} \end{array}$$

and transitive:

$$\begin{array}{ll} & \preceq \cdot \preceq \\ = & \quad \{\ (163) \text{ twice; associativity of composition } \} \\ & h^\circ \cdot \sqsubseteq \cdot (h \cdot h^\circ) \cdot \sqsubseteq \cdot h \end{array}$$

$$
\begin{array}{ll}
\subseteq & \quad \{\ h \text{ is simple (9)}\ \} \\
& h^\circ \cdot \sqsubseteq \cdot \sqsubseteq \cdot h \\
\subseteq & \quad \{\ \sqsubseteq \text{ is a preorder, thus transitive}\ \} \\
& h^\circ \cdot \sqsubseteq \cdot h \\
= & \quad \{\ (163)\ \} \\
& \preceq
\end{array}
$$

Relations $\sqsubseteq$ and $\preceq$ in (163) will be referred to below as the *base* preorder and the *derived* one, respectively, the former being at most the latter iff h is a $\sqsubseteq$-monotonic endofunction over B,

$$\sqsubseteq \subseteq \preceq \quad \Leftrightarrow \quad h \textit{ is } \sqsubseteq\textit{-monotonic} \tag{164}$$

cf.

$$
\begin{array}{ll}
& \sqsubseteq \subseteq \preceq \\
\Leftrightarrow & \quad \{\ \text{definition of } \preceq \text{ (163) followed by (23)}\ \} \\
& h \cdot \sqsubseteq \ \subseteq \ \sqsubseteq \cdot h \\
\Leftrightarrow & \quad \{\ \text{definition of } \sqsubseteq\text{-monotonicity}\ \} \\
& h \textit{ is } \sqsubseteq\textit{-monotonic}
\end{array}
$$

D.2 Example

The *injectivity* preorder defined by (39) in the main body of the report is an example of this construction, for $h := ker$, $\preceq := \leq^\circ$ and $\sqsubseteq := \subseteq$:

$$\leq^\circ \ = \ ker^\circ \cdot \subseteq \cdot ker \tag{165}$$

that is,

$$\leq \ = \ ker^\circ \cdot \subseteq^\circ \cdot ker$$

(Note the extra converse operator in $\subseteq^\circ$.) Since *ker* is monotonic, (40) in the main body of the report is an instance of (164): $R \subseteq S$ implies $S \leq R$. (Again note the effect of converse.)

D.3 Preorder homomorphism

By construction, (163) establishes h as a preorder homomorphism — cf.

$$a \preceq a' \quad \Leftrightarrow \quad h\ a \ \sqsubseteq \ h\ a'$$

in pointwise notation — which can be exploited to "lift" results from the $\sqsubseteq$ to the $\preceq$ order. We present two such results, one concerning monotonicity and the other concerning Galois connections. For space economy, both will be presented restricted to endo-functions. (The general formulation is similar.)

D.4 Lifting monotonicity

Let $A \xleftarrow{\sqsubseteq} A$, $A \xleftarrow{h} B$ and $B \xleftarrow{\preceq} B$ be as above. Let $A \xleftarrow{k} A$ be a $\sqsubseteq$-monotonic endo-function, and k' be such that

$$h \cdot k' \quad = \quad k \cdot h \tag{166}$$

holds, cf. diagram:

$$\begin{array}{ccccc} B & \xleftarrow{\preceq} & B & \xleftarrow{k'} & B \\ {\scriptstyle h}\downarrow & & {\scriptstyle h}\downarrow & & \downarrow{\scriptstyle h} \\ A & \xleftarrow[\sqsubseteq]{} & A & \xleftarrow[k]{} & A \end{array}$$

Then k' is $\preceq$-monotonic,

$$k' \cdot \preceq \quad \subseteq \quad \preceq \cdot k' \tag{167}$$

cf.

$$\begin{array}{ll} & (167) \\ \Leftrightarrow & \quad \{ \text{ shunting } \} \\ & \preceq \quad \subseteq \quad k'^{\circ} \cdot \preceq \cdot k' \\ \Leftrightarrow & \quad \{ \text{ (163) twice ; (14) } \} \\ & h^{\circ} \cdot \sqsubseteq \cdot h \quad \subseteq \quad (h \cdot k')^{\circ} \cdot \sqsubseteq \cdot h \cdot k' \\ \Leftrightarrow & \quad \{ \text{ (166) twice ; (14) } \} \\ & h^{\circ} \cdot \sqsubseteq \cdot h \quad \subseteq \quad h^{\circ} \cdot k^{\circ} \cdot \sqsubseteq \cdot k \cdot h \\ \Leftarrow & \quad \{ \ (h^{\circ} \cdot \_ \cdot h) \text{ is monotonic } \} \\ & \sqsubseteq \quad \subseteq \quad k^{\circ} \cdot \sqsubseteq \cdot k \\ \Leftrightarrow & \quad \{ \text{ since } k \text{ is assumed } \sqsubseteq\text{-monotonic } \} \\ & \text{TRUE} \end{array}$$

Note that (166) equivales $k(h \leftarrow h)k'$ — recall (90) — meaning that they are h-homomorphic.

D.5 Examples

From

$$ker\,(R \cdot T) \quad = \quad T^{\circ} \cdot (ker\,R) \cdot T \tag{168}$$

we identify $h = ker$, $k = (T^{\circ} \cdot \_ \cdot T)$ and $k' = (\ \cdot T)$ satisfying (166). Since *ker* is $\subseteq$-monotonic, from (167) we draw that $(\ \cdot T)$ is $\leq^{\circ}$-monotonic, which is equivalent to being $\leq$-monotonic. This justifies equation (42) in the main body of the report.

A similar argument can be provided to justify $\leq$-monotonicity of any relator F (Backhouse et al., 1992; Bird and de Moor, 1997),

$$R \leq S \quad \Rightarrow \quad \mathsf{F}\, R \leq \mathsf{F}\, S \tag{169}$$

for $k = k' = \mathsf{F}$, since F is $\subseteq$-monotonic and

$$\mathit{ker}\,(\mathsf{F}\, R) = \mathsf{F}(\mathit{ker}\, R)$$

holds.

D.6 Lifting Galois connections

Suppose that functions $A \xleftarrow{k,j} A$ are Galois connected under preorder $\sqsubseteq$

$$k^\circ \cdot \sqsubseteq \quad = \quad \sqsubseteq \cdot j \tag{170}$$

and that k', j' are h-homomorphic to lower-adjoint k and upper-adjoint j, respectively:

$$h \cdot k' \quad = \quad k \cdot h \tag{171}$$

$$h \cdot j' \quad = \quad j \cdot h \tag{172}$$

Then k', j' are $\preceq$-Galois connected,

$$k'^\circ \cdot \preceq \quad = \quad \preceq \cdot j' \tag{173}$$

as proved below:

$$\begin{array}{ll} & k'^\circ \cdot \preceq \\ = & \quad \{\ (163)\ \} \\ & k'^\circ \cdot h^\circ \cdot \sqsubseteq \cdot h \\ = & \quad \{\ (171) \text{ and converses}\ \} \\ & h^\circ \cdot k^\circ \cdot \sqsubseteq \cdot h \\ = & \quad \{\ (170) \text{ followed by } (172)\ \} \\ & h^\circ \cdot \sqsubseteq \cdot h \cdot j' \\ = & \quad \{\ (163)\ \} \\ & \preceq \cdot j' \end{array}$$

D.7 Examples

Consider instances $T := g^\circ$ and $T := g$ in (168), for some function g of appropriate type. Then build the corresponding instances of k, k' (renamed j, j' in the second case to avoid name clashing):

$$T := g^\circ \qquad \begin{cases} k' = (\ \cdot g^\circ) \\ k = (g \cdot \_ \cdot g^\circ) \end{cases}$$

$$T := g \qquad \begin{cases} j' = (\ \cdot g) \\ j = (g^\circ \cdot \_ \cdot g) \end{cases}$$

The fact that k and j are Galois connected is an instance of (87):

$$g \cdot X \cdot g^\circ \subseteq Y \quad \Leftrightarrow \quad X \subseteq g^\circ \cdot Y \cdot g$$

Then, from (173) we draw

$$k'^\circ \cdot \leq^\circ \quad = \quad \leq^\circ \cdot j'$$

which, taking converses, is the same as

$$j'^\circ \cdot \leq \quad = \quad \leq \cdot k'$$

that is,

$$(\ \cdot g)^\circ \cdot \leq \quad = \quad \leq \cdot (\ \cdot g^\circ)$$

Thus (59) holds.

A similar argument will justify Galois connection (50), stemming from relational *split* being *ker*-homomorphic to relational *meet* (49), which is the upper-adjoint in its defining Galois connection:

$$T \subseteq R \cap S \quad \Leftrightarrow \quad T \subseteq R \wedge T \subseteq S \tag{174}$$

Because of the extra converse in $\leq^\circ$ in (165), the fact that *meet* is the upper-adjoint wrt. $\subseteq$ casts *split* as the lower-adjoint wrt. $\leq$:

$$\langle R, S \rangle \leq T \quad \Leftrightarrow \quad R \leq T \ \wedge \ S \leq T$$

Readers wishing to check this more explicitly are invited to follow the argument which follows:

$$\begin{array}{ll} & \langle R, S \rangle \leq T \\ \Leftrightarrow & \quad \{ \text{ (39) and (49) } \} \\ & \mathit{ker}\, T \subseteq (\mathit{ker}\, R) \cap (\mathit{ker}\, S) \\ \Leftrightarrow & \quad \{ \text{ (174) } \} \\ & \mathit{ker}\, T \subseteq \mathit{ker}\, R \quad \wedge \quad \mathit{ker}\, T \subseteq \mathit{ker}\, S \\ \Leftrightarrow & \quad \{ \text{ (39) twice } \} \\ & R \leq T \quad \wedge \quad S \leq T \end{array}$$