Transposing Relations: From Maybe Functions to Hash Tables

Jos´e Nuno Fonseca deOliveira and C´esar de Jesus Pereira CunhaRodrigues

2002

22/ June / 2012

Samih Eisa

Paper Context

- Functional Transposition (FT)
- Converting Relations into functions
- To develop relational algebra via the algebra of functions
- In particular, transposition of binary relations

$$A \xrightarrow{R} B$$

Binary Relation

Binary relations	Description
$R \cdot S$	composition
$R \cup S$	union
$\perp$	empty relation
id	identity relation
$R \subseteq S$	inclusion
$R \subseteq id$, $\neg R = id - R$	coreflexive relations
$\delta\, R$	domain of R

Why we need FT ?

- Functions have rich theory

- They can be
  - Dualized – injection
  - Galois connected – converse
  - Parametrically polymorphic

- Therefore, we can exploit the calculation power of functions

- namely “ free Theorems” reasoning

But

- Functions are not enough for some situations

- Undefined for some of their input data (Partial function)

- Functions might give non-deterministic output (Maybe values rather than values)
  - Where Maybe is data type
    - Maybe a = Nothing| just a

Cope with Non-deterministic output

- Functional Programmer structured the codomain of such functions as set or list of values

- Such Powerset valued functions are models of binary relations

$$bRa \text{ means } b \in (f\ a)$$

- Any R is uniquely transposed into set value function

Set value Fucntion

$$f = \Lambda\, R \equiv (bRa \equiv b \in f\ a)$$

- Λ : Transpose Operator
- Analogy, we can define the conversion of Maybe-value Function as follows:
-

$$f = \Gamma\ R \equiv (bRa \equiv (f\ a = Just\, b))$$

- Λ is not enough for transposing relations

Binary relation Taxonomy

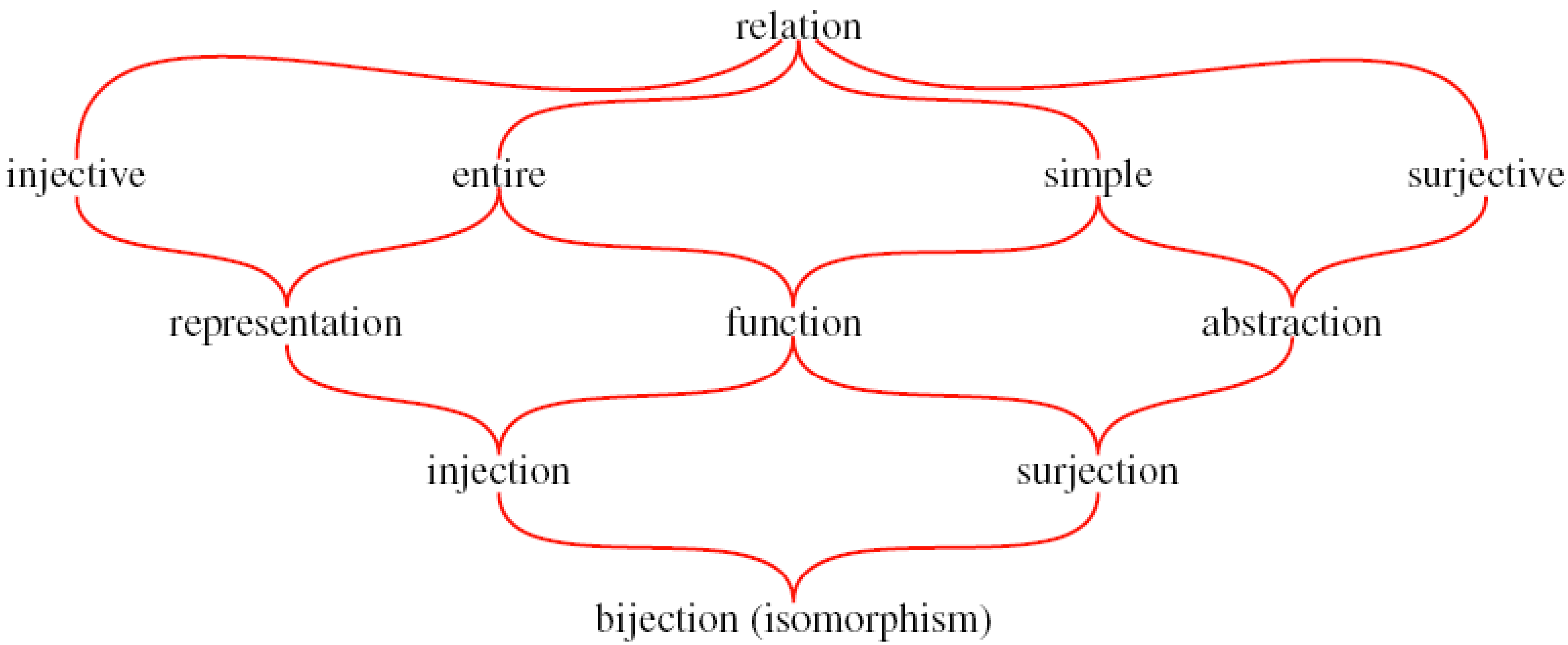

Fig. 1. Binary relation taxonomy

Need

- Unified, generic transpose construct to collect type other than powerset valued functions

- Solution :

- Using hash tables for efficient data representation

Generic Transposition

- How to derive laws of relational combinators as free Theorems

- Power-transpose
- Maybe-transpose

$$
\begin{aligned}
f = \Lambda\, R &\equiv (R = \in \cdot f) \\
f = \Gamma\, R &\equiv (R = i_1^{\circ} \cdot f)
\end{aligned}
$$

Hash Transpose

- Hash tables are static and dynamic storage of date

- Random access is normally achieved by a hash function

$$B \xleftarrow{h} A$$

Data Collision can be handled either by
•Linear probing or
•Over follow

Hash Transpose

- Overflow handling consists in partitioning a given data collection into n-many disjoint buckets

- Each one addressed by hash index
- Can be modeled as:

$$a \in S \equiv a \in t(h\ a)$$

Hashing as a transpose

- Derive previous equation

- Hash transpose:

$$t = \Lambda(S \cdot h^{\circ})$$

The Paper in Points

- Basis for Generic transposition
- Two instances of transposition are considered
  - Any relations
  - Simple relations
- Relate the topic of functional transposition with hashing for data representation