Where is the Science in the Software Sciences?

MAPi 2017

FEUP, 15th Sep 2017

J.N. Oliveira

INESC TEC & University of Minho

Informatics ≃ Computer "Science"?

Vinton Cerf (1943-)

ACM President

DOI:10.1145/2347736.2347737

Where is the Science in Computer Science?

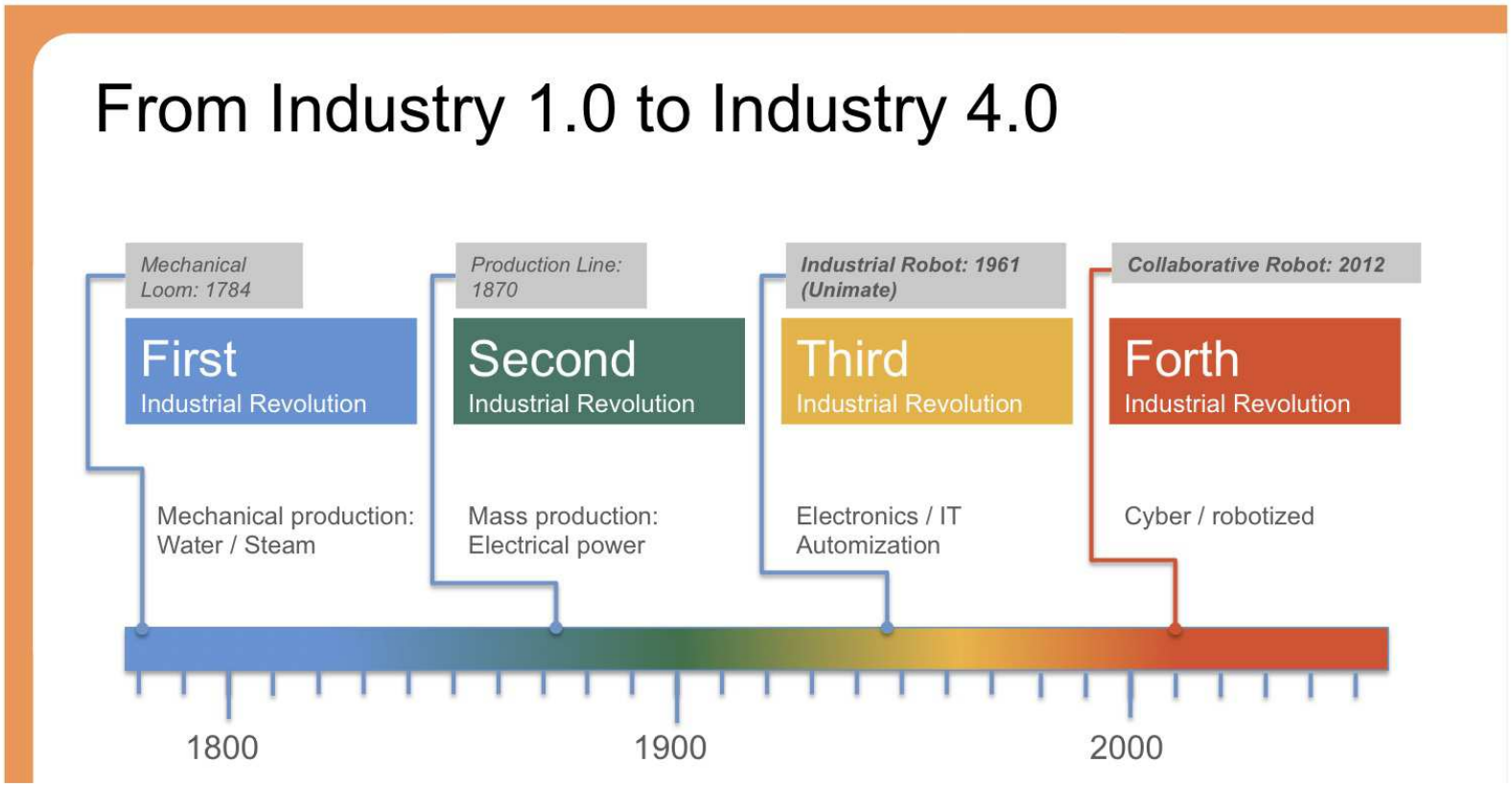
"... we have a responsibility to pursue the science in computer science. We must develop better tools and much deeper understanding of the systems we invent and a far greater ability to make predictions about the behavior of these complex, connected, and interacting systems.".

(Vinton G. Cerf, Letter from the ACM President, CACM 55(10), Oct. 2012)

MAPi DOCTORAL PROGRAM IN COMPUTER SCIENCE

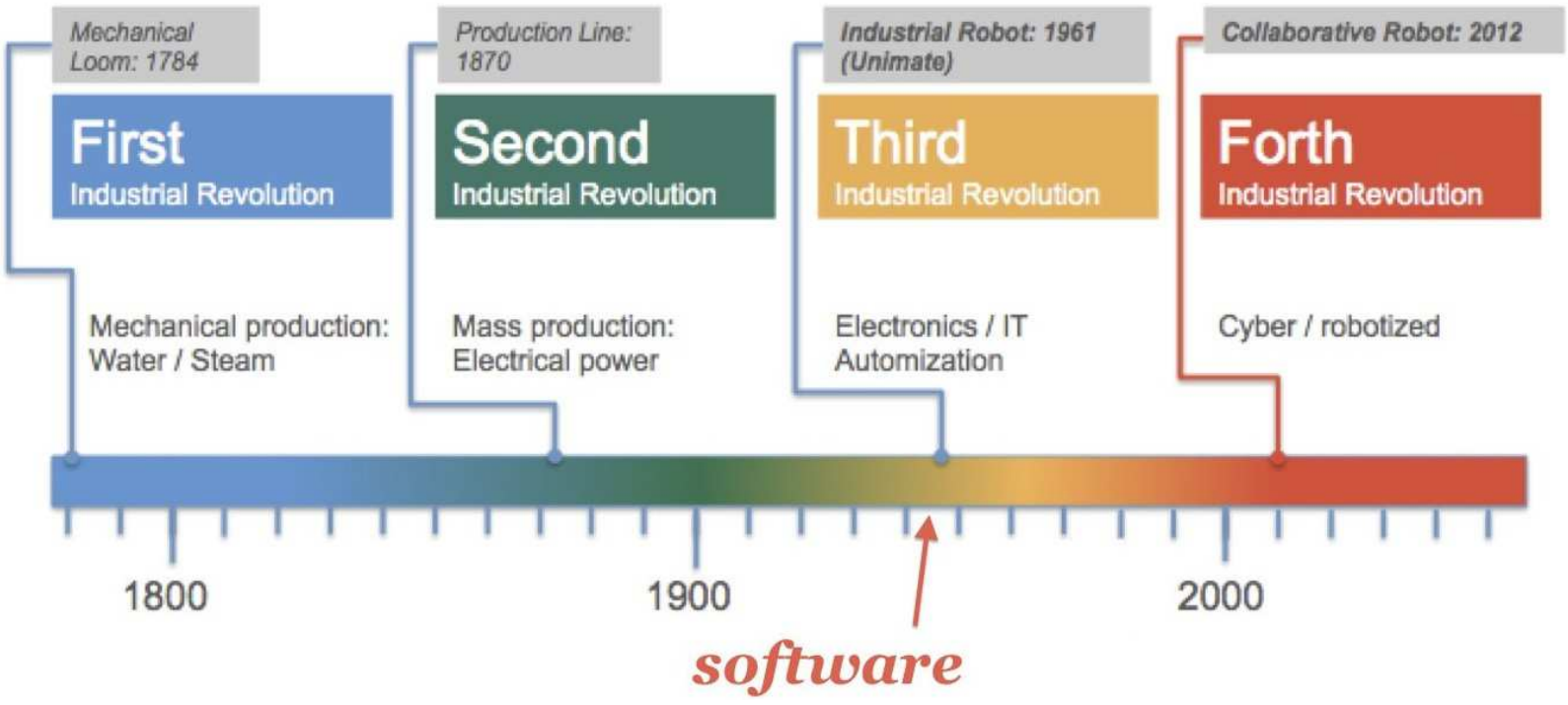
for $i = 1$ *to* 4 **do** $\{\, industry\ (i); \,\}$

HASLab INESCTEC



Credit: nelmia Robotics Insight (2015)

Turning point

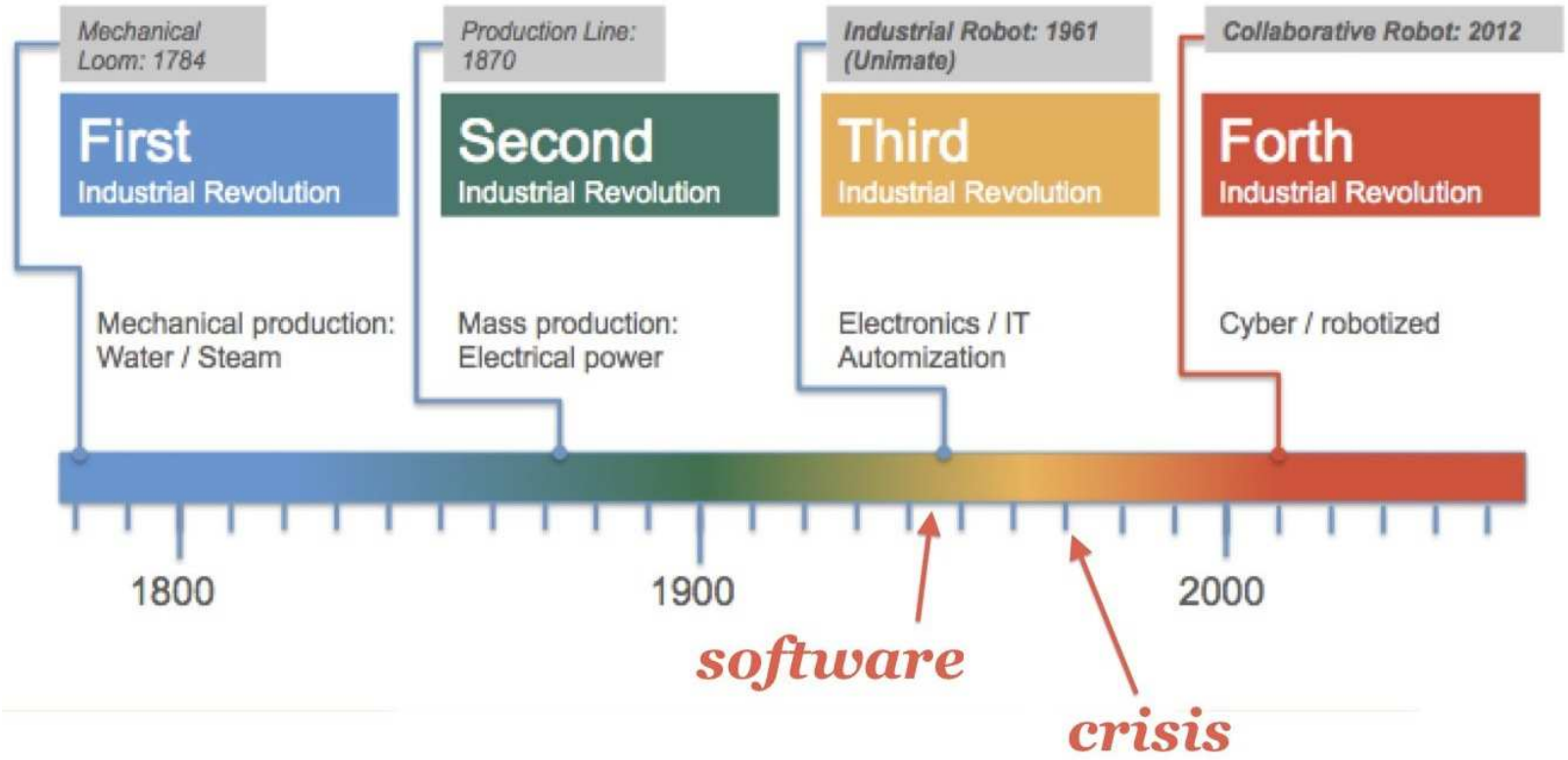


("L'enfant terrible" is born)

MAPi DOCTORAL PROGRAM IN COMPUTER SCIENCE

Crisis

HASLab INESCTEC



1st NATO Conference on Software Engineering, Darmstadt, October 1968

MAPi DOCTORAL PROGRAM IN COMPUTER SCIENCE

Meanwhile (50 years)

HASLab HIGH-ASSURANCE SOFTWARE LABORATORY INESCTEC

Software {
- **Process** — 
- **Product** — 

MAPi DOCTORAL PROGRAM IN COMPUTER SCIENCE

"L'enfant terrible"

HASLab INESCTEC

Hardware and other "traditional" industrial **products** fabricated according to the laws of **physics**.

Software is not governed by the laws of physics:

- it does not weight / does not smell
- it does not warm up / cool down
- it is chemically neutral ...

Anthony Oettinger (ACM President, 1967):

"(...) the scientific, rigorous component of computing, is more like **mathematics** *than it is like* **physics".**

MAPi DOCTORAL PROGRAM IN COMPUTER SCIENCE

Software = mathematics in motion

HASLab HIGH-ASSURANCE SOFTWARE LABORATORY INESCTEC

Can one *pretend* that **software production** is not affected by its **special nature** and move on?

Many people have tried to do so for 50 years...

Still Oettinger **(already** in 1967):

"It is a matter of **complexity**. *Once you start putting thousands of these instructions together you create a* **monster** *which is* **unintelligible** *to anyone save its creator and, most of the time, unfortunately even to the creator."*

MAPi DOCTORAL PROGRAM IN COMPUTER SCIENCE

1976 — vintage year

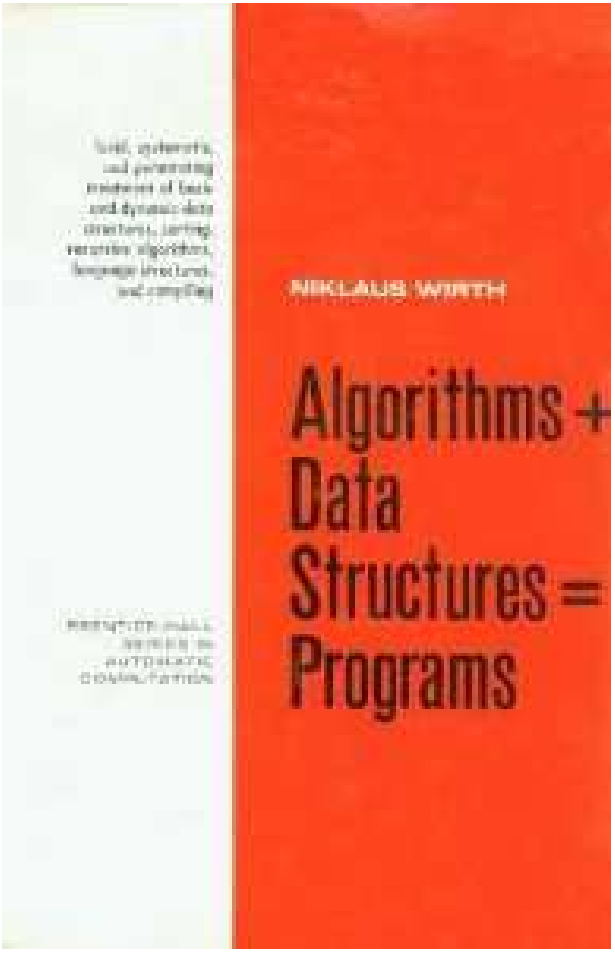
HASLab INESCTEC

Famous textbook by Niklaus Wirth (1934-) that inspired **structured programming** and made history in its time.

Published in 1976.

PASCAL — the language "par exellence" (mostly in Europe).

No **Java** yet. Object orientation stimulated by MODULA, a PASCAL derivative.



MAPi DOCTORAL PROGRAM IN COMPUTER SCIENCE

1976 — vintage year

HASLab INESCTEC

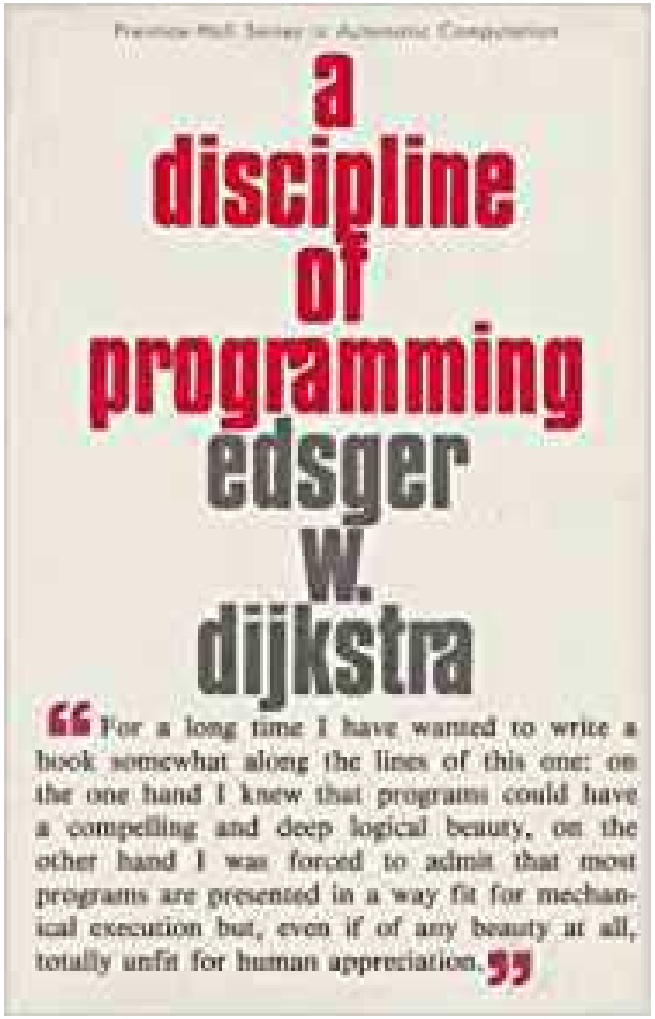
The "textbook of all textbooks".

Emphasis on elegance and abstraction.

Also published in 1976.

Programming as a scientific activity.

E.W. Dijkstra (1930-2002), Turing Award, one of the most eminent computer scientists ever.



MAPi DOCTORAL PROGRAM IN COMPUTER SCIENCE

Shortly afterwards

HASLab INESCTEC

Famous "dragon book" by A.V. Aho (1941-) e J.D. Ullman (1942-).

Published in 1977.

Inspiration for generations of compiler writers.

Compiler writing made systematic.

Scientific basis: *The Theory of Parsing, Translating & Compiling* (1972) by the same authors.

MAPi DOCTORAL PROGRAM IN COMPUTER SCIENCE

Still people complain (1976)

HASLab INESCTEC

E.W. Dijkstra (1920-2002)

(...) the term "Database Technology", although sometimes used, is immature, for there is hardly any underlying "science" that could justify the use of the term "technology".

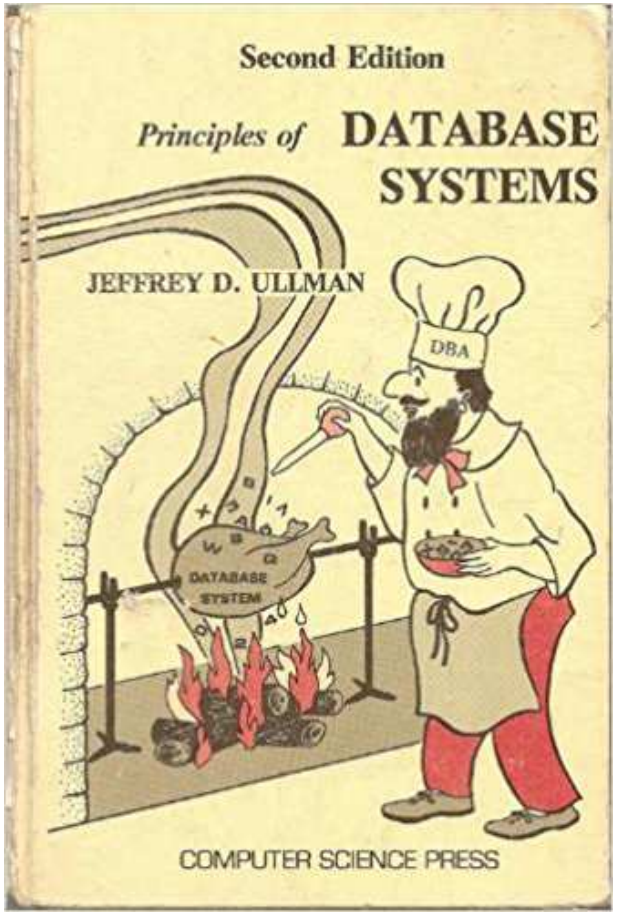
(...) They seem to form an inbred crowd with very little knowledge of computing science in general (...) Often they seemed to be mentally trapped by the intricacies of early, rather **ad hoc** *solutions (...)*

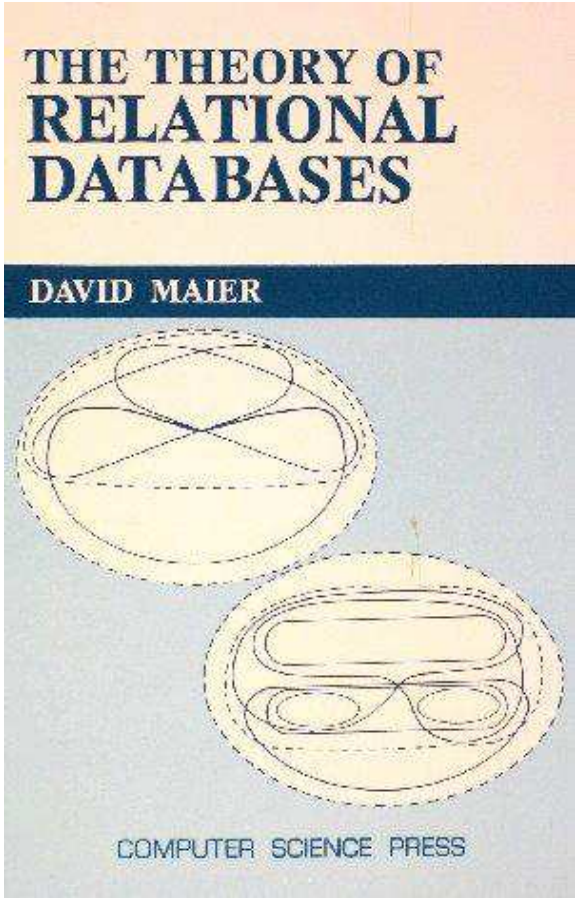
(E.W. Dijkstra, EWD577, 1976)

MAPi DOCTORAL PROGRAM IN COMPUTER SCIENCE

Reaction (early 1980s)

HASLab INESCTEC



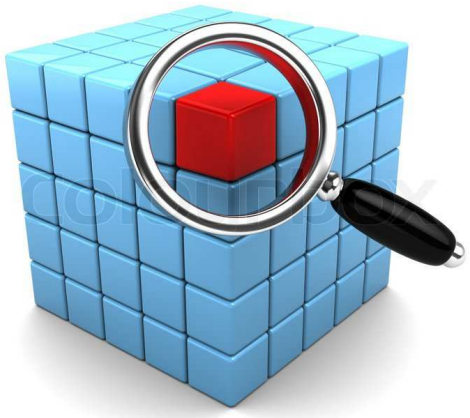


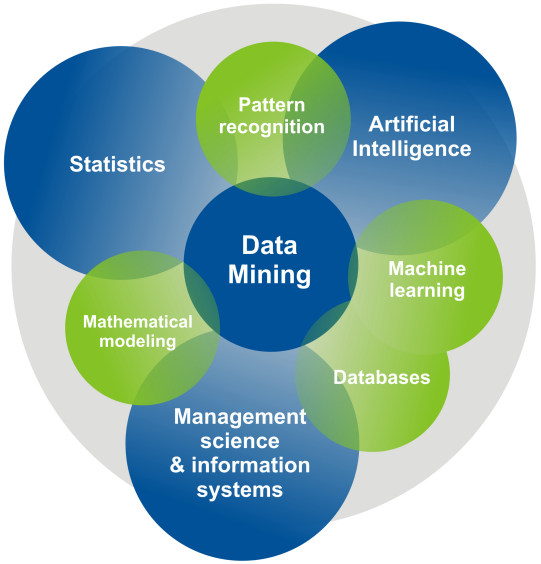
Relational database science — books by Ullman (1982) and Maier (1983).

MAPi DOCTORAL PROGRAM IN COMPUTER SCIENCE

>30 years later

HASLab HIGH-ASSURANCE SOFTWARE LABORATORY INESCTEC





Have scientific standards kept up with so higher demands in technology?

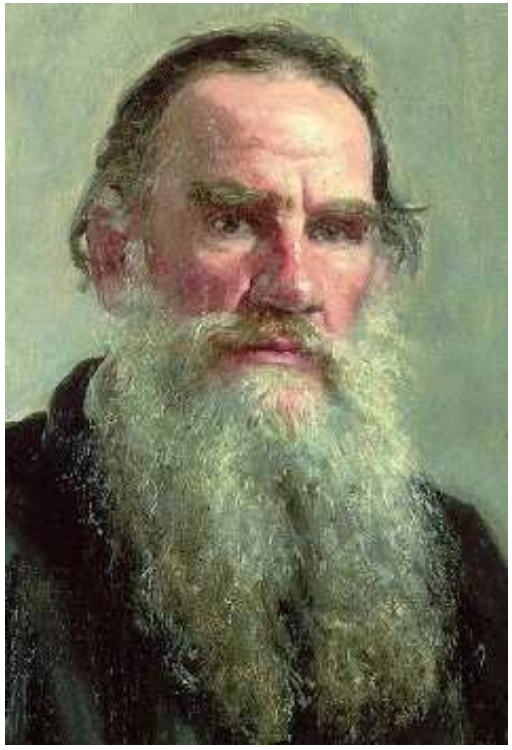
MAPi DOCTORAL PROGRAM IN COMPUTER SCIENCE

A much older concern

HASLab INESCTEC

"Only by taking infinitesimally small units for observation (the **differential** *of history, that is, the individual tendencies of men) and attaining to the art of* **integrating** *them (that is, finding the sum of these infinitesimals) can we hope to arrive at the laws of history."*

Leo Tolstoy, "War and Peace"
- Book XI, Chap.II (1869)



L. Tolstoy (1828–1910)

The oldest definition I know of **data-mining**.

MAPi DOCTORAL PROGRAM IN COMPUTER SCIENCE

The age of data

HASLab INESCTEC

Note the **maths** flavour of Tolstoy's text

How maths-strong is data-mining?

Have we attained the "**art of integration**" foreseen by Tolstoy?

Again the **glorious 1970s** — since the early days of psychometrics in the **social sciences** — **linear algebra** (LA) central to data analysis (e.g. Tucker tensor decompositions etc).

All done? Anything else to do?

L.R. Tucker (1910-2004)

MAPi DOCTORAL PROGRAM IN COMPUTER SCIENCE

Yes, perhaps...

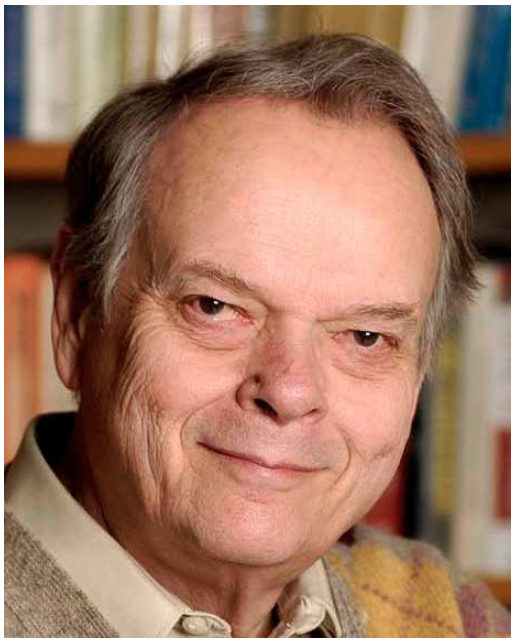
HASLab HIGH-ASSURANCE SOFTWARE LABORATORY INESCTEC

Data mining seems to be ignoring the role of **types** and **type parametricity** in software — one of the most significant advances in CS.

Nice theory called **parametric polymorphism** (John Reynolds, CMU).

So nice that you can derive **properties** of your (typed) operations **before writing them** (!)

What can **parametricity** possibly mean in the **data-analysis** setting?



J.C. Reynolds
(1935–2013)

Example — parametric data cubes

Back to square one — some raw data:

$t =$

#	Model	Year	Color	Sale
1	*Chevy*	1990	*Red*	5
2	*Chevy*	1990	*Blue*	87
3	*Ford*	1990	*Green*	64
4	*Ford*	1990	*Blue*	99
5	*Ford*	1991	*Red*	8
6	*Ford*	1991	*Blue*	7

Columns — attributes — the **observables**
Rows — records (n-many) — the **infinitesimals**

Column-orientation — each column (attribute) A represented by a function $t_A : n \to A$ such that $a = t_A\,(i)$ means "a is the value of attribute A in record nr i".

MAP i DOCTORAL PROGRAM IN COMPUTER SCIENCE

Records are tuples

HASLab HIGH-ASSURANCE SOFTWARE LABORATORY INESCTEC

Can records be rebuilt from such **attribute projection** functions?

Yes — by **tupling** them.

Tupling: *Given functions* $f : A \to B$ *and* $g : A \to C$, *their tupling is the function* $f \triangledown g$ *such that*

$$(f \triangledown g)\ a = (f\ a, g\ a)$$

For instance,

$$(t_{Color} \triangledown t_{Model})\ 2 = (Blue, Chevy),$$
$$(t_{Year} \triangledown (t_{Color} \triangledown t_{Model}))\ 3 = (1990, (Green, Ford))$$

and so on.

MAP i DOCTORAL PROGRAM IN COMPUTER SCIENCE

The typed LA approach

HASLab HIGH-ASSURANCE SOFTWARE LABORATORY INESCTEC

Represent functions by Boolean matrices.

Given (finite) types A and B, any function

$$f : A \to B$$

can be represented by a matrix $[\![f]\!]$ with A-many columns and B-many rows such that, for any $b \in B$ and $a \in A$, matrix cell

$$b\ [\![f]\!]\ a = \begin{cases} 1 \Leftarrow b = f\ a \\ 0\ \mathit{otherwise} \end{cases}$$

NB: Following the **infix** notation usually adopted for relations (which are Boolean matrices) — for instance $y \leqslant x$ — we write $y\ M\ x$ to denote the contents of the cell in matrix M addressed by row y and column x.

MAPi DOCTORAL PROGRAM IN COMPUTER SCIENCE

The typed LA approach

HASLab INESCTEC

One projection function (matrix) per **dimension** attribute:

t_{Model}	1	2	3	4	5	6
Chevy	1	1	0	0	0	0
Ford	0	0	1	1	1	1

t_{Year}	1	2	3	4	5	6
1990	1	1	1	1	0	0
1991	0	0	0	0	1	1

t_{Color}	1	2	3	4	5	6
Blue	0	1	0	1	0	1
Green	0	0	1	0	0	0
Red	1	0	0	0	1	0

#	**Model**	**Year**	**Color**	**Sale**
1	Chevy	1990	Red	5
2	Chevy	1990	Blue	87
3	Ford	1990	Green	64
4	Ford	1990	Blue	99
5	Ford	1991	Red	8
6	Ford	1991	Blue	7

NB: we tend to abbreviate $[\![f]\!]$ by f where the context is clear.

The typed LA approach

Note how the inverse of a function is also represented by a Boolean matrix, e.g.

t°_{Model}	Chevy	Ford
1	1	0
2	1	0
3	0	1
4	0	1
5	0	1
6	0	1

versus

t_{Model}	1	2	3	4	5	6
Chevy	1	1	0	0	0	0
Ford	0	0	1	1	1	1

— no longer a function, but still a relation, a Boolean matrix :-)

Clearly,

$$j\ t^{\circ}_{Model}\ a = a\ t_{Model}\ j$$

Given a matrix M, M° is known as the **transposition** of M.

The typed LA approach

We **type** matrices in the same way as functions: $M : A \to B$ means a matrix M with A-many columns and B-many rows.

Matrices are **arrows**: $A \xrightarrow{M} B$ denotes a matrix from A (source) to B (target), where A, B are (finite) types.

Writing $B \xleftarrow{M} A$ means the same as $A \xrightarrow{M} B$.

Composition — *aka* matrix multiplication:

$$B \xleftarrow{M} A \xleftarrow{N} C \qquad M \cdot N : A \to C$$

$$b(M \cdot N)c \;=\; \langle \sum\, a \;::\; (b\; M\; a) \times (a\; N\; c) \rangle$$

MAPi DOCTORAL PROGRAM IN COMPUTER SCIENCE

The typed LA approach

HASLab INESCTEC

Function composition implemented by matrix multiplication, $[\![f \cdot g]\!] = [\![f]\!] \cdot [\![g]\!]$

Identity — the identity matrix id corresponds to the identity function and is such that

$$M \cdot id = M = id \cdot M \tag{1}$$

Function tupling corresponds to the so-called **Khatri-Rao product** $M \triangledown N$ defined index-wise by

$$(b, c)\ (M \triangledown N)\ a \quad = \quad (b\ M\ a) \times (c\ N\ a) \tag{2}$$

Khatri-Rao is a "column-wise" version of the well-known **Kronecker product** $M \otimes N$:

$$(y, x)\ (M \otimes N)\ (b, a) \quad = \quad (y\ M\ b) \times (x\ N\ a) \tag{3}$$

Typing data

The raw data given above is represented in typed LA by the expression

$$v = (t_{Year} \triangledown (t_{Color} \triangledown t_{Model})) \cdot (t^{Sale})^{\circ}$$

of type

$$v : 1 \to (Year \times (Color \times Model))$$

depicted aside.

Year x (Color x Model)			ALL
1990	Blue	Chevy	87
		Ford	99
	Green	Chevy	0
		Ford	64
	Red	Chevy	5
		Ford	0
1991	Blue	Chevy	0
		Ford	7
	Green	Chevy	0
		Ford	0
	Red	Chevy	0
		Ford	8

v is a **multi-dimensional** column vector — a **tensor**. Datatype $1 = \{\text{ALL}\}$ is the so-called **singleton** type.

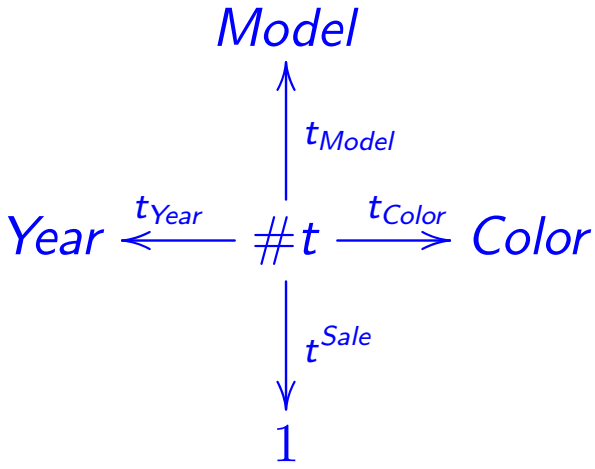
Dimensions and measures

Sale is a special kind of data — a **measure**. Measures are encoded as **row** vectors, e.g.

t^{Sale}	1	2	3	4	5	6
1	5	87	64	99	8	7

recall

#	Model	Year	Color	Sale
1	Chevy	1990	Red	5
2	Chevy	1990	Blue	87
3	Ford	1990	Green	64
4	Ford	1990	Blue	99
5	Ford	1991	Red	8
6	Ford	1991	Blue	7



Summary: **dimensions** *are* **matrices**, **measures** *are* **vectors**.

Measures provide for **integration** in Tolstoy's sense — *aka* **consolidation**

Totalisers

There is a unique function in type $A \to 1$, usually named $A \xrightarrow{!} 1$. This corresponds to a row vector wholly filled with 1s.

Example: $2 \xrightarrow{!} 1 = \begin{bmatrix} 1 & 1 \end{bmatrix}$

Given $M : B \to A$, the expression $! \cdot M$ (where $A \xrightarrow{!} 1$) is the row vector (of type $B \to 1$) that contains all column **totals** of M,

$$\begin{bmatrix} 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 50 & 40 & 85 & 115 \\ 50 & 10 & 85 & 75 \end{bmatrix} = \begin{bmatrix} 100 & 50 & 170 & 190 \end{bmatrix}$$

Given type A, define its **totalizer** matrix $A \xrightarrow{\tau_A} A + 1$ by

$$\begin{aligned} \tau_A &: A \to A + 1 \\ \tau_A &= \begin{bmatrix} id \\ \hline ! \end{bmatrix} \end{aligned} \tag{5}$$

Thus $\tau_A \cdot M$ yields a copy of M on top of the corresponding totals.

MAP i DOCTORAL PROGRAM IN COMPUTER SCIENCE

Cubes

HASLab HIGH-ASSURANCE SOFTWARE LABORATORY INESCTEC

Data **cubes** can be obtained from products of totalizers.

Recall the Kronecker (tensor) product $M \otimes N$ of two matrices $A \xrightarrow{M} B$ and $C \xrightarrow{N} D$, which is of type $A \times C \xrightarrow{M \otimes N} B \times D$.

The matrix

$$A \times B \xrightarrow{\tau_A \otimes \tau_B} (A+1) \times (B+1)$$

provides for totalization on the **two dimensions** A and B.

Indeed, type $(A+1) \times (B+1)$ is isomorphic to $A \times B + A + B + 1$, whose four parcels represent the four elements of the "**dimension powerset** of $\{A, B\}$".

Cube = muti-dimensional totalisation

Recalling

$$v = (t_{Year} \triangledown (t_{Color} \triangledown t_{Model})) \cdot (t^{Sale})^{\circ}$$

build

$$c = (\tau_{Year} \otimes (\tau_{Color} \otimes \tau_{Model})) \cdot v$$

This is the multidimensional vector (tensor) representing the **data cube** for

- **dimensions** *Year*, *Color*, *Model*
- **measure** *Sale*

depicted aside.

(Year+1) x ((Color+1) x (Model+1))			ALL
1990	Blue	Chevy	87
		Ford	99
		ALL	186
	Green	Chevy	0
		Ford	64
		ALL	64
	Red	Chevy	5
		Ford	0
		ALL	5
	ALL	Chevy	92
		Ford	163
		ALL	255
1991	Blue	Chevy	0
		Ford	7
		ALL	7
	Green	Chevy	0
		Ford	0
		ALL	0
	Red	Chevy	0
		Ford	8
		ALL	8
	ALL	Chevy	0
		Ford	15
		ALL	15
ALL	Blue	Chevy	87
		Ford	106
		ALL	193
	Green	Chevy	0
		Ford	64
		ALL	64
	Red	Chevy	5
		Ford	8
		ALL	13
	ALL	Chevy	92
		Ford	178
		ALL	270

MAPi DOCTORAL PROGRAM IN COMPUTER SCIENCE

Generalizing data cubes

HASLab INESCTEC

In our approach a **cube** is not necessarily one such column vector.

The key to generic data cubes is (generic) **vectorization**, a kind of "**matrix currying**": given $A \times B \xrightarrow{M} C$ with $A \times B$-many columns and C-many rows, reshape M into its **vectorized** version $B \xrightarrow{\mathbf{vec}_A\, M} A \times C$ with B-many columns and $A \times C$-many rows.

Matrices M and $\mathbf{vec}_A\, M$ are **isomorphic** — they contain the same information in different formats, cf.

$$c\ M\ (a, b) \quad = \quad (a, c)\ (\mathbf{vec}_A\, M)\ b \tag{6}$$

for every a, b, c.

MAPi DOCTORAL PROGRAM IN COMPUTER SCIENCE

Generalizing data cubes

HASLab INESCTEC

Vectorization thus has an inverse operation — **unvectorization**:

$$A \times B \to C \quad \underset{\mathbf{unvec}_A}{\overset{\mathbf{vec}_A}{\cong}} \quad B \to A \times C$$

That is, M can be retrieved back from $\mathbf{vec}_A\, M$ by unvectorizing it:

$$N = \mathbf{vec}_A\, M \quad \Leftrightarrow \quad \mathbf{unvec}_A\, N = M \tag{7}$$

Vectorization has a rich algebra, e.g. a **fusion**-law

$$(\mathbf{vec}\, M) \cdot N \quad = \quad \mathbf{vec}\, (M \cdot (id \otimes N)) \tag{8}$$

and an **absorption**-law:

$$\mathbf{vec}\, (M \cdot N) \quad = \quad (id \otimes M) \cdot \mathbf{vec}\, N \tag{9}$$

(De)vectorization

Devectorizing our starting tensor, across dimension *Year*:

$$\textit{Year} \times (\textit{Color} \times \textit{Model}) \xleftarrow{\quad} 1$$

$$\textbf{\textit{unvec}}_{\textit{Year}} \left(\begin{array}{lll|l} & & & \text{ALL} \\ \hline \multirow{6}{*}{1990} & \multirow{2}{*}{Blue} & Chevy & 87 \\ & & Ford & 99 \\ & \multirow{2}{*}{Green} & Chevy & 0 \\ & & Ford & 64 \\ & \multirow{2}{*}{Red} & Chevy & 5 \\ & & Ford & 0 \\ \hline \multirow{6}{*}{1991} & \multirow{2}{*}{Blue} & Chevy & 0 \\ & & Ford & 7 \\ & \multirow{2}{*}{Green} & Chevy & 0 \\ & & Ford & 0 \\ & \multirow{2}{*}{Red} & Chevy & 0 \\ & & Ford & 8 \end{array} \right) =$$

$$\textit{Color} \times \textit{Model} \xleftarrow{\quad} \textit{Year}$$

		1990	1991
Blue	Chevy	87	0
	Ford	99	7
Green	Chevy	0	0
	Ford	64	0
Red	Chevy	5	0
	Ford	0	8

There is room for further unvectorizing the outcome, this time across *Color* — next slide:

(De)vectorization

Further unvectorization:

$$\textbf{unvec}_{Color}\left(\begin{array}{|ll|cc|}\hline & & 1990 & 1991 \\ \hline Blue & Chevy & 87 & 0 \\ & Ford & 99 & 7 \\ \hline Green & Chevy & 0 & 0 \\ & Ford & 64 & 0 \\ \hline Red & Chevy & 5 & 0 \\ & Ford & 0 & 8 \\ \hline\end{array}\right)_{Color \times Model \longleftarrow Year} = \underset{Model \longleftarrow Color \times Year}{}$$

	Blue		Green		Red	
	1990	1991	1990	1991	1990	1991
Chevy	87	0	0	0	5	0
Ford	99	7	64	0	0	8

and so on.

MAPi DOCTORAL PROGRAM IN COMPUTER SCIENCE

Generic cubes

HASLab INESCTEC

It turns out **that** cubes can be calculated for any such two-dimensional versions of our original data tensor, for instance,

$$\textbf{cube}\ N : \ \mathit{Model} + 1 \longleftarrow (\mathit{Color} + 1) \times (\mathit{Year} + 1)$$

$$\textbf{cube}\ N = \tau_{\mathit{Model}} \cdot N \cdot (\tau_{\mathit{Color}} \otimes \tau_{\mathit{Year}})^{\circ}$$

where N stands for the second matrix of the previous slide, yielding

	Blue			*Green*			*Red*			ALL		
	1990	1991	ALL	1990	1991	ALL	1990	1991	ALL	1990	1991	ALL
Chevy	87	0	87	0	0	0	5	0	5	92	0	92
Ford	99	7	106	64	0	64	0	8	8	163	15	178
ALL	186	7	193	64	0	64	5	8	13	255	15	270

See how the 36 entries of the original cube have been rearranged in a 3*12 rectangular layout, as dictated by the **dimension** cardinalities.

The **cube** (LA) operator

Definition (Cube)

Let M be a matrix of type

$$\Pi_{j=1}^{n} B_j \xleftarrow{M} \Pi_{i=1}^{m} A_i \tag{10}$$

We define matrix **cube** M, the *cube of* M, as follows

$$\textbf{cube}\ M \quad = \quad (\bigotimes_{j=1}^{n} \tau_{B_j}) \cdot M \cdot (\bigotimes_{i=1}^{m} \tau_{A_i})^{\circ} \tag{11}$$

where $\bigotimes$ is finite Kronecker product.

So **cube** M has type $\Pi_{j=1}^{n}(B_j + 1) \longleftarrow \Pi_{i=1}^{m}(A_i + 1)$.

□

MAPi DOCTORAL PROGRAM IN COMPUTER SCIENCE

Properties of data cubing

HASLab INESCTEC

Linearity:

$$\textbf{cube}\ (M + N) \quad = \quad \textbf{cube}\ M + \textbf{cube}\ N \tag{12}$$

Proof: Immediate by bilinearity of matrix composition:

$$M \cdot (N + P) \quad = \quad M \cdot N + M \cdot P \tag{13}$$

$$(N + P) \cdot M \quad = \quad N \cdot M + P \cdot M \tag{14}$$

This can be taken advantage of not only in **incremental** data cube construction but also in **parallelizing** data cube generation.

Properties of data cubing

Updatability: by Khatri-Rao product linearity,

$$(M + N) \triangledown P = M \triangledown P + N \triangledown P$$
$$P \triangledown (M + N) = P \triangledown M + P \triangledown N$$

the **cube** operator commutes with the usual CRUDE operations, namely record **updating**. For instance, suppose record

#	Model	Year	Color	Sale
5	*Ford*	1991	*Red*	8

cf

t_{Model}	1	2	3	4	5	6
Chevy	1	1	0	0	0	0
Ford	0	0	1	1	1	1

is updated to

#	Model	Year	Color	Sale
5	*Chevy*	1991	*Red*	8

cf

t'_{Model}	1	2	3	4	5	6
Chevy	1	1	0	0	1	0
Ford	0	0	1	1	0	1

Properties of data cubing

One just has to compute the "delta" projection,

$$\delta_{Model} = t'_{Model} - t_{Model} =$$

	1	2	3	4	5	6
Chevy	0	0	0	0	1	0
Ford	0	0	0	0	-1	0

then the "delta cube",

$$d = (\tau_{Year} \otimes (\tau_{Color} \otimes \tau_{Model})) \cdot v'$$

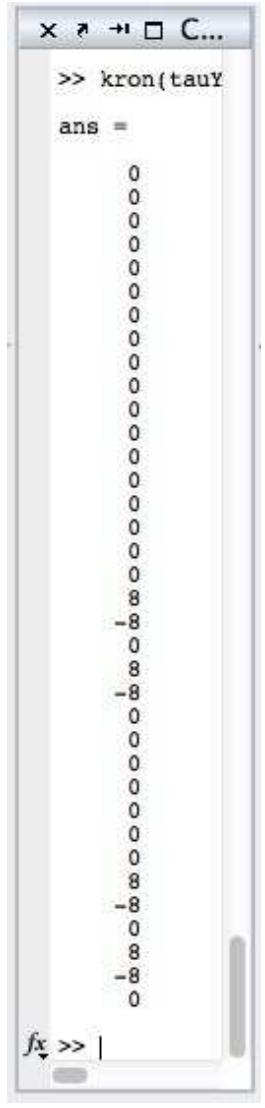
where

$$v' = (t_{Year} \triangledown (t_{Color} \triangledown \delta_{Model})) \cdot (t^{Sale})^{\circ}$$

and finally **add** the "delta cube" to the original cube:

$$c' = c + d.$$

— see the simulation aside (MATLab).



Properties of data cubing

Commutativity: cube commutes with **vectorization**:

Let $X \xleftarrow{M} Y \times C$ *and* $Y \times X \xleftarrow{\mathbf{vec}\, M} C$ *be its* Y*-vectorization. Then*

$$\mathbf{vec}\,(\mathbf{cube}\ M) = \mathbf{cube}\ (\mathbf{vec}\, M) \tag{15}$$

holds. $\square$

Type diagrams:

$$\begin{array}{ccc} Y \times X & \xleftarrow{\mathbf{vec}_Y\, M} & C \\ {\scriptstyle \tau_Y \otimes \tau_M}\downarrow & & \uparrow{\scriptstyle \tau_C^\circ} \\ (Y+1) \times (X+1) & \underset{\mathbf{vec}_{Y+1}\,(\mathbf{cube}\ M)}{\overset{\mathbf{cube}\,(\mathbf{vec}_Y\, M)}{\rightleftarrows}} & C+1 \end{array} \qquad \cong \qquad \begin{array}{ccc} X & \xleftarrow{M} & Y \times C \\ {\scriptstyle \tau_X}\downarrow & & \uparrow{\scriptstyle (\tau_Y \otimes \tau_C)^\circ} \\ X+1 & \xleftarrow{\mathbf{cube}\, M} & (Y+1) \times (C+1) \end{array}$$

Proof in (Oliveira and Macedo, 2017).

MAP i DOCTORAL PROGRAM IN COMPUTER SCIENCE

Properties of data cubing

HASLab INESCTEC

The following theorem shows that changing the dimensions of a data cube does not change its totals.

Theorem (Free theorem)

Let $B \xleftarrow{M} A$ *be cubed into* $B + 1 \xleftarrow{\textbf{cube } M} A + 1$ *, and* $r : C \to A$ *and* $s : D \to B$ *be arbitrary functions. Then*

$$\textbf{cube}\ (s^{\circ} \cdot M \cdot r) \quad = \quad (s^{\circ} \oplus id) \cdot (\textbf{cube}\ M) \cdot (r \oplus id) \qquad (16)$$

holds, where $M \oplus N = \begin{bmatrix} M & 0 \\ 0 & N \end{bmatrix}$ *is matrix* **direct sum**.

□

The proof given in the same paper resorts to the **free theorem** of **polymorphic** operators due to J. Reynolds and popularized by Wadler (1989) under the heading **Theorems for free!**.

MAP i DOCTORAL PROGRAM IN COMPUTER SCIENCE

Cube universality — slicing

HASLab INESCTEC

Slicing is a specialized filter for a particular value in a dimension.

Suppose that from our starting cube

$$c : 1 \to (\mathit{Year} + 1) \times ((\mathit{Color} + 1) \times (\mathit{Model} + 1))$$

one is only interested in the data concerning year 1991.

It suffices to regard data values as (categorial) **points**: given $p \in A$, constant function $\underline{p} : 1 \to A$ is said to be a *point* of A, for instance

$$\underline{1991} : 1 \to \mathit{Year} + 1$$

$$\underline{1991} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

MAPi DOCTORAL PROGRAM IN COMPUTER SCIENCE

Cube universality — slicing

HASLab HIGH-ASSURANCE SOFTWARE LABORATORY INESCTEC

Example:

$$1 \xrightarrow{c} (Year+1) \times ((Color+1) \times (Model+1)) \xrightarrow{\underline{1991}^{\circ} \otimes id} 1 \times ((Color+1) \times (Model+1)) = \begin{bmatrix} 0 \\ 7 \\ 7 \\ 0 \\ 0 \\ 0 \\ 0 \\ 8 \\ 8 \\ 0 \\ 15 \\ 15 \end{bmatrix}$$

Cube universality — rolling-up

Gray et al. (1997) say that *going up the levels [of aggregated data] is called rolling-up*.

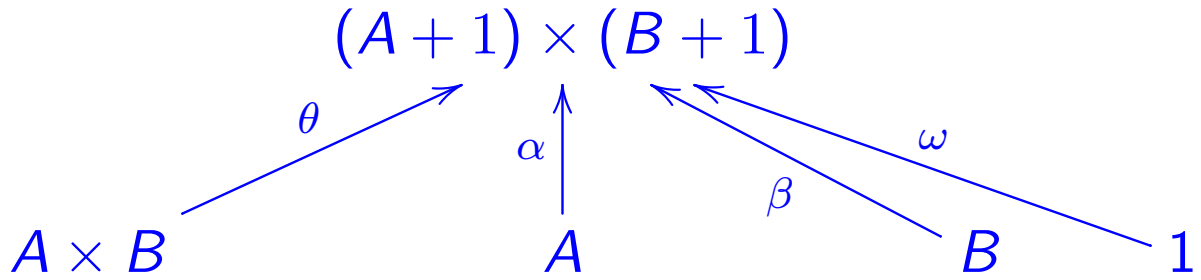
In this sense, a **roll-up** operation over dimensions A, B and C could be the following form of (increasing) summarization:

$$A \times (B \times C)$$
$$A \times B$$
$$A$$
$$1$$

How does this work over a data cube? We take the simpler case of two dimensions A, B as example.

Cube universality — rolling-up

The dimension powerset for A, B is captured by the corresponding matrix **injections** onto the cube target type $(A+1) \times (B+1)$:



where

$$\theta = i_1 \otimes i_1$$
$$\alpha = i_1 \triangledown i_2 \cdot !$$
$$\beta = i_1 \cdot ! \triangledown i_2$$
$$\omega = i_2 \triangledown i_2$$

NB: the injections i_1 and i_2 are such that $[i_1|i_2] = id$, where $[M|N]$ denotes the horizonal gluing of two matrices.

Cube universality — rolling-up

One can build compound injections, for instance

$$\rho : (A + 1) \times (B + 1) \leftarrow A \times B + (A + 1)$$
$$\rho = [\theta | [\alpha | \omega]]$$

Then, for $M : C \to A \times B$:

$$\rho^{\circ} \cdot (\textbf{cube}\ M) = \left[\frac{M}{\left[\frac{fst \cdot M}{! \cdot M} \right]} \right] \cdot \tau_C^{\circ}$$

extracts from **cube** M the corresponding **roll-up**.

The next slides give a concrete example.

MAPi DOCTORAL PROGRAM IN COMPUTER SCIENCE

Cube universality — rolling-up

HASLab HIGH-ASSURANCE SOFTWARE LABORATORY INESCTEC

Let M be the (generalized) data cube

		1990	1991	ALL
	Chevy	87	0	87
Blue	*Ford*	99	7	106
	ALL	186	7	193
	Chevy	0	0	0
Green	*Ford*	64	0	64
	ALL	64	0	64
	Chevy	5	0	5
Red	*Ford*	0	8	8
	ALL	5	8	13
	Chevy	92	0	92
ALL	*Ford*	163	15	178
	ALL	255	15	270

Cube universality — rolling-up

Building the injection matrix $\rho = [\theta|\,[\alpha|\omega]]$ for types $\mathit{Color} \times \mathit{Model} + \mathit{Color} + 1 \to (\mathit{Color} + 1) \times (\mathit{Model} + 1)$ we get the following matrix (already transposed):

		Blue			*Green*			*Red*			ALL		
		Chevy	*Ford*	ALL	*Chevy*	*Ford*	ALL	*Chevy*	*Ford*	ALL	*Chevy*	*Ford*	ALL
Blue	*Chevy*	1	0	0	0	0	0	0	0	0	0	0	0
	Ford	0	1	0	0	0	0	0	0	0	0	0	0
Green	*Chevy*	0	0	0	1	0	0	0	0	0	0	0	0
	Ford	0	0	0	0	1	0	0	0	0	0	0	0
Red	*Chevy*	0	0	0	0	0	0	1	0	0	0	0	0
	Ford	0	0	0	0	0	0	0	1	0	0	0	0
	Blue	0	0	1	0	0	0	0	0	0	0	0	0
	Green	0	0	0	0	0	1	0	0	0	0	0	0
	Red	0	0	0	0	0	0	0	0	1	0	0	0
	ALL	0	0	0	0	0	0	0	0	0	0	0	1

Cube universality — rolling-up

Then

$$\rho^{\circ} \cdot \textbf{cube}\ M =$$

		1990	1991	ALL
Blue	*Chevy*	87	0	87
	Ford	99	7	106
Green	*Chevy*	0	0	0
	Ford	64	0	64
Red	*Chevy*	5	0	5
	Ford	0	8	8
	Blue	186	7	193
	Green	64	0	64
	Red	5	8	13
	ALL	255	15	270

Note how a roll-up is a particular "subset" of a cube.

Matrix ρ° performs the (quantitative) selection of such a subset.

MAPi DOCTORAL PROGRAM IN COMPUTER SCIENCE

Summary

HASLab INESCTEC

There is a **lot** of SCIENCE in CS that CS people simply **ignore**.

Abadir and Magnus (2005) stress on the need for a **standardized** notation for **linear algebra** in the field of econometrics and **statistics**.

This talk suggests such a notation should be **properly typed**.

Since (Macedo and Oliveira, 2013) the author has invested in **typing** linear algebra in a way that makes it closer to modern **typed** languages.

This extends previous efforts on applying LA to **OLAP** (Macedo and Oliveira, 2015)

Opportunities

A lot to be done!

MAPi is the right **environment** for this kind of cross-breeding:

- 3 **universities**
- many **labs**
- different **cultures**, nice cooperation...

MAPi DOCTORAL PROGRAM IN COMPUTER SCIENCE

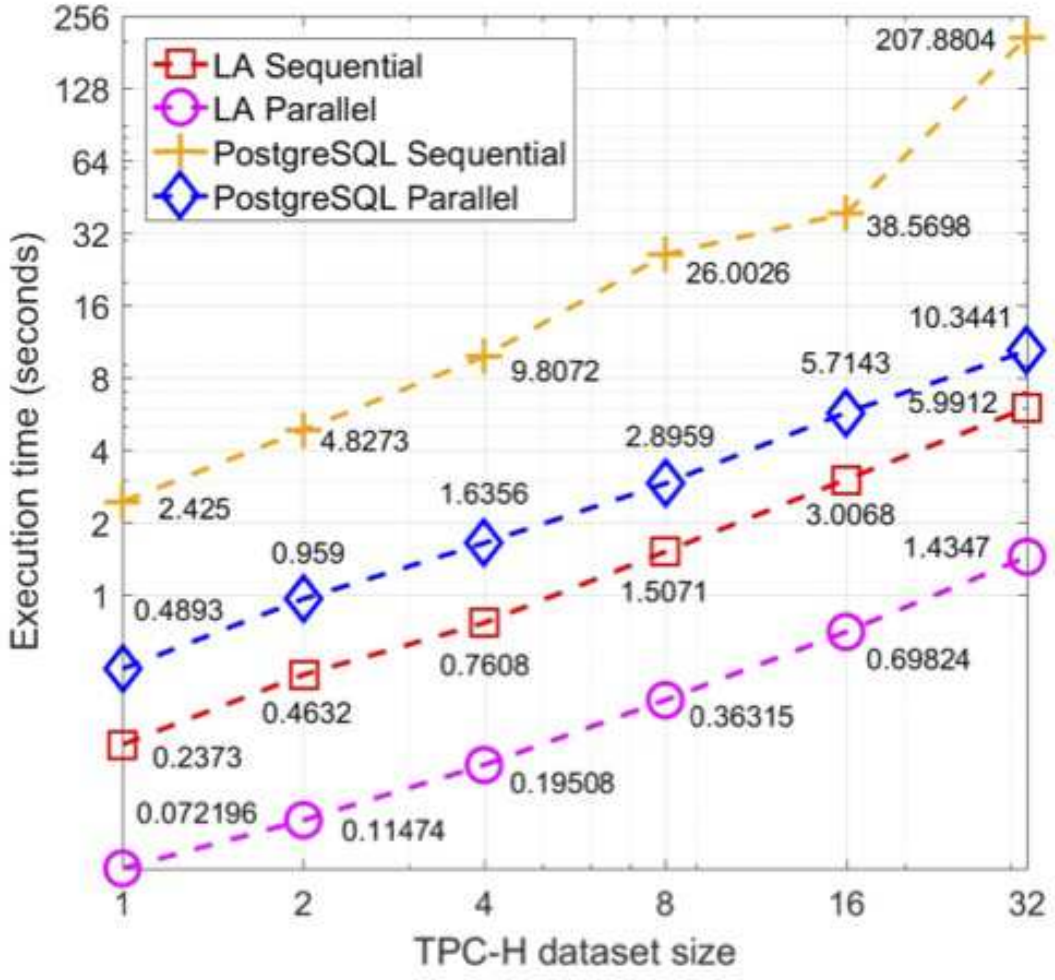
Annex

HASLab INESCTEC

(for those who care mostly about efficiency)

Theory pays off!

Plot taken from a recent MSc report on TPC-H benchmarking LA approach:



References

K.M. Abadir and J.R. Magnus. *Matrix algebra. Econometric exercises 1*. C.U.P., 2005.

Jim Gray, Surajit Chaudhuri, Adam Bosworth, Andrew Layman, Don Reichart, Murali Venkatrao, Frank Pellow, and Hamid Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. *J. Data Mining and Knowledge Discovery*, 1(1):29–53, 1997. URL `citeseer.nj.nec.com/article/gray95data.html`.

H.D. Macedo and J.N. Oliveira. Typing linear algebra: A biproduct-oriented approach. *SCP*, 78(11):2160–2191, 2013.

H.D. Macedo and J.N. Oliveira. A linear algebra approach to OLAP. *FAoC*, 27(2):283–307, 2015.

J. N. Oliveira and H. D. Macedo. The data cube as a typed linear algebra operator. In *Proc. of the 16th Int. Symposium on Database Programming Languages*, DBPL '17, pages 6:1–6:11, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-5354-0. doi: 10.1145/3122831.3122834.

P.L. Wadler. Theorems for free! In *4th International Symposium*

on Functional Programming Languages and Computer Architecture, pages 347–359, London, Sep. 1989. ACM.