Programming Languages for Pre-Mechanical Calculating Tools

Baptiste Mélès 梅乐思

Archives Henri Poincaré – Université de Lorraine (Nancy 南锡, France)

"Cultures of Mathematics and Logic",
Guangzhou 广州, China, November 2012

"Man is an animal who thinks with his fingers"

"As for the algebra that Chinese developped by applying it first to the resolution of elementary problems of geometry, it can not be dissociated from an ability to manipulations which is unequally cultivated in the different civilizations. As said Marcel Mauss, quoting Maurice Halbwachs: "**Man is an animal who thinks with his fingers**" (« L'homme est un animal qui pense avec ses doigts »), a beautiful maxim which warns us against too intellectualist approaches, and which Chinese mathematicians would probably have approved. The history of a civilization implies at once socio-political structures, notions, mental attitudes, behaviours, gestures, and everything that the ethnologists name techniques of the body (« techniques du corps »)."
— Jacques Gernet, *L'Intelligence de la Chine*, "Histoire sociale et intellectuelle", p. 254.

Question

- Initial question: What functions can be computed with different kinds of non-mechanical calculating tools?
- Examples of calculating tools: abacus, Chinese counting-rods, logarithm tables, slide rule, counting on paper...
- Examples of limitations:
  - logarithm tables, slide rule: no addition;
  - abacus: no resolution of systems of n linear equations? no logarithm?

Question

- Initial question: What functions can be computed with different kinds of non-mechanical calculating tools?
- Examples of calculating tools: abacus, Chinese counting-rods, logarithm tables, slide rule, counting on paper...
- Examples of limitations:
  - logarithm tables, slide rule: no addition;
  - abacus: no resolution of systems of n linear equations? no logarithm?

Question

- Initial question: What functions can be computed with different kinds of non-mechanical calculating tools?
- Examples of calculating tools: abacus, Chinese counting-rods, logarithm tables, slide rule, counting on paper...
- Examples of limitations:
    - logarithm tables, slide rule: no addition;
    - abacus: no resolution of systems of n linear equations? no logarithm?

Classical answers

- Two possible answers:
  - *A posteriori* (history of mathematics): "just read your classics".
    - Counting-rods: Sunzi 孫子 (ca. 300), *Sunzi Suanjing* 孫子算經;
    - Abacus: Cheng Dawei 程大位 (1533–1606), *Suanfa Tongzong* 算法統宗 (1592).
  - *A priori* (computer science): "just make a machine".
    - Alan Turing (1912–1954), "On Computable Numbers" (1936): the definition of "Turing machines" begins with the behaviour of a human "computer" (i.e. the calculating *man*);
    - Joachim Lambek (born 1922), "How to Program an Infinite Abacus" (1961).

Classical answers

- Two possible answers:
  - *A posteriori* (history of mathematics): "just read your classics".
    - Counting-rods: Sunzi 孫子 (ca. 300), *Sunzi Suanjing* 孫子算經;
    - Abacus: Cheng Dawei 程大位 (1533–1606), *Suanfa Tongzong* 算法統宗 (1592).
  - *A priori* (computer science): "just make a machine".
    - Alan Turing (1912–1954), "On Computable Numbers" (1936): the definition of "Turing machines" begins with the behaviour of a human "computer" (i.e. the calculating *man*);
    - Joachim Lambek (born 1922), "How to Program an Infinite Abacus" (1961).

Classical answers

- Two possible answers:
  - *A posteriori* (history of mathematics): "just read your classics".
    - Counting-rods: Sunzi 孫子 (ca. 300), *Sunzi Suanjing* 孫子算經;
    - Abacus: Cheng Dawei 程大位 (1533–1606), *Suanfa Tongzong* 算法統宗 (1592).
  - *A priori* (computer science): "just make a machine".
    - Alan Turing (1912–1954), "On Computable Numbers" (1936): the definition of "Turing machines" begins with the behaviour of a human "computer" (i.e. the calculating *man*);
    - Joachim Lambek (born 1922), "How to Program an Infinite Abacus" (1961).

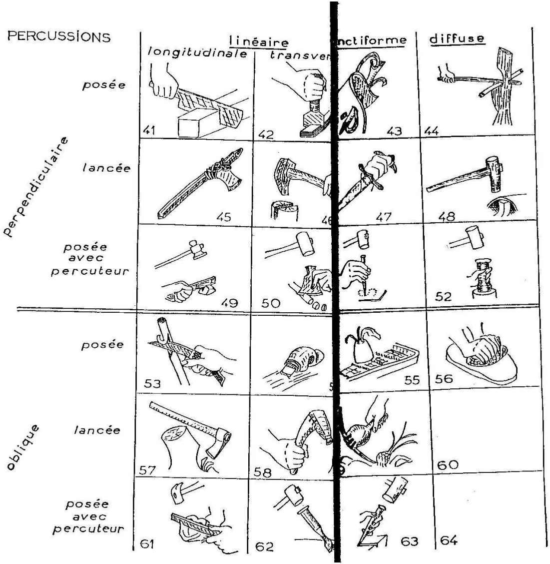
Limits

- *A posteriori* (history of mathematics): "just read your classics".
  - But the classics only describe what has been done, not necessarily what can be done.
- *A priori* (computer science): "just make a machine".
  - But we lose the anthropological side of the problem. Idealized computers are not computing *men*. (And infinite abaci do not exist.)

Structural anthropology and technology

- To answer our question, we would need to mix those two approches — anthropology and *a priori* science. But is it possible?
- Yes, it seems to be! There is at least one example. The idea would be to have something like that:

André Leroi-Gourhan (1911–1986), *L'Homme et la matière* (1943):



Percussions:

linear / dot-shaped / diffuse

perpendicular / oblique

put down / throwed / put down with a percussion tool

Formal and anthropological properties of calculating tools

- Our hypothesis will be than one could do with non-mechanical calculating tools what Leroi-Gourhan did for percussion tools.

$$\frac{\text{simple machines}}{\text{percussion tools}} = \frac{\textit{programming languages}}{\text{calculating tools}}.$$

- This would be a classification, not of calculating tools themselves (see Peter Moor, "Three Myths of Computer Science", 1978), but of their uses in algorithms.
- In order to show it, let us examine and compare the formal and anthropological properties of two Chinese computing tools: the abacus and the counting-rods.

Formal and anthropological properties of calculating tools

- Our hypothesis will be than one could do with non-mechanical calculating tools what Leroi-Gourhan did for percussion tools.

$$\frac{\text{simple machines}}{\text{percussion tools}} = \frac{\textit{programming languages}}{\text{calculating tools}}.$$

- This would be a classification, not of calculating tools themselves (see Peter Moor, "Three Myths of Computer Science", 1978), but of their uses in algorithms.
- In order to show it, let us examine and compare the formal and anthropological properties of two Chinese computing tools: the abacus and the counting-rods.

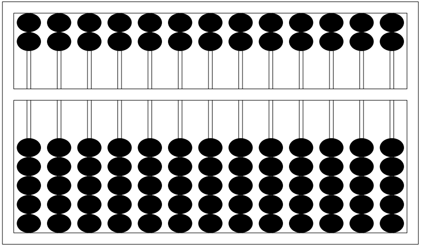
Formal and anthropological properties of calculating tools

- Our hypothesis will be than one could do with non-mechanical calculating tools what Leroi-Gourhan did for percussion tools.

$$\frac{\text{simple machines}}{\text{percussion tools}} = \frac{\textit{programming languages}}{\text{calculating tools}}.$$

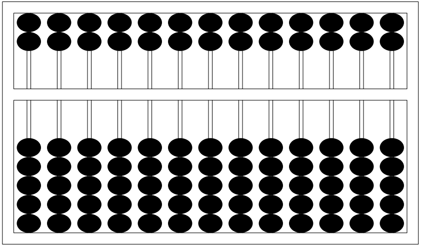
- This would be a classification, not of calculating tools themselves (see Peter Moor, "Three Myths of Computer Science", 1978), but of their uses in algorithms.
- In order to show it, let us examine and compare the formal and anthropological properties of two Chinese computing tools: the abacus and the counting-rods.

Abacus



Data structures

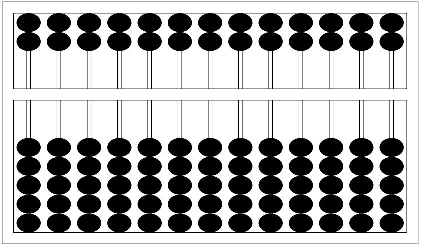
First question: what are the data structures of the abacus?



Data structures

There are three main kinds of abacus:

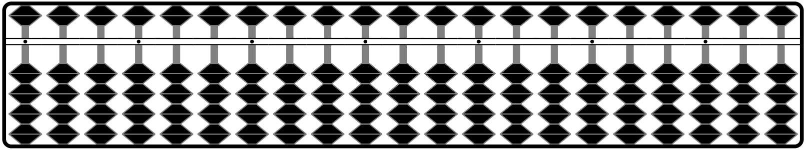
1. Chinese *suanpan* 算盤:



Data structures

There are three main kinds of abacus:

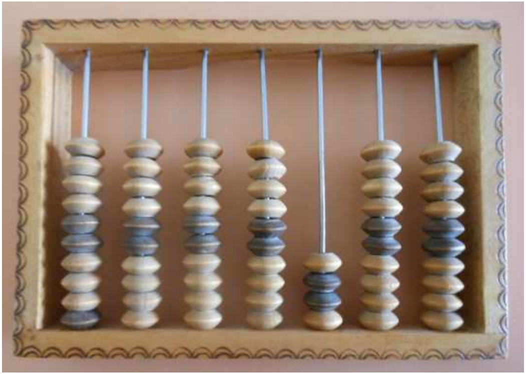
2. Japanese *soroban* 算盤:



Data structures

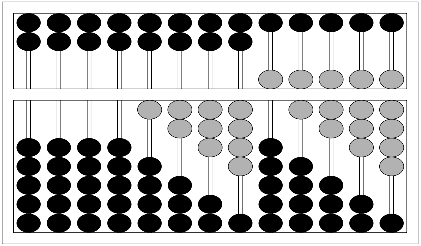
There are three main kinds of abacus:

3. Russian *schoty* (счёты):



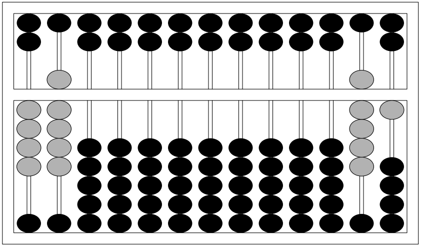
Data structures

Examples with the *suanpan*.
We can encode **one** number, for example 123 456 789, on the *suan-pan*:



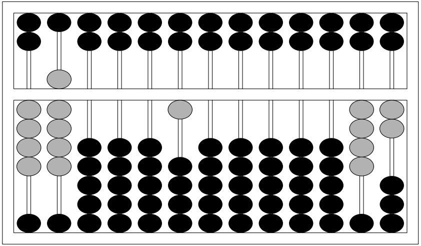
Data structures

Sometimes, it can be useful to write down **two** numbers, typically for the greater common divisor (for instance of 49 and 91):



Data structures

And sometimes even **three** numbers, as for "euclidian" division (divisor, quotient, dividend or rest), as for $91 = 49 \times 1 + 42$:



Data structures

Properties of the abacus

1. **Data structure**: one register containing a list of a few numbers.

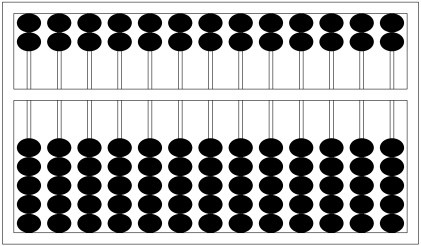
Programming style

Second question: what is the programming style of the abacus?

Programming style

Let us calculate $12 + 81$ with an *suanpan*.

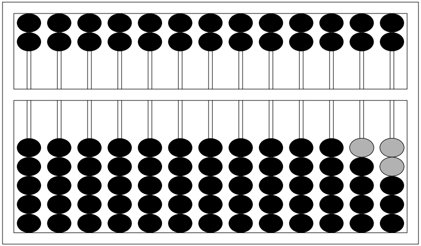
1. Reset the abacus:



Programming style

Let us calculate $12 + 81$ with an *suanpan*.

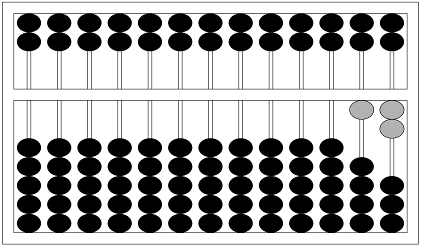
2. Encode the first number:



Programming style

Let us calculate $12 + 81$ with an *suanpan*.

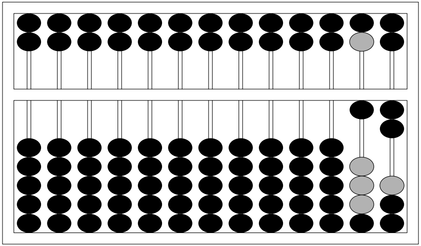
2. Encode the first number:



Programming style

Let us calculate $12 + 81$ with an *suanpan*.

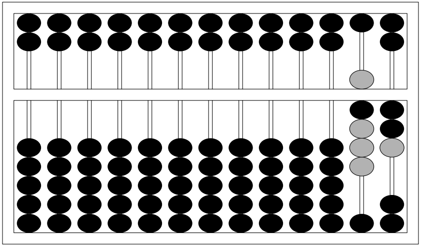
3. Add the second number:



Programming style

Let us calculate $12 + 81$ with an *suanpan*.

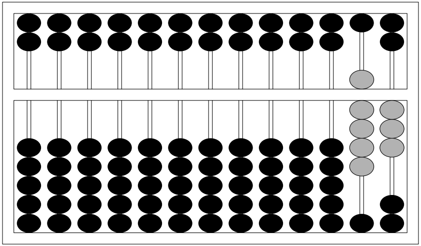
3. Add the second number:



Programming style

Let us calculate $12 + 81$ with an *suanpan*.

4. Read the result:



Theorem

$12 + 81 = 93.$

"Curryfication"

- What about the programming style?
- Operations are curryfied.

Definition

Curryfication (Schönfinkel, Curry) = transformation of an n-ary function into a composition of unary functions (provided that their respective "values" can themselves be functions).

"Curryfication"

- What about the programming style?
- Operations are curryfied.

Definition

Curryfication (Schönfinkel, Curry) = transformation of an *n*-ary function into a composition of unary functions (provided that their respective "values" can themselves be functions).

"Curryfication"

Definition

Curryfication (Schönfinkel, Curry) = transformation of an *n*-ary function into a composition of unary functions (provided that their respective "values" can themselves be functions).

Example

Without curryfication	After curryfication
$\begin{tikzcd} 12 \arrow[dr] & & \\ & (12, 81) \arrow[r, "+"] & 93 \\ 81 \arrow[ur] & & \end{tikzcd}$	$\begin{tikzcd} & 81 \arrow[d, "+'"] & \\ 12 \arrow[rr, "+81"'] & {} & 93 \end{tikzcd}$

"Curryfication"

What is the difference?

- The asymmetrical roles of the operands: 12 is passive (*object*), whereas 81 is active (belongs to an *act*).
- Then the result does not occupy a distinct place.
- Then there is **no variable assignment**!

"Curryfication"

What is the difference?

- The asymmetrical roles of the operands: 12 is passive (*object*), whereas 81 is active (belongs to an *act*).
- Then the result does not occupy a distinct place.
- Then there is **no variable assignment**!

"Curryfication"

What is the difference?

- The asymmetrical roles of the operands: 12 is passive (*object*), whereas 81 is active (belongs to an *act*).
- Then the result does not occupy a distinct place.
- Then there is **no variable assignment**!

Functional programming style

- The programming style of the abacus is *purely functional*.

Definition

Functional programming (John Backus, "Can programming be liberated from the von Neumann style?: a functional style and its algebra of programs", 1978):

- no variable assignment;
- algorithm = composition (vs. sequence) of functions.

Examples

Functional programming languages: pure Lisp, Haskell...

Parallelism

The abacus can even support parallelism! But functional parallelism.

Example

The greater common divisor:

- both numbers alternatively act on each other.
- but the programming style remains functional.

Properties of the abacus

Properties of the abacus

1. **Data structure**: one register.
2. **Programming style**: functional.

- What does it change?
- Addition is not *so* commutative...
  1. *A priori:* comparison between $14 + 7$ and $7 + 14$;
  2. *A posteriori:* Fibonacci's and Cheng Dawei's "addition" tables.

- What does it change?
- Addition is not *so* commutative...
  1. *A priori*: comparison between $14 + 7$ and $7 + 14$;
  2. *A posteriori*: Fibonacci's and Cheng Dawei's "addition" tables.

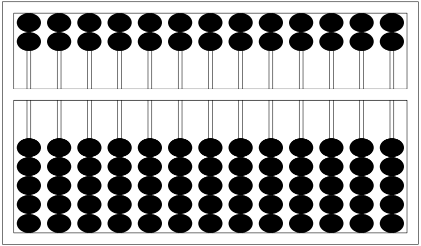
- What does it change?
- Addition is not *so* commutative...
  1. *A priori*: comparison between $14 + 7$ and $7 + 14$;
  2. *A posteriori*: Fibonacci's and Cheng Dawei's "addition" tables.

Is addition commutative?

Let us compare $14 + 7$ and $7 + 14$.

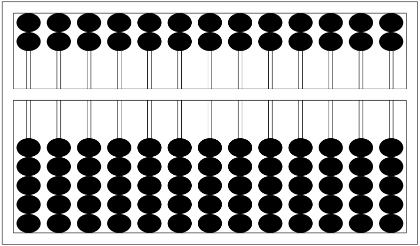
$14 + 7$:

Reset the abacus:



$7 + 14$:

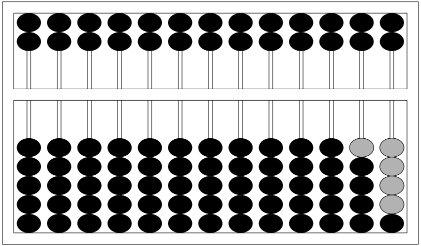
Reset the abacus:



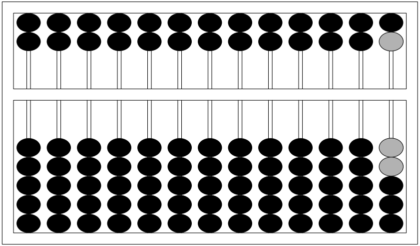
Is addition commutative?

Let us compare $14 + 7$ and $7 + 14$.

$14 + 7$:
Encode the first number:



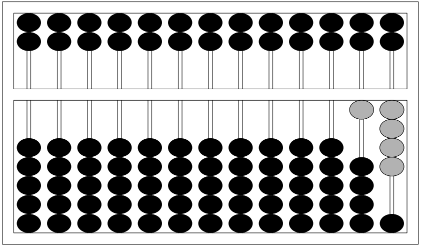
$7 + 14$:
Encode the first number:



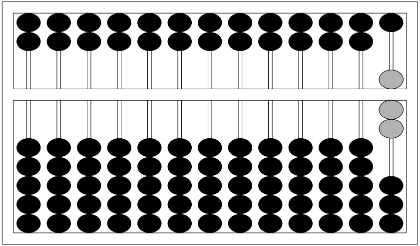
Is addition commutative?

Let us compare $14 + 7$ and $7 + 14$.

$14 + 7$:
Encode the first number:



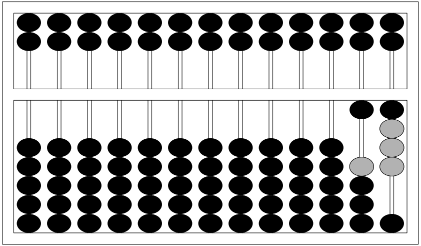
$7 + 14$:
Encode the first number:



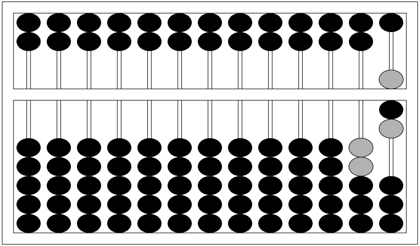
Is addition commutative?

Let us compare $14 + 7$ and $7 + 14$.

$14 + 7$:
Add the second number:



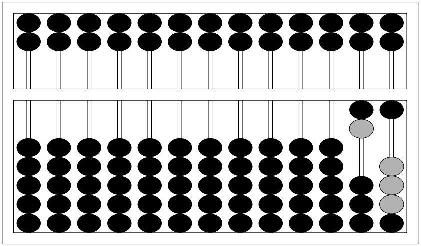
$7 + 14$:
Add the second number:



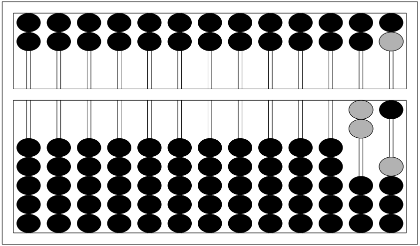
Is addition commutative?

Let us compare $14 + 7$ and $7 + 14$.

$14 + 7$:
Add the second number:



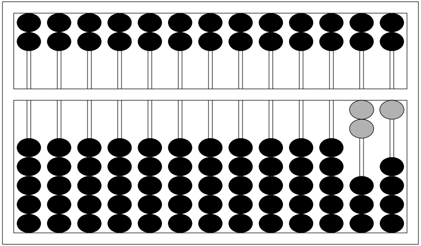
$7 + 14$:
Add the second number:



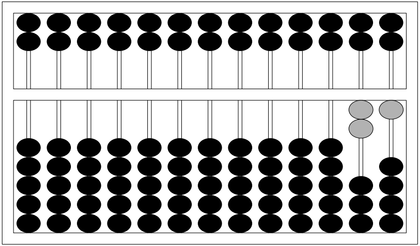
Is addition commutative?

Let us compare $14 + 7$ and $7 + 14$.

$14 + 7$:
Add the second number:



$7 + 14$:
Add the second number:



Theorem

$14 + 7 = 21$ *and* $7 + 14 = 21$ *(but not for the same reasons).*

Is addition commutative?

Addition on an abacus is:

- *denotationally* commutative (the result is the same),
- but *operationally* not commutative (the operations are different).

$$\begin{tikzcd} & 0 \arrow[dl, "+14"'] \arrow[dr, "+7"] & \\ 14 \arrow[dr, "+7"'] & & 7 \arrow[dl, "+14"] \\ & 21 & \end{tikzcd}$$

operational semantics

denotational semantics

Historical evidence

This surprising fact becomes obvious when one compares Fibonacci's and Cheng Dawei's "addition tables":

Fibonacci
Commutative addition
(triangular addition table)

+	1	2	3	⋯
1	2			
2	3	4		
3	4	**5**	6	
⋮	⋮	⋮	⋮	⋱

denotational table

Cheng Dawei
"Not so commutative" addition
(square addition table)

+	一	二	三	⋯
一	上一	上二	上三	⋯
二	上一	上二	下五除二	⋯
三	上一	下五除三	下五除二	⋯
⋮	⋮	⋮	⋮	⋱

operational table

Fibonacci, *Liber abaci*: triangular addition table

2	*and*	2	*make*	4
2		3		5
2		4		6
2		5		7
2		6		8
2		7		9
2		8		10
2		9		11
2		10		12

	Key	*for*	*Three*	
3	*and*	3	*make*	6
3		4		7
3		5		8
3		6		9
3		7		10
3		8		11
3		9		12
3		10		13

	Key	*for*	*Four*	
4	*and*	4	*make*	8
4		5		9
4		6		10
4		7		11
4		8		12
4		9		13
4		10		14

	Key	*for*	*Five*	
5	*and*	5	*make*	10
5		6		11
5		7		12
5		8		13
5		9		14
5		10		15

	Key	*for*	*Six*	
6	*and*	6	*make*	12
6		7		13
6		8		14
6		9		15
6		10		16

	Key	*for*	*Seven*	
7	*and*	7	*make*	14
7		8		15
7		9		16
7		10		17

	Key	*for*	*Eight*	
8	*and*	8	*make*	16
8		9		17
8		10		18

	Key	*for*	*Nine*	
9	*and*	9	*make*	18
9		10		19

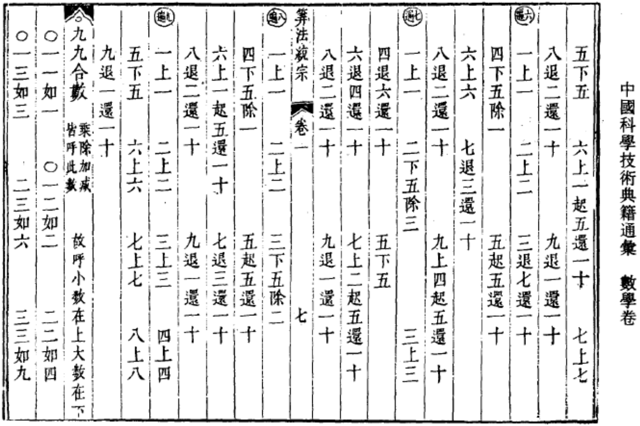
	Key	*for*	*Ten*	
10	*and*	10	*make*	20
20	*and*	20	*make*	40

Cheng Dawei, *Suanfa Tongzong*: square "addition table" (1)

當以初行爲右次行爲左以理而推之法當從右言
在左此乃不易之位也
九九八十一 便蒙通用
遍一 一上一 二上二 三上三 四上四
五下五 六上六 七上七 八上八
九上九
遍二 一上一 二上二 三下五除二
四下五除一 五起五還一十
六上一起五還一十 七上二起五還一十
八退二還一十 九退一還一十
算法統宗 卷一 六
遍三 一上一 二下五除三 三上三
四退六還一十 五下五 六上六
七退三還一十 八退二還一十
九退一還一十
遍四 一上一 二上二 三退七還一十
四下五除一 五起五還一十
六退四還一十 七退三還一十
八上三起五還一十 九退一還一十
遍五 一下五除四 二退八還一十
三下五除二 四退六還一十

二 | 一二三一

Cheng Dawei, *Suanfa Tongzong*: square "addition table" (2)



中國科學技術典籍通彙 數學卷

五下五 六上一起五還一十 七上七
八退二還一十 九退一還一十
遇六 一上一 二上二 三退七還一十
四下五除一 五起五還一十
六上六 七退三還一十
八退二還一十 九上四起五還一十
遇七 一上一 二下五除三 三上三
四退六還一十 五下五
六退四還一十 七上二起五還一十
八退二還一十 九退一還一十

算法統宗 卷一 七

遇八 一上一 二上二 三下五除二
四下五除一 五起五還一十
六上一起五還一十 七退三還一十
八退二還一十 九退一還一十
遇九 一上一 二上二 三上三 四上四
五下五 六上六 七上七 八上八
九退一還一十

九九合數 乘除加減皆呼此數 故呼小數在上大數在下
〇一一如一 〇一二如二 二二如四
〇一三如三 二三如六 三三如九

Pedagogical strategies

- The anthropological aspect of the problem can not be avoided: the user has to *learn tables*.
- This pedagogical aspect is cultural, not technical.

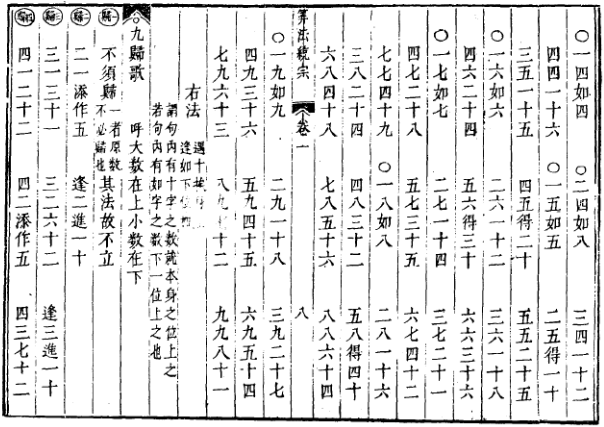
Pedagogical strategies

- There are pedagogical strategies:
  - **tables to learn**: multiplication table instead of long additions (Takashi Kojima, *Advanced Abacus: Theory and Practice*);
  - **tables to forget!**: Cheng Dawei's table of divisions until the 1930s (Takashi Kojima, *The Japanese Abacus: its Use and Theory*).

Pedagogical strategies

- There are pedagogical strategies:
  - **tables to learn**: multiplication table instead of long additions (Takashi Kojima, *Advanced Abacus: Theory and Practice*);
  - **tables to forget!**: Cheng Dawei's table of divisions until the 1930s (Takashi Kojima, *The Japanese Abacus: its Use and Theory*).

Cheng Dawei, *Suanfa Tongzong*: division table (1)


〇一四如四 〇二四如八 三四一十二
四四一十六 〇一五如五 二五得一十
三五一十五 四五得二十 五五二十五
〇一六如六 二六一十二 三六一十八
四六二十四 五六得三十 六六三十六
〇一七如七 二七一十四 三七二十一
四七二十八 五七三十五 六七四十二
七七四十九 〇一八如八 二八一十六
三八二十四 四八三十二 五八得四十
六八四十八 七八五十六 八八六十四
算法統宗 卷一 八
〇一九如九 二九一十八 三九二十七
四九三十六 五九四十五 六九五十四
七九六十三 八九七十二 九九八十一
右法 遇十[illegible] 逢如下[illegible]
謂句內有十字之數就本身之位上之
若句內有如字之數下一位上之也
九歸歌 呼大數在上小數在下
歸一 不須歸 一者原數不必歸也 其法故不立
歸二 二一添作五 逢二進一十
歸三 三一三十一 三二六十二 逢三進一十
歸四 四一二十二 四二添作五 四三七十二

Cheng Dawei, *Suanfa Tongzong*: division table (2)

逢四進一十

五歸 五一倍作二　五二倍作四　五三倍作六

五四倍作八　逢五進一十

六歸 六一下加四　六二三十二　六三添作五

六四六十四　六五八十二　逢六進一十

七歸 七一下加三　七二下加六　七三四十二

七四五十五　七五七十一　七六八十四

逢七進一十

八歸 八一下加二　八二下加四　八三下加六

八四添作五　八五六十二　八六七十四

八七八十六　逢八進一十

九歸 九歸隨身下　逢九進一十

○右法與九九合數相混但記句法惟辨多數在先少數在次即九歸之句如八六七十四是歸六八四十八是因之類已上句法併後各樣歌訣皆學者所當熟記

因乘法者單位曰因位數多曰乘通而言之乘也置所有物爲實以所求價爲法皆從末位而起如法乘之呼九字相生之數次第乘之呼如須次位言十在本身陞積謂之乘其數雖陞而位反降矣必須用定位之法而治之詳見于後

Conclusion on the abacus

Properties of the abacus

1. **data structure**: one register;
2. **programming style**: functional;
3. a given set of **tables** (addition, multiplication, substraction, division...), with **pedagogical** strategies (learning or forgetting).

- Those properties mix formal and anthropological data.

Conclusion on the abacus

Properties of the abacus

1. **data structure**: one register;
2. **programming style**: functional;
3. a given set of **tables** (addition, multiplication, substraction, division...), with **pedagogical** strategies (learning or forgetting).

- Those properties mix formal and anthropological data.

Conclusion on the abacus

Properties of the abacus

1. **data structure**: one register;
2. **programming style**: functional;
3. a given set of **tables** (addition, multiplication, substraction, division...), with **pedagogical** strategies (learning or forgetting).

- Those properties mix formal and anthropological data.

Conclusion on the abacus

Properties of the abacus

1. **data structure**: one register;
2. **programming style**: functional;
3. a given set of **tables** (addition, multiplication, substraction, division...), with **pedagogical** strategies (learning or forgetting).

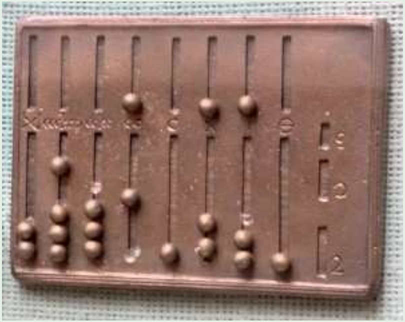
- Those properties mix formal and anthropological data.

Generalization of the method

Let us use those properties, not to describe, but to compare.

Examples

Comparisons between **different cultures**: the roman abacus is isomorphic with the Chinese *suanpan*:



Are the counting-rods isomorphic with the abacus?

Within the **"same" culture**: are the Ancient Chinese counting-rods isomorphic with the modern Chinese abacus?

Fig. 66. 'Discussions on difficult problems between Master and Pupil', a view of a counting-board (frontispiece of the *Suan Fa Thung Tsung*, +1593).

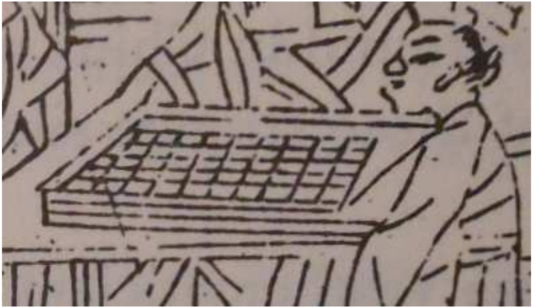
Are the counting-rods isomorphic with the abacus?

For a full description:

- Karine Chemla and Guo Shuchun, *Les Neuf Chapitres* 九章算術, 2004.
- Karine Chemla, "Le jeu d'opérations opposées mais complémentaires dans les textes mathématiques chinois anciens", 1996.
- Karine Chemla and Michael Lackner, "Des pratiques de la position en Chine", 1996.
- Karine Chemla, "Positions et changements en mathématiques", 1996.
- Karine Chemla, "Should they read FORTRAN as if it were English?", 1987.

Data structure

First question: what are the data structures of the counting-board?



Data structure

- Vertical disposition of numbers. *Sunzi Suanjing* (I.16) for the multiplication:

		8	1
6	5	6	1
	8	1	

- Horizontal disposition of numbers. *Nine Chapters* (VIII.1) for a system of linear equations:

	1			2			3
	2			3			2
	3			1			1
2	6		3	4		3	9

- Positions are moved (see Karine Chemla).

Data structure

- Vertical disposition of numbers. *Sunzi Suanjing* (I.16) for the multiplication:

		8	1
6	5	6	1
	8	1	

- Horizontal disposition of numbers. *Nine Chapters* (VIII.1) for a system of linear equations:

	1			2			3
	2			3			2
	3			1			1
2	6		3	4		3	9

- Positions are moved (see Karine Chemla).

Data structure

- Vertical disposition of numbers. *Sunzi Suanjing* (I.16) for the multiplication:

		8	1
6	5	6	1
	8	1	

- Horizontal disposition of numbers. *Nine Chapters* (VIII.1) for a system of linear equations:

	1			2			3
	2			3			2
	3			1			1
2	6		3	4		3	9

- Positions are moved (see Karine Chemla).

Formal properties

Formal properties of the counting-rods

1. **Data structure**: a two-dimensional array of numbers.

Data structures

Comparative consequences of the data structures:

- every algorithm working for the abacus works for the counting-rods,
- but the converse is not true.

Example

Algorithms for solving systems of linear equations:

- A simple algorithm for 2 linear equations works for the abacus as well as for the counting-rods;
- a general algorithm for *n* linear equations (like in the *Nine Chapters*, VIII: *fangcheng* 方程) works for the counting-rods, but not for the abacus.

Data structures

Comparative consequences of the data structures:

- every algorithm working for the abacus works for the counting-rods,
- but the converse is not true.

Example

Algorithms for solving systems of linear equations:

- A simple algorithm for 2 linear equations works for the abacus as well as for the counting-rods;
- a general algorithm for n linear equations (like in the *Nine Chapters*, VIII: *fangcheng* 方程) works for the counting-rods, but not for the abacus.

Programming style

Second question: what is the programming style of the counting-rods?

Programming style

- Is it functional?
- It can be: all algorithms working with the abacus work with the counting-rods.

Programming style

- Is it functional?
- It can be: all algorithms working with the abacus work with the counting-rods.

Programming style

But the programming style *must* not be functional.
(1) The result of an operation does not necessarily rise from the data themselves.

Example: Sunzi's multiplication algorithm

The result (output) is put in a new place, distinct from the input data's places:

		8	1
6	**5**	**6**	**1**
	8	1	

Programming style

(2) Sometimes, values are put “aside” (副 *fu*, «en auxiliaire»).

Example: the *Nine Chapters*' square root algorithm

(Karine Chemla and Guo Shuchun, *Les Neuf Chapitres*, p. 325)

		2			quotient
1	5	2	2	5	dividend
	4				divisor
		1			auxiliary (副 *fu*)

Consequence: there *is* variable assignment. (See Karine Chemla, *Les Neuf Chapitres*, p. 25.)

Programming style

The programming is imperative.

Definition

Imperative programming:

- variable assignment;
- algorithm = sequence (vs. composition) of instructions.

Examples

Imperative programming languages: MIX machine (Donald Knuth, *The Art of computer programming*), assembly, C...

Formal properties

Formal properties of the counting-rods

1. **Data structure**: a two-dimensional array of numbers.
2. **Programming style**: imperative.

Cultural aspects

- Do such structural facts have cultural echoes?
- One can think so:
  - Jacques Gernet: Chinese mathematics are correlated with calculating tools;
  - Jean-Claude Martzloff, *A History of Chinese Mathematics*, p. 20: "The abacus, a calculating instrument used by merchants, slowly supplanted the counting-rods used in the erudite practice of computation, and mathematics was restricted to commercial arithmetic."

Cultural aspects

- Do such structural facts have cultural echoes?
- One can think so:
  - Jacques Gernet: Chinese mathematics are correlated with calculating tools;
  - Jean-Claude Martzloff, *A History of Chinese Mathematics*, p. 20: "The abacus, a calculating instrument used by merchants, slowly supplanted the counting-rods used in the erudite practice of computation, and mathematics was restricted to commercial arithmetic."

Conclusions

1. Cultures and calculating tools:
   - internally, calculating tools have a cultural kernel (pedagogical strategies);
   - externally, they might be correlated with cultural facts (kinds of mathematics, social function...).
2. A classification of calculating tools should at least take into account:
   - the data structures;
   - the programming style (functional or imperative);
   - the set of memorized tables.
3. The very idea of a classification suggests that various cultures explore various areas of an unique and eternal frame of human computing (like Leroi-Gourhan's table).

Conclusions

1. Cultures and calculating tools:
    - internally, calculating tools have a cultural kernel (pedagogical strategies);
    - externally, they might be correlated with cultural facts (kinds of mathematics, social function...).
2. A classification of calculating tools should at least take into account:
    - the data structures;
    - the programming style (functional or imperative);
    - the set of memorized tables.
3. The very idea of a classification suggests that various cultures explore various areas of an unique and eternal frame of human computing (like Leroi-Gourhan's table).

Conclusions

1. Cultures and calculating tools:
   - internally, calculating tools have a cultural kernel (pedagogical strategies);
   - externally, they might be correlated with cultural facts (kinds of mathematics, social function...).
2. A classification of calculating tools should at least take into account:
   - the data structures;
   - the programming style (functional or imperative);
   - the set of memorized tables.
3. The very idea of a classification suggests that various cultures explore various areas of an unique and eternal frame of human computing (like Leroi-Gourhan's table).

Conclusions

1. Cultures and calculating tools:
   - internally, calculating tools have a cultural kernel (pedagogical strategies);
   - externally, they might be correlated with cultural facts (kinds of mathematics, social function...).
2. A classification of calculating tools should at least take into account:
   - the data structures;
   - the programming style (functional or imperative);
   - the set of memorized tables.
3. The very idea of a classification suggests that various cultures explore various areas of an unique and eternal frame of human computing (like Leroi-Gourhan's table).

Limits

For a really *a priori* classification, we would need a satisfying classification of programming languages (an exhaustive list of the criteria). It still does not exist.

Thanks

Many thanks to Chen Yifu 陳怡夫 and Xie Jing 谢晶!

Bibliography

- Sunzi 孫子, *Sunzi Suanjing* 孫子算經, ca. 300.
- Cheng Dawei 程大位, *Suanfa Tongzong* 算法統宗, 1592.
- John Backus, "Can programming be liberated from the von Neumann style?: a functional style and its algebra of programs", 1978.
- Karine Chemla and Guo Shuchun, *Les Neuf Chapitres*, 2004.
- Kojima Takashi, *The Japanese Abacus: its Use and Theory*, 1954; *Advanced Abacus: Theory and Practice*, 1963.
- Joachim Lambek, "How to Program an Infinite Abacus", 1961.
- Peter Moor, "Three Myths of Computer Science", 1978.
- Alain Schärlig, *Compter avec des cailloux*, 2001.