Schlimm, D., & Neth, H. (2008). Modeling ancient and modern arithmetic practices: Addition and multiplication with Arabic and Roman numerals. In V. Sloutsky, B. Love & K. McRae (Eds.), Proceedings of the 30th annual meeting of the Cognitive Science Society (pp. 2097–2102). Austin, TX: Cognitive Science Society.

Modeling Ancient and Modern Arithmetic Practices: Addition and Multiplication with Arabic and Roman Numerals

Dirk Schlimm (dirk.schlimm@mcgill.ca)
Department of Philosophy, McGill University
855 Sherbrooke St. W., Montreal, QC H3A 2T7, Canada

Hansjörg Neth (nethh@rpi.edu)
Cognitive Science Department, Rensselaer Polytechnic Institute
Carnegie 108, 110 8th Street, Troy, NY 12180, U.S.A.

Abstract

To analyze the task of mental arithmetic with external representations in different number systems we model algorithms for addition and multiplication with Arabic and Roman numerals. This demonstrates that Roman numerals are not only informationally equivalent to Arabic ones but also computationally similar—a claim that is widely disputed. An analysis of our models' elementary processing steps reveals intricate trade-offs between problem representation, algorithm, and interactive resources. Our simulations allow for a more nuanced view of the received wisdom on Roman numerals. While symbolic computation with Roman numerals requires fewer internal resources than with Arabic ones, the large number of needed symbols inflates the number of external processing steps.

Keywords: Numeral systems; arithmetic algorithms; mathematical practice; immediate interactive behavior

Introduction

Everybody knows that it is very difficult to do arithmetic with Roman numerals. When asked to compute CXII plus MMMDCCCCXX or LVI times LXXII most readers will find this task rather daunting. But how much of this sentiment is based on inherent limitations of Roman numerals as opposed to our lack of familiarity with this particular representational system? This paper investigates whether an educated Roman citizen would have shared or mocked our present difficulties.

The intuition that arithmetic computations with Roman numerals are impossible or extremely difficult is widely shared by influential historians of mathematics (e. g., Cajori, 1919; Menninger, 1969). Some authors contend that Roman addition is simpler than ours (Norman, 1993), but almost all agree on the infeasibility of Roman multiplication. These views are regularly reiterated in popular expositions of mathematics (Hankel, 1874; Hogben, 1951; Dantzig, 1954; Murray, 1978; Ifrah, 1985; Kaplan, 2000), and echoed by cognitive scientists (Marr, 1982; Dehaene, 1997; Butterworth, 1999).

Although a few authors suggest that our familiar algorithms can be readily adapted to Roman numerals (most notably Turner, 1951; Maher & Makowski, 2001), they merely provide rough sketches that lack procedural details or multiplication tables. These contributions do not seem to have been much noticed. On the contrary, in order to show that computations with Roman numerals are possible in principle, some authors have presented rather involved and complicated-looking algorithms which may have only reinforced the view that such computations are inherently troublesome (Detlefsen et al., 1975; Kennedy, 1981). Thus, there almost appears to be some kind of conspiracy against the suitability of Roman numerals for multiplication. This paper aims to dispel this myth by exemplifying a modeling framework that illustrates the trade-offs between cognitive, perceptual and motor resources afforded by different number systems.

Number Systems In a *place-value* system with base p, the value of a numeral $a_n a_{n-1} \ldots a_2 a_1 a_0$ depends on the system's base and the value and position of each symbol. Thus, any multi-digit numeral (like 12) denotes different numbers in different bases and the symbol '2' has different meanings in 12 and 21. Place-value notation allows for a very concise representation of numbers, but also necessitates the consideration of symbol positions during computations. While early place-value notations have been used by Babylonians and in India, we will refer to our common base-10 place-value system as the *Arabic* numeral system.

The Roman numeral system discussed in this paper is a purely *additive system*, in which each symbol represents a fixed value and the value of a numeral is obtained by adding the values of all its symbols. (The subtractive notation, in which 4 is represented by IV, became common only in the Middle ages.) The basic Roman symbols are I, V, X, L, C, D, and M, representing the values 1, 5, 10, 50, 100, 500, and 1000, respectively. Thus, the Roman numeral for 512 is DXII. With the 7 symbols mentioned above, however, Roman numerals can only represent values up to 4,999. To overcome this limitation, Romans introduced new symbols by writing a bar over a numeral to signify that the value of the basic symbol was to be multiplied by 1,000 (Hankel, 1874). For convenience, we will use lower case letters to represent the values from 5,000 (v) to 1,000,000 (m). Thus, with the additional symbols v, x, l, m, c, d, and m we can express values up to 4,999,999.

From Mental to 'Environmental' Arithmetic Cognitive scientists have long been interested in the properties of representational systems and the interactions between cognitive agents, tasks, and tools (Nickerson, 1988; Norman, 1993; Zhang & Norman, 1995). On one hand, numbers are abstract entities whose properties (e. g., being a prime) and manifold relations (e. g., being a multiple of) are studied by mathematicians and are independent of any notational system. On the other hand, numbers need to be expressed in some system of numerals to be perceived and manipulated by humans. Thus,

Schlimm, D., & Neth, H. (2008). Modeling ancient and modern arithmetic practices: Addition and multiplication with Arabic and Roman numerals. In V. Sloutsky, B. Love & K. McRae (Eds.), Proceedings of the 30th annual meeting of the Cognitive Science Society (pp. 2097–2102). Austin, TX: Cognitive Science Society.

number systems are representational *artifacts* that act as interfaces between cognition and the realm of numbers.

Different representations of the same entity can vary in their informational and computational characteristics (Simon, 1978; Larkin & Simon, 1987). Two representations are *informationally* equivalent if they allow for the same information to be represented. For *computational* equivalence information that can be inferred 'easily and quickly' from one representation also needs to be available 'easily and quickly' in the other. As it is readily apparent that Arabic and Roman numerals are informationally equivalent our investigation concerns their computational characteristics.

Recent research has provided strong support for the view that most real-world cognition recruits external resources and achieves its goals through an intricate process of interaction with the physical environment (Hutchins, 1995; Kirsh, 1996; Clark, 1997). Whenever problem solving is studied in the context of environments that provide means of physical interaction, humans spontaneously distribute memory demands over internal and external resources (e. g., Cary & Carlson, 2001) and use their hands and other available resources to arrange, add, and count items (Kirsh & Maglio, 1994; Neth & Payne, 2001). To account for these phenomena, cognitive science has seen an upsurge of approaches that try to cross the traditional divide between thought and action by mapping the links between mental processes, tools, and task environments (e. g., Suchman, 1987; Hollan, Hutchins, & Kirsh, 2000).

Number systems provide a great arena to study cognition in action: Different numerals can denote the same number; each notational system introduces regularities for the representation and manipulation of symbols; and the familiarity with number systems can easily be controlled. The combination of these properties make number systems an ideal drosophila for investigations into the nature of distributed cognition.

Our Approach Our initial motivation to compare the Arabic and Roman number systems was rooted in surprise and disbelief. Given the myth that Roman numerals are unsuited for arithmetic computations it is puzzling how Romans could conduct commerce, administer armies, or rule an empire.

Methodological hurdles that had to be overcome included the fact that most details about how Romans actually used their numerals have been forgotten (Maher & Makowski, 2001) and that it is impossible to recruit experimental participants that have the same level of familiarity with Roman and Arabic numerals. To sidestep these practical problems we adopted a computational cognitive modeling approach. Although this method abstracts away from many constraints, it targets our main focus in a very systematic fashion. For a principled investigation of each system's capabilities and trade-offs, the ability to study them independently of the vagaries of historical accidents is more of a boon than a burden.

Modeling Arithmetic Practices

In this section we describe a total of four computational agents—one for each combination of notational system (Arabic and Roman) and mathematical algorithm (addition and multiplication).

General Assumptions Our models were implemented in Lisp and inspired by the ACT-R cognitive architecture (Anderson et al., 2004). Consequently, they share many of ACT-R's basic assumptions concerning the modular and functional organization of cognition. For instance, we assume a general distinction between declarative (factual) and procedural (how-to) knowledge, even though our current code does not implement any symbolic or subsymbolic dynamics of memory activation or learning.

All our artificial agents possess the ability to recognize and interpret the numerical quantities of symbols in their respective notational systems and are endowed with knowledge structures and routines that enable them to operate upon these quantities and their visuo-spatial representations. For instance, agents possess the perceptual-motor abilities to read and write externally represented symbols. To navigate to and encode different elements of an external problem display we assume that our agents' visual attention can only be directed at a single location at any particular moment.

To traverse a two-dimensional array of alpha-numeric symbols, agents shift their location of visual attention either relative to its current location (e. g., one row up) or to some extreme position in the currently attended row or column (e. g., to the top of a column). By contrast, shifting attention to any absolute coordinate requires storage and retrieval of a previously attended location index from working-memory.

All procedural operations (shifting attention, encoding symbols, storing and retrieving pointers, and perceptuo-motor operations) are modeled in a manner reminiscent of the *elementary information processes* (EIPs) of Payne, Bettman, and Johnson (1993). In this way the same EIPs can consistently be used across all four model agents discussed in this paper. In addition, different versions of one algorithm and algorithms operating on different numeral systems can be compared by quantifying the types and frequencies of EIPs employed to complete the task. To facilitate the presentation of our results, we categorize EIPs into functional units of perception, attention, memory, and motor actions, even though any such alignment of algorithmic operations with psychological constructs contains some arbitrary elements.

Our agents are disembodied entities that exist merely in the ephemeral realms of code. Nevertheless, the nature of their inputs and outputs profoundly affects their organization and operation. Not only are the agents presented with an external (but simulated) representation of a problem, but this representation is changed throughout an algorithm's execution (e. g., by writing or deleting symbols). This yields a modified output representation that contains not just the desired result but also traces of the completed process.

Arabic Addition Agent Our Arabic addition agent has declarative knowledge of the sums of all 100 single-digit addition facts (from $0+0$ to $9+9$). To invoke its procedural knowledge, it first must be presented with a problem. More precisely, the agent requires a written representation of Arabic numerals in which all addends have been written out in right-justified rows of a two-dimensional array, each addend in its own row. For instance, to add 3920 and 112 the agent would have to be presented with the following array:

```
- - - -
3 9 2 0
- 1 1 2
? ? ? ?
```

Initially, the agent directs its visual attention to the rightmost column of the bottom number row. The symbol there is encoded and recognized as denoting the numerical quantity 2, which initializes an internal memory slot (ims-01) to store intermediate results. Next, the agent shifts its attention one row up to encounter and encode the numeral 0. It then retrieves the addition fact that $2+0=2$, updates its ims-01 to the current sum of 2, and shifts its attention up another cell. Upon encoding that the attended location is empty it attends the bottom of the column and writes down the numeral representing the current ims-01 value of 2. The agent then shifts its attention one column to the left and proceeds to add its elements in the same way.

As most readers will be familiar with this algorithm, we only point out some non-intuitive aspects: First, the single-digit limit on addition facts necessitates an underlying knowledge structure that represents double-digit numbers into separate digits for tens (ims-10) and units (ims-01). Whenever ims-01$+a$ reaches or crosses a decade boundary multiple memory updates (of ims-01 and ims-10) ensue. For instance, adding $28+6=34$ requires the retrieval of *two* addition facts ($8+6=14$ and $(2+1=3)$ and sets ims-01 to 4 and ims-10 to 3. Secondly, whenever ims-10 is non-zero after adding all numbers of a given column, its value is written into the uppermost cell of the column to its left as a *carry*. As the agent always checks one cell above the row containing the first addend when traversing a column from bottom to top, it automatically encodes and adds any carry when processing the next column.

The agent's output is a processed number array containing the resulting sum 4032 (in the last row), but also any written carries in the top row as markers of its process:

```
1 - - -
3 9 2 0
- 1 1 2
4 0 3 2
```

Notice that supplying the agent with an input array of right-justified numerals caters towards certain properties of place-value notations and facilitates the processing of columns by vertically aligning numerals of the same level of magnitude.

Roman Addition Agent The knowledge structures necessary to compute with Roman numerals include the order of basic symbols (according to their value) and a simplification rule for each symbol, like IIIII $\rightarrow$ V, or VV $\rightarrow$ X. Applying these rules requires the agent to be able to count up to 5 symbols. Simplification is similar to the 'carry' used in the Arabic algorithm, since it results in a change in the symbols representing values at the next order of magnitude.

The most direct algorithm for adding two Roman numerals would just write them one after the other, order the symbols by their values, and then simplify. While this procedure is conceptually very simple, it involves numerous write and delete operations that are costly on paper. Thus, rather than computing with the Roman numerals as a single string of symbols, it is more convenient to arrange the symbols in a separate 'working table,' each row of which contains all symbols of a particular kind. (For illustrative purposes, these rows are labeled "Line I", "Line V," etc. below.)

The problem presentation to the Roman addition agent is in the form of a right-justified two-dimensional array. Thus, the task of adding 3920 and 112 is presented as:

```
M M M D C C C C X X
- - - - - - C X I I
```

Each addition task is divided into two sub-tasks: (i) Copy every symbol into the respective row of the working table, and then (ii) simplify this table according to the simplification rules. In the first sub-task, as long as there are symbols in the array, the agent has to shift its attention to a specific symbol in the array, read it, delete it (so that it will not be copied again), then shift its attention to the first empty position in the respective row of the table, and finally write the symbol into the table. Since there are 14 symbols in the array shown above, this series of activities has to be carried out 14 times, resulting in the following working table:

```
Line M : M M M
Line D : D
Line C : C C C C C
Line L : -
Line X : X X X
Line V : -
Line I : I I
```

The exact number and kind of the various required shifts of attention depend entirely on the specific addition algorithm that is being modeled. For example, instead of deleting each symbol that the agent read from the array and later searching the array for the next non-empty position (which results in a large number of read operations and attention shifts to the next element on the right), the agent could also have remembered the array position of read symbols and later recalled them to shift its attention to the next symbol on the right. Such an alternative addition algorithm would have resulted in fewer read and attention-shift operations at the cost of storing and retrieving absolute locations in working memory. Thus, our modeling approach allows to identify strengths and weaknesses of particular algorithms and uses them to assess specific demands on cognitive abilities, motor skills, etc.

Before the final result of the addition can be read off, the working table needs to be simplified. In this sub-task the agent traverses each of the lines in the table, beginning with the one on the bottom. In each line, it counts the symbols until the maximum number of allowed symbols is reached, in which case these symbols are deleted and a new symbol is added in the line above. In our model the knowledge about the number of symbols maximally allowed in a line is coded in the respective simplification rule, which is stored in the agent's long-term memory. Thus, the agent has to count up to 5 symbols in each line until it can apply the simplification rules (in this example both CCCCC → D and DD → M are applied once), which puts additional demands on working memory. The simplifications themselves involve writing one `D` and two new `M`s into the table and deleting a total of 7 symbols (five `C`s and two `D`s). Once all lines are simplified, the end result can be read off line by line (from top to bottom) as MMMMXXXII, representing 4032.

Arabic Multiplication Agent Our Arabic multiplication model implements the long multiplication algorithm taught in grade schools. The corresponding agent is based on the Arabic addition agent, extended by declarative knowledge of all 100 single-digit multiplication facts (0×0 to 9×9) and procedural knowledge that allows to process written representations of arbitrary 2-factor multiplication problems. For instance, to multiply 56×72 the agent's input is simply '`5 6 * 7 2`'.

In essence, the long multiplication algorithm multiplies each digit of the second factor (72) with each digit of the first (56), requiring four separate multiplications. Whenever a retrieved product exceeds 9 (e. g., $2 \times 6 = 12$) only the 'ones' digit (ims-01=2) is offloaded into the mid-section of the array, whereas the 'tens' digit (ims-10=1) is carried to the top of the next digit of the second factor (above the 5) to be added to the next intermediate product ($2 \times 5 = 10$).

In contrast to the Arabic addition agent, the multiplication model routinely relies on its ability to store and retrieve the position of currently processed digits. Without this working memory resource it neither would be able to jump back and forth between currently focused digits nor would it find the locations to which it needs to offload intermediate results for subsequent addition. To find these locations, the model not only uses remembered positional indices but also relies on the symbols it has already written. Again, the details of the model's processing characteristics jointly depend on the problem, its procedural knowledge, and the outputs of previous processing steps.

Another novelty in the Arabic multiplication model is the necessity to delete already added carries (operationalized by writing 'x' in their location). This is done to prevent the model from processing them repeatedly. Instantly dealing with carries adds complexity to the overall process and increases mental and perceptual-motor efforts (by interleaving addition with multiplication), but yields the benefit of reducing the number of addends to be processed later.

After completing all multiplications, the model's external problem representation is:

```
x - - - -
5 6 * 7 2
- - - - -
- 3 9 2 -
- - 1 1 2
? ? ? ? ?
```

Table 1: Multiplication table for Roman numerals.

×	I	V	X	L	C	D	M
I	I	V	X	L	C	D	M
V	V	XXV	L	CCL	D	MMD	v
X	X	L	C	D	M	v	x
L	L	CCL	D	MMD	v	xxv	l
C	C	D	M	v	x	l	c
D	D	MMD	v	xxv	l	ccl	d
M	M	v	x	l	c	d	m

At this point, the model needs to add the two intermediate factors ($3920 + 112$). This problem has already been described to illustrate the Arabic addition agent. Thus, mastery of addition is a prerequisite for multiplication with Arabic numerals just as applying simplification rules is an integral part of addition with Roman numerals.

Roman Multiplication Agent Analogous to the Arabic multiplication agent the Roman multiplication agent requires declarative knowledge of all necessary single-symbol multiplication facts (see Table 1). Interestingly, the Roman multiplication table has only 49 entries (for factors up to M) and is thus smaller than the Arabic one, and it also shows a greater degree of regularity. As before, these multiplication facts are assumed to reside in the agent's long-term memory. As with the other agents, the factors to be multiplied are presented in an input array of symbols, e. g., for 56×72:

```
- - L V I
L X X I I
```

Our algorithm for multiplying two Roman numerals is simple (in terms of required cognitive resources) and could easily be taught. Its two phases are identical to those of multiplying Arabic numerals: (i) Multiply each symbol of one factor with each symbol of the other factor, and (ii) add the intermediate results. The Roman multiplication agent begins by moving through the two input rows and multiplying each symbol of the first factor with each symbol of the second one. The results of these single-symbol multiplications are retrieved from memory and are written directly in a working table like that used for addition. In the example of 56×72, the 15 single-symbol multiplications yield the following table:

```
Line M : M M
Line D : D D D
Line C : C C
Line L : L L L L L
Line X : X X
Line V : V V
Line I : I I
```

Notice that no special operations for adding the intermediate results are necessary after all products have been written into the working table. After simplification the table shows the final product MMMMXXXII. Although this final result is the same as in the previous addition example the intermediate working table that had to be simplified was different. Thus, different steps were involved in the simplification process, using different elementary operations. We notice again how the necessary cognitive and motor operations depend crucially on the external format of the problem at hand.

Table 2: Operator counts for simulating 1,000,000 additions of the form $x+y+z$ (for $x,y,z = 1\ldots100$) and 10,000 multiplications of the form $x\times y$ (for $x,y = 1\ldots100$). A unit corresponds to 1,000 operations and values were rounded.

		Addition			Multiplication		
Module	Measure	Arabic	Roman	*Ratio*	Arabic	Roman	*Ratio*
Input	Total number of symbols	5,760	15,030	2.61	38	100	2.63
Perception	Total READs	8,116	244,166	30.08	325	1,332	4.10
Attention	Total SHIFTs (abs. and rel.)	16,625	336,413	20.24	601	2,891	4.81
Memory (LTM)	Addition facts recalled	4,437	–		36	–	
	Multiplication facts recalled	–	–		37	251	6.78
	Simplification rules applied	–	3,670		–	88	
Memory (WM/STM)	Carries during addition	1,608	–		5	–	
	Carries during multiplication	–	–		24	–	
	ims-01 changes	8,953	–		135	–	
	ims-10 changes	3,498	–		37	–	
	STOREs of locations	–	3,670		56	329	5.88
	RETRIEVEs of locations	–	3,670		93	329	3.54
	Counting steps	–	31,380		–	636	
Output/Motor Actions	Total WRITEs	3,637	18,700	5.14	102	359	3.52
	Total DELETEs	–	27,710		10	277	27.70

Methodology Due to the intricate dependencies between task, algorithms, and internal and external resources, the selection of problems profoundly affects the complexity of computational processes. For instance, if the Roman addition agent was asked to add IIII, V, and XXXX, no simplifications would be necessary after the inputs had been written into the working table. Likewise, the Arabic multiplication agent would not need any operations for dealing with carries when asked to multiply 124 by 201.

In order not to favor or discriminate against any agent or system we selected the ranges of our simulations solely on the basis of numerosity. For the addition agent we simulated all additions of three numbers ranging from 1 to 100, i. e., $x+y+z$ (for $x,y,z = 1\ldots100$), and for the multiplication agents all multiplications of two numbers of the same range, i. e., $x\times y$ (for $x,y = 1\ldots100$). Thus, our results are based on agent solutions to 1,000,000 addition problems and 10,000 multiplication problems. This large range of problems far exceeds the number of problems that could be performed in an experimental setting by human subjects, which demonstrates the power of our modeling approach. Moreover, we believe that there is some face validity to our samples, as arithmetic problems with small numbers of addends and factors occur frequently in naturalistic settings.

Results and Discussion

The results of our simulations are summarized in terms of the total number of elementary processing steps in Table 2. Prior to any further analysis, the mere fact that our agents could successfully solve all problems demonstrates that addition and multiplication with Roman numerals can be carried out with the same interactive (cognitive and perceptual-motor) resources as computations with Arabic numerals.

A closer look reveals that some operations are needed in one system but not the other. Roman agents never need to recall any addition facts, but rely on memorized rules to simplify intermediate results. Notice, however, that the requirement to remember seven simplification rules contrasts sharply with the 100 single-digit addition facts to be remembered by the Arabic agents. Similarly, the Roman addition agent only needs to count up to five symbols during the simplification phase, whereas its Arabic counterpart has to compute and retain many different intermediate sums in working memory.

Comparisons between agents reveal that the Roman agent's perceptual steps, attention shifts, and motor actions are vastly more numerous than those of the Arabic agent. This is in part due to the fact that the first sub-task when adding with Roman numerals is to copy the input into the working table, which requires many elementary operations. One reason for the lower counts for Arabic numerals is that the assumed problem presentation favors the Arabic agent by reducing its perceptual and attentional steps. Indeed, the ratios of the number of operations between the Arabic and the Roman algorithms for multiplication are much lower.

A striking difference between the Arabic and the Roman agents is the different lengths of their inputs, whose ratio is roughly 1 : 2.6. This lets the number of required multiplications grow at a rate of $2.6^2 = 6.8$. Consequently, the Roman multiplication agent recalls individual multiplication facts much more frequently. However, due to its smaller set of multiplication facts we can assume that the retrieval of these facts is much more practiced and error-proof than in the Arabic case.

The overall complexity of the computations depends crucially on the precise format of external inputs and the availability of interactive resources. Imagine how the addition algorithms would differ if the input was presented as a linear sequence of numerals: Whereas the Roman agent could remain virtually unchanged, the Arabic agent would either have to rewrite the numerals into its standard column format or use greater working-memory resources to keep track of the relative positions within numerals. Similarly, using artifacts like fingers or an abacus instead of paper and pencil profoundly affects the balance of elementary operations. In fact, by using an abacus instead of the working table, our Roman agents could dramatically reduce their write and delete operations, as well as the information to be held in working memory.

Using our models to identify the precise effects of changes in task format or interactive resources presents important opportunities for optimizations. For instance, the Roman algorithms could be made much more efficient by streamlining the second phase of simplifying the working table, e. g. by replacing the counting and deleting of individual symbols by steps that allow for the perception of symbol strings akin to the reading of words (instead of individual letters).

In future work we intend to address numerous limitations of our current models: Adding time estimates (e. g. by empirical validation studies) will allow to use temporal duration as a common currency for the comparison between algorithms. Re-implementing our agents in ACT-R will allow to explore dynamic memory and learning processes. Models of higher fidelity will enable us to study alternative algorithms (e. g., Arabic addition with minimal internalization and optimized Roman algorithms), generalize to other numeral systems (e. g., binary arithmetic and the sexagesimal system used by the ancient Babylonians), and combine algorithms with artifacts (e. g., an abacus) to analyze the resulting trade-offs between cognitive and motor skills.

Conclusion

We demonstrated that symbolic computations are certainly possible with Roman numerals, thus dispelling the myth mentioned in the introduction. Moreover, by modeling arithmetic algorithms we were able to conduct a quantitative analysis of their performance characteristics, which allowed us to provide a more nuanced assessment of the computational properties of the Arabic and Roman numeral systems.

We can now qualify the view that addition with Roman numerals is simpler than with Arabic ones: It is simpler only insofar as it does not require recalling any addition facts from long-term memory, nor keeping intermediate results in working memory. However, it does require the repeated counting of up to five symbols, the copying of symbols into the working table, and a vast number of basic perceptual-motor operations during the simplification process. Similar trade-offs between perceptual-motor and memory operations have been found for multiplication, for which the longer length of Roman numerals becomes a considerable disadvantage.

Overall, we have shown that the Achilles heel of Roman numerals is not that they are (qualitatively) unsuitable for algorithmic usage, but their greater cost in terms of the number of necessary elementary information processes, especially those using external resources. Hence, one must not always believe what 'everybody knows.'

Acknowledgements

We are grateful to PhiMSAMP and the DFG for providing the opportunity and funding that sparked this research.

References

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., & Qin, Y. (2004). An integrated theory of the mind. *Psychological Review*, *111*(4), 1036–1060.

Butterworth, B. (1999). *The mathematical brain.* London: Papermac.

Cajori, F. (1919). *A history of mathematics.* Macmillan, New York.

Cary, M., & Carlson, R. A. (2001). Distributing working memory resources during problem solving. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *27*(3), 836–848.

Clark, A. (1997). *Being there: Putting brain, body, and world together again.* Cambridge, MA: The MIT Press.

Dantzig, T. (1954). *Number, the language of science* (4th, revised and augmented ed.). Macmillan. (First edition 1930.)

Dehaene, S. (1997). *The number sense: How the mind creates mathematics.* New York: Oxford University Press.

Detlefsen, M., Erlandson, D. K., Heston, J. C., & Young, C. M. (1975). Computation with Roman numerals. *Archive for History of Exact Sciences*, *15*, 141–148.

Hankel, H. (1874). *Zur Geschichte der Mathematik in Alterthum und Mittelalter.* Leipzig: B. G. Teubner.

Hogben, L. (1951). *Mathematics for the million* (3rd ed.). New York: W. W. Norton. (First edition 1937.)

Hollan, J., Hutchins, E., & Kirsh, D. (2000). Distributed cognition: toward a new foundation for human-computer interaction research. *ACM Transactions on Computer-Human Interaction (TOCHI)*, *7*(2), 174–196.

Hutchins, E. (1995). *Cognition in the wild.* Cambridge, MA: The MIT Press.

Ifrah, G. (1985). *From one to zero: A universal history of numbers.* New York: Viking.

Kaplan, R. (2000). *The nothing that is. A natural history of zero.* Oxford: Oxford University Press.

Kennedy, J. G. (1981). Arithmetic with Roman numerals. *American Mathematical Monthly*, *88*(1), 29–32.

Kirsh, D. (1996). Adapting the environment instead of oneself. *Adaptive Behavior*, *4*(3–4), 415–452.

Kirsh, D., & Maglio, P. (1994). On distinguishing epistemic from pragmatic action. *Cognitive Science*, *18*(4), 513–549.

Larkin, J. H., & Simon, H. A. (1987). Why a diagram is (sometimes) worth ten thousand words. *Cognitive Science*, *11*(1), 65–100.

Maher, D. W., & Makowski, J. F. (2001). Literary evidence for Roman arithmetic with fractions. *Class. Philology*, *96*(4), 376–99.

Marr, D. (1982). *Vision: A computational investigation into the human representation and processing of visual information.* San Francisco: W. H. Freeman.

Menninger, K. (1969). *Number words and number symbols.* Cambridge, MA: MIT Press.

Murray, A. (1978). *Reason and society in the middle ages.* Oxford: Clarendon Press.

Neth, H., & Payne, S. J. (2001). Addition as Interactive Problem Solving. In J. D. Moore & K. Stenning (Eds.), *Proceedings of the Twenty-Third Annual Conference of the Cognitive Science Society* (pp. 698–703). Mahwah, NJ: Lawrence Erlbaum Associates.

Nickerson, R. S. (1988). Counting, computing, and the representation of numbers. *Human Factors*, *30*(2), 181–199.

Norman, D. A. (1993). *Things that make us smart. defending human attributes in the age of the machine.* Cambridge, MA: Perseus Books.

Payne, J. W., Bettman, J. R., & Johnson, E. J. (1993). *The adaptive decision maker.* Cambridge University Press.

Simon, H. A. (1978). On the forms of mental representation. In C. W. Savage (Ed.), *Perception and cognition. Issues in the foundations of psychology* (Vol. IX, pp. 3–18). University of Minnesota Press, Minneapolis.

Suchman, L. A. (1987). *Plans and situated actions: The problem of human-machine communication.* Cambridge, UK: Cambridge University Press.

Turner, J. H. (1951, Nov.). Roman elementary mathematics: The operations. *The Classical Journal*, *47*(2), 63–74, 106–108.

Zhang, J., & Norman, D. A. (1995). A representational analysis of numeration systems. *Cognition*, *57*, 271–295.