METAPOST macros for drawing Chinese and Japanese abaci

Denis Roegel

for 荷花

Abstract

This article shows how Chinese (算盘, suànpan) and Japanese abaci (算盤, soroban) can be drawn with METAPOST, and is illustrated with the details of a simple algorithm.

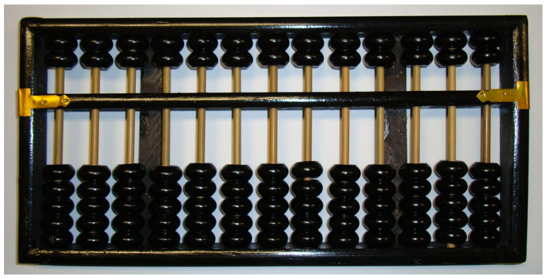

Figure 1: A traditional Chinese abacus (算盘, suànpan) with all its beads set to 0. (Photograph: author's collection)

1 Introduction

One of the oldest calculating tools still in use today is the abacus (Knott, 1886; Smith and Mikami, 1914; Li Shu-T'ien, 1959; Needham and Wang Ling, 1959; Moon, 1971; Ifrah, 2000; Martzloff, 2006). It is now mainly used in Asia for performing arithmetical calculations. Until recently, the use of the abacus was still taught in Chinese schools and there were abacus proficiency tests for applying for certain occupations. In Japan, the first such proficiency test was held in Tokyo in 1928.

An experienced abacist can be very fast, faster than a person using a handheld calculator, at least for small values and basic processes, such as addition or multiplication. Abaci can be used for more advanced tasks, such as extracting a square or cube root, but these tasks may require a non-standard abacus, large enough to store all values.

An abacus is basically a tool to store numerical values by the position of beads on rods. The values which are stored can be changed following an algorithm and an abacist can operate very quickly using automatic patterns which are applied in sequence.

Abaci have a long history and there have been many variants which will not be considered here. The Chinese abacus probably goes back 1000 years or more. Other civilizations, such as Rome and Greece, have used related tools where the stored values were marked by pebbles, or special tokens.

In this article, we describe METAPOST macros to draw the common Asian abaci as well as the operations which are performed on them.

2 Types of abaci

We will consider only two types of abaci, namely the typical current Chinese and Japanese abaci which are still in use.

2.1 The *suànpan*

The Chinese abacus is called 算盘 (*suànpan*). The Chinese word 算 (suàn) means "to calculate" and 盘 (pan) is the word for a "tray". A *suànpan* can come in various widths. The standard *suànpan* has 13 rods with five beads in the lower deck and two beads in the upper one (figure 1). Each bead in the upper deck is worth five beads in the lower one. Four of the lower and one of the upper beads are normally enough for decimal computation, but it seems that the extra beads were originally used to represent an hexadecimal digit, which was useful for the traditional weighing system where one jīn (斤) is equal to sixteen liǎng (两) (about 50 grams). However, these extra beads were also useful to simplify (and accelerate) some computations (Moon, 1971, p. 85).

2.2 The *soroban*

The 算盤 (そろばん, *soroban*) is the Japanese form of the Chinese *suànpan*, and was derived from it. "盤" is the traditional character for "tray", still used in Japan. The basic *soroban* usually also has 13 rods, but there are only four beads in the lower deck and one in the upper deck. In some cases, there are five beads in the lower deck, but only one in the upper deck. A *soroban* has an additional feature which distinguishes it from the *suànpan*, namely that every third rod is marked by a dot. These are the *unit rods*. This makes it easier for calculations and for setting values on the *soroban*.

3 The suanpan METAPOST package

In order to show how to operate an abacus, we have written a METAPOST package to produce simple — but flexible — abaci representations. METAPOST is a powerful graphical tool, well suited for technical or geometrical drawings (Goossens, Mittelbach, Rahtz, Roegel, and Voss, 2008; Hobby, 2008). All the figures in this article were produced with the suanpan METAPOST package, available on CTAN. This package should however be seen only as a basis and it can easily be extended, for instance to vary the shape of the beads, or to automatically demonstrate more complex algorithms than what we show here.

There are currently two other packages by Alain Delmotte for drawing a *soroban* with PSTricks or PGF, but these packages do not (yet) implement calculation algorithms (Delmotte, 2007a; Delmotte, 2007b).

4 Algorithms on abaci

Calculating on an abacus amounts to resetting the abacus to a standard position, then setting (storing) a value, and then performing some operation, following a known algorithm. The result is then read off the abacus.

4.1 Initial position

Figure 1 shows the initial position of a Chinese *suànpan* and figure 2 compares the Chinese and Japanese abaci. The two decks are divided by a bar known as the reckoning bar. In the standard position, all the beads are moved away from the reckoning bar, and this represents the value 0. Each rod represents one decimal (or sometimes hexadecimal) place, the units being normally at the right. The rods are usually numbered, but this feature can be deactivated using the boolean `rod_numbers` as shown below.

Using the `suanpan` macros, the initial position of a *suànpan* is obtained as follows:

```
input suanpan
setup_abacus(N=13,NBL=5,NBU=2,
             bead="suanpan",units=0);
beginfig(1);
  rod_numbers:=false;
  reset_abacus;draw_abacus;
endfig
end
```

The `setup_abacus` macro sets the number of rods (`N`), as well as the number of beads in each deck (`NBL` and `NBU`), the type of bead (`bead`) and the unit rods (`units`). The arguments are given as *key=value* pairs. Currently two bead types are possible, corresponding to the strings `"suanpan"` (almost round beads) and `"soroban"` (biconal beads).

4.2 Setting a value

Setting a value on an abacus is equivalent to moving some of the beads towards the reckoning bar. A bead from the lower deck represents one unit of the corresponding place, and a bead from the upper deck represents five units. Using four of the lower beads and one upper bead, one can therefore set values up to $5 + 4 = 9$. If all the beads of a Chinese abacus are used, and the upper beads are still weighing 5 lower beads, then the maximum value on a rod is $5 + 5 + 5 = 15$. All values between 0 and 15 can be expressed that way.

In the `suanpan` macros, the number of beads set in each deck is stored in two arrays, and all values of this array can be set by hand as follows (figure 3, left):

```
beginfig(3);
  reset_abacus;
  valL[1]:=2;valL[3]:=5;
  valU[2]:=1;valU[4]:=2;
  draw_abacus;
endfig;
```

Proceeding this way can be useful when the abacus needs to be set in a non-standard decimal position. This is the case above, with one of the rods having 5 beads set in the lower deck. The `suanpan` macros do currently not support hexadecimal computations, but they could easily be handled, based on the implementation for decimal numbers.

In the usual case, at most four beads are set in the lower deck. There is a macro `set_abacus_val` which automates the setting of an initial value (figure 3, right):

```
beginfig(4);
  reset_abacus;
  set_abacus_val("651324");
  draw_abacus;
endfig;
```

If n is the number of rods, only the rightmost n digits of the initial value are taken into account.

4.3 Adding a value

Once a value is stored in the abacus, we can apply simple algorithms to change this value. In this article, we will consider only addition. Even for addition, one can contemplate different methods, and one typical algorithm performs the addition not from right to left, but from left to right. In order to demonstrate the process, the `suanpan` package provides a macro `add_val` which decomposes the addition in a number of steps. This command *should not* be used inside a `beginfig`/`endfig` pair, as it generates a number of such environments. We demonstrate the calculation on a *soroban*, using the initial value 651324 of the above example (figure 4).

```
setup_abacus(N=13,NBL=4,NBU=1,
             bead="soroban",units=1);
set_abacus_val("651324");
add_val(v="82363456",iv=100,fig=true);
```

If the main file is `abacus.mp`, the above command produces files `abacus.100`, `abacus.101`, ..., `abacus.108`, which can then be included in a LaTeX file.

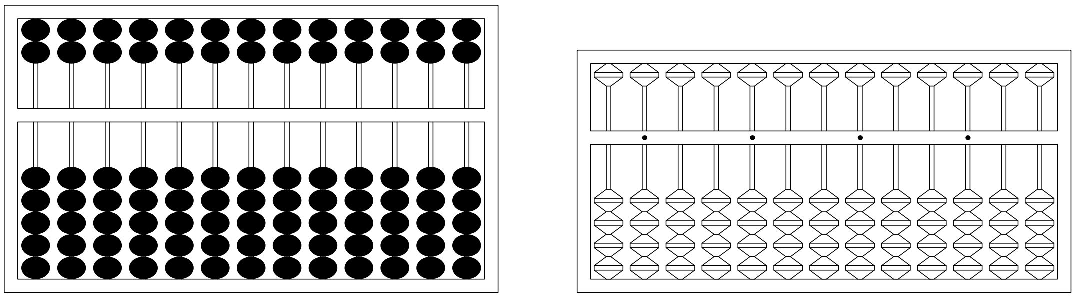

Figure 2: Initial setting of a 算盘 (*suànpan*, left) and of a 算盤 (*soroban*, right).

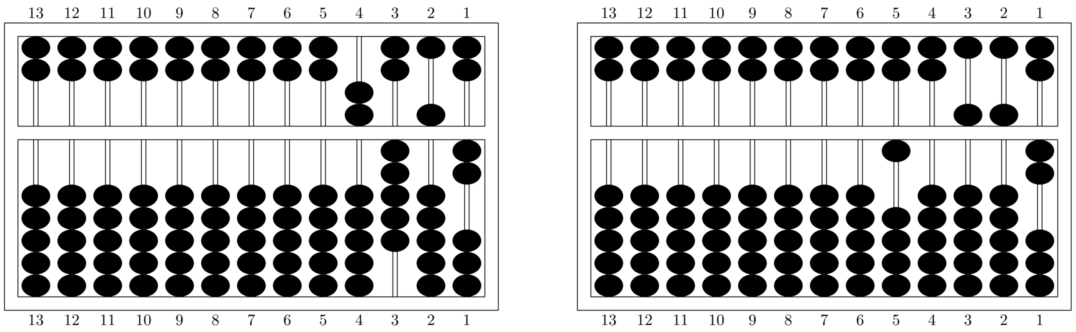

Figure 3: The decimal value 10552 represented in a non-standard way (left) and a standard one (right).

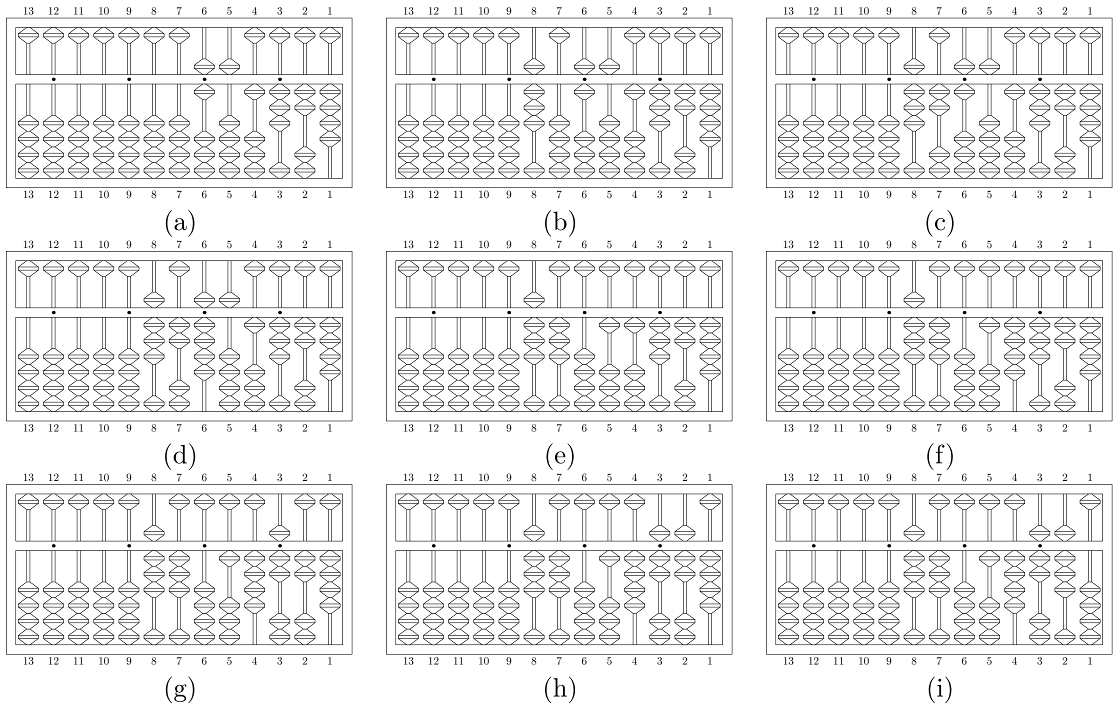

Figure 4: The decomposition of an addition in nine steps on a *soroban*.
(a) represents the initial value 651324, and we add 82363456 in eight steps, one for each digit.

Figure 4 (a) shows the initial state of the abacus, with the value 651324. In a first step (b), we add 8 to rod 8, hence moving three beads from the lower deck and one bead from the upper deck. The other beads are not moved. Then, step (c) adds 2 to rod 7. So far, the changes were straightforward, since these two rods were initially set to 0. In step (d), 3 is added to rod 6, which now contains the value 9. Step (e) adds 6 to rod 5 which contained 5, and this leads to the value 11, hence only 1, and a carry of 1. So, this configuration shows one bead set in the lower deck of rod 5, no bead set in the upper deck, and an additional bead carried to rod 6. However, rod 6 already contained the value 9, and this leads itself to another carry. Finally, it is rod 7 which has an additional bead set in its lower deck. This process goes on digit by digit, until the units have been added. Every single digit addition is therefore sometimes decomposed in multiple steps which are not detailed here. They could be made explicit by other macros.

The `add_val` macro can also be used without generating new drawings, by giving `false` for its `fig` argument. It then only applies the standard addition algorithm and produces the result in the stored arrays.

Of course, additions can also be simulated by doing the computation externally and setting new values for each step. As such, the `suanpan` macros could be used as a back-end for other tools.

Here is for instance an addition not producing any intermediate figures:

```
beginfig(200);
  reset_abacus;reset_abacus_gray;
  set_abacus_val("82951324");
  draw_abacus;
endfig;

beginfig(201);
  set_abacus_val("82951324");
  add_val(v="60000",iv=100,fig=false);
  draw_abacus;
endfig;
```

An addition can produce an overflow, and this sets the `overflow` boolean to `true`. The `add_val` macro resets this value to `false` before performing the addition.

4.4 Shortcuts for fast computation

In order to become proficient with the abacus, it is useful to memorize a number of patterns which recur very frequently and which enable automating much of the computation. A simple example will show what is meant.

If one of the rods of the abacus has three beads set in the lower deck, and one more bead has to be set, then this additional bead can merely be moved towards the reckoning bar. However, if three units had to be added instead of one, then a novice user of the abacus would probably mentally compute $3 + 3 = 6$, then remove 5 and set only one bead in the lower deck, while adding one too in the upper deck. However, this is inefficient, because the burden of the computation is on the user. Instead, if three beads cannot be moved, one should consider $3 = 5 - 2$ and therefore perform two operations: *adding* one (5) to the upper deck, and *removing* 2 from the lower deck. This is a typical shortcut, which does not require the calculation of $3 + 3 = 6$, and only requires to notice that three more beads cannot be set in the lower deck.

Some of the operations in the upper deck may also be impossible, and may require similar rewritings. If one bead (5) cannot be added in the upper deck part, we can instead write $5 = 10 - 5$ and add one (10) to the units of the next rod, and remove one bead from the upper deck of the current rod. This process then repeats until the computations have been entirely performed.

If five or more beads have to be added in the lower deck, the number of beads to be added can be written either as $5 + a$ or as $10 - b$, and whichever is possible must then be applied. For instance, if we still have three beads in the lower deck, and if we have to add six beads, we can write either $6 = 5+1$ or $6 = 10-4$. The second decomposition cannot be performed, because it amounts to removing four beads from the lower deck. But the first decomposition is possible, and so we set one more bead in the lower part, as well as one more bead in the upper deck. If the latter is not possible, we again decompose the calculation.

More complex operations, such as multiplications, divisions, square roots, etc., can be performed efficiently using tables that the abacist has to memorize. Examples of such tables for the *soroban* are given by Knott (Knott, 1886).

5 Special abaci macros

In order to explain how to operate an abacus, it is sometimes useful to mark some of the beads. Two possibilities are provided by the `suanpan` package: some of the beads can be shown in gray, or they can be marked with a label.

Using the macro `set_abacus_gray`, it is easy to put some of the beads in gray. This macro takes three arguments, given as *key*=*value* pairs. The `deck` key identifies the deck (lower or upper), and the other

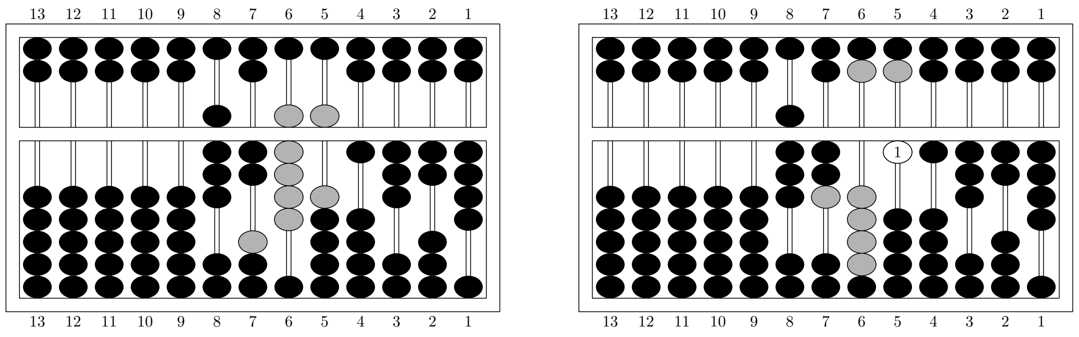

Figure 5: Highlighting one of the steps of the addition, with special marks. All the beads which have been moved have been drawn in gray, and, in addition, the sole bead moved from the lower deck of the fifth rod has been marked with ‘1’.

two are strings with one digit for consecutive rods starting from the right. The key `below` corresponds to the beads which are in the lowest positions in a deck. If the value is 2, for instance, it means that the two *top* beads in the lower part of the deck (upper or lower) will be grayed. These are the first beads that would be moved if two beads had to be set (in the lower deck) or reset (in the upper deck).

There is currently no automated way to produce these special marks, but their automation is of course possible. There are however so many different imaginable schemes, that we have decided not to implement them for the moment. Only low-level commands are currently supported.

We illustrate these commands by considering again the addition seen previously, but this time marking all the changes. The resulting configurations are shown in figure 5 and the code which produces them is the following:

```
beginfig(202);
  reset_abacus;reset_abacus_gray;
  set_abacus_val("82951324");
  set_abacus_gray(deck="lower",
     below="1010000",above="0400000");
  set_abacus_gray(deck="upper",
     below="0110000",above="0000000");
  draw_abacus;
endfig;

beginfig(203);
  reset_abacus_gray;
  add_val(v="60000",iv=100,fig=false);
  set_abacus_gray(deck="lower",
     below="0400000",above="1010000");
  set_abacus_gray(deck="upper",
     below="0000000",above="0110000");
  draw_abacus;
  mark_abacus(5,5)(btex 1 etex);
endfig;
```

In these examples, `reset_abacus_gray` merely resets all the grayed beads. It will be easy to see how the gray encoding translates to the figures.

The other macro to mark beads is `mark_abacus`. This macro overwrites a bead with a (short) label. `mark_abacus(3,5)(btex 1 etex)` writes ‘1’ over the fifth bead (from the bottom) in the third rod from the right. One of the advantages of this encoding is that even if a bead is moved, the mark will still remain on it and the command will not need to be altered.

6 Abaci in other bases

As was explained before, the Chinese abacus can be used both for computing with decimal values and with hexadecimal values, depending on the use of the two extra beads in the lower and upper decks. We can imagine other abaci, adapted to other bases.

Figure 6, for instance, shows an addition with a base-8 abacus. Each rod has three beads in the lower deck and one bead in the upper deck, and a rod can hold values from 0 to 7. The figure on the left represents the value 3401256_8. Adding 1234_8, we get 3402512_8.

Producing these drawings is straightforward: in addition to the number of beads in each deck, there is a variable `vbu` representing the value of one bead in the upper deck. In a natural base-8 abacus, the upper beads would be worth 4 lower beads, and so, we can just write the following to produce the two configurations, before and after the addition. Similar constructions are possible in other bases, but experienced users would have to adapt all their mnemonic rules to fit these new configurations.

```
vbu:=4; % upper deck value of a bead
setup_abacus(N=13,NBL=3,NBU=1,
             bead="suanpan",units=0);
```

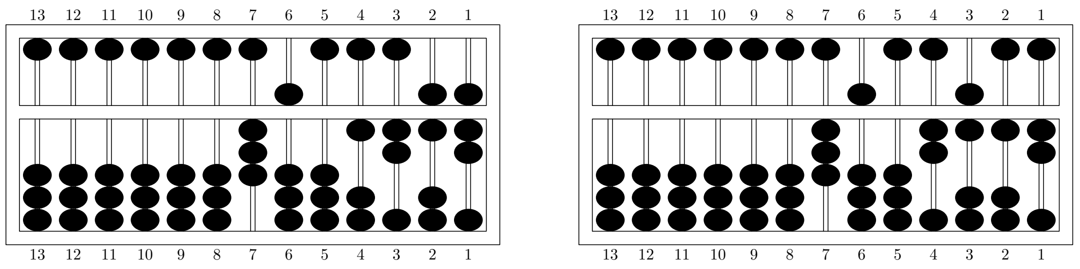

Figure 6: Base-8 abacus addition. The value on the left is 3401256_8 and we add 1234_8, which produces 3402512_8 on the right.

```
beginfig(300);
  reset_abacus;reset_abacus_gray;
  set_abacus_val("3401256");
  draw_abacus;
endfig;

beginfig(301);
  add_val(v="1234",iv=100,fig=false);
  draw_abacus;
endfig;
```

7 Conclusion and future extensions

This article is meant as an illustration of some simple METAPOST macros for drawing Chinese or Japanese abaci, and we have only strived to provide a good foundation. Many improvements are possible, both graphically and algorithmically. The abaci can be made more realistic, and in particular other bead shapes could be supported. The main possible improvements, however, concern the implementation of new algorithms. So far, we have only concentrated on the implementation of one addition algorithm, but other algorithms are possible, for instance one where the additions are performed from the right-most rod to the left-most one. More complex operations, such as multiplication, division, the calculation of square or cube roots, etc., could also be supported (Knott, 1886; Kojima, 1963; Moon, 1971; Heffelfinger and Flom, 2007). For each of these cases, it would be desirable to provide automatic output detailing each algorithm. This could easily be built upon the existing macros.

References

Delmotte, Alain. "Soroban abacus: package `pgf-soroban`". 2007a. Available on CTAN.

Delmotte, Alain. "Soroban abacus: package `pst-soroban`". 2007b. Available on CTAN.

Goossens, Michel, F. Mittelbach, S. Rahtz, D. Roegel, and H. Voss. *The LaTeX Graphics Companion, Second Edition*. Boston: Addison-Wesley, 2008.

Heffelfinger, Totton, and G. Flom. "算盤 Abacus: Mystery of the Bead". 2007. `http://webhome.idirect.com/\~{}totton/abacus`.

Hobby, John. "METAPOST: A User's Manual". 2008. Updated version of the original manual; available at `http://tug.org/docs/metapost/mpman.pdf`.

Ifrah, Georges. *The Universal History of Computing: From the Abacus to the Quantum Computer*. New York: John Wiley, 2000.

Knott, Cargill Gilston. "The Abacus in its Historic and Scientific Aspects". *Transactions of the Asiatic Society of Japan* **14**, 18–71, 1886.

Kojima, Takashi. *Advanced Abacus: Japanese Theory & Practice*. Tokyo: Charles E. Tuttle & Company, 1963.

Li Shu-T'ien. "Origin and Development of the Chinese Abacus". *Journal of the ACM* **6**(1), 102–110, 1959.

Martzloff, Jean-Claude. *A history of Chinese Mathematics*. New York: Springer, 2006.

Moon, Parry. *The Abacus: Its history; its design; its possibilities in the modern world*. New York: Gordon and Breach Science Publishers, 1971.

Needham, Joseph, and Wang Ling. *Science and Civilisation in China, volume 3: Mathematics and the Sciences of the Heavens and Earth*. Cambridge: Cambridge University Press, 1959.

Smith, David Eugene, and Y. Mikami. *A history of Japanese mathematics*. Chicago: The Open Court Publishing Company, 1914.

◇ Denis Roegel
LORIA — BP 239
54506 Vandœuvre-lès-Nancy cedex
France
`roegel (at) loria dot fr`
`http://www.loria.fr/~roegel`