

H.A. Cohen, *Making Algebra Concrete using a Microcomputer: An Algebraic Programming Language for the Control of Robots*, in P. Williamson (Editor), *Learn to Love Mathematics*, Mathematics Association of Victoria, Melbourne, 1980, pp 336-350. Preprint + Afterthought

MODELLING BOOLEAN ALGEBRA WITH TINKERTOY OR MECCANO: HOW TO CONSTRUCT NIM PLAYING MACHINE

Dr Harvey A. Cohen

Computer Science and Applied Mathematics Departments
La Trobe University Bundoora,
Victoria 3083

Boolean Algebra is the basis for digital computers. From the perspective of pure mathematics boolean algebra is rather simple, as it involves quantities which can have only the value 0 or 1. The two basic operations in Boolean algebra are simply an addition and a multiplication operation. The algebra was discovered by George Boole, who interpreted it in terms of the logical calculus, with the value 1 representing the truth value TRUE, and 0 representing the truth value FALSE. Thus to take a different perspective, Boolean algebra can be modelled by the predicate calculus of (symbolic) logic. But logic despite its significance cannot be the medium of interpretation of Boolean algebra in an introductory course which could be presented at a High School level. Some more concrete modelling must be supplied in order to make this algebra intelligible. There are no commercial teaching materials that provide concrete modelling of Boolean algebra. So in this paper it is explained how models for the operations of Boolean algebra, which are at the same time mechanical models for the corresponding purely logical operations can be designed. These models could be constructed firstly by the teacher, and secondly as science fair projects for senior students.

Actually to describe Boolean algebra as the basis for computing is true but inadequate. Computing is a process that takes place in time, with the result of one step the input for the next. Really interesting mechanical models bearing on computing must in some way demonstrate this feature. Fortunately as I hope to demonstrate it is fairly easy to design not only mechanical models for a static algebra, but mechanical counterparts of electronic computer components. Thus using Tinkertoy or Meccano one can construct a mechanical .flip-flop., a basic unit in the Arithmetical Logical Unit (ALU) of a digital computer.

Thus given enough Tinkertoy one could build a .real, programmable computer. This idea of a mechanical computer is not at all new. In the middle of the nineteenth century Babbage designed his .Analytical Engine., a general purpose programmable computer involving many thousands of finely engineered components. But previously Babbage had used 15000 pounds supplied by the British Admiralty, and 7000 sterling of his own money in the unsuccessful endeavour to construct » simpler computer, .The Difference Engine., and the Analytical Engine had never been constructed. Although the title of this paper may suggest that Babbage might have fared better if Meccano or Tinkertoy had been invented in 1850, more seriously one must deduce that it is not practical to construct general purpose computers from mechanical components.

H.A. Cohen, *Making Algebra Concrete using a Microcomputer: An Algebraic Programming Language for the Control of Robots*, in P. Williamson (Editor), *Learn to Love Mathematics*, Mathematics Association of Victoria, Melbourne, 1980, pp 336-350. Preprint + Afterthought

Even though even today it might be impossible to construct Babbage's Analytical Engine from mechanical components, it is certainly possible to construct special purpose computers without electronics. A special purpose computer is one which executes just one particular program, which is in electronics jargon .hardwired.. About 1890 Torres Y Quevedo constructed a mechanical chess playing machine, which played the end-game King and Rook versus Rook. To play such an end-game involves Quevedo's machine in making for each move a decision out of a great range of possibilities. A far simpler game is NIM played with one pile for which this paper describes an easily constructed machine.

The plan of this paper is to first define what Boolean algebra is in mathematical and logical terms. We then indicate how the simplest logical gates can be designed and constructed from Tinkertoy, Meccano, or similar construction material. The idea of a flip-flop is then presented, with an indication of one of several ways to model its functioning. Finally we turn to NIM, and present a discussion of this well known game from a computing perspective. With the aid of sketches of the components, we indicate how the authors own NIM MACHINE functions in playing a perfect game of NIM against a mere human competitor. This model of NIM MACHINE required most of a Tinkertoy Giant Engineers Construction Set for its construction.

BOOLEAN ALGEBRA FOR BEGINNERS

Suppose one was dealing with quantities, x, y, z , etc, which are rather like ordinary numbers, but can only have values 0 and one. How could one proceed to define an arithmetic based on 0 and 1 only? Firstly lets look at multiplication. In ordinary arithmetic:

$$\begin{aligned}1.1 &= 1 \\ 1.0 &= 0 \\ 0.1 &= 0 \\ 0.0 &= 0\end{aligned}$$

The symbol $.$ is used above to indicate the multiplication operation. Clearly, if one starts with two quantities that are zero or unity, then in ordinary arithmetic the result of multiplication yields zero or one. So in this new arithmetic under exploration the old rules for multiplication are retained, the operation being termed the dot product to avoid any possible misunderstandings. The rules for the dot operation can be summed up in a table like so:

H.A. Cohen, *Making Algebra Concrete using a Microcomputer: An Algebraic Programming Language for the Control of Robots*, in P. Williamson (Editor), *Learn to Love Mathematics*, Mathematics Association of Victoria, Melbourne, 1980, pp 336-350. Preprint + Afterthought

x	y	x.y
1	1	1
1	0	0
0	1	0
0	0	0

Note the remarkable fact that $x.x = x$ for any x .

If multiplication is so easy, how can one define addition consistently? Clearly we can retain the rules of ordinary arithmetic that

$$1+0=0+1=1$$

$0+0=0$ but we must abandon the sacred $1+1=2$ rule.

There are in fact two choices:

$$1+1=0$$

$$1+1=1$$

In his 1854 book Boole took for the operation $+$, the first choice, with $1+1=0$. (The alert reader will notice that this particular algebra of Boole is simply modular two arithmetic). However Boole's noted successor W.S. Jevons used $+$ in this context so that $1+1=1$. Jevon's notation, NOT that of Boole himself, is used nowadays in those discussions of Boolean algebra that use the $+$ symbol. We will retain BOTH choices, and introduce TWO addition-like operations (and so compensate for losing $+$). He use as symbol for these operations a circled X like so, (\otimes) , and a \vee . The tables for these two operations are now presented for later reference.

x	y	$x\otimes y$	$x\vee y$
1	1	0	1
1	0	1	1
0	1	1	1
0	0	0	0

To complete this purely arithmetic account of Boolean algebra we must define one further

H.A. Cohen, *Making Algebra Concrete using a Microcomputer: An Algebraic Programming Language for the Control of Robots*, in P. Williamson (Editor), *Learn to Love Mathematics*, Mathematics Association of Victoria, Melbourne, 1980, pp 336-350. Preprint + Afterthought operator, the «*», through the rules:

$$\sim 1 = 0$$

$$\sim 0 = 1$$

The significance of the .> operation is highlighted by the trivial identities:

$$x \cdot \sim x = 0$$

$$x \vee \sim x = 1$$

$$x \vee \sim x = 1$$

These three little identities are most intelligible in terms of logic.

LOGIC

We now turn to giving the algebra introduced above a meaning in terms of logic. Logic deals with the truth and falsity of derivations from various premises.

The most basic idea in logic is that of the truth or falsity of some statement or other. In order to formalise the idea of logical truth logicians especially apply it to a carefully specified class of statements that are called propositions. A particular proposition, be it Maths is Fun. or whatever, can be TRUE or FALSE. Another way of talking about the truth/falsity of propositions is in terms of their truth value; if a proposition has truth value 1 then it is TRUE, whereas if it has truth value 0 it is false.

$$1 \rightarrow \text{TRUE}$$

$$0 \rightarrow \text{FALSE}$$

The link between logic and Boolean algebra is then provided by taking the truth value of a proposition to be an element (i.e., 0 or 1) of Boolean algebra. The proposition that is the denial of proposition X is called the negation of X, and is written as $\sim X$. If x is the truth value of X, then clearly the truth value of $\sim X$ is $\sim x$.

Suppose that by x we mean the truth value of some sentence (or more formally a propositional function) X, and likewise y is the truth value of the sentence/proposition Y. Then $X \cdot Y$ is the proposition that BOTH X and Y are true. Now $X \cdot Y$ can only be true if both X and Y are true, so it follows that $X \cdot Y$ has the truth value $x \cdot y$ as given by the table above. Now $X \cdot Y$ is often read as X AND Y., so that some writers adopt the most confusing term of phrase in calling the DOT operation logical addition.. A better terminology is to simply refer to the DOT operation.

The normal common language use of the conjunction or is often ambiguous. In logic two different logical OR relations are defined:

H.A. Cohen, *Making Algebra Concrete using a Microcomputer: An Algebraic Programming Language for the Control of Robots*, in P. Williamson (Editor), *Learn to Love Mathematics*, Mathematics Association of Victoria, Melbourne, 1980, pp 336-350. Preprint + Afterthought

$X \vee Y$: Either X OR Y ; INCLUSIVE OR

$X \times Y$: Either X OR Y But NOT both ; EXCLUSIVE OR

As $X \vee Y$ would be true if one or both of X or Y is true, i.e., the truth value of $X \vee Y$ is 1

when one or both of x and y are 1, it can be seen on examination that this truth value is given by the Boolean algebra construct $x \vee y$ of the table above. Likewise the truth value of $X \times Y$ is given by $x \times y$.

Thus Boolean algebra provides a representation of the basic operations of the logical calculus (and vice versa!). For this reason tables such as those above are commonly called truth tables.

The various Laws of Logic are propositional formulae that are TRUE for all propositions. Corresponding to such a formula of propositions is a formula of Boolean algebra that may be simply handled by algebraic rules to prove the logical law. For example there is the following law of Logic.

$X \vee Y$

Expressible in words very clumsily somehow like so:

Either X or Y is true is true if and only if it is false that NOT X AND NOT Y.

This law of Logic is known as de Morgan's Law or Theorem. In terms of the truth values of the corresponding propositions de Morgan's Law is the identity

$$x \vee y = \sim(\sim x \cdot \sim y)$$

To say that the above formula is an identity is to state that for any value of x and y, the value of $x \vee y$ and $\sim(\sim x \cdot \sim y)$ are the same. Hence a neat way of proving de Morgans Law is to simply tabulate the truth values of these quantities for all values of x and y, and to compare these four values of each quantity.

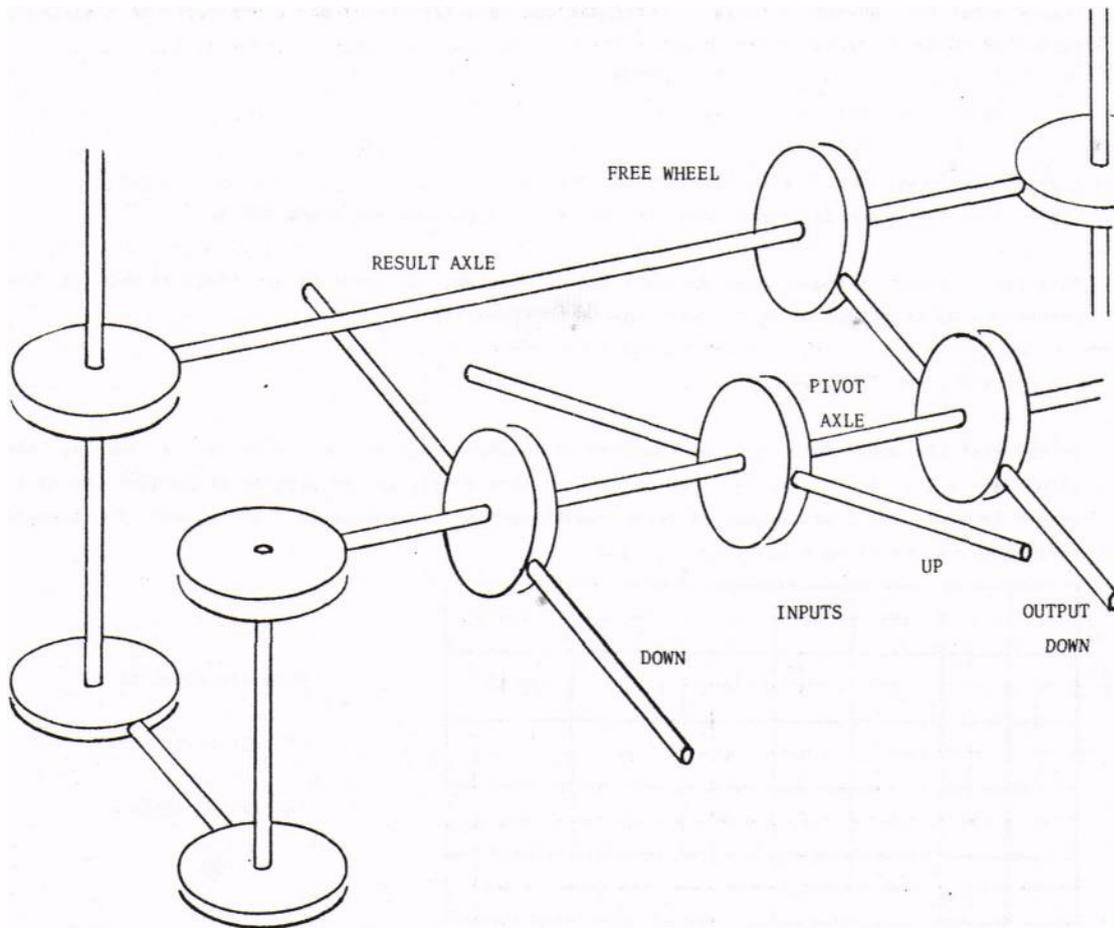
x	y	$x \vee y$	$\sim x$	$\sim y$	$\sim \sim x \cdot \sim y$	$\sim(\sim x \cdot \sim y)$
1	1	1	0	0	0	1
1	0	1	0	1	0	1
0	1	1	1	0	0	1
0	0	0	1	1	1	0

*

*

Note the equality of entries in the two marked columns

Suppose one models Boolean algebra with a mechanical device for which say, UP corresponds to 1, and DOWN corresponds to a truth value of 0. Then it is very easy to model a so-called inverter gate, a logical device with just one input and one output. If the input is x , the output is $\neg x$. That is in mechanical terms, if input is UP, output is DOWN, and viceversa. This is the law of the See-Saw.



AND/OR GATE OF TINKERTOY

H.A. Cohen, *Making Algebra Concrete using a Microcomputer: An Algebraic Programming Language for the Control of Robots*, in P. Williamson (Editor), *Learn to Love Mathematics*, Mathematics Association of Victoria, Melbourne, 1980, pp 336-350. Preprint + Afterthought

The AND gate is a logical device with two inputs, of truth value x and y say. The AND gate has a single output of truth value $x.y$. The reader is referred to the table above for the truth table of $x.y$. We translate this table into concrete terms as follows:

X	Y	X.Y
UP	UP	UP
UP	DOWN	DOWN
DOWN	UP	DOWN
DOWN	DOWN	DOWN

Using Tink'ertoy, a good model of the AND gate is supplied by the construction.

Note in the above model that the X and Y inputs are modelled by the UP/DOWN states of two rods. The output is also modeled by the UP/DOWN state of a third rod. To repeat the information supplied by the above table, for all three rods:

UP \leftrightarrow 1 \leftrightarrow TRUE

DOWN \leftrightarrow 0 \leftrightarrow FALSE

Now we turn to model an OR gate. We look at the truth table for $x \vee y$ above, and look again at the AND model for inspiration. One doesn't need a new model, the OR model will do quite nicely with reinterpretation of the X , Y , and $X \vee Y$ rod states:

UP \leftrightarrow 0 \leftrightarrow FALSE

DOWN \leftrightarrow 1 \leftrightarrow TRUE

That this one model can serve to model the two gates is a reflection of de Morgan's Law:

$$X.Y = \sim (\sim X \vee \sim Y)$$

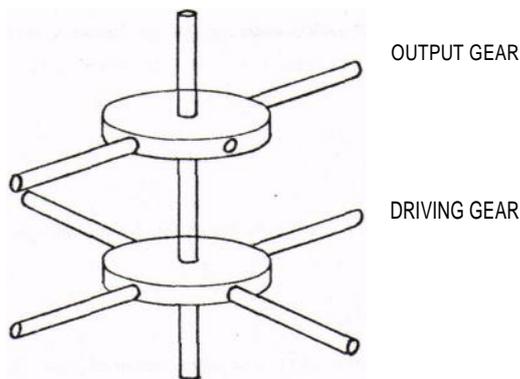
Likewise it is easy to devise models for the OR gate that are more natural, which also serve as models for the AND gate. Thus de Morgan's Law lends itself to discovery using purely concrete materials.

FLIP-FLOPS

Time does not really enter into the simpler logic gates, although it is clear that the output state is not achieved until some little time after the input states have settled down. However significant computation involves time in crucial ways. This is well shown in the various categories of FLIP-FLOPS which are used to construct registers, counters, adders in a computer. It is beyond the scope of this paper to detail all the varieties of flip-flops, but the general idea will be presented.

A flip-flop is a logic gate that has one input which if of logic value 1 changes the current output to its logical negation. Thus if there is a string of ones input, the output of the flip-flop is a string 0 1 0 1 0 1. However (especially in the JK flip-flop) there must be a means for setting the output, to initialise it. This is done in the JK flip-flop by using a second input, which determines that at the next state, the output will be flippable by the prime input, else will be set equal to the prime input.

The way that time enters into a flip-flop is that after the inputs have been set, there is another special input called clock that yields the output. This whole story is admittedly becoming complex to explain verbally. However central to any flip-flop logic gate is a capacity to flip-flop. In the Tinkertoy Manual there are plenty of examples of flip-flopping devices, for example a piston driver, for which the throws of the shaft are to alternate sides. About the simplest device that could serve the purpose would be this:



FLIP-FLOP SUB-ASSEMBLY

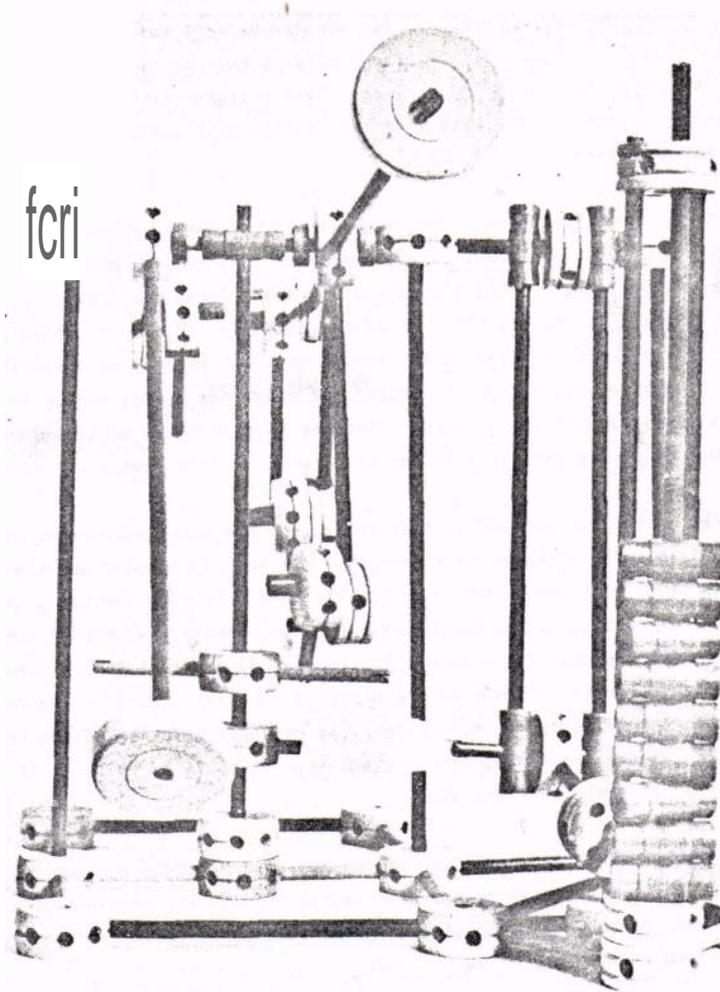
In this model, the top gear can represent the state of the output is alternatively 0 and 1. The bottom gear degree turn of the prime input. To complete the model device that gives just one thrust to one of the bottom

the output. So that for 90 degree turns has four spokes, so it can provide the 90 requires the input to take the form of a spokes.

H.A. Cohen, *Making Algebra Concrete using a Microcomputer: An Algebraic Programming Language for the Control of Robots*, in P. Williamson (Editor), *Learn to Love Mathematics*, Mathematics Association of Victoria, Melbourne, 1980, pp 336-350. Preprint + Afterthought

NIrt MACHINE

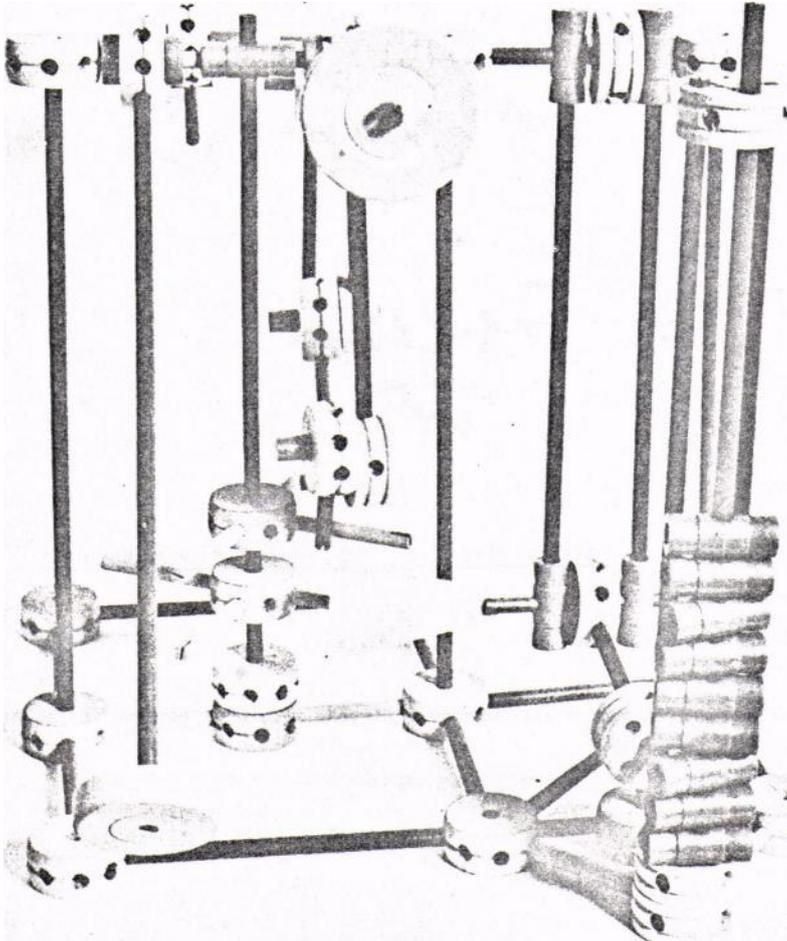
The player's arm is on the left, the machine's arm is in the middle. Both arms have on the end of the handle a large wooden wheel; these wheels serve as counterweights, and must be brought upwards to effect a move. To the right is the vertical stack of orange-jointing pieces; the bottom piece in the stack is only held in place by the weight of the pieces above, and is popped forward off the stack on a thrust from the striker rod. The player's arm is now parked. It is now the machine's turn. The player must give the machine arm just one complete stroke. In the picture the first finger of the machine's arm is just striking a spoke of the upper spoked gear. The corresponding spoke of the lower gear is just touching the flap of the ballistic pendulum, initiating a movement of the striker. The question to be asked at this stage is who/what is going to win this game?



H.A. Cohen, *Making Algebra Concrete using a Microcomputer: An Algebraic Programming Language for the Control of Robots*, in P. Williamson (Editor), *Learn to Love Mathematics*, Mathematics Association of Victoria, Melbourne, 1980, pp 336-350. Preprint + Afterthought

HIM MACHINE

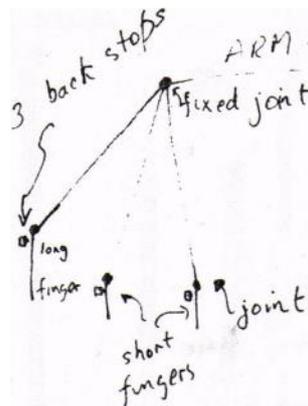
This photo is of an entirely different game with the NIM machine than that of the previous photo. All three finger's of the machine's hand passed through the gears and the second and third fingers have already been returned to the park side of the gears. At the conclusion of the machine's move the handle is being returned to the park position. The first finger of the machine's hand is about to strike a spoke of the top gear, but coming from this direction there is no thrust on the ballistic pendulum. The player makes his move by making up to three upwards strokes of the player arm; on each stroke an orange piece is pushed from the bottom of the stack (while on the return stroke there is no ejection). Considering that it is now the player's turn, can he play and win?



H.A. Cohen, *Making Algebra Concrete using a Microcomputer: An Algebraic Programming Language for the Control of Robots*, in P. Williamson (Editor), *Learn to Love Mathematics*, Mathematics Association of Victoria, Melbourne, 1980, pp 336-350. Preprint + Afterthought

REFERENCES

- (1) M.V. Wilkes, .How Babbage's dream came true., Nature Vol 25? Oct 16 1975 pp 541-4.
- (2) J.M. Dubbey, .The Mathematical Work of Charles Babbage., Cambridge Univ Press, 1978.
- (3) George Boole and others - extracts - in Part 13, .Mathematics and Logic. Vol 3 of James R. Neuman, .The World of Mathematics., George Allen and Unwin, 1960.
- (4) Article by Mickie in Clark (Editor), .Advances in Computer Chess., Edinburgh University Press, 1977. (Describes Quevedo's mechanical chess player).

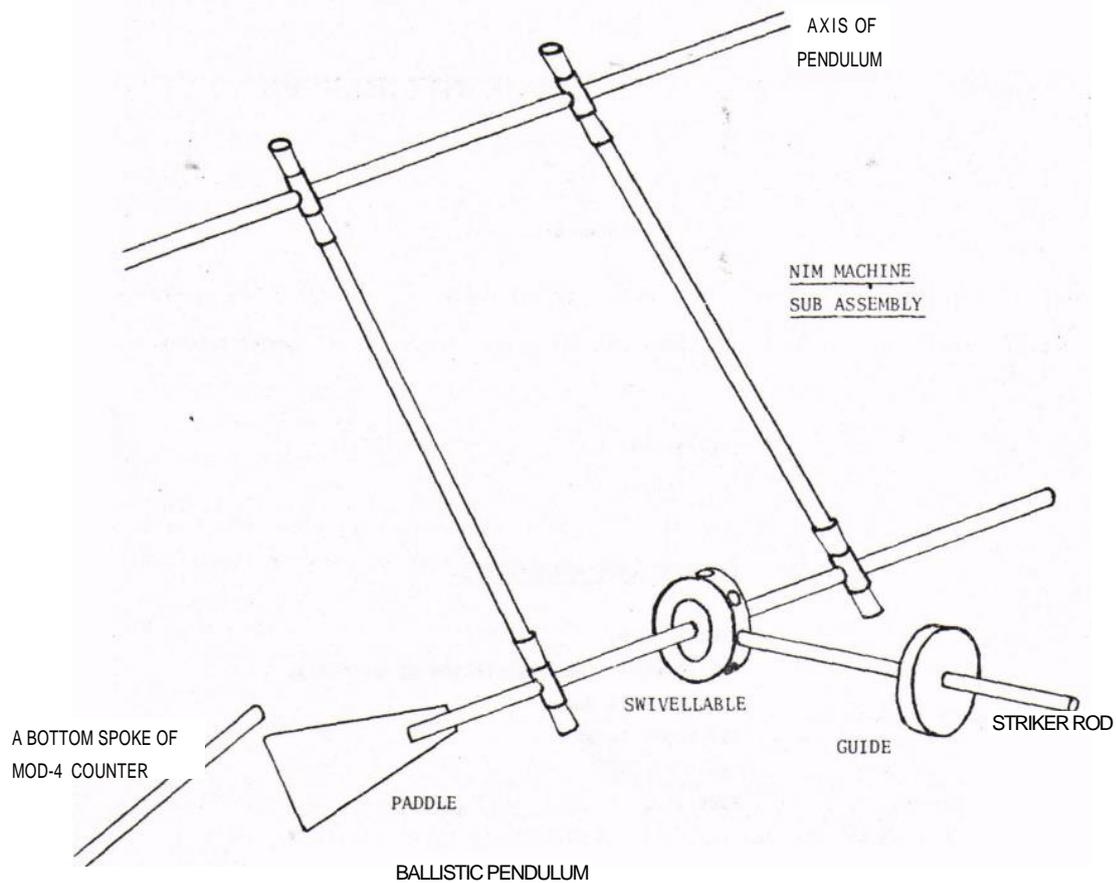


□

DESIGN SCHEME OF THE MACHINE'S PLAYING HAND

QUESTION : Which one of the machine s fingers should be longest for an arbitrary setting of heap size?

H.A. Cohen, *Making Algebra Concrete using a Microcomputer: An Algebraic Programming Language for the Control of Robots*, in P. Williamson (Editor), *Learn to Love Mathematics*, Mathematics Association of Victoria, Melbourne, 1980, pp 336-350. Preprint + Afterthought



In offering this presentation to the Mathematics Association of Victoria I give the following challenges to the younger mathematicians of Victoria:

- * Construct out of Tinkertoy or the like a machine that plays (and wins!) the Game of Eight.
- * Construct a simple adder out of mechanical flip-flops.
- * Construct a mechanical counterpart of a USART: a device that on receiving a strobe pulse (or its analogue) sends out a chain of pulses over some propagation network whose steps and troughs correspond to the static settings of a set of binary inputs. A USART must also be able to RECEIVE:; take in a chain of pulses and translate to static values.

H.A. Cohen, *Making Algebra Concrete using a Microcomputer: An Algebraic Programming Language for the Control of Robots*, in P. Williamson (Editor), *Learn to Love Mathematics*, Mathematics Association of Victoria, Melbourne, 1980, pp 336-350. Preprint + Afterthought

AFTERTHOUGHT

NOTES added to preprint. June 2005 (25 years after publication).

The NIM playing machine described is one in which there is a single player versus the machine, and a single heap of doovers - these being orange jointing pieces arranged in a column through which a Tinker Toy stick passes.

At his/her turn, the player uses the lever to the left to remove one two or three doovers from the heap at his/her discretion...

The machine's response is found by (manually) moving the Machine's handle - the handle to the right - when again - in accord with the built-in algorithm - the machine ejects one two or three doovers.

Shortage of space - and lack of any actual publications to cite in 1980 - made it impossible to properly explain the origins of the design for the NIM Playing machine presented above. In point of fact my initial model (of a NIM Playing Machine) was made in 1974, when I was a member of the LOGO Group, directed by Seymour Papert located within the MIT Artificial Intelligence Lab. At that stage, Danny Hillis, then an undergraduate at MIT, and others were constructing a Tic Tac Toe Playing machine out of Tinkertoy. (Daniel Hillis wrote his PhD on the Connection Machine - a massively parallel computer -- which was actually realized as a commercial Product). Hillis guided me in the use of Tinkertoy, and showed how a Mod 4 counter could be simply realized - the basis for this NIM machine.

Of course the emphasis in the paper above on **Making Algebra Concrete** was quite foreign to the atmosphere and objectives of the LOGO Group. However, again, to give due citation, I must acknowledge a huge intellectual debt to Radia Perlman, (another undergraduate at MIT in 1974) who produced a "Button Box Logo" for use by primary students, in which I could see a thorough going concretization of algebra.

The paper was published in

LEARN TO LOVE MATHEMATICS

Many years out of print which was published by
THE MATHEMATICAL ASSOCIATION OF VICTORIA
Clunies Ross House
191 Royal Parade
Parkville 3052
1980

Printed by
ACADIA PRESS PTY LTD
14 Blackburn Road
Blackburn 3130

ISBN 0 909315 04 3
