KNOT AUTOMATA

Louis H. Kauffman

Reprinted from THE TWENTY-FOURTH INTERNATIONAL SYMPOSIUM
ON MULTIPLE-VALUED LOGIC
Boston, Massachusetts, May 25 - 27, 1994

Knot Automata

Louis H. Kauffman
Dept. of Math., Stat. and Comp. Sci. (M/C 249)
851 South Morgan
The University of Illinois at Chicago
Chicago, Illinois 60607-7045

FAX: (312) 996-1491
e-mail: U10451@UICVM.BITNET

Abstract

This paper studies a mathematical model for automata as direct abstractions of digital circuitry. We give a rigorous model for distributed delays in terms of a precedence order of operations. The model is applied to automata that arise in the study of topological invariants of knots in three dimensional space and to digital design.

1 Introduction

In this paper we consider a class of automata that are direct abstractions of digital circuitry. A real digital circuit instantiates this structure into hardware. The circuits that are described here are a well defined class of abstract automata that include standard models for digital circuits and other examples such as the knot automata discussed herein.

The paper is organized as follows. Section 2 discusses the general concept of a circuit automaton with distributed delays. A rigorous model for states and transitions is formulated and the concept of a circuit that is determinate (behaviour independent of the value distribution of delays) is discussed. In section 3 we discuss a class of circuit automata that arise from the formal structure of the theory of knots and links in three dimensional space. We show how these automata and their properties can be used to find topological information, and we raise questions about the structure of these circuits as automata in their own right.

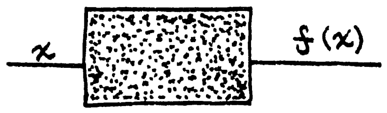
In section 4, we apply the concept of circuit automaton to those automata that are built from the abstract analogues of NOR gates. These automata contain important examples of digital circuit design, and we discuss one new design for a reductor (our term for the basic one-input one-output device that builds a flip flop and concatenates to build counting circuits). In the well formulated framework of the model we conjecture that a reductor requires at least six NOR gates. The proof of this conjecture appears to be a deep combinatorial question.

It gives the author pleasure to thank James Flagg and G. Spencer-Brown for helpful conversations, the National Science Foundation for support under NSF Grant Number DMS 9205277, and the Program for Mathematics and Molecular Biology of the University of California at Berkeley, Berkeley, CA for partial support during the preparation of this paper.

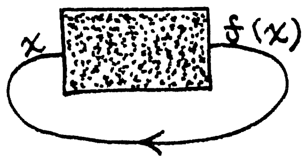
2 Automata With Distributed Delays

The automata considered in this paper are composed of building blocks with specified structure. Each block will have a specified set of input lines and a specified set of output lines, and a behaviour with respect to these inputs and outputs so that the outputs may be regarded as a well defined function of the inputs. The interconnection of such blocks in forms that include closed circuits produces automata whose behaviour can be interpreted. We shall give a model that defines states and transitions for such interconnections.

The purpose of this section is to give a general presentation of the point of view used in the rest of the paper. To this end, let $f : R \to R$ be a function defined on a set R. Diagram f as a black box with input line labelled x and output line labelled $f(x)$.



Now form the automaton obtained by connecting the output line of this box to its input line.



We say that *this automaton has the equation* $x = f(x)$, meaning that the input is taken to be the output. The equation $x = f(x)$ is taken to be a symbol or shorthand for the automaton that is obtained by taking the indicated feedback with a given delay. The symbolism does not contain the temporal behaviour, but that behaviour can be unfolded from this symbolism. Assume that there is a delay time Δt associated with the box so that a change in x results in a corresponding change in $f(x)$ only after time Δt. The temporal behaviour is then given by the equation $x(t + \Delta t) = f(x(t))$ so that $x(t + n\Delta t) = f^{(n)}(x(t))$ where $f^{(n)}$ denotes the composition of f with itself n times.

Consider the following example: $y = f(x)$, $x = g(y)$. We indicate the corresponding circuit by an interconnection of two boxes, one for f and one for g. The output of the f-box goes to the input of the g-box; the output of the g-box goes to the input of the f-box. Each box has its own delay time so that the structure of the transition may depend upon which time is the shorter.

The general model that sits behind these ideas is the following. Let the circuit automaton be defined by a system of a n equations.

1. $x_1 = f_1(x_1, \ldots, x_n, a_1, \ldots, a_m)$
2. $x_2 = f_2(x_1, \ldots, x_n, a_1, \ldots, a_m)$

...

n. $x_n = f_n(x_1, \ldots, x_n, a_1, \ldots, a_m)$

An *instantiation* of this automaton consists in a choice of permutation π of the numbers $12 \cdots n$ with $\pi(i)$ denoting the ith element of the permutation. With ths choice of permutation in hand, we define the action of the automaton on initial values $(x_1(0), \ldots, x_n(0), a_1, \ldots, a_m)$ as follows:

1. If $(X(i), A) = (x_1(i), \ldots, x_n(i), a_1, \ldots, a_m)$ satisfies the system of equations then $(X(i+1), A) = (X(i), A)$.

2. If $(X(i), A)$ does not satisfy the system of equations, then choose the least k such that the π_k-th equation is not satisfied. Define

   $$X\pi_k(i+1) = f_{\pi_k}(X_1(i), \ldots, X_n(i), a_1, \ldots, a_m)$$

   and

   $$X_s(i+1) = X_s(i) \text{ for } s \text{ not equal to } \pi_k.$$

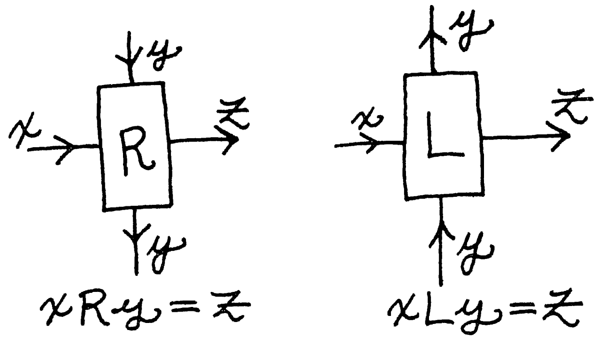
In other words, the automation resets the output value of the "first" unbalanced equation in the order chosen by the permutation.

In this instantiation, the choice of permutation determines the relative values of the delays in the different parts of the circuit. In many cases we are particularly interested in those circuit automata that have the same behaviour (with respect to given inputs and outputs) independent of the choice of permutation. Such a circuit is one that will behave the same way independent of the choice of distributed delay times.

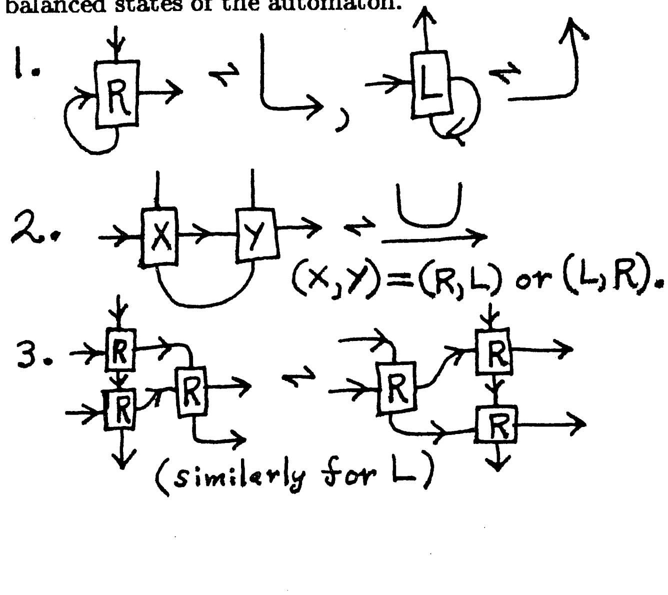
The program outlined above is used for examining the behaviour of a given automaton. In a computer simulation, one way to see the independence of delay choices is to have the computer change the permutation at random. If this has no apparent effect on the output behaviour of the automaton over a long run, it is good evidence that the automaton has the desired independence property. We have done such testing on the digital designs in section 4, and on other more complex designs that will be reported on elsewhere.

3 Knot Automata

This section considers a class of circuit automata that are based on the theory of knots and links in three dimensional space. The basic circuit element for these automata has an equation of the form $z = xRy$ or $z = xLy$ with box depictions as shown below. Note the orientations on the lines.

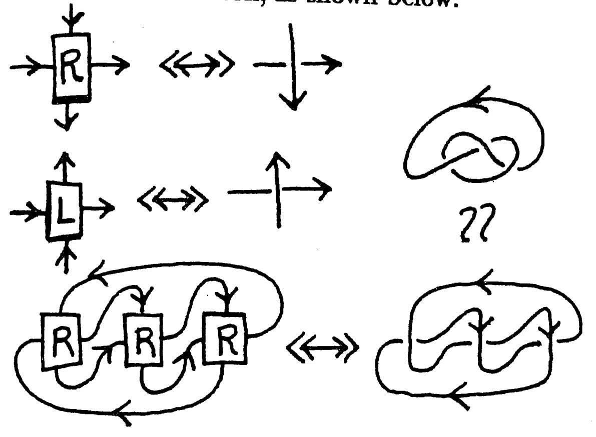


Here R and L denote the two types of operations, dependeing upon left and right orientations in the plane. The circuit box for $z = xRy$ is a box with inputs y and x and outputs y and z. The box is regarded as passing without processing it, the value of y, while it transforms x to $z = xRy$ by some, as yet unspecified, rule. In this way, the action of the box is dependent upon the y value, but its action does not affect this value. It is part of the rules of the game, that the circuit diagram for such an automaton must be drawn in the plane, and that it must satisfy the following diagrammatic exchanges without affecting the balanced states of the automaton.



This means that if a given automaton has a balanced state, then all the automata obtained from it by transformations as shown will also have balanced states. By examining properties of the states of two given automata it is often possible to show that there is no sequence of transformations from one of them to the other, due to differences in particularities of the states.

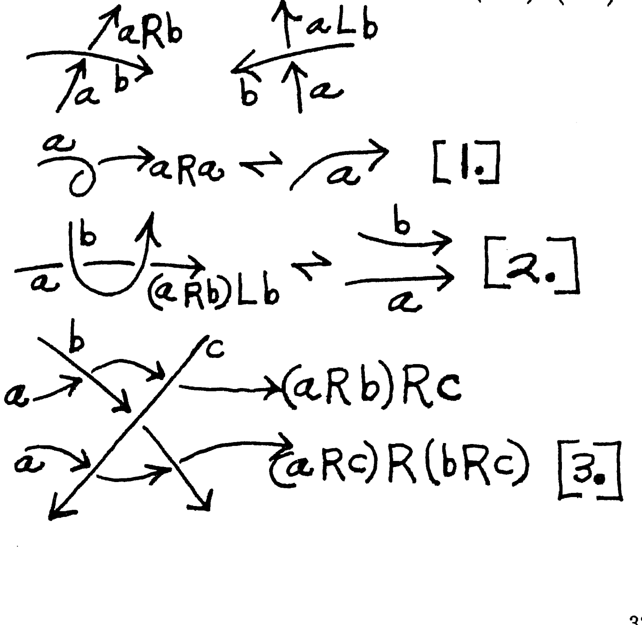
A sequence of transformations from one of these structures to another has a topological interpretation: It is possible to associate a diagram for a knot or a link to each automaton, as shown below.



In this way the transformations that we have indicated become topological transformations of the diagrams, and these three types of transformation are known to generate all possible topological transformations of knots and links in three dimensional space (See [16], [8], [11]).

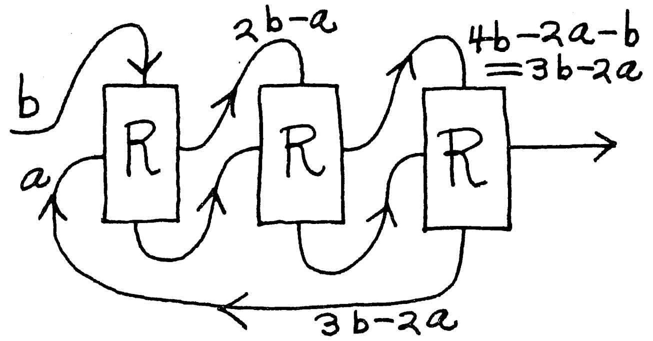
Returning to the automata, the three moves translate into the demands

1. $aRa = a$, $aLa = a$
2. $(aRb)Lb = a$, $(aLb)Rb = a$
3. $(aRb)Rc = (aRc)R(bRc)$, $(aLb)Lc = (aLc)L(bLc)$



The second and the third are the most significant demands, asking that the operations R_b and L_b are invertible and inverses of each other for any b, and that the operations R and L are self-distributive. The resulting algebraic structure is called a *quandle* (See [5], [2], [3]).

For our purposes, the simplest example of a quandle is the structure $aRb = aLb = 2b - a$ where a and b are elements of an additive abelian group. Consider the trefoil knot automata with this local structure:



$$a = 3b - 2a$$
$$\Rightarrow 3(b-a) = 0.$$

We see that the feedback loop in this automaton forces the conclusion that $3(b-a) = 0$ for any a and b in the group. Hence 3 must divide the order of this group in order for the trefoil automaton to have any balanced states. If we take the group to be $G = Z/3Z$, then we can analyze the resulting balanced states and show that there is no possible sequence of transformations of the trefoil automaton to the unknot automaton shown below. Thus *the knottedness of the trefoil can be seen to be a consequence of using a three valued logic in the signals of an automaton associated with the diagram of the knot.*

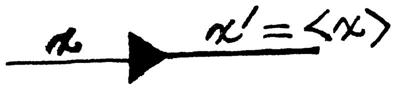
It remains to be seen how the transition behaviour of these automata is related to the topology.

In general, each such automaton has a well-defined modulus, so that its states can be described in terms of values in the modular arithmetic Z/NZ. For this reason, this class of automata is interesting for multiple valued logic since each automaton has its own naturally associated set of values, and this set is invariant under the transformations that we have indicated.

There are further aspects of this situation that are of particular interest to topologists and algebraists. For example, if the values for the knot automaton lie in a module over the ring $Z[t, t^{-1}]$, then the following algebraic rules satisfy the axioms given above for a quandle: $aRb = ta + (1-t)b$, $aLb = sa + (1-s)b$ with $s = t^{-1}$. We have discussed the case with $t = -1$. In the general case, the modulus is a Laurent polynomial in t, and is known as the Alexander polynomial [11] of the knot and link. Stronger invariants of the structural transformations of these circuits arise through the use of such algebras and moduli.

4 The Digital Circuit Model

The basic digital element discussed here is an inverter, diagrammed as shown below.

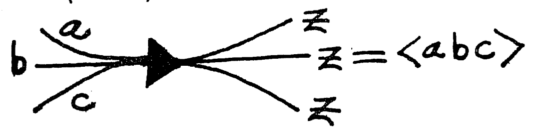


We use two valued logic with values 0 and 1. We take $0' = 1$ and $1' = 0$, $00 = 0$, $01 = 10 = 0$, $11 = 1$. This operation of juxtaposition $(a, b \rightarrow ab)$ can be interpreted as logical "or" for the interpretation of 0 as the value True. With more than one input the inverter becomes a NOR gate: $a, b, c, \ldots \rightarrow (abc\cdots)'$.

All models in this section can be generalized to multiple-valued logic, but we concentrate here on the special case of two values. See example 4.2 for a circuit that requires multiple values.

Notation

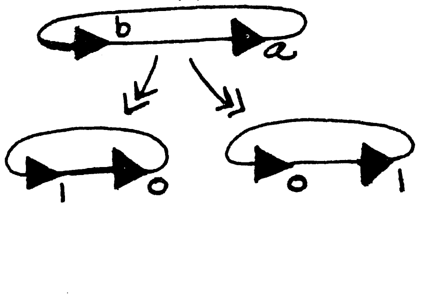
Let $\langle abc\cdots\rangle$ denote $(abc\cdots)'$.



In a circuit diagram, a *state* is a coloring of the arcs that start from one inverter's output and terminate at another inverter's input. The colors are chosen from the set $\{0, 1\}$. *All arcs emanating from a given inverter are colored identically in a given state.* (In this model an inverter has only one output value in any given state.)

As a consequence of this stipulation we can write a single equation that describes the action of a given inverter in the circuit. Let z denote the label for the outgoing lines of the inverter. Let $a, b, c, \ldots$ denote the labels of its ingoing lines. Then $z = (abc\cdots)' = \langle abc\cdots\rangle$ (see the notational remark above) is the equation describing the action of the inverter. In a given state these equations may not be satisfied at some places in the circuit.

A state is said to be *balanced* if the equation $z = \langle abc\cdots\rangle$ is satisfied at every inverter in the diagram. Here $z = \langle abc\cdots\rangle$ denotes the equation that defines the operation of the given inverter. Thus in the circuit below the balanced states are choices of values for a and b such that $b = \langle a\rangle$ and $a = \langle b\rangle$.

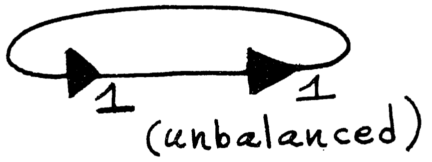


This circuit has exactly two balanced states: $a = 0$, $b = 1$ and $a = 1$, $b = 0$.

If S is an unbalanced state of a circuit C, then there will be one or more equations of the form $z = \langle abc\cdots\rangle$ that are not satisfied by the coloring. A *transition* consists in reassigning the value of z for the outgoing arcs z of one inverter at which there is an imbalance. The new state achieved by the transition may or may not itself be balanced. Transitions are formalized according to the schema of section 2.

4.1 Example

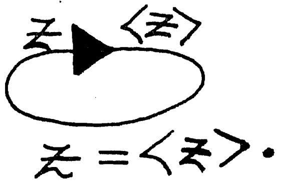
In the circuit below there are two possible transitions: $a = 1, b = 1 \rightarrow a = 1, b = 0$ and $a = 1, b = 1 \rightarrow a = 0, b = 1$. The states that result from this transition are both balanced. Call this circuit a *memory*. It has the equations $a = \langle b\rangle$, $b = \langle a\rangle$.



4.2 Example

In the circuit below there is one possible transition $a = 1 \rightarrow a = 0$, but the resulting state is not balanced, and its transition $a = 0 \rightarrow a = 1$ returns the circuit to its original state.

This circuit has the equation $z = \langle z\rangle$, for which there are no Boolean solutions.



The circuit $z = \langle z\rangle$ embodies the Liar paradox. If $z = 0$ then $z = 1$. If $z = 1$, then $z = 0$. Its behaviour is an oscillation between 0 and 1.

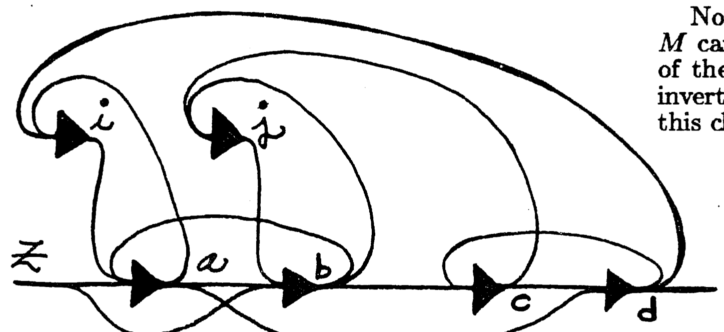
Circuits of this type are analogous to the knot automata of section 1 in that they demand a multiple valued logic in order to achieve stable states.

We are interested in designing circuits with specified *behaviours*. The behaviour of a circuit consists in a summary of its circuit action - what balanced states it can achieve from a given set of unbalanced states that are relevant to the design problem. In this regard we say that a circuit action is *determinate* if it has only one possible end state independent of the possible sequences of transitions that may lead to this end state. Thus we can ask of a given unbalanced state whether the resulting circuit action is determinate. In the first example above the action is not determinate. In the second example the action is determinate, but the set of possible balanced end-states is empty.

4.3 Example

The equations for this automaton, M, are

$$\begin{aligned} a &= \langle biz\rangle \\ b &= \langle ajz\rangle \\ c &= \langle bd\rangle \\ d &= \langle ac\rangle \\ i &= \langle ad\rangle \\ j &= \langle bc\rangle \end{aligned}$$



Here we regard z as an input to the system. For each value of z there are two balanced states of M. If $z = 0$, then $V = (a,b,c,d,i,j) = A$ or C where $A = (1,1,0,1,0,1)$ and $C = (1,1,1,0,1,0)$. If $z = 1$, then $V = B$ or D where and $B = (1,0,1,0,1,1)$ and $D = (0,1,0,1,1,1)$. One can then verify that for a given value of z and balanced state S, the transition that ensues upon changing z (from 0 to 1 or from 1 to zero) is determinate. The result is that the sequence of values $z = 0,1,0,1,0,1,\ldots$ results in the sequence of states $A,B,C,D,A,B,C,D,\ldots$ (Assuming that we start with $z = 0$ in state A.).

We assume that each change in z is held fixed long enough for the automaton to accomplish its transition to the next state. Transitions are accomplished according to the algorithm explained in section 2. This means that the model assumes delays associated with each inverter. There are no delays associated with the connecting lines in the graph. This method of distributing the delays is a mathematical abstraction, but it is sufficiently realistic so that these circuits can actually work at the hardware level. If the automaton is mathematically determinate (as in this example), then it will behave in the same way for any choice of actual delays - so long as the input varies more slowly than the time needed for internal transition.

The circuit in this example converts an input oscillation $z : 010101\cdots$ to internal oscillations of twice the period. For example we have in the above state sequence $d : 100110011001100\cdots$. By taking d as an output, we therefore obtain a black box B with input line z and output line d with this behaviour. This is exactly the behaviour needed to make circuits that count in binary. A series connection of n such black boxes produces an automaton that cycles through 2^{n+1} distinct states as the input z oscillates between 0 and 1.

Discussion

Note the basic behaviour of the black box B. *If z changes from 0 to 1 then the output d changes its value. If z changes from 1 to 0, then the output d does not change its value.* Call a determinate automaton with this behaviour (or the corresponding behaviour with 0 and 1 interchanged, and also the possibility of starting with z and d the same value) a **reductor**.

Note that the number of leads in the automaton M can be read from its equations by making a chart of the inverters (labelled a,b,c,d,i,j) to which each inverter or input is connected. For our automaton M this chart takes the form

z	:	ab
a	:	bdi
b	:	acj
c	:	dj
d	:	ci
i	:	a
j	:	b.

Here each line in the chart is of the form

R: List of inverters to which R is connected

where R is either an inverter or an input (z). The number of leads (14) is the number of letters occurring after the colons in this chart.

Thus we have a notion of the complexity of a reductor in terms of the number of inverters and the number of leads. We shall say that M is of type (6, 14), meaning that it has 6 inverters and 14 leads. Until recently I had thought that this design, which I discovered in 1978, was the reductor of minimal complexity. However, G. Spencer-Brown informed me in the Fall of 1992 that he has found a reductor of type (6, 13) [18]. It may be that (6, 13) is the true minimum for this design. I conjecture this to be the case.

A more general conjecture is the following.

Conjecture

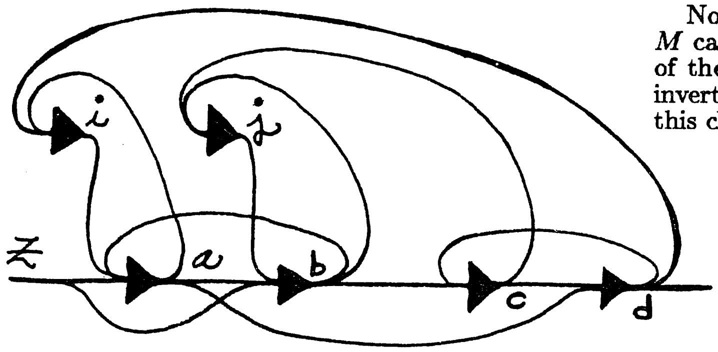
It is not possible to make a determinate (asynchronous) reductor with less than six inverters.

The designs in common use such as the asynchronous JK flip flop tend to use more inverters (NOR gates or NAND gates) and more leads. The least number of inverters in a published asynchronous flip flop design that I have encountered is nine (See [6]). It is quite possible that the combinatorics underlying this conjecture will be illuminated by considering its generalizations in asynchronous multiple valued logic design (Compare [4]).

4.3 Example

The equations for this automaton, M, are

$$\begin{aligned} a &= \langle biz \rangle \\ b &= \langle ajz \rangle \\ c &= \langle bd \rangle \\ d &= \langle ac \rangle \\ i &= \langle ad \rangle \\ j &= \langle bc \rangle \end{aligned}$$



Here we regard z as an input to the system. For each value of z there are two balanced states of M. If $z = 0$, then $V = (a,b,c,d,i,j) = A$ or C where $A = (1,1,0,1,0,1)$ and $C = (1,1,1,0,1,0)$. If $z = 1$, then $V = B$ or D where and $B = (1,0,1,0,1,1)$ and $D = (0,1,0,1,1,1)$. One can then verify that for a given value of z and balanced state S, the transition that ensues upon changing z (from 0 to 1 or from 1 to zero) is determinate. The result is that the sequence of values $z = 0,1,0,1,0,1,\ldots$ results in the sequence of states $A,B,C,D,A,B,C,D,\ldots$ (Assuming that we start with $z = 0$ in state A.).

We assume that each change in z is held fixed long enough for the automaton to accomplish its transition to the next state. Transitions are accomplished according to the algorithm explained in section 2. This means that the model assumes delays associated with each inverter. There are no delays associated with the connecting lines in the graph. This method of distributing the delays is a mathematical abstraction, but it is sufficiently realistic so that these circuits can actually work at the hardware level. If the automaton is mathematically determinate (as in this example), then it will behave in the same way for any choice of actual delays - so long as the input varies more slowly than the time needed for internal transition.

The circuit in this example converts an input oscillation $z : 010101\cdots$ to internal oscillations of twice the period. For example we have in the above state sequence d : $100110011001100\cdots$. By taking d as an output, we therefore obtain a black box B with input line z and output line d with this behaviour. This is exactly the behaviour needed to make circuits that count in binary. A series connection of n such black boxes produces an automaton that cycles through 2^{n+1} distinct states as the input z oscillates between 0 and 1.

Discussion

Note the basic behaviour of the black box B. *If z changes from 0 to 1 then the output d changes its value. If z changes from 1 to 0, then the output d does not change its value.* Call a determinate automaton with this behaviour (or the corresponding behaviour with 0 and 1 interchanged, and also the possibility of starting with z and d the same value) a **reductor**.

Note that the number of leads in the automaton M can be read from its equations by making a chart of the inverters (labelled a, b, c, d, i, j) to which each inverter or input is connected. For our automaton M this chart takes the form

z	:	ab
a	:	bdi
b	:	acj
c	:	dj
d	:	ci
i	:	a
j	:	b.

Here each line in the chart is of the form

R: List of inverters to which R is connected

where R is either an inverter or an input (z). The number of leads (14) is the number of letters occurring after the colons in this chart.

Thus we have a notion of the complexity of a reductor in terms of the number of inverters and the number of leads. We shall say that M is of type (6, 14), meaning that it has 6 inverters and 14 leads. Until recently I had thought that this design, which I discovered in 1978, was the reductor of minimal complexity. However, G. Spencer-Brown informed me in the Fall of 1992 that he has found a reductor of type (6, 13) [18]. It may be that (6, 13) is the true minimum for this design. I conjecture this to be the case.

A more general conjecture is the following.

Conjecture

It is not possible to make a determinate (asynchronous) reductor with less than six inverters.

The designs in common use such as the asynchronous JK flip flop tend to use more inverters (NOR gates or NAND gates) and more leads. The least number of inverters in a published asynchronous flip flop design that I have encountered is nine (See [6]). It is quite possible that the combinatorics underlying this conjecture will be illuminated by considering its generalizations in asynchronous multiple valued logic design (Compare [4]).

Summary

This paper has consisted in a rigorous formulation of the concept of a circuit automaton with examples from the theory of digital circuits and from the theory of knots and links in three dimensional space. In the case of knot theory we have reformulated the structure of the quandle [5] as a circuit automaton, and this leads to new and as-yet-unexplored questions about the behaviour of this class of automata. The knot theoretic automata are an example of a class of automata where each given automaton has an optimal multiple valued logic associated with its operation.

References

[1] P. Aczel, The Theory of Non-Well Founded Sets, CSLI Lecture Notes #14.

[2] E. Brieskorn, Automorphic sets and braids and singularities. Proceedings of Santa Cruz Conf. on Artin Braid Group. contemp. Math. *AMS* **78** (1998), pp. 45-115.

[3] P. DeHornoy, "Free distributive groupoids," *J. Pure and Applied Algebra,* **16** (1989), pp. 123-146.

[4] G. Epstein, *Multiple Valued Logic Design: An Introduction,* IOP Publishing, Bristol (1993).

[5] D. Joyce, "A Classifying Invariant of Knots, The Knot Quandle," *J. Pure and Appl. Alg.* **23** (1983), 37-65.

[6] J.R. Gibson, *Electronic Logic Circuits,* Edward Arnold Pub. (1992).

[7] L.H. Kauffman, *Formal Knot Theory,* Princeton University Press Mathematical Notes #30 (1983).

[8] L.H. Kauffman, *On Knots,* Annals Study **115**, Princeton University Press (1987).

[9] L.H. Kauffman, "State Models and the Jones Polynomial," *Topology,* **26** (1987), 395-407.

[10] L.H. Kauffman, "Statistical Mechanics and the Jones Polynomial," *AMS Contemp. Math.* Series (Proceedings of 1986 Conference on the Artin Braid Group - Santa Cruz, CA.) **78** (1989), 263-297.

[11] L.H. Kauffman, *Knots and Physics,* World Sci. Pub. (1991).

[12] L.H. Kauffman, Imaginary Values in Mathematical Logic, 17th International Symposium on Multiple-Valued Logic, IEEE Pub. (1987).

[13] L.H. Kauffman, "Self-reference and recursive forms," *J. Soc. Bio. Strs.* (1987), #10, pp. 53-72.

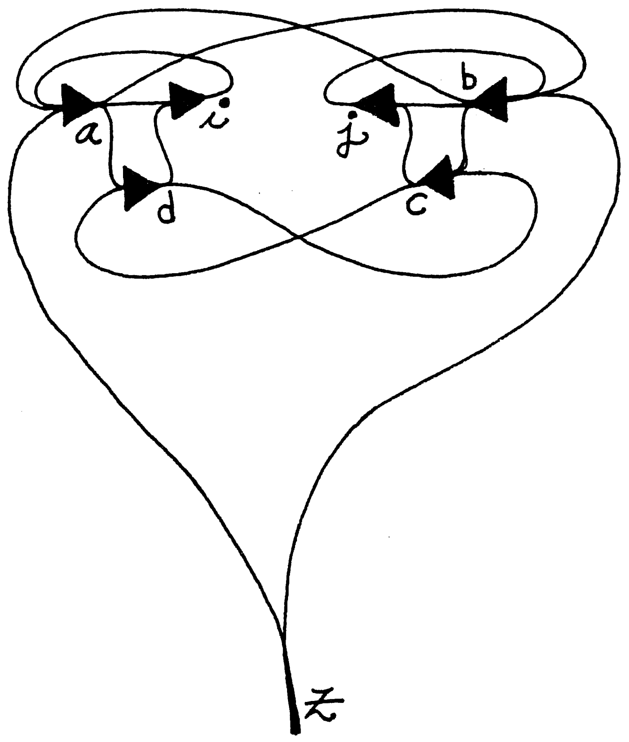
[14] L.H. Kauffman and F.J. Varela, "Form dynamics," *J. Social and Bio. Struct.* (1980), vol. 3, pp. 171-206.

[15] L.H. Kauffman, "Network Synthesis and Varela's calculus," *Int. J. Gen. Syst.* **4** (1978), pp. 179-187.

[16] K. Reidemeister, *Knotentheorie,* (1932), Julius Springer, Berlin, republished by Chelsea (1948).

[17] G. Spencer-Brown, *Laws of Form,* George Allen and Unwin Ltd., London (1969).

[18] G. Spencer-Brown, Private Communication (1992).



An Equivalent Form of Example 4.3.

Figure 1