A Truth-Matrix view into Unary Computing

Advait Madhavan
National Institute of Standards and Technology & University of Maryland
advait.madhavan@nist.gov

Georgios Tzimpragos
University of California Santa Barbara
gtzimpragos@cs.ucsb.edu

Mark Stiles
National Institute of Standards and Technology
mark.stiles@nist.gov

Timothy Sherwood
University of California Santa Barbara
sherwood@cs.ucsb.edu

***Abstract*—Our community has been exploring Time-of-arrival based codes as a candidate for very low energy information processing [3], [1]. A "space-time" algebra has been recently proposed that captures the essential features of such a paradigm [2]. In order to gain some insight into the behaviour of such a new representation, we propose using temporal equivalents of conventional Boolean truth tables. We use the basics of "space-time algebra" as a launching point, and re-examine key ideas such as normalization, invariance, non-prescience and causality, as well as the behaviour of operators from the perspective of these temporal truth matrices. We stress on the importance of coincidence and use these tables to provide a visual understanding of why coincidence ends up being an essential feature of universality in this new paradigm. We end with a simple example of how coincidence can be used to perform edge detection on images and compare it with classical edge detectors.**

I. Introduction

Karnaugh maps and truth tables play an essential role in the Boolean world, providing a visual understanding of how the basic operators of Boolean algebra work. As we make our first steps into the temporal world, a similar notion of truth tables can be constructed. In this paper, we describe how such truth matrices can give us insight into the operations of temporally coded, unary computers. But first we describe how to interpret numeric values, in this new "space-time" paradigm.

In our temporal world, since information in encoded in the arrival time of events, each wire can encode multiple logic levels. As opposed to the Boolean logic case in which the information encoding on a wire is 2 level, taken from the $[0, 1]$ set, space-time algebra can be thought of as a multi-valued logic with $k + 2$ levels, taken from $[0, 1, 2....., k, \infty]$ set. Each element of the set represents a discrete time step at which an event arrives on that wire, where ∞ means that the time at which the wire fires is too late to affect any computation.

	0	1	2	...	k	∞
0	$X_{(0,0)}$	$X_{(0,1)}$	$X_{(0,2)}$	...	$X_{(0,k)}$	$X_{(0,\infty)}$
1	$X_{(1,0)}$	$X_{(1,1)}$	$X_{(1,2)}$	...	$X_{(1,k)}$	$X_{(1,\infty)}$
2	$X_{(2,0)}$	$X_{(2,1)}$	$X_{(2,2)}$	...	$X_{(2,k)}$	$X_{(2,\infty)}$
...	...	...	...	...	...	...
k	$X_{(k,0)}$	$X_{(k,1)}$	$X_{(k,2)}$	...	$X_{(k,k)}$	$X_{(k,\infty)}$
∞	$X_{(\infty,0)}$	$X_{(\infty,1)}$	$X_{(\infty,2)}$	...	$X_{(\infty,k)}$	$X_{(\infty,\infty)}$

Fig. 1. General Truth matrix for a , k+2 level, 2 input space-time function.

Since we are working with a $(k + 2)$-valued logic family, for a two input function, we have $(k+2)^2$ entries and hence it becomes more convenient to represent it as a truth matrix, as shown in Figure 1, as opposed to a Boolean four entry truth table. Such a two-input truth matrix, though simple, is enough to demonstrate some important ideas and hence will be used as a foundation for further exploration.

Each value in this truth matrix is indexed by the times of arrival of its two inputs, which are taken from the finite set $[0, 1, 2....., k, \infty]$, defined as $\mathrm{V_k}$. The values $X_{(i,j)}$ themselves are also drawn from $\mathrm{V_k}$, but represent times at which the output of the function triggers. Again drawing a direct comparison with Boolean logic, in which there are 2^{2^2}, two input functions, the multi-valued formulation of space-time algebra has a total of $(k+2)^{(k+2)^2}$ functions. But unlike the Boolean domain where all of the input functions can be implemented with switches, the way information is encoded in such a temporal domain, makes some functions impossible to implement. It turns out that space-time functions can only implement a subset of the total functions, and specifically those which are constrained by the properties of the passage of physical time.

II. Truth Matrices and space time algebra

As far as we know, all physical systems have to obey the layman's notion of causation. It seems painfully obvious to state that the output of a system is affected by its inputs and hence happens after them. But in a domain where the information is itself encoded in time, this property constrains the values that the outputs of a function can take. Two "1"s as inputs causing an output "0" makes sense in the Boolean domain(last entry in the XOR truth table) but in the temporal domain, it implies acausal behaviour of the system (outputs happen before the input). This constraining of the space of "space-time" functions was first described in [2]. The author decomposes this layman's notion of causation into well defined properties of *invariance* and *causality*. Though we use *invariance* as it is, we introduce a modification of *causality* defined as *non-prescience* and explore these properties with truth matrices.

	0	1	2	...	k	∞
0	$X_{(0,0)}$	$X_{(0,1)}$	$X_{(0,2)}$	...	$X_{(0,k)}$	$X_{(0,\infty)}$
1	$X_{(1,0)}$	$X_{(0,0)}+1$	$X_{(0,1)}+1$	...	$X_{(0,k-1)}+1$	$X_{(0,k)}+1$
2	$X_{(2,0)}$	$X_{(1,0)}+1$	$X_{(0,0)}+2$	...	$X_{(0,k-2)}+2$	$X_{(0,k-1)}+2$
...	...	...	...	...	...	...
k	$X_{(k,0)}$	$X_{(k-1,0)}+1$	$X_{(k-2,0)}+2$	...	$X_{(0,0)}+k$	$X_{(0,1)}+k$
∞	$X_{(\infty,0)}$	$X_{(k,0)}+1$	$X_{(k-1,0)}+2$	...	$X_{(1,0)}+k$	$X_{(0,0)}+\infty$

Fig. 2. General Truth matrix for a, k+2 level, 2 input space-time function, constrained by invariance. Note the diagonal relationships between the elements of the matrix.

A. Invariance:

A function is defined to be time invariant if its behaviour is independent of "when" it is used. In other words, if one is to shift all of the inputs in time by a constant amount, the output will shift by that same amount[2]. More formally, a function $f\colon V_k^n \rightarrow V_k$ is invariant if and only if $\forall c \in \mathrm{V_k}$:

$$f(i_0 + c, i_1 + c, \ldots, i_n + c) = f(i_0, i_1, \ldots, i_n) + c$$

This definition of invariance says something about the structure of all invariant truth matrices as shown in figure 2. Note that column 0 and row 0 of the matrix taken together determine fully all of the other values. *The reason for this is that, as one varies* c *in the above definition of invariance, one relates entries diagonally across the matrix.* This holds for all values of c, hence fully specifying all of the entries along that diagonal.

This results in an important property of space-time functions – *their projective nature*. The value of any element in the matrix can be determined by tracing its projection along the diagonal back to the surface. Note that elements on the surface have the important property of at-least one of its input values being equal to zero. This is the reason for the notion of normalization is described in [2]. Hence, all normalized function tables described in [2] lie on the surface of a higher dimensional truth matrix. It is instructive to visualize moving along the diagonal as following the passage of time, while after the arrival of the first input, our movement is constrained along the surface of the truth matrix. Constraints along the surface are described by non-prescience and are discussed in the next section.

The reason behind using the word "surface" is to note that this property extends to diagonals of n dimensional truth matrices, and that the $n-1$ dimensional "surface" of that matrix (e.g. for a cube it is the three plane where $x = 0$, $y = 0$, and $z = 0$) fully defines the "interior" of the truth matrix. Hence, while there are $(k+2)^{(k+2)^2}$ possible 2-input functions, only $(k+2)^{2k+3}$ of them are invariant.

Finally, an important ramification of invariance is that no output will ever be triggered before at least one input signal arrives. This can be noted by looking at the bottom right element in Figure 2. Note that this also implies that the smallest value that the output can take has to be the *min* of the inputs.

B. Non-prescience:

Non-prescience captures the property that a function cannot use "future" information to decide what output to select. Specifically, if we are observing the operation of a non-prescient function at time step t_i, it's output at this time step cannot depend upon inputs that become available at later time steps. Unlike invariance which relates elements along the diagonal, non-prescience is more subtle and constrains elements along the surface of the truth matrix.

To describe this property we take snapshots of a regular truth matrix at each time step t_0, t_1, t_2 and t_3, and label them as C_0, C_1, C_2 and C_3 respectively. This is shown in figure 3.

Let us analyze the structure of the first snapshot of the truth matrix, namely C_0. It has 3 colored cells and 1 white cell. The white cell represents the output whose inputs are completely determined at the time step in question, namely t_0. The colored cells on the other hand represent outputs for which either one of the two inputs (represented yellow and orange cells), or both inputs (represented by the green cells), have not arrived yet. The key insight about non-prescience here is that, *if any element of the colored groups take the value "0", then all the elements of that group have to take the value "0"*. Let us use the elements of the orange group (a_i) for discussion. The case that any of the values of a_i are "0", and a_i depends on the column input leads to a contradiction, since it implies that a current output value depends on a future input, hence breaking non-prescience. Therefore, the value could have depended only on the row input and hence if one value is zero, then all values have to be zero. This can be extended without loss of generality to an arbitrary time step t_i and matrix C_i. Therefore, even though each element in the non-prescient surface of a particular truth matrix can take $k+2$ values, some values end up constraining others in the aforementioned way. *This results in non-prescient functions having the property of ending in a run of constants.*

Hence combining notions of invariance and non-prescience, we arrive at a similar definition to the one proposed in [2] where functions that are both invariant and non-prescient, are causal functions, and can be implemented in the space-time paradigm. Hence, a generic 2 input "space-time" function is shown in 4(f). Such a function has a particular form in the truth table. It can be seen that invariance limits the region of interest to the surface while non-prescience ensures that the entries end in a run of constant values.

Though such a definition may be more round-about than the one defined in [2], it calls for explicit evaluation of the function behavior at each time step. When

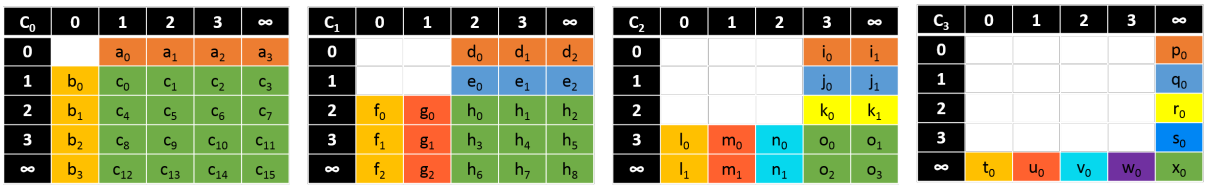

Fig. 3. Snapshots of an non-prescient truth matrices for a, k+2 level, 2 input space time function at various point in time, namely t_0, t_1, t_2, t_3 respectively.

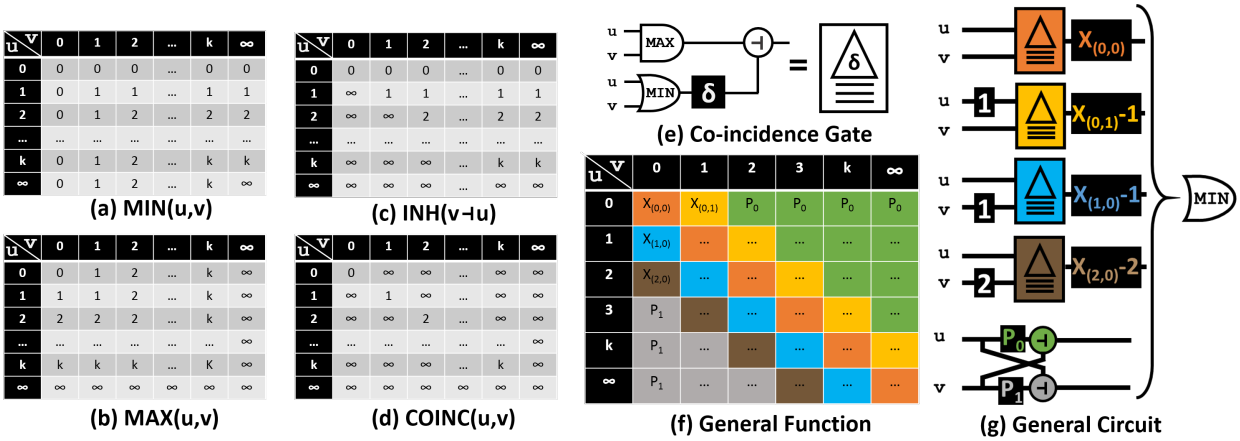

Fig. 4. Space time Operators and General construction: Panels (a-d) represent the MIN, MAX, INHIBIT and CO-INCIDENCE operators respectively. Panel (e) shows the co-incidence circuit(with a δ window) constructed out of MIN, MAX and INHIBT gates, with the black box being the delay. Panel (f) shows the truth matrix of an arbitrary space time function which obeys both invariance and non-prescience. The equivalent circuit diagram is shown in panel (g). The regions of the circuit that correspond to the regions in the truth matrix are color coded correspondingly.

combined with the invariance property, this allows for enumeration of the functional space, by using series sums and counting arguments. The analytic forms involve Pochhamer series and Holonomic functions and hence have not been included here.

C. Operators

Now that we have described the properties of "space-time" functions, let us look at some familiar 2-input space-time primitives.

a) Min and Max Operators:: The truth matrices for MIN and MAX operators are shown in figure 4 (a) and (b) respectively. Notice that both functions are symmetric and that they return the same value along the major diagonal.

b) Inhibit Operator:: The structure of the inhibitory truth table is shown in figure 4(c). The INHIBIT function[2] has two inputs: an inhibiting signal and a data-signal (that gets inhibited). If the inhibiting signal arrives first, the output is prevented from ever going high, which corresponds to ∞. If the data signal arrives before or at the same time as the inhibiting signal, the former is allowed to pass through unchanged. Hence this asymmetric gate allows only earlier arriving or coincident data-signals to pass.

c) Coincidence Operator:: Constructing the co-incidence operator by combining the MIN and MAX operators via the INHIBIT operator was first proposed in [2]. It's general form is redrawn in figure 4(e), with a delay window of δ. For the truth matrix shown in figure 4(d) the value of δ is 0. Note that the co-incidence operator specifies the diagonal elements of the truth matrix completely, while leaving all other elements as ∞ – in reality, the coincidence operator, only fills out the (0,0) spot while invariance takes care of the rest. Hence by changing the delays added to any of the inputs of the co-incidence operator, arbitrary points on the surface of the truth matrix can be labelled, even when scaled up to higher dimensions. The trailing constants that emerge from non-prescience are represented by fixed delays. This shows how the co-incidence operator can be combined with shifted version of itself, thus enabling construction of arbitrary space-time functions as shown in figure 4(g).

III. Co-incidence Based Edge Detection

Edges have been known to capture the essential features of a scene and have hence taken a central role in a variety of computer vision algorithms. The discrete nature of pixels and the digital nature of the values that are encoded can lead to some non-trivial

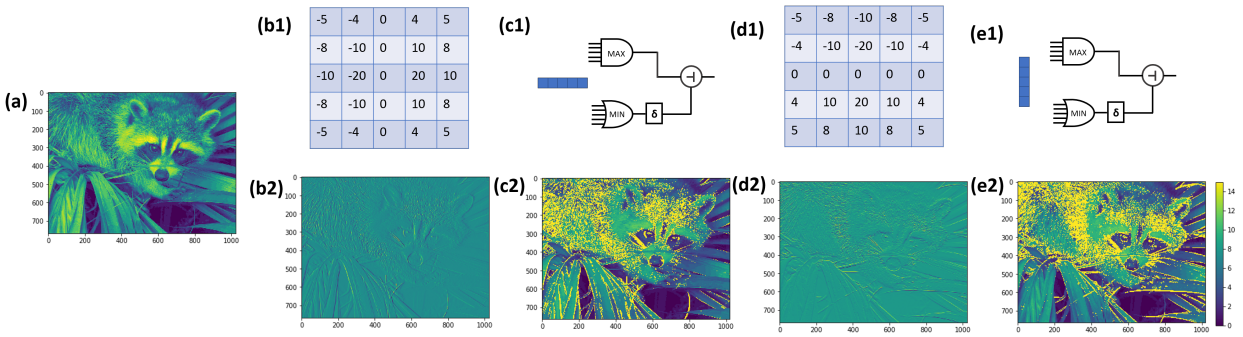

Fig. 5. Comparison between edge detectors: Panel (a) shown the original image. Panels (b1) and (d1) show the classical x and y Sobel edge detectors while panels (c1) and (e1) show the equivalent co-incidence based x and y edge detectors. Panels (b2, c2, d2, e2) show the resultant images after respective transformations have been applied.

issues with determining the threshold of the pixel-value discontinuities that constitutes an edge. Overestimation of this value results in a common problem of spurious edges. Most edge detectors include a smoothing step that reduces this noise, without loss of information, hence allowing false-edge suppression.

The windowed co-incidence detector which has been described above maps well to the edge detection functionality. Such an edge detector is shown in Figures 5 c1 and e1. It works by producing a value based on the magnitudes of the specific pixels from the receptive fields in question. If the magnitudes are close enough to each other (meaning that they lie within a co-incidence window represented by δ in figure 5) in the intensity encoding, they can be thought of as being similar to each other, and hence in the temporal context, being co-incident with each other. Put simply if the input pixel values arrive within δ delay of each other, they output an edge based on the MAX function. If they are not, it represents a discontinuity in pixel intensity which constitutes an edge, which is represented by a ∞ value presented at the output. Note that the specific value of δ is a parameter that can be adjusted. If a small value of δ is causing a lot of spurious edges, it can be increased such that only edges of interest are detected, hence performing a kind of temporal low-pass filtering, akin to the smoothing step in classical filters.

Figures 5(b,d) and (c,e) show a side by side comparison of a classical Sobel edge detector with a co-incidence based edge detector. Though these detectors provide similar functionality, they do so in different ways. The Sobel edge detector kernels which are shown in figures 5(b1 and d1), by virtue of living in a algebra where $\times$ (multiply) and $+$ (sum) are allowed, perform a kind of differentiation based gradient detection. Specifically, figure 5(b1) shows a gradient detector along the horizontal (x) axis. Any differential variation in magnitude across the "column of zeros", which represents an edge, results in a positive or negative value, while similar magnitudes result in a zero value. Hence, sharp vertical lines are detected by this filter, as is clearly visible by observing the foliage on the bottom left region of figure 5(b2). Similar functionality is performed by the co-incidence circuit in figure 5(c1), by outputting an ∞ if a gradient larger than δ is observed. It is worth stressing that this is performed without any summation or multiplication, and only with MIN, MAX and delay operators. The resultant image can be seen in figure 5(c2). Close inspection shows that the edges in figures 5(b2 and c2) are in similar locations. Similarly, vertical(y axis) intensity discontinuities, which can be thought of as horizontal edges, are detected by orthogonal kernels as shown in figures 5(d1 and e1). Another important distinction between the two edge detectors, is that the co-incidence based detector, finds edge locations and "overlays" them on top of the original image, whereas the Sobel detector throws away all non edge related information.

IV. Conclusion

As architects, our intuitions are born in the Boolean realm, and hence do not always map well to the temporal domain. We hope that such truth-matrix based notions make it easier to reason about the behaviors of unary, temporally coded systems. With that aim, we use a truth matrix approach to show how coincidence ends up being fundamental in this domain, and give a toy example to demonstrate its utility.

References

[1] A. Madhavan, T. Sherwood, and D. Strukov. A 4-mm2 180-nm-cmos 15-giga-cell-updates-per-second dna sequence alignment engine based on asynchronous race conditions. In *Custom Integrated Circuits Conference (CICC), 2017 IEEE*, pages 1–4. IEEE, 2017.

[2] J. Smith. Space-time algebra: A model for neocortical computation. In *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*, pages 289–300, June 2018.

[3] G. Tzimpragos, A. Madhavan, D. Vasudevan, D. Strukov, and T. Sherwood. Boosted race trees for low energy classification. In *Proceedings of the Twenty-Forth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '19, April 2019.