DCFS 2010 (Draft)

Descriptional Complexity of Formal Systems
Deadline for submissions: April 25, 2010
Final versions: July 8, 2010

On the Complexity of the Evaluation of Transient Extensions of Boolean Functions

Janusz Brzozowski[(A)] Baiyu Li[(A)] Yuli Ye[(B)]

[(A)]David R. Cheriton School of Computer Science
University of Waterloo, Waterloo, ON – Canada N2L 3G1
{brzozo, b5li}@uwaterloo.ca

[(B)]Department of Computer Science
University of Toronto, Toronto, ON – Canada M5S 3G4
y3ye@cs.toronto.edu

Abstract. Transient algebra is a multi-valued algebra for hazard detection in gate circuits. Sequences of alternating 0's and 1's, called transients, represent signal values, and gates are modeled by extensions of boolean functions to transients. Formulas for computing the output transient of a gate from the input transients are known for NOT, AND, OR and XOR gates and their complements, but, in general, even the problem of deciding whether the length of the output transient exceeds a given bound is NP-complete. We propose a method of evaluating extensions of general boolean functions. We introduce and study a class of functions with the following property: Instead of evaluating an extension of a boolean function on a given set of transients, it is possible to get the same value by using transients derived from the given ones, but having length at most 3. We prove that all functions of three variables, as well as certain other functions, have this property, and can be efficiently evaluated.

Keywords: algebra, boolean function, circuit, complexity, evaluation, gate, hazard, multi-valued, transient, transient extension

1 Introduction

In 2003 Brzozowski and Ésik [2] proposed an infinite algebra as a basis for a theory of hazards in gate circuits. The fundamental concept in this theory is that of a "transient", which is a nonempty alternating sequence of 0's and 1's representing a series of signal values. Boolean functions that are normally used to model gates are extended to transients. Given a boolean function $f(x_1, \ldots, x_n)$, and n transients $\mathbf{x}_1, \ldots, \mathbf{x}_n$, the extension $\mathbf{f}(\mathbf{x}_1, \ldots, \mathbf{x}_n)$ of f to transients is defined as the longest transient that can be obtained by considering all possible orders of changes of the input variables.

This research was supported by the Natural Sciences and Engineering Research Council of Canada under grant No. OGP0000871 and a Postgraduate Scholarship, and by a Graduate Award from the Department of Computer Science, University of Toronto.

For example, consider the circuit of Fig. 1. A change in the input x from 0 to 1 is represented by the transient 01. The output of the inverter will then have the transient 10. If the lower input to the AND gate changes before the upper input because of a delay in the inverter, the AND gate might have a hazard pulse denoted by the transient 010. This pulse is called a *static hazard*; it is static because the AND gate's output is not supposed to change, but there may be an undesired transient pulse. Now, if the OR gate's top input changes twice before its bottom input changes, we might have the transient 0101 at the output of the OR gate. This is a *dynamic hazard*, because the output of the OR gate is supposed to change; however, instead of changing just once it changes three times. Since the detection of hazards is an important problem in circuit analysis and design [2, 3], we are interested in evaluating extensions of boolean functions to predict the worst-case hazards, that is, the longest possible transients.

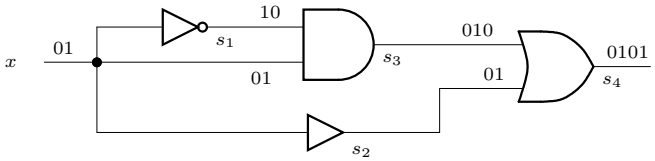

Figure 1. Circuit with hazards

While the definition of the value of the extension of a boolean function is straightforward, it involves the construction of an n-dimensional directed graph in which all possible orders of input changes are shown. Since the size of this graph is exponential in the length of the input transients, this method is inefficient.

There exist simple formulas [2] for common boolean functions like NOT, AND, OR and XOR, and such formulas have been extended to any function obtained from the set {OR, XOR} by complementing any number of inputs, and/or the output [4]. However, function composition does not preserve extensions [2], and the evaluation problem remains open for general functions. Functions that are more complex than OR, AND, XOR, NOR, NAND and XNOR occur frequently in CMOS implementations [3]. We study extensions of general boolean functions, and propose ways of evaluating them. In particular, we introduce a method in which an arbitrary vector $\mathbf{x}$ of transients is replaced by a vector $\tilde{\mathbf{x}}$ of "characteristic transients" which are of length at most 3. We show that evaluating $\mathbf{f}(\mathbf{x})$ can be reduced to evaluating $\mathbf{f}(\tilde{\mathbf{x}})$ for some functions. This makes it possible to efficiently evaluate all the functions of three or fewer variables, and some other functions with special properties.

The remainder of the paper is structured as follows. Transients, vectors of transients, and extensions of boolean functions to transients are defined in Section 2. The evaluation of functions in a certain class $\mathcal{G}$ is considered in Section 3, where we define the "cost" of a transient vector $\mathbf{x}$ to be the difference between the number of changes in $\mathbf{x}$ and the number of changes in $\mathbf{f}(\mathbf{x})$. The concept of cost is extended to paths in digraphs and walks in boolean cubes in Section 4.

Characteristic vectors are defined in Section 5. In Section 6 we prove that all 3-variable functions can be efficiently evaluated using characteristic vectors, and that there exists a 5-variable function that cannot be so evaluated. Section 7 concludes the paper.

2 Transients, vectors, and extensions of functions

The cardinality of a set S is $|S|$. For $n \geq 1$, let $[n] = \{1, \ldots, n\}$. If A is an alphabet, then A^* (A^+) denotes the free monoid (free semigroup) generated by A. The length of a word $w \in A^*$ is $l(w)$, and the first and last letters of $w \in A^+$ are $\alpha(w)$ and $\omega(w)$, respectively. For boolean operations, we use x' for complement, xy for AND, $x + y$ for OR, and $x \oplus y$ for XOR (exclusive OR).

Let $B = \{0, 1\}$; a *binary word* is any word in B^*. A *transient* is a binary word in B^+ of alternating 0's and 1's; thus the set $\mathbf{T}$ of all transients is $0(10)^* + (01)^*01 + (10)^*10 + (10)^*1$, in regular-expression notation. Transients are denoted by boldface letters. A transient can be obtained from any nonempty binary word by *contraction*, *i.e.*, elimination of all duplicates immediately following a symbol; thus contraction is a function from B^+ to $\mathbf{T}$. We denote the contraction of a word w by $\overleftrightarrow{w}$. For example, $\overleftrightarrow{001000} = 010$. For $\mathbf{s}, \mathbf{t} \in \mathbf{T}$, $\mathbf{s} \circ \mathbf{t}$ is concatenation followed by contraction, *i.e.*, $\mathbf{s} \circ \mathbf{t} = \overleftrightarrow{\mathbf{st}}$. The $\circ$ operation is associative.

If $\mathbf{t} = t_1 \cdots t_m$ is a transient, $t_i \in B$ for $i \in [m]$, then $\Delta(\mathbf{t}) = l(\mathbf{t}) - 1 = m - 1$ is the *number of changes* in $\mathbf{t}$. A transient $\mathbf{t}$ is completely determined by its beginning $\alpha(\mathbf{t})$ and the number of changes $\Delta(\mathbf{t})$; thus we have another representation of $\mathbf{t}$ which we indicate by angle brackets: $\mathbf{t} = t_1 \cdots t_m = \langle \alpha(\mathbf{t}); \Delta(\mathbf{t}) \rangle = \langle t_1; m - 1 \rangle$. The number of 0's in a transient $\mathbf{t}$ is $z(\mathbf{t})$, and the number of 1's ("units") is $u(\mathbf{t})$.

A *prefix* of a transient $\mathbf{t} = t_1 \cdots t_m$ is any transient $\mathbf{u} = t_1 \cdots t_i$, where $i \in [m]$. A *suffix* of a transient is defined similarly. Note that we do not allow the empty word to be a prefix or suffix, because the empty word is not a transient. However, $\mathbf{t}$ *is* a prefix and suffix of itself. If $\mathbf{u}$ is a prefix of $\mathbf{t}$ and $l(\mathbf{u}) < l(\mathbf{t})$, then there exists a transient $\mathbf{v}$, a suffix of $\mathbf{t}$, such that $\mathbf{t} = \mathbf{uv}$. A transient $\mathbf{s}$ is the *successor* of a transient $\mathbf{t} = t_1 \cdots t_m$ if and only if $\mathbf{s} = t_1 \cdots t_m t_{m+1}$, where $t_i \in B$, for $i \in [m + 1]$. For example, 010 is the successor of 01.

A *transient vector,* or simply a *vector,* is a tuple $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_n) \in \mathbf{T}^n$. By convention, if $\mathbf{x}$ is a vector, then $\mathbf{x}_i$ is a component of $\mathbf{x}$. The $\circ$ operation is extended to transient vectors component-wise. The *length* of a vector $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ is $l(\mathbf{x}) = \sum_{i=1}^{n} l(\mathbf{x}_i)$. The *number of changes* of $\mathbf{x}$ is $\Delta(\mathbf{x}) = \sum_{i=1}^{n} \Delta(\mathbf{x}_i) = l(\mathbf{x}) - n$. We also define vectors $\alpha(\mathbf{x}) = (\alpha(\mathbf{x}_1), \ldots, \alpha(\mathbf{x}_n))$, and $\omega(\mathbf{x}) = (\omega(\mathbf{x}_1), \ldots, \omega(\mathbf{x}_n))$. A vector is completely determined by its beginning $\alpha(\mathbf{x})$ and the number of changes $\Delta(\mathbf{x}_i)$ of each component of $\mathbf{x}$; thus we have another representation $\mathbf{x} = \langle \alpha(\mathbf{x}); \Delta(\mathbf{x}_1), \ldots, \Delta(\mathbf{x}_n) \rangle$. A vector $\mathbf{u} = (\mathbf{u}_1, \ldots, \mathbf{u}_n)$ is a *prefix* (*suffix*) of vector $\mathbf{v} = (\mathbf{v}_1, \ldots, \mathbf{v}_n)$ if $\mathbf{u}_i$ is a prefix (suffix) of $\mathbf{v}_i$ for all $i \in [n]$. A vector $\mathbf{u} = (\mathbf{u}_1, \ldots, \mathbf{u}_n)$ is a *successor* of $\mathbf{v} = (\mathbf{v}_1, \ldots, \mathbf{v}_n)$ if $\mathbf{u}_i$ is the successor of $\mathbf{v}_i$ for some $i \in [n]$ and $\mathbf{u}_j = \mathbf{v}_j$, for all $j \neq i$.

Our terminology on graphs is from [1]. If $f : B^n \to B$ is a boolean function and $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_n) \in \mathbf{T}^n$ is a vector, we construct the *transient digraph* $D = D_f(\mathbf{x}) = (V, E, \psi, \lambda)$ of f for $\mathbf{x}$, where (V, E, ψ) is a digraph, V (the set of *vertices*) is the set of all prefixes of $\mathbf{x}$, E (the set of *arcs*) is $E = \{e = (\mathbf{u}, \mathbf{v}) \mid \mathbf{v}$ is a successor of $\mathbf{u}\}$, ψ (the *incidence function* assigning to each arc of D an ordered pair of vertices of D) is $\psi(e) = \psi((\mathbf{u}, \mathbf{v})) = (\mathbf{u}, \mathbf{v})$, and $\lambda : V \to B$ is the *output function* assigning the value $f(\omega(\mathbf{v}))$ to every $\mathbf{v} \in V$. Each directed path $P = \mathbf{v}_1, \ldots, \mathbf{v}_m$ in D from $\mathbf{v}_1 = \alpha(\mathbf{x})$ to $\mathbf{v}_m = \mathbf{x}$ has length $m = \sum_{i=1}^{n} \Delta(\mathbf{x}_i)$. We extend λ to paths: $\lambda(P) = \lambda(\mathbf{v}^1) \cdots \lambda(\mathbf{v}^m)$. Paths are always from $\alpha(\mathbf{x})$ to $\mathbf{x}$.

Definition 1. Let $f(x) : B^n \to B$ be a boolean function. The *transient extension* (or simply *extension*) of f is a function $\mathbf{f}(\mathbf{x}) : \mathbf{T}^n \to \mathbf{T}$, such that for any $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_n) \in \mathbf{T}^n$, $\mathbf{f}(\mathbf{x}) = \overleftrightarrow{\lambda(P)}$, where P is a path in $D_f(\mathbf{x})$ and $\overleftrightarrow{\lambda(P)}$ is of maximal length; we call such a path P *optimal*.

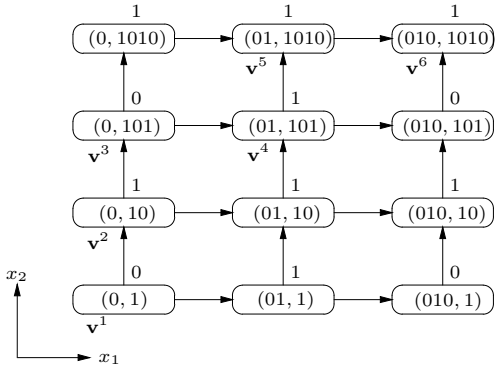

Figure 2. Digraph $D_f(010, 1010)$ for $f = x_1 + x_2'$.

Example 1. *In Fig. 2 we show the digraph* $D_f(010, 1010)$ *for* $f = x_1 + x_2'$*, where changes in* x_1 *are horizontal, and in* x_2*, vertical. The initial vertex is* $\alpha(010, 1010) = (0, 1)$*. If the inputs are changed in the order* x_2, x_2, x_1, x_2, x_1 *(path* $\mathbf{v}^1, \mathbf{v}^2, \ldots, \mathbf{v}^6$*), then the binary word defined by* λ *is* 010111*, and its contraction is* 0101*. The longest output is* $(01)^3$*, corresponding to the optimal path in which the changes are made in the order* x_1, x_1, x_2, x_2, x_2*, and so* $\mathbf{f}(010, 1010) = (01)^3$*.*

For vector $\mathbf{x} \in \mathbf{T}^n$, let $\varphi(\mathbf{x})$ be the number of paths in $D_f(\mathbf{x})$, let $m = \Delta(\mathbf{x})$, and let $m_i = \Delta(\mathbf{x}_i)$ for $i \in [n]$. Then

$$\varphi(\mathbf{x}) = \binom{m}{m_1, \ldots, m_n} = \frac{m!}{m_1! \cdots m_n!}; \tag{1}$$

that is, $\varphi(\mathbf{x})$ is a multinomial coefficient. The maximal value of $\varphi(\mathbf{x})$ has the following approximation [7]:

$$\varphi(\mathbf{x}) \approx (2\pi m)^{\frac{1-n}{2}} n^{m+\frac{n}{2}}. \tag{2}$$

We usually consider n to be small or fixed; then $\varphi(\mathbf{x})$ is exponential in m. Consequently, the obvious way to evaluate $\mathbf{f}(\mathbf{x})$ is not feasible because of the large number of paths to explore.

3 Functions in class $\mathcal{G}$

In contrast to the general case above, for NOT, XOR, OR and AND there are simple formulas [2]: If $\mathbf{t} = t_1 \cdots t_m$, then

$$\mathbf{t}' = (t_1 \cdots t_m)' = t_1' \cdots t_m'. \tag{3}$$

If $f(x_1, \ldots, x_n) = x_1 \oplus \cdots \oplus x_n$ is XOR, then, for all $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_n) \in \mathbf{T}^n$,

$$\alpha(\mathbf{f}(\mathbf{x})) = \alpha(\mathbf{x}_1) \oplus \cdots \oplus \alpha(\mathbf{x}_n), \quad \omega(\mathbf{f}(\mathbf{x})) = \omega(\mathbf{x}_1) \oplus \cdots \oplus \omega(\mathbf{x}_n), \tag{4}$$

$$l(\mathbf{f}(\mathbf{x})) = 1 + \sum_{i=1}^{n} (l(\mathbf{x}_i) - 1). \tag{5}$$

If $f(x_1, \ldots, x_n) = x_1 + \cdots + x_n$ is OR, then, for all $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_n) \in \mathbf{T}^n$,

$$\alpha(\mathbf{f}(\mathbf{x})) = \alpha(\mathbf{x}_1) + \cdots + \alpha(\mathbf{x}_n), \quad \omega(\mathbf{f}(\mathbf{x})) = \omega(\mathbf{x}_1) + \cdots + \omega(\mathbf{x}_n), \tag{6}$$

$$z(\mathbf{f}(\mathbf{x})) = \begin{cases} 0, & \text{if } \exists i \in [n]\ \mathbf{x}_i = 1; \\ 1 + \sum_{i=1}^{n} (z(\mathbf{x}_i) - 1), & \text{otherwise.} \end{cases} \tag{7}$$

If $f(x_1, \ldots, x_n) = x_1 \cdots x_n$ is AND, then, for all $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_n) \in \mathbf{T}^n$,

$$\alpha(\mathbf{f}(\mathbf{x})) = \alpha(\mathbf{x}_1) \cdots \alpha(\mathbf{x}_n), \quad \omega(\mathbf{f}(\mathbf{x})) = \omega(\mathbf{x}_1) \cdots \omega(\mathbf{x}_n), \tag{8}$$

$$u(\mathbf{f}(\mathbf{x})) = \begin{cases} 0, & \text{if } \exists i \in [n]\ \mathbf{x}_i = 0; \\ 1 + \sum_{i=1}^{n} (u(\mathbf{x}_i) - 1), & \text{otherwise.} \end{cases} \tag{9}$$

Using these formulas we can evaluate transient extensions of NOT, XOR, OR and AND in the time linear in the length of the input vector. For example, to evaluate OR for $\mathbf{x} \in \mathbf{T}^n$, we compute $\alpha(\mathbf{f}(\mathbf{x}))$, $\omega(\mathbf{f}(\mathbf{x}))$, and the number of 0's in $\mathbf{x}$.

The class $\mathcal{G}$ of boolean functions is defined as follows [4]:

Definition 2. Let $\mathcal{H} = \{\text{OR}, \text{XOR}\}$ and let $\mathcal{G}$ be the set of functions obtained by complementing any number of inputs and/or the output of functions from $\mathcal{H}$; here OR and XOR may have any non-zero number of inputs, including one.

Note that a 1-input OR or XOR function is the identity function, and that $\mathcal{G}$ includes all the boolean functions of two variables, except the constants 0 and 1, as well as AND, NOR, NAND and XNOR functions with any numbers of inputs.

It was proved in [4] that functions in $\mathcal{G}$ can be evaluated by complementing the input transients of any complemented arguments, and by complementing the output transient, if the function itself is complemented. Consequently, we have

Proposition 1. *Functions in $\mathcal{G}$ can be evaluated in the time linear in the length of the input vector.*

For example, if $f(x_1, x_2) = (x_1 + x_2')'$, then $\mathbf{f}(010, 10) = (010 + (10)')' = (010 + 01)' = (0101)' = 1010$. However, in general, function composition does not preserve extensions [2]. For example, by (4) and (5), $01 \oplus 101 = 1010$, but if we express $\mathbf{s} \oplus \mathbf{t}$ as $\mathbf{st}' + \mathbf{s}'\mathbf{t}$, we get 101010. For this reason, we need to consider functions that are not in $\mathcal{G}$ separately.

As we have seen, evaluating a transient extension from the transient digraph is not efficient. In [2] it is shown that even the problem of estimating the length of $\mathbf{f}(\mathbf{x})$ is NP-complete. However, the concept of "cost" that we are about to define makes the calculation feasible for some functions.

Definition 3. Let $f : B^n \to B$ be a boolean function, and $\mathbf{f} : \mathbf{T}^n \to \mathbf{T}$, its extension. Let $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_2) = \langle \alpha(\mathbf{x}); \Delta(\mathbf{x}_1), \ldots, \Delta(\mathbf{x}_n) \rangle$ be a transient vector. The *cost of* $\mathbf{x}$ *for* f is $c_f(\mathbf{x}) = \Delta(\mathbf{x}) - \Delta(\mathbf{f}(\mathbf{x})) = \sum_{i=1}^{n} \Delta(\mathbf{x}_i) - \Delta(\mathbf{f}(\mathbf{x}))$.

The following upper bound for $l(\mathbf{f}(\mathbf{x}))$ is given in [2]:

$$l(\mathbf{f}(\mathbf{x})) \leq 1 + \sum_{i=1}^{n} (l(\mathbf{x}_i) - 1) = 1 + \sum_{i=1}^{n} \Delta(\mathbf{x}_i) = 1 + \Delta(\mathbf{x}). \tag{10}$$

Thus $\Delta(\mathbf{f}(\mathbf{x})) = l(\mathbf{f}(\mathbf{x})) - 1 \leq \Delta(\mathbf{x})$, and so $c_f(\mathbf{x})$ is a non-negative integer. If we know $\mathbf{x}$ and its cost $c_f(\mathbf{x})$, then we can easily evaluate $\mathbf{f}(\mathbf{x})$ as follows:

$$\mathbf{f}(\mathbf{x}) = \langle \alpha(\mathbf{f}(\mathbf{x})); \Delta(\mathbf{f}(\mathbf{x})) \rangle = \langle \alpha(\mathbf{f}(\mathbf{x})); \Delta(\mathbf{x}) - c_f(\mathbf{x}) \rangle. \tag{11}$$

Example 2. *For* $\mathbf{x} = (0101, 10101) = \langle (0, 1), 3, 4 \rangle$, $\Delta(\mathbf{x}) = 7$. *For* $f(x) = x_1 + x_2'$, *using (3), (6) and (7), we have* $\mathbf{f}(\mathbf{x}) = (01)^4$, $\Delta(\mathbf{f}(\mathbf{x})) = 7$, *and* $c_f(\mathbf{x}) = 7 - 7 = 0$. *For* $\mathbf{y} = (0101, 0101010) = \langle (0, 0), 3, 6 \rangle$, *if* $f(y) = y_1 + y_2'$, *we have* $\mathbf{f}(\mathbf{y}) = (10)^4 1$, *and* $c_f(\mathbf{y}) = 9 - 8 = 1$.

A binary vector $(x_1, \ldots, x_n)$ with $x_i = 0$, for all $i \in [n]$ is denoted by 0^n. We define *non-negative subtraction* $m \ominus n$ of integer n from integer m as $m \ominus n = m - n$ if $m \geq n$, and $m \ominus n = 0$, otherwise. A transient $\mathbf{t} = t_1 \cdots t_m$ is *proper* if its length is at least 2, *i.e.*, if it contains at least one change. A vector is *proper* if all of its components are proper.

Theorem 4. *If* $\mathbf{x}$ *is proper, then*

$$c_{xor}(\mathbf{x}) = 0, \tag{12}$$

$$c_{or}(\mathbf{x}) = (u(\alpha(\mathbf{x})) \ominus 1) + (u(\omega(\mathbf{x})) \ominus 1), \tag{13}$$

$$c_{and}(\mathbf{x}) = (z(\alpha(\mathbf{x})) \ominus 1) + (z(\omega(\mathbf{x})) \ominus 1). \tag{14}$$

Proof: If f is XOR, we have $c_{xor} = \Delta(\mathbf{x}) - \Delta(\mathbf{f}(\mathbf{x})) = \sum_{i=1}^{n} \Delta(\mathbf{x}_i) - \Delta(\mathbf{f}(\mathbf{x})) = \sum_{i=1}^{n} (l(\mathbf{x}_i) - 1) - (l(\mathbf{x}) - 1) = 0$, where we have used Equation (5).

If f is OR, we consider three cases:

(I) $u(\alpha(\mathbf{x}))=0$. We have $u(\alpha(\mathbf{x}))\ominus 1=0$. Since $\alpha(\mathbf{x}_i)=0$ for all $i\in[n]$, we have $\alpha(f(\mathbf{x}))=0$. Let $S=\{i\mid \omega(\mathbf{x}_i)=1\}$, $T=\{i\mid \omega(\mathbf{x}_i)=0\}$. Then

$$\begin{aligned}\Delta(\mathbf{x}) &= \textstyle\sum_{i=1}^{n}(l(\mathbf{x}_i)-1)=\sum_{i\in S}(l(\mathbf{x}_i)-1)+\sum_{i\in T}(l(\mathbf{x}_i)-1)\\ &= \textstyle\sum_{i\in S}(2z(\mathbf{x}_i)-1)+\sum_{i\in T}(2z(\mathbf{x}_i)-2)\\ &= \textstyle\sum_{i=1}^{n}(2z(\mathbf{x}_i)-2)+|S|=2(z(f(\mathbf{x}))-1)+u(\omega(\mathbf{x})),\end{aligned}$$

where the last equality uses Equation (7). If $u(\omega(\mathbf{x}))=0$, then $\omega(\mathbf{x}_i)=0$ for all $i\in[n]$, and $\omega(f(\mathbf{x}))=0$. Thus we have $l(f(\mathbf{x}))=2z(f(\mathbf{x}))-1$, and $\Delta(\mathbf{x})=l(f(\mathbf{x}))-1=\Delta(\mathbf{f}(\mathbf{x}))$. Then $c_{or}(\mathbf{x})=\Delta(\mathbf{x})-\Delta(\mathbf{f}(\mathbf{x}))=0=(u(\alpha(\mathbf{x}))\ominus 1)+(u(\omega(\mathbf{x}))\ominus 1)$. Otherwise, $u(\omega(\mathbf{x}))\geq 1$. Then $\omega(f(\mathbf{x}))=1$, and $l(f(\mathbf{x}))=2z(f(\mathbf{x}))$. Thus

$$\begin{aligned}\Delta(\mathbf{x}) &= 2(z(f(\mathbf{x}))-1)+u(\omega(\mathbf{x}))=l(f(\mathbf{x}))-1+u(\omega(\mathbf{x}))-1\\ &= l(f(\mathbf{x}))-1+u(\omega(\mathbf{x}))\ominus 1=\Delta(\mathbf{f}(\mathbf{x}))+u(\omega(\mathbf{x}))\ominus 1.\end{aligned}$$

Hence, $c_{or}(\mathbf{x})=u(\omega(\mathbf{x}))\ominus 1=(u(\alpha(\mathbf{x}))\ominus 1)+(u(\omega(\mathbf{x}))\ominus 1)$.

(II) $u(\omega(\mathbf{x}))=0$. This case is symmetric to Case I.

(III) $u(\alpha(\mathbf{x}))\neq 0$, and $u(\omega(\mathbf{x}))\neq 0$. Since $\mathbf{x}$ is proper, then for all $i\in[n]$, we have $l(\mathbf{x}_i)\geq 2$, and there is at least one 0 in $\mathbf{x}_i$. Let $\mathbf{x}_i=\mathbf{v}_i\circ\mathbf{u}_i$, where $\omega(\mathbf{v}_i)=\alpha(\mathbf{u}_i)=0$, $\mathbf{v}=(\mathbf{v}_1,\ldots,\mathbf{v}_n)$, and $\mathbf{u}=(\mathbf{u}_1,\ldots,\mathbf{u}_n)$. Then $\mathbf{x}=\mathbf{v}\circ\mathbf{u}$, and $\omega(\mathbf{v})=\alpha(\mathbf{u})=(0,\ldots,0)=0^n$. By Cases I and II, $c_{or}(\mathbf{v})=u(\alpha(\mathbf{x}))\ominus 1$, and $c_{or}(\mathbf{u})=u(\omega(\mathbf{x}))\ominus 1$. Therefore, $c_{or}(\mathbf{x})=c_{or}(\mathbf{v})+c_{or}(\mathbf{u})=(u(\alpha(\mathbf{x}))\ominus 1)+(u(\omega(\mathbf{x}))\ominus 1)$.

By AND/OR duality, we have $c_{and}(\mathbf{x})=(z(\alpha(\mathbf{x}))\ominus 1)+(z(\omega(\mathbf{x}))\ominus 1)$. □

4 Costs of paths in digraphs and walks in cubes

For $n\geq 1$, the *boolean n-cube* is a graph $C^n=(V,E,\psi)$, where $V=B^n$ (*vertices*), $E=\{e=(v^i,v^j)\mid v^i,v^j\in V$ and v^i and v^j differ in exactly one coordinate$\}$ (*edges*), and $\psi(e)=\psi((v^i,v^j))=(v^i,v^j)$ (*incidence funtion*). For a boolean function $f:B^n\to B$, $n\geq 1$, the *cube* C_f^n of f is the n-cube where $f(v)$ is assigned to each vertex $v\in V=B^n$. If n is understood, we denote C_f^n by C_f.

In a function cube C_f, an edge $e=(v^i,v^j)\in E$ is *live* if $f(v^i)\neq f(v^j)$; otherwise it is *dead.* A *live graph* of f is that subgraph L_f of C_f that consists of all the live edges and their incident vertices.

Instead of considering paths in a digraph $D_f(\mathbf{x})$, we will examine walks in the cube C_f. The size of a digraph $D_f(\mathbf{x})$ increases as the length of $\mathbf{x}$ increases, and the length of any path in $D_f(\mathbf{x})$ increases accordingly. However, the size of a cube C_f is independent of any vector $\mathbf{x}$, if the dimension of $\mathbf{x}$ is fixed.

Example 3. *The cube of $f=x_1+x_2'$ is shown in Fig. 3 (a), where a vertex is white if $f(v^i)=0$, and black otherwise. We also write 00 instead of $(0,0)$, etc., to simplify the notation. The live edges of f are shown by thick lines in Fig. 3 (b). The live graph of f is shown in Fig. 3 (c).*

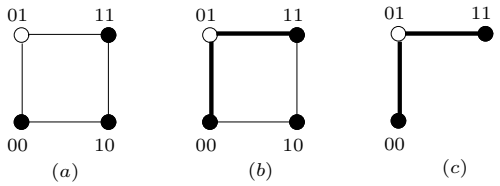

Figure 3. Graphs for $f = x_1 + x_2'$: (a) C_f; (b) live edges; (c) L_f.

We extend the concept of cost to paths in digraphs and walks in cubes. For a path $P = \mathbf{v}^1, \ldots, \mathbf{v}^m$ in $D_f(\mathbf{x})$, let $c_P = |E_=(P)|$, where $E_=(P) = \{(\mathbf{v}^i, \mathbf{v}^{i+1}) \mid \lambda(\mathbf{v}^i) = \lambda(\mathbf{v}^{i+1}),\ i = 1, \ldots, m-1\}$. Thus c_P is the number of arcs in P whose endpoints have the same λ values. For any walk $W = w^1, w^2, \cdots, w^m$ in C_f, let $c_W = |E_=(W)|$, where $E_=(W) = \{(w^j, w^{j+1}) \mid f(w^j) = f(w^{j+1}),\ j = 1, \ldots, m-1\}$. Thus c_W is the number of edges in W whose endpoints have the same f values, that is, the number of edges in W which are not in the live graph L_f.

Definition 5. Let $f : B^n \to B$ be a boolean function, and $\mathbf{f}$, its extension. Let $\mathbf{x} \in \mathbf{T}^n$ be a vector, and $P = \mathbf{v}^1, \ldots, \mathbf{v}^r$, a path in $D_f(\mathbf{x})$ from $\mathbf{v}^1 = \alpha(\mathbf{x})$ to $\mathbf{v}^r = \mathbf{x}$. Let $W(P)$ be the sequence $W(P) = \omega(\mathbf{v}^1), \ldots, \omega(\mathbf{v}^r)$. Conversely, let $W = w^1, \ldots, w^r$ be any walk in C_f, where $w^i \in B^n$, for $i = 1, \ldots, r$. Let $P(W)$ be the sequence $P(W) = w^1, w^1 \circ w^2, \ldots, w^1 \circ \cdots \circ w^r$.

Theorem 6. *If P is a path in $D_f(\mathbf{x})$ then $W(P)$ is a walk in C_f and $c_P = c_{W(P)}$. If $W = w^1, \ldots, w^r$ is a walk in C_f, let $\mathbf{x} = w^1 \circ \cdots \circ w^r$. Then $P(W)$ is a path in $D_f(\mathbf{x})$ and $c_W = c_{P(W)}$. Moreover, if P is a path in $D_f(\mathbf{x})$, then $P(W(P)) = P$, and if W is a walk in C_f, then $W(P(W)) = W$.*

A walk W is *optimal* if path $P(W)$ is optimal.

Example 4. *In Fig. 2, $P = \mathbf{v}^1, \ldots, \mathbf{v}^6$ is a path from $\mathbf{v}^1 = \alpha(\mathbf{x})$ to $\mathbf{v}^6 = \mathbf{x}$, and $c_P = |\{(\mathbf{v}^4, \mathbf{v}^5), (\mathbf{v}^5, \mathbf{v}^6)\}| = 2$. Let $w^i = \omega(\mathbf{v}^i)$, for $i = 1, \ldots, 6$; then $W(P) = w^1, \ldots, w^6$ is a walk in Fig. 3 (a). Note that (w^4, w^5) and (w^5, w^6) are not in the live graph L_f; thus $c_{W(P)} = 2 = c_P$.*

Conversely, for $W = w^1, \ldots, w^6 = (01, 00, 10, 11, 10, 00)$, let $\mathbf{x}^1 = w^1 = (0, 1)$, $\mathbf{x}^2 = w^1 \circ w^2 = (0, 1) \circ (0, 0) = (0 \circ 0, 1 \circ 0) = (0, 10)$, $\mathbf{x}^3 = (0, 101)$, $\mathbf{x}^4 = (01, 101)$, $\mathbf{x}^5 = (01, 1010)$, and $\mathbf{x}^6 = (010, 1010)$. Then $P(W) = \mathbf{x}^1, \ldots, \mathbf{x}^6$ is a path in $D_f(\mathbf{x})$, and $c_{P(W)} = 2 = c_W$. In addition, $P(W) = P(W(P)) = P$. Here $c_P \neq 0$ and W is not a walk in L_f. If $\mathbf{u}^1 = (0, 1)$, $\mathbf{u}^2 = (01, 1)$, $\mathbf{u}^3 = (010, 1)$, $\mathbf{u}^4 = (010, 10)$, $\mathbf{u}^5 = (010, 101)$, $\mathbf{u}^6 = (010, 1010)$, and $P = \mathbf{u}^1 \cdots \mathbf{u}^6$, then $c_P = 0$, and $W(P) = 01, 11, 01, 00, 01, 00$ is a walk in L_f.

In the rest of the paper we consider walks in the n-cube C_f. A walk $W = w^1, \ldots, w^r$ is a *walk for a vector* $\mathbf{x}$ if $\mathbf{x} = w^1 \circ \cdots \circ w^r$. To evaluate $\mathbf{f}(\mathbf{x})$ we find a walk $W = w^1, \ldots, w^r$ for $\mathbf{x}$ with minimal cost, and then $\mathbf{f}(\mathbf{x}) = f(w^1) \circ \cdots \circ f(w^r)$. This approach takes the advantage of the fact that the size of the n-cube C_f is independent of the length $l(\mathbf{x})$ of the input vector $\mathbf{x}$.

5 Characteristic vectors

Now we are interested only in proper vectors. A vector is *minimal* if each component has length 2 or 3. If a transient $\mathbf{t} = t_1 \cdots t_m$ is proper, the *characteristic transient* of $\mathbf{t}$ is $\tilde{\mathbf{t}}$, where $\tilde{\mathbf{t}} = t_1 t_2$ if m is even, and $\tilde{\mathbf{t}} = t_1 t_2 t_3$ if m is odd. Note that $\tilde{\mathbf{t}}$ is a prefix of $\mathbf{t}$, $\alpha(\tilde{\mathbf{t}}) = \alpha(\mathbf{t})$, $\omega(\tilde{\mathbf{t}}) = \omega(\mathbf{t})$, and $\Delta(\mathbf{t}) \equiv \Delta(\tilde{\mathbf{t}})$, where $\equiv$ is equivalence modulo 2. If $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ is a proper vector, then $\tilde{\mathbf{x}} = (\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_n)$ is the *characteristic vector* of $\mathbf{x}$. Also, $\tilde{\mathbf{x}}$ is a prefix of $\mathbf{x}$, $\alpha(\tilde{\mathbf{x}}) = \alpha(\mathbf{x})$, $\omega(\tilde{\mathbf{x}}) = \omega(\mathbf{x})$, and $\Delta(\mathbf{x}_i) \equiv \Delta(\tilde{\mathbf{x}}_i)$, for $i \in [n]$. The characteristic vector of any vector is minimal, and every minimal vector is the characteristic vector of some vector. Any vector $\mathbf{x}$ which has $\tilde{\mathbf{x}}$ as its characteristic vector is an *prolongation* of $\tilde{\mathbf{x}}$.

A function $f : B^n \to B$ *depends* on its k-th argument if there exist $x_i \in B$, such that $f(x_1, \ldots, x_{k-1}, 0, x_{k+1}, \ldots, x_n) \neq f(x_1, \ldots, x_{k-1}, 1, x_{k+1}, \ldots, x_n)$. In this section, we only consider functions that depend on all of their arguments. If $f : B^n \to B$ depends on x_k, then there exists at least one live edge $e = (w^i, w^j)$ in the cube C_f of f, where w^i and w^j differ only in x_k.

Definition 7. A boolean function $f : B^n \to B$ that depends on all of its variables is *convenient* if, for every proper vector $\mathbf{x}$, the cost $c_f(\mathbf{x})$ of $\mathbf{x}$ is equal to the cost $c_f(\tilde{\mathbf{x}})$ of its characteristic vector $\tilde{\mathbf{x}}$; otherwise, f is *inconvenient*.

For any vector $\mathbf{x} \in \mathbf{T}^n$, let $\tilde{\varphi}(\mathbf{x}) = \varphi(\tilde{\mathbf{x}})$ be the number of walks for $\tilde{\mathbf{x}}$. When $\Delta(\tilde{\mathbf{x}}_1) = \cdots = \Delta(\tilde{\mathbf{x}}_n) = 2$, the maximal value of $\tilde{\varphi}(\mathbf{x})$ is obtained from Equation (2) by setting $m = 2n$, and is approximately

$$\tilde{\varphi}(\mathbf{x}) \approx (4\pi)^{\frac{1-n}{2}} n^{2n+\frac{1}{2}}, \tag{15}$$

which is independent of the length m of $\mathbf{x}$; hence the evaluation of a particular function $\mathbf{f}$ can be much more efficient if f is convenient. We first search for an optimal walk for $\tilde{\mathbf{x}}$, and get its cost c; we then compute the boolean value $f(\alpha(\mathbf{x}))$ and construct a transient with $\Delta(\mathbf{x}) - c$ changes beginning with $f(\alpha(\mathbf{x}))$. With fixed n, this can be done in time linear in the length of $\mathbf{x}$.

The next claim follows immediately from Theorem 4.

Corollary 8. *All functions in $\mathcal{G}$ are convenient.*

To prove that a function f is convenient we must verify that, for every vector $\mathbf{x}$, the cost of $\mathbf{x}$ for f is the same as the cost of $\tilde{\mathbf{x}}$ for f. Equivalently, we need to show that, for every minimal vector $\tilde{\mathbf{x}}$, the cost for f of any prolongation $\mathbf{x}$ of $\tilde{\mathbf{x}}$ is the same as the cost of $\tilde{\mathbf{x}}$ for f.

An edge e in a cube corresponds to the unique coordinate x_i that has complementary values in the two vertices of that edge; we say that e is *an edge in coordinate x_i*. An edge is *incident* to a walk W if it shares at least one vertex with W. A walk W is *complete* if, for every coordinate, there is a live edge in that coordinate incident to W. A vertex v in a cube C_f of a boolean function f is a *focus* if every edge incident to v is live. Any walk through a focus is complete.

Proposition 2. *Let $f : B^n \to B$ be a boolean function, and let C_f be its cube. Let $\mathbf{x}$ be a transient vector and $\tilde{\mathbf{x}}$ its characteristic vector. If an optimal walk W in C_f for $\tilde{\mathbf{x}}$ is complete, then the cost of any optimal walk for $\mathbf{x}$ is c_W.*

Proof: Let W be an optimal walk for $\tilde{\mathbf{x}}$. Since the difference between the number of changes in $\mathbf{x}_i$ and in $\tilde{\mathbf{x}}_i$ is even for any i, the additional changes in $\mathbf{x}_i$ can be inserted after a vertex incident to the live edge in that coordinate is reached. □

6 3-variable functions

Suppose $f : B^n \to B$ and $g : B^n \to B$ are boolean functions. If $g(y_1, \ldots, y_n)$ can be obtained from $f(x_1, \ldots, x_n)$ by renaming the variables and complementing some number of inputs and/or the output, then we write $f \sim g$, where $\sim$ is an equivalence relation [5, 6], and we say that f and g are in the same *symmetry class*. For example, we can start with $x+y$, rename y as z to get $x+z$, complement z to get $x + z'$, and complement this result to get $x'z$. Thus $x'z \sim x + y$. If we know how to evaluate the transient extension of f, then we can also evaluate the transient extensions of all the functions that are in the same symmetry class as f [4]. Hence we consider only one representative function of each symmetry class.

For $n = 3$, there are 256 functions which can be reduced to 14 symmetry classes [5, 6]. Four classes, represented by 0, x, $x+y$, and $x \oplus y$, contain degenerate functions; these classes account for 38 functions which can all be evaluated using the formulas in Section 2. The remaining 218 functions can be reduced to 2 symmetry classes (18 functions) in $\mathcal{G}$ and 8 classes (200 functions) represented by the functions shown in Fig. 4, where the circled vertices can be ignored for now.

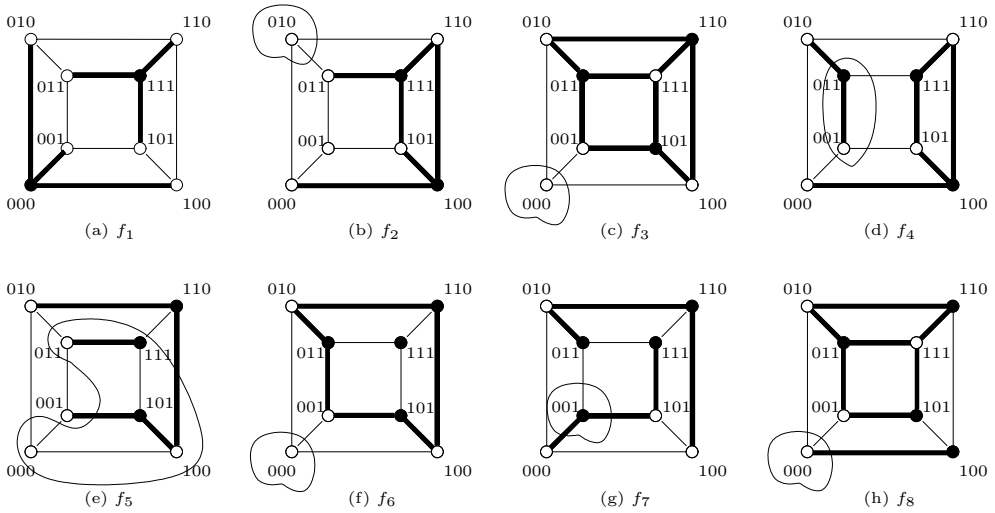

Figure 4. Representatives of the eight symmetry classes of 3-variable functions.

The main result in this section is the following:

Theorem 9. *All 3-variable functions are convenient.*

To prove Theorem 9, it is sufficient to examine the eight functions of Fig. 4. The following results are useful in proving that certain walks are optimal:

Lemma 10. *Let W_1 and W_2 be two walks from vertex u to vertex v. Then the difference between the cost of W_1 and that of W_2 is a multiple of 2.*

Corollary 11. *Let $c \geq 0$ be any integer. If there is no walk from u to v of cost less than or equal to $c-1$, and there is a walk W of cost $c+1$, then W is optimal. In particular, every walk of cost 1 from u to v is optimal.*

Lemma 12. *For a 3-variable function and a minimal vector $\mathbf{x}$, if $\alpha(\mathbf{x})$ is within distance 1 from a focus, then there is a complete optimal walk for $\mathbf{x}$.*

For any 3-variable function f, a walk $v_0v_1v_2v_3v_4v_5$ on C_f is *alternating* if each subwalk $v_iv_{i+1}v_{i+2}v_{i+3}$, $i = 0, 1, 2$, contains a change in every coordinate.

Lemma 13. *Let $\mathbf{x}$ be a minimal vector for a 3-variable function, let $U = v_0v_1v_2v_3$ be any walk such that $v_0 = \alpha(\mathbf{x})$, and let $W = Uv_4v_5$. If W is alternating, $c(U) = 0$, and $c(W) \leq 1$, then there exists a complete optimal walk V for $\mathbf{x}$.*

We are now ready to sketch a proof of Theorem 9.

Proof: For each of the eight functions in Fig. 4, we enumerate all minimal vectors $\mathbf{x} = \langle \alpha(\mathbf{x}); \Delta_1, \Delta_2, \Delta_3 \rangle$, and prove that there is a complete optimal walk for each $\mathbf{x}$. The vertices that need to be considered are circled in the figure. Since there are eight functions and each of them has eight possible starting vertices $\alpha(\mathbf{x})$ and eight change vectors $(\Delta_1, \Delta_2, \Delta_3)$, there are 512 cases to analyze. We reduce this number significantly by using Corollary 11, Lemmas 12 and 13, and symmetry. The value $4b_1 + 2b_2 + b_3$ represents vertex (b_1, b_2, b_3), and we denote walks by words, rather than sequences. The eight functions are treated as follows:

1. For f_1, every vertex is within distance 1 from a focus; by Lemma 12, no vertex needs to be considered.
2. For $f_2 = x_1(x_2 \oplus x_3)$, only vertices 1 and 2 are not within distance 1 of a focus. Since f_2 is symmetric in x_2 and x_3, we consider only one of 1 and 2, say 2. We list the complete walks in pairs $(\Delta_1\Delta_2\Delta_3, W)$, where W is a complete optimal walk for $\langle 2; \Delta_1, \Delta_2, \Delta_3 \rangle$. There are no minimal walks of cost 0. The walks of cost 1 are: $(111, 2645)$, $(112, 26454)$, $(121, 26467)$, $(122, 264676)$, $(212, 267640)$, and $(221, 264673)$. Walks $(211, 26451)$ and $(222, 2646762)$ are of cost 2; by Corollary 11, they are optimal. Each walk is complete for it goes through a focus.
3. For f_3, only 0 is not within distance 1 of a focus, so we consider it.
4. For $f_4 = x_2x_3 + x_1x_2'x_3'$, $1, 2, 3, 7$ are not within distance 1 of a focus. Since f_4 is symmetric in x_2 and x_3, we consider only 1 and not 2. For 7, there is a walk $W = 754023$, which satisfies the conditions of Lemma 13. So there is a complete optimal walk for any minimal vector starting at 7, and we consider only 1 and 3.
5. The function $f_5 = x_1(x_2 + x_3)$ has no focus. Since it is symmetric in x_2 and

x_3, we consider only 1 (and not 2) and 5 (and not 6), say. Walk 154623 satisfies the conditions of Lemma 13, taking care of 1. So we consider 0, 3, 4, 5, and 7.
6. For $f_6 = x_1x_2 + x_2x_3 + x_3x_1$, there is no focus. Because f is symmetric in all three variables, it suffices to consider 1 (and not 2 and 4), 6 (and not 3 and 5), 0 and 7. Moreover, if we complement the function, the live and dead edges are preserved. Hence 1 and 6 are symmetric in the cube as are 0 and 7, and we consider only 0 and 1. However, for vertex 1, walk 132645 meets the conditions of Lemma 13. Hence we look only at 0.
7. For f_7, vertices 0, 3, 4, and 7 are symmetric in the live graph, as are 1, 2, 5, and 6. The alternating walk 015762 takes care of 0, leaving only 1 to consider.
8. For f_8, only 0 and 4 are not within distance 1 of a focus, and they are symmetric with respect to live edges. So we examine only 0. □

We now show a function which is not convenient. Let $S_i(x_1, x_2, x_3, x_4)$ be the symmetric function of four variables that is 1 if and only if precisely i of its variables are 1. Also, let $S_{2,3}(x_1, x_2, x_3, x_4) = S_2(x_1, x_2, x_3, x_4) + S_3(x_1, x_2, x_3, x_4)$.

Proposition 3. *$f = S_{2,3}(x_1, x_2, x_3, x_4) + x_0x_1x_2x_3x_4$ is inconvenient.*

7 Conclusions

The evaluation of extensions of boolean functions is simplified if we use walks in boolean cubes instead of paths in digraphs. The evaluation of extensions of convenient functions can be done in polynomial time if we use characteristic vectors. All 3-variable functions are convenient, but there exist inconvenient 5-variable functions. It remains open whether there is an inconvenient 4-variable function. The problem of characterizing convenient functions is also open.

References

[1] J. A. Bondy & U. S. R. Murty (1976): *Graph Theory with Applications*. American Elsevier.

[2] J. Brzozowski & Z. Ésik (2003): *Hazard algebras. Formal Methods in System Design* 23(3), pp. 223–256.

[3] J. Brzozowski & C-J. Seger (1995): *Asynchronous Circuits*. Springer.

[4] J. Brzozowski & Y. Ye (2010): *Gate circuits with feedback in finite multivalued algebras of transients. J. of Mult.-Valued Logic & Soft Computing* 16(1–2), pp. 155–176.

[5] S. H. Caldwell (1958): *Switching Circuits and Logical Design*. Wiley.

[6] M. A. Harrison (1965): *Introduction to Switching and Automata Theory*. Mc-Graw-Hill.

[7] L. B. Richmond & J. Shallit (2009): *Counting abelian squares. Electronic J. Combinatorics* 16(1), p. #R72.