Supporting Increment and Decrement Operations in Balancing Networks*

William Aiello† Costas Busch‡ Maurice Herlihy‡

Marios Mavronicolas§ Nir Shavit¶ Dan Touitou||

November 6, 1998

Abstract

Counting networks are a class of distributed data structures that support highly concurrent implementations of shared *Fetch&Increment* counters. Applications of these counters include shared pools and stacks, load balancing, and software barriers [4, 16, 18, 23]. A limitation of counting networks is that the resulting shared counters can be incremented, but not decremented.

A recent result by Shavit and Touitou [23] showed that the subclass of tree-shaped counting networks can support, in addition, decrement operations. This paper generalizes their result, showing that *any* counting network can be extended to support atomic decrements in a simple and natural way. Moreover, it is shown that decrement operations can be supported in networks that provide weaker properties,

*This paper combines, unifies, and extends results appearing in preliminary form in [2] and [6]. A preliminary version of the paper will appear at the *16th International Symposium on Theoretical Aspects of Computer Science (STACS'99)* March 4-6, 1999, Trier, Germany.

†Bellcore, Morristown, NJ 07960. Email: aiello@wind.bellcore.com

‡Department of Computer Science, Brown University, Providence, RI 02915. Email: {cb, mph}@cs.brown.edu

§Department of Computer Science, University of Cyprus, Nicosia CY-1678, Cyprus. Partially supported by funds for the promotion of research at University of Cyprus. Part of the work of this author was performed while at AT&T Labs – Research, Florham Park, NJ, as a visitor to the Special Year on Networks, DIMACS Center for Discrete Mathematics and Theoretical Computer Science, Piscataway, NJ. Email: mavronic@turing.cs.ucy.ac.cy

¶Department of Computer Science, School of Mathematical Sciences, Tel-Aviv University, Tel-Aviv 69978, Israel. Email: shanir@math.tau.ac.il

||IDC Herzliya, Tel-Aviv, Israel. Email: danidin@math.tau.ac.il

such as *K-smoothing*. In general, we identify a broad class of properties, which we call *boundedness properties,* that are preserved by the introduction of decrements: if a balancing network satisfies a particular boundedness property for increments alone, then it continues to satisfy that property for both increments and decrements.

Our proofs are purely combinatorial and rely on the novel concept of a *fooling pair* of input vectors that we introduce.

1 Introduction

1.1 Motivation-Overview

Counting networks were originally introduced by Aspnes *et al.* [4] and subsequently extended in [1, 13, 15]. They are designed to provide highly concurrent implementations of shared *Fetch&Increment* counters, shared pools and stacks, load balancing modules, and software barriers (see, e.g., [4, 16, 18, 23]).

Counting networks are constructed from basic computing elements called *balancers*. On an abstract level, a balancer can be thought of as a routing switch for elements called *tokens*. It has a collection of input wires and a collection of output wires, respectively called the balancer's *fan-in* and *fan-out*. Tokens arrive asynchronously on arbitrary input wires, and are routed to successive output wires in a "round-robin" fashion. If one thinks of a balancer as having a state 'toggle" variable tracking which output wire the next token should exit on, then a token traversal amounts to a *Fetch&Toggle* operation, retrieving the value of the output wire and changing the toggle state to point to the next wire. The distribution of tokens on the output wires of a balancer thus satisfies the *step property* [4]: if y_i tokens exit on output wire i, then $0 \leq y_i - y_j \leq 1$ for any $j > i$.

A *balancing network* is a network of balancers, constructed by connecting balancers' output wires with other balancers' input wires in an acyclic fashion, in a way similar to the way *comparator networks* are constructed from *comparators* [9, Chapter 28]. The network itself has a number of input and output wires. A token enters the network on an input wire, traverses a sequence of balancers, and exits on an output wire. A balancing network is a *K-smoothing network* [1, 4] if, when all tokens have exited the network, the difference between the maximum and minimum number of tokens that exit on any output wire is bounded by K, regardless of the distribution of input tokens. Smoothing networks can be used for distributed load balancing.

A 1-smoothing network is a *counting network* if it satisfies the same step property as a balancer: when all tokens have traversed the network, if y_i tokens exit on output wire i, then $0 \leq y_i - y_j \leq 1$ for any $j > i$. Counting networks can be used to implement *Fetch&Increment* counters: the l-th token to exit on the j-th output wire returns the value $j + (l-1)w_{out}$, where w_{out} is the network's fan-out.

A limitation of counting networks is that they support increments but not decrements. Many synchronization algorithms and tools require the ability to decrement shared objects. For example, the classical synchronization constructs of *semaphores* [12], *critical-regions* [17], and *monitors* [14] all rely on applying both increment and decrement operations on shared counters for both correctness and efficiency (see, e.g., [25, Chapter 6]). Moreover, several concurrent algorithms for classical multiprocessor synchronization problems such as the *mutual exclusion problem* [11] and the *readers-writers problem* [10], involve coordination using both the incrementation and the decrementation of shared counters.

Shavit and Touitou [23] provided the first counting network algorithm to support decrements for the class of networks that have the layout of a binary tree. They did so by introducing a new type of token for the decrement operation, which they named the *antitoken*.* Unlike a token, which traverses a balancer by fetching the toggle value and then advancing it, an antitoken sets the toggle back and then fetches it. Informally, an antitoken "cancels" the effect of the most recent token on the balancer's toggle state, and vice versa. In the same paper, Shavit and Touitou introduced the *gap step property,* as an extension to the step property correctness criterion, intended to capture correctness when there are both tokens and antitokens traversing the network. They provide an operational proof that *counting trees* [24] satisfy the gap step property when traversed by tokens and antitokens.

Shavit and Touitou [23] also introduced the notion of "elimination," which, they show, can be used to implement a highly concurrent "pool" or "stack" of items using a counting tree. This is done by having each token represent an enqueue opeartion of an item, and each antitoken represent a dequeue operation, a request to take out an item. If a token and an antitoken meet at a balancer in the counting tree, they can "eliminate" one another; that is, the process shepherding the token can hand the value to the process shepherding the antitoken, and both processes can exit without need to traverse the rest of the tree. Even with "elimination," the counting

*The name was actually suggested by Yehuda Afek (personal communication).

tree still preserves the desired gap step property.

It is natural to ask whether the same properties hold for *arbitrary* counting networks.[†] More generally, what properties of balancing networks are preserved by the introduction of antitokens? In this paper, we give the first general answer to this question. We show the following results.

- If a balancing network is a counting network when inputs are tokens, then it remains a counting network when inputs include both tokens and antitokens. This result implies that *any* counting network can be extended to support a *Fetch&Decrement* operation.

- Any counting network, not just elimination trees, permits tokens and antitokens to eliminate one another when implementing a concurrent pool.

- If a balancing network is a K-smoothing network when inputs are tokens, then it remains a K-smoothing network when inputs include both tokens and antitokens.

- More generally, we identify a broad class of properties, which we call *boundedness properties,* that are preserved by the introduction of antitokens: if a balancing network satisfies a particular boundedness property when inputs are tokens, then it continues to satisfy that property when inputs include both tokens and antitokens. The step property and the K-smoothing property are examples of boundedness properties.

An interesting aspect of our work is that, unlike [23], our proofs are combinatorial, not operational. They rely on the novel concept of a *fooling pair* of input vectors, which, we believe, is of independent interest.

1.2 Our Techniques and Main Result

Aspnes *et al.* [4] introduced the convention of assigning the value 1 to each token, a convention maintained by later constructions [1, 5, 8, 13, 15, 19, 20, 24]. Shavit and Touitou [23] assigned the value 1 to both tokens and antitokens, while embedding the cancelling nature of antitokens in the gap

[†]This is especially so since efficient (low contention) implementations of tree-shaped counting networks require the use of randomized "diffraction" mechnisms [24], which cannot be "hardwired." Having a hardwired (fixed layout) network is significant for hardware implementations, and for systems with real-time constraints.

step property. Here, however, it is convenient to assign the value -1 to each antitoken, and 1 to each token.

Generalizing the approach taken by Aspnes *et al.* [4], we represent a balancer as an "operator" carrying an integer *input vector* to an integer *output vector*. The i-th entry in the input vector represents the *algebraic sum* of tokens and antitokens received on the i-th input wire, and similarly for the output vector. For example, if this value is zero, then the same number of tokens and antitokens have arrived on that wire. In the original definition of counting networks, which permitted only tokens, all such values were non-negative. We treat a balancing network in the same way, as an "operator" on integer vectors.

We show that any balancing network satisfying the step property for non-negative, integer input vectors, also satisfies the step property for arbitrary integer input vectors. implying that any counting network can be extended to support decrements. Moreover, we show that any balancing network that satisfies the K-smoothing property for non-negative, integer input vectors, satisfies the same property for arbitrary integer input vectors. In fact, we show the following general result. Think of a set of possible output vectors as defining a *property* of a balancing network. A *boundedness property* satisfies two conditions:

- it is a subset of the K-smoothing property, for some $K \geq 1$, and
- it is closed under the addition of any *constant* vector.

Both the K-smoothing and the step property are examples of boundedness properties. Our principal result is that any balancing network that satisfies a boundedness property when its input vector consists only of non-negative integers, it will continue to satisfy that boundedness property even when its input vector consists of arbitrary integers.

To prove our result, we introduce the concept of a fooling pair of input vectors. Let the *state* of any given balancer, viewed as a toggle mechanism, be the "position" of its toggle. Say that two input vectors are a *fooling pair* to any given balancer if they "drive" the balancer, starting from the same state, to identical states. The state of a balancing network is defined as the collection of the states of its balancers. Naturally, two input vectors are a *fooling pair* to the balancing network if they drive it, starting from the same state, to identical states. For a specific initial state of a balancing network, fooling pairs partition the set of input vectors into equivalence classes. We establish interesting combinatorial properties of fooling pairs.

Roughly speaking, we prove our main equivalence result as follows. Consider any balancing network with some boundedness property; take any arbitrary, integer input vector and the corresponding integer output vector. By adding to the input vector an appropriate vector that belongs to the equivalence class for some given initial state, we obtain a new input vector such that all of its entries are non-negative integers. We show that the output vector corresponding to the new input vector is, in fact, equal to the original output vector plus a constant vector. Hence, our main equivalence result follows from closure of the boundedness property under addition with a constant vector.

The rest of this paper is organized as follows. Section 2 provides a framework for our discussion. Fooling pairs and their properties are treated in Section 3. Section 4 presents our main equivalence result. We conclude, in Section 5, with a discussion and some open problems.

2 Framework

In this section, we present some general definitions, which extend those in [1, 8, 13, 15] to account for both tokens and antitokens; they somehow simplify and formalize corresponding ones in [23].

This section is organized as follows. Section 2.1 specifies some notation. Balancers and balancing networks are introduced in Sections 2.2 and 2.3, respectively. Section 2.4 treats boundedness properties.

2.1 Notation

For any integer $g \geq 2$, $\mathbf{x}^{(g)}$ denotes the vector $\langle x_0, x_1, \ldots, x_{g-1}\rangle^{\mathrm{T}}$, while $\lceil \mathbf{x}^{(g)} \rceil$ denotes the integer vector $\langle \lceil x_0 \rceil, \lceil x_1 \rceil, \ldots, \lceil x_{g-1} \rceil \rangle^{\mathrm{T}}$. For any vector $\mathbf{x}^{(g)}$, denote $\|\mathbf{x}\|_1 = \sum_{i=0}^{g-1} x_i$. We use $\mathbf{0}^{(g)}$ to denote $\langle 0, 0, \ldots, 0\rangle^{\mathrm{T}}$, a vector with g zero entries; similarly, we use $\mathbf{1}^{(g)}$ to denote $\langle 1, 1, \ldots, 1\rangle^{\mathrm{T}}$, a vector with g unit entries. We use $\mathbf{r}^{(g)}$ to denote the *ramp vector* $\langle 0, 1, \ldots, g-1\rangle^{\mathrm{T}}$. A *constant vector* is any vector of the form $c\,\mathbf{1}^{(g)}$, for any constant c.

For any integer x and positive integer δ, denote x div δ and x mod δ the integer quotient and remainder, respectively, of the division of x by δ; note that $0 \leq x \bmod \delta \leq \delta - 1$ and $x = (x \operatorname{div} \delta)\,\delta + x \bmod \delta$. Say that δ *divides* $\mathbf{x}^{(g)}$ if δ divides each entry of $\mathbf{x}^{(g)}$.

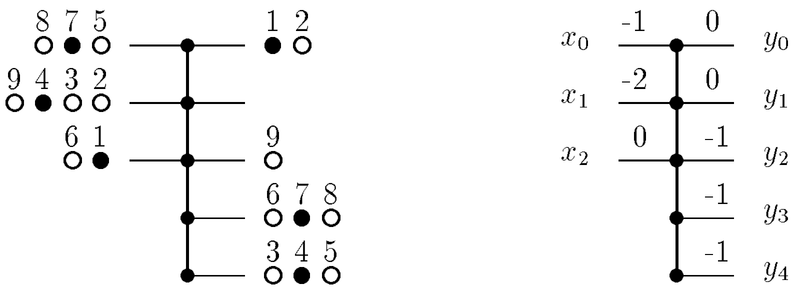

Figure 1: A balancer

2.2 Balancers

Balancing networks are constructed from acyclically wired elements, called balancers, that route *tokens* and *antitokens* through the network, and *wires.* For the sake of generality, our balancers are defined as "multibalancers," in the style of Aharonson and Attiya [1], Felten *et al.* [13], and Hardavellas *et al.* [15]; however, we follow Shavit and Touitou [23] to insist that our balancers handle both tokens and antitokens. We think of a token and an antitoken as the basic "positive" and "negative" unit, respectively, that are routed through the network.

So, for any pair of positive integers f_{in} and f_{out}, an (f_{in}, f_{out})-*balancer,* or *balancer* for short, is a routing element receiving tokens and antitokens on f_{in} input wires, numbered $0, 1, \ldots, f_{in} - 1$, and sending out tokens and antitokens to f_{out} output wires, numbered $0, 1, \ldots, f_{out} - 1$; f_{in} and f_{out} are called the balancer's *fan-in* and *fan-out,* respectively. Tokens and antitokens arrive on the balancer's input wires at arbitrary times, and they are output on its output wires. Roughly speaking, a balancer acts like a "generalized" *toggle,* which, on a stream of input tokens and antitokens, alternately forwards them to its output wires, going either down or up on each input token and antitoken, respectively. For clarity, we assume that all tokens and antitokens are distinct. Figure 1 depicts a balancer with three input wires and five output wires, stretched horizontally; the balancer is stretched vertically. In the left part, tokens and antitokens are denoted with full and empty circles, respectively; the numbering reflects the real-time order of tokens and antitokens in an execution where they traverse the balancer one by one (called a *sequential* execution).

For each input index i, $0 \leq i \leq f_{in} - 1$, we denote by x_i the *balancer input state variable* that stands for the algebraic sum of the numbers of tokens and antitokens that have entered on input wire i; that is, x_i is the number of tokens that have entered on input wire i *minus* the number of antitokens that have entered on input wire i. Denote $\mathbf{x}^{(f_{in})} = \langle x_0, x_1, \ldots, x_{f_{in}-1} \rangle^{\mathrm{T}}$; call $\mathbf{x}^{(w_{in})}$ an *input vector.* For each output index j, $0 \leq j \leq f_{out} - 1$, we denote by y_j the *balancer output state variable* that stands for the algebraic sum of the numbers of tokens and antitokens that have exited on output wire j; that is, y_j is the number of tokens that have exited on output wire j *minus* the number of antitokens that have exited on output wire j. The right part of Figure 1 shows the corresponding input and output state variables. Denote $\mathbf{y}^{(f_{out})} = \langle y_0, y_1, \ldots, y_{f_{out}-1} \rangle^{\mathrm{T}}$; call $\mathbf{y}^{(f_{out})}$ an *output vector.*

The *configuration* of a balancer at any given time is the tuple $\langle \mathbf{x}^{(f_{in})}, \mathbf{y}^{(f_{out})} \rangle$; roughly speaking, the configuration is the collection of its input and output state variables. In the *initial configuration,* all input and output wires are empty; that is, in the initial configuration, $\mathbf{x}^{(f_{in})} = \mathbf{0}^{(f_{in})}$, and $\mathbf{y}^{(f_{out})} = \mathbf{0}^{(f_{out})}$. A configuration of a balancer is *quiescent* if there are no tokens or antitokens in the balancer. Note that the initial configuration is a quiescent one. The following formal properties are required for an (f_{in}, f_{out})-balancer.

1. *Safety property:* in any configuration, a balancer never creates either tokens or antitokens spontaneously.

2. *Liveness property:* for any finite number t of tokens and t' of antitokens that enter the balancer, the balancer reaches within a finite amount of time a quiescent configuration where all the t tokens and t' antitokens have exited the network; that is, a balancer never "swallows" tokens or antitokens.

3. *Step property:* in any quiescent configuration, for any pair of output indices j and k such that $0 \leq j < k \leq f_{out} - 1$, $0 \leq y_j - y_k \leq 1$.

From the safety and liveness properties it follows that for any quiescent configuration $\langle \mathbf{x}^{(f_{in})}, \mathbf{y}^{(f_{out})} \rangle$ of a balancer $\|\mathbf{x}^{(f_{in})}\|_1 = \|\mathbf{y}^{(f_{out})}\|_1$; that is, in a quiescent configuration, the algebraic sum of tokens and antitokens that exited the balancer is equal to the algebraic sum of tokens and antitokens that entered it.

We are interested in quiescent configurations of a balancer. For any input vector $\mathbf{x}^{(f_{in})}$, denote $\mathbf{y}^{(f_{out})} = b(\mathbf{x}^{(f_{in})})$ the output vector in the quiescent configuration that b will reach after all $\|\mathbf{x}^{(f_{in})}\|_1$ tokens and antitokens that entered b have exited; write also $b : \mathbf{x}^{(f_{in})} \rightarrow \mathbf{y}^{(f_{out})}$ to denote the balancer b. The output vector can also be written [1, 4, 8] as

$$\mathbf{y}^{(f_{out})} \quad = \quad \left\lceil \frac{\|\mathbf{x}^{(f_{in})}\|_1 \, \mathbf{1}^{(f_{out})} - \mathbf{r}^{(f_{out})}}{f_{out}} \right\rceil .$$

For any quiescent configuration $\langle \mathbf{x}^{(f_{in})}, \mathbf{y}^{(f_{out})} \rangle$ of a balancer $b : \mathbf{x}^{(f_{in})} \rightarrow \mathbf{y}^{(f_{out})}$, the *state* of the balancer b, denoted $\mathrm{state}_b(\langle \mathbf{x}^{(f_{in})}, \mathbf{y}^{(f_{out})} \rangle)$, is defined to be

$$\mathrm{state}_b(\langle \mathbf{x}^{(f_{in})}, \mathbf{y}^{(f_{out})} \rangle) \quad = \quad \|\mathbf{x}^{(f_{in})}\|_1 \bmod f_{out} \, ;$$

since the configuration is quiescent, it follows that

$$\mathrm{state}_b(\langle \mathbf{x}^{(f_{in})}, \mathbf{y}^{(f_{out})} \rangle) \quad = \quad \|\mathbf{y}^{(f_{out})}\|_1 \bmod f_{out} \, .$$

Thus, for the sake of simplicity, we will denote

$$\mathrm{state}_b(\mathbf{x}^{(f_{in})}) \quad = \quad \mathrm{state}_b(\langle \mathbf{x}^{(f_{in})}, \mathbf{y}^{(f_{out})} \rangle) \, .$$

We remark that the state of an (f_{in}, f_{out})-balancer is some integer in the set $\{0, 1, \ldots, f_{out} - 1\}$, which captures the "position" to which it is set as a toggle mechanism. This integer is determined by either the balancer input state variables or the balancer output state variables in the quiescent configuration. Note that the state of the balancer in the initial configuration is 0. Moreover, the linearity of the modulo taking operation immediately implies the linearity property for the state of a balancer.

Lemma 2.1 *Consider a balancer $b : \mathbf{x}^{(f_{in})} \rightarrow \mathbf{y}^{(f_{out})}$. Then, for any input vectors $\mathbf{x}_1^{(f_{in})}$ and $\mathbf{x}_2^{(f_{in})}$,*

$$\mathit{state}_b(\mathbf{x}_1^{(f_{in})} + \mathbf{x}_2^{(f_{in})}) \quad = \quad (\mathit{state}_b(\mathbf{x}_1^{(f_{in})}) + \mathit{state}_b(\mathbf{x}_2^{(f_{in})})) \ \mathit{mod} \ f_{out} \, .$$

2.3 Balancing Networks

A (w_{in}, w_{out})*-balancing network* $\mathcal{B}$ is a collection of interwired balancers, where output wires are connected to input wires, having w_{in} designated input wires, numbered $0, 1, \ldots, w_{in} - 1$, which are not connected to output wires of balancers, having w_{out} designated output wires, numbered $0, 1, \ldots, w_{out} - 1$, similarly not connected to input wires of balancers, and containing no cycles. Tokens and antitokens arrive on the network's input

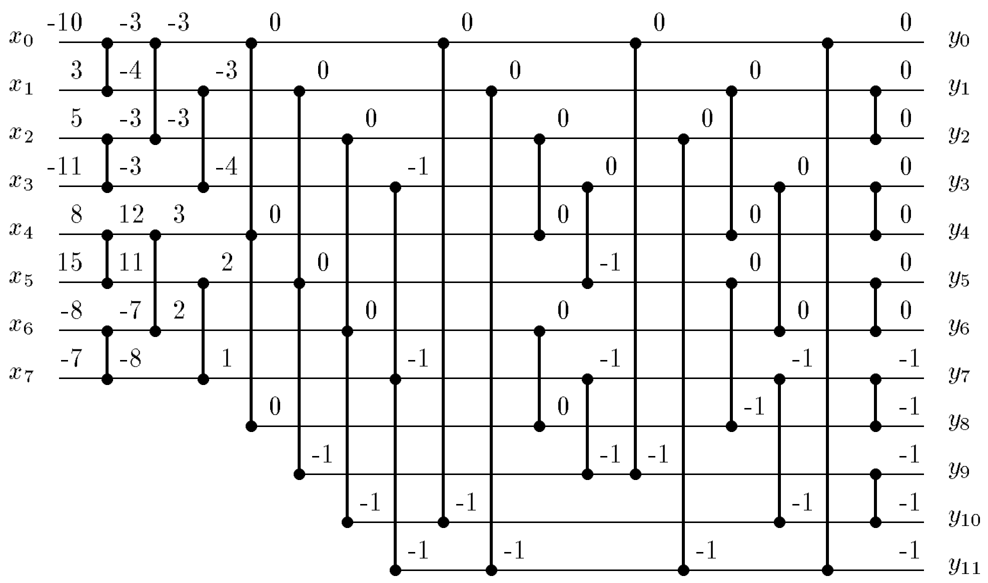

Figure 2: A balancing network

wires at arbitrary times, and they traverse a sequence of balancers in the network in a completely asynchronous way till they exit on the output wires of the network. Figure 2 depicts a balancing network with eight input wires and twelve output wires, using the same conventions as in Figure 1.

For each input index i, $0 \leq i \leq w_{in} - 1$, we denote by x_i the *network input state variable* that stands for the algebraic sum of the numbers of tokens and antitokens that have entered on input wire i; that is, x_i is the difference of the number of tokens that have entered on input wire i *minus* the number of antitokens that have entered on input wire i. Denote $\mathbf{x}^{(w_{in})} = \langle x_0, x_1, \ldots, x_{w_{in}-1}\rangle^{\mathrm{T}}$; call $\mathbf{x}^{(w_{in})}$ an *input vector.* For each output index j, $0 \leq j \leq w_{out} - 1$, we denote by y_j the *network output state variable* that stands for the algebraic sum of the numbers of tokens and antitokens that have exited on output wire j; that is, y_j is the number of tokens that have exited on output wire j *minus* the number of antitokens that have exited on output wire i. Denote $\mathbf{y}^{(w_{out})} = \langle y_0, y_1, \ldots, y_{w_{in}-1}\rangle^{\mathrm{T}}$; call $\mathbf{y}^{(w_{out})}$ an *output vector.*

The *configuration* of a network at any given time is the tuple of configurations of its individual balancers. In the *initial configuration,* all input and output wires of balancers are empty. The safety and liveness property for a balancing network follow naturally from those of its balancers. Thus, a balancing network eventually reaches a *quiescent configuration* in which all tokens and antitokens that entered the network have exited. In any quiescent configuration of $\mathcal{B}$ we have $\|\mathbf{x}^{(w_{in})}\|_1 = \|\mathbf{y}^{(w_{out})}\|_1$; that is, in a quiescent configuration, the algebraic sum of tokens and antitokens that exited the network is equal to the algebraic sum of tokens and antitokens that entered it.

Naturally, we are interested in quiescent configurations of a network. For any quiescent configuration of a network $\mathcal{B}$ with corresponding input and output vectors $\mathbf{x}^{(w_{in})}$ and $\mathbf{y}^{(w_{out})}$, respectively, the *state* of $\mathcal{B}$, denoted $\text{state}_{\mathcal{B}}(\mathbf{x}^{(w_{in})})$, is defined to be the collection of the states of its individual balancers. We remark that we have specified $\mathbf{x}^{(w_{in})}$ as the single argument of $\text{state}_{\mathcal{B}}$, since $\mathbf{x}^{(w_{in})}$ uniquely determines all input and output vectors of balancers of $\mathcal{B}$, which are used for defining the states of the individual balancers. Note that the state of the network in its initial configuration is a collection of 0's. For any input vector $\mathbf{x}^{(w_{in})}$, denote $\mathbf{y}^{(w_{out})} = \mathcal{B}(\mathbf{x}^{(w_{in})})$ the output vector in the quiescent configuration that $\mathcal{B}$ will reach after all $\|\mathbf{x}^{(w_{in})}\|_1$ tokens and antitokens that entered $\mathcal{B}$ have exited; write also $\mathcal{B} : \mathbf{x}^{(w_{in})} \rightarrow \mathbf{y}^{(w_{out})}$ to denote the network $\mathcal{B}$.

Not all balancing networks satisfy the step property. A (w_{in}, w_{out})-*counting network* is a (w_{in}, w_{out})-balancing network for which, in any quiescent configuration, for any pair of indices j and k such that $0 \leq j < k \leq w_{out} - 1$, $0 \leq y_j - y_k \leq 1$; that is, the output of a counting network has the step property. For example, the network depicted in Figure 2 is a counting network [8], where the integers on the wires represent the input and output state variables of the balancers that correspond to some particular input vector.

The definition of a counting network can be weakened as follows [1, 4]. For any integer $K \geq 1$, a (w_{in}, w_{out})-*K-smoothing network* is a (w_{in}, w_{out})-balancing network for which, in any quiescent configuration, for any pair of indices j and k such that $0 \leq j, k \leq w_{out} - 1$, $0 \leq |y_j - y_k| \leq K$; that is, the output vector of a K-smoothing network has the *K-smoothing property*: all outputs are within K to each other.

For a balancing network $\mathcal{B}$, the *depth* of $\mathcal{B}$, denoted $\text{depth}(\mathcal{B})$, is defined to be the maximum distance from any of its input wires to any of its output wires. In case $\text{depth}(\mathcal{B}) = 1$, $\mathcal{B}$ will be called a *layer*. If $\text{depth}(\mathcal{B}) = d$ is greater than one, then $\mathcal{B}$ can be uniquely partitioned into layers $\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_d$ from left to right in the obvious way.

2.4 Boundedness Properties

Fix any integer $g \geq 2$. For any integer $K \geq 1$, the *K-smoothing property* [1] is defined to be the set of all vectors $\mathbf{y}^{(g)}$ such that for any entries y_j and y_k of $\mathbf{y}^{(g)}$, where $0 \leq j, k \leq g-1$, $|y_j - y_k| \leq K$. A *boundedness property* is any subset of some K-smoothing property, for any integer $K \geq 1$, that is closed under addition with a constant vector. Clearly, the K-smoothing property is trivially a boundedness property; moreover, the set of all vectors $\mathbf{y}^{(g)}$ that have the *step property* [4] is a boundedness property, since any step vector is 1-smooth (but not vice versa). We remark that there are infinitely many boundedness properties.

A boundedness property captures precisely the two properties possessed by both K-smooth and step vectors upon which our later proofs will rely. Although we are unaware of any interesting property, other than the K-smoothing and step, that is a boundedness one, we chose to state our results for any general boundedness property in order to make explicit the two critical properties that are common to the classes of K-smooth vectors and step vectors; moreover, arguing in terms of a boundedness property will allow for a single proof of all claims found to hold for both the K-smoothing property and the step property.

Say that *a vector* $\mathbf{y}$ *has the boundedness property* $\mathbf{\Pi}$ if $\mathbf{y} \in \mathbf{\Pi}$. Say that *a balancing network* $\mathcal{B} : \mathbf{x}^{(w_{in})} \rightarrow \mathbf{y}^{(w_{out})}$ *has the boundedness property* $\mathbf{\Pi}$ if for every input vector $\mathbf{x}^{(w_{in})}$, $\mathcal{B}(\mathbf{x}^{(w_{in})}) \in \mathbf{\Pi}$.

3 Fooling Pairs

In this section, we introduce fooling pairs and prove several properties of them.

Say that input vectors $\mathbf{x}_1^{(f_{in})}$ and $\mathbf{x}_2^{(f_{in})}$ are a *fooling pair to balancer* $b : \mathbf{x}^{(f_{in})} \rightarrow \mathbf{y}^{(f_{out})}$ if

$$\text{state}_b(\mathbf{x}_1^{(f_{in})}) \quad = \quad \text{state}_b(\mathbf{x}_2^{(f_{in})})\ ;$$

roughly speaking, a fooling pair "drives" the balancer to identical states in the two corresponding quiescent configurations. We start by showing:

Proposition 3.1 *Consider a balancer $b : \mathbf{x}^{(f_{in})} \rightarrow \mathbf{y}^{(f_{out})}$. Take any input vectors $\mathbf{x}_1^{(f_{in})}$ and $\mathbf{x}_2^{(f_{in})}$ that are a fooling pair to balancer b. Then, for any input vector $\mathbf{x}^{(f_{in})}$,*

(1) *the input vectors* $\mathbf{x}_1^{(f_{in})} + \mathbf{x}^{(f_{in})}$ *and* $\mathbf{x}_2^{(f_{in})} + \mathbf{x}^{(f_{in})}$ *are a fooling pair to balancer* b*;*

(2) $b(\mathbf{x}_1^{(f_{in})} + \mathbf{x}^{(f_{in})}) - b(\mathbf{x}_1^{(f_{in})}) = b(\mathbf{x}_2^{(f_{in})} + \mathbf{x}^{(f_{in})}) - b(\mathbf{x}_2^{(f_{in})})$.

Proof: We first show (1). Clearly, by Lemma 2.1,

$$\text{state}_b(\mathbf{x}_1^{(f_{in})} + \mathbf{x}^{(f_{in})}) = (\text{state}_b(\mathbf{x}_1^{(f_{in})}) + \text{state}_b(\mathbf{x}^{(f_{in})})) \bmod f_{out}\,,$$

and

$$\text{state}_b(\mathbf{x}_2^{(f_{in})} + \mathbf{x}^{(f_{in})}) = (\text{state}_b(\mathbf{x}_2^{(f_{in})}) + \text{state}_b(\mathbf{x}^{(f_{in})})) \bmod f_{out}\,.$$

Since $\mathbf{x}_1^{(f_{in})}$ and $\mathbf{x}_2^{(f_{in})}$ are a fooling pair to b, $\text{state}_b(\mathbf{x}_1^{(f_{in})}) = \text{state}_b(\mathbf{x}_2^{(f_{in})})$, it follows that

$$\text{state}_b(\mathbf{x}_1^{(f_{in})} + \mathbf{x}^{(f_{in})}) = \text{state}_b(\mathbf{x}_2^{(f_{in})} + \mathbf{x}^{(f_{in})});$$

thus, $\mathbf{x}_1^{(f_{in})} + \mathbf{x}^{(f_{in})}$ and $\mathbf{x}_2^{(f_{in})} + \mathbf{x}^{(f_{in})}$ are a fooling pair to b, as needed.

We continue to show (2). Clearly,

$$\begin{aligned}
& b(\mathbf{x}_1^{(f_{in})} + \mathbf{x}^{(f_{in})}) - b(\mathbf{x}_1^{(f_{in})}) \\
= {} & \left\lceil \frac{\|\mathbf{x}_1^{(f_{in})} + \mathbf{x}^{(f_{in})}\|_1\, \mathbf{1}^{(f_{out})} - \mathbf{r}^{(f_{out})}}{f_{out}} \right\rceil - \left\lceil \frac{\|\mathbf{x}_1^{(f_{in})}\|_1\, \mathbf{1}^{(f_{out})} - \mathbf{r}^{(f_{out})}}{f_{out}} \right\rceil \\
= {} & \left\lceil \frac{\|\mathbf{x}_1^{(f_{in})}\|_1\, \mathbf{1}^{(f_{out})} + \|\mathbf{x}^{(f_{in})}\|_1\, \mathbf{1}^{(f_{out})} - \mathbf{r}^{(f_{out})}}{f_{out}} \right\rceil - \left\lceil \frac{\|\mathbf{x}_1^{(f_{in})}\|_1\, \mathbf{1}^{(f_{out})} - \mathbf{r}^{(f_{out})}}{f_{out}} \right\rceil \\
= {} & \left\lceil \frac{(\|\mathbf{x}_1^{(f_{in})}\|_1 \text{ div } f_{out})\, f_{out}\, \mathbf{1}^{(f_{out})} + (\|\mathbf{x}_1^{(f_{in})}\|_1 \bmod f_{out})\, \mathbf{1}^{(f_{out})} + \|\mathbf{x}^{(f_{in})}\|_1\, \mathbf{1}^{(f_{out})} - \mathbf{r}^{(f_{out})}}{f_{out}} \right\rceil - \\
& \left\lceil \frac{(\|\mathbf{x}_1^{(f_{in})}\|_1 \text{ div } f_{out})\, f_{out}\, \mathbf{1}^{(f_{out})} + (\|\mathbf{x}_1^{(f_{in})}\|_1 \bmod f_{out})\, \mathbf{1}^{(f_{out})} - \mathbf{r}^{(f_{out})}}{f_{out}} \right\rceil \\
= {} & \left\lceil \frac{(\|\mathbf{x}_1^{(f_{in})}\|_1 \text{ div } f_{out})\, f_{out}\, \mathbf{1}^{(f_{out})} + \text{state}_b(\mathbf{x}_1^{(f_{in})})\, \mathbf{1}^{(f_{out})} + \|\mathbf{x}^{(f_{in})}\|_1\, \mathbf{1}^{(f_{out})} - \mathbf{r}^{(f_{out})}}{f_{out}} \right\rceil - \\
& \left\lceil \frac{(\|\mathbf{x}_1^{(f_{in})}\|_1 \text{ div } f_{out})\, f_{out} \mathbf{1}^{(f_{out})} + \text{state}_b(\mathbf{x}_1^{(f_{in})})\, \mathbf{1}^{(f_{out})} - \mathbf{r}^{(f_{out})}}{f_{out}} \right\rceil \\
& \text{(by definition of state}_b) \\
= {} & \left\lceil (\|\mathbf{x}_1^{(f_{in})}\|_1 \text{ div } f_{out})\, \mathbf{1}^{(f_{out})} + \frac{\text{state}_b(\mathbf{x}_1^{(f_{in})})\, \mathbf{1}^{(f_{out})} + \|\mathbf{x}^{(f_{in})}\|_1\, \mathbf{1}^{(f_{out})} - \mathbf{r}^{(f_{out})}}{f_{out}} \right\rceil -
\end{aligned}$$

$$\begin{aligned}
& \left\lceil (\|\mathbf{x}_1^{(f_{in})}\|_1 \text{ div } f_{out})\, \mathbf{1}^{(f_{out})} + \frac{\text{state}_b(\mathbf{x}_1^{(f_{in})})\, \mathbf{1}^{(f_{out})} - \mathbf{r}^{(f_{out})}}{f_{out}} \right\rceil \\
= \ & (\|\mathbf{x}_1^{(f_{in})}\|_1 \text{ div } f_{out})\, \mathbf{1}^{(f_{out})} + \left\lceil \frac{\text{state}_b(\mathbf{x}_1^{(f_{in})})\, \mathbf{1}^{(f_{out})} + \|\mathbf{x}^{(f_{in})}\|_1\, \mathbf{1}^{(f_{out})} - \mathbf{r}^{(f_{out})}}{f_{out}} \right\rceil - \\
& (\|\mathbf{x}_1^{(f_{in})}\|_1 \text{ div } f_{out})\, \mathbf{1}^{(f_{out})} - \left\lceil \frac{\text{state}_b(\mathbf{x}_1^{(f_{in})})\, \mathbf{1}^{(f_{out})} - \mathbf{r}^{(f_{out})}}{f_{out}} \right\rceil \\
= \ & \left\lceil \frac{\text{state}_b(\mathbf{x}_1^{(f_{in})})\, \mathbf{1}^{(f_{out})} + \|\mathbf{x}^{(f_{in})}\|_1\, \mathbf{1}^{(f_{out})} - \mathbf{r}^{(f_{out})}}{f_{out}} \right\rceil - \left\lceil \frac{\text{state}_b(\mathbf{x}_1^{(f_{in})})\, \mathbf{1}^{(f_{out})} - \mathbf{r}^{(f_{out})}}{f_{out}} \right\rceil .
\end{aligned}$$

Similarly, we can show that

$$\begin{aligned}
& b(\mathbf{x}_2^{(f_{in})} + \mathbf{x}^{(f_{in})}) - b(\mathbf{x}_2^{(f_{in})}) \\
= \ & \left\lceil \frac{\text{state}_b(\mathbf{x}_2^{(f_{in})})\, \mathbf{1}^{(f_{out})} + \|\mathbf{x}^{(f_{in})}\|_1\, \mathbf{1}^{(f_{out})} - \mathbf{r}^{(f_{out})}}{f_{out}} \right\rceil - \left\lceil \frac{\text{state}_b(\mathbf{x}_2^{(f_{in})})\, \mathbf{1}^{(f_{out})} - \mathbf{r}^{(f_{out})}}{f_{out}} \right\rceil .
\end{aligned}$$

Since $\mathbf{x}_1^{(f_{in})}$ and $\mathbf{x}_2^{(f_{in})}$ are a fooling pair to b, $\text{state}_b(\mathbf{x}_1^{(f_{in})}) = \text{state}_b(\mathbf{x}_2^{(f_{in})})$. It follows that

$$b(\mathbf{x}_1^{(f_{in})} + \mathbf{x}^{(f_{in})}) - b(\mathbf{x}_1^{(f_{in})}) \quad = \quad b(\mathbf{x}_2^{(f_{in})} + \mathbf{x}^{(f_{in})}) - b(\mathbf{x}_2^{(f_{in})}) ,$$

as needed. ■

We continue to extend the concept of a fooling pair from a single balancer to a network. Say that input vectors $\mathbf{x}_1^{(w_{in})}$ and $\mathbf{x}_2^{(w_{in})}$ are *a fooling pair to network* $\mathcal{B} : \mathbf{x}^{(w_{in})} \rightarrow \mathbf{y}^{(w_{out})}$ if for each balancer b of $\mathcal{B}$, the input vectors of b in quiescent configurations corresponding to $\mathbf{x}_1^{(w_{in})}$ and $\mathbf{x}_2^{(w_{in})}$, respectively, are a fooling pair to b; roughly speaking, a fooling pair "drives" all balancers of the network to identical states in the two corresponding quiescent configurations. We continue to generalize Proposition 3.1 to the case of a balancing network.

Proposition 3.2 *Consider a balancing network* $\mathcal{B} : \mathbf{x}^{(w_{in})} \rightarrow \mathbf{y}^{(w_{out})}$*. Take any input vectors* $\mathbf{x}_1^{(w_{in})}$ *and* $\mathbf{x}_2^{(w_{in})}$ *that are a fooling pair to network* $\mathcal{B}$*. Then, for any input vector* $\mathbf{x}^{(w_{in})}$*,*

(1) *the input vectors* $\mathbf{x}_1^{(w_{in})} + \mathbf{x}^{(w_{in})}$ *and* $\mathbf{x}_2^{(w_{in})} + \mathbf{x}^{(w_{in})}$ *are a fooling pair to network* $\mathcal{B}$*;*

(2) $\mathcal{B}(\mathbf{x}_1^{(w_{in})} + \mathbf{x}^{(w_{in})}) - \mathcal{B}(\mathbf{x}_1^{(w_{in})}) = \mathcal{B}(\mathbf{x}_2^{(w_{in})} + \mathbf{x}^{(w_{in})}) - \mathcal{B}(\mathbf{x}_2^{(w_{in})})$.

Proof: By induction on the depth d of the network $\mathcal{B}$. For the basis case, where $d = 1$, $\mathcal{B}$ is a single layer, and the claim follows immediately by applying Proposition 3.1 to each of the balancers of B separately.

Assume inductively that the claim holds for all networks of depth at most $d-1$. For the induction step, let $\mathcal{B} = \mathcal{B}'\mathcal{B}''$, where $\mathcal{B}' : \mathbf{x}^{(w_{in})} \to \mathbf{z}^{(w_{med})}$ and $\mathcal{B}'' : \mathbf{z}^{(w_{med})} \to \mathbf{y}^{(w_{out})}$ are networks of depth at most $d-1$; that is, $\mathcal{B}$ is the "cascade" of $\mathcal{B}'$ and $\mathcal{B}''$.

For the induction step, we start by showing (1). Since the input vectors $\mathbf{x}_1^{(w_{in})}$ and $\mathbf{x}_2^{(w_{in})}$ are a fooling pair to $\mathcal{B}$, it follows that they are a fooling pair to $\mathcal{B}'$. Thus, by induction hypothesis (1) for $\mathcal{B}'$, for any input vector $\mathbf{x}^{(w_{in})}$, the input vectors $\mathbf{x}_1^{(w_{in})} + \mathbf{x}^{(w_{in})}$ and $\mathbf{x}_2^{(w_{in})} + \mathbf{x}^{(w_{in})}$ are a fooling pair to $\mathcal{B}'$. It remains to show that the vectors $\mathcal{B}'(\mathbf{x}_1^{(w_{in})} + \mathbf{x}^{(w_{in})})$ and $\mathcal{B}'(\mathbf{x}_2^{(w_{in})} + \mathbf{x}^{(w_{in})})$ are a fooling pair to $\mathcal{B}''$.

By induction hypothesis (2) for $\mathcal{B}'$,

$$\mathcal{B}'(\mathbf{x}_1^{(w_{in})} + \mathbf{x}^{(w_{in})}) \quad = \quad \mathcal{B}'(\mathbf{x}_1^{(w_{in})}) + \mathcal{B}'(\mathbf{x}_2^{(w_{in})} + \mathbf{x}^{(w_{in})}) - \mathcal{B}'(\mathbf{x}_2^{(w_{in})}) \, ,$$

and

$$\mathcal{B}'(\mathbf{x}_2^{(w_{in})} + \mathbf{x}^{(w_{in})}) \quad = \quad \mathcal{B}'(\mathbf{x}_2^{(w_{in})}) + \mathcal{B}'(\mathbf{x}_1^{(w_{in})} + \mathbf{x}^{(w_{in})}) - \mathcal{B}'(\mathbf{x}_1^{(w_{in})}) \, .$$

Furthermore, by induction hypothesis (2) for $\mathcal{B}'$,

$$\mathcal{B}'(\mathbf{x}_2^{(w_{in})} + \mathbf{x}^{(w_{in})}) - \mathcal{B}'(\mathbf{x}_2^{(w_{in})}) \quad = \quad \mathcal{B}'(\mathbf{x}_1^{(w_{in})} + \mathbf{x}^{(w_{in})}) - \mathcal{B}'(\mathbf{x}_1^{(w_{in})}) \, ,$$

while, by assumption, $\mathcal{B}'(\mathbf{x}_1^{(w_{in})})$ and $\mathcal{B}'(\mathbf{x}_2^{(w_{in})})$ are a fooling pair for $\mathcal{B}''$. We apply induction hypothesis (1) to $\mathcal{B}''$, taking $\mathcal{B}'(\mathbf{x}_1^{(w_{in})})$ for $\mathbf{x}_1^{(w_{in})}$, $\mathcal{B}'(\mathbf{x}_2^{(w_{in})})$ for $\mathbf{x}_2^{(w_{in})}$, and $\mathcal{B}'(\mathbf{x}_2^{(w_{in})} + \mathbf{x}^{(w_{in})}) - \mathcal{B}'(\mathbf{x}_2^{(w_{in})}) = \mathcal{B}'(\mathbf{x}_1^{(w_{in})} + \mathbf{x}^{(w_{in})}) - \mathcal{B}'(\mathbf{x}_1^{(w_{in})})$ for $\mathbf{x}^{(w_{in})}$; it implies that the input vectors

$$\mathcal{B}'(\mathbf{x}_1^{(w_{in})}) + \mathcal{B}'(\mathbf{x}_2^{(w_{in})} + \mathbf{x}^{(w_{in})}) - \mathcal{B}'(\mathbf{x}_2^{(w_{in})}) \quad = \quad \mathcal{B}'(\mathbf{x}_1^{(w_{in})} + \mathbf{x}^{(w_{in})}) \, ,$$

and

$$\mathcal{B}'(\mathbf{x}_2^{(w_{in})}) + \mathcal{B}'(\mathbf{x}_1^{(w_{in})} + \mathbf{x}^{(w_{in})}) - \mathcal{B}'(\mathbf{x}_1^{(w_{in})}) \quad = \quad \mathcal{B}'(\mathbf{x}_2^{(w_{in})} + \mathbf{x}^{(w_{in})})$$

are a fooling pair to $\mathcal{B}''$, as needed.

We continue to show (2). Since the input vectors $\mathbf{x}_1^{(w_{in})}$ and $\mathbf{x}_2^{(w_{in})}$ are a fooling pair to $\mathcal{B}$, it follows that they are a fooling pair to $\mathcal{B}'$. So, by induction hypothesis (2) for $\mathcal{B}'$,

$$\mathcal{B}'(\mathbf{x}_1^{(w_{in})} + \mathbf{x}^{(w_{in})}) \quad = \quad \mathcal{B}'(\mathbf{x}_1^{(w_{in})}) + \mathcal{B}'(\mathbf{x}_2^{(w_{in})} + \mathbf{x}^{(w_{in})}) - \mathcal{B}'(\mathbf{x}_2^{(w_{in})})\,.$$

Thus,

$$\begin{aligned}
& \mathcal{B}(\mathbf{x}_1^{(w_{in})} + \mathbf{x}^{(w_{in})}) - \mathcal{B}(\mathbf{x}_1^{(w_{in})}) \\
= \;& \mathcal{B}''(\mathcal{B}'(\mathbf{x}_1^{(w_{in})} + \mathbf{x}^{(w_{in})})) - \mathcal{B}''(\mathcal{B}'(\mathbf{x}_1^{(w_{in})})) \\
= \;& \mathcal{B}''(\mathcal{B}'(\mathbf{x}_1^{(w_{in})}) + \mathcal{B}'(\mathbf{x}_2^{(w_{in})} + \mathbf{x}^{(w_{in})}) - \mathcal{B}'(\mathbf{x}_2^{(w_{in})})) - \mathcal{B}''(\mathcal{B}'(\mathbf{x}_1^{(w_{in})}))\,.
\end{aligned}$$

Since the input vectors $\mathbf{x}_1^{(w_{in})}$ and $\mathbf{x}_2^{(w_{in})}$ are a fooling pair to $\mathcal{B}$, it follows that the vectors $\mathcal{B}'(\mathbf{x}_1^{(w_{in})})$ and $\mathcal{B}'(\mathbf{x}_2^{(w_{in})})$ are a fooling pair to $\mathcal{B}''$. Thus, we apply induction hypothesis (2) for B'' by taking $\mathcal{B}'(\mathbf{x}_1^{(w_{in})})$ for $\mathbf{x}_1^{(w_{in})}$, $\mathcal{B}'(\mathbf{x}_2^{(w_{in})})$ for $\mathbf{x}_2^{(w_{in})}$, and $\mathcal{B}'(\mathbf{x}_2^{(w_{in})}+\mathbf{x}^{(w_{in})}) - \mathcal{B}'(\mathbf{x}_2^{(w_{in})})$ for $\mathbf{x}^{(w_{in})}$, to obtain that

$$\begin{aligned}
& \mathcal{B}''(\mathcal{B}'(\mathbf{x}_1^{(w_{in})}) + \mathcal{B}'(\mathbf{x}_2^{(w_{in})} + \mathbf{x}^{(w_{in})}) - \mathcal{B}'(\mathbf{x}_2^{(w_{in})})) - \mathcal{B}''(\mathcal{B}'(\mathbf{x}_1^{(w_{in})})) \\
= \;& \mathcal{B}''(\mathcal{B}'(\mathbf{x}_2^{(w_{in})}) + \mathcal{B}'(\mathbf{x}_2^{(w_{in})} + \mathbf{x}^{(w_{in})}) - \mathcal{B}'(\mathbf{x}_2^{(w_{in})})) - \mathcal{B}''(\mathcal{B}'(\mathbf{x}_2^{(w_{in})})) \\
= \;& \mathcal{B}''(\mathcal{B}'(\mathbf{x}_2^{(w_{in})} + \mathbf{x}^{(w_{in})})) - \mathcal{B}''(\mathcal{B}'(\mathbf{x}_2^{(w_{in})})) \\
= \;& \mathcal{B}(\mathbf{x}_2^{(w_{in})} + \mathbf{x}^{(w_{in})}) - \mathcal{B}(\mathbf{x}_2^{(w_{in})})\,.
\end{aligned}$$

It follows that $\mathcal{B}(\mathbf{x}_1^{(w_{in})}+\mathbf{x}^{(w_{in})}) - \mathcal{B}(\mathbf{x}_1^{(w_{in})}) = \mathcal{B}(\mathbf{x}_2^{(w_{in})}+\mathbf{x}^{(w_{in})}) - \mathcal{B}(\mathbf{x}_2^{(w_{in})})$, which completes the proof of (2). ■

We continue to establish further interesting properties of fooling pairs. Say that $\mathbf{x}^{(w_{in})}$ is a *null vector* to network $\mathcal{B} : \mathbf{x}^{(w_{in})} \rightarrow \mathbf{y}^{(w_{out})}$ if the vectors $\mathbf{x}^{(w_{in})}$ and $\mathbf{0}^{(w_{in})}$ are a fooling pair to $\mathcal{B}$. Intuitively, a null vector "hides" itself in the sense that it does not alter the state of $\mathcal{B}$ by traversing it. We state an immediate property of null vectors.

Proposition 3.3 *Consider a balancing network $\mathcal{B} : \mathbf{x}^{(w_{in})} \rightarrow \mathbf{y}^{(w_{out})}$. Take any input vectors $\mathbf{x}_1^{(w_{in})}$ and $\mathbf{x}_2^{(w_{in})}$ that are a fooling pair to network $\mathcal{B}$. Assume that $\mathbf{x}_2^{(w_{in})}$ is a null vector to network $\mathcal{B}$. Then, $\mathbf{x}_1^{(w_{in})}$ is also a null vector to network $\mathcal{B}$.*

We continue to show:

Proposition 3.4 *Consider a balancing network $\mathcal{B} : \mathbf{x}^{(w_{in})} \rightarrow \mathbf{y}^{(w_{out})}$. Take any input vector $\mathbf{x}^{(w_{in})}$ that is null to $\mathcal{B}$. Then, for any integer $k \geq 0$,*

(1) *$\mathcal{B}(k\mathbf{x}^{(w_{in})}) = k\mathcal{B}(\mathbf{x}^{(w_{in})})$;*

(2) *$k\mathbf{x}^{(w_{in})}$ is a null vector to $\mathcal{B}$.*

Proof: We start by showing (1). We proceed by induction on k. For the basis case where $k = 0$, the claim holds trivially.

Assume inductively that the claim holds for all integers k' such that $k' \leq k - 1$. For the induction step, consider the integer k. Clearly,

$$k\mathbf{x}^{(w_{in})} = (k-1)\mathbf{x}^{(w_{in})} + \mathbf{x}^{(w_{in})},$$

so that

$$\mathcal{B}(k\mathbf{x}^{(w_{in})}) = \mathcal{B}((k-1)\mathbf{x}^{(w_{in})} + \mathbf{x}^{(w_{in})}).$$

We apply Proposition 3.2(2) by taking $\mathbf{x}^{(w_{in})}$ for $\mathbf{x}_1^{(w_{in})}$, $\mathbf{0}^{(w_{in})}$ for $\mathbf{x}_2^{(w_{in})}$, and $(k-1)\mathbf{x}^{(w_{in})}$ for $\mathbf{x}^{(w_{in})}$, to obtain that

$$\begin{aligned}
\mathcal{B}((k-1)\mathbf{x}^{(w_{in})} + \mathbf{x}^{(w_{in})}) &= \mathcal{B}((k-1)\mathbf{x}^{(w_{in})}) + \mathcal{B}(\mathbf{x}^{(w_{in})}) - \mathcal{B}(\mathbf{0}^{(w_{in})}) \\
&= (k-1)\mathcal{B}(\mathbf{x}^{(w_{in})}) + \mathcal{B}(\mathbf{x}^{(w_{in})}) - \mathcal{B}(\mathbf{0}^{(w_{in})}) \\
&\quad \text{(by induction hypothesis)} \\
&= k\mathcal{B}(\mathbf{x}^{(w_{in})}),
\end{aligned}$$

as needed.

We continue to show (2). We proceed by induction on k. For the basis case where $k = 0$, the claim holds trivially. Assume the claim holds for all integers k' such that $k' \leq k - 1$. For the induction step, consider the integer k. Clearly,

$$k\mathbf{x}^{(w_{in})} = (k-1)\mathbf{x}^{(w_{in})} + \mathbf{x}^{(w_{in})}.$$

By assumption, $\mathbf{x}^{(w_{in})}$ is a null vector to $\mathcal{B}$. We apply Proposition 3.2 by taking $\mathbf{x}^{(w_{in})}$ for $\mathbf{x}_1^{(w_{in})}$, $\mathbf{0}^{(w_{in})}$ for $\mathbf{x}_2^{(w_{in})}$, and $(k-1)\mathbf{x}^{(w_{in})}$ for $\mathbf{x}^{(w_{in})}$, to obtain that the input vectors

$$\mathbf{x}^{(w_{in})} + (k-1)\mathbf{x}^{(w_{in})} = k\mathbf{x}^{(w_{in})}$$

and

$$\mathbf{0}^{(w_{in})} + (k-1)\mathbf{x}^{(w_{in})} = (k-1)\mathbf{x}^{(w_{in})}$$

are a fooling pair to $\mathcal{B}$. By induction hypothesis, $(k-1)\mathbf{x}^{(w_{in})}$ is a null vector to $\mathcal{B}$. It follows, by Proposition 3.3, that $k\mathbf{x}^{(w_{in})}$ is a null vector to $\mathcal{B}$, as needed. ■

For any balancing network $\mathcal{B}$, denote $W_{out}(\mathcal{B})$, the product of the fan-outs of balancers of $\mathcal{B}$. The next claim establishes a sufficient condition for null vectors.

Proposition 3.5 *Consider a balancing network* $\mathcal{B} : \mathbf{x}^{(w_{in})} \rightarrow \mathbf{y}^{(w_{out})}$. *Assume that* $W_{out}(\mathcal{B})$ *divides* $\mathbf{x}^{(w_{in})}$. *Then,* $\mathbf{x}^{(w_{in})}$ *is a null vector to* $\mathcal{B}$.

Proof: By induction on the depth d of $\mathcal{B}$. For the basis case where $d = 1$, $\mathcal{B}$ is a single layer. Take any single balancer $b : \mathbf{x}^{(f_{in})} \rightarrow \mathbf{y}^{(f_{out})}$ of $\mathcal{B}$. By definition of $W_{out}(\mathcal{B})$, f_{out} divides $W_{out}(\mathcal{B})$. Since $W_{out}(\mathcal{B})$ divides $\mathbf{x}^{(w_{in})}$, it follows that f_{out} divides $\mathbf{x}^{(w_{in})}$. Hence, f_{out} divides the restriction $\mathbf{x}^{(f_{in})}$ of $\mathbf{x}^{(w_{in})}$, so that $\mathbf{x}^{(f_{in})}$ is a null vector to b. Since b was chosen arbitrarily, this implies that $\mathbf{x}^{(w_{in})}$ is a null vector to $\mathcal{B}$, as needed.

Assume inductively that the claim holds for all balancing networks of depth at most $d-1$. Write $\mathcal{B} = \mathcal{B}'\mathcal{B}''$, where each of $\mathcal{B}'$ and $\mathcal{B}''$ is a balancing network of depth at most $d-1$. Since $W_{out}(\mathcal{B})$ divides $\mathbf{x}^{(w_{in})}$ and $W_{out}(\mathcal{B}) = W_{out}(\mathcal{B}')W_{out}(\mathcal{B}'')$, it follows that $W_{out}(\mathcal{B}')$ divides $\mathbf{x}^{(w_{in})}$. It follows by induction hypothesis for network $\mathcal{B}'$, that $\mathbf{x}^{(w_{in})}$ is a null vector to network $\mathcal{B}'$.

It remains to show that $\mathcal{B}'(\mathbf{x}^{(w_{in})})$ is a null vector to network $\mathcal{B}''$. Since $W_{out}(\mathcal{B})$ divides $\mathbf{x}^{(w_{in})}$, Proposition 3.4(1) implies that $W_{out}(B)$ divides $\mathcal{B}'(\mathbf{x}^{(w_{in})})$. Since $W_{out}(\mathcal{B}) = W_{out}(\mathcal{B}')W_{out}(\mathcal{B}'')$, it follows that $W_{out}(B'')$ divides $\mathcal{B}'(\mathbf{x}^{(w_{in})})$. Thus, by induction hypothesis for network $\mathcal{B}''$, $\mathcal{B}'(\mathbf{x}^{(w_{in})})$ is a null vector to network $\mathcal{B}''$, which completes the proof. ■

4 Equivalence Result

In this section, we show our equivalence result. We start by proving:

Proposition 4.1 *Fix any boundedness property* $\mathbf{\Pi}$. *Consider any balancing network* $\mathcal{B} : \mathbf{x}^{(w_{in})} \rightarrow \mathbf{y}^{(w_{out})}$ *that has the boundedness property* $\mathbf{\Pi}$. *Assume that* $W_{out}(\mathcal{B})$ *divides* $\mathbf{x}^{(w_{in})}$. *Then,* $\mathbf{y}^{(w_{out})}$ *is a constant vector.*

Proof: Assume, by way of contradiction, that there exist entries y_j and y_k of $\mathbf{y}^{(w_{out})}$ such that $y_j \neq y_k$. Thus, clearly, $|y_j - y_k| \geq 1$.

Since $\mathcal{B}$ has the boundedness property, it follows that $|y_j - y_k| \leq K$ for some universal integer $K \geq 1$.

Since $W_{out}(\mathcal{B})$ divides $\mathbf{x}^{(w_{in})}$, it follows by Proposition 3.5 that $\mathbf{x}^{(w_{in})}$ is a null vector to network $\mathcal{B}$. Thus, by Proposition 3.4(1), $\mathcal{B}((K+1)\mathbf{x}^{(w_{in})}) = (K+1)\mathcal{B}(\mathbf{x}^{(w_{in})}) = (K+1)\mathbf{y}^{(w_{out})}$.

Consider the jth and kth entries y'_j and y'_k, respectively, of $\mathcal{B}((K+1)\mathbf{x}^{(w_{in})})$. Since $\mathcal{B}$ has the boundedness property, $|y'_j - y'_k| \leq K$. However,

$$\begin{aligned} y'_j - y'_k &= (K+1)y_j - (K+1)y_k \\ &= (K+1)(y_j - y_k)\,, \end{aligned}$$

so that

$$\begin{aligned} |y'_j - y'_k| &= (K+1)|y_j - y_k| \\ &\geq K+1\,, \end{aligned}$$

a contradiction. ■

We finally show our main result:

Theorem 4.2 *Fix any boundedness property* $\mathbf{\Pi}$. *Consider any balancing network* $\mathcal{B} : \mathbf{x}^{(w_{in})} \rightarrow \mathbf{y}^{(w_{out})}$ *such that* $\mathbf{y}^{(w_{out})}$ *has the boundedness property* $\mathbf{\Pi}$ *whenever* $\mathbf{x}^{(w_{in})}$ *is a non-negative vector. Then,* $\mathcal{B}$ *has the boundedness property* $\mathbf{\Pi}$.

Proof: Consider any arbitrary input vector $\mathbf{x}^{(w_{in})}$. We will show that $\mathcal{B}(\mathbf{x}^{(w_{in})})$ has the boundedness property $\mathbf{\Pi}$.

Construct from $\mathbf{x}^{(w_{in})}$ an input vector $\tilde{\mathbf{x}}^{(w_{in})}$ such that for each index i, $0 \leq i \leq w_{in} - 1$, $\tilde{x}_i$ is the least multiple of $W_{out}(\mathcal{B})$ such that $\tilde{x}_i + x_i \geq 0$. Clearly, $W_{out}(\mathcal{B})$ divides $\tilde{\mathbf{x}}^{(w_{in})}$. By Proposition 4.1, $\mathcal{B}(\tilde{\mathbf{x}}^{(w_{in})})$ is a constant vector, while by Proposition 3.5, $\tilde{\mathbf{x}}^{(w_{in})}$ is a null vector to network $\mathcal{B}$. We apply Proposition 3.2(2) with $\tilde{\mathbf{x}}^{(w_{in})}$ for $\mathbf{x}_1^{(w_{in})}$, $\mathbf{0}^{(w_{in})}$ for $\mathbf{x}_2^{(w_{in})}$, and $\mathbf{x}^{(w_{in})}$ for $\mathbf{x}^{(w_{in})}$; we obtain that

$$\begin{aligned} \mathcal{B}(\tilde{\mathbf{x}}^{(w_{in})} + \mathbf{x}^{(w_{in})}) &= \mathcal{B}(\mathbf{x}^{(w_{in})}) + \mathcal{B}(\tilde{\mathbf{x}}^{(w_{in})}) - \mathcal{B}(\mathbf{0}^{(w_{in})}) \\ &= \mathcal{B}(\mathbf{x}^{(w_{in})}) + \mathcal{B}(\tilde{\mathbf{x}}^{(w_{in})})\,, \end{aligned}$$

so that

$$\mathcal{B}(\mathbf{x}^{(w_{in})}) = \mathcal{B}(\tilde{\mathbf{x}}^{(w_{in})} + \mathbf{x}^{(w_{in})}) - \mathcal{B}(\tilde{\mathbf{x}}^{(w_{in})})\,.$$

Since $\tilde{\mathbf{x}}^{(w_{in})} + \mathbf{x}^{(w_{in})}$ is a non-negative input vector, it follows, by assumption on $\mathcal{B}$, that $\mathcal{B}(\tilde{\mathbf{x}}^{(w_{in})} + \mathbf{x}^{(w_{in})})$ has the boundedness property $\mathbf{\Pi}$. Since $\mathcal{B}(\tilde{\mathbf{x}}^{(w_{in})})$ is a constant vector, it follows, by closure of a boundedness property under addition with a constant vector, that $\mathcal{B}(\mathbf{x}^{(w_{in})})$ has the boundedness property $\mathbf{\Pi}$, as needed. $\blacksquare$

5 Conclusion

We have shown that any balancing network that satisfies any given boundedness property on all non-negative input vectors, it will also satisfy the same property for any arbitrary input vector. Interesting examples of such properties are the step property and the K-smoothing property. A significant consequence of our result is that all known (deterministic) constructions of counting and smoothing networks [1, 3, 4, 5, 8, 13, 15, 19, 20, 24] will correctly handle both tokens and antitokens, and therefore simultaneously support both the increment and decrement operations. Another significant consequence is that the sufficient timing conditions for linearizability in counting networks established in [21, 22] immediately carry over when introducing antitokens in addition to tokens. Our work leaves open several interesting questions and problems.

Aiello *et al.* [3] present an interesting construction of a *randomized* counting network; they use *randomized balancers,* which distribute tokens on output wires according to some random permutation. Does this network operate correctly when simultaneously traversed by both tokens and antitokens? It seems to appear that the randomized balancers need to somehow "remember" the entire history of the random permutations in order for antitokens (resp., tokens) to trace back the paths of tokens (resp., antitokens).

Other interesting properties of balancing networks include the *threshold property* [4, 7] and the *weak threshold property* [7], outlined below. Let $\mathbf{x}^{(w_{in})}$ and $\mathbf{y}^{(w_{out})}$ be the input vector and the corresponding output vector of a balancing network that has any of these properties. For the threshold property, we require that $y_0 = \lceil \|\mathbf{x}^{(w_{in})}\|_1 / w_{out} \rceil$, while for the weak threshold property, we require that there is *some* output index j, possibly $j \neq 0$, such that $y_j = \lceil \|\mathbf{x}^{(w_{in})}\|_1 / w_{out} \rceil$. Since we have not established that either of these properties is a boundedness property, our result does not apply a fortiori to these properties. It would be extremely interesting to see whether these properties are preserved by the introduction of antitokens. We conjecture that different techniques are required for handling these questions.

References

[1] E. Aharonson and H. Attiya. Counting networks with arbitrary fan-out. *Distributed Computing*, 8(4):163–169, 1995.

[2] W. Aiello, M. Herlihy, N. Shavit, and D. Touitou. Inc/dec counting networks. Manuscript, Dec. 1995.

[3] W. Aiello, R. Venkatesan, and M. Yung. Coins, weights and contention in balancing networks. In *Proceedings of the 13th Annual ACM Symposium on Principles of Distributed Computing (PODC'94)*, pages 193–205, Los Angeles, Aug. 1994.

[4] J. Aspnes, M. Herlihy, and N. Shavit. Counting networks. *Journal of the ACM*, 41(5):1020–1048, Sept. 1994.

[5] C. Busch, N. Hardavellas, and M. Mavronicolas. Contention in counting networks (abstract). In *Proceedings of the 13th annual ACM Symposium on Principles of Distributed Computing (PODC'94)*, page 404, Los Angeles, Aug. 1994.

[6] C. Busch and M. Mavronicolas. The strength of counting networks (abstract). In *Proceedings of the 15th Annual ACM Symposium on Principles of Distributed Computing (PODC'96)*, page 311, Philadelphia, May 1996.

[7] C. Busch and M. Mavronicolas. Impossibility results for weak threshold networks. *Information Processing Letters*, 63(2):85–90, July 1997.

[8] C. Busch and M. Mavronicolas. An efficient counting network. In *Proceedings of the 1st Merged International Parallel Processing Symposium and Symposium on Parallel and Distributed Processing (IPPS/SPDP'98)*, pages 380–385, Mar. 1998.

[9] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to algorithms.* MIT Press and McGraw-Hill Book Company, Cambridge, MA, 1992.

[10] P. J. Courtois, F. Heymans, and D. L. Parnas. Concurrent control with "readers" and "writers". *Communications of the ACM*, 14(10):667–668, Oct. 1971.

[11] E. W. Dijkstra. Solution of a problem in concurrent programming control. *Communications of the ACM*, 8(9):569, Sept. 1965.

[12] E. W. Dijkstra. Cooperating sequential processes. In *Programming Languages*, pages 43–112. Academic Press, 1968.

[13] E. W. Felten, A. LaMarca, and R. Ladner. Building counting networks from larger balancers. Technical Report TR 93-04-09, University of Washington, Apr. 1993.

[14] P. B. Hansen. *Operating System Principles.* Prentice-Hall, Englewood Cliffs, NJ, 1973.

[15] N. Hardavellas, D. Karakos, and M. Mavronicolas. Notes on sorting and counting networks. In *Proceedings of the 7th International Workshop on Distributed Algorithms (WDAG'93)*, volume 725 of *Lecture Notes in Computer Science*, pages 234–248, Lausanne, Switzerland, Sept. 1993. Springer-Verlag.

[16] M. Herlihy, B.-H. Lim, and N. Shavit. Scalable concurrent counting. *ACM Transactions on Computer Systems*, 13(4):343–364, Nov. 1995.

[17] C. A. R. Hoare and R. N. Periott. *Operating Systems Techniques.* Academic Press (New York NY), London, 1972.

[18] S. Kapidakis and M. Mavronicolas. Distributed, low contention task allocation. In *Proceedings of the 8th IEEE Symposium on Parallel and Distributed Processing (SPDP'96)*, pages 358–365, Washington, Oct. 1996.

[19] M. Klugerman. *Small-Depth Counting Networks and Related Topics.* PhD thesis, Department of Mathematics, Massachusetts Institute of Technology, Sept. 1994.

[20] M. Klugerman and C. G. Plaxton. Small-depth counting networks. In *Proceedings of the 24th Annual ACM Symposium on the Theory of Computing (STOC'92)*, pages 417–428, Victoria, B.C., Canada, May 1992.

[21] N. Lynch, N. Shavit, A. Shvartsman, and D. Touitou. Counting networks are practically linearizable. In *Proceedings of the 15th Annual ACM Symposium on Principles of Distributed Computing (PODC'96)*, pages 280–289, New York, May 1996.

[22] M. Mavronicolas, M. Papatriantafilou, and P. Tsigas. The impact of timing on linearizability in counting networks. In *Proceedings of the 11th International Parallel Processing Symposium (IPPS'97)*, pages 684–688, Los Alamitos, Apr. 1997.

[23] N. Shavit and D. Touitou. Elimination trees and the construction of pools and stacks. *Theory of Computing Systems*, 30(6):545–570, Nov./Dec. 1997.

[24] N. Shavit and A. Zemach. Diffracting trees. *ACM Transactions on Computer Systems*, 14(4):385–428, Nov. 1996.

[25] A. Silberschatz and P. B. Galvin. *Operating System Concepts.* Addison Wesley, 4th edition, 1994.