A Combinatorial Characterization of Properties Preserved by Antitokens[*]

Costas Busch[†] *Neophytos Demetriou*[‡] *Maurice Herlihy*[§]
Marios Mavronicolas[¶]

May 15, 2000

[*]An extended abstract of this work will appear in the *Proceedings of the European Conference on Parallel Processing (EuroPar 2000),* Munich, Germany, August/September 2000.

[†]Department of Computer Science, Brown University, Providence, RI 02912. Email: `cb@cs.brown.edu`

[‡]Department of Computer Science, University of Cyprus, Nicosia CY-1678, Cyprus. Email: `cs97dn1@ucy.ac.cy`

[§]Department of Computer Science, Brown University, Providence, RI 02912. Email: `mph@cs.brown.edu`

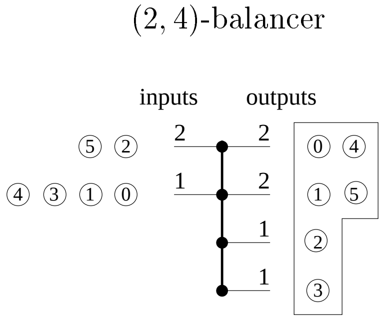
[¶]**Correspondent author.** Department of Computer Science, University of Cyprus, Nicosia CY-1678, Cyprus. Part of the work of this author was performed while at Department of Computer Science and Engineering, University of Connecticut. Email: `mavronic@ucy.ac.cy`

Abstract

Balancing networks are highly distributed data structures that are used for providing efficient solutions to multiprocessor synchronization problems. Traditionally, balancing networks have been designed to be accessed by *tokens,* which correspond to *increment* operations. The distribution of tokens on the network's output specifies the correctness property of the network. However, tokens alone may be inadequate for synchronization problems that require *decrement* operations, such as semaphores and critical regions. For such problems, *antitokens* have been introduced to implement the decrement operation [21].

It has been shown that several kinds of networks that satisfy the *step property,* the *smoothing property* and the *threshold property* for tokens alone preserve their properties when antitokens are introduced [2, 5, 21]. Thus, such networks are able to solve synchronization problems that require decrements. A fundamental question that has been left open is to formally characterize all properties of balancing networks that are preserved under the introduction of antitokens.

In this work, we provide a simple, combinatorial characterization for all properties warranted by balancing networks which are preserved when antitokens are introduced. This characterization serves as a theoretical tool for identifying the properties that are preserved by antitokens.



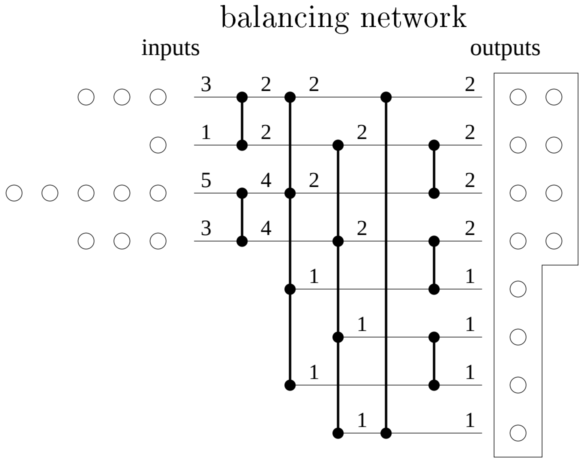


Figure 1: A balancer and a balancing network

1 Introduction

1.1 Motivation

Balancing networks were devised by Aspnes *et al.* [4] as a novel class of distributed data structures that provide highly-concurrent, low-contention solutions to a variety of synchronization problems. Balancing networks attract a lot of interest and attention due to their nice performance and scalability properties (see, e.g., [1, 2, 3, 5, 6, 7, 8, 14, 15, 16, 18, 19, 20, 21, 22, 23]).

A balancing network is constructed from elementary switches with p *input wires* and q *output wires,* called (p, q)-balancers. As illustrated in Figure, a (p, q)-balancer accepts a stream of tokens on its p input wires. The i-th token to enter the balancer leaves on output wire $i \bmod q$ (where $i = 0, 1, \ldots$). In Figure 1, we write on each wire the overall number of tokens that appear on the wire.

One can think of a balancer as having a "toggle" state variable tracking which output wire the next token should exit from. A token traversal amounts to a *Fetch&Increment* operation to the toggle variable. The operation includes reading first the current state of the toggle, which is the wire the token should exit from, and then setting the toggle to point to the next output wire. The distribution of the tokens on the output wires satisfies the *step property* (described below).

A balancing network is an acyclic network of balancers, where output wires of some balancers are linked to input wires of other balancers (sse Figure 1). The network's *input wires* are those input wires of balancers that are not linked from any other balancer, and similarly for the network's *output wires.* Tokens enter the network on the input wires, typically several per wire, propagate asynchronously through the balancers, and leave on the output wires, typically several per wire.

Balancing networks are classified according to the distribution of the exiting tokens on the output wires. *Counting networks* [4] are those balancing networks for which the exiting tokens are divided uniformly among the output wires and any excess tokens appear on the upper wires. Note that the balancing network in Figure 1 is a counting network. We say that the exiting tokens of a counting network satisfy the *step property* (see Figure 1). For smoothing networks [1, 4], the output tokens satisfy the *K-smoothing property,* for any integer $K \geq 1$, where the sum of tokens on any two output wires may differ by no more than K. In the case of *threshold networks* [4, 8], the output tokens satisfy the *threshold property,* whereby the number of tokens on the bottom wire increases by one per bunch of w tokens, where w is the number of output wires of the network (also called *fan-out* [1]). In the *weak threshold property,* the output wire can be redefined to be any output wire.

Based on balancing networks, simple and elegant algorithms have been developed to solve a variety of synchronization problems that appear in distributed systems. For example, counting networks are used to implement efficient *Fetch&Increment* counters [4] as well as *linearizable* counters [15]. Furthermore, smoothing networks solve *load sharing* problems [25], while threshold networks provide solutions to *barrier synchronization* problems [11, 12]. For applications of balancing networks, see [4, 15, 16, 18, 21, 23].

A limitation of balancing networks is that they can be accessed by tokens only. Recall that a token can be thought of as an increment operation issued by the process inserting the token into the network. Using tokens only, the capabilities of balancing networks are limited to using only increment operations. However, many distributed algorithms require the ability to decrement shared objects as well. For example, the classical synchronization constructs of *semaphores* [9], *critical regions* [17], and *monitors* [13] all rely on applying both increment and decrement operations on shared counters (see, e.g., [24, Chapter 6]).

In order to solve such kinds of problems, Shavit and Touitou [21] invented the *antitoken,* an entity that a processor shepherds through the network in order to perform a decrement operation. Unlike a token, which traverses a balancer by fetching the toggle value and then advancing it, an antitoken first 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. Furthermore, when a token and an antitoken "meet" while they are each individually traversing

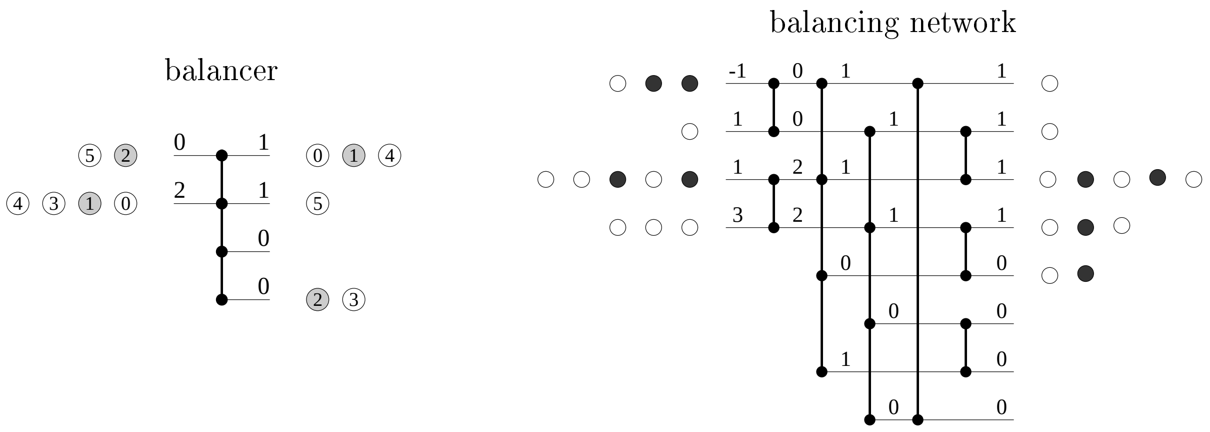


Figure 2: A balancer and a balancing network accessed simultaneously by tokens and antitokens

the network, they can eliminate each other, thus avoiding the need to traverse the rest of the network. Antitokens are depicted in Figure 2, where an antitoken is drawn with a filled circle, meant to correspond to the algebraic quantity -1, whereas a token is drawn with an empty circle, meant to correspond to the algebraic quantity +1.

In the same paper, Shavit and Touitou provide an operational proof that a specific kind of counting networks which have the form of binary trees count correctly even when they are traversed by both tokens and antitokens. Namely, they showed that, for such networks, the step property is preserved by the introduction of antitokens.

Subsequently, Aiello *et al.* [2] generalized the results of Shavit and Touitou [21] to far more general classes of balancing networks and properties of balancing networks. More specifically, Aiello *et al.* considered *boundedness properties,* a generalization of the step and K-smoothing properties. They showed that boundedness properties are preserved by the introduction of antitokens. Busch *et al.* [5] considered the *threshold property* [4], and they showed that this property is also preserved by the introduction of antitokens. Furthermore, they showed that for *regular* balancing networks, where the number of input and output wires is the same for each balancer, the *weak threshold property* [7] is preserved too.

1.2 Contribution

A fundamental question that was left open by the results in [2, 5, 21] is to formally characterize all properties of balancing networks that are preserved under the introduction of antitokens. Such characterization would be used for identifying properties that are preserved by antitokens.

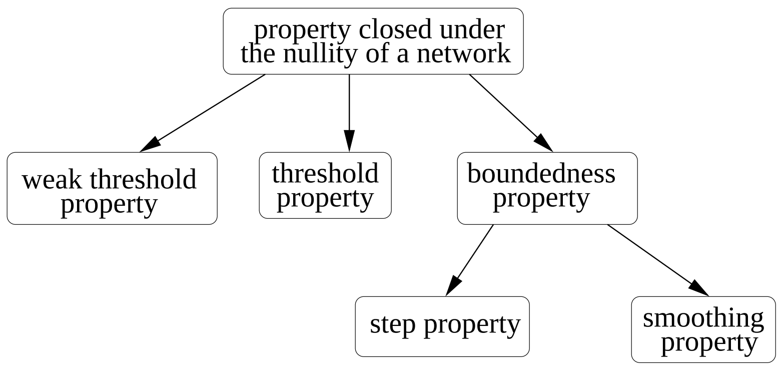


Figure 3: The characterization of properties preserved by antitokens

Moreover, the characterization would enable practitioners to rule out balancing networks satisfying properties that are not preserved by antitokens from being used in specific application algorithms that involve decrement operations. In this work, we provide the first answer to this fundamental question.

We provide a simple, combinatorial characterization of properties warranted by balancing networks which are preserved when decrement operations via antitokens are introduced.

For any arbitrary balancing network, we define a new, natural class of properties, that we call *closed under the nullity of the balancing network,* which precisely characterize all the properties that are preserved by antitokens. This characterization provides necessary and sufficient conditions for all the properties that are preserved. Specifically, for any property that is satisfied by a balancing network for tokens only, the property is still satisfied by the network when antitokens are introduced if and only if the property is closed under the nullity of the network.

This characterization provides a theoretical tool for identifying which properties are preserved by the introduction of antitokens. Consider any propertyof a balancing network which we know a balancing network satisfies for tokens only. In order to prove that that this property will be preserved by the introduction of antitokens, we only need to show that this property is closed under the nullity of the network. Equipped with this theoretical tool, the practitioner can determine if a specific property of balancing networks can be used to implement algorithms that require decrements.

Furthermore, the necessity of the characterization enables us to classify all the properties which are already known to be preserved by antitokens. Necessity implies that all these properties must satisfy the characterization; therefore, they are closed under the nullity of a balancing networks providing the property. Consequently, the counting property, the K-smoothing property, and, in general, any boundedness property are each closed under the nullity of a balancing network providing the property (see Figure 3). Furthermore, the threshold and the weak threshold properties (for regular networks) are each closed under the nullity of a balancing network that satisfies it.

Our proof techniques are purely combinatorial. We manage to abstract out the main ideas from the proofs in [2, 5] that were dealing with *specific* properties; we generalize these ideas to an abstract, property-independent setting, and this yields the characterization.

1.3 Road Map

The rest of this paper is organized as follows. Section 2 provides some background for our discussion and introduces some formal definitions. Our main combinatorial characterization result is presented in Section 3. We conclude, in Section 4, with a discussion of our results and some open problems.

2 Framework

Our formal framework is patterned after [2, Sections 2 & 3].

2.1 Vectors

We consider integer vectors. For any integer $g \geq 2$, $\mathbf{x}^{(g)}$ denotes the vector $\langle x_0, x_1, \ldots, x_{g-1} \rangle^{\mathrm{T}}$. For any vector $\mathbf{x}^{(g)}$, denote $\sum \mathbf{x}^{(g)} = \sum_{i=0}^{g-1} x_i$. We use $\mathbf{0}^{(g)}$ to denote $\langle 0, 0, \ldots, 0 \rangle^{\mathrm{T}}$, a vector with g zero entries. In a *constant vector,* all entries are equal to some constant c. In a *non-negative vector,* all entries are non-negative integers. Say that an integer d *divides* a vector $\mathbf{x}^{(g)}$ if each entry of $\mathbf{x}^{(g)}$ is some integer multiple of d.

2.2 Balancers

This section is adapted from [2, Section 2.2].

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, we define balancers as "multibalancers," in the style of Aharonson and Attiya [1], Felten *et al.*

[10], and Hardavellas *et al.* [14]; however, we follow Shavit and Touitou [21] Aiello *et al.* [2], and Busch *et al.* [5] 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 balancer.

So, for any pair of positive integers f_{in} and f_{out}, an $(f_{\text{in}}, f_{\text{out}})$*-balancer,* or *balancer* for short, is a routing element receiving tokens and antitokens on f_{in} input wires, numbered $0, 1, \ldots, f_{\text{in}}-1$, and sending out tokens and antitokens to f_{out} output wires, numbered $0, 1, \ldots, f_{\text{out}} - 1$; f_{in} and f_{out} are called the balancer's *fan-in* and *fan-out,* respectively. A *regular* balancer is an $(f_{\text{in}}, f_{\text{out}})$-balancer such that $f_{\text{in}} = f_{\text{out}}$; that is, fan-in equals fan-out for a regular balancer.

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.

For each input index i, $0 \leq i \leq f_{\text{in}} - 1$, 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 $\mathbf{x}^{(f_{\text{in}})} = \langle x_0, x_1, \ldots, x_{f_{\text{in}}-1}\rangle^{\text{T}}$; call $\mathbf{x}^{(f_{\text{in}})}$ an *input vector.* For each output index j, $0 \leq j \leq f_{\text{out}} - 1$, 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. Denote $\mathbf{y}^{(f_{\text{out}})} = \langle y_0, y_1, \ldots, y_{f_{\text{out}}-1}\rangle^{\text{T}}$; call $\mathbf{y}^{(f_{\text{out}})}$ an *output vector.*

The *configuration* of a balancer at any given time is the tuple $\langle \mathbf{x}^{(f_{\text{in}})}, \mathbf{y}^{(f_{\text{out}})}\rangle$; 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, $\mathbf{x}^{(f_{\text{in}})} = \mathbf{0}^{(f_{\text{in}})}$, and $\mathbf{y}^{(f_{\text{out}})} = \mathbf{0}^{(f_{\text{out}})}$.

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 $(f_{\text{in}}, f_{\text{out}})$-balancer.

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

2. *Liveness property:* for any finite numbers t of tokens and a of antitokens that enter the balancer, the balancer reaches within a finite amount of time a quiescent configuration where $t-e$ tokens and $a-e$ antitokens have exited the network, where e, $0 \leq e \leq \min\{t, a\}$, is the number of tokens and antitokens that are "eliminated" in the balancer.

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

From the safety and liveness properties, it follows that for any quiescent configuration $\langle \mathbf{x}^{(f_{\text{in}})}, \mathbf{y}^{(f_{\text{out}})} \rangle$ of a balancer, $\|\mathbf{x}^{(f_{\text{in}})}\|_1 = \|\mathbf{y}^{(f_{\text{out}})}\|_1$; 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. Note that the equality holds even though some of the tokens and antitokens may be "eliminated" in the balancer.

We are mostly interested in quiescent configurations of a balancer. For any input vector $\mathbf{x}^{(f_{\text{in}})}$ to balancer b, denote $\mathbf{y}^{(f_{\text{out}})} = b(\mathbf{x}^{(f_{\text{in}})})$ the output vector in the quiescent configuration that b will reach after all tokens and antitokens that entered b have exited (or eliminated themselves); write also $b : \mathbf{x}^{(f_{\text{in}})} \rightarrow \mathbf{y}^{(f_{\text{out}})}$ to denote the balancer b.

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

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

by definition of quiescent configuration, it follows that

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

Thus, for the sake of simplicity, we will denote

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

We remark that the state of an $(f_{\text{in}}, f_{\text{out}})$-balancer is some integer in the set $\{0, 1, \ldots, f_{\text{out}} - 1\}$, 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.

Figure 2 depicts a $(2, 4)$-balancer with two input wires and four output wires, stretched horizontally; the balancer is stretched vertically. In the left part, tokens and antitokens are denoted with empty and full circles, respectively; the numbering reflects the real-time order of tokens and antitokens in an execution where they traverse the balancer one by one (such an execution is called a *sequential* execution). On each wire we write the total algebraic sum of tokens and antitokens that appear on the wire.

2.3 Balancing Networks

This section is adopted from [2, Section 2.3].

A $(w_{\text{in}}, w_{\text{out}})$-*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 $0, 1, \ldots, w_{\text{in}} - 1$, which are not connected to output wires of balancers, having w_{out} designated *output wires,* numbered $0, 1, \ldots, w_{\text{out}} - 1$, similarly not connected to input wires of balancers, and containing no cycles. A balancing network is *regular* if each of its interwired balancers is regular.

Tokens and antitokens arrive on the network's input wires at arbitrary times, and they traverse a sequence of balancers in the network in a completely asynchronous way till they exit (or eliminate themselves) on the output wires of the network.

For each input index i, $0 \leq i \leq w_{\text{in}} - 1$, 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 $\mathbf{x}^{(w_{\text{in}})} = \langle x_0, x_1, \ldots, x_{w_{\text{in}}-1}\rangle^{\mathrm{T}}$; call $\mathbf{x}^{(w_{\text{in}})}$ an *input vector.* For each output index j, $0 \leq j \leq f_{\text{out}} - 1$, 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 j. Denote $\mathbf{y}^{(w_{\text{out}})} = \langle y_0, y_1, \ldots, y_{w_{\text{in}}-1}\rangle^{\mathrm{T}}$; call $\mathbf{y}^{(w_{\text{out}})}$ 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 either exited the network or pairwise "eliminated" themselves. In any quiescent configuration of $\mathcal{B}$ we have $\|\mathbf{x}^{(w_{\text{in}})}\|_1 = \|\mathbf{y}^{(w_{\text{out}})}\|_1$; 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 $\mathbf{x}^{(w_{\text{in}})}$ and $\mathbf{y}^{(w_{\text{out}})}$, respectively, the *state* of $\mathcal{B}$, denoted $\text{state}_{\mathcal{B}}(\mathbf{x}^{(\mathrm{w}_{\text{in}})})$, is defined to be the collection of the states of its individual balancers. We remark that we have specified $\mathbf{x}^{(w_{\text{in}})}$ as the single argument of $\text{state}_{\mathcal{B}}$, since $\mathbf{x}^{(w_{\text{in}})}$ 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 $\mathbf{x}^{(w_{\text{in}})}$, denote $\mathbf{y}^{(w_{\text{out}})} = \mathcal{B}(\mathbf{x}^{(w_{\text{in}})})$ the output vector in the quiescent configuration that $\mathcal{B}$ will reach after all tokens and antitokens that entered $\mathcal{B}$ have exited (or eliminated themselves); write also $\mathcal{B} : \mathbf{x}^{(w_{\text{in}})} \rightarrow \mathbf{y}^{(w_{\text{out}})}$ to denote the network $\mathcal{B}$. Clearly, $\mathcal{B}(\mathbf{0}^{(w_{\text{in}})}) = \mathbf{0}^{(w_{\text{out}})}$.

Figure 2 depicts a balancing network with four input wires and eight output wires using the same conventions as for the $(2,4)$-balancer in the left part of the same figure (see also Section 2.2).

2.4 Fooling Pairs

Our presentation follows [2, Section 4].

Say that input vectors $\mathbf{x}_1^{(f_{\text{in}})}$ and $\mathbf{x}_2^{(f_{\text{in}})}$ are *a fooling pair to balancer* $b : \mathbf{x}^{(f_{\text{in}})} \rightarrow \mathbf{y}^{(f_{\text{out}})}$ [2, Section 4] if $\text{state}_{\mathrm{b}}(\mathbf{x}_1^{(\mathrm{f}_{\text{in}})}) = \text{state}_{\mathrm{b}}(\mathbf{x}_2^{(\mathrm{f}_{\text{in}})})$; roughly speaking, a fooling pair "drives" the balancer to identical states in the two corresponding quiescent configurations. The concept of a fooling pair can be extended from a single balancer to a network in the natural way. Say that input vectors $\mathbf{x}_1^{(w_{\text{in}})}$ and $\mathbf{x}_2^{(w_{\text{in}})}$ are *a fooling pair to network* $\mathcal{B} : \mathbf{x}^{(w_{\text{in}})} \rightarrow \mathbf{y}^{(w_{\text{out}})}$ if for each balancer b of $\mathcal{B}$, the input vectors of b in quiescent configurations corresponding to $\mathbf{x}_1^{(w_{\text{in}})}$ and $\mathbf{x}_2^{(w_{\text{in}})}$, 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.

The next result relates the output vectors of any balancing network on certain combinations of a fooling pair of input vectors.

Lemma 2.1 (Aiello et al. [2]) *Consider a balancing network* $\mathcal{B} : \mathbf{x}^{(w_{\text{in}})} \rightarrow \mathbf{y}^{(w_{\text{out}})}$. *Take any input vectors* $\mathbf{x}_1^{(w_{\text{in}})}$ *and* $\mathbf{x}_2^{(w_{\text{in}})}$ *that are a fooling pair to network* $\mathcal{B}$. *Then, for any input vector* $\mathbf{x}^{(w_{\text{in}})}$,

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

We continue to survey some further combinatorial properties of fooling pairs that we will use in our later proofs. Say that $\mathbf{x}^{(w_{\text{in}})}$ is a *null vector to network* $\mathcal{B} : \mathbf{x}^{(w_{\text{in}})} \rightarrow \mathbf{y}^{(w_{\text{out}})}$ [2, Section 3] if the vectors $\mathbf{x}^{(w_{\text{in}})}$ and $\mathbf{0}^{(w_{\text{in}})}$ are a fooling pair to $\mathcal{B}$. Intuitively, a null vector "hides" itself from the network $\mathcal{B}$ in the sense that it does not alter the state of $\mathcal{B}$ while traversing it. The next claim determines the output of a balancing network on any non-negative multiple of a null vector.

Lemma 2.2 (Aiello et al. [2]) *Consider a balancing network* $\mathcal{B} : \mathbf{x}^{(w_{\text{in}})} \rightarrow \mathbf{y}^{(w_{\text{out}})}$. *Take any vector* $\mathbf{x}^{(w_{\text{in}})}$ *that is null to* $\mathcal{B}$. *Then, for any integer* $k \geq 0$,

$$\mathcal{B}(k\mathbf{x}^{(w_{\text{in}})}) \;\; = \;\; k\,\mathcal{B}(\mathbf{x}^{(w_{\text{in}})})\,.$$

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

Lemma 2.3 (Aiello et al. [2]) *Consider a balancing network* $\mathcal{B} : \mathbf{x}^{(w_{\text{in}})} \rightarrow \mathbf{y}^{(w_{\text{out}})}$. *Assume that* $W_{\text{out}}(\mathcal{B})$ *divides* $\mathbf{x}^{(w_{\text{in}})}$. *Then,* $\mathbf{x}^{(w_{\text{in}})}$ *is a null vector to* $\mathcal{B}$.

We continue to show some further combinatorial properties of null vectors, which are new.

Lemma 2.4 *Consider any balancing network* $\mathcal{B} : \mathbf{x}^{(w_{\text{in}})} \rightarrow \mathbf{y}^{(w_{\text{out}})}$. *Assume,* $\mathbf{x}^{(w_{\text{in}})}$ *is a null vector of* $\mathcal{B}$. *Then,* $\mathcal{B}(-\mathbf{x}^{(w_{\text{in}})}) = -\mathcal{B}(\mathbf{x}^{(w_{\text{in}})})$.

Proof: By assumption $\mathbf{x}^{(w_{\text{in}})}$ is a null vector of $\mathcal{B}$. We apply Lemma 2.1(2) with $\mathbf{0}^{(w_{\text{in}})}$ for $\mathbf{x}_1^{(w_{\text{in}})}$, $\mathbf{x}^{(w_{\text{in}})}$ for $\mathbf{x}_2^{(w_{\text{in}})}$, and $-\mathbf{x}^{(w_{\text{in}})}$ for $\mathbf{x}^{(w_{\text{in}})}$; we obtain that

$$\mathcal{B}(\mathbf{0}^{(w_{\text{in}})} - \mathbf{x}^{(w_{\text{in}})}) - \mathcal{B}(\mathbf{0}^{(w_{\text{in}})}) \;\; = \;\; \mathcal{B}(\mathbf{x}^{(w_{\text{in}})} - \mathbf{x}^{(w_{\text{in}})}) - \mathcal{B}(\mathbf{x}^{(w_{\text{in}})})\,,$$

so that

$$\mathcal{B}(-\mathbf{x}^{(w_{\text{in}})}) \;\; = \;\; -\mathcal{B}(\mathbf{x}^{(w_{\text{in}})})\,,$$

as needed. ■

We continue to show:

Lemma 2.5 *Consider any balancing network $\mathcal{B} : \mathbf{x}^{(w_{\text{in}})} \rightarrow \mathbf{y}^{(w_{\text{out}})}$. Assume, $\mathbf{x}^{(w_{\text{in}})}$ is a null vector of $\mathcal{B}$. Then, $-\mathbf{x}^{(w_{\text{in}})}$ is a null vector of $\mathcal{B}$.*

Proof: We apply Lemma 2.1.(1) with $\mathbf{0}^{(w_{\text{in}})}$ for $\mathbf{x}_1^{(w_{\text{in}})}$, $\mathbf{x}^{(w_{\text{in}})}$ for $\mathbf{x}_2^{(w_{\text{in}})}$, and $-\mathbf{x}^{(w_{\text{in}})}$ for $\mathbf{x}^{(w_{\text{in}})}$; we obtain that $\mathbf{0}^{(w_{\text{in}})} - \mathbf{x}^{(w_{\text{in}})}$ and $\mathbf{x}^{(w_{\text{in}})} - \mathbf{x}^{(w_{\text{in}})}$ are a fooling pair to network $\mathcal{B}$. Thus, $-\mathbf{x}^{(w_{\text{in}})}$ is a null vector of $\mathcal{B}$, as needed. ■

2.5 Properties

A property $\mathbf{\Pi}$ is a (computable) predicate on integer vectors. We identify $\mathbf{\Pi}$ with the set of (integer) vectors satisfying it.

Say that *a vector $\mathbf{y}^{(w_{\text{out}})}$ has the property $\mathbf{\Pi}$* if $\mathbf{y}^{(w_{\text{out}})}$ satisfies $\mathbf{\Pi}$. Say that *a balancing network $\mathcal{B} : \mathbf{x}^{(w_{\text{in}})} \rightarrow \mathbf{y}^{(w_{\text{out}})}$ has a property $\mathbf{\Pi}$* if all its output vectors $\mathbf{y}^{(w_{\text{out}})}$ have the property $\mathbf{\Pi}$.

2.5.1 Boundedness Properties

Boundedness properties were introduced by Aiello *et al.* [2]. Our presentation summarizes [2, Section 2.4]. Fix throughout 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 $0 \leq j, k \leq g-1$, $|y_j - y_k| \leq K$; any vector in the K-smoothing property is a *K-smooth vector.* A *smoothing property* is a K-smoothing property, for some integer $K \geq 1$. A *boundedness property* [2, Section 2.4] is any subset of some K-smoothing property, for any integer $K \geq 1$, that is closed under addition with a constant vector. Thus, a boundedness property is a strict generalization of the smoothing property. Clearly, there are infinitely many boundedness properties.

The *step property* [4] 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 $0 \leq j < k \leq g-1$, $0 \leq y_j - y_k \leq 1$; any vector in the step property is a *step vector.* Clearly, the step property is a boundedness property, since any step vector is 1-smooth (but not vice versa). A *counting network* [4] is a balancing network that has the step property. Similarly, a *K-smoothing network* [1, 4] is a balancing network that has the K-smoothing property.

The main result of Aiello *et al.* [2] establishes that allowing negative inputs does not spoil the boundedness property of a balancing network.

Theorem 2.6 (Aiello et al. [2]) *Fix any boundedness property* $\mathbf{\Pi}$. *Consider any balancing network* $\mathcal{B} : \mathbf{x}^{(w_{\text{in}})} \to \mathbf{y}^{(w_{\text{out}})}$ *such that* $\mathbf{y}^{(w_{\text{out}})}$ *has the boundedness property* $\mathbf{\Pi}$ *whenever* $\mathbf{x}^{(w_{\text{in}})}$ *is a non-negative vector. Then,* $\mathcal{B}$ *has the boundedness property* $\mathbf{\Pi}$.

2.5.2 Threshold-Like Properties

Say that a vector $\mathbf{y}^{(w_{\text{out}})}$ is a *threshold vector* [4] if $y_{w_{\text{out}}-1} = \lfloor \|\mathbf{y}^{(w_{\text{out}})}\|_1 / w_{\text{out}} \rfloor$. The *threshold property* is the set of all threshold vectors $\mathbf{y}^{(w_{\text{out}})}$. Say that a vector $\mathbf{y}^{(w_{\text{out}})}$ is a *weak threshold vector* [7] if there is some output index j, possibly $j \neq w_{\text{out}}-1$, such that $y_j = \lfloor \|\mathbf{y}^{(w_{\text{out}})}\|_1 / w_{\text{out}} \rfloor$. The *weak threshold property* is the set of all weak threshold vectors $\mathbf{y}^{(w_{\text{out}})}$. A *threshold network* [4] is a balancing network $\mathcal{B} : \mathbf{x}^{(w_{\text{in}})} \to \mathbf{y}^{(w_{\text{out}})}$ that has the threshold property; similarly, a *weak threshold network* [7] is a balancing network $\mathcal{B} : \mathbf{x}^{(w_{\text{in}})} \to \mathbf{y}^{(w_{\text{out}})}$ that has the weak threshold property.

It has been observed in [5] that the (weak) threshold property is not a boundedness property in all non-trivial cases (where $w_{\text{out}} > 2$). Thus, Theorem 2.6 does not apply a fortiori to either threshold networks or weak threshold networks. The main result of Busch *et al.* [5] establishes that allowing negative inputs does not spoil the threshold property of a balancing network.

Theorem 2.7 (Busch et al. [5]) *Consider any balancing network* $\mathcal{B} : \mathbf{x}^{(w_{\text{in}})} \to \mathbf{y}^{(w_{\text{out}})}$ *such that* $\mathbf{y}^{(w_{\text{out}})}$ *has the threshold property whenever* $\mathbf{x}^{(w_{\text{in}})}$ *is a non-negative vector. Then,* $\mathcal{B}$ *has the threshold property.*

In addition, Busch *et al.* [5] establish that allowing negative inputs does not spoil the weak threshold property of a *regular* balancing network.

Theorem 2.8 (Busch et al. [5]) *Consider any regular balancing network* $\mathcal{B} : \mathbf{x}^{(w_{\text{in}})} \to \mathbf{y}^{(w_{\text{out}})}$ *such that* $\mathbf{y}^{(w_{\text{out}})}$ *has the weak threshold property whenever* $\mathbf{x}^{(w_{\text{in}})}$ *is a non-negative vector. Then,* $\mathcal{B}$ *has the weak threshold property.*

3 Combinatorial Characterization

In this section, we present a combinatorial characterization of properties of balancing networks that are preserved under the introduction of antitokens.

This section is organized as follows. Section 3.1 gives the closure under nullity definition. Our main result is shown in Section 3.3. We conclude, in Section 3.3, with some remarks.

3.1 Closure Under Nullity

We start with the definition of a new predicate on pairs of a property and a balancing network.

Definition 3.1 *Consider any balancing network $\mathcal{B} : \mathbf{x}^{(w_{\text{in}})} \rightarrow \mathbf{y}^{(w_{\text{out}})}$. A property $\mathbf{\Pi}$ is* closed under the nullity of $\mathcal{B}$ *if for all non-negative input vectors $\mathbf{x}^{(w_{\text{in}})}$ and for all non-negative null vectors $\tilde{\mathbf{x}}^{(w_{\text{in}})}$ of $\mathcal{B}$, it holds that $\mathcal{B}(\mathbf{x}^{(w_{\text{in}})}) \in \mathbf{\Pi}$ implies $\mathcal{B}(\mathbf{x}^{(w_{\text{in}})}) + B(\tilde{\mathbf{x}}^{(w_{\text{in}})}) \in \mathbf{\Pi}$.*

Definition 3.1 considers any pair of a balancing network $\mathcal{B}$ and a property $\mathbf{\Pi}$ and specifies when we can say that $\mathbf{\Pi}$ is closed under the nullity of $\mathcal{B}$. Roughly speaking, we can say so if for every non-negative input vector of $\mathcal{B}$ such that the corresponding output vector satisfies the property, and for every non-negative vector that is null for $\mathcal{B}$, overimposing these two vectors as an input to $\mathcal{B}$ results in an output vector that satisfies $\mathbf{\Pi}$. By using only non-negative input and null vectors in Definition 3.1, we can reason whether a property of a balancing network is closed under the nullity of the network by examining how the network behaves for tokens only.

3.2 Main Result

In the next two claims, we will establish that being closed under the nullity of a balancing network is a necessary and sufficient condition for a property satisfied by the network in the presence of tokens alone to be preserved under the introduction of antitokens. We start with the sufficiency of the characterization.

Proposition 3.1 *Fix any property $\mathbf{\Pi}$, which is closed under the nullity of a balancing network $\mathcal{B} : \mathbf{x}^{(w_{\text{in}})} \rightarrow \mathbf{y}^{(w_{\text{out}})}$. Assume that $\mathbf{y}^{(w_{\text{out}})} \in \mathbf{\Pi}$ whenever $\mathbf{x}^{(w_{\text{in}})}$ is non-negative. Then, $\mathcal{B}$ satisfies property $\mathbf{\Pi}$.*

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

Construct from $\mathbf{x}^{(w_{\text{in}})}$ a non-negative input vector $\tilde{\mathbf{x}}^{(w_{\text{in}})}$ such that for each index i, $0 \leq i \leq w_{\text{in}} - 1$:

- if $x_i \geq 0$, then $\tilde{x}_i = 0$;
- if $x_i < 0$, then $\tilde{x}_i$ is the least positive multiple of $W_{\text{out}}(\mathcal{B})$ such that $x_i + \tilde{x}_i \geq 0$.

Intuitively, $\tilde{\mathbf{x}}^{(w_{\text{in}})}$ supplies positive multiples of $W_{\text{out}}(\mathcal{B})$ to $\mathbf{x}^{(w_{\text{in}})}$ so that it brings each entry of the resulting vector above zero.

Clearly, $W_{\text{out}}(\mathcal{B})$ divides $\tilde{\mathbf{x}}^{(w_{\text{in}})}$. By Lemma 2.3, it follows that $\tilde{\mathbf{x}}^{(w_{\text{in}})}$ is a null vector of $\mathcal{B}$. We apply Lemma 2.1(2) with $\tilde{\mathbf{x}}^{(w_{\text{in}})}$ for $\mathbf{x}_1^{(w_{\text{in}})}$, $\mathbf{0}^{(w_{\text{in}})}$ for $\mathbf{x}_2^{(w_{\text{in}})}$, and $\mathbf{x}^{(w_{\text{in}})}$ for $\mathbf{x}^{(w_{\text{in}})}$; we obtain that

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

so that

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

Since $\tilde{\mathbf{x}}^{(w_{\text{in}})}$ is a null vector of $\mathcal{B}$, it follows, by Lemma 2.4, that

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

Since each entry of $\tilde{\mathbf{x}}^{(w_{\text{in}})} + \mathbf{x}^{(w_{\text{in}})}$ lies in the interval $[0, W_{\text{out}}(\mathcal{B})]$, it follows, by assumption, that $\mathcal{B}(\tilde{\mathbf{x}}^{(w_{\text{in}})} + \mathbf{x}^{(w_{\text{in}})}) \in \mathbf{\Pi}$. By Lemma 2.5, it follows that $-\mathbf{x}^{(w_{\text{in}})}$ is null. Since $\mathbf{\Pi}$ is closed under the nullity of $\mathcal{B}$, it follows that $\mathcal{B}(\mathbf{x}^{(w_{\text{in}})}) \in \mathbf{\Pi}$, as needed. ■

We continue to show that a network may not satisfy any property that is not closed under its nullity.

Proposition 3.2 *Fix a property $\mathbf{\Pi}$, such that $\mathbf{\Pi}$ is not closed under the nullity of a balancing network $\mathcal{B} : \mathbf{x}^{(w_{\text{in}})} \to \mathbf{y}^{(w_{\text{out}})}$. Then, $\mathcal{B}$ does not satisfy property $\mathbf{\Pi}$.*

Proof: Assume, by way of contradiction, that $\mathcal{B}$ satisfies property $\mathbf{\Pi}$. Consider any arbitrary input vector $\mathbf{x}^{(w_{\text{in}})}$ and any null vector $\tilde{\mathbf{x}}^{(w_{\text{in}})}$ of $\mathcal{B}$.

Since $\mathcal{B}$ satisfies $\mathbf{\Pi}$, it follows that $\mathcal{B}(\mathbf{x}^{(w_{\text{in}})}) \in \mathbf{\Pi}$. Since, $\tilde{\mathbf{x}}^{(w_{\text{in}})}$ is a null vector of $\mathcal{B}$, by Lemma 2.1(2), it follows that $\mathcal{B}(\mathbf{x}^{(w_{\text{in}})}) + \mathcal{B}(\tilde{\mathbf{x}}^{(w_{\text{in}})}) = \mathcal{B}(\mathbf{x}^{(w_{\text{in}})} + \tilde{\mathbf{x}}^{(w_{\text{in}})})$. Since $\mathcal{B}$ satisfies $\mathbf{\Pi}$, it follows that $\mathcal{B}(\mathbf{x}^{(w_{\text{in}})} + \tilde{\mathbf{x}}^{(w_{\text{in}})}) \in \mathbf{\Pi}$. Thus, $\mathcal{B}(\mathbf{x}^{(w_{\text{in}})}) + \mathcal{B}(\tilde{\mathbf{x}}^{(w_{\text{in}})}) \in \mathbf{\Pi}$.

Since $\mathbf{x}^{(w_{\text{in}})}$ and $\tilde{\mathbf{x}}^{(w_{\text{in}})}$ were chosen to be any aritrary input and null vectors of $\mathcal{B}$, respectively, it follows that $\mathbf{\Pi}$ is closed under the nullity of $\mathcal{B}$. A contradiction. ■

We are now ready to show our main result.

Theorem 3.3 *Fix a property* $\mathbf{\Pi}$*. Consider any balancing network* $\mathcal{B} : \mathbf{x}^{(w_{\mathrm{in}})} \to \mathbf{y}^{(w_{\mathrm{out}})}$ *such that* $\mathbf{y}^{(w_{\mathrm{out}})} \in \mathbf{\Pi}$ *whenever* $\mathbf{x}^{(w_{\mathrm{in}})}$ *has all of its entries in the interval* $[0, W_{\mathrm{out}}(\mathcal{B})]$*. Then,* $\mathcal{B}$ *satisfies* $\mathbf{\Pi}$ *if and only if* $\mathbf{\Pi}$ *is closed under the nullity of* $\mathcal{B}$*.*

Proof: Assume first that $\mathbf{\Pi}$ is closed under the nullity of $\mathcal{B}$. Since $\mathbf{y}^{(w_{\mathrm{out}})} \in \mathbf{\Pi}$ whenever $\mathbf{x}^{(w_{\mathrm{in}})}$ has all of its entries in the interval $[0, W_{\mathrm{out}}(\mathcal{B})]$, Proposition 3.1 implies that $\mathcal{B}$ satisfies property $\mathbf{\Pi}$, as needed.

Assume now that $\mathcal{B}$ satisfies property $\mathbf{\Pi}$. By Proposition 3.2, $\mathbf{\Pi}$ is closed under the nullity of $\mathcal{B}$, as needed. ■

An immediate corollary of Theorem 3.3 precisely identifies the properties satisfied by balancing networks that are preserved by the introduction of antitokens.

Corollary 3.4 *Fix a property* $\mathbf{\Pi}$*. Consider any balancing network* $\mathcal{B} : \mathbf{x}^{(w_{\mathrm{in}})} \to \mathbf{y}^{(w_{\mathrm{out}})}$ *such that* $\mathbf{y}^{(w_{\mathrm{out}})} \in \mathbf{\Pi}$ *whenever* $\mathbf{x}^{(w_{\mathrm{in}})}$ *is a non-negative input vector. Then,* $\mathcal{B}$ *satisfies* $\mathbf{\Pi}$ *if and only if* $\mathbf{\Pi}$ *is closed under the nullity of* $\mathcal{B}$*.*

3.3 Remarks

Since boundedness properties and the threshold property were shown in [2, 5] to be preserved under the introduction of antitokens, Corollary 3.4 implies that these properties are closed with respect to the nullity of any balancing network satisfying each in particular.

We remark that Proposition 3.1 implies that, given any balancing network $\mathcal{B}$, any property that is closed under the nullity of $\mathcal{B}$ is a *finiteness* property [6]: in order to verify that $\mathcal{B}$ satisfies the property $\mathbf{\Pi}$, it satisfies to verify that all vectors with entries in the (finite) interval $[0, W_{\mathrm{out}}(\mathcal{B})]$ satisfy $\mathbf{\Pi}$. This is the *first* finiteness result established for properties satisfied by balancing networks that are traversed by both tokens and antitokens.

We remark the the proofs in [2, 5] established that the boundedness properties and the threshold property, respectively, are preserved by the introduction of antitokens as follows:

- They first characterized the output of a network satisfying the property on a null vector of the network; it was found that this output is a constant vector if the network satisfies the boundedness property, or a *saturated vector* [5] if the network satisfies the threshold property;

- they used this characterization and the closure under addition with such a vector (explicitly assumed for the boundedness properties or established for the threshold property) to show the preservation property.

These are the two main ideas that we abstracted out from these previous works [2, 5]. We used these ideas, and their instantiation for specific properties, to define when a general (abstract) property is closed under the nullity of a balancing network. We used this definition to carry out our own proofs in a property-independent setting.

4 Conclusion

We have provided a combinatorial characterization of the properties satisfied by balancing networks traversed by tokens alone that are preserved when antitokens are introduced. Our results close the main problem left open by the results in [2, 5]. An interesting question still left open by our work is to provide a corresponding characterization for *randomized* balancing networks [3].

References

[1] E. Aharonson and H. Attiya, "Counting Networks with Arbitrary Fan-Out," *Distributed Computing,* Vol. 8, pp. 163–169, 1995.

[2] W. Aiello, C. Busch, M. Herlihy, M. Mavronicolas, N. Shavit, and D. Touitou, "Supporting Increment and Decrement Operations in Balancing Networks," *Proceedings of the 16th International Symposium on Theoretical Aspects of Computer Science,* G. Meinel and S. Tison eds., pp. 377–386, Vol. 1563, Lecture Notes in Computer Science, Springer-Verlag, Trier, Germany, March 1999. Full version accepted to *Chicago Journal of Theoretical Computer Science.*

[3] W. Aiello, R. Venkatesan and M. Yung, "Coins, Weights and Contention in Balancing Networks," *Proceedings of the 13th Annual ACM Symposium on Principles of Distributed Computing,* pp. 193–205, Los Angeles, California, August 1994.

[4] J. Aspnes, M. Herlihy and N. Shavit, "Counting Networks," *Journal of the ACM,* Vol. 41, No. 5, pp. 1020–1048, September 1994.

[5] C. Busch, N. Demetriou, M. Herlihy and M. Mavronicolas, "Threshold Counters with Increments and Decrements," *Proceedings of the 6th International Colloquium on Structural Information and Communication Complexity,* C. Gavoille, J.-C. Bermond and A. Raspalid eds., pp. 47–61, Proceedings in Informatics 5, Carleton Scientific, Lacanau, France, July 1999.

[6] C. Busch and M. Mavronicolas, "A Combinatorial Treatment of Balancing Networks," *Journal of the ACM,* Vol. 43, No. 5, pp. 794–839, September 1996.

[7] C. Busch and M. Mavronicolas, "Impossibility Results for Weak Threshold Networks," *Information Processing letters,* Vol. 63, No. 2, pp. 85–90, July 1997.

[8] C. Busch and M. Mavronicolas, "An Efficient Counting Network," *Proceedings of the 1st Merged International Parallel Processing Symposium and IEEE Symposium on Parallel and Distributed Processing,* pp. 380–385, May 1998.

[9] E. W. Dijkstra, "Cooperating Sequential Processes," *Programming Languages,* pp. 43–112, Academic Press, 1968.

[10] E. W. Felten, A. LaMarca and R. Ladner, "Building Counting Networks from Larger Balancers," Technical Report 93-04-09, Department of Computer Science and Engineering, University of Washington, April 1993.

[11] D. Grunwald and S. Vajracharya, "Efficient Barriers for Distributed Shared Memory Computers," *Proceedings of the 8th International Parallel Processing Symposium,* IEEE Computer Society Press, April 1994.

[12] R. Gupta and C. R. Hill, "A Scalable Implementation of Barrier Synchronization Using an Adaptive Tree," *International Journal of Parallel Programming,* Vol. 18, No. 3, pp. 161–180, June 1989.

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

[14] N. Hardavellas, D. Karakos and M. Mavronicolas, "Notes on Sorting and Counting Networks," *Proceedings of the 7th International Workshop on Distributed Algorithms,* A. Schiper ed., pp. 234–248, Vol. 725, Lecture Notes in Computer Science, Springer-Verlag, Lausanne, Switzerland, September 1993.

[15] M. Herlihy, B.-C. Lim and N. Shavit, "Concurrent Counting," *ACM Transactions on Computer Systems,* Vol. 13, No. 4, pp. 343–364, 1995.

[16] M. Herlihy, N. Shavit and O. Waarts, "Linearizable Counting Networks," *Distributed Computing,* Vol. 9, pp. 193–203, 1996.

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

[18] S. Kapidakis and M. Mavronicolas, "Distributed, Low Contention Task Allocation," *Proceedings of the 8th IEEE Symposium on Parallel and Distributed Processing,* pp. 358–365, New Orleans, Louisiana, October 1996.

[19] M. Klugerman and C. G. Plaxton, "Small-Depth Counting Networks," *Proceedings of the 24th Annual ACM Symposium on Theory of Computing,* pp. 417–428, May 1992.

[20] M. Mavronicolas, M. Merritt, and G. Taubenfeld, "Sequentially Consistent versus Linearizable Counting Networks," *Proceedings of the 18th Annual ACM Symposium on Principles of Distributed Computing,* pp. 133–142, May 1999.

[21] N. Shavit and D. Touitou, "Elimination Trees and the Construction of Pools and Stacks," *Theory of Computing Systems,* Vol. 30, No. 6, pp. 545–570, November/December 1997.

[22] N. Shavit, E. Upfal and A. Zemach, "A Steady State Analysis of Diffracting Trees," *Theory of Computing Systems,* Vol. 31, No. 4, pp. 403–423, July/August 1998.

[23] N. Shavit and A. Zemach, "Diffracting trees," *ACM Transactions on Computer Systems,* Vol. 14, No. 4, pp. 385–428, November 1996.

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

[25] S. Zhou, X. Zheng, J. Wang and P. Delisle, "Utopia: A Load Sharing Facility for Large, Heterogeneous Distributed Computer Systems," *Software–Practice and Experience,* Vol. 23, No. 12, pp. 1305–1336, December 1993.