ABELIAN NETWORKS III. THE CRITICAL GROUP

BENJAMIN BOND AND LIONEL LEVINE

ABSTRACT. The critical group of an abelian network is a finite abelian group that governs the behavior of the network on large inputs. It generalizes the sandpile group of a graph. We show that the critical group of an irreducible abelian network acts freely and transitively on recurrent states of the network. We exhibit the critical group as a quotient of a free abelian group by a subgroup containing the image of the Laplacian, with equality in the case that the network is rectangular. We generalize Dhar's burning algorithm to abelian networks, and estimate the running time of an abelian network on an arbitrary input up to a constant additive error.

1. INTRODUCTION

Associated to a finite connected graph G with marked vertex s is a finite abelian group called its *sandpile group*. This group arose independently in three different fields: the Neron model of a curve in arithmetic geometry [27, 28], the abelian sandpile model in statistical physics [12], and discrete potential theory on graphs [2, 6]. In this paper we are going to exhibit the sandpile group as an instance of a more general construction, the critical group $\operatorname{Crit} \mathcal{N}$ of an abelian network $\mathcal{N}$.

The abelian sandpile model (also known as chip-firing) is a discrete dynamical system that redistributes chips on the vertices of G according to certain moves called topplings. From our perspective, the abelian sandpile model is rather special for a few reasons:

(1) The total number of chips is conserved by toppling.
(2) One vertex s plays a distinct role: it can receive chips but cannot topple.
(3) Chips are regarded as indistinguishable.

A goal of this paper is to show that the basic theory of the sandpile group can be adapted to a setting where the first two properties are removed and the third is relaxed.

In our setting the chips are replaced by letters passed between "processors", which are automata located at the vertices of G. Each processor has its own state space and input alphabet. Depending on its internal state and the letter it reads, a processor may pass zero, one or more letters to one or more neighboring processors. For example, in the abelian sandpile model, the processor at a vertex of degree

d has state space $\{0, 1, \ldots, d-1\}$ and a one-letter input alphabet. Whenever it reads a letter, the processor increments its state by one modulo d; if its new state is 0 then it passes one letter to each neighboring processor (and otherwise it passes nothing). Then, regardless of its starting state, reading d letters returns the processor to its starting state and exactly one letter has been passed to each neighbor. This is precisely the toppling move of the sandpile model.

Now consider varying the state transition rule or the message passing rule of one or more processors. In general, the resulting network of automata need not have a conserved quantity, which removes item (1) from the above list. The special vertex s in item (2) may be replaced by a condition that is both weaker and more symmetric, namely that the network halts on all inputs. The input alphabet of a processor need not consist of just one letter, which allows us to relax item (3). In order to generalize the theory of the sandpile group, items (1)–(3) are not essential. Rather, what is essential is that each processor is *abelian*.

1.1. **Abelian processor axioms.** An *abelian network* is a collection of abelian processors indexed by the vertices of a directed graph. The formal definition of an *abelian processor* appears in Section 2. In words, an abelian processor is an automaton with output, satisfying two axioms:

(i) For any initial state and any two input words that are equal up to permutation, the resulting final states are equal.

(ii) For any initial state and any two input words that are equal up to permutation, the resulting output words are equal up to permutation.

For example, suppose a abelian processor starts in some state q and reads the input word ab, and that as a result, it ends in some state q' and outputs the word $ccddc$. If instead we were to input the word ba to the same processor in the same starting state q, then axiom (i) says the final state will still be q'; and axiom (ii) says the resulting output word need not be $ccddc$, but must have exactly 5 letters: 3 c's and 2 d's.

Deepak Dhar [13] proposed abelian networks as models of self-organized criticality in physics, generalizing the abelian sandpile model of [3, 12]. From the point of view of computer science, abelian networks are an interesting class of automata networks because they can compute asynchronously: an abelian network produces the same final output in the same number of steps regardless of the order of events at individual nodes of the network [7].

Examples of abelian networks are surveyed in [7]. They include sandpile and rotor networks and their non-unary cousins, oil and water networks and abelian mobile agents. Besides the sandpile groups, critical groups of abelian networks have been studied in a few other particular cases: rotor networks [32] and height-arrow networks [11]. The group Crit Rotor(G, s) associated to a rotor network with a sink turns out to be isomorphic to the sandpile group, Crit Sand(G, s) [32, 26]. We will see this isomorphism as a case of a more general phenomenon: Homotopic abelian networks have isomorphic critical groups (Corollary 3.12).

The sandpile group has several different constructions. Babai and Toumpakari [1] realized the sandpile group as the minimal ideal of a commutative monoid.

Their approach, developed further in [9], is well-suited for generalizing to abelian networks. In §1.3 we review the small amount of monoid theory we will need. In §2 we recall the definition of an abelian network and relevant results from [7, 8], and prove a few basic lemmas including a local-to-global principle for irreducibility, Lemma 2.6.

1.2. **Main results.** The setting for all of our main results is a finite irreducible abelian network $\mathcal{N}$ that halts on all inputs. In §3.1 we define the critical group $\operatorname{Crit}\mathcal{N}$ and the set $\operatorname{Rec}\mathcal{N}$ of recurrent states, and show that the former acts freely and transitively on the latter. In §3.2 we consider a Markov chain defined by sending random input to the network; we relate the algebraic and probabilistic definitions of "recurrent," and show that the stationary distribution of the chain is uniform on recurrent states. In §3.3 we find the expected time for $\mathcal{N}$ to halt on a given input to a uniform random recurrent state. In §3.4 we give generators and relations for the critical group, and in §3.5 we compute its order.

In §4 we estimate the time for $\mathcal{N}$ to halt on an arbitrary input, up to a constant additive error.

In §5.1 we give an efficient test for whether a given state of $\mathcal{N}$ is recurrent, generalizing Dhar's burning algorithm [12] and Speer's script algorithm [34]. We define a *burning element* to be an input that returns the network to its initial state if and only if that state is recurrent. We show that any integer vector satisfying a certain set of linear inequalities is a burning element. In §5.2 we give an efficient way to find such a vector.

1.3. **Review of monoid actions.** Here we recall a few facts about actions of finite commutative monoids. Most of this material is implicit in the semigroup literature [19, 33, 20, 35]. The proofs are straightforward and can be found in [8]. For refinements and a generalization to certain infinite semigroups, see Grillet [21].

Let M be a commutative monoid, X a set and $\mu : M \times X \to X$ a monoid action. Commutativity of M implies that the relation $\sim$ on X defined by

$$x \sim x' : \qquad \exists m, m' \in M \text{ such that } mx = m'x' \tag{1}$$

is an equivalence relation. We say that μ is *irreducible* if $\sim$ has only one equivalence class.

For the rest of this section we assume that M is finite and μ is irreducible. By finiteness, every $m \in M$ has an idempotent power: $m^j = m^{2j}$ for some $j \geq 1$. The *minimal idempotent* $e := \prod_{f \in M, ff=f} f$ is the unique idempotent accessible from all of M (that is, $ee = e$ and $e \in Mm$ for all $m \in M$).

Lemma 1.1. *eM is an abelian group with identity element e.*

Indeed, the existence of inverses is immediate from $e \in Mm$.

Lemma 1.2. (Recurrent Elements Of An Irreducible Monoid Action) *The following are equivalent for $x \in X$:*

   (1) $x \in My$ for all $y \in X$

(2) $x \in M(mx)$ *for all* $m \in M$
(3) $x \in mX$ *for all* $m \in M$
(4) $x \in eX$
(5) $x = ex$

An element $x \in X$ is called *recurrent* if it satisfies the equivalent conditions of Lemma 1.2. To explain this terminology, suppose we are given a probability distribution α on M such that $\{m \in M : \alpha(m) > 0\}$ generates M as a monoid, and consider the Markov chain on X that transitions from x to mx with probability $\alpha(m)$. The states x satisfying the equivalent conditions of Lemma 1.2 are precisely those visited infinitely often by the Markov chain (see §3.2).

We say that $m \in M$ *acts invertibly* on a subset Y of X if the map $y \mapsto my$ is a permutation of Y.

Lemma 1.3. *Every* $m \in M$ *acts invertibly on* eX.

We say that a monoid action $\mu : M \times X \to X$ is *faithful* if there do not exist distinct elements $m, m' \in M$ such that $mx = m'x$ for all $x \in X$.

Let G be a group with identity element e. Recall that a group action $G \times Y \to Y$ is called *transitive* if $Gy = Y$ for all $y \in Y$, and is called *free* if for all $g \neq e$ there does not exist $y \in Y$ such that $gy = y$. If the action is both transitive and free, then for any two elements $y, y' \in Y$ there is a unique $g \in G$ such that $gy = y'$; in particular, $\#G = \#Y$.

Theorem 1.4. (Group Actions Arising From Monoid Actions) *Let* M *be a finite commutative monoid and* $\mu : M \times X \to X$ *an irreducible action. The restriction of* μ *to* $eM \times eX$ *is a transitive group action*

$$e\mu : eM \times eX \to eX.$$

In addition, if μ *is faithful, then* $e\mu$ *is free.*

2. Abelian networks

We now recall the definition and basic properties of abelian networks, refering the reader to [7, 8] for details.

2.1. **Abelian processors.** Let Q be a set of "states" and $\mathrm{End}\,(Q)$ the monoid of all set maps $Q \to Q$ with the operation of composition. An *abelian processor* with input alphabet I and state space Q is a collection of maps $(t_a)_{a \in I}$ where each $t_a \in \mathrm{End}\,(Q)$, such that $t_a t_b = t_b t_a$ for all $a, b \in I$. This commutativity implies axiom (i) from the introduction.

2.2. **Abelian processors with output.** So far an abelian processor can take input and change state. Next we will enable it to send output, so that it can pass messages to other processors in a network. If a processor has several neighbors in the network, we will allow it to pass a different message to each. So an abelian processor may have several output alphabets (one for each neighbor).

Let U be a set of "output feeds". An *abelian processor with output alphabets* $(A_u)_{u \in U}$ has in addition to the state transition maps t_a a message passing function

$o_a^u : Q \to A_u^*$ for each $a \in I$ and $u \in U$. Here A^* denotes the free monoid of all finite words in an alphabet A. These functions are required to satisfy a commutativity condition: namely, if two input words $w, w' \in I^*$ are equal up to permutation, then for each $u \in U$ the resulting output words in A_u^* are equal up to permutation. This condition is axiom (ii) from the introduction.

2.3. **Abelian networks.** An *abelian network* on a directed graph $G = (V, E)$ with alphabet $A = \sqcup_{v \in V} A_v$ and state space $Q = \prod_{v \in V} Q_v$ is a collection $(\mathcal{P}_v)_{v \in V}$, where each $\mathcal{P}_v$ is an abelian processor with input alphabet A_v, state space Q_v and output alphabets $(A_u)_{(v,u) \in E}$ indexed by the outgoing edges from v. When $\mathcal{P}_v$ in state $q \in Q_v$ processes letter $a \in A_v$, it transitions to state $t_a(q)$ and sends the message $o_a^u(q)$ to each neighboring processor $\mathcal{P}_u$. Note that $o_a^u(q)$ might be the empty word, which signifies that no message is sent.

The total state of an abelian network $\mathcal{N} = (\mathcal{P}_v)_{v \in V}$ is described by an element $\mathbf{q} \in Q$ indicating the internal states of all processors, together with a vector $\mathbf{x} \in \mathbb{Z}^A$ indicating how many letters of each type are waiting to be processed. We use the notation $\mathbf{x}.\mathbf{q}$ for this pair (in which the decimal point is meant to suggest that the states $\mathbf{q}$ represent "fractional letters" that have not yet been output). Note that A is a disjoint union, so each letter belongs to the input alphabet of a unique processor.

2.4. **Sandpiles, rotor networks, toppling networks.** These will be our running examples of abelian networks. Let $G = (V, E)$ be a finite directed graph and $s \in V$ a vertex such that from every other vertex $v \in V$ there is a directed path from v to s. Each processor $\mathcal{P}_v$ has alphabet $A_v = \{v\}$ and state space $Q_v = \mathbb{Z}/r_v\mathbb{Z}$ for a positive integer r_v called the *threshold* of v. The state transition is $t_v(q) = q + 1 \pmod{r_v}$.

It remains to describe the message passing. In the networks $\mathtt{Sand}(G, s)$ and $\mathtt{Rotor}(G, s)$, processor $\mathcal{P}_s$ is a *sink* (which means it has just one state and never passes any messages) and we take $r_v = d_v$, the outdegree of v, for all $v \neq s$. In the sandpile network $\mathtt{Sand}(G, s)$, whenever processor $\mathcal{P}_v$ transitions from state $d_v - 1$ to state 0 it passes d_v letters: one letter u along each outgoing edge (v, u). The message passing for the rotor network $\mathtt{Rotor}(G, s)$ is specified by fixing an ordering $e_1, \ldots, e_{d_v}$ of the outgoing edges from v. Whenever it transitions from state q to $q + 1 \pmod{d_v}$, processor $\mathcal{P}_v$ passes exactly one letter, along the edge e_{q+1}.

Toppling networks are a generalization of sandpiles, where we allow $r_v \neq d_v$; however, when processor $\mathcal{P}_v$ in a toppling network transitions from state $r_v - 1$ to state 0, it passes d_v letters just as in a sandpile network: one letter u along each outgoing edge (v, u). The importance of the sink s in $\mathtt{Sand}(G, s)$ and $\mathtt{Rotor}(G, s)$ is to ensure the network halts on all inputs. Depending on the thresholds r_v, a toppling network may halt on all inputs even if no vertex is a sink.

Rotor and toppling networks are *unary*, meaning that each alphabet A_v is a singleton. See [7] for two examples of non-unary abelian networks, oil and water and abelian mobile agents.

2.5. **Executions.** An *execution* is a finite word $w \in A^*$. It prescribes an order in which letters are to be processed. We write $\pi_w(\mathbf{x}.\mathbf{q})$ for the result of executing w starting from $\mathbf{x}.\mathbf{q}$; this is another pair $\mathbf{x}'.\mathbf{q}'$ that can be computed using the state transitions and message passing functions. It is important to note that some coordinates of $\mathbf{x}'$ may be negative! For example, if w consists of a single letter a and $\mathbf{x}_a = 0$, and processing a does not cause any letters a to be passed, then $\mathbf{x}'_a = -1$. The interpretation is that a processor was instructed to process letter a even though no letter a was present; the processor follows the instruction and keeps track of the "debt" that it is owed one letter a. Note however that messages passed from one processor to another are always nonnegative.

The axioms (see §1.1) of an abelian processor imply that $\pi_w(\mathbf{x}.\mathbf{q})$ depends only on the vector $|w| \in \mathbb{N}^A$ where $|w|_a$ is the number of letters a in w. We write π_w and $\pi_{|w|}$ interchangeably. Writing $w = a_1 \cdots a_r$ and $\pi_{a_1 \cdots a_i}(\mathbf{x}.\mathbf{q}) = \mathbf{x}^i.\mathbf{q}^i$, we say that w is *legal* for $\mathbf{x}.\mathbf{q}$ if $\mathbf{x}^{i-1}_{a_i} \geq 1$ for all $i = 1, \ldots, r$. We say that w is *complete* for $\mathbf{x}.\mathbf{q}$ if $\mathbf{x}^r \leq \mathbf{0}$ (inequalities between vectors are coordinatewise). In words, a legal execution is one that incurs no "debts," and a complete execution is one that removes all letters from the network. Note that if w is both legal and complete, then $\mathbf{x}^i \geq \mathbf{0}$ for $i = 1, \ldots, r-1$ and $\mathbf{x}^r = \mathbf{0}$.

The least action principle [7, Lemma 4.3] says that if w is any legal execution for $\mathbf{x}.\mathbf{q}$ and w' is any complete execution for $\mathbf{x}.\mathbf{q}$, then $|w| \leq |w'|$. It follows that $|w|$ is the same for all complete legal executions of $\mathbf{x}.\mathbf{q}$. If there exists a complete legal execution for $\mathbf{x}.\mathbf{q}$ then we say that $\mathcal{N}$ *halts* on input $\mathbf{x}.\mathbf{q}$. In this case the *odometer* $[\mathbf{x}.\mathbf{q}]$ is defined as

$$[\mathbf{x}.\mathbf{q}] := |w|$$

where w is any complete legal execution for $\mathbf{x}.\mathbf{q}$. The odometer is a vector in $\mathbb{N}^A$ whose ath coordinate is the total number of letters a processed.

The final state $\mathbf{q}^r_v$ of each processor $\mathcal{P}_v$ can be determined from its initial state $\mathbf{q}_v$ and the odometer coordinates $[\mathbf{x}.\mathbf{q}]_a$ for $a \in A_v$, namely

$$\mathbf{q}^r_v = \left(\prod_{a \in A_v} t_a^{[\mathbf{x}.\mathbf{q}]_a} \right) \mathbf{q}_v. \tag{2}$$

where the product denotes composition of the commuting maps t_a. To make its dependence on $\mathbf{x}$ explicit we will use the notation $\mathbf{x} \rhd\rhd \mathbf{q}$ for the final state $\mathbf{q}^r$. Like the odometer $[\mathbf{x}.\mathbf{q}]$, the state $\mathbf{x} \rhd\rhd \mathbf{q}$ depends only on $\mathbf{x}$ and $\mathbf{q}$ and not on the choice of complete legal execution.

2.6. **The global action.** Let $\mathcal{N}$ be an abelian network that halts on all inputs: that is, $\mathbf{x}.\mathbf{q}$ has a finite complete legal execution for all $\mathbf{x} \in \mathbb{N}^A$ and all $\mathbf{q} \in Q$. We will see shortly that $(\mathbf{x}, \mathbf{q}) \mapsto \mathbf{x} \rhd\rhd \mathbf{q}$ defines a monoid action $\mathbb{N}^A \times Q \to Q$. This *global action* is the main object of interest in the present paper: we will use this action to define the critical group in §3 and characterize the recurrent states of this action in §5. We have chosen the notation $\rhd\rhd$ to distinguish the global action from the *local action* $\rhd$ of [8]; below we recall how $\rhd$ is defined and relate the two actions.

Lemma 2.1. *If w is a legal execution from $\mathbf{x}.\mathbf{q}$ to $\mathbf{y}.\mathbf{r}$, then*

$$\mathbf{x} \rhd\rhd \mathbf{q} = \mathbf{y} \rhd\rhd \mathbf{r}$$

and

$$[\mathbf{x}.\mathbf{q}] = |w| + [\mathbf{y}.\mathbf{r}].$$

Proof. Let w' be a complete legal execution for $\mathbf{y}.\mathbf{r}$. Then the concatenation ww' is a complete legal execution for $\mathbf{x}.\mathbf{q}$. ☐

The next lemma verifies that $\rhd\rhd$ defines a monoid action of $\mathbb{N}^A$ on Q.

Lemma 2.2. *For $\mathbf{x}, \mathbf{y} \in \mathbb{N}^A$ and $\mathbf{q} \in Q$ we have*

$$(\mathbf{x} + \mathbf{y}) \rhd\rhd \mathbf{q} = \mathbf{y} \rhd\rhd (\mathbf{x} \rhd\rhd \mathbf{q})$$

and

$$[(\mathbf{x} + \mathbf{y}).\mathbf{q}] = [\mathbf{x}.\mathbf{q}] + [\mathbf{y}.(\mathbf{x} \rhd\rhd \mathbf{q})].$$

Proof. If w is a complete legal execution for $\mathbf{x}.\mathbf{q}$, then w is a legal execution from $(\mathbf{x} + \mathbf{y}).\mathbf{q}$ to $\mathbf{y}.(\mathbf{x} \rhd\rhd \mathbf{q})$. Since $|w| = [\mathbf{x}.\mathbf{q}]$ the result follows from Lemma 2.1. ☐

2.7. **The local action.** In [8] we defined a monoid action $\rhd$ of $\mathbb{N}^A$ on $\mathbb{Z}^A \times Q$,

$$\mathbf{x} \rhd (\mathbf{y}.\mathbf{q}) = \pi_\mathbf{x}((\mathbf{x} + \mathbf{y}).\mathbf{q}).$$

This is called the *local action* because each processor $\mathcal{P}_v$ processes only the letters that were added at v (namely $\mathbf{x}_a$ letters a for each $a \in A_v$) and not any additional letters passed from other processors. We write $\mathbf{x} \rhd \mathbf{q}$ as a shorthand for $\mathbf{x} \rhd (\mathbf{0}.\mathbf{q})$.

The next lemma relates the local and global actions.

Lemma 2.3. *If $\mathbf{x} \rhd \mathbf{q} = \mathbf{y}.\mathbf{r}$, then $\mathbf{x} \rhd\rhd \mathbf{q} = \mathbf{y} \rhd\rhd \mathbf{r}$ and $[\mathbf{x}.\mathbf{q}] = \mathbf{x} + [\mathbf{y}.\mathbf{r}]$.*

Proof. If $\mathbf{x} \rhd \mathbf{q} = \mathbf{y}.\mathbf{r}$ then there is a legal execution w from $\mathbf{x}.\mathbf{q}$ to $\mathbf{y}.\mathbf{r}$ with $|w| = \mathbf{x}$, so the result follows from Lemma 2.1. ☐

One way to compute the global action $\rhd\rhd$ is by iterative application of $\rhd$: given $\mathbf{x} \in \mathbb{N}^A$ and $\mathbf{q} \in Q$, set $\mathbf{x}_0 = \mathbf{x}$ and $\mathbf{q}_0 = \mathbf{q}$ and

$$\mathbf{x}_n.\mathbf{q}_n = \mathbf{x}_{n-1} \rhd \mathbf{q}_{n-1} \tag{3}$$

for $n \geq 1$. This amounts to making a particular choice of execution, called *parallel update*. In parallel update the execution occurs in rounds $n = 1, 2, \ldots$. At the beginning of round n there are $(\mathbf{x}_n)_a$ letters a waiting to be processed for each $a \in A$. During round n we execute a word w_n with $|w_n| = \mathbf{x}_n$. The concatenation $w = w_1 w_2 \cdots$ is a legal execution. Since $\mathcal{N}$ halts on all inputs, every legal execution is finite, so there is some N such that all words w_n for $n > N$ are empty. Then $\mathbf{x}_n = \mathbf{0}$ for all $n \geq N$, and $w = w_1 \cdots w_N$ is a finite complete legal execution. In particular, the final state is given by

$$\mathbf{x} \rhd\rhd \mathbf{q} = \mathbf{q}_N$$

and the odometer is

$$[\mathbf{x}.\mathbf{q}] = |w| = \sum_{n=1}^{N} \mathbf{x}_n.$$

We record a few more identities to be used later. We extend the domain of $\rhd\!\rhd$ to $\mathbb{N}^A \times Q$ by defining for $\mathbf{y} \in \mathbb{N}^A$

$$\mathbf{x} \rhd\!\rhd (\mathbf{y}.\mathbf{q}) := (\mathbf{x} + \mathbf{y}) \rhd\!\rhd \mathbf{q}.$$

Lemma 2.4. *Given* $\mathbf{x} \in \mathbb{N}^A$ *and* $\mathbf{q} \in Q$, *let* $\mathbf{k} = [\mathbf{x}.\mathbf{q}]$. *Then*

(i) $\pi_{\mathbf{k}}(\mathbf{x}.\mathbf{q}) = \mathbf{0}.(\mathbf{x} \rhd\!\rhd \mathbf{q})$
(ii) $\mathbf{k} \rhd (\mathbf{x}.\mathbf{q}) = \mathbf{k}.(\mathbf{x} \rhd\!\rhd \mathbf{q})$
(iii) $\mathbf{x} \rhd\!\rhd (\mathbf{y} \rhd \mathbf{q}) = (\mathbf{x} + \mathbf{y}) \rhd\!\rhd \mathbf{q}$

Proof. Part (i) follows from the definition of $\mathbf{x} \rhd\!\rhd \mathbf{q}$.

Part (ii) follows from (i) since

$$\mathbf{k} \rhd (\mathbf{x}.\mathbf{q}) = \pi_{\mathbf{k}}((\mathbf{x} + \mathbf{k}).\mathbf{q}) = \mathbf{k}.(\mathbf{x} \rhd\!\rhd \mathbf{q}).$$

To prove part (iii), both states $\mathbf{x} \rhd\!\rhd (\mathbf{y} \rhd \mathbf{q})$ and $(\mathbf{x} + \mathbf{y}) \rhd\!\rhd \mathbf{q}$ are the result of performing a complete legal execution for $(\mathbf{x} + \mathbf{y}).\mathbf{q}$, and any two complete legal executions for $(\mathbf{x} + \mathbf{y}).\mathbf{q}$ result in the same final state. $\square$

2.8. **Local monoids.** The *transition monoid* of an abelian processor with state space Q and transition maps $t_a : Q \to Q$ is the submonoid $M = \langle t_a \rangle_{a \in A} \subseteq \text{End}\,(Q)$, where $\text{End}\,(Q)$ is the monoid of all set maps $Q \to Q$ with the operation of composition. Since M is defined as a submonoid of $\text{End}\,(Q)$ it has a faithful monoid action $M \times Q \to Q$.

Each processor in an abelian network $\mathcal{N} = (\mathcal{P}_v)_{v \in V}$ has a transition monoid

$$M_v := \langle t_a \rangle_{a \in A_v} \subset \text{End}\,(Q_v). \tag{4}$$

We call this M_v the *local monoid* at v. The product $\prod_{v \in V} M_v$ acts coordinatewise on $Q = \prod_{v \in V} Q_v$. To relate this action to the global and local actions defined above, let $t_v : \mathbb{N}^{A_v} \to M_v$ be the monoid homomorphism sending basis elements 1_a to the commuting generators t_a, and (recalling $A = \sqcup A_v$) write

$$t : \mathbb{N}^A \to \prod_{v \in V} M_v$$

for the Cartesian product of the maps t_v. Each t_v is surjective, so t is surjective. Equations (2) (there is one equation for each $v \in V$) can be written more succinctly as the single equation

$$\mathbf{x} \rhd\!\rhd \mathbf{q} = t([\mathbf{x}.\mathbf{q}])\mathbf{q}. \tag{5}$$

To relate t to the local action, note that if $\mathbf{x} \rhd \mathbf{q} = \mathbf{y}.\mathbf{r}$ then $\mathbf{r} = t(\mathbf{x})\mathbf{q}$.

2.9. **Global monoid.** If $\mathcal{N}$ halts on all inputs, then we can view the entire network as a single abelian processor (see [7, Lemma 4.7]) with input alphabet $A = \sqcup A_v$ and state space $Q = \prod Q_v$. In this case $\mathcal{N}$ has a *global monoid*, defined by

$$M := \langle \tau_a \rangle_{a \in A} \subset \text{End}\,(Q) \tag{6}$$

where $\tau_a(\mathbf{q}) := 1_a \rhd\!\rhd \mathbf{q}$. Note that M is not the same as the product of local monoids $\prod M_v$ of (4): the local monoids depend only on the state transition maps t_a, but M depends also on the message passing functions (because $\rhd\!\rhd$ does).

Write

$$\tau : \mathbb{N}^A \to M \qquad (7)$$

for the monoid homomorphism sending basis elements 1_a to generators τ_a. By Lemma 2.2 we have $\mathbf{x} \rhd\rhd \mathbf{q} = \tau(\mathbf{x})\mathbf{q}$.

If $\mathcal{N}$ is a finite abelian network (that is, V is finite, and the alphabet A_v and the state space Q_v of each processor are finite) then M is a finite commutative monoid. In this case we denote by e the minimal idempotent of M (§1.3). We will use the following property of e repeatedly.

Lemma 2.5. *For any $\mathbf{x} \in \mathbb{N}^A$ there exists $\mathbf{z} \geq \mathbf{x}$ such that $\tau(\mathbf{z}) = e$.*

Proof. Since e is accessible from all of M we have $\tau(\mathbf{x})m = e$ for some $m \in M$. Since τ is surjective we have $m = \tau(\mathbf{y})$ for some $\mathbf{y} \in \mathbb{N}^A$. Now take $\mathbf{z} = \mathbf{x} + \mathbf{y}$. □

2.10. **Locally irreducible implies globally irreducible.** An abelian processor with transition monoid M and state space Q is called *irreducible* if the defining action $M \times Q \to Q$ is irreducible (§1.3). Next we prove a local-to-global principle for irreducibility.

Lemma 2.6. *Let $\mathcal{N} = (\mathcal{P}_v)_{v \in V}$ be an abelian network that halts on all inputs. If each processor $\mathcal{P}_v$ is irreducible, then $\mathcal{N}$ is irreducible.*

Proof. Let $\mathbf{q}, \mathbf{q}' \in Q$. For each $v \in V$, since $\mathcal{P}_v$ is irreducible, there exist $m_v, m_v' \in M_v$ such that $m_v \mathbf{q}_v = m_v' \mathbf{q}_v'$. Since t is surjective we can choose $\mathbf{x}, \mathbf{x}' \in \mathbb{N}^A$ with $t(\mathbf{x}) = m$ and $t(\mathbf{x}') = m'$. Write

$$\mathbf{x} \rhd \mathbf{q} = \mathbf{y}.\mathbf{r}, \qquad \mathbf{x}' \rhd \mathbf{q}' = \mathbf{y}'.\mathbf{r}'$$

where $\mathbf{r} = \mathbf{r}'$ since $\mathbf{r}_v = m_v \mathbf{q}_v = m_v' \mathbf{q}_v' = \mathbf{r}_v'$ for all $v \in V$. Then by Lemma 2.4(iii),

$$(\mathbf{x} + \mathbf{y}') \rhd\rhd \mathbf{q} = \mathbf{y}' \rhd\rhd (\mathbf{x} \rhd \mathbf{q})$$
$$= \mathbf{y}' \rhd\rhd (\mathbf{y}.\mathbf{r})$$
$$= (\mathbf{y} + \mathbf{y}') \rhd\rhd \mathbf{r}$$
$$= \mathbf{y} \rhd\rhd (\mathbf{y}'.\mathbf{r})$$
$$= \mathbf{y} \rhd\rhd (\mathbf{x}' \rhd \mathbf{q}')$$
$$= (\mathbf{x}' + \mathbf{y}) \rhd\rhd \mathbf{q}'.$$

Hence $\tau(\mathbf{x}+\mathbf{y}')\mathbf{q} = \tau(\mathbf{x}'+\mathbf{y})\mathbf{q}'$, so the global action $M \times Q \to Q$ is irreducible. □

In light of Lemma 2.6 we will drop "locally" from "locally irreducible" when referring to an abelian network that halts on all inputs.

2.11. **Total kernel and production matrix.** In [8] we used the local action to associate two basic algebraic objects to an irreducible abelian network, the *total kernel K* and *production map $P : K \to \mathbb{Z}^A$*. Together K and P constitute a kind of coarse-grained description of an abelian network: They do not specify the network in full detail, but they capture its "large scale" features (for instance, the asymptotic behavior on large inputs). In this paper we will see that many properties of interest depend only on K and P.

Let e_v be the minimal idempotent of the local monoid M_v. A state $\mathbf{q} \in Q$ is called *locally recurrent* if $\mathbf{q}_v \in e_v Q_v$ for all $v \in V$. By Lemma 1.3, each $m \in M_v$ acts invertibly on $e_v Q_v$, so we have a group action of $\mathbb{Z}^{A_v}$ on $e_v Q_v$. The total kernel is defined as

$$K = \prod_{v \in V} K_v \subset \mathbb{Z}^A$$

where K_v is the kernel of the action $\mathbb{Z}^{A_v} \times e_v Q_v \to e_v Q_v$. For example, in the case of a sandpile network (§2.4) we have $e_v Q_v = Q_v = \{0, 1, \ldots, d_v - 1\}$ (all states are locally recurrent) and $\mathbb{Z}^{A_v} = \mathbb{Z}$ acts by $q \mapsto q + 1 \pmod{d_v}$. So in this case $K_v = d_v \mathbb{Z}$.

If $\mathcal{N}$ is a finite abelian network, then its total kernel K is a finite index subgroup of $\mathbb{Z}^A$ ([8, Lemma 4.5]). In particular, it is generated as a group by $K \cap \mathbb{N}^A$. The nonnegative points in K can be characterized as follows.

Lemma 2.7. [8, Lemma 4.8] *If $\mathcal{N}$ is finite and irreducible, then the following are equivalent for $\mathbf{x} \in \mathbb{N}^A$.*

(1) $t(\mathbf{x})\mathbf{q} = \mathbf{q}$ *for some locally recurrent $\mathbf{q}$.*
(2) $t(\mathbf{x})\mathbf{q} = \mathbf{q}$ *for all locally recurrent $\mathbf{q}$.*
(3) $\mathbf{x} \in K$.

The total kernel depends only on the state transition maps. Next we define the production map, which depends also on the message passing functions. Given a locally recurrent state $\mathbf{q}$ and a vector $\mathbf{k} \in K \cap \mathbb{N}^A$ we have

$$\mathbf{k} \triangleright \mathbf{q} = P_{\mathbf{q}}(\mathbf{k}).\mathbf{q} \tag{8}$$

for some vector $P_{\mathbf{q}}(\mathbf{k}) \in \mathbb{N}^A$.

Nonnegative elements of the total kernel can be thought of as reset vectors: if $\mathbf{k} \in K \cap \mathbb{N}^A$ then processing $\mathbf{k}_a$ letters a for all $a \in A$ returns all processors to their initial (locally recurrent) states, and the vector $P_{\mathbf{q}}(\mathbf{k})$ specifies how many letters of each type are passed as a result. For example, in the case of a sandpile network, the vectors $\{d_v 1_v\}_{v \in V}$ are an $\mathbb{N}$-basis for K, and the production map is $\mathbb{N}$-linear with

$$P_{\mathbf{q}}(d_v 1_v) = \sum_{(v,u) \in E} 1_u.$$

This equation just says that the local action of d_v letters v is to send one letter u along each outgoing edge (v, u) and to return the processor at v to its initial state. Note that $P_{\mathbf{q}}$ does not depend on $\mathbf{q}$ in this example! The next lemma generalizes these two observations (linearity and independence of $\mathbf{q}$) to a general abelian network.

Lemma 2.8. [8, Lemmas 4.6 and 4.9] *Let $\mathcal{N}$ be a finite abelian network.*

(1) $P_{\mathbf{q}} : K \cap \mathbb{N}^A \to \mathbb{N}^A$ *extends to a group homomorphism $K \to \mathbb{Z}^A$.*
(2) *If $\mathcal{N}$ is irreducible then $P_{\mathbf{q}} = P_{\mathbf{r}}$ for all $\mathbf{q}, \mathbf{r} \in Q$.*

In light of (2), when $\mathcal{N}$ is irreducible we will often drop the subscript and denote the production map simply by P. In a slight abuse of notation, we also denote

by P the $A \times A$ matrix of the linear map $\mathbb{Q}^A \to \mathbb{Q}^A$ obtained by tensoring the production map with $\mathbb{Q}$. More explicitly, for any $x \in \mathbb{Q}^A$ there exists n such that $nx \in K$ and we set $P(x) := (1/n)P(nx)$. The *Laplacian* of $\mathcal{N}$ is defined as the $A \times A$ matrix

$$L = (I - P)D \tag{9}$$

where I is the $A \times A$ identity matrix and D is the diagonal matrix with diagonal entries

$$r_a = \min\{m \geq 1 : m1_a \in K\}. \tag{10}$$

This set is nonempty because K is a finite index subgroup of $\mathbb{Z}^A$. For example, in a sandpile network (§2.4) we have $r_a = d_a$, the outdegree of vertex a. In general one can think of r_a as a "reset number": If $a \in A_v$ then inputting r_a letters a resets processor $\mathcal{P}_v$ to its initial (locally recurrent) state.

Theorem 2.9. [8, Theorem 5.6 and Corollary 6.4] *Let $\mathcal{N}$ be a finite irreducible abelian network $\mathcal{N}$ with production matrix P and Laplacian L. The following are equivalent.*

    (1) *$\mathcal{N}$ halts on all inputs.*
    (2) *The spectral radius of P is strictly less than 1.*
    (3) *All principal minors L are positive.*

The proof in [8] uses Dickson's Lemma and the Perron-Frobenius theorem. We sketch here just the equivalence of (1) and (2). If P has Perron-Frobenius eigenvalue $\lambda \geq 1$, then the corresponding eigenvector is used to construct an input for which $\mathcal{N}$ fails to halt. Conversely, if $\mathcal{N}$ fails to halt on some input, then by Dickson's Lemma there exists a legal execution from some $\mathbf{y}.\mathbf{q}$ to some $\mathbf{z}.\mathbf{q}$ (with the *same* state $\mathbf{q}$) satisfying $\mathbf{z} \geq \mathbf{y}$. This implies $P(\mathbf{y}) \geq \mathbf{y}$ and hence $\lambda \geq 1$.

Regarding (3), note that L need not be symmetric. A real square matrix with nonpositive off-diagonal entries and positive principal minors is called a *toppling matrix* (or "M-matrix"). The positive principal minor condition is one way to extend the notion of "positive definite" to non-symmetric matrices. Fiedler and Ptak [16, Theorem 4.3] list of thirteen equivalent conditions of which this is one. The classic example of a toppling matrix is the reduced Laplacian of a strongly connected directed graph. Extending theorems about graph Laplacians to toppling matrices is an ongoing topic of research [31, 22].

3. Critical Group

The main results begin in this section. Throughout this section, we take $\mathcal{N}$ to be a finite irreducible abelian network that halts on all inputs.

3.1. **Action on recurrent states.** Since the global monoid M of (6) is finite and commutative it has a minimal idempotent e, and eM is a finite abelian group with identity element e (see §1.3).

Definition 3.1. The *critical group* $\mathrm{Crit}\,\mathcal{N}$ is the group eM.

Definition 3.2. A state $x \in Q$ is *recurrent* if it satisfies the equivalent conditions of Lemma 1.2 (for example, $x = ex$) for the defining action $M \times Q \to Q$.

Denote by $\operatorname{Rec} \mathcal{N}$ the set of recurrent states of $\mathcal{N}$.

For example, if $\mathcal{N} = \texttt{Rotor}(G, s)$ is a simple rotor network on a directed graph G with sink vertex s, then $\operatorname{Rec} \mathcal{N}$ can be identified with spanning trees of G oriented toward s [23, Lemma 3.16]. The critical group of $\mathcal{N}$ is isomorphic to the sandpile group $\operatorname{Crit} \texttt{Sand}(G, s)$ [32, 26]. We will deduce this isomorphism as a special case of Theorem 3.11, below.

The following theorem generalizes [23, Lemmas 3.13 and 3.17], where it was shown that $\operatorname{Crit} \texttt{Rotor}(G, s)$ acts freely and transitively on the set of spanning trees of G oriented toward s.

Theorem 3.3. *Let $\mathcal{N}$ be a finite irreducible abelian network that halts on all inputs. The action of the critical group on recurrent states*

$$\operatorname{Crit} \mathcal{N} \times \operatorname{Rec} \mathcal{N} \to \operatorname{Rec} \mathcal{N}$$

is free and transitive. In particular, $\#\operatorname{Crit} \mathcal{N} = \#\operatorname{Rec} \mathcal{N}$.

Proof. Let M be the transition monoid of $\mathcal{N}$, and let e be the minimal idempotent of M. Then $\operatorname{Crit} \mathcal{N} = eM$ and $\operatorname{Rec} \mathcal{N} = eQ$. The monoid action $M \times Q \to Q$ is faithful by definition and irreducible by Lemma 2.6. Hence the group action $eM \times eQ \to eQ$ is free and transitive by Theorem 1.4. $\qquad\square$

3.2. **Markov chain.** Next we formalize one way in which $\operatorname{Crit} \mathcal{N}$ and its action on $\operatorname{Rec} \mathcal{N}$ govern the "long term behavior" of $\mathcal{N}$. Let α be a probability distribution on the total alphabet A. Consider the Markov chain $(\mathbf{q}_n)_{n \geq 0}$ on state space Q where the initial state $\mathbf{q}_0$ can be arbitrary, and subsequent states are defined by

$$\mathbf{q}_{n+1} = 1_{a_n} \rhd\rhd \mathbf{q}_n, \qquad n \geq 0 \tag{11}$$

where the inputs $a_n \in A$ for $n \geq 0$ are drawn independently at random with distribution α.

Recall that $\mathbf{q}$ is called *recurrent* for the Markov chain if

$$\Pr(\mathbf{q}_n = \mathbf{q} \text{ for some } n \geq 1 \,|\, \mathbf{q}_0 = \mathbf{q}) = 1. \tag{12}$$

Next we relate this notion of recurrence to the algebraic notion. Let M_α be the submonoid of M generated by $\{\tau_a : \alpha(a) > 0\}$. We say that $\mathbf{q}$ is *accessible* from $\mathbf{q}'$ if $\mathbf{q} \in M_\alpha \mathbf{q}'$. By the elementary theory of Markov chains, (12) holds if and only if

$$\mathbf{q} \text{ is accessible from } m\mathbf{q} \text{ for all } m \in M_\alpha \tag{13}$$

(here we are using the assumption that Q is finite). Mutual accessibility is an equivalence relation on Q. Its equivalence classes are called *communicating classes*.

Write e_α for the minimal idempotent of M_α.

Lemma 3.4. $\mathbf{q}$ *is recurrent for the Markov chain* (11) *if and only if* $e_\alpha \mathbf{q} = \mathbf{q}$.

Proof. Given $m \in M_\alpha$, let g be the inverse of $e_\alpha m$ in the group $e_\alpha M_\alpha$. Since $g = ge_\alpha$ we have $gm = g(e_\alpha m) = e_\alpha$, so for any $\mathbf{q} \in Q$

$$g(m\mathbf{q}) = e_\alpha \mathbf{q}.$$

Therefore if $e_\alpha \mathbf{q} = \mathbf{q}$ then (13) holds. Conversely, if (13) holds then in particular $\mathbf{q}$ is accessible from $e_\alpha \mathbf{q}$, so $\mathbf{q} \in e_\alpha Q$ and hence $e_\alpha \mathbf{q} = \mathbf{q}$. $\qquad\square$

If the support of α is too small, it may happen that $e_\alpha \neq e$, or it may happen that $e_\alpha = e$ but eM_α is a proper subgroup of eM. We say that α has *adequate support* if

$$e_\alpha M_\alpha = eM.$$

Note that this holds trivially if $\alpha(a) > 0$ for all $a \in A$ (in which case $M_\alpha = M$).

Lemma 3.5. *If α has adequate support, then $\operatorname{Rec} \mathcal{N}$ is the unique communicating class of recurrent states for the Markov chain* (11).

Proof. If $e_\alpha M_\alpha \subseteq eM$ then $e_\alpha = e$. By Lemma 3.4 the set of recurrent states for the Markov chain is $e_\alpha Q = eQ = \operatorname{Rec} \mathcal{N}$.

By Theorem 3.3, the group eM acts transitively on $\operatorname{Rec} \mathcal{N}$, so if $eM \subseteq e_\alpha M_\alpha$ then any two states of $\operatorname{Rec} \mathcal{N}$ are mutually accessible. $\qquad\square$

The next two theorems generalize results of [12], where they are proved for sandpile networks.

Theorem 3.6. *For any α, the uniform distribution on $\operatorname{Rec} \mathcal{N}$ is stationary for the Markov chain* (11). *If α has adequate support, then the stationary distribution is unique.*

Proof. Fix $\mathbf{q}_0 \in \operatorname{Rec} \mathcal{N}$, and let g be uniform random element of $\operatorname{Crit} \mathcal{N}$. By Theorem 3.3, $g \rhd\rhd \mathbf{q}_0$ is a uniform random element of $\operatorname{Rec} \mathcal{N}$. If $a \in A$ is independent of g, then $\tau_a g$ is also a uniform random element of $\operatorname{Crit} \mathcal{N}$; hence if $\mathbf{q}_n$ is uniform on $\operatorname{Rec} \mathcal{N}$, then $\mathbf{q}_{n+1}$ is again uniform on $\operatorname{Rec} \mathcal{N}$.

If α has adequate support, then $\operatorname{Rec} \mathcal{N}$ is the unique recurrent communicating class by Lemma 3.5, so the stationary distribution is unique. $\qquad\square$

3.3. **Expected time to halt.** Recall our assumptions that $\mathcal{N}$ is finite and irreducible and halts on all inputs. In particular, the production matrix P is well defined and has spectral radius < 1 by Theorem 2.9, so $I - P$ is invertible where I is the $A \times A$ identity matrix. Our next result gives an interpretation for the entry $(I - P)_{ab}^{-1}$: it is the expected number of letters a processed before the network halts, when one letter b is input to a uniform recurrent state.

Theorem 3.7. (Expected Time To Halt) *Let $\mathbf{q}$ be a uniform random element of $\operatorname{Rec} \mathcal{N}$. Then for all $\mathbf{x} \in \mathbb{N}^A$ we have*

$$\mathbb{E}[\mathbf{x}.\mathbf{q}] = (I - P)^{-1}\mathbf{x}.$$

Later, in Theorem 4.2, we will bound the difference $[\mathbf{x}.\mathbf{q}] - \mathbb{E}[\mathbf{x}.\mathbf{q}]$. To build up to the proofs of these results, consider the group homomorphism

$$\phi : \mathbb{Z}^A \to \operatorname{Crit} \mathcal{N}$$

defined on generators by $\mathbf{1}_a \mapsto e\tau_a$ for each $a \in A$. For $x \in \mathbb{N}^A$ and $\mathbf{q} \in Q$ we have

$$\phi(\mathbf{x})\mathbf{q} = \left(\prod_{a \in A} (e\tau_a)^{\mathbf{x}_a} \right) \mathbf{q} = \left(\prod_{a \in A} \tau_a^{\mathbf{x}_a} \right) e\mathbf{q} = \mathbf{x} \triangleright\triangleright e\mathbf{q}. \tag{14}$$

First we observe that input vectors in the kernel of ϕ act trivially on recurrent states.

Lemma 3.8. *If* $\mathbf{x} \in (\ker \phi) \cap \mathbb{N}^A$ *and* $\mathbf{q} \in \operatorname{Rec} \mathcal{N}$, *then* $\mathbf{x} \triangleright\triangleright \mathbf{q} = \mathbf{q}$.

Proof. For $\mathbf{q} \in \operatorname{Rec} \mathcal{N}$ we have $\mathbf{q} = e\mathbf{q}$. Now by (14), since $\phi(\mathbf{x}) = e$,

$$\mathbf{x} \triangleright\triangleright \mathbf{q} = \mathbf{x} \triangleright\triangleright e\mathbf{q} = \phi(\mathbf{x})\mathbf{q} = e\mathbf{q} = \mathbf{q}. \qquad \square$$

Write e_v for the minimal idempotent of the local monoid M_v (§2.8).

Lemma 3.9. (Recurrent Implies Locally Recurrent) *If* $e\mathbf{q} = \mathbf{q}$ *then* $e_v\mathbf{q}_v = \mathbf{q}_v$ *for all* $v \in V$.

Proof. By Lemma 2.5 there exists $\mathbf{z} \geq \mathbf{1}$ be such that $\tau(\mathbf{z}) = e$. Let $\mathbf{u} = [\mathbf{z}.\mathbf{q}] \geq \mathbf{1}$. For each $v \in V$ let $\mathbf{y}_v \in \mathbb{N}^{A_v}$ be such that $t_v(\mathbf{y}_v) = e_v$. Then for all sufficiently large n we have $n\mathbf{u}_v \geq \mathbf{y}_v$ and hence $t_v(n\mathbf{u}_v) \in e_v M_v$. Since $e_v M_v$ is a finite group, there exists $n_v \in \mathbb{N}$ such that $t_v(n_v\mathbf{u}_v) = e_v$.

Now if $e\mathbf{q} = \mathbf{q}$ then for all $n \in \mathbb{N}$ we have $(n\mathbf{z}) \triangleright\triangleright \mathbf{q} = \mathbf{q}$ and $[n\mathbf{z}.\mathbf{q}] = n\mathbf{u}$ by Lemma 2.2, and hence $t(n\mathbf{u})\mathbf{q} = \mathbf{q}$ by (5). Taking $n = n_v$ we obtain

$$e_v\mathbf{q}_v = t_v(n_v\mathbf{u}_v)\mathbf{q}_v = \mathbf{q}_v. \qquad \square$$

The converse of Lemma 3.9 is usually false: For example, all states of $\texttt{Rotor}(G, s)$ and $\texttt{Sand}(G, s)$ are locally recurrent, but states of the former containing an oriented cycle of rotors are not recurrent, nor are states of the latter containing "forbidden" subconfigurations (the simplest of which is a pair of adjacent vertices both in state 0). Theorem 3.20 gives conditions when the converse does hold.

Now we come to the main ingredient in the proof of Theorem 3.7: if the network starts in a locally recurrent state $\mathbf{q}$ and halts in the same state $\mathbf{q}$, then the odometer $[\mathbf{x}.\mathbf{q}]$ belongs to the total kernel K.

Lemma 3.10. *Fix* $\mathbf{x} \in \mathbb{N}^A$ *and a locally recurrent state* $\mathbf{q} \in Q$. *Let* $\mathbf{k} = [\mathbf{x}.\mathbf{q}]$. *If* $\mathbf{x} \triangleright\triangleright \mathbf{q} = \mathbf{q}$, *then* $\mathbf{k} \in K$ *and* $\mathbf{x} = (I - P)\mathbf{k}$.

Proof. If $\mathbf{x} \triangleright\triangleright \mathbf{q} = \mathbf{q}$, then $t(\mathbf{k})\mathbf{q} = \mathbf{q}$ by equation (5). By Lemma 2.7 it follows that $\mathbf{k} \in K$. Now by Lemma 2.4(ii) and the definition (8) of the production matrix,

$$\mathbf{k}.\mathbf{q} = \mathbf{k} \triangleright (\mathbf{x}.\mathbf{q}) = (P(\mathbf{k}) + \mathbf{x}).\mathbf{q}$$

Hence $\mathbf{k} = P(\mathbf{k}) + \mathbf{x}$. $\qquad \square$

Proof of Theorem 3.7. Since $\operatorname{Crit} \mathcal{N}$ is a finite group, for any $\mathbf{x} \in \mathbb{Z}^A$ there is a positive integer n such that $n\mathbf{x} \in \ker \phi$. Fix $\mathbf{q} \in \operatorname{Rec} \mathcal{N}$ and let $\mathbf{k} = [n\mathbf{x}.\mathbf{q}]$. We have $n\mathbf{x} \triangleright\triangleright \mathbf{q} = \mathbf{q}$ by Lemma 3.8. Moreover $\mathbf{q}$ is locally recurrent by Lemma 3.9, so by Lemma 3.10 it follows that $n\mathbf{x} = (I - P)\mathbf{k}$. In particular, since $I - P$ is invertible, $\mathbf{k}$ does not depend on $\mathbf{q}$.

Now by Lemma 2.2,

$$\mathbf{k} = \sum_{j=0}^{n-1} [\mathbf{x}.(j\mathbf{x} \rhd\rhd \mathbf{q})].$$

By Theorem 3.6, if $\mathbf{q}$ is uniform on $\mathrm{Rec}\,\mathcal{N}$ then $j\mathbf{x} \rhd\rhd \mathbf{q}$ is uniform on $\mathrm{Rec}\,\mathcal{N}$ for all $j \in \mathbb{N}$. Taking expectations, we obtain

$$\mathbf{k} = \mathbb{E}\mathbf{k} = n\mathbb{E}[\mathbf{x}.\mathbf{q}].$$

Dividing by n yields the result. $\qquad\square$

3.4. **Generators and relations.** In this section we give generators and relations for the critical group. The group homomorphism $\phi : \mathbb{Z}^A \to \mathrm{Crit}\,\mathcal{N}$ sending $a \mapsto e\tau_a$ is surjective, since $\mathrm{Crit}\,\mathcal{N} = eM$ and M is generated by $\{\tau_a\}_{a\in A}$. To describe the kernel of ϕ we will use the *production map* from §2.11,

$$P : K \to \mathbb{Z}^A$$

where $K \subset \mathbb{Z}^A$ is the total kernel. Write I for the inclusion $K \hookrightarrow \mathbb{Z}^A$.

Theorem 3.11. *The natural map $\phi : \mathbb{Z}^A \to \mathrm{Crit}\,\mathcal{N}$ induces an isomorphism of abelian groups*

$$\mathrm{Crit}\,\mathcal{N} \simeq \mathbb{Z}^A/(I-P)K.$$

Proof. We must show that $\ker \phi = (I-P)K$. Fix $\mathbf{q} \in \mathrm{Rec}\,\mathcal{N}$. For any $\mathbf{k} \in K \cap \mathbb{N}^A$ we have

$$\mathbf{k} \rhd \mathbf{q} = P(\mathbf{k}).\mathbf{q}$$

so $\mathbf{k} \rhd\rhd \mathbf{q} = P(\mathbf{k}) \rhd\rhd \mathbf{q}$ by Lemma 2.3. Hence $\phi(\mathbf{k})\mathbf{q} = \phi(P(\mathbf{k}))\mathbf{q}$ by (14). By Theorem 3.3 the action of $\mathrm{Crit}\,\mathcal{N}$ on $\mathrm{Rec}\,\mathcal{N}$ is free, so we conclude $\phi(\mathbf{k}) = \phi(P(\mathbf{k}))$. This shows that $(I - P)\mathbf{k} \in \ker \phi$ for all $\mathbf{k} \in K \cap \mathbb{N}^A$. Since K is generated as a group by $K \cap \mathbb{N}^A$, it follows that $(I - P)K \subset \ker \phi$.

To show the reverse inclusion, given $\mathbf{x} \in (\ker \phi) \cap \mathbb{N}^A$ we have $\phi(\mathbf{x}) = e$ and hence $\mathbf{x} \rhd\rhd e\mathbf{q} = e\mathbf{q}$ for all $\mathbf{q} \in Q$ by (14). By Lemma 3.10, $\mathbf{x} = (I - P)\mathbf{k}$ where $\mathbf{k} = [\mathbf{x}.e\mathbf{q}] \in K$. It follows that $(\ker \phi) \cap \mathbb{N}^A \subset (I - P)K$. Since $\mathrm{Crit}\,\mathcal{N}$ is finite, $\ker \phi$ is a full rank subgroup of $\mathbb{Z}^A$, so it is generated as a group by $(\ker \phi) \cap \mathbb{N}^A$, which shows that $\ker \phi \subset (I - P)K$. $\qquad\square$

Irreducible abelian networks $\mathcal{N}$ and $\mathcal{N}'$ on the same graph with the same total alphabet are called *homotopic*, written $\mathcal{N} \approx \mathcal{N}'$, if they have the same total kernel K and the same production map P. An example of homotopic networks are the sandpile and simple rotor networks on a connected graph $G = (V, E)$. These have total kernel $K = \prod_{v\in V}(d_v\mathbb{Z})$ where d_v is the degree of vertex v: in the sandpile case processing d_v letters v causes v to topple exactly once, while in the rotor case it causes the rotor at v to serve each neighbor u of v exactly once. In both cases exactly one letter is sent to each neighbor, so the production map is given by $P(\mathbf{k})_u = \sum_{(v,u)\in E} \mathbf{k}_v/d_v$.

By Theorem 3.11 the critical group depends only on the total kernel K and production matrix P, so homotopic networks have isomorphic critical groups.

Corollary 3.12. *If $\mathcal{N} \approx \mathcal{N}'$, then $\mathrm{Crit}\,\mathcal{N} \simeq \mathrm{Crit}\,\mathcal{N}'$.*

Corollary 3.12 generalizes the isomorphism $\mathrm{Crit}\,\mathtt{Rotor}(G, s) \simeq \mathrm{Crit}\,\mathtt{Sand}(G, s)$ between the rotor and sandpile groups of a graph with sink vertex s.

Next we relate the critical group to the cokernel of the Laplacian (9).

Definition 3.13. An abelian network $\mathcal{N}$ is *rectangular* if its total kernel is $K = \prod_{a \in A}(r_a \mathbb{Z})$, where r_a are the reset numbers (10). (That is, K is a rectangular sublattice of $\mathbb{Z}^A$.)

Corollary 3.14. *The natural map $\mathbb{Z}^A \to \mathrm{Crit}\,\mathcal{N}$ induces a surjective group homomorphism*
$$\phi : \mathbb{Z}^A/L\mathbb{Z}^A \twoheadrightarrow \mathrm{Crit}\,\mathcal{N}.$$
If $\mathcal{N}$ is rectangular, then ϕ is an isomorphism.

Proof. By definition, $D\mathbb{Z}^A \subset K$ with equality if $\mathcal{N}$ is rectangular. Since $L = (I - P)D$, we have $L\mathbb{Z}^A \subset (I - P)K$ with equality if $\mathcal{N}$ is rectangular. $\square$

Note that any unary network (and in particular any toppling network) is rectangular. In [8] we defined the *sandpilization* $\mathcal{S}(\mathcal{N})$ as the locally recurrent toppling network with the same Laplacian as $\mathcal{N}$. To spell this out, let us first describe the underlying graph of $\mathcal{S}(\mathcal{N})$, which has vertex set A rather than V.

Definition 3.15. The *production graph* $\Gamma(\mathcal{N})$ is the directed graph with vertex set A and edge set $\{(a, b) : P_{ba} > 0\}$.

To form the sandpilization, we split each processor $\mathcal{P}_v$ into $\#A_v$ unary processors $\mathcal{R}_a$ for $a \in A_v$. Each $\mathcal{R}_a$ has state space $\{0, 1, \ldots, r_a - 1\}$ where r_a is the reset number (10). When $\mathcal{R}_a$ transitions from state $r_a - 1$ to state 0, it sends $r_a P_{ba}$ letters b to each $\mathcal{R}_b$. Note that $r_a P_{ba} = P(r_a 1_a)_b$ is an integer since $r_a 1_a \in K$.

Definition 3.16. The *sandpilization* of $\mathcal{N}$ is the unary network
$$\mathcal{S}(\mathcal{N}) = (\mathcal{R}_a)_{a \in A}$$
with underlying graph $\Gamma(\mathcal{N})$.

Since $\mathcal{N}$ and $\mathcal{S}(\mathcal{N})$ have the same Laplacian, the following is immediate from Corollary 3.14.

Corollary 3.17. $\mathrm{Crit}\,\mathcal{S}(\mathcal{N}) \simeq \mathbb{Z}^A/L\mathbb{Z}^A$ *and* $\mathrm{Crit}\,\mathcal{S}(\mathcal{N}) \twoheadrightarrow \mathrm{Crit}\,\mathcal{N}$.

Example. To see that the map $\mathrm{Crit}\,\mathcal{S}(\mathcal{N}) \to \mathrm{Crit}\,\mathcal{N}$ need not be an isomorphism, consider the following non-rectangular network with vertices i, j where j is a sink. Let $Q_i = \{0, 1\}$, $A_i = \{a, b\}$, $A_j = \{c\}$,
$$T_i(q, a) = T_i(q, b) = q + 1 \ (\mathrm{mod}\ 2)$$
and
$$\begin{aligned}
T_{(i,j)}(0, a) &= c \\
T_{(i,j)}(1, a) &= cc \\
T_{(i,j)}(0, b) &= \epsilon \\
T_{(i,j)}(1, b) &= c.
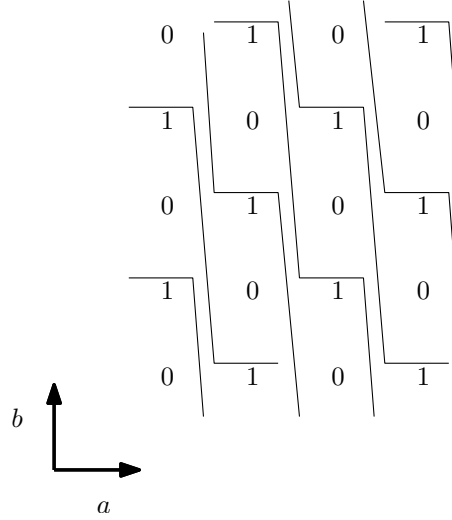\end{aligned}$$
Figure 1 shows the state diagram of $\mathcal{P}_i$.

FIGURE 1. State diagram of an abelian processor with two states $0, 1$ and two inputs a, b. Its kernel is $\{(m, n) \in \mathbb{Z}^2 : m + n \equiv 0 \pmod{2}\}$, so it is not rectangular. Each line crossed results in output of one letter c.

The production matrix and Laplacian of this network, with rows and columns indexed by a, b, c in that order, are:

$$P = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ \frac{3}{2} & \frac{1}{2} & 0 \end{pmatrix} \qquad L = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ -3 & -1 & 1 \end{pmatrix}$$

We have $\operatorname{Crit} \mathcal{S}(\mathcal{N}) = \mathbb{Z}^3 / L\mathbb{Z}^3 = (\mathbb{Z}/2\mathbb{Z})^2$. On the other hand, $\tau_a = \tau_b$ in $\operatorname{Crit} \mathcal{N}$ (both send $0 \mapsto 1 \mapsto 0$) so $\operatorname{Crit} \mathcal{N} = \mathbb{Z}/2\mathbb{Z}$.

Processor $\mathcal{P}_i$ in this example has another curious feature: The input words aa, ab, bb to state 0 all result in the same sequence of states $0, 1, 0$ yet they produce different outputs (ccc, cc, c respectively). $\square$

3.5. **Order of the critical group.** Now we turn to the problem of counting recurrent states, or equivalently (by Theorem 3.3) finding the order of the critical group. Let

$$\iota = [K : D\mathbb{Z}^A]$$

be the index of $D\mathbb{Z}^A$ as a subgroup of K. Recalling that $K = \prod_{v \in V} K_v$, we can write $\iota = \prod_{v \in V} \iota_v$ as a product of local indices

$$\iota_v = [K_v : D_v \mathbb{Z}^{A_v}]$$

where $D_v \mathbb{Z}^{A_v} := \prod_{a \in A_v} (r_a \mathbb{Z})$. Note $\iota = 1$ if and only if $\mathcal{N}$ is rectangular.

Theorem 3.18.

$$\#\operatorname{Rec} \mathcal{N} = \#\operatorname{Crit} \mathcal{N} = \frac{\det L}{\iota}.$$

Proof. The first equality follows from Theorem 3.3. For the second, we have by Theorem 3.11

$$\# \mathrm{Crit}\, \mathcal{N} = [\mathbb{Z}^A : (I-P)K] = \frac{[\mathbb{Z}^A : L\mathbb{Z}^A]}{[(I-P)K : L\mathbb{Z}^A]}$$

$$= \frac{|\det L|}{[(I-P)K : (I-P)D\mathbb{Z}^A]}.$$

By Theorem 2.9, since $\mathcal{N}$ halts on all inputs, $I-P$ has full rank and $\det L > 0$, so the right side equals $\frac{\det L}{[K:D\mathbb{Z}^A]}$. $\qquad\square$

Our last goal in this section is to address the converse of Lemma 3.9: When does locally recurrent imply recurrent? We will find several equivalent conditions.

Lemma 3.19. *Let $\mathcal{S}$ be a toppling network with production graph Γ. For each directed cycle $a_1 \to \ldots \to a_m \to a_1$ of Γ we have*

$$\max\{\mathbf{q}(a_1), \ldots, \mathbf{q}(a_m)\} \geq 1$$

for all $\mathbf{q} \in \mathrm{Rec}\, \mathcal{S}$.

Proof. Let $\mathbf{r} = (r_a)_{a \in A}$ be the vector of toppling thresholds. By Lemma 2.5 there exists $\mathbf{z} \geq \mathbf{r}$ such that $\tau(\mathbf{z}) = e$. If $\mathbf{q} \in \mathrm{Rec}\, \mathcal{S}$ then $\mathbf{z} \triangleright\triangleright \mathbf{q} = \mathbf{q}$ and each vertex topples at least once during the stabilization of $\mathbf{z}.\mathbf{q}$. If a_i is the last vertex on the cycle to finish toppling, then a_{i+1} receives a chip from a_i and does not topple thereafter, so $\mathbf{q}(a_{i+1}) \geq 1$. $\qquad\square$

Theorem 3.20. *Let $\mathcal{N}$ be a finite irreducible abelian network that halts on all inputs, and let $\mathcal{S}(\mathcal{N})$ be its sandpilization. The following are equivalent.*

   (1) *Every locally recurrent state of $\mathcal{N}$ is recurrent.*
   (2) $\det L = \det D$.
   (3) *Every state of $\mathcal{S}(\mathcal{N})$ is recurrent.*
   (4) *The state $\mathbf{0}$ of $\mathcal{S}(\mathcal{N})$ is recurrent.*
   (5) *The production graph Γ has no directed cycles.*
   (6) *The producrtion matrix P is nilpotent.*

Proof. Recall from §2.11 the action of $\mathbb{Z}^{A_v}$ on $e_v Q_v$ for each vertex v. Since $\mathcal{N}$ is irreducible this action is transitive, and its kernel is K_v. So the number of locally recurrent states of $\mathcal{N}$ is

$$\prod_{v \in V} \#(e_v Q_v) = \prod_{v \in V} \#(\mathbb{Z}^{A_v}/K_v) = [\mathbb{Z}^A : K].$$

By Theorem 3.18, the number of recurrent states of $\mathcal{N}$ is

$$\frac{\det L}{[K : D\mathbb{Z}^A]} = \frac{\det L}{\det D}[\mathbb{Z}^A : K].$$

Now from Lemma 3.9 it follows that (1) $\Leftrightarrow$ (2).

All states of the sandpilization $\mathcal{S}(\mathcal{N})$ are locally recurrent, and $\mathcal{S}(\mathcal{N})$ has the same matrices L and D as $\mathcal{N}$, so (2) $\Rightarrow$ (3).

Trivially (3) $\Rightarrow$ (4).

If Γ has a directed cycle, then the state $\mathbf{0}$ of $\mathcal{S}(\mathcal{N})$ is not recurrent by Lemma 3.19, which shows $(4) \Rightarrow (5)$.

If Γ has no directed cycles, then each entry of P^k is a weighted sum over directed paths in Γ with $k+1$ distinct vertices. Taking $k = \#A$ we obtain $P^k = \mathbf{0}$, which shows $(5) \Rightarrow (6)$.

If $\lambda_1, \ldots, \lambda_n$ are the eigenvalues of P with multiplicity, then

$$\frac{\det L}{\det D} = \det(I - P) = \prod_{i=1}^{n}(1 - \lambda_i).$$

If P is nilpotent then $\lambda_1 = \cdots = \lambda_n = 0$, so $\det L = \det D$. Hence $(6) \Rightarrow (2)$, completing the proof. $\square$

4. Time to halt

In this section we show that the production matrix of $\mathcal{N}$ determines its running time on any input up to an additive constant (Theorem 4.2). In the special case of rotor networks, additive error bounds of this type appear in the work of Cooper and Spencer [10] and Holroyd and Propp [24]. For an upper bound in the case of sandpiles, see [23, Prop. 4.8]. A related but distinct question is: given $\mathbf{x}$ and $\mathbf{q}$, how quickly can one compute the final state $\mathbf{x} \rhd\rhd \mathbf{q}$ and the odometer $[\mathbf{x}.\mathbf{q}]$? According to Theorem 4.2 the abelian network $\mathcal{N}$ performs this computation (asynchronously) in time approximately $\mathbf{1}^T(I - P)^{-1}\mathbf{x}$, but in some cases [5, 17] one can design a (sequential) algorithm that is much faster.

As usual we take $\mathcal{N}$ to be a finite irreducible abelian network that halts on all inputs. Let K be its total kernel and $P : K \to \mathbb{Z}^A$ its production map. Write I for the inclusion $K \hookrightarrow \mathbb{Z}^A$.

Lemma 4.1. *Suppose $\mathbf{k} \in K$ satisfies $P\mathbf{k} \leq \mathbf{k}$. If $\mathbf{q} \in \operatorname{Rec}\mathcal{N}$, then*

$$(I - P)\mathbf{k} \rhd\rhd \mathbf{q} = \mathbf{q}.$$

Proof. Let $\mathbf{x} = (I - P)\mathbf{k} \in \mathbb{N}^A$. By Theorem 3.11, if $\mathbf{k} \in K$ then $\mathbf{x} \in \ker \phi$. The conclusion now follows from Lemma 3.8. $\square$

Remark. Recalling that $I - P$ is injective (Theorem 2.9) and that the matrix of $(I-P)^{-1}$ has nonnegative entries (Theorem 3.7), we see that the condition $P\mathbf{k} \leq \mathbf{k}$ in the above lemma implies $\mathbf{k} \geq \mathbf{0}$.

For $\mathbf{u} \in \mathbb{Z}^A$ write $\|\mathbf{u}\|_\infty = \max_{a \in A} |\mathbf{u}_a|$.

Theorem 4.2. (Time To Halt) *Let $\mathcal{N}$ be an irreducible finite abelian network that halts on all inputs. There is a constant C depending only on $\mathcal{N}$, such that for all $\mathbf{x} \in \mathbb{N}^A$ and all $\mathbf{q} \in Q$,*

$$\left\| [\mathbf{x}.\mathbf{q}] - (I - P)^{-1}\mathbf{x} \right\|_\infty \leq C.$$

Proof. Since $(I - P)K$ has full rank in $\mathbb{Z}^A$, there is a constant c such that for any $\mathbf{x} \in \mathbb{N}^A$ there exists $\mathbf{y} = (I - P)\mathbf{k} \in (I - P)K$ such that $\mathbf{0} \leq \mathbf{y} - \mathbf{x} \leq c\mathbf{1}$. Since $(I - P)^{-1}$ has nonnegative entries,

$$\mathbf{0} \leq \mathbf{k} - (I - P)^{-1}\mathbf{x} \leq (I - P)^{-1}c\mathbf{1}.$$

Therefore it suffices to bound $\|[\mathbf{x}.\mathbf{q}] - \mathbf{k}\|_\infty$.

To do this, fix $\mathbf{z} \in \mathbb{N}^A$ with $\tau(\mathbf{z}) = e$. We will show

$$\|[\mathbf{x}.e\mathbf{q}] - \mathbf{k}\|_\infty \leq C_1 \tag{15}$$

and

$$\|[\mathbf{x}.e\mathbf{q}] - [\mathbf{x}.\mathbf{q}]\|_\infty \leq C_2 \tag{16}$$

where $C_1 = \max_{\mathbf{q}' \in Q} \|[c\mathbf{1}.\mathbf{q}']\|_\infty$ and $C_2 = \max_{\mathbf{q}' \in Q} \|[\mathbf{z}.\mathbf{q}']\|_\infty$.

Since $\mathbf{y} \geq \mathbf{0}$, we have $P\mathbf{k} \leq \mathbf{k}$, so $\mathbf{y} \rhd\rhd e\mathbf{q} = e\mathbf{q}$ by Lemma 4.1. Hence by Lemmas 3.10 and 2.2

$$\mathbf{k} = [\mathbf{y}.e\mathbf{q}] = [\mathbf{x}.e\mathbf{q}] + [(\mathbf{y} - \mathbf{x}).(\mathbf{x} \rhd\rhd e\mathbf{q})]$$

which shows (15). Two more applications of Lemma 2.2 give

$$[\mathbf{z}.\mathbf{q}] + [\mathbf{x}.e\mathbf{q}] = [(\mathbf{x} + \mathbf{z}).\mathbf{q}] = [\mathbf{x}.\mathbf{q}] + [\mathbf{z}.(\mathbf{x} \rhd\rhd \mathbf{q})]$$

which shows (16).   □

By Theorem 4.2 and the triangle inequality,

$$\|[\mathbf{x}.\mathbf{q}] - [\mathbf{x}.\mathbf{r}]\|_\infty \leq 2C$$

for all $\mathbf{x} \in \mathbb{N}^A$ and all $\mathbf{q}, \mathbf{r} \in Q$. We will use this bound in the next section.

5. Burning test

Dhar's burning test [12] is an efficient algorithm for determining whether a state of $\mathtt{Sand}(G, s)$ recurrent. In its simplest form it applies to Eulerian directed graphs G (strongly connected, $\mathrm{indegree}(v) = \mathrm{outdegree}(v)$ for $v \in V$); see [23, §4]. Speer [34] treated general directed graphs. A variant of Speer's algorithm can be found in [30]. The goal of this section is to generalize these algorithms to the setting of an arbitrary finite irreducible abelian network $\mathcal{N}$ that halts on all inputs.

Definition 5.1. A *burning element* for $\mathcal{N}$ is a vector $\beta \in \mathbb{N}^A$ such that for $\mathbf{q} \in Q$

$$\mathbf{q} \in \mathrm{Rec}\,\mathcal{N} \quad \Leftrightarrow \quad \beta \rhd\rhd \mathbf{q} = \mathbf{q}.$$

To see that burning elements exist, we can use Lemma 1.2(5): $\mathbf{q} \in \mathrm{Rec}\,\mathcal{N}$ if and only if $\mathbf{q} = e\mathbf{q}$, where e is the minimal idempotent of the transition monoid M. Recalling the map τ of (7), any $\beta \in \mathbb{N}^A$ such that $\tau(\beta) = e$ is a burning element. However, such elements are typically large. The power of the burning test derives from the fact that one can often identify a small burning element β to reduce the running time $[\beta.\mathbf{q}]$.

In the classical setting of $\mathcal{N} = \mathtt{Sand}(G, s)$, where G is a directed graph with globally accessible sink s, there is a pointwise minimal burning element β. In the case that G is Eulerian, the usual formula for the burning element is $\beta_v = d_{vs}$, the number of edges into v from s. The special role of the sink is undesirable for us (since in general, an abelian network need not have a sink in order to halt on all inputs). To remove it, notice that

$$\beta = L\mathbf{1}$$

where L is the Laplacian of (9). (To make the connection to the graph Laplacian $\Delta_G = D_G - A_G$, where D_G is the diagonal matrix of outdegrees, and A_G is the adjacency matrix of G, we have $L = \Delta_{G'}$ where G' is the graph obtained from G by removing all outgoing edges from s. Since G is Eulerian we have $\Delta_G \mathbf{1} = \mathbf{0}$, so $L\mathbf{1} = -\Delta_G \mathbf{1}_s = \beta$.)

When G is not Eulerian, $L\mathbf{1}$ may have negative entries. Speer [34] observed that there is a pointwise smallest vector $\mathbf{y} \geq \mathbf{1}$ such that $L\mathbf{y} \geq \mathbf{0}$, and showed that $\beta = L\mathbf{y}$ is a burning element. Our burning test for abelian networks, Theorem 5.5, reduces to Speer's in the case $\mathcal{N} = \mathtt{Sand}(G, s)$.

5.1. **Large inputs.** To lay the ground for the burning test, we show in this section that states obtained from sufficiently large inputs to $\mathcal{N}$ must be recurrent. We consider two interpretations of "large": either the number of letters in the input $\mathbf{x}$ is large (Lemma 5.2), or the odometer $[\mathbf{x}.\mathbf{q}]$ is large (Lemmas 5.3 and 5.4).

Lemma 5.2. *Let $\mathbf{z} \in \mathbb{N}^A$ be such that $\tau(\mathbf{z}) = e$. If $\mathbf{x} \geq \mathbf{z}$ then $\mathbf{x} \rhd\!\rhd \mathbf{q}$ is recurrent for all $\mathbf{q} \in Q$.*

Proof. Writing $\mathbf{x} = \mathbf{y} + \mathbf{z}$ for $\mathbf{y} \in \mathbb{N}^A$, we have

$$\mathbf{x} \rhd\!\rhd \mathbf{q} = \mathbf{z} \rhd\!\rhd (\mathbf{y} \rhd\!\rhd \mathbf{q}) = em\mathbf{q}.$$

where $m = \tau(\mathbf{y})$. By Lemma 1.2(4) the state $em\mathbf{q}$ is recurrent. $\qquad\square$

Given $\mathbf{q}, \mathbf{r} \in Q$ we say that $\mathbf{q}$ is *locally accessible* from $\mathbf{r}$ if $\mathbf{q} = \mathbf{mr}$ for some $\mathbf{m} \in \prod_{v \in V} M_v$. In particular, if there is an execution (say w) from $\mathbf{x}.\mathbf{r}$ to $\mathbf{y}.\mathbf{q}$, then $\mathbf{q}$ is locally accessible from $\mathbf{r}$ (namely $\mathbf{q} = t(|w|)\mathbf{r}$).

Lemma 5.3. *If there exists $\mathbf{x} \in \mathbb{N}^A$ such that $\mathbf{x} \rhd\!\rhd \mathbf{r} = \mathbf{r}$ and $[\mathbf{x}.\mathbf{q}] \geq \mathbf{1}$ for all states $\mathbf{q}$ locally accessible from $\mathbf{r}$, then $\mathbf{r}$ is recurrent.*

Proof. Let $\mathbf{z} \in \mathbb{N}^A$ be such that $\tau(\mathbf{z}) = e$. We will find a $\mathbf{y} \geq \mathbf{z}$ and a state $\mathbf{q}$ such that $\mathbf{r} = \mathbf{y} \rhd\!\rhd \mathbf{q}$.

Let $n = \sum_{a \in A} \mathbf{z}_a$. Fix any sequence $a_1, \ldots, a_n$ such that $\mathbf{z} = \sum_{i=1}^n \mathbf{1}_{a_i}$. Let $\mathbf{q}^0 = \mathbf{r}$ and inductively define states $\mathbf{q}^1, \ldots, \mathbf{q}^n$, all locally accessible from $\mathbf{r}$, as follows.

For each $i = 1, \ldots, n$, since $\mathbf{q}^{i-1}$ is locally accessible from $\mathbf{r}$ we have $[\mathbf{x}.\mathbf{q}^{i-1}]_{a_i} \geq 1$, so there is a legal execution w_i from $\mathbf{x}.\mathbf{q}^{i-1}$ to some state $\mathbf{y}^i.\mathbf{q}^i$ satisfying $\mathbf{y}^i_{a_i} \geq 1$. The concatenation $w_1 \cdots w_n$ is a legal execution from $(n\mathbf{x}).\mathbf{r}$ to $\mathbf{y}.\mathbf{q}^n$ with $\mathbf{y} = \mathbf{y}^1 + \cdots + \mathbf{y}^n \geq \mathbf{z}$. Hence by Lemma 2.1,

$$\mathbf{r} = (n\mathbf{x}) \rhd\!\rhd \mathbf{r} = \mathbf{y} \rhd\!\rhd \mathbf{q}^n$$

and the right side is recurrent by Lemma 5.2. $\qquad\square$

For $\mathbf{u} \in \mathbb{N}^A$ and $\mathbf{q} \in Q$, define

$$R_{\mathbf{u}}(\mathbf{q}) = \left\{ \mathbf{x} \rhd\!\rhd \mathbf{q} \mid \mathbf{x} \in \mathbb{N}^A, [\mathbf{x}.\mathbf{q}] \geq \mathbf{u} \right\}.$$

and write $R_{\mathbf{u}}$ for the set of states $\mathbf{q} \in Q$ such that $\mathbf{q} \in R_{\mathbf{u}}(\mathbf{q})$.

We have defined recurrent states in Lemma 1.2 by a list of equivalent monoid-theoretic properties. Now we can add to this list a characterization that is specific

to abelian networks. According to the next lemma, a state $\mathbf{q}$ is recurrent if and only if it there exist inputs with arbitrarily large odometers that fix $\mathbf{q}$.

Lemma 5.4.
$$\operatorname{Rec} \mathcal{N} = \bigcap_{\mathbf{u} \in \mathbb{N}^A} R_{\mathbf{u}}.$$

Proof. Suppose that $\mathbf{r} \in \bigcap R_{\mathbf{u}}$. Then for any $\mathbf{u} \in \mathbb{N}^A$ there is an input $\mathbf{x} \in \mathbb{N}^A$ with $\mathbf{r} = \mathbf{x} \triangleright\triangleright \mathbf{r}$ and $[\mathbf{x}.\mathbf{r}] \geq \mathbf{u}$. Take $\mathbf{u} = (2C+1)\mathbf{1}$. By Theorem 4.2 it follows that $[\mathbf{x}.\mathbf{q}] \geq \mathbf{1}$ for all $\mathbf{q} \in Q$. Now from Lemma 5.3 it follows that $\mathbf{r}$ is recurrent.

It remains to show that $\operatorname{Rec} \mathcal{N} \subset R_{\mathbf{u}}$ for all $\mathbf{u} \in \mathbb{N}^A$. Since $\operatorname{Crit} \mathcal{N}$ is a finite group, the kernel of $\phi : \mathbb{Z}^A \to \operatorname{Crit} \mathcal{N}$ has nonempty intersection with $\mathbb{N}^A + \mathbf{u}$. Let $\mathbf{x}$ be a point in this intersection. Then for any $\mathbf{q} \in \operatorname{Rec} \mathcal{N}$ we have $\mathbf{x} \triangleright\triangleright \mathbf{q} = \mathbf{q}$ by Lemma 3.8. Since $[\mathbf{x}.\mathbf{q}] \geq \mathbf{x} \geq \mathbf{u}$, it follows that $\mathbf{q} \in R_{\mathbf{u}}$. $\qquad\square$

Finally, we show that if $\mathbf{k} \geq \mathbf{1}$ then the converse to Lemma 4.1 holds.

Theorem 5.5. (Burning Test) *Let* $\mathbf{k} \in K$ *be such that* $\mathbf{k} \geq \mathbf{1}$ *and* $P\mathbf{k} \leq \mathbf{k}$. *Then* $\mathbf{q} \in Q$ *is recurrent if and only if* $(I - P)\mathbf{k} \triangleright\triangleright \mathbf{q} = \mathbf{q}$.

Proof. If $\mathbf{q}$ is recurrent, then $(I - P)\mathbf{k} \triangleright\triangleright \mathbf{q} = \mathbf{q}$ by Lemma 4.1.

For the converse, let $\mathbf{x} = (I - P)\mathbf{k}$ and $\mathbf{u} = [\mathbf{x}.\mathbf{q}]$. Suppose that $\mathbf{x} \triangleright\triangleright \mathbf{q} = \mathbf{q}$. Then $\mathbf{x} = (I - P)\mathbf{u}$ by Lemma 3.10. Since $I - P$ is injective, we obtain $\mathbf{u} = \mathbf{k}$. Now for any $n \in \mathbb{N}$ we have $n\mathbf{x} \triangleright\triangleright \mathbf{q} = \mathbf{q}$ and $[n\mathbf{x}.\mathbf{q}] = n\mathbf{k}$. Hence

$$\mathbf{q} \in \bigcap_{n \geq 1} R_{n\mathbf{k}}.$$

Since $\mathbf{k} \geq \mathbf{1}$ the right side equals $\bigcap_{\mathbf{u} \in \mathbb{N}^A} R_{\mathbf{u}}$, which equals $\operatorname{Rec} \mathcal{N}$ by Lemma 5.4. $\quad\square$

Remark. The preceding theorem can be improved slightly by weakening $\mathbf{k} \geq \mathbf{1}$ to $\mathbf{k} \geq \mathbf{1}_C$, where C is the set of all $a \in A$ that lie on a directed cycle of the production graph Γ (Definition 3.15). The reason is that if $a \in A$ does not lie on a directed cycle, then $\{a\}$ is a strong component of Γ, and all locally recurrent states of the strong component $\mathcal{N}^a$ are recurrent. Writing a^ω for the minimal idempotent of the transition monoid of $\mathcal{N}^a$, one checks that $e = \prod_{a \in C} a^\omega$ and hence that $\operatorname{Rec} \mathcal{N} = \bigcap_{a \in C} \operatorname{Rec} \mathcal{N}^a$. See [9] where this improvement is carried out in detail for sandpile networks.

5.2. **Finding a burning element.** In this section we show that there is always a burning element β satisfying $\mathbf{0} \leq \beta \leq \mathbf{r}$, where $\mathbf{r}_a$ is the smallest positive integer such that $\mathbf{r}_a \mathbf{1}_a \in K$.

Definition 5.6. A *burning odometer* is a vector $\mathbf{k}$ satisfying

$$(I - P)\mathbf{k} \geq \mathbf{0}, \qquad \mathbf{k} \geq \mathbf{1}, \qquad \mathbf{k} \in K. \tag{17}$$

The corresponding *burning element* is $\beta = (I - P)\mathbf{k}$.

According to Theorem 5.5, to check whether $\mathbf{q}$ is recurrent it suffices to find a burning odometer $\mathbf{k}$ and then compute $\beta \triangleright\triangleright \mathbf{q}$, where $\beta = (I - P)\mathbf{k}$. The proof also shows that if $\mathbf{q}$ is recurrent, then the local run time of this computation is

$[\beta.\mathbf{q}] = \mathbf{k}$ (that is, for each $a \in A$ it requires processing $\mathbf{k}_a$ letters a). Therefore we are interested in finding a burning odometer $\mathbf{k}$ as small as possible.

Recall that $K \subset D\mathbb{Z}^A$, with equality if $\mathcal{N}$ is rectangular. For a sandpile network $\mathcal{N} = \texttt{Sand}(G, s)$, a burning odometer is $\mathbf{k} = D\mathbf{y}$ where $\mathbf{y}$ is the "burning script" of [34]. More generally, if $\mathcal{N}$ is rectangular, then writing $\mathbf{k} = D\mathbf{y}$, (17) is equivalent to

$$Ly \geq \mathbf{0}, \qquad \mathbf{y} \geq \mathbf{1}, \qquad \mathbf{y} \in \mathbb{Z}^A \tag{18}$$

where $L = (I - P)D$ is the Laplacian of $\mathcal{N}$. Setting $\mathbf{x} = \mathbf{y} - \mathbf{1}$ we find that (17) is equivalent to

$$L\mathbf{x} \geq -L\mathbf{1}, \qquad \mathbf{x} \geq \mathbf{0}, \qquad \mathbf{x} \in \mathbb{Z}^A. \tag{19}$$

Minimizing $\mathbf{1}^T\mathbf{x}$ subject to these constraints is an integer program of the class solved by toppling networks [7, Remark 4.9]. Specifically, consider the sandpilization $\mathcal{S}(\mathcal{N})$, enlarged to allow negative chip counts. In this network, vertex a has toppling threshold $\mathbf{r}_a$, where $\mathbf{r}$ is the vector of diagonal entries of the diagonal matrix D. By Theorem 2.9, if $\mathcal{N}$ halts on all inputs then $\mathcal{S}(\mathcal{N})$ halts on all inputs (since $\mathcal{S}(\mathcal{N})$ has the same Laplacian as $\mathcal{N}$). For a chip configuration $\mathbf{q} \in \mathbb{Z}^A$, write $\mathbf{q}^\circ$ for the stabilization of $\mathbf{q}$ in $\mathcal{S}(\mathcal{N})$.

Corollary 5.7. ($\mathcal{S}(\mathcal{N})$ computes a minimal burning element for $\mathcal{N}$) *Let $\mathcal{N}$ be an irreducible rectangular network that halts on all inputs. Then $\mathcal{N}$ has a pointwise smallest burning odometer $\mathbf{k} = D\mathbf{y}$. The corresponding burning element is given by*

$$\beta = L\mathbf{y} = \mathbf{r} - \mathbf{1} - (\mathbf{r} - \mathbf{1} - L\mathbf{1})^\circ.$$

Moreover, $\mathbf{0} \leq \beta \leq \mathbf{r}$.

Proof. Let $\mathbf{q} = \mathbf{r} - \mathbf{1} - L\mathbf{1}$, and for $a \in A$ let $\mathbf{x}_a$ be the number of times a topples during the stabilization of $\mathbf{q}$ in $\mathcal{S}(\mathcal{N})$. By [7, Remark 4.9], $\mathbf{x}$ is the pointwise smallest vector satisfying (19), and $\mathbf{q}^\circ = \mathbf{q} - L\mathbf{x}$. Setting $\mathbf{y} = \mathbf{x} + \mathbf{1}$, we conclude that $\mathbf{k} = D\mathbf{y}$ is the pointwise smallest vector satisfying (17), and

$$\beta = (I - P)\mathbf{k} = L(\mathbf{x} + \mathbf{1}) = L\mathbf{1} + \mathbf{q} - \mathbf{q}^\circ = \mathbf{r} - \mathbf{1} - \mathbf{q}^\circ.$$

Noting that $L\mathbf{1} \leq \mathbf{r}$, we have $-\mathbf{1} \leq \mathbf{q}^\circ \leq \mathbf{r} - \mathbf{1}$ and hence $\mathbf{0} \leq \beta \leq \mathbf{r}$. $\qquad\square$

Remark. If we replace $\mathbf{1}$ by $\mathbf{1}_C$ in (17), then we obtain a slightly smaller burning element β which satisfies $\mathbf{0} \leq \beta \leq \mathbf{r} - \mathbf{1}$. To compute this smaller element using $\mathcal{S}(\mathcal{N})$, take $\mathbf{q} = \mathbf{r} - \mathbf{1} - L\mathbf{1}_C$.

In practice, it is more direct to find the burning element by "untopplings" instead of topplings, which amounts to the following procedure to find the minimal $\mathbf{y}$ satisfying (18).

Procedure 5.8. *Start with $\mathbf{y} = \mathbf{1}$. If $L\mathbf{y} \geq \mathbf{0}$, then stop. Otherwise, choose some $a \in A$ such $(L\mathbf{y})_a < 0$ and increase $\mathbf{y}_a$ by 1. Repeat until $L\mathbf{y} \geq \mathbf{0}$.*

Remark. In the case that the row sums of L are nonnegative, the procedure halts immediately with $\mathbf{y} = \mathbf{1}$. In particular, this includes the special case $\mathcal{N} = \texttt{Sand}(G, s)$ for an Eulerian graph G.

In the case $\mathcal{N}$ is not rectangular, the inclusion $K \subset D\mathbb{Z}^A$ is strict. Corollary 5.7 and Procedure 5.8 will identify the minimal burning odometer $D\mathbf{y} \in D\mathbb{Z}^A$. Unlike the rectangular case, there may not be a unique minimal burning odometer in K.

If $\mathbf{k} = D\mathbf{y}$ is the minimal burning odometer in $D\mathbb{N}^A$, then the global burning test $(I - P)\mathbf{k} \rhd\rhd \mathbf{q}$ runs in time $\mathbf{k}$. Since $\beta = (I - P)\mathbf{k} \leq \mathbf{r}$, an upper bound for this run time is $\mathbf{k} \leq (I - P)^{-1}\mathbf{r}$.

6. Concluding Remarks

We conclude with a few directions for future research.

Combinatorics of recurrent states. The recurrent states of the rotor network $\texttt{Rotor}(G, s)$ are the oriented spanning trees of G rooted at s. The recurrent states of the sandpile network $\texttt{Sand}(G, s)$ on an undirected (or Eulerian directed) graph G have a characterization in terms of "forbidden subconfigurations" [12] which puts them naturally in bijection with the G-parking functions of Postnikov and Shapiro [31]. Recently Guzmán and Klivans [22] have generalized this correspondence to toppling networks. Hopkins and Perkinson [25] relate the G-parking functions to the Pak-Stanley labeling of a bigraphical hyperplane arrangement. It would be interesting to find combinatorial characterizations of the recurrent states of other abelian networks.

The rank function in Baker and Norine's Riemann-Roch theorem for graphs [4] has the following interpretation: given a state of the sandpile network $\texttt{Sand}(G)$ with no sink, what is the smallest number of letters that can be input to cause it to run forever? Are there analogues of the Baker-Norine theorem and the Lorenzini zeta function [29] for more general abelian networks?

Duval, Klivans and Martin [14, 15] define a higher dimensional critical group as the cokernel (over $\mathbb{Z}$) of the Laplacian of a simplicial complex. Does this group have a dynamical interpretation in terms of a "hypernetwork" in which a set of nodes can interact if they form a face of the simplicial complex? Can the spanning trees of a directed hypergraph, as defined by Gorodezky and Pak [18], be realized as the recurrent states of a suitable hypernetwork?

Critical networks. Let us call an abelian network $\mathcal{N}$ *critical* if its production matrix has spectral radius 1. By Theorem 2.9, a critical network has inputs that cause it to run forever, so it does not have a critical group in the sense of §3. However, one could try to define $\text{Crit}\,\mathcal{N}$ by generalizing the sinkless construction of the sandpile group: $(\mathbb{Z}^V)_0/L\mathbb{Z}^V$, where $(\mathbb{Z}^V)_0$ is the kernel of the map $\mathbf{x} \mapsto \mathbf{1}^T\mathbf{x}$.

For $\alpha, \beta \in \mathbb{N}^A \times Q$, write $\alpha \to \beta$ if there exists a legal execution from α to β. Let us call α *recurrent* if for any β such that $\alpha \to \beta$ we have $\beta \to \alpha$. In a critical network $\mathcal{N}$, is there an efficient test analogous to the burning algorithm to check whether α is recurrent?

In the case of a simple rotor network $\texttt{Rotor}(G)$ with no sink, a state $\alpha = 1_v.\mathbf{q}$ with just one letter is recurrent in this sense if and only if its rotors $(u, \mathbf{q}_u)_{u \in V}$ form a *cycle-rooted spanning tree* (a spanning subgraph with a single oriented cycle) with v lying on the cycle [23, Theorem 3.8]. Does this result extend to

networks of abelian mobile agents (a non-unary generalization of rotors proposed in [7])? What about states α with more than one letter?

Finer algebraic invariants. The critical group $\mathrm{Crit}\,\mathcal{N}$ depends only on the homotopy type of $\mathcal{N}$ (Corollary 3.12). On the other hand, the global monoid M of (6) can detect finer information about $\mathcal{N}$. To see that the monoid *action* $M \times Q \to Q$ is not a homotopy invariant, note that the sandpile network $\mathtt{Sand}(G, s)$ has a state $\mathbf{0} \in Q$ which can access all other states: $M\mathbf{0} = Q$. If $G - \{s\}$ has two directed cycles that share an edge, then $\mathtt{Rotor}(G, s)$ has no such state, because a progressed cycle of rotors once broken can never be reformed.

Neither is M itself a homotopy invariant: for instance, for the discrete torus $G = \mathbb{Z}/n \times \mathbb{Z}/n$ one can show that the minimal burning element β of $M(\mathtt{Rotor}(G, s))$ has $\beta^n = e$, whereas the corresponding exponent for β in $M(\mathtt{Sand}(G, s))$ grows quadratically in n.

Recall that there are many distinct rotor networks $\mathtt{Rotor}(G, s)$ depending on the choice of ordering of the outgoing edges of each vertex. An interesting question is whether M can distinguish between these networks.

Acknowledgments

References

[1] László Babai and Evelin Toumpakari, A structure theory of the sandpile monoid for directed graphs, 2010. http://people.cs.uchicago.edu/~laci/REU10/evelin.pdf

[2] Roland Bacher, Pierre de La Harpe and Tatiana Nagnibeda, The lattice of integral flows and the lattice of integral cuts on a finite graph, *Bull. Soc. Math. France* **125**(2):167–198, 1997.

[3] Per Bak, Chao Tang and Kurt Wiesenfeld, Self-organized criticality: an explanation of the $1/f$ noise, *Phys. Rev. Lett.* **59**(4):381–384, 1987.

[4] Matthew Baker and Serguei Norine. Riemann-Roch and Abel-Jacobi theory on a finite graph, *Adv. Math.* **215**(2):766–788, 2007.

[5] Matthew Baker and Farbod Shokrieh, Chip-firing games, potential theory on graphs, and spanning trees, *J. Combin. Theory A* **120**(1):164–182, 2013

[6] Norman L. Biggs, Chip-firing and the critical group of a graph, *J. Algebraic Combin.* **9**(1):25–45, 1999.

[7] Benjamin Bond and Lionel Levine, Abelian networks I. Foundations and examples. Preprint, 2013. arXiv:1309.3445

[8] Benjamin Bond and Lionel Levine, Abelian networks II. Halting on all inputs. *Selecta Math.*, to appear. arXiv:1409.0169

[9] Scott Chapman, Rebecca Garcia, Luis David Garca-Puente, Martin E. Malandro and Ken W. Smith, Algebraic and combinatorial aspects of sandpile monoids on directed graphs, *J. Comb. Theory A* **120**(1):245–265, 2013. arXiv:1105.2357

[10] Joshua Cooper and Joel Spencer, Simulating a random walk with constant error, *Combin. Probab. Comput.* **15**:815–822, 2006.

[11] Arnoud Dartois and Dominique Rossin, Height-arrow model, *Formal Power Series and Algebraic Combinatorics*, 2004.

[12] Deepak Dhar, Self-organized critical state of sandpile automaton models, *Phys. Rev. Lett.* **64**:1613–1616, 1990.

[13] Deepak Dhar, The abelian sandpile and related models, *Physica A* **263**:4–25, 1999. arXiv:cond-mat/9808047

[14] Art M. Duval, Caroline J. Klivans and Jeremy L. Martin, Simplicial matrix-tree theorems, *Trans. Amer. Math. Soc.* **361**(11):6073–6114, 2009. arXiv:0802.2576

[15] Art M. Duval, Caroline J. Klivans and Jeremy L. Martin, Critical groups of simplicial complexes, *Ann. Combin.* **17**:53–70, 2013. arXiv:1101.3981

[16] M. Fiedler and V. Ptak, On matrices with non-positive off-diagonal elements and positive principal minors, *Czechoslovak Math. J.* **12**:382–400, 1962.

[17] Tobias Friedrich and Lionel Levine, Fast simulation of large-scale growth models, *Random Struct. Alg.* **42**:185–213, 2013. arXiv:1006.1003.

[18] Igor Gorodezky and Igor Pak, Generalized loop-erased random walks and approximate reachability, *Random Struct. Alg.* **44**(2):201–223, 2014.

[19] J. A. Green, On the structure of semigroups, *Ann. of Math.* 54:163–172, 1951.

[20] Pierre A. Grillet, *Commutative semigroups*, Klower Academic Publishers, 2001.

[21] Pierre A. Grillet, Commutative actions, *Acta Sci. Math. (Szeged)* **73**:91–112, 2007.

[22] Johnny Guzmán and Caroline Klivans, Chip-firing and energy minimization on M-matrices, *J. Combin. Theory A* **132**:14–31, 2015. arXiv:1403.1635

[23] Alexander E. Holroyd, Lionel Levine, Karola Mészáros, Yuval Peres, James Propp and David B. Wilson, Chip-firing and rotor-routing on directed graphs, in *In and out of equilibrium 2*, pages 331–364, Progress in Probability **60**, Birkhäuser, 2008. arXiv:0801.3306

[24] Alexander E. Holroyd and James G. Propp, Rotor walks and Markov chains, in *Algorithmic Probability and Combinatorics*, American Mathematical Society, 2010. arXiv:0904.4507

[25] Sam Hopkins and David Perkinson, Bigraphical arrangements, *Trans. Amer. Math. Soc.*, to appear, 2014. arXiv:1212.4398

[26] Itamar Landau and Lionel Levine, The rotor-router model on regular trees, *J. Combin. Theory A* **116**: 421–433, 2009. arXiv:0705.1562

[27] Dino J. Lorenzini, Arithmetical graphs, *Math. Ann.* **285**(3):481–501, 1989.

[28] Dino J. Lorenzini, A finite group attached to the Laplacian of a graph, *Discrete Math.* **91**(3):277–282, 1991.

[29] Dino J. Lorenzini, Two-variable zeta-functions on graphs and Riemann-Roch theorems, *Int. Math. Res. Notices* **22**:5100–5131, 2012.

[30] David Perkinson, Jacob Perlman and John Wilmes, Primer for the algebraic geometry of sandpiles, in *Tropical and Non-Archimedean Geometry*, pages 211–256, Contemp. Math. **605**, American Mathematical Society, 2013. arXiv:1112.6163

[31] Alexander Postnikov and Boris Shapiro, Trees, parking functions, syzygies, and deformations of monomial ideals. *Trans. Amer. Math. Soc.* **356**(8):3109–3142, 2004. arXiv:math.CO/0301110

[32] V. B. Priezzhev, Deepak Dhar, Abhishek Dhar and Supriya Krishnamurthy, Eulerian walkers as a model of self-organised criticality, *Phys. Rev. Lett.* **77**:5079–5082, 1996. arXiv:cond-mat/9611019

[33] Marcel-Paul Schützenberger, $\overline{\mathcal{D}}$ représentation des demi-groupes, *C. R. Acad. Sci. Paris* 244:1994–96, 1957.

[34] Eugene R. Speer, Asymmetric abelian sandpile models. *J. Stat. Phys.* **71**:61–74, 1993.

[35] Benjamin Steinberg, A theory of transformation monoids: combinatorics and representation theory, *Electr. J. Combin.* **17**:R164, 2010. arXiv:1004.2982

BENJAMIN BOND, DEPARTMENT OF MATHEMATICS, STANFORD UNIVERSITY, STANFORD, CALIFORNIA 94305. http://stanford.edu/~benbond

Lionel Levine, Department of Mathematics, Cornell University, Ithaca, NY 14853.
http://www.math.cornell.edu/~levine