Abelian Networks

Lionel Levine

Berkeley combinatorics seminar

November 7, 2011

An overview of abelian networks

- Dhar's model of abelian distributed processors
- Example: abelian sandpile (a.k.a. chip-firing)

Themes:

1. Local-to-global principles
2. Halting problem
3. Critical group

Joint work with:

Anne Fey (Delft), Yuval Peres (Microsoft), James Propp (Lowell), Ben Bond, Giuliano Giacaglia, Linda Zayas-Palmer (MIT).

Mathematical model of a distributed network

- Finite or infinite directed graph G.
- At each vertex v is an automaton ("processor") with state space Q_v and input alphabet A_v.
- Each processor has
  - A single input feed.
  - Multple output feeds, one for each directed edge (v, u).

Transition and message-passing functions

- Formally: for each v and each edge $e = (v, u)$, we are given maps

$$T_v : Q_v \times A_v \to Q_v \qquad \text{(new internal state)}$$
$$T_e : Q_v \times A_v \to A_u^* \qquad \text{(messages sent along } e\text{)}$$

where $A^* = \bigcup_{n \geq 0} A^n$.

Transition and message-passing functions

- Formally: for each v and each edge $e = (v, u)$, we are given maps

$$T_v : Q_v \times A_v \to Q_v \qquad \text{(new internal state)}$$
$$T_e : Q_v \times A_v \to A_u^* \qquad \text{(messages sent along } e\text{)}$$

  where $A^* = \bigcup_{n \geq 0} A^n$.

- Input: User sends messages to one or more processors.
- Output: States of the processors when no messages remain.

Abelian Distributed Processors

- "In many applications, especially in computer science, one considers such networks where the speed of the individual processors is unknown, and where the final state and outputs generated should not depend on these speeds. Then it is essential to construct the network so that the order at which messages arrive to the processors is immaterial."

  – Deepak Dhar (1998)

A Wish List

Regardless of the order in which individual processors act:

- The halting status is the same.
- If the computation halts, then the final output is the same.

A Wish List

Regardless of the order in which individual processors act:

- The halting status is the same.
- If the computation halts, then the final output is the same.
- The run time (number of messages processed) is the same.
- The local run times are the same.
- The specific local run times are the same.

A Wish List

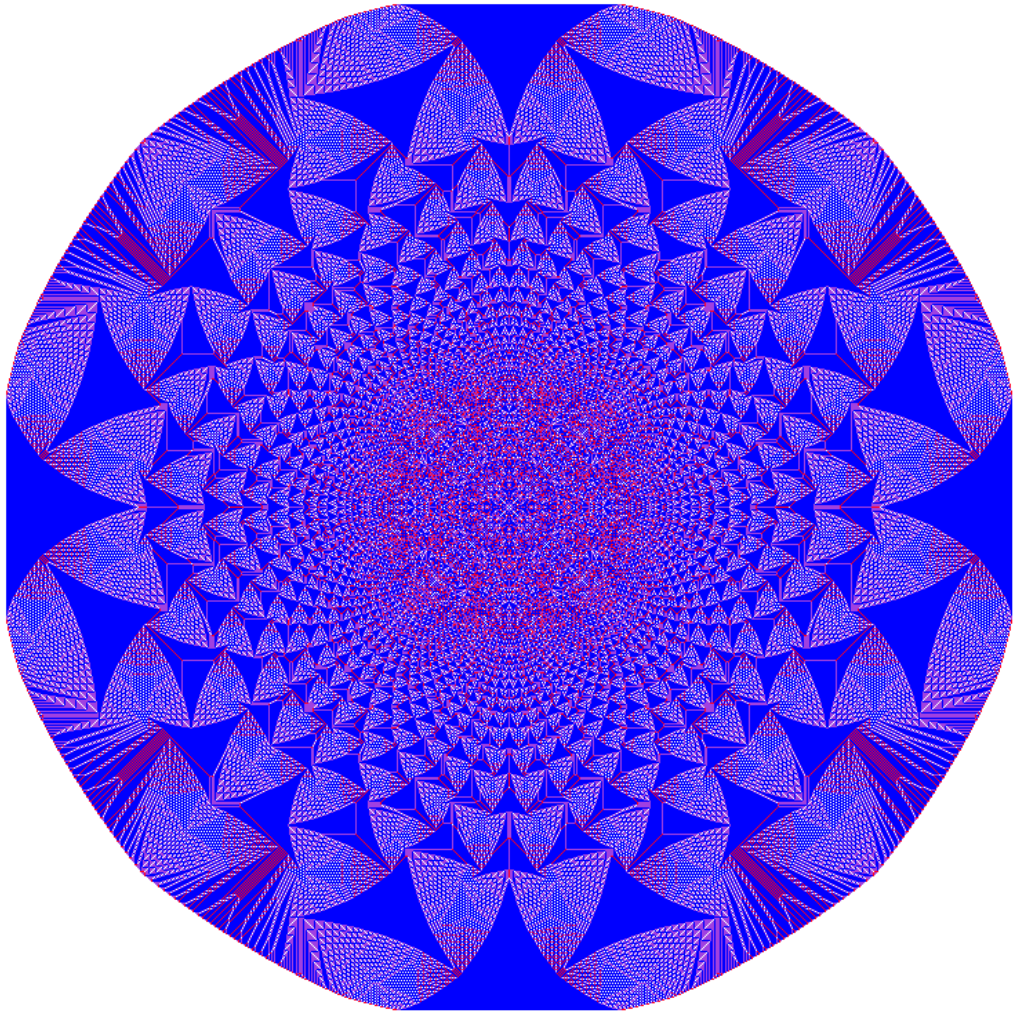
Regardless of the order in which individual processors act:

- The halting status is the same.
- If the computation halts, then the final output is the same.
- The run time (number of messages processed) is the same.
- The local run times are the same.
- The specific local run times are the same.

But are there any interesting examples...?

Are there any interesting examples...?

Yes!



The abelian sandpile model (a.k.a. chip-firing)

- Start with a pile of n chips at the origin in $\mathbb{Z}^2$.
- Each site $(x, y) \in \mathbb{Z}^2$ has 4 neighbors

$$(x \pm 1, y) \quad \text{and} \quad (x, y \pm 1).$$

- Any site with at least 4 chips is unstable, and topples by sending one chip to each neighbor.

The abelian sandpile model (a.k.a. chip-firing)

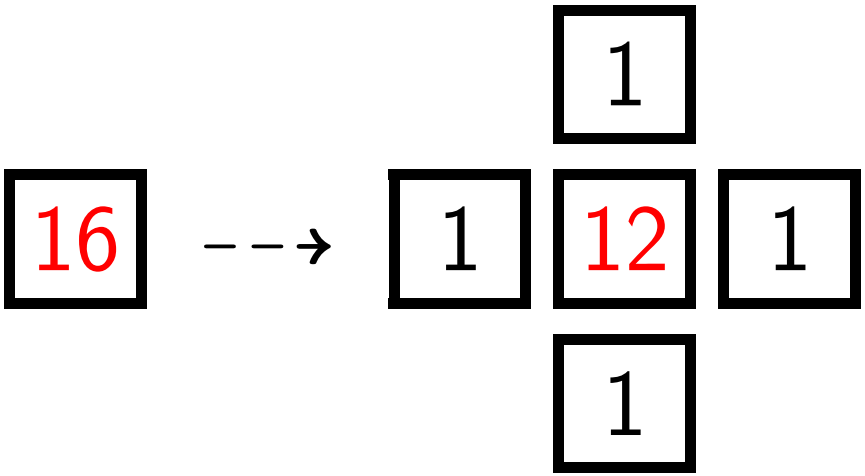
- Start with a pile of n chips at the origin in $\mathbb{Z}^2$.
- Each site $(x, y) \in \mathbb{Z}^2$ has 4 neighbors

$$(x \pm 1, y) \quad \text{and} \quad (x, y \pm 1).$$

- Any site with at least 4 chips is unstable, and topples by sending one chip to each neighbor.
- This may create further unstable sites, which also topple.
- Continue until there are no more unstable sites.
- **Bak-Tang-Wiesenfeld** 1987, **Dhar** 1990, ...
- **Björner-Lovász-Shor** 1991, **Biggs** 1999, ...

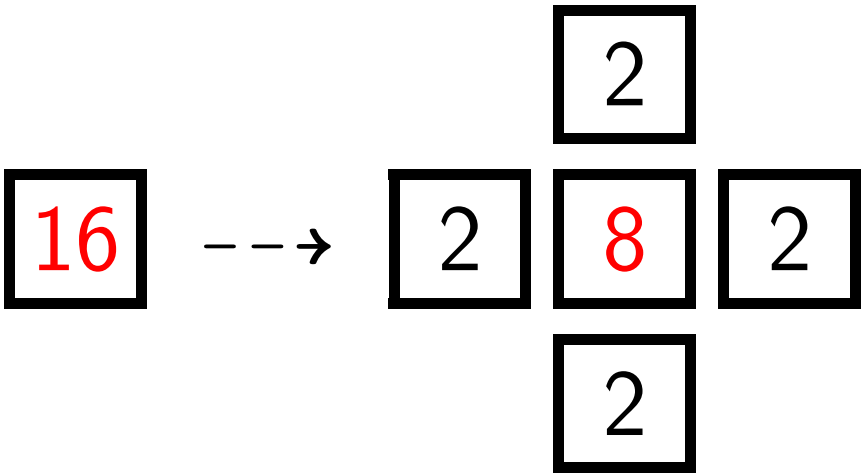
Toppling to Stabilize A Sandpile

- Example: n=16 chips in $\mathbb{Z}^2$.
- Sites with 4 or more chips are unstable.



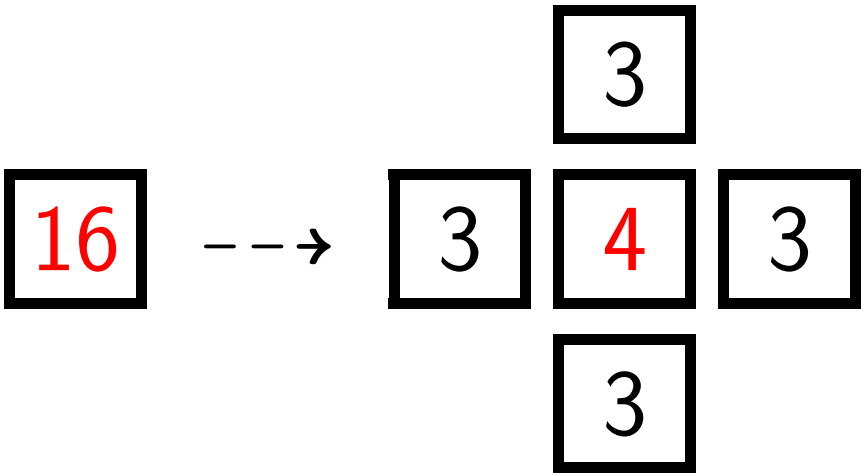
Toppling to Stabilize A Sandpile

- Example: n=16 chips in $\mathbb{Z}^2$.
- Sites with 4 or more chips are unstable.



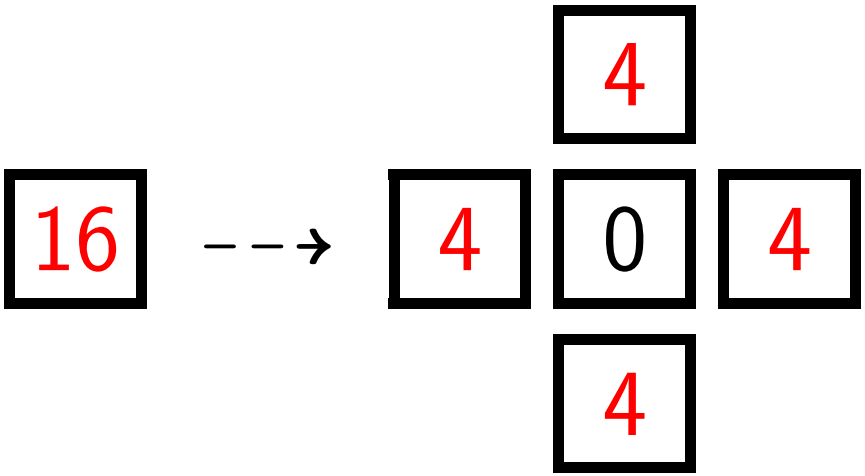
Toppling to Stabilize A Sandpile

- Example: n=16 chips in $\mathbb{Z}^2$.
- Sites with 4 or more chips are unstable.



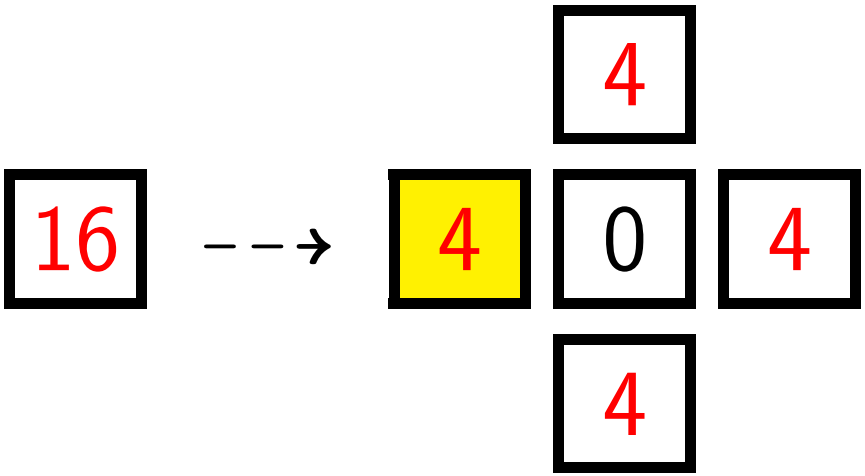
Toppling to Stabilize A Sandpile

- Example: n=16 chips in $\mathbb{Z}^2$.
- Sites with 4 or more chips are unstable.



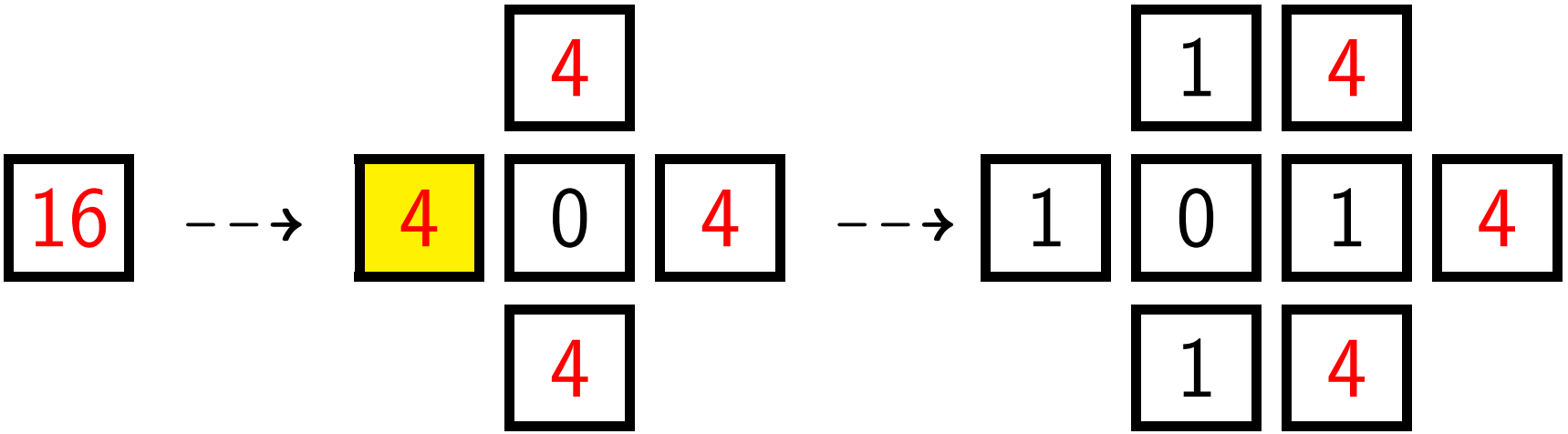
Toppling to Stabilize A Sandpile

- Example: n=16 chips in $\mathbb{Z}^2$.
- Sites with 4 or more chips are unstable.



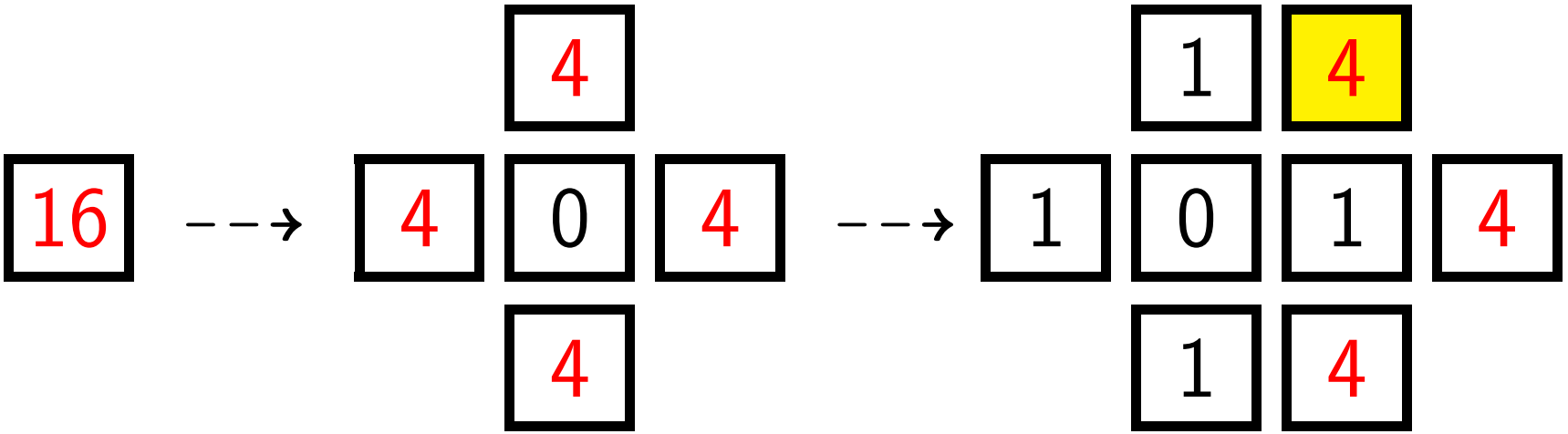
Toppling to Stabilize A Sandpile

- Example: n=16 chips in $\mathbb{Z}^2$.
- Sites with 4 or more chips are unstable.



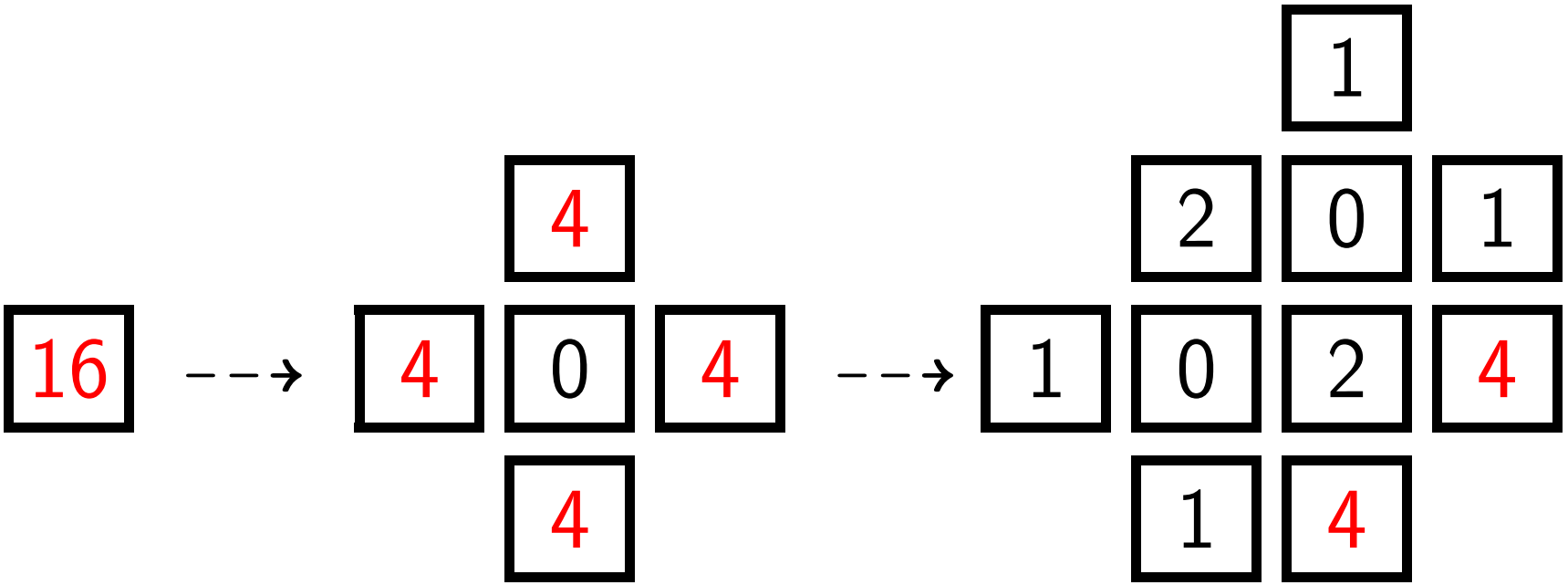
Toppling to Stabilize A Sandpile

- Example: n=16 chips in $\mathbb{Z}^2$.
- Sites with 4 or more chips are unstable.



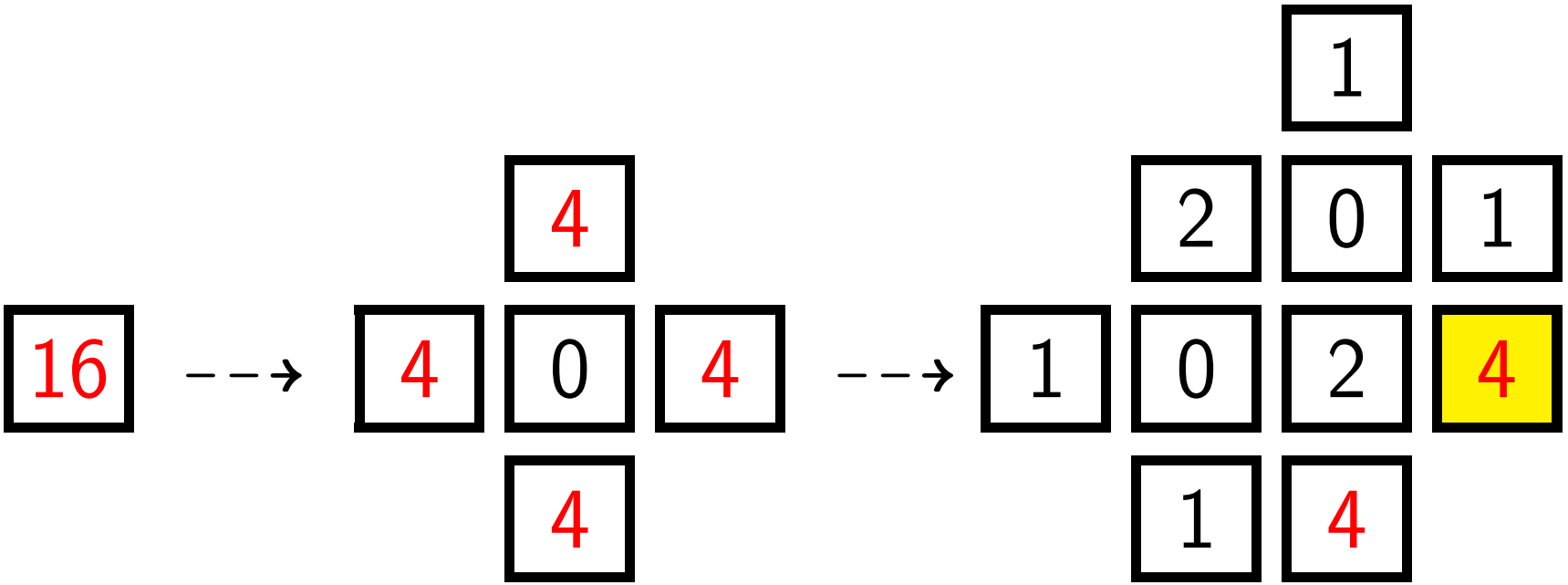
Toppling to Stabilize A Sandpile

- Example: n=16 chips in $\mathbb{Z}^2$.
- Sites with 4 or more chips are unstable.



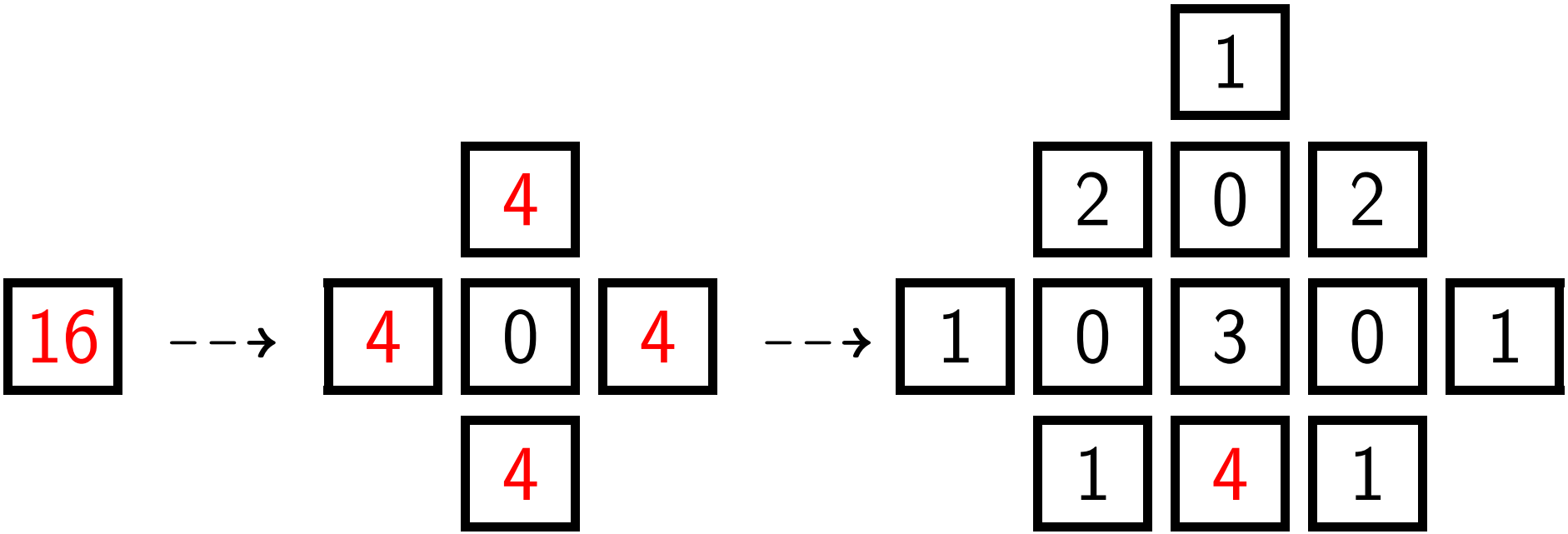
Toppling to Stabilize A Sandpile

- Example: n=16 chips in $\mathbb{Z}^2$.
- Sites with 4 or more chips are unstable.



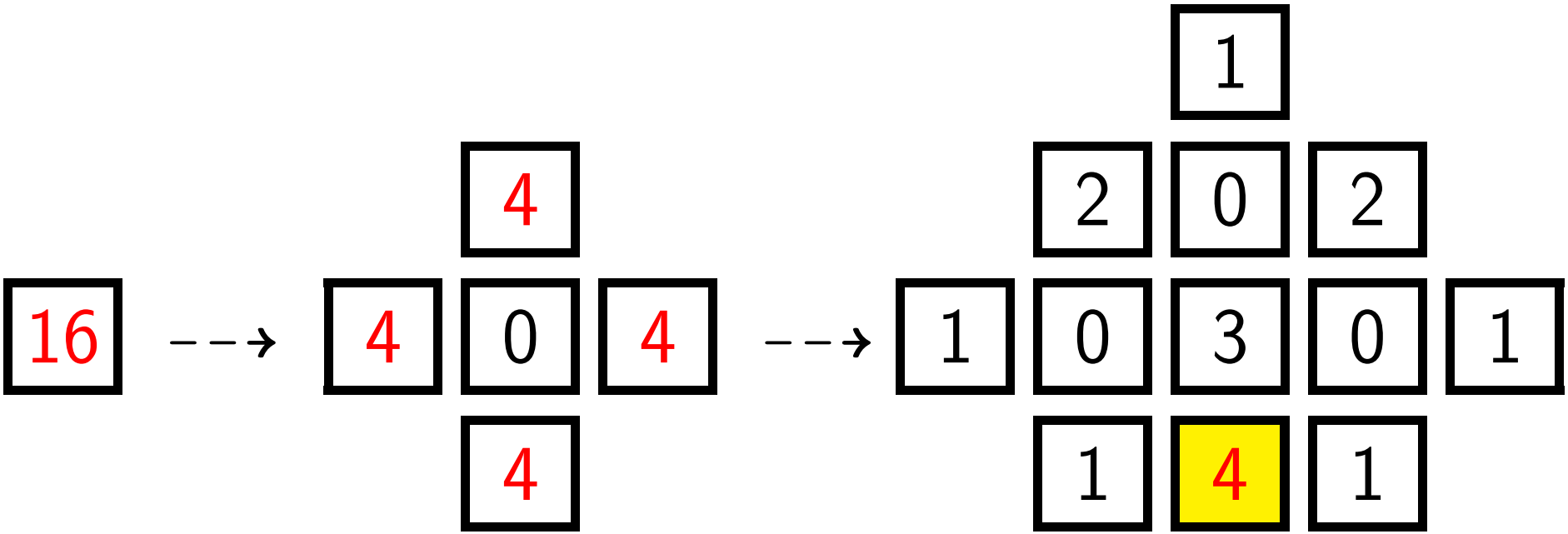
Toppling to Stabilize A Sandpile

- Example: n=16 chips in $\mathbb{Z}^2$.
- Sites with 4 or more chips are unstable.



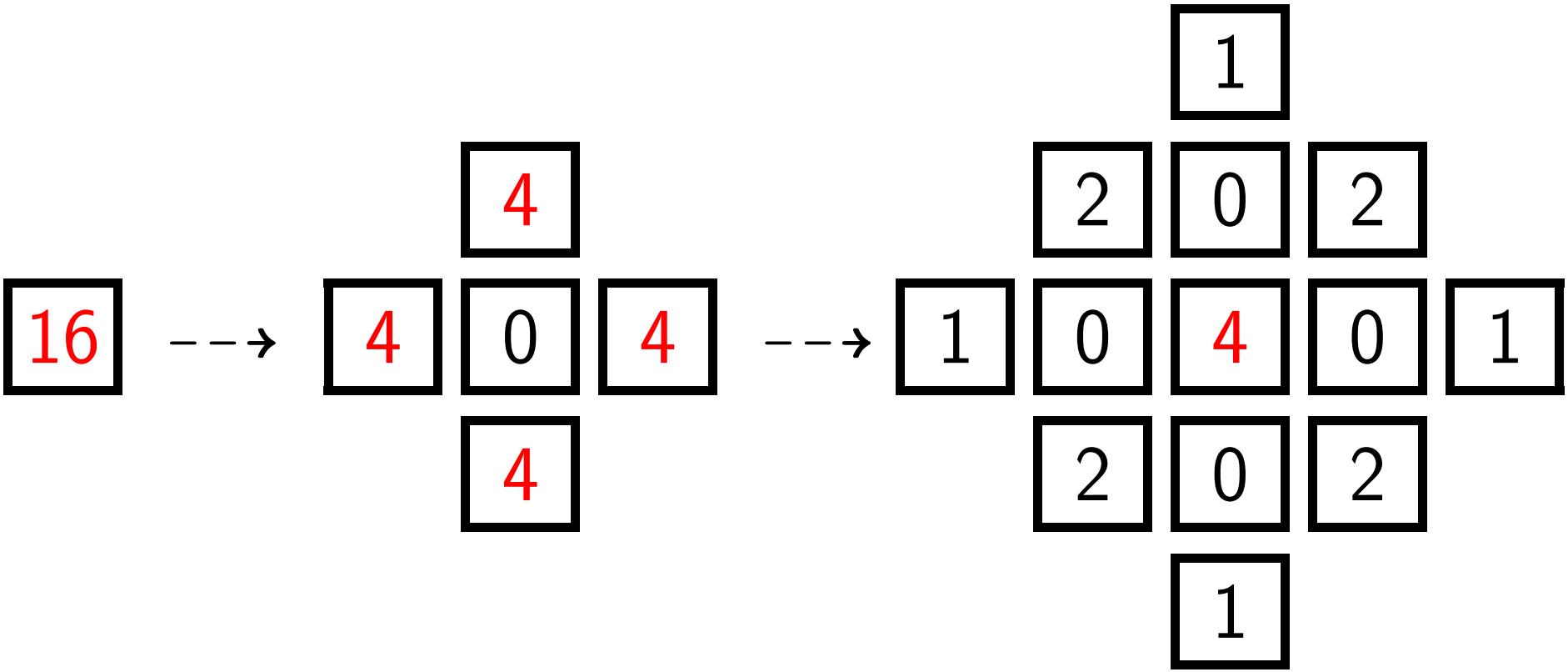
Toppling to Stabilize A Sandpile

- Example: n=16 chips in $\mathbb{Z}^2$.
- Sites with 4 or more chips are unstable.



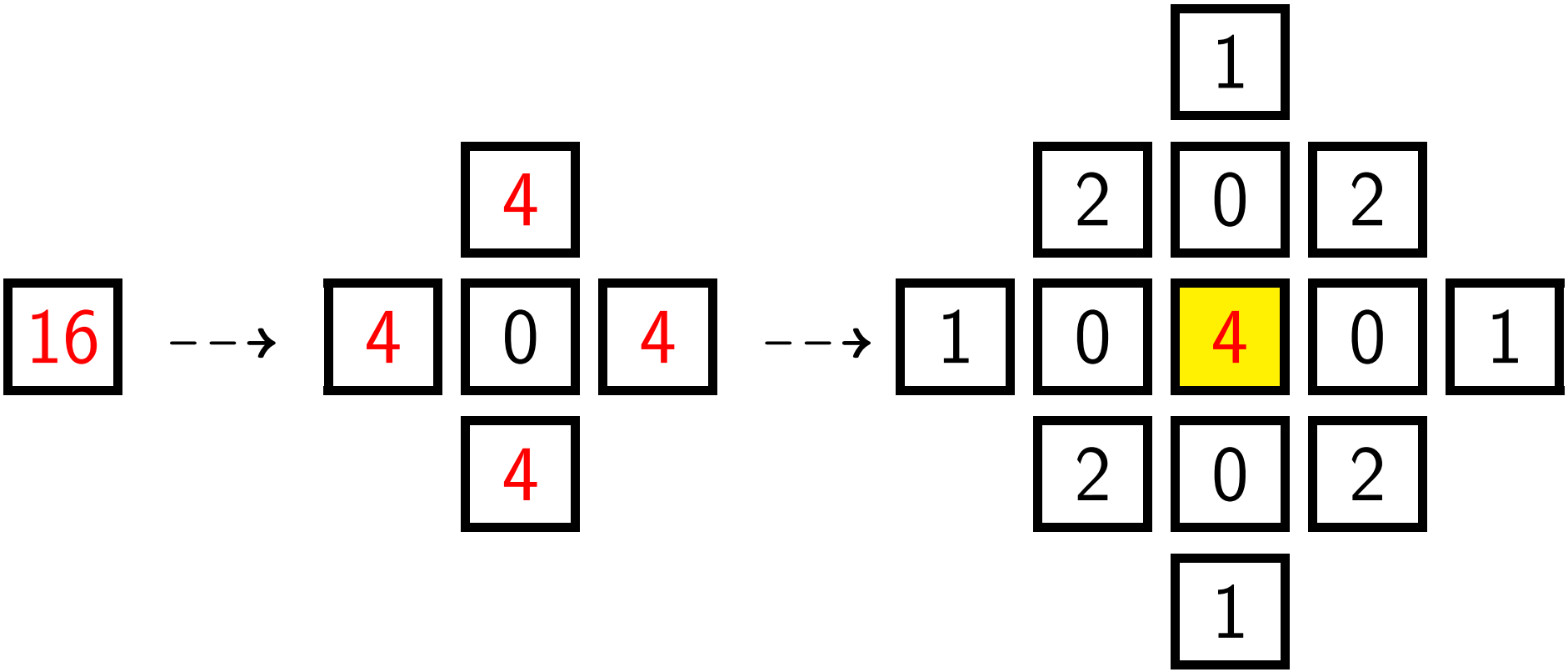
Toppling to Stabilize A Sandpile

- Example: n=16 chips in $\mathbb{Z}^2$.
- Sites with 4 or more chips are unstable.



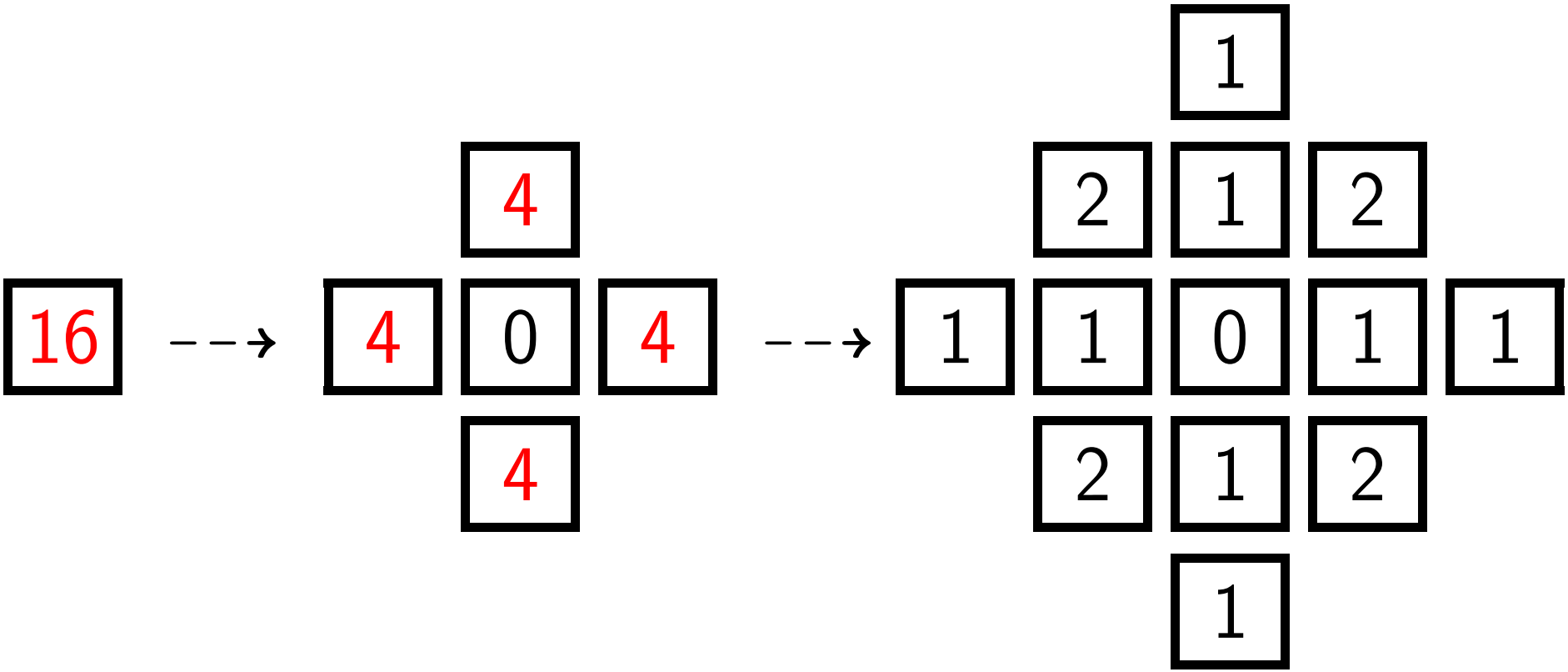
Toppling to Stabilize A Sandpile

- Example: n=16 chips in $\mathbb{Z}^2$.
- Sites with 4 or more chips are unstable.



Toppling to Stabilize A Sandpile

- Example: n=16 chips in $\mathbb{Z}^2$.
- Sites with 4 or more chips are unstable.

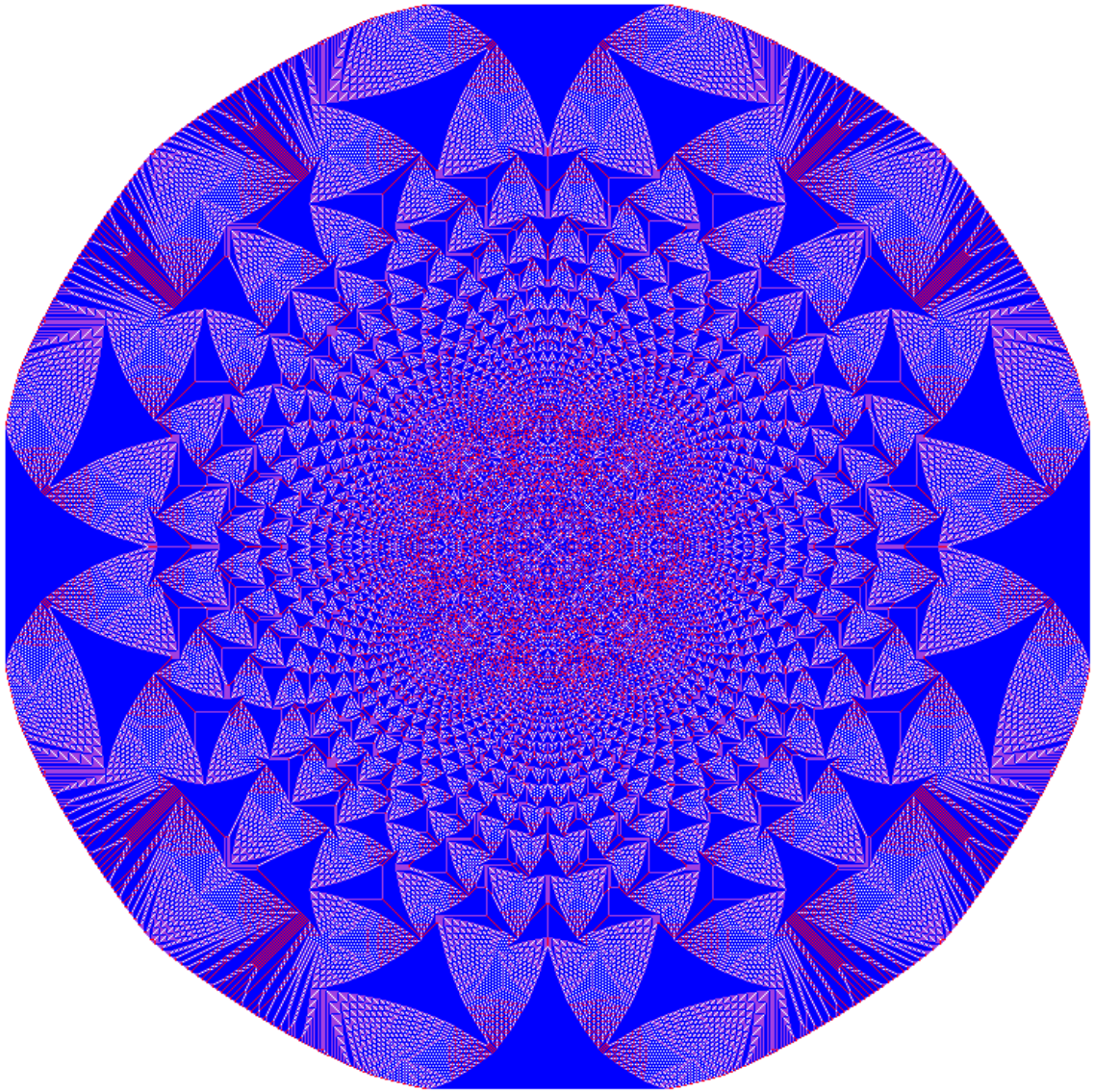


Stable.

Abelian Property

- The final stable configuration does not depend on the order of topplings.
- Neither does the number of times a given vertex topples.

Sandpile of $1\,000\,000$ chips in $\mathbb{Z}^2$



Sandpiles as an abelian network

- State space

$$Q_v = \{0, 1, \ldots, d_v - 1\}$$

  where d_v is the outdegree of vertex v.
- Unary alphabet $|A_v| = 1$. (think of messages as chips)

Sandpiles as an abelian network

- State space

$$Q_v = \{0, 1, \ldots, d_v - 1\}$$

  where d_v is the outdegree of vertex v.
- Unary alphabet $|A_v| = 1$. (think of messages as chips)
- Transition function:

$$T_v : Q_v \to Q_v$$
$$q \mapsto q + 1 \pmod{d_v}$$

- Messge passing function for each edge $e = (v, u)$:

$$T_e(q) = \begin{cases} 1 & \text{if } q = 0 \\ 0 & \text{if } q > 0. \end{cases}$$

Abelian networks

- Recall: directed graph G; for each v and each edge $e = (v, u)$ we have maps

$$T_v : Q_v \times A_v \to Q_v \qquad \text{(new internal state)}$$
$$T_e : Q_v \times A_v \to A_u^* \qquad \text{(messages sent along } e\text{)}$$

- Input: User sends one or more messages to one processor.
- Output: States of the processors when no messages remain.

Main definition

- An *abelian network* $\mathcal{N}$ is a directed graph $G = (V, E)$ with an abelian processor $\mathcal{P}_v$ at each vertex $v \in V$.

Main definition

- An *abelian network* $\mathcal{N}$ is a directed graph $G = (V, E)$ with an abelian processor $\mathcal{P}_v$ at each vertex $v \in V$.
- Processor $\mathcal{P}_v$ is called *abelian* if for any state $q \in Q_v$ and any two inputs $w, w' \in A_v^*$ such that $|w| = |w'|$,

$$T_v(q, w) = T_v(q, w')$$

Main definition

- An *abelian network* $\mathcal{N}$ is a directed graph $G = (V, E)$ with an abelian processor $\mathcal{P}_v$ at each vertex $v \in V$.
- Processor $\mathcal{P}_v$ is called *abelian* if for any state $q \in Q_v$ and any two inputs $w, w' \in A_v^*$ such that $|w| = |w'|$,

$$T_v(q, w) = T_v(q, w') \quad \text{and} \quad |T_e(q, w)| = |T_e(q, w')|$$

for all edges $e = (v, u)$.

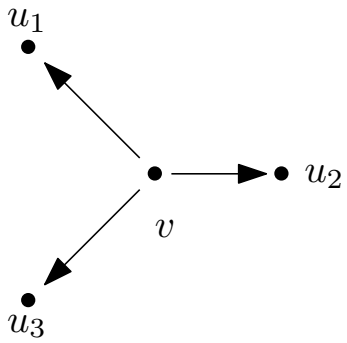
Main definition

- An *abelian network* $\mathcal{N}$ is a directed graph $G = (V, E)$ with an abelian processor $\mathcal{P}_v$ at each vertex $v \in V$.
- Processor $\mathcal{P}_v$ is called *abelian* if for any state $q \in Q_v$ and any two inputs $w, w' \in A_v^*$ such that $|w| = |w'|$,

$$T_v(q, w) = T_v(q, w') \quad \text{and} \quad |T_e(q, w)| = |T_e(q, w')|$$

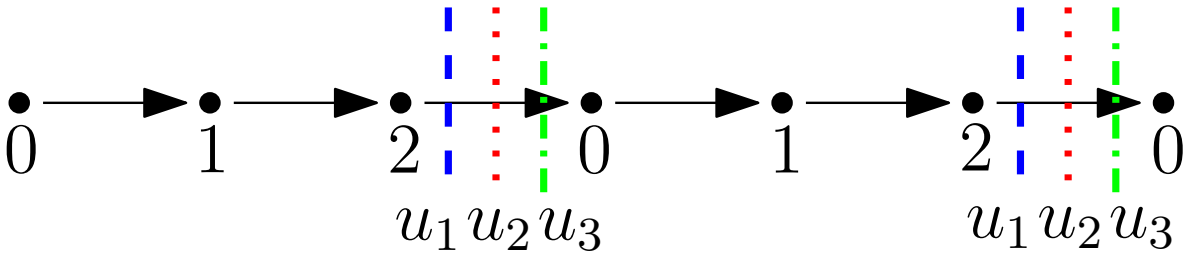
for all edges $e = (v, u)$.

- Note: the definition is local.
- Note: A unary processor ($|A_v| = 1$) is trivially abelian.

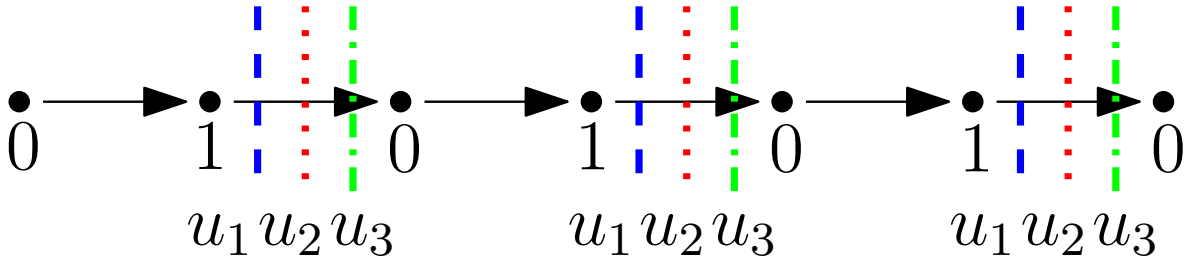


Diagramming a unary processor

- Processor at v in a sandpile network:

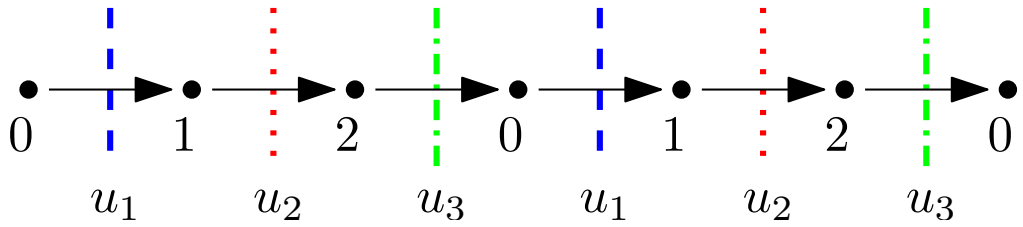


- Processor at v in a toppling network:

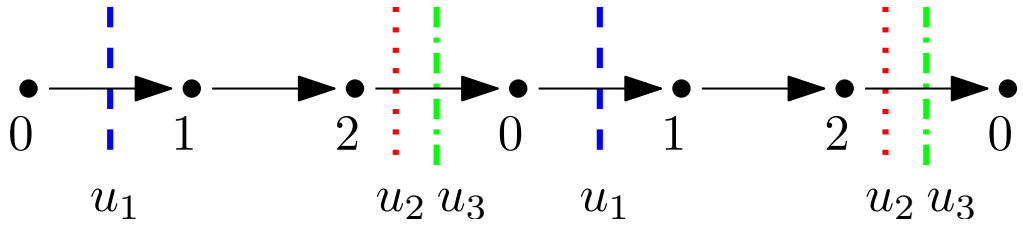


A zoo of unary processors

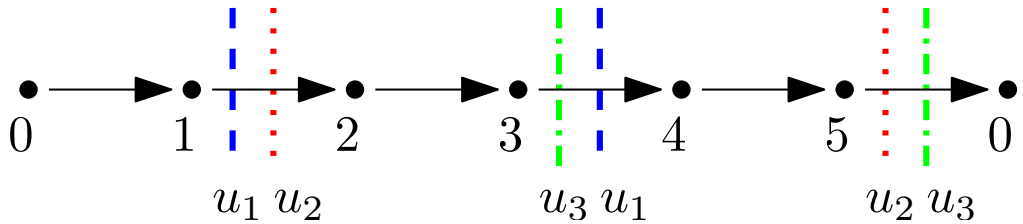
- Rotor-router (**PDDK** 1996, **WLS** 1996, **Propp** 2000):



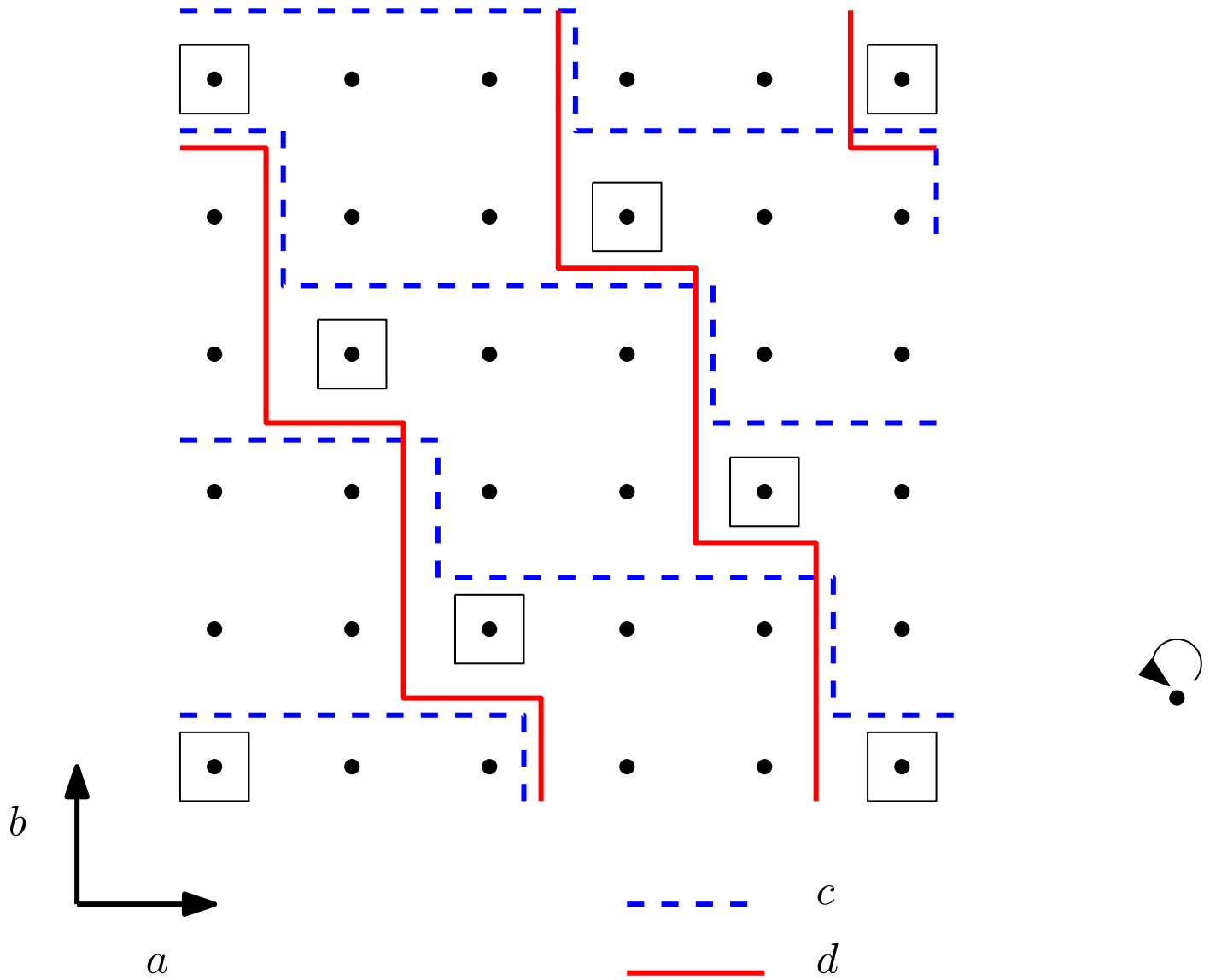
- Periodically mutating game (**Eriksson** 1996):



- Height-arrow model (**Dartois-Rossin** 2004):



State diagram of a single abelian processor with input alphabet $\{a, b\}$ and output alphabet $\{c, d\}$



Least Action Principle

- Execution sequence: word $w = a_1 \cdots a_n \in A^*$ where $A = \sqcup A_v$.
- **Lemma**: if w is legal and w' is complete, then

$$|w|_a \leq |w'|_a \qquad \text{for all } a \in A.$$

- Generalizes **Diaconis-Fulton** 1991 (and the proof is no harder).

Least Action Principle

- Execution sequence: word $w = a_1 \cdots a_n \in A^*$ where $A = \sqcup A_v$.
- **Lemma**: if w is legal and w' is complete, then

$$|w|_a \leq |w'|_a \qquad \text{for all } a \in A.$$

- Generalizes **Diaconis-Fulton** 1991 (and the proof is no harder).

- w is *legal* for $(\mathcal{N}, q_0)$ if at least one message of type a_i is present after processing $a_1 \cdots a_{i-1}$, for $i = 1, \ldots, n$.
- w' is *complete* for $(\mathcal{N}, q_0)$ if no messages are present after processing w'.

Local Abelianness Implies Global Abelianness

Corollaries of LAP: For fixed $(\mathcal{N}, q_0)$,

- **Halting status**: If there is a complete word of length n, then all legal words have length $\leq n$.

Local Abelianness Implies Global Abelianness

Corollaries of LAP: For fixed $(\mathcal{N}, q_0)$,

- **Halting status**: If there is a complete word of length n, then all legal words have length $\leq n$.
- **Run times**: if w and w' are both complete and legal, then

$$|w|_a = |w'|_a \qquad \text{for all } a \in A.$$

(i.e., w and w' have the same specific local run times!)

Local Abelianness Implies Global Abelianness

Corollaries of LAP: For fixed $(\mathcal{N}, q_0)$,

- **Halting status**: If there is a complete word of length n, then all legal words have length $\leq n$.
- **Run times**: if w and w' are both complete and legal, then

$$|w|_a = |w'|_a \qquad \text{for all } a \in A.$$

  (i.e., w and w' have the same specific local run times!)
- **Output**: Any two complete legal words produce the same output.

Local-to-global principles

Theorem (Bond-L.; Giacaglia-L.-Propp-Zayas):

If each processor $\mathcal{P}_v$ is

abelian, then $\mathcal{N}$ is abelian.

Local-to-global principles

Theorem (Bond-L.; Giacaglia-L.-Propp-Zayas):

If each processor $\mathcal{P}_v$ is

abelian,	then $\mathcal{N}$ is abelian.
irreducible,	then $\mathcal{N}$ is irreducible.
a periodic rotor,	then $\mathcal{N}$ is a periodic rotor.
a palindromic rotor,	then $\mathcal{N}$ is a palindromic rotor.

Rotors

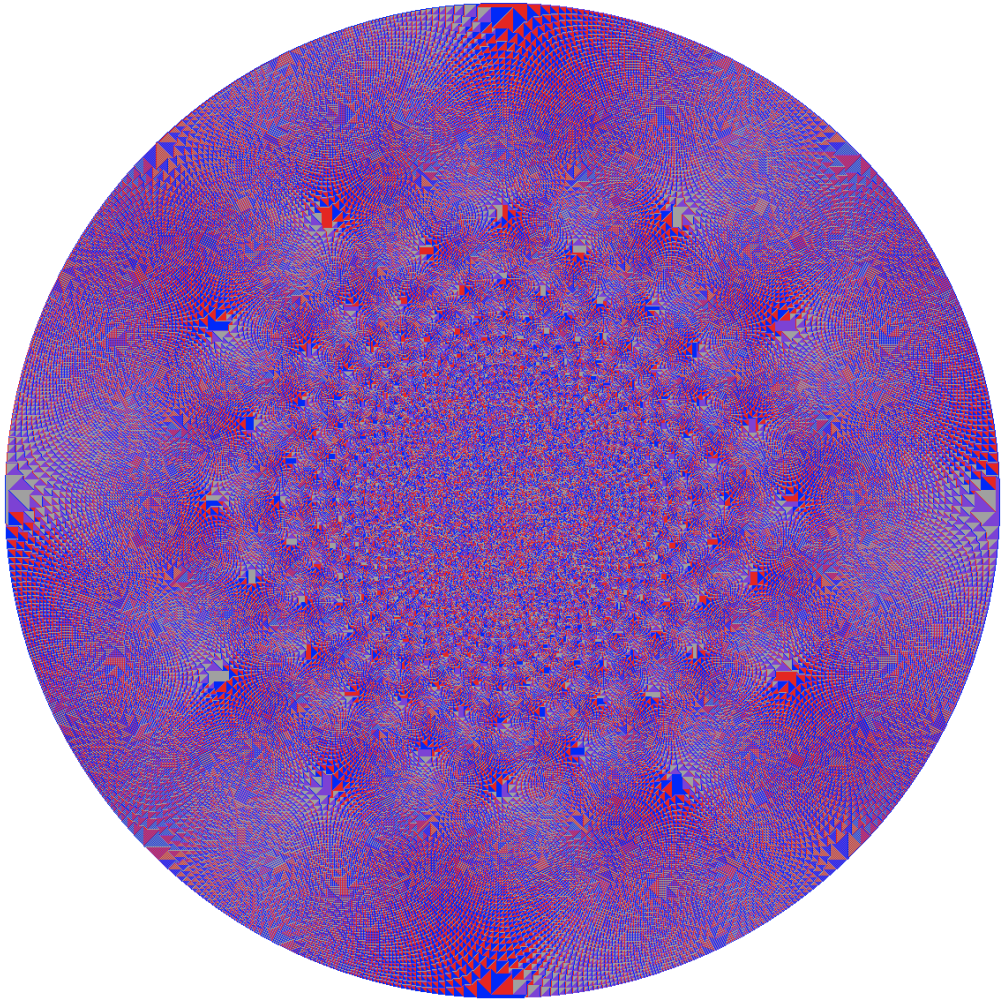
- A *rotor* is a unary processor ($|A_v|=1$) that sends exactly one output message for each input message.
- Single message input to a rotor network on graph G gives an infinite walk in G.

Rotors

- A *rotor* is a unary processor ($|A_v|=1$) that sends exactly one output message for each input message.
- Single message input to a rotor network on graph G gives an infinite walk in G.
- Invented by

  Wagner-Lindenbaum-Bruckstein 1996 ("ant walk")
  Priezzhev-Dhar-Dhar-Krishnamurthy 1996 ("Eulerian walkers")
  Propp c. 2000 ("rotor walk")

Output of a rotor network on $\mathbb{Z}^2$ on input of 1000000 messages to the origin.



http://rotor-router.mpi-inf.mpg.de/

Local-to-global principles

Theorem (Bond-L.; Giacaglia-L.-Propp-Zayas):

If each processor $\mathcal{P}_v$ is

abelian,	then $\mathcal{N}$ is abelian.
irreducible,	then $\mathcal{N}$ is irreducible.
a periodic rotor,	then $\mathcal{N}$ is a periodic rotor.
a palindromic rotor,	then $\mathcal{N}$ is a palindromic rotor.

- A rotor is called *periodic mod d* if it has state space $Q_v = \mathbb{Z}/d\mathbb{Z}$ and transition $T_v(q) = q + 1 \pmod{d}$.

The halting problem for abelian networks

- **Theorem** (Bond-L.): A finite irreducible abelian network $\mathcal{N}$ halts on all inputs if and only if all eigenvalues of its production matrix have absolute value < 1.

The halting problem for abelian networks

- **Theorem** (Bond-L.): A finite irreducible abelian network $\mathcal{N}$ halts on all inputs if and only if all eigenvalues of its production matrix have absolute value < 1.
- Production matrix: $P = (p_{ab})_{a,b \in A}$, where $A = \sqcup A_v$ and

$$p_{ab} = \lim_{n \to \infty} \frac{1}{n} |T_e(q, b^n)|_a$$

  for $a \in A_v, b \in A_u, e = (v, u)$.
- The limit does not depend on the initial state q.

Transition monoids

- Each letter $a \in A_v$ induces a map $\delta_a = T_v(\cdot, a) : Q_v \to Q_v$.
- The *local monoid* at vertex v is the submonoid $M_v \subset \operatorname{End} Q_v$ generated by $\{\delta_a\}_{a \in A_v}$.

Transition monoids

- Each letter $a \in A_v$ induces a map $\delta_a = T_v(\cdot, a) : Q_v \to Q_v$.
- The *local monoid* at vertex v is the submonoid $M_v \subset \operatorname{End} Q_v$ generated by $\{\delta_a\}_{a \in A_v}$.
- M_v is commutative since $\delta_a \circ \delta_b = \delta_b \circ \delta_a$
  and finite if Q_v is finite.

Structure of a finite commutative monoid M

- Let $e \in M$ be the product of all idempotents ($e^2 = e$).
- Then eM is a finite abelian group with identity element e.

Example: cyclic monoid $M = \mathbb{N}/(k = n)$ for fixed $k < n$.

Structure of a finite commutative monoid M

- Let $e \in M$ be the product of all idempotents ($e^2 = e$).
- Then eM is a finite abelian group with identity element e.

Example: cyclic monoid $M = \mathbb{N}/(k = n)$ for fixed $k < n$.

- e is the unique multiple of $n-k$ between k and $n-1$.
- $eM = \{k, k+1, \ldots, (n-1)\}$
  $\simeq \mathbb{Z}/(n-k)\mathbb{Z}$ (generated by $e+1$).

The critical group of an abelian network

- $\mathcal{N}$: finite abelian network that halts on all inputs.
- View $\mathcal{N}$ as a single processor with

$$\begin{aligned} &\text{state space} && Q = \prod Q_v, \\ &\text{input alphabet} && A = \sqcup A_v. \end{aligned}$$

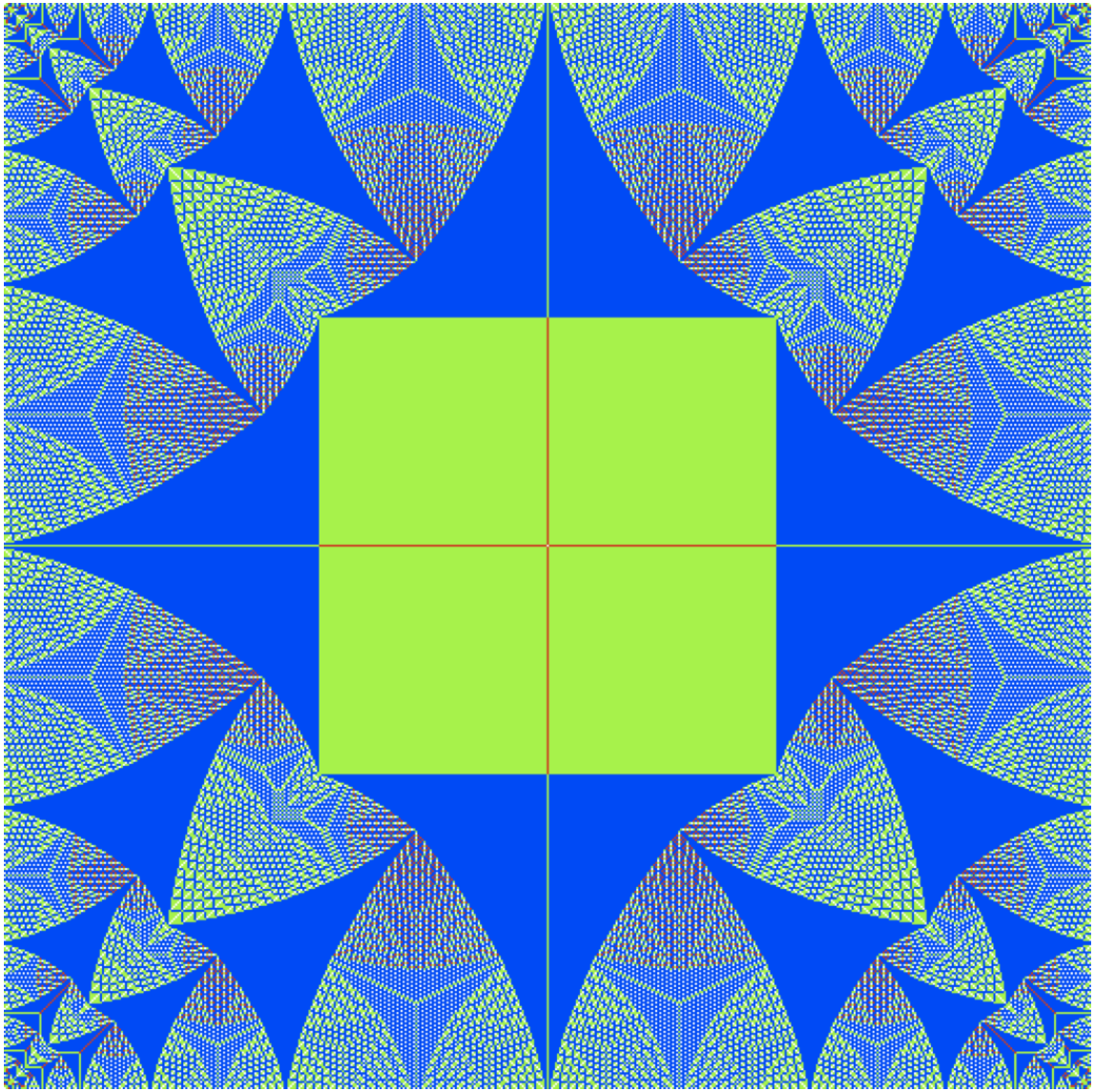
The critical group of an abelian network

- $\mathcal{N}$: finite abelian network that halts on all inputs.
- View $\mathcal{N}$ as a single processor with

$$\begin{aligned} &\text{state space} && Q = \prod Q_v, \\ &\text{input alphabet} && A = \sqcup A_v. \end{aligned}$$

- The *global monoid* of $\mathcal{N}$ is the submonoid $M \subset \mathrm{End}(Q)$ generated by $\{\delta_a\}_{a \in A}$.
- The *critical group* of $\mathcal{N}$ is the finite abelian group eM.
- Generalizes the **Babai-Toumpakari** construction of the sandpile group.

Identity element e of a sandpile network on $\mathbb{Z}_{521} \times \mathbb{Z}_{521}$



Irreducible monoid actions

- $M \times Q \to Q$ is *irreducible* if there is no partition

$$Q = Q_1 \cup Q_2$$

for Q_1, Q_2 disjoint, nonempty and $MQ_1 \subset Q_1$ and $MQ_2 \subset Q_2$.

Recurrent states of a monoid action

Let M be a finite commutative monoid and $M \times Q \to Q$ an irreducible action. The following are equivalent for $q \in Q$:

1. $q \in Mq'$ for all $q' \in Q$.
2. $q \in mQ$ for all $m \in M$.
3. $q \in eQ$.
4. $q = eq$.

The states q satisfying these conditions are called *recurrent*.

Example: spanning trees

- $\mathcal{N}$: rotor network on graph G with sink vertex s.
- The rotor at each vertex v has period d_v and serves each neighbor once.
- State space: for each v, pick an edge $e = (v, u)$.
- Recurrent states: Spanning trees oriented toward s.
- The sandpile network on G has the same number of recurrent states! Why?

From monoid action to group action

If M is a finite commutative monoid, then any monoid action

$$M \times Q \to Q$$

induces by restriction a corresponding group action

$$eM \times eQ \to eQ.$$

Freedom and transitivity

- **Lemma**: If $\mu : M \times Q \to Q$ is an irreducible monoid action, then the restriction of μ to $eM \times eQ$ is a transitive group action

$$eM \times eQ \to eQ.$$

  If in addition μ is faithful, then this group action is free.

- (We say that μ is *faithful* if there do not exist distinct $m, m' \in M$ such that $mq = m'q$ for all $q \in Q$.)

Action of the critical group

Theorem (Bond-L.) Let $\mathcal{N}$ be a finite abelian network that halts on all inputs. If each processor $\mathcal{P}_v$ is irreducible, then the action of the critical group on recurrent states

$$\mathrm{Crit}\,\mathcal{N} \times \mathrm{Rec}\,\mathcal{N} \to \mathrm{Rec}\,\mathcal{N}$$

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

Questions

- Halting problem: decide if $\mathcal{N}$ halts on a *particular* input.

Questions

- Halting problem: decide if $\mathcal{N}$ halts on a *particular* input.
- LAP shows that abelian networks can solve certain integer programs. What else can they compute? What can't they compute?

Questions

- Halting problem: decide if $\mathcal{N}$ halts on a *particular* input.
- LAP shows that abelian networks can solve certain integer programs. What else can they compute? What can't they compute?
- Understand the intricate patterns formed by sandpile and rotor networks.
- Are some patterns "inherently nonabelian"?

Thank You!

References:

- B. Bond, L. Levine, Abelian networks I. Foundations and examples, 2011. (*draft available by request; will be posted soon at* http://www.math.cornell.edu/~levine/abelian-networks-I.pdf)
- D. Dhar, The abelian sandpile and related models, 1998. arXiv:cond-mat/9808047
- D. Dhar, Theoretical studies of self-organized criticality, *Physica A*, 2006.
- A. Fey, L. Levine, Y. Peres, Growth rates and explosions in sandpiles, *J. Stat. Phys.*, 2010. arXiv:0901.3805
- G. Giacaglia, L. Levine, J. Propp, L. Zayas-Palmer, Local-to-global principles for rotor walk, 2011. arXiv:1107.4442
- J. Propp, Discrete analog computing with rotor routers, *Chaos*, 2010. arXiv:1007.2389