Classes of Lattices Induced by Chip Firing (and Sandpile) Dynamics

Clémence Magnien

Abstract

In this paper we study three classes of models widely used in physics, computer science and social science: the Chip Firing Game, the Abelian Sandpile Model and the Chip Firing Game on a mutating graph. We study the set of configurations reachable from a given initial configuration, called the *configuration space* of a model, and try to determine the main properties of such sets. We study the order induced over the configurations by the evolution rule. This makes it possible to compare the power of expression of these models. It is known that the configuration spaces we obtain are lattices, a special kind of partially ordered set. Although the Chip Firing Game on a mutating graph is a generalization of the usual Chip Firing Game, we prove that these models generate exactly the same configuration spaces. We also prove that the class of lattices induced by the Abelian Sandpile Model is strictly included in the class of lattices induced by the Chip Firing Game, but contains the class of distributive lattices, a very well known class.

Introduction

The Chip Firing Game (CFG), the Abelian Sandpile Model (ASM), and the Chip Firing Game on a mutating graph (which we will call Mutating Chip Firing Game or MCFG) are closely related models studied in physics [BTW87, DRSV95], computer science [GMP98, BLS91, BL92] and social science [Big97, Big99, Heu99]. These three models are variations of the following game: given a graph and a distribution of chips on its vertices (called *configuration*), one may select a vertex that contains at least as many chips as its outdegree, and move one chip from this vertex along each of its outgoing edges, to the vertex at the other extremity. Many questions have naturally arisen and given matter for research about this game: given a graph and an initial configuration, does the game stop after some time, or can it be played forever [BLS91, BL92, Eri96] ? For a given graph, what are the properties of the configurations such that no move is possible ? This has led to the algebraic study of some of these configurations called *recurrent configurations* [DRSV95, CR00, Big99]. Given a graph, what can be said about the set of the configurations that can be reached from a given initial configuration [LP01, MPV01] ?

This paper is devoted to the study of this last question. Our purpose is to study the set of configurations reachable from a given initial configuration, which we will call the *configuration space* of the game. This set is naturally ordered by the relation of reachability induced by the evolution rule. In [MPV01] it

was proved that the configuration spaces induced by CFGs have a very strong structure, and that the class they form is situated between two very well-known classes of orders (the distributive and the ULD lattices). Our aim is to determine if the same kind of results also holds for the two other models. We will try to determine the differences between the classes of configuration spaces induced by each model, *i.e.* given the configuration space of a game, we will try to decide if it is isomorphic to the configuration space of a game of another type. For instance, given the configuration space of a CFG, does there exist a MCFG or an ASM such that its configuration space is isomorphic to it ?

In Section 1, we give the definitions and known results used in this paper, as well as the definitions of the models. In Section 2, we show that CFGs and MCFGs induce exactly the same class of configuration spaces, and in Section 3, we will compare the configuration spaces of CFGs and ASMs. Since an ASM is a special CFG, we will study under which condition a CFG can be transformed into an ASM. We will give a sufficient (but not necessary) condition of such a transformation, and we will give an example of a CFG that cannot be transformed into an ASM.

1 Definitions and known results

We will need some definitions from order and lattice theory to describe the configuration spaces of the models. We give them first, after what we give the definition of each model, as well as the previously known results about them.

1.1 Posets and lattices

A *partially ordered set* (or *poset*) is a set equipped with an order relation $\leq$ (*i.e.* a transitive, reflexive and antisymmetric relation). If x and y are two elements of a poset, we say that x is *covered* by y (or y *covers* x), and write $x \prec y$ (or $y \succ x$) if $x < y$ and $x \leq z < y$ implies $z = x$. We then say that x is a *lower cover* of y (or that y is an *upper cover* of x). The *interval* $[x, y]$ is the set $\{z, x \leq z \leq y\}$. To represent a poset P we will use its Hasse diagram, defined as follows :

- each element x of P is represented by a point p_x of the plane,
- if $x < y$, then p_x is lower than p_y, and
- p_x and p_y are joined by a line if and only if $x \prec y$.

Two posets P and P' are *isomorphic* if there exists a bijection $\varphi : P \longrightarrow P'$ satisfying: for all $x, y \in P$, $x \leq y \iff \varphi(x) \leq \varphi(y)$.

A poset L is a *lattice* if any two elements x, y of L have a least upper bound (called *join* and denoted by $x \vee y$) and a greatest lower bound (called *meet* and denoted by $x \wedge y$). The join $x \vee y$ is the (unique) smallest element greater than both x and y, and $x \wedge y$ is defined dually. All the lattices considered here are finite, therefore they have a unique maximal and a unique minimal element.

A lattice is a *hypercube of dimension* n if it is isomorphic to the set of all subsets of a set of n elements, ordered by inclusion. Hypercubes are also called *boolean lattices*. A lattice is *upper locally distributive* (denoted by *ULD*) [Mon90] if the interval between any element and the join of all its upper covers is a

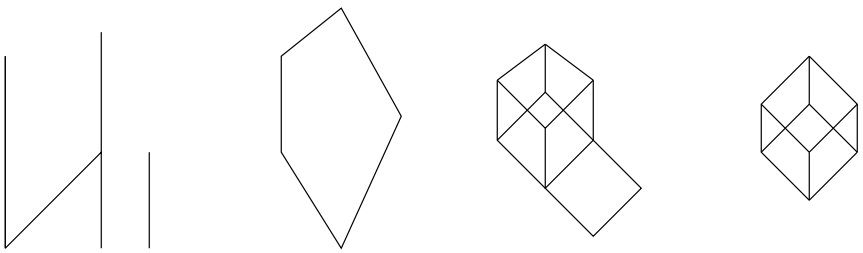

Figure 1: From left to right: a poset, a lattice, a distributive lattice and a hypercube of dimension 3

hypercube. *Lower locally distributive* (LLD) lattices are defined dually. All ULD lattices are ranked, *i.e.* all the paths in the covering relation from the minimal to the maximal element have the same length. A lattice L is *distributive* if it satisfies one of the two following relations of distributivity (that are equivalent and imply each other):

$$\text{for all } x, y, z \in L, x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$$

$$\text{for all } x, y, z \in L, x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z).$$

A distributive lattice is a lattice that is at the same time upper *and* lower locally distributive, *i.e.* if the interval between any element and, on the one hand the join of its upper covers, and on the other hand the meet of all its lower covers both are hypercubes. In the sequel we will only be concerned with distributive and ULD lattices. Figure 1 shows examples of posets and different types of lattices.

For a more complete introduction to posets and lattices, see for instance [DP90].

1.2 The different kinds of Chip Firing Games

In this section we give the definitions of the Abelian Sandpile Model, the Chip Firing Game and the Mutating Chip Firing Game. We begin by presenting the features shared by the three models. Then we detail what is specific about each of them. We will see that some models are generalizations of others. Finally, we give some results about the CFG which will be useful in this paper.

1.2.1 Definitions

Each model is defined over a graph $G = (V, E)$, called the *support graph* of the game (undirected graphs will be regarded as directed by replacing each undirected edge $\{i, j\}$ by the two directed edges (i, j) and (j, i)). All graphs are supposed to be multigraphs, *i.e.* multiple edges between two vertices are allowed, therefore all edge sets are also supposed to be multisets. A *configuration* of the game is a mapping $\sigma : V \mapsto \mathbb{N}$ which associates a weight to each vertex; this weight may be considered as a number of *chips* stored in the vertex. The game is played with respect to the following evolution rule, also called the *firing* rule: if a vertex v contains at least as many chips as its outdegree, we can transfer a chip from v along each of its outgoing edges to the corresponding

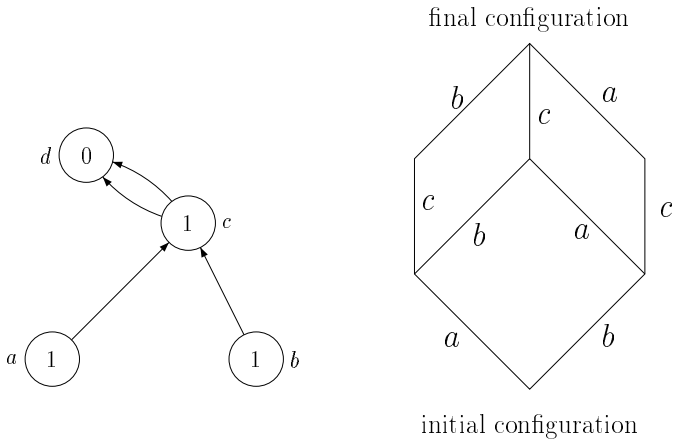

Figure 2: An initial configuration of a CFG and its configuration space. Each edge (σ, σ') is labelled by the name of the vertex fired to reach σ' from σ.

vertex. We call this process *firing* v. If σ is the configuration we start from, and σ' is obtained from σ by firing v, we write $\sigma \xrightarrow{v} \sigma'$, and we call σ a *predecessor* of σ'.

Note 1.1 We consider that the firing rule cannot be applied to a vertex with no outgoing edges because firing it does not change the configuration of the game and is therefore of no interest to us.

The Chip Firing Game (CFG) [BL92] is defined over a directed graph. We give an example of a CFG together with its configuration space in Figure 2. The Abelian Sandpile Model (ASM) [BTW87] is defined over an undirected graph, with a distinguished vertex called the *sink* and denoted by $\perp$. The sink can never be fired. Finally the Chip Firing Game on a mutating graph (MCFG) [Eri96] is played on a directed graph that changes during the game in the following way: after the firing of a vertex, its outgoing edges are removed and new ones are added in a pre-determined way (the vertex set V remains the same). A *position* of a MCFG consists in:

- a directed graph on V,
- on each node $v \in V$ a nonnegative number of chips,
- for each node $v \in V$ an infinite sequence M_v of multisets of nodes in V, which is the mutation sequence of v.

A vertex v may be fired if the number of chips on v is at least $d^+(v)$ (the current number of outgoing edges of v). The mutation of the graph takes place after the firing of the vertex. The outgoing edges of v are removed, and if $M_v = (S_v^1, S_v^2, \ldots)$ is the mutation sequence of v, then a new edge (v, w) is created for each $w \in S_v^1$. Finally, S_v^1 is removed from M_v, so that M_v becomes $(S_v^2, S_v^3, \ldots)$. We define the *initial support graph* of a MCFG to be its support graph in the initial position. An example of a firing sequence in a MCFG is given in Figure 3.

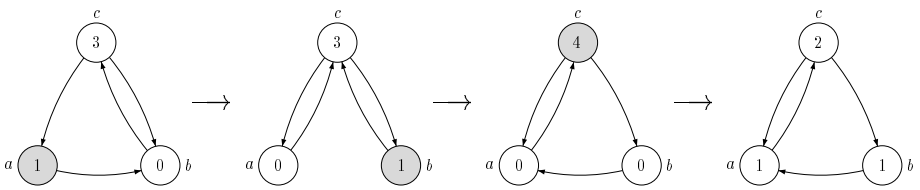

Figure 3: An example of a firing sequence in a MCFG, where the beginning of the mutation sequences of the vertices are : $M_a = (\{c\}, \ldots), M_b = (\{a\}, \ldots), M_c = (\{a, b\}, \ldots)$

These three games are strongly convergent games [Eri93], which implies that, given an initial configuration, either a given game can be played forever or it reaches a unique fixed point (where no firing is possible), called the *final configuration*, which does not depend on the order in which the vertices are fired. We will only consider *convergent* games,*i.e.* games that reach a fixed point. We call *execution* of a game any sequence of firing that, starting from the initial configuration, reaches the final configuration.

We call *configuration space* of a game C, and we denote it by $L(C)$, the set of all the configurations reachable from the initial configuration, ordered by the reflexive and transitive closure of the predecessor relation. It is known [BLS91, LP01, Eri96] that when a game is convergent, its configuration space is a ULD lattice. This is a very strong property, because ULD lattices are very structured sets. For instance, an immediate consequence of this is the fact that all the firing sequences, in any convergent ASM, CFG or MCFG, from the initial configuration to the final configuration, have the same length. We will say that two convergent games C and C' are *equivalent* if $L(C)$ is isomorphic to $L(C')$.

We denote by $L(CFG)$, $L(ASM)$ and $L(MCFG)$ the classes of lattices that are the configurations spaces of convergent CFGs, ASMs and MCFGs respectively. It is possible to guarantee that a CFG is convergent by the presence in the support graph of a sink (a vertex with no outgoing edges) reachable from all vertices [LP01]. This also holds for MCFGs. In the ASM, the presence of the distinguished vertex which can never be fired (called the sink because it has the same function as in the directed case) guarantees that the game is always convergent.

These three models are very close to one another, and it is easy to see that some of these are generalizations of others (the class of lattices they induce are included in one another): since undirected graphs are particular directed graphs, one can consider an ASM as a particular CFG, where each undirected edge $\{u, v\}$ with $u, v \neq \perp$ is replaced by two opposite directed edges (u, v) and (v, u), and where each undirected edge $\{v, \perp\}$ is replaced only by the directed edge $(v, \perp)$. Therefore, we obtain that $L(ASM) \subseteq L(CFG)$. Likewise, a CFG can be regarded as a MCFG where the graph remains the same after each mutation, therefore $L(CFG) \subseteq L(MCFG)$. Our aim is to study the other relations between these classes of lattices.

1.2.2 Previous results

Now let us give some definitions and known results about CFGs, useful for simplifying the notations and proofs in the sequel.

Definition 1.2 *A convergent ASM, CFG or MCFG is* simple *if, during an execution, each vertex is fired at most once.*

Theorem 1.3 *[MPV01] Any convergent CFG is equivalent to a simple CFG.*

Thanks to this result, all the CFGs considered in the sequel will be simple, and such that their support graph has exactly one sink (denoted by $\perp$) and such that all vertices except $\perp$ are fired during an execution. This is always possible because, if this is not the case:

- either there is no sink and all vertices are fired during an execution; then we can add an isolated vertex to the graph, which becomes the sink,
- or there exists a vertex v that is never fired during any execution but is not a sink; then we can remove its outgoing edges without changing the configuration space of the CFG, and v becomes the sink,
- or there is more than one sink; then we can merge them into a single vertex without changing the configuration space.

In the sequel, we will restrict ourselves to CFGs for which the support graph has no loops (*i.e.* no (v, v) edges). This is always possible because, for a simple CFG C, if there is a loop on a vertex v (v cannot be the sink), we can replace it by an edge $(v, \perp)$, and the resulting CFG is equivalent to C, since C is *simple*.

Given a vertex v, we denote by $d^-(v)$ its indegree, by $d^+(v)$ its outdegree, by $d_\perp(v)$ the number of edges from v to $\perp$, and we define $d(v) = d^+(v) - d_\perp(v)$. Given two vertices u and v, we denote by $d(u, v)$ the number of edges from u to v. Given a CFG and a vertex v of its support graph, the initial number of chips in v is denoted by $\sigma_0(v)$, and the number of chips in v in the final configuration by $\sigma_f(v)$ (since all the CFGs considered are simple, we have $\sigma_f(v) = \sigma_0(v) + d^-(v) - d^+(v)$). The number of chips *needed to fire* v is the difference between $\sigma_0(v)$ and $d^+(v)$, *i.e.* 0 if $\sigma_0(v) \geq d^+(v)$, and $d^+(v) - \sigma_0(v)$ otherwise.

The configuration spaces of simple CFGs can be described more easily than in the general case. Indeed we have the following results:

Lemma 1.4 *[MPV01, LP01] In a simple CFG, if, starting from the same configuration, two sequences of firing lead to the same configuration, then the vertices fired in each sequence are the same.*

This allows us to define the *shot-set* $s(\sigma)$ of a configuration σ as the set of vertices fired to reach σ from the initial configuration. Given a CFG with support graph (V, E), we say that a subset $X \subseteq V$ is a *valid* shot-set if there exists a configuration σ reachable from the initial configuration such that $s(\sigma) = X$. A list $(v_1, \ldots, v_n)$ of vertices is a *valid firing sequence* if, for each i, $\{v_1, \ldots, v_i\}$ is a valid shot-set. The configuration space of a CFG is isomorphic to the lattice of the shot-sets of its configurations, ordered by inclusion. The join of any two elements a and b is given by the following formula [LP01]:

$$s(a \vee b) = s(a) \cup s(b).$$

Here we give another way to characterize the configuration space of a CFG with respect to the shot-sets: given a CFG and its vertex set, we can associate

with each vertex v, the set of the configurations in which v can be fired. Among these, we distinguish the smallest configurations (the ones such that their shot-set is minimal with respect to the inclusion), and we say that their shot-set represent the *first times* at which v can be fired. For instance, in Figure 2, the shot-sets of the minimal configurations in which c can be fired are $\{a\}$ and $\{b\}$, and we say that the first times at which c can be fired are a and b. The minimal configuration in which a and b can be fired is the minimal point of the lattice, and its shot-set is the empty set. We say that the first time at which a and b can be fired is the beginning of any execution. The knowledge, for all vertices, of the first times at which they can be fired, is a characterization of the configuration space of a CFG, as it is stated in the following proposition:

Proposition 1.5 *Let C be a simple CFG and let $L = L(C)$. Let $\{v_1, \ldots, v_n, \perp\}$ be the set of vertices of C. For each $i, \in \{1, \ldots, n\}$, let X_i be the set representing the first times at which v_i can be fired (*i.e. *X_i is the set of the shot-sets of the minimal configurations at which v_i can be fired). Then L is completely determined by $\{v_1, \ldots, v_n\}$ and $\{X_1, \ldots, X_n\}$.*

Proof : We will construct L from $\{v_1, \ldots, v_n\}$ and $\{X_1, \ldots, X_n\}$. Notice that, for each $i = 0, \ldots, n$, and for each $x \in X_i$, the set $\{v_i\} \cup x$ is a valid shot-set of C: by definition, x is the shot-set of a configuration in which v_i can be fired. We claim that all subsets Y of $\{v_1, \ldots, v_n\}$ satisfying the following condition:

$$\text{for all } v_i \in Y, \text{ there exists } Z \in X_i, \text{ such that } Z \subseteq Y$$

are valid shot-sets of elements of L. Let Y be such a set. Y can be decomposed into a union of sets $Y_1, \ldots, Y_k$ such that, for each $j = 1, \ldots, k$, there exists i such that Y_j is the union of $\{v_i\}$ and of an element x of X_i (notice that the sets $Y_1, \ldots, Y_k$ are not a partition of Y, some sets may overlap). Therefore, each of the sets Y_j is a valid shot-set of C. Since the shot-sets are stable by union, Y is itself a valid shot-set of C. On the other hand, every shot-set of C satisfies the condition above.

Therefore the set of subsets of $\{v_1, \ldots, v_n\}$ that satisfy this condition is the set of shot-sets of C, which, ordered by inclusion, is isomorphic to L. □

This result will be useful in the sequel when we have to guarantee that a modification made to a game does not change its configuration space. We now have all the definitions and tools needed to compare the classes of lattices induced by the models we have introduced.

2 Comparison of MCFGs and CFGs

In this section we show that the CFGs and the MCFGs induce exactly the same class of configuration spaces. We already know from Section 1.2.1 that any CFG is equivalent to a MCFG, therefore we need to show that any MCFG is equivalent to a CFG.

We begin by proving that any MCFG is equivalent to a simple one. If a MCFG is not simple, then there exists a vertex that is fired more than once during an execution. In [Eri96] it is shown that the number of times this vertex

is fired is the same in *every* possible execution of the game: this vertex is fired the same number of times in all the firing sequences that lead from the initial to the final configuration. We need this result to prove the next theorem, which is an equivalent of Theorem 1.3 for MCFGs.

Theorem 2.1 *Any convergent MCFG is equivalent to a* simple *MCFG.*

Proof : Let C be a MCFG with vertex set V and initial configuration σ. Suppose C is not simple: there exists a vertex a which is fired twice or more during any execution of C. The idea of the proof is to split a into two vertices a_0 and a_1 which will be fired alternatively (the first firing of a in C corresponds to a firing of a_0 in C', the second to a firing of a_1, and so on), so that each of them is fired strictly less often than a during an execution. Therefore, by iterating this process, one can transform C into a simple MCFG.

Let G be the initial support graph of C. If for each vertex v, the sequence of mutation is $M_v = (S_v^1, S_v^2, S_v^3, \ldots)$, we define S_v^0 to be the set of outgoing edges of v in G. Therefore from now on, a MCFG will be defined by a set of vertices, an initial configuration, and for each vertex v an inifinite sequence of edge sets $M_v = (S_v^0, S_v^1, S_v^2, \ldots)$. At any time in the game, if a vertex v with mutating sequence $(S_v^0, S_v^1, S_v^2, \ldots)$ has been fired i times, then the current outgoing edges of v are those given by S_v^i. We then denote by $l_i(v)$ the number of loops on v (*i.e.* the number of occurrences of v in S_v^i), by ${d_i}^>(v)$ the number of edges going out of v which are not loops (*i.e.* ${d_i}^>(v) = d_i^+(v) - l_i(v)$), and we define ${d_i}^<(v)$ dually: ${d_i}^<(v) = {d_i}^-(v) - l_i(v)$). Finally we define N to be twice the number of chips present in the game.

Let us define the MCFG C' in the following way:

- the vertex set of C' is $V' = V \backslash \{a\} \cup \{a_0, a_1\}$, with $a_0 \notin V$ and $a_1 \notin V$.
- for each $v \neq a \in V$, and for each $i \in \mathbb{N}$ we define S_v^i, the i-th set in the mutation sequence of v, in the following way:
  - for each occurrence of a vertex $v' \neq a$ in S_v^i, there are two occurrences of v' in S_v^i,
  - for each occurrence of a in S_v^i, there is one occurrence of a_0 and one occurrence of a_1 in S_v^i.
- for each $i \in \mathbb{N}$, we define the sets $S_{a_0}^i$ and $S_{a_1}^i$, the i-th sets in the mutating sequences of a_0 and a_1, in the following way:
  - for each occurrence of a vertex $w \neq a$ in S_a^{2i} (resp. S_a^{2i+1}), there are two occurrences of w in $S_{a_0}^i$ (resp. $S_{a_1}^i$).
  - for each occurrence of a in S_a^{2i} (resp. S_a^{2i+1}), there is one occurrence of a_0 in $S_{a_0}^i$ (resp. one occurrence of a_1 in $S_{a_1}^i$).
  - there are $N - {d_{2i}}^>(a)$ occurrences of a_1 in $S_{a_0}^i$, and $N - {d_{2i+1}}^>(a)$ occurrences of a_0 in $S_{a_1}^i$.
- for any vertex $v \neq a$, $\sigma'(v) = 2\sigma(v)$.
- $\sigma'(a_0) = \sigma(a) + N$.
- $\sigma'(a_1) = \sigma(a)$.

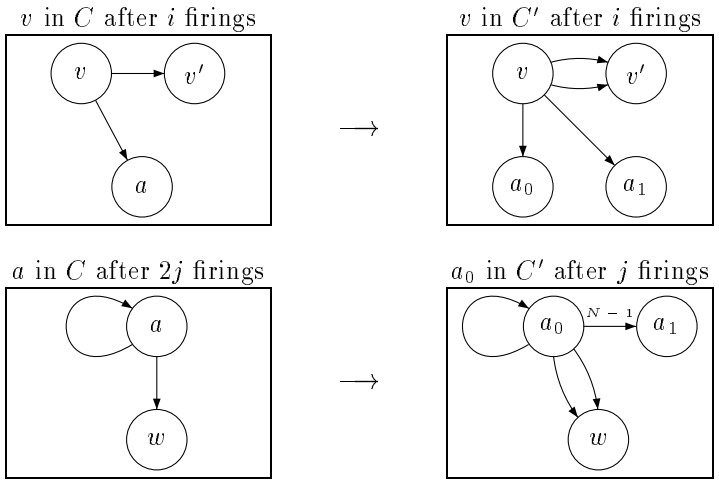

Figure 4: The outgoing edges of the vertices in the MCFG constructed during the simplification of a MCFG.

An example of the construction of the graph is given in Figure 4.

We call the *double of* C the CFG obtained from C by multiplying by two the initial configuration of all vertices, as well as the number of edges between each pair of vertices. This CFG obviously is equivalent to C, and in the sequel we will show that C' is equivalent to the double of C. Notice that, except the two vertices a_0 and a_1, C' is the same as the double of C. So if the firings of a_0 and a_1 take place in turn as stated before, then we can easily show that $L(C')$ is isomorphic to $L(C)$.

We first prove by induction that any firing sequence s of C, leading to a configuration σ, corresponds to a valid firing sequence s' in C' (by replacing the occurrences of a by occurrences of a_0 and a_1 in turn), leading to a configuration σ', satisfying Proposition 1 and Proposition 2 or 2′.

$$\text{for all } v \neq a_0, a_1, \sigma'(v) = 2\sigma(v) \tag{1}$$

$$\sigma'(a_0) = N + \sigma(a), \text{ and } \sigma'(a_1) = \sigma(a) \tag{2}$$

$$\sigma'(a_0) = \sigma(a), \text{ and } \sigma'(a_1) = N + \sigma(a) \tag{2'}$$

The initial configuration of C and C' satisfy Propositions 1 and 2. So does any configuration reached by a firing sequence that does not contain a, because, except a_0 and a_1, C' is the same as the double of C, and because for each $v \neq a_0, a_1$, there is the same number of edges from v to a_0 than from v to a_1.

Let now s be a firing sequence of C that does not contain a, but leading to a configuration σ where a can be fired. s corresponds to a firing sequence s' in C', leading to a configuration σ' satisfying Proposition 1 and Proposition 2. In σ' a_0 contains $N + \sigma(a)$ chips. By construction, the outdegree of a_0 is: $2{d_0}^{>}(a) + l_0(a) + (N - {d_0}^{>}(a)) = N + d_0^+(a)$. Therefore, the fact that a can be fired in configuration σ implies that a_0 can be fired in configuration σ'. Let σ_2 and ${\sigma_2}'$ be the configurations reached from σ and σ' by firing a and a_0. We will show that these configurations satisfy Proposition 1 and Proposition 2′. We have $\sigma_2(a) = \sigma(a) - {d_0}^{>}(a)$, and ${\sigma_2}'(a_0) = \sigma'(a_0) - {d_0}^{>}(a_0) = N + \sigma(a) - (N +$

$d_0{}^>(a)) = \sigma_2(a)$. The number of chips in a_1 becomes: $\sigma_2{}'(a_1) = \sigma'(a_1) + N - d_0{}^>(a) = N + \sigma(a) - d_0{}^>(a) = \sigma_2(a)$.

With the same arguments, we obtain that any firing sequence s of C corresponds to a valid firing sequence of C' satisfying Proposition 1 and Proposition 2 or $2'$. Notice that, for each configuration σ of C corresponding to a configuration σ' in C', the vertices that can be fired in both configurations are the same. Therefore, there is no other valid firing sequence in C' than those corresponding to firing sequences of C. Therefore $L(C')$ is isomorphic to $L(C)$.

□

For simple MCFGs, the shot-set of a configuration is defined in the same way as for CFGs. Likewise, the configuration space of any convergent MCFG is isomorphic to the set of its shot-sets ordered by inclusion. We now show that any MCFG is equivalent to a CFG.

Theorem 2.2 *Any convergent MCFG is equivalent to a CFG.*

Proof : Let C be a convergent MCFG. From Theorem 2.1 we can suppose that C is simple. Let $G = (V, E)$ and σ be the initial support graph and configuration of C. To prove our result, we will use C to construct an equivalent (simple) CFG. Given a vertex v, we define the initial degree of v to be its degree in G. When v is fired (if fired at all), a mutation occurs and its outdegree changes. Since C is simple, v will be fired only once before the end of an execution, so its outdegree changes only once. We define v's *final* degree to be its degree in the final configuration (in the case where v is not fired during an execution, the initial and final degrees are the same).

In the final configuration, v contains less chips than its final outdegree (otherwise v could be fired and we would not be in the final configuration). Two cases can occur: either v contains also less chips than its *initial* outdegree, or not. In the first case, if we suppress the mutation of v (the outgoing edges of v are the same before and after the mutation), v still cannot be fired in the final configuration. If this is the case for all vertices, we can suppress all mutations to obtain a simple classical CFG.

In the other case, there exists a vertex w which contains at the end of an execution more chips than its initial outdegree (which means that the outdegree of the vertex has been increased by the mutation). We will modify C to prevent this from happening. We proceed in the following way: let $d^+(w)$ be the initial outdegree of w, $\sigma_f(w)$ be the number of chips in w in the final configuration, and n be the number of chips needed to fire w (if $\sigma(v) \geq d$, then $n = 0$). We add $\sigma_f(w) + 1 - d^+(w)$ edges from w to the sink, and we set the initial configuration of w to $\sigma_f(w) + 1 - n$ chips. Figure 5 shows an example of this construction. After this modification, w still needs n chips to be fired, therefore the first times at which w can be fired are not modified, and w now contains less chips at the end of an execution than its initial outdegree. We do this for every vertex that does not satisfy this property, obtaining thus a MCFG that can be considered as a CFG. □

Although the MCFG is a generalization of the CFG, this theorem shows that in fact it generates no more lattices than the usual Chip Firing Game. As we will see in the next section, the Chip Firing Game generates more lattices than

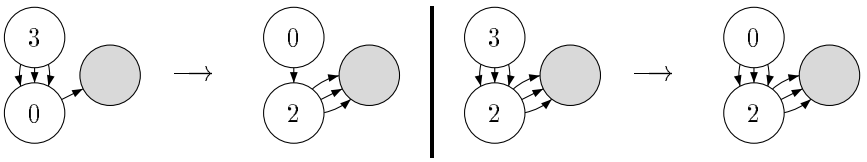

Figure 5: An example of a simple MCFG transformed into a CFG. On the left, the initial and final configurations of the MCFG. Notice that the lower vertex contains in the final configuration more chips than its initial outdegree. On the right, the initial and final configuration of the corresponding CFG.

the Abelian Sandpile Model, which is a CFG on an undirected graph. Finding out if a model defined as a Mutating Abelian Sandpile Model generates the same lattices as the usual ASM would help understand these differences better.

3 Comparison of CFGs and ASMs

In this section, we compare the classes of configuration spaces induced by CFGs and ASMs. Since we know from Section 1.2.1 that any ASM is equivalent to a CFG, we try to determine at which conditions a CFG is equivalent to an ASM. We will show that this is always the case when the CFG is simple and its support graph has no cycle, by giving an algorithm to transform such a CFG into an ASM. Since we know that any distributive lattice is the configuration space of a simple CFG with no cycle [MPV01], we obtain as a corollary that the class of lattices induced by ASM contains the distributive lattices. However, we will prove that not all CFGs are equivalent to an ASM. Since examples of lattices in $L(ASM)$ that are not distributive appear often in the study of the model, we obtain the result that $L(ASM)$ is strictly between the distributive lattices and the lattices induced by CFG, which is surprising because these two classes are very close to one another.

We will try to transform CFGs into ASMs using local transformations on the support graph of a CFG that do not change its configuration space. Using a combination of these transformations, we will see that it is possible in some cases to obtain a CFG such that its support graph contains one edge (u, v) for each edge (v, u), *i.e.* the graph can be viewed as undirected, so that we have an ASM. We begin by giving the two basic transformations we will use. They do not change the vertex set of a CFG, only its edge set and its initial configuration. Thanks to Proposition 1.5, we will prove that these modifications preserve the configuration space by checking that the first times each vertex can be fired are not changed, and that the CFG remains simple (indeed, in some cases, a modification does not change the first times at which a given vertex can be fired, but allows this vertex to be fired more than once).

The first modification is a basic one, that will be used several times with small variations in the sequel.

Modification 1: Grounding
The grounding modification applied with factor n to a vertex v of a CFG consists in adding n chips on v in the initial configuration, and adding n edges from v to the sink.

Modification 2: Multipliying
The multiplying modification consists in multiplying by an integer factor n the indegree and initial configuration of a given vertex v, without modifying the rest of the CFG. It consists in:

- multiplying by n the initial configuration of v,
- adding $(n-1) \cdot d^+(v)$ edges from v to the sink,
- for each immediate predecessor u of v, adding $(n-1) \cdot d(u,v)$ edges (u,v), and adding as much chips to the initial configuration of u.

The next two lemmas state that these modifications do not change the configuration space of the CFG to which they are applied. The first result being obvious, its proof is ommitted.

Lemma 3.1 *The CFG obtained by applying the Grounding Modification to a CFG C is equivalent to C.*

Lemma 3.2 *The CFG obtained by applying the Multiplying Modification to a CFG C is equivalent to C.*

Proof : Let C' be the CFG obtained from C by applying the Multiplying Modification to a vertex v with a factor n. $L(C')$ is isomorphic to $L(C)$ since:

- The outdegree and the initial configuration of v have been multiplied by n, thus v needs n times more chips to be fired in C' than in C. Since, for each immediate predecessor u of v the number of edges from u to v has also been multiplied by n, the first times at which v can be fired do not change. We can easily see that v can be fired only once in C', since:

$$\begin{aligned} \sigma'(v) + \sum d'(u,v) &= n\sigma(v) + n \sum d(u,v) \\ &< nd^+(v) \\ &< d'^+(v) \end{aligned}$$

- the modifications made to the predecessors of v (the modification of the initial configuration and the edges added towards v) are similar to applying the Grounding Modification to these vertices (with v playing the role of the sink).

□

Now we give the main theorem of this section:

Theorem 3.3 *Algorithm 1 transforms a simple CFG with no cycle into an equivalent ASM in linear time with respect to the size of its configuration space.*

Proof : The goal of the algorithm is to obtain an ASM from a CFG, by adding a *reverse* edge to each edge in the graph (*i.e.* add an edge (v,u) for each edge (u,v)). We recall that, when a CFG is modified, we obtain an equivalent CFG if:

Algorithm 1 Transformation of a simple CFG with no cycle into an equivalent ASM

Input: A simple CFG C with support graph $G = (V, E)$ and initial configuration σ_0, such that G has no cycle

Output: An ASM equivalent to C

Compute $L(C)$;
For each $v \in V \setminus \{\perp\}$, $D[v] \leftarrow \max_{\sigma \in L(C)}(d^+(v) - \sigma(v))$;
Mark $\perp$;
$L \leftarrow$ empty-list;
while *there are unmarked vertices* **do**
    Choose an unmarked vertex v with all successors marked;
    Mark v;
    $L \leftarrow L, v$;

Step 1: while L *is not empty* **do**
    $v \leftarrow$ head(L)
    if $D[v] \leq d(v)$ **then**
        Apply the Multiplying Modification to v with factor $\lceil (d(v) + 1)/D[v] \rceil$;
        For each edge (u, v) added by the Multiplying Modification, add 1 to $D[u]$;
    $L \leftarrow \text{tail}(L)$;

Step 2: for each $v \in V \setminus \{\perp\}$ **do**
    if $\sigma_0(v) + d^-(v) + d(v) \geq 2d^+(v)$ **then**
        Apply the Grounding Modification to v with factor $\lceil (d(v) + d^-(v) + 1)/(2 \cdot d^+(v) - \sigma_0(v)) \rceil$;

Step 3: for each $v \in V \setminus \{\perp\}$ **do**
    for each *edge* (u, v) *in* E **do**
        Add one chip to the initial configuration of v;
        Add one edge (v, u) to E;

- the first times at which each vertex can be fired do not change, and
- each vertex can be fired only once during an execution.

The purpose of the first two steps of the algorithm is to prepare the graph for this reversal of edges. We first show that, at each step of the algorithm, the current CFG is equivalent to C.

Step 1 In this step the indegree of a vertex is increased so that it exceeds by a certain factor its outdegree. This is possible because the graph has no cycle, and the graph has only one sink. In this step, the only modification applied is the Multiplying Modification, therefore by Lemma 3.2, at each step the current CFG is equivalent to C. At the end of this step, we have, for each vertex v and for all configuration σ:

$$d^+(v) - \sigma(v) > d(v).$$

Step 2 The only modification applied in this step is the Grounding Modification, therefore, by Lemma 3.1, at each step the current CFG is equivalent to C. At the end of this step we have, for each vertex v:

$$\sigma_0(v) + d^-(v) + d(v) < 2d^+(v).$$

Step 3 In this step an edge (u, v) is added for each pre-existing edge (v, u). The resulting CFG is equivalent to C because:

- a chip is added to the initial configuration of u for each edge (u, v) added. This is similar to applying the Grounding Modification to u, with v playing the role of the sink. Figure 6 (1-3) illustrates why this is necessary.
- After Step 2 we know that $\sigma_0(v)+d^-(v)+d(v) < 2d^+(v)$. This means that, after we have added $d(v, u)$ edges from u to v, we still have $\sigma_0(v)+d^-(v) < 2d^+(v)$, which means that v can be fired only once in the resulting CFG. Figure 6 (4) illustrates this.
- After Step 1 we have, for all configuration σ: $d^+(v) - \sigma(v) > d(v)$. Therefore adding edges from u to v does not change the first times at which v can be fired: there is no configuration σ in which v cannot be fired, and in which adding edges from u to v allows v to be fired. Figures 7 and 8 illustrate this.

□

Corollary 3.4 *Let C be a simple CFG with support graph $G = (V, E)$ such that G has no cycle. Then C is equivalent to an ASM.*

We have just shown that any simple CFG with no cycle is equivalent to an ASM. This seems like a strong restriction, but in fact a large part of lattices of $L(CFG)$ are the configuration space of a simple CFG with no cycle. Indeed, the next theorem states that this is the case for all distributive lattices, which are an important part of the ULD lattices, and therefore of $L(CFG)$.

Theorem 3.5 *[MPV01] Given any distributive lattice L, there exists a CFG C with no cycle, such that $L(C)$ is isomorphic to L.*

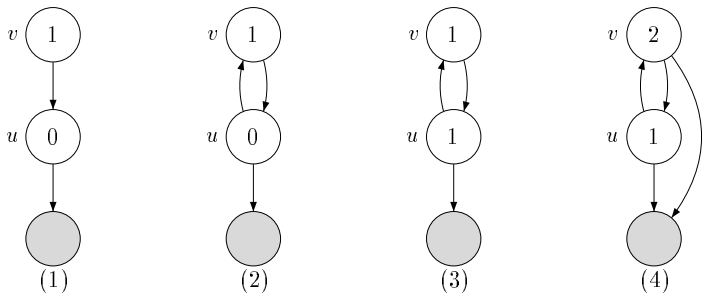

Figure 6: **(2)**: Simply adding an edge (u, v) to the CFG (1) changes the first times at which u can be fired: u can not be fired at all. **(3)**: therefore, we add a chip to the initial configuration of u, and u can be fired after v, as in (1). **(4)**: In (3) v can be fired more than once. Therefore, we apply the Grounding Modification to v (*i.e.* we add a chip to the initial configuration of v and we add one edge $(v, \perp)$) before adding the edge (u, v). The resulting CFG is equivalent to (1).

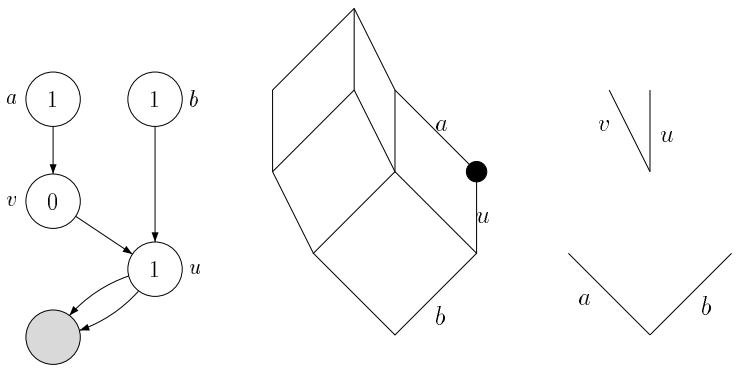

Figure 7: A CFG and its configuration space. Notice that v needs one chip to be fired, and that there exactly is one edge (v, u). After b and u have been fired (the configuration is marked with a dot in the configuration space), only a can be fired.

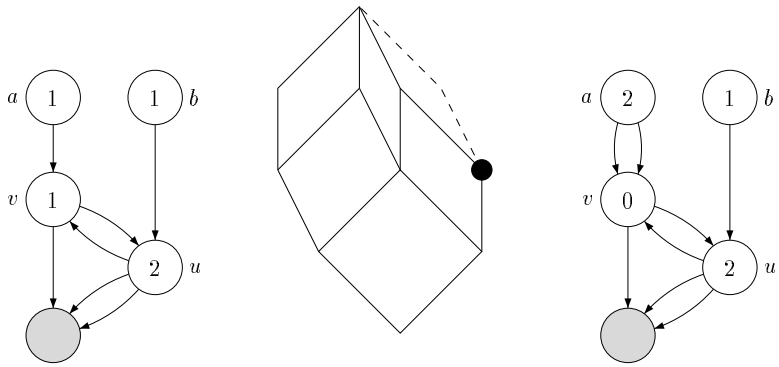

Figure 8: If we want to add one edge (u, v) to the CFG of Figure 7, we have first to add one chip to the initial configuration of u, and to apply the Grounding Modification to v, as seen before. We thus obtain the CFG shown on the left. In this CFG, it is possible to fire v after b and u, and this adds the dotted lines to the configuration space. Therefore, we must apply the Multiplying Modification to v to multiply its indegree by two, before we add the edge (u, v). The resulting CFG (on the right) is equivalent to the original CFG (see Figure 7).

Corollary 3.6 *Any distributive lattice is the configuration space of an ASM.*

The distributive lattices are not the only ones that can be obtained both by CFG and ASM. Indeed, most of the simple CFGs we have considered during our studies have no cycle, and are therefore equivalent to an ASM. Moreover, we give in Figure 9 an example of a CFG which has a cycle in its support graph, but is equivalent to an ASM.

However, we show that not all CFGs are equivalent to ASMs.

Theorem 3.7 $L(ASM) \subsetneq L(CFG)$.

Proof : We know that $L(ASM) \subseteq L(CFG)$. We will show that the lattice L of Figure 10, which is the configuration space of the CFG of Figure 11, cannot be the configuration space of any ASM (*i.e.* a CFG such that there exists an edge (u, v) for each edge (v, u)). This will show that the inclusion is strict. The proof is in two steps: first we show that L can be obtained only by a simple CFG (*i.e.* a CFG such that its configuration space is L *must* have six different vertices, each fired once during an execution, and a sink); we have to show that L can only be obtained by a simple CFG, otherwise we must prove also that L cannot be the configuration space of an ASM with five vertices or less. Then we prove some inequalities on the number of edges between two pairs of vertices, from which we will show that there exist two vertices a and b such that the number of edges (a, b) can never be the same as the number of edges (b, a). Therefore we can never consider that the support graph of the CFG is undirected, and this proves that L cannot be obtained by ASM.

Let us prove that L can be only obtained by a simple CFG. The minimal element has three immediate successors, so there are three different vertices that can be fired in the initial configuration. Let us call them a, b and c. The configuration denoted by a has three immediate successors, so there are three different vertices that can be fired in configuration a. Two of them are b and

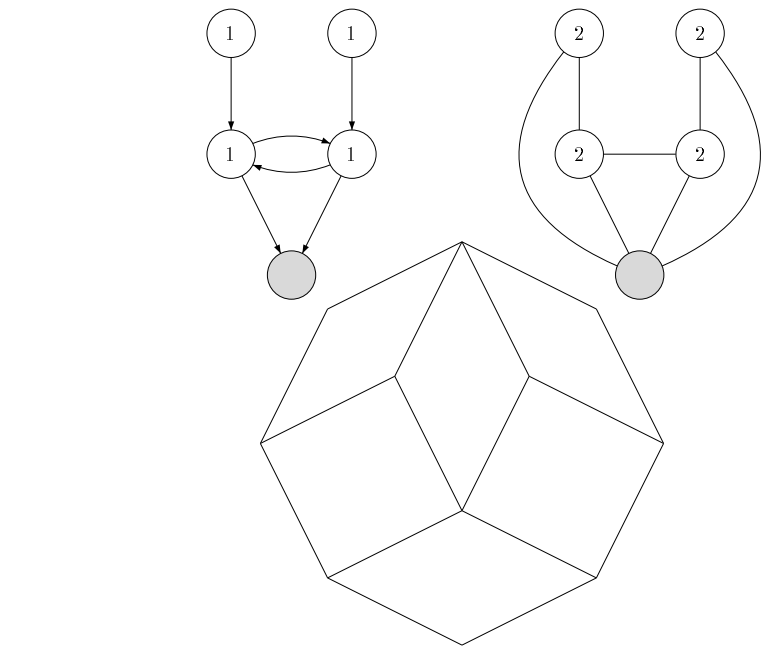

Figure 9: A CFG with a cycle in its support graph that is equivalent to an ASM. We present here the CFG, an equivalent ASM, and their configuration space

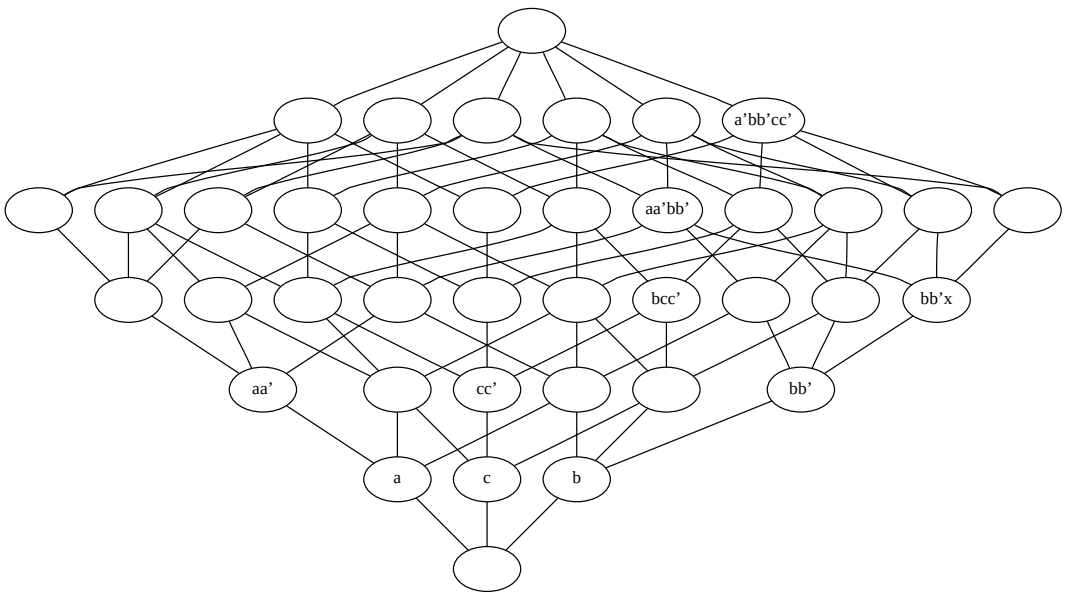

Figure 10: A lattice that can be obtained by CFG but not by ASM

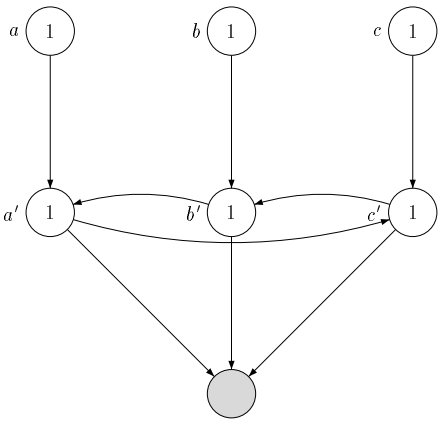

Figure 11: A CFG the configuration space of which is the lattice of Figure 10. Notice the cycle in the support graph.

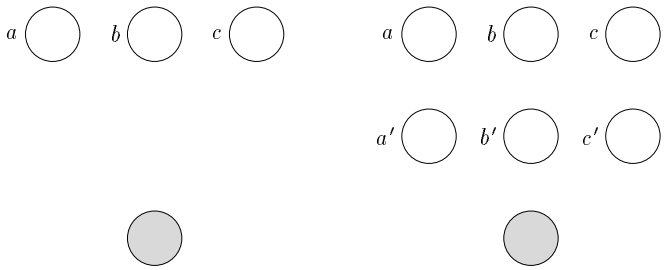

Figure 12: The possible vertex sets of a CFG that generates the lattice of Figure 10

c, and the third one, that we denote by a', is therefore a vertex different from b and c. a' is either a fourth vertex (that can be fired immediatly after a), or a itself (which would mean that a can be fired twice in a row). The same argument holds for b and c. So the vertex set of a CFG C such that $L = L(C)$ is something in-between the two vertex sets of Figure 12.

We will now prove that a', b' and c' are different from a, b and c, and that they are all distinct from each other. First we prove that a' is different from a: the configuration bb' has three immediate successors, therefore there are three different vertices that can be fired in configuration bb'. Two of them are a and c, so the third one (denoted by x, and the firing of which leads to the configuration labelled by $bb'x$) is either a' (in which case a' is distinct from a) or c' (in which case c' is distinct form c). We observe that the join of aa' and bb' (labelled by $aa'bb'$) is greater than $bb'x$, which means that $\{bb'x\} \subseteq \{aa'bb'\}$. From this we conclude that the third vertex is a', and therefore a' is distinct from a. The same reasonning can be applied to b' and c'. Now we show that a', b' and c' are distinct from each other: if a' and b' are the same vertex, the join of aa' and bb' would be reached from aa' by the firing of b, and from bb' by the firing of a. Therefore, $aa'bb'$ would be at distance one of aa' and of bb', whereas it is at distance two. Therefore, a' and b' are two different vertices. With the same argument, we can show that a', b' and c' are all distinct from each other, therefore L can only be obtained by a CFG where six different vertices are fired

during an execution.

Now we prove that L cannot be the configuration space of any ASM. First we claim that $d(b',a') > d(c',a')$. Let n be the number of chips that a' needs to be fired. We know that a' can be fired after b and b'. So we know that $n \leq d(b,a') + d(b',a')$. Suppose our claim is not true, *i.e.* $d(c',a') \geq d(b',a')$. Then we have $n \leq d(c',a') + d(b',a')$, and therefore $n \leq d(c',a') + d(b',a') + d(c,a')$. This means that a' can be fired after c, c' and b'. But at point $b'cc'$, only two vertices can be fired, and they are a and b'. Therefore we obtain a contradiction, and we must have $d(b',a') > d(c',a')$. By similar arguments we can establish that $d(c',b') > d(a',b')$ and $d(a',c') > d(b',c')$. Now if L was the configuration space of an ASM, we would have $d(a',b') = d(b',a')$, $d(a',c') = d(c',a')$ and $d(b',c') = d(c',b')$, which is in contradiction with the inequalities established above. We conclude that L cannot be the configuration space of an ASM. □

By combining Theorem 3.7 and Corollary 3.6, we obtain that $L(ASM)$ is situated strictly between the class of distributive lattices and $L(CFG)$. This shows the complexity of the problems raised by the Chip Firing Game and the Abelian Sandpile Model in lattice theory: $L(CFG)$ and $L(ASM)$ are both between the distributive and ULD lattices, while there is no previously known lattices class satisfying this condition.

Conclusion and perspectives

In this paper, we have studied from the lattice point of view the Chip Firing Game and two closely related models: the Abelian Sandpile Model and the Mutating Chip Firing Game. It was already known that all these models generate lattices, and that these lattices are in the class of Upper Locally Distributive (ULD) lattices. Our goal was to characterize the classes of lattices $L(CFG)$, $L(ASM)$ and $L(MCFG)$ induced by each model (*i.e.* determine, given a ULD lattice, by which model(s) it can be obtained). We have first shown that the mutating CFG and the CFG generate exactly the same set of lattices, by giving a way to transform a MCFG into a classical CFG. This means that, although the MCFG is a generalization of the CFG, it brings no more information in lattice terms.

We know that every ASM is equivalent to a CFG, and we have given an example of a CFG that is not equivalent to any ASM. This implies that $L(ASM) \subsetneq L(CFG)$. Then we have given a sufficient but not necessary condition at which a CFG can be transformed into an ASM, from which it can be shown that the class D of distributive lattices is included in $L(ASM)$. The class of lattices induced by CFG being somewhere between the distributive and the ULD lattices, we obtain that the class of lattices induced by ASM is a new class between the distributive lattices and $L(CFG)$. In other words, we have proved the following relation:

$$D \subsetneq L(ASM) \subsetneq L(CFG) = L(MCFG) \subsetneq \text{ULD}$$

This illustrates the complexity of the problem in lattice theory, because there exists no known lattice class between the distributive and ULD lattices. Therefore, an exact characterization of the two classes $L(ASM)$ and $L(CFG)$ would be a very interesting result in lattice theory.

The CFG and the ASM share the same definition, except that the first is defined on a directed graph, and the latter on an undirected graph. This might seem like a strong difference, and the first idea that comes to mind is that $L(ASM)$ is much smaller than $L(CFG)$. The sufficient but not necessary condition we have given at which a CFG is equivalent to an ASM is that the support graph must contain no directed cycle. This is a strong restriction, but it nonetheless implies that $L(ASM)$ is a very significant part of $L(CFG)$ (it contains more than the distributive lattices class, which is a large part of the ULD class).

The fact that three important models, used in various domains like physics, computer science and social science, all induce strongly structured sets precisely situated between two classical types of lattices, shows the importance of the use of order theory in the context of dynamical models studies. In our case it is even more interesting to notice that the models introduce new classes of lattices which one may study from the order theoretical point of view.

We have seen that the difference between these models does not lie in the fact that the graph is directed or not, but on the existence or not of directed cycles in the graph. Simple CFGs with no cycle in their support graph are very strongly linked with the class $\mathcal{CN}$ of lattices obtained by a sequence of duplications of convex sets. It would be a natural sequel of this work to find out in all lattices of $\mathcal{CN}$ induced by CFG can be obtained by ASM. However, if this is true this does not give the result that $L(ASM) = L(CFG) \cap \mathcal{CN}$, because we have examples of lattices that are in $L(ASM)$, but not in $\mathcal{CN}$.

Finally, lattices also appear in some other kinds of discrete dynamical models defined in the context of tiling theory: for some classes of tiling problems, one can define a local rearrangement of tiles, called *flip*, which transforms a tiling of a given region into another tiling of the same region. In some cases (mainly tilings with dominoes or with three lozenges [Rem99, BL01]), it has been proved that the flip relation gives the distributive lattice structure to the set of all possible tilings of a given region. A work similar to the one done in this paper, comparing the classes of lattices induced by each tiling problem, may lead to interesting result, both in tiling theory and in the study of discrete dynamical models.

References

[Big97] N. Biggs. Algebraic potential theory on graphs. *Bull. London Math. Soc.*, 29:641–682, 1997.

[Big99] N. Biggs. Chip firing and the critical group of a graph. *Journal of Algebraic Combinatorics*, 9:25–45, 1999.

[BL92] A. Björner and L. Lovász. Chip-firing games on directed graphs. *Journal of Algebraic Combinatorics*, 1:304–328, 1992.

[BL01] O. Bodini and M. Latapy. Generalized tilings with height functions. 2001. Submitted.

[BLS91] A. Björner, L. Lovász, and W. Shor. Chip-firing games on graphs. *European Journal of Combinatorics*, 12:283–291, 1991.

[BTW87] P. Bak, C. Tang, and K. Wiesenfeld. Self-organized criticality: an explanation of the $1/f$ noise. *Physics Review Letters*, 59(4):381–384, 1987.

[CR00] Robert Cori and Dominique Rossin. On the sandpile group of a graph. *European Journal of Combinatorics*, 21(4):447–459, 2000.

[DP90] B.A. Davey and H.A. Priestley. *Introduction to Lattices and Orders*. Cambridge university press, 1990.

[DRSV95] Deepak Dhar, P. Ruelle, S. Sen, and D. Verma. Algebraic aspects of sandpile models. *Journal of Physics*, A 28:805–831, 1995.

[Eri93] Kimmo Eriksson. *Strongly Convergent Games and Coxeter Groups*. PhD thesis, Kungl Tekniska Hogskolan, Sweden, 1993.

[Eri96] Kimmo Eriksson. Chip firing games on mutating graphs. *SIAM J. Discrete Math.*, 9:118–128, 1996.

[GMP98] E. Goles, M. Morvan, and H.D. Phan. Lattice structure and convergence of a game of cards. 1998. To appear in Annals of Combinatorics.

[Heu99] Jan van den Heuvel. Algorithmic aspects of a chip firing game. *London School of Economics, CDAM Research Reports.*, 1999. Preprint available at `http://www.cdam.lse.ac.uk/Reports/reports99.html`.

[LP01] Matthieu Latapy and Ha Duong Phan. The lattice structure of chip firing games. *Physica D*, 115:69–82, 2001. Preprint available at `http://www.liafa.jussieu.fr/~latapy/`.

[Mon90] Bernard Monjardet. The consequences of dilworth's work on lattices with unique irreductible decompositions. In K. P. Bogart, R. Freese, and J. Kung, editors, *The Dilworth theorems Selected papers of Robert P. Dilworth*, pages 192–201. Birkhauser, Boston, 1990.

[MPV01] Clémence Magnien, Ha Duong Phan, and Laurent Vuillon. Characterization of lattices induced by (extended) chip firing games. In *Discrete Mathematics and Theoretical Computer Science, Proceedings of the 1-st international conference Discrete Models: Combinatorics, Computation, and Geometry (DM-CCG'01)*, pages 229–244. MIMD, July 2001. Preprint available at `http://www.liafa.jussieu.fr/~magnien/`.

[Rem99] Eric Remila. The lattice structure of the set of domino tilings of a polygon. *LIP - ENS Lyon - Research Report 1999-25*, 1999. To appear in DMTCS, special issue, proceedings of ORDAL'99.