# LATTICES INDUCED BY CHIP FIRING GAMES AND RELATED MODELS

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 sets of configurations reachable from a given initial configuration, called the *configuration space* of a model, and try to determine their main properties. In order to acheive this, 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, *i.e.* determine which orders can be obtained by each model. It is known that these orders all are lattices, a special kind of partially ordered set. Although the Chip Firing Game on a mutating graph is a generalisation of the usual Chip Firing Game, we prove that both models exactly generate the same configuration spaces. We prove also that the class of lattices induced by the Abelian Sandpile Model is strictly included in the class of lattices induced by Chip Firing Game, but contains the class of distributive lattices, a very well known class. This leads to interesting questions concerning orders and lattices theory, since the class of lattices induced by Chip Firing Game is itself an uncharacterised class of lattices, therefore two new classes of lattices naturally appeared in this context.

Résumé. Dans cet article nous étudions trois classes de modèles étudiés en physiques, en informatique et en sciences sociales: le Chip Firing Game, l'Abelian Sandpile Model et le Chip Firing Game sur des graphes changeants. Nous étudions les ensembles de configurations atteignables à partir d'une configuration initiale donnée. Ces ensembles sont appelés espaces des configurations du modèle, et nous essayons de déterminer leurs propriétés principales. Pour cela, nous étudions l'ordre induit sur les configurations par la règle d'évolution.

Cela permet de comparer le pouvoir d'expression de ces modèles, c'est à dire de déterminer quels ordres peuventêtre obtenus par chaque modèle. Il est connu que ces ordres sont tous des treillis, un genre d'ordre particulier. Bien que le Chip Firing Game sur des graphes changeants soit une généralisation du Chip Firing Game habituel, nous prouvons que ces deux modèles génèrent exactement les mêmes espaces de configurations.

Nous prouvons également que la classe de treillis induits par l'Abelian Sandpile Model est incluse strictement dans la classe des treillis induits par Chip Firing Game, mais qu'elle contient la classe des treillis distributifs, une classe très connue. Cela mène à des questions intéressantes concernant la théorie des ordres et des treillis, car la classe des treillis induits par Chip Firing Game est elle-mme une classe de treillis non caractérisée, donc deux nouvelles classes de treillis apparaissent naturellement dans ce contexte.

## 1. Introduction

The Chip Firing Game (CFG) and the Abelian Sandpile Model (ASM) are closely related models studied in physics [BTW87], computer science [GMP98] and social science [Big97, Big99, Heu99]. These three models are variations on the following game: given a graph and a distribution of chips on its vertices, one may select a vertex that contains at least as many chips as its outdegree, and move one chip from this vertex to each of its neighbours along each outgoing edge. Many questions have naturally arisen and given matter for reseach about this game: given a graph and an initial distribution of chips, 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 distribution of chips such that no move is possible ? This has led to the algebraic study of some of these distributions called *recurrent configurations*, [DRSV95, CR00, Big99]. Given a graph, what can be said about all the distributions that can be reached from a given initial distribution [LP01, MPV01] ?

This paper is concerned with this last question. Our purpose is to study the set of reachable distribution of chips, or *configurations* of a game, 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. We will study the structure of such sets, and try to determine the differences between the models with respect to this aspect, *i.e.* given such an order, which is the configuration space of a game, we will try to decide if it is isomorphic to the configuration space of another game. 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 ? We will do so by trying to transform the CFG into an ASM or a MCFG without changing its configuration space.

In Section 2, we give the definitions and known results used in this paper, as well as the exact definition of each model. 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 an transformation, and we will give an example of a CFG that cannot be transformed into an ASM.

## 2. Definitions and known results

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

2.1. **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 $z = x$. The *interval* $[x, y]$ is the set $\{z, z \geq x, z \leq y\}$. 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 $x \wedge y$). $x \vee y$ is the (unique) smallest element greater than both $x$ and $y$. $x \wedge y$ is defined dually. All the lattices considered here are finite, therefore they have a least and a greatest element, respectively denoted by $0_L$ and $1_L$.

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. 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 hypercube. 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 two following laws of distributivity (which are equivalent and imply each other):

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

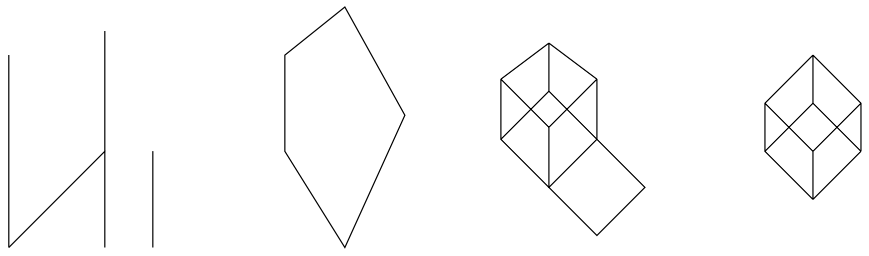


FIGURE 1. From left to right: a poset, a lattice, a distributive lattice and a hypercube

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

A distributive lattice is a lattice that is at the same time upper *and* lower locally distributive, *i.e.* if it is an ULD lattice, and if the interval between any element and the meet of all its lower covers is also a hypercube. In the sequel we will only be concerned with distributive and ULD lattices. Figure 1 shows examples of posets and different lattices.

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

### 2.2. **The different kinds of chip firing games.**

In this section we give the definition 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 generalisations of others. Finally, we give some results about the CFG which will be useful in the paper.

2.2.1. *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.* there might be more than one edge between two vertices, therefore all edge sets are 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 can 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 vertex. We call this process *firing v*. If $\sigma$ is the configuration we start from, and $\sigma'$ is obtained from $\sigma$ by firing $v$, we write $\sigma \xrightarrow{v} \sigma'$, and we call $\sigma$ a *predecessor* of $\sigma'$.

**Note 2.1.** *We consider that the firing rule cannot be applied to a sink (a vertex with no outgoing edges) because firing a sink 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. does not change throughout the game. We give an example of a CFG together with its configuration space in Figure 2 (in this example we have label led each edge $(\sigma, \sigma')$ with the name of the vertex fired to reach $\sigma'$ from $\sigma$). The Abelian Sandpile Model (ASM) [BTW87] is defined over an undirected graph, with a distinguished vertex called the *sink*. The sink can never be fired.

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*, that does not depend on the order in which the vertices were fired. We will only consider games that reach a fixed point. We call these *convergent* games. We call *execution* of a game any sequence of firing which, starting from the initial configuration, reaches the final configuration.

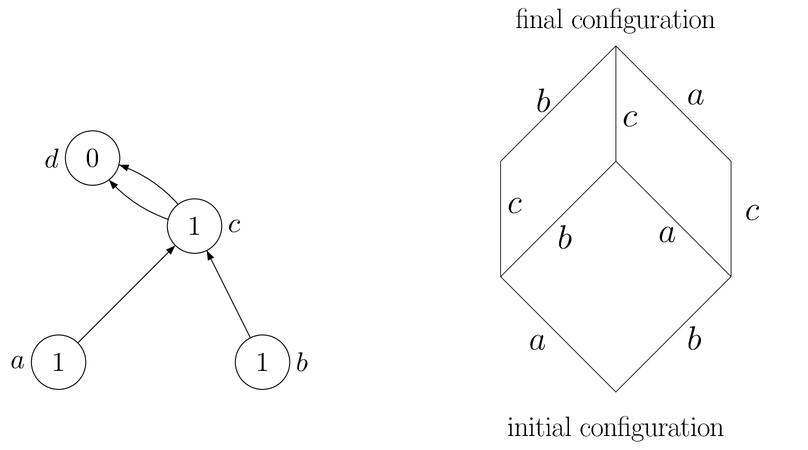


FIGURE 2. A CFG in its initial configuration and its configuration space

We call *configuration space* of a game the set of all 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 an ULD lattice. Given a game $C$, we denote its configuration space by $L(C)$. We will say that two 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. It is possible to guarantee that a CFG is convergent by the presence in the support graph of a sink reachable from all vertices [LP01]. In the ASM, the presence of the sink which can never be fired (although it is not a sink in the undirected graph) 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 them are generalisations 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 is replaced by two opposite directed edges, 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 firing, therefore $L(CFG) \subseteq L(MCFG)$.

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

**Definition 2.2.** *A convergent game is* simple *if, during an execution, each vertex is fired at most once.*

**Theorem 2.3.** [MPV01] *Any convergent CFG is equivalent to a simple CFG.*

All CFGs considered in the sequel will be such that 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 that are *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 [MPV01, LP01]:

**Lemma 2.4.** *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 $\sigma$ as the set of vertices fired to reach $\sigma$ 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 $\sigma$ 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. Moreover, 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]:

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

We give here another way of characterising the configuration space of a CFG with respect to the shot-sets: given a CFG and its vertex set, we can associate to each vertex $v$ the configurations in which $v$ can be fired. Amongst 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 moments* at which $v$ can be fired. For instance, in Figure 2, 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 $a$ and $b$ can be fired from the beginning of any execution. The shot-sets of the minimal configurations in which $c$ can be fired are $\{a\}$ and $\{b\}$, and we say that $c$ can be fired after $a$ *or* after $b$. The knowledge, for all vertices, of the first moments at which they can be fired, is a characterisation of the configuration space of a CFG, as it is stated in the following proposition:

**Proposition 2.5.** *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 $i, i = 1, \ldots, n$, let $X_i$ be the set representing the first moments at which $v_i$ can be fired,* i.e. *$X_i$ contains 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 $\{X_1, \ldots, X_n\}$.*

## 3. Comparison of CFGs and ASMs

In this section, we compare the classes of configuration spaces induced by CFGs and ASMs. Since 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 support graph of the CFG has no cycle. Since we know that any distributive lattice is the configuration space of a CFG without cycles, we obtain as a corollary that the class of lattices induced by ASM contains the distributive lattices. However, not all CFGs are equivalent to some ASM. Since one can easily find examples of lattices in $L(ASM)$ that are not distributive, we obtain the result that $L(ASM)$is strictly between the distributive lattices and the lattices induced by CFG, which is surprising because these classes are very close.

We will try to transform CFGs into ASMs by using local transformations on the support graph of a game that do not change its configuration space. By using a combination of these transformations, 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 is undirected therefore we have an ASM. We begin by giving the two basic transformations we will use on CFGs, which preserve the configuration space of a CFG. These transformations do not change the vertex set of a CFG, only its edge set and its initial configuration. Since we know from Proposition 2.5 that the configuration space is preserved if the first moments at which each vertex can be fired are not changed, we will ensure that these modifications do not change the first moments at which each vertex can be fired. Morover, a modification may not change this, but still allow a vertex to be fired more than once during an execution. We will also ensure that this does not happen.

**Modification 1: Grounding**
The grounding modification applied on a vertex $v$ of a CFG consists in adding one chip to the initial configuration of $v$ and adding one edge from $v$ to the sink.

The purpose of the next modification is to multiply the indegree and the initial configuration of a given vertex by a given integer $n$.

**Modification 2: Multipliying**
The multiplying modification consists in multiplying by an integer factor 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 $v$,
- add $(n-1)d^+(v)$ edges from $v$ to the sink,
- for each immediate predecessor $u$ of $v$, multiplying by $n$ the number of edges $(u, v)$, and
- for each immediate predecessor $u$ of $v$, adding $(n-1)d(u, v)$ chips to the initial configuration of $u$.

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

**Lemma 3.1.** *The CFG obtained by aplying 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 $C$.*

Now we give the main theorem of this Section:

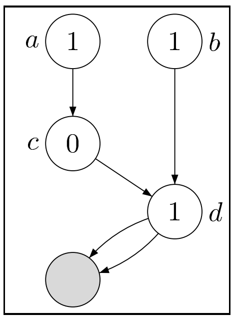
**Theorem 3.3.** *Algorithm 1 transforms a simple CFG with no cycle into an ASM in linear time.*

We will now give an example of the execution of Algorithm 1.

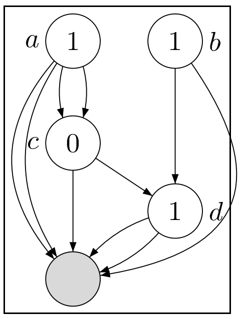
**Algorithm 1** Transformation of a simple CFG graph with no cycle into an equivalent ASM

```
Input:  A simple CFG $C$ with support graph $G = (V, E)$ and initial configuration $\sigma_0$,
        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 $\perp$.
$L = [\,]$.
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
    $v \leftarrow \text{first}(L)$
    if $D[v] \leq d(v)$ then
        Apply the Multiplying Modification to $v$ with a 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 a 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 $E$ ending on $v$ do
        Add one chip to the initial configuration of $v$.
        Add one edge $(v, u)$ to $E$.
```

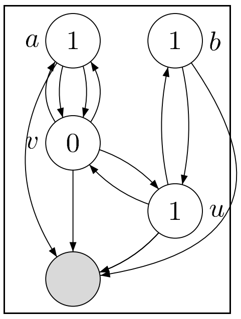
Original CFG. The first loops gives the list of vertices: $L = [d, c, b, a]$.



Step 1. Notice that in the obtained CFG, we have, for any vertex $v$ and any configuration $\sigma$: $d^+(v) - \sigma(v) > d(v)$. Step 2. The CFG is not modified because we have, for any vertex $v$: $\sigma_0(v) + d^-(v) + d(v), 2 \cdot d^+(v)$.



Step 3. The adjuction of the edges $(u, v)$ do not change the configuration space of the CFG because: the modifications applied to $u$ are similar to applying the Grounding Modification to $u$, with $v$ playing the role of the sink. After Step 2 we had $\sigma_0(v) + d^-(v) + d(v) < 2d^+(v)$. Therefore, 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 still be fired only once. After Step 1 we had, for all configuration $\sigma$: $d^+(v) - \sigma(v) > d(v)$. Therefore, there is no configuration $\sigma$ in which $v$ cannot be fired, and in which adding edges from $u$ to $v$ allows $v$ to be fired. The moments at which $v$ can be fired are not modified.



**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.*

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

**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 by ASM. Indeed there are many CFGs without cycle that do not induce distributive lattices. Moreover, some CFGs with a cycle in their support graph can be transformed in ASM. An exemple of these is given in Figure 3.

Finally, it can be proved that the lattice of Figure 4, which is in $L(CFG)$, in not the configuration space of any ASM. Therefore we have the following theorem.

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

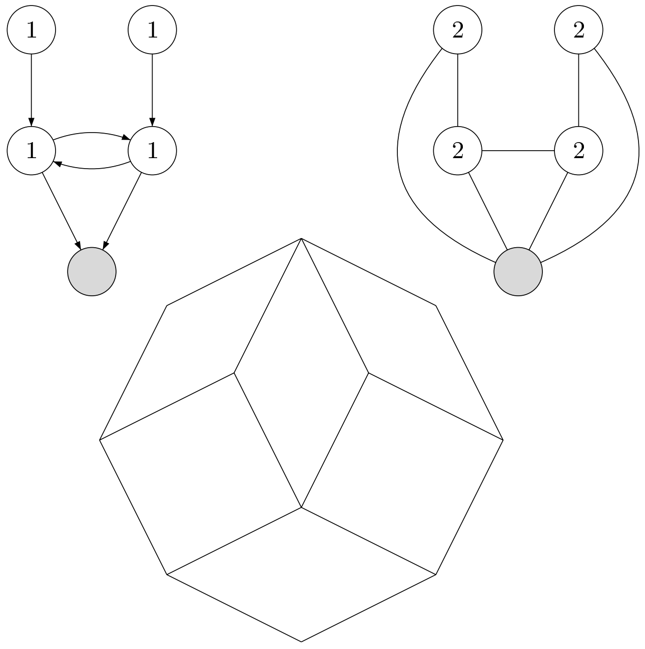


FIGURE 3. A CFG with a cycle in its support graph that is equivalent to an ASM. We give here the CFG, an equivalent ASM, and their configuration space

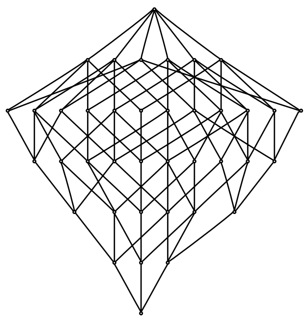


FIGURE 4. A lattice that can be obtained by CFG but not by ASM

By combining Theorem 3.7 and Corollary 3.6, we obtain that $L(ASM)$ is situated strictly between the 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 a closely related model: the Abelian Sandpile Model. It was already known that these models generate lattices, and that these lattices are in the class of Upper Locally Distributive (ULD) lattices. Our goal was to characterise the classes of lattices $L(CFG)$, $L(ASM)$ induced by each model (*i.e.* determine, given a ULD lattice, by which model(s) it can be obtained).

We have given an example of a CFG which is equivalent to no ASM. Since any ASM is equivalent (in lattice terms) to a CFG, 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:

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

The CFG and the ASM share the same definition, except that the first is defined on a directed graph, and the last 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 condition, but in nonetheless shows that $L(ASM)$ is a very significant part of $L(CFG)$ (most of the CFGs we have studied are in fact equivalent to CFGs with no cycle). It also shows that the difference between these models does not reside in the fact that the graph is oriented or not, but on the existence or not of oriented cycles in the graph.

The fact that two important models, used in various domains like physics, computer science and social science, both 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.

## 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.

[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/`.

LIAFA, Univ. Paris 7, 2, place Jusseiu, 75251 Paris Cedex 05, Franc
*E-mail address*: `magnien@liafa.jussieu.fr`