# Combinatorial Structures to Modeling Simple Games and Applications

Xavier Molinero[1,a)]

[1]*Department of Mathematics, Universitat Politècnica de Catalunya (Spain)*
*Avda. Bases de Manresa, 61-73, E-08240 Manresa (Spain).*

[a)]Corresponding author: xavier.molinero@upc.edu
URL: http://lacova.upc.es/~molinero/

**Abstract.** We connect three different topics: combinatorial structures, game theory and chemistry. In particular, we establish the bases to represent some simple games, defined as influence games, and molecules, defined from atoms, by using combinatorial structures. First, we characterize simple games as influence games using influence graphs. It let us to modeling simple games as combinatorial structures (from the viewpoint of structures or graphs). Second, we formally define molecules as combinations of atoms. It let us to modeling molecules as combinatorial structures (from the viewpoint of combinations). It is open to generate such combinatorial structures using some specific techniques as genetic algorithms, (meta-)heuristics algorithms and parallel programming, among others.

**Keywords:** Combinatorial Structures, Generating Simple Games, Generating Influence Games, Generating Molecules.

## INTRODUCTION

In this paper we establish a new connection among different topics. Combinatorial structures let us represent some influence games (depending on the considered graph) and, consequently, let us represent simple game (because any influence game have an associated simple game and viceversa). In a similar way, combinatorial structures let us represent combinations of atoms, that is, molecules.

First section describes combinatorial structures, second section introduces simple and influence games. Afterwards, we give new results about representing simple (as influence games) and molecules (as combinations of atoms). Finally, we make some considerations about future work.

## COMBINATORIAL STRUCTURES

In this section we give the formal definition of admissible combinatorial classes which define combinatorial structures. We also consider the corresponding generating functions to count how many objects there are of each size. Finally, we introduce the so-called admissible operators to define the more useful combinatorial structures.

### Combinatorial Classes and Generating Functions

Most of the material can be found in [23, 8, 9]. However, to make this subsection more self-contained and to fix notation, we will briefly introduce some basic definitions and concepts. We begin with the formal definition of a combinatorial class. A *combinatorial class* is a pair $(\mathcal{A}, |\cdot|_{\mathcal{A}})$ such that $\mathcal{A}$ is a finite or infinite denumerable set and $|\cdot|_{\mathcal{A}} : \mathcal{A} \to \mathbb{N}$ is a *size* function such that, for all $n \geq 0$, $\mathcal{A}_n = \{\alpha \in \mathcal{A} \mid |\alpha|_{\mathcal{A}} = n\}$ is finite.

We use upper case script letters $(\mathcal{A}, \mathcal{B}, \mathcal{C}, \ldots)$ to denote combinatorial classes. Also, we use subscripts under a class' name to denote the subset of objects of that class $\mathcal{A}$ with a given size $n$, for example, $\mathcal{A}_n$. In a similar way, $\mathcal{A}_{>n}$, $\mathcal{A}_{<n}$, $\mathcal{A}_{\geq n}$ and $\mathcal{A}_{\leq n}$ denote the subset of objects of that class $\mathcal{A}$ with size *larger*, *smaller*, *larger or equal* and *smaller or equal* than $n$. The convention of using the corresponding Roman upper case letters for counting generating functions

(see below) will also be used throughout this paper. Moreover, if the class is implied, we will drop the subscript in $|\cdot|_{\mathcal{A}}$. Shall no confusion arise, we will use the same name for the class and for the set of objects belonging to that class.

Typically, complex objects in a given class are composed of smaller units, called *atoms* and generically denoted by $Z$. Atoms are objects of size 1 and the size of an object is the number of atoms it contains. For instance, a string is composed by the concatenation of symbols, where each of these is an atom, and the size of the string is its length or the number of symbols it is composed of. Similarly, a tree is built out of nodes —its atoms— and the size of the tree is its number of nodes. Objects of size 0 are normally denoted by $\epsilon$[1]. Two main types of combinatorial classes can be defined depending on whether the atoms that compose a given object can be distinguished or not. In the former case, we say that the class is *labelled* whereas in the latter we say the class is *unlabelled*. Examples of labelled classes include permutations, Cayley trees, functional graphs and a host of other important combinatorial classes. A valid standard labelling of an object of size $n$ is a bijection from the object's atoms to $[1..n]$, or equivalently, a permutation of size $n$.

As it will become apparent, an efficient solution to the problem of counting, namely, given a specification of a class and a size, compute the number of objects with the given size, is fundamental for our approach to the iteration, unranking and ranking problems. Hence, we turn our attention to so-called *admissible combinatorial classes*. Those are constructed from *admissible operators* over classes that yield new classes, and such that the number of objects of a given size in the new class can be computed from the number of objects of that size or smaller sizes in the constituent classes. We can formalize the notion of admissibility through the notion of *counting generating functions*. The *(counting) generating function of an unlabelled combinatorial class* $\mathcal{A}$ is the ordinary generating function $A(z)$ for the sequence $\{a_n\}_{n\geq 0}$, where $a_n = \#\mathcal{A}_n$ is the number of unlabelled objects in $\mathcal{A}$ of size $n$. The $n$-th coefficient of $A(z) = \sum_{n\geq 0} a_n z^n = \sum_{\alpha\in\mathcal{A}} z^{|\alpha|}$ is $a_n = [z^n]A(z)$. On the other hand, the *(counting) generating function of a labelled combinatorial class* $\mathcal{A}$ is the exponential generating function $A(z)$ for the sequence $\{a_n\}_{n\geq 0}$, where $a_n = \#\mathcal{A}_n$ is the number of labelled objects in $\mathcal{A}$ of size $n$. The $n$-th coefficient of $A(z) = \sum_{n\geq 0} a_n \frac{z^n}{n!} = \sum_{\alpha\in\mathcal{A}} \frac{z^{|\alpha|}}{|\alpha|!}$ is $a_n = n! \cdot [z^n]A(z)$.

## Admissible Operators

We now define *admissible operators* to describe the more usual combinatorial structures. An *operator* (also called *constructor*) $\Psi$ over combinatorial classes $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_k$ is *admissible* if and only if there exists some operator $\Phi$ over the corresponding counting generating functions $A_1(z), \ldots, A_k(z)$ such that $\mathcal{C} = \Psi(\mathcal{A}_1, \ldots, \mathcal{A}_k) \Rightarrow C(z) = \Phi(A_1(z), \ldots, A_k(z))$, where $C(z)$ is the counting generating function of $\mathcal{C}$.

Examples of admissible labelled operators include *disjoint unions* (denoted by '+' or $\mathsf{Union}$), *partitional* (or labelled) *products* ('$\star$' or $\mathsf{Prod}$), *sequences* ($\mathsf{Seq}$), *sets* ($\mathsf{Set}$), *cycles* ($\mathsf{Cycle}$), *substitutions* ('$\circ$' or $\mathsf{Subst}$) and *sequences*, *sets* and *cycles with restricted cardinality*. Analogous operators are admissible in the unlabelled case: *disjoint unions* ('+' or $\mathsf{Union}$), *Cartesian* (or unlabelled) *products* ('$\times$' or $\mathsf{Prod}$), *sequences* ($\mathsf{Seq}$), *powersets* ($\mathsf{PSet}$), *sets* ($\mathsf{Set}$)[2], *cycles* ($\mathsf{Cycle}$), *substitutions* ('$\circ$' or $\mathsf{Subst}$) and *sequences*, *powersets*, *sets* and *cycles with restricted cardinality*. In Table 1 we summarize the relations between these constructions and the corresponding generating functions[3] (see also [10]). We briefly describe now the combinatorial operators mentioned above:

- We take the *disjoint union* (also called *sum*) of two classes $\mathcal{A}$ and $\mathcal{B}$ to represent the union of two disjoint copies, $\mathcal{A}^o$ and $\mathcal{B}^o$, of $\mathcal{A}$ and $\mathcal{B}$. One way to formalize this notion is to introduce two distinct "markers" $\epsilon_{\mathcal{A}}$ and $\epsilon_{\mathcal{B}}$, each of size zero, and define the (disjoint) union $\mathcal{A} + \mathcal{B}$ by $(\mathcal{A} \times \epsilon_{\mathcal{A}}) \cup (\mathcal{B} \times \epsilon_{\mathcal{B}})$. Disjoint union is thus equivalent to a standard union whenever it is applied to disjoint sets. The size of $\gamma \in \mathcal{A} + \mathcal{B}$, i.e., $|\gamma|_{\mathcal{A}+\mathcal{B}}$, is $|\gamma|_{\mathcal{A}}$ if $\gamma \in \mathcal{A}$ or $|\gamma|_{\mathcal{B}}$ if $\gamma \in \mathcal{B}$.
- The *Cartesian product* of $\mathcal{A}$ and $\mathcal{B}$ is composed by the pairs $(\alpha,\beta)$ such that $\alpha \in \mathcal{A}$ and $\beta \in \mathcal{B}$, $\mathcal{A} \times \mathcal{B} = \bigcup_{\alpha\in\mathcal{A}} \bigcup_{\beta\in\mathcal{B}} (\alpha,\beta)$, with $|(\alpha,\beta)|_{\mathcal{A}\times\mathcal{B}} = |\alpha|_{\mathcal{A}} + |\beta|_{\mathcal{B}}$.
- If $\mathcal{A}$ is a class then the *sequence*[4] class is defined as the infinite sum, $\mathsf{Seq}(\mathcal{A}) = \epsilon + \mathcal{A} + (\mathcal{A}\times\mathcal{A}) + (\mathcal{A}\times\mathcal{A}\times\mathcal{A}) + \ldots$ To guarantee the number of sequences of each size will be finite it is necessary to impose that $\mathcal{A}$ contains no object of size 0 (i.e., $a_0 = 0$). From the definition of size for sums and products, the size of a sequence is the sum of the sizes of its components: if $\gamma = (\alpha_1, \ldots, \alpha_k) \in \mathsf{Seq}(\mathcal{A})$ then $|\gamma| = |\alpha_1| + \ldots + |\alpha_k|$.

[1]Some authors use $\lambda$ to denote an object of size 0, called the empty object. The same symbol $(\epsilon,\lambda)$ is often used to denote the class which does only contain the empty object. Likewise, $Z$ often denotes an atomic class, shall no confusion arise.

[2]Also called multisets or bags, for they allow repetitions.

[3]Substitution $\mathcal{A} \circ \mathcal{B}$ and the composition $(A \circ B)(z)$ are also denoted by $\mathcal{A}[\mathcal{B}]$ and $A(B(z))$, respectively [5, 11].

[4]It is analogous to the *Kleene star operator* $\mathcal{A}^*$.

**TABLE 1.** Combinatorial operators and generating functions ('EGF' and 'OGF' denote exponential and ordinary generating functions, respectively).

| Class | EGF | OGF |
|---|---|---|
| $\mathsf{Union}(\mathcal{A},\mathcal{B}) = \mathcal{A}+\mathcal{B}$ | $A(z)+B(z)$ | $A(z)+B(z)$ |
| $\mathsf{Prod}(\mathcal{A},\mathcal{B}) = \mathcal{A}\star\mathcal{B}$ | $A(z)\,B(z)$ | — |
| $\mathsf{Prod}(\mathcal{A},\mathcal{B}) = \mathcal{A}\times\mathcal{B}$ | — | $A(z)\,B(z)$ |
| $\mathsf{Seq}(\mathcal{A})$ | $\frac{1}{1-A(z)}$ | $\frac{1}{1-A(z)}$ |
| $\mathsf{PSet}(\mathcal{A})$ | — | $\exp\left(\sum_{n>0}(-1)^{n-1}\frac{A(z^n)}{n}\right)$ |
| $\mathsf{Set}(\mathcal{A})$ | $\exp(A(z))$ | $\exp\left(\sum_{n>0}\frac{A(z^n)}{n}\right)$ |
| $\mathsf{Cycle}(\mathcal{A})$ | $\log\left(\frac{1}{1-A(z)}\right)$ | $\sum_{n>0}\frac{\phi(n)}{n}\log\left(\frac{1}{1-A(z^n)}\right)$ |
| $\mathsf{Subst}(\mathcal{A},\mathcal{B}) = \mathcal{A}\circ\mathcal{B}$ | $(A\circ B)(z)$ | $(A\circ B)(z)$ |

- The *powerset* class of $\mathcal{A}$ is defined as the class consisting of all finite subsets of $\mathcal{A}$-objects without repeated components. The following isomorphism is often used: $\mathsf{PSet}(\mathcal{A}) = \prod_{\alpha\in\mathcal{A}}(\epsilon+\alpha)$. Powersets do not require that $a_0 = 0$, but we assume it to be the use.
- *Sets* are like powersets except that the repetitions of components are allowed. Hence $\mathsf{Set}(\mathcal{A}) = \prod_{\alpha\in\mathcal{A}}\mathsf{Seq}(\alpha)$. As in the case of sequences, $\mathcal{A}$ must not contain objects of size 0.
- *Cycles* are just sequences defined up to cyclic permutations: $\mathsf{Cycle}(\mathcal{A}) = \mathsf{Seq}(\mathcal{A})/\!\sim$, with $\sim$ the equivalence relation between sequences defined by $(\alpha_1,\ldots,\alpha_r)\sim(\beta_1,\ldots,\beta_r)$ if and only if there exists some cyclic permutation $\sigma$ of $[1..n]$ such that for all $j, \beta_{\sigma(j)} = \alpha_j$; in other words, for some $d, \beta_j = \alpha_{1+(j+d)\mathbf{mod}n}$.
- The *substitution* of $\mathcal{B}$ into $\mathcal{A}$ (or *composition* of $\mathcal{A}$ and $\mathcal{B}$) is defined by $\mathcal{A}\circ\mathcal{B} = \bigcup_{k\geq 0}\mathcal{A}_k\times\mathsf{Seq}(\mathcal{B}, card=k)$, where $\mathsf{Seq}(\mathcal{B}, card=k)$ is the class of $k$-tuples of $\mathcal{B}$ objects. Objects of $\mathcal{A}\circ\mathcal{B}$ may thus be viewed as obtained by selecting in all possible ways an $\alpha$ object of $\mathcal{A}$ and substituting for each of its atoms by an arbitrary object $\beta$ of $\mathcal{B}$. Formally, $C = \{(\alpha,\beta_1,\ldots,\beta_{|\alpha|}) \mid \alpha\in\mathcal{A}$ *and*
  $\beta_1,\ldots,\beta_{|\alpha|}\in\mathcal{B}\ \}$, and $|(\alpha,\beta_1,\ldots,\beta_{|\alpha|})| = |\beta_1|+\cdots+|\beta_{|\alpha|}|$.
  $\mathcal{A}\circ\mathcal{B}$ is well defined if and only if either $b_0 = 0$ or $|\mathcal{A}| < +\infty$; however, in general, we assume that $a_0 = b_0 = 0$.

Labelled operators like unions, sequences, sets and cycles are defined as their unlabelled counterparts. But products and substitutions have some differences:

- Given two labelled objects $\alpha$ and $\beta$ of sizes $j$ and $n-j$, respectively, their partitional product is a set of $\binom{n}{j}$ labelled objects of size $n$ which result from the $\binom{n}{j}$ consistent relabellings of the pair $(\alpha,\beta)$ so that each atom of the pair has a distinct label in the range $[1..n]$ while respecting the order of the original labels of $\alpha$ and $\beta$. For instance, if $\alpha$=132 and $\beta$=21 (these two objects belong to the labelled class $\mathsf{Seq}(Z)$, i.e., permutations) then[5] $\alpha\star\beta = \{13254, 14253, 14352,\ldots,35421\}$. The labelled product of the labelled classes $\mathcal{A}$ and $\mathcal{B}$ is defined then $\mathcal{A}\star\mathcal{B} = \bigcup_{\alpha\in\mathcal{A}}\bigcup_{\beta\in\mathcal{B}}\alpha\star\beta$, with $|\alpha\star\beta|_{\mathcal{A}\star\mathcal{B}} = |\alpha|_{\mathcal{A}} + |\beta|_{\mathcal{B}}$.
- Labelled substitution is defined by $\mathcal{A}\circ\mathcal{B} = \bigcup\limits_{k>0}\mathcal{A}_k\times\mathsf{Set}(\mathcal{B}, card=k)$. The component with the smallest label in the object of $\mathsf{Set}(\mathcal{B}, card=k)$ (the *leader*) substitutes the atom with the smallest label of the $\mathcal{A}$ object, and the same procedure is repeated with the remaining components of the object of $\mathsf{Set}(\mathcal{B}, card=k)$ and the remaining atoms of the $\mathcal{A}$ object until all atoms have been replaced by components of the object in $\mathsf{Set}(\mathcal{B}, card=k)$.

Besides these operators, other operators will also be considered: *boxed products* ($^{\square}\star$), *pointings* ($\vartheta$), *diagonals* ($\Delta$), etc. These operators represent combinatorial structures that depend on the considered isomorphism and ordering to define them [19, 20, 15, 14]. It is known [15, 14] that such combinatorial structures with size $n$ can be generated with worst-case time complexity $O(n^2)$ arithmetic operations for the so-called lexicographic ordering, and $O(n\log(n))$ arithmetic operations for the so-called boustrophedonic ordering.

[5]We are making a slight abuse of notation here: we have refrained from writing $\alpha = (Z_1,(Z_3,(Z_2,\epsilon))), \beta = (Z_2,(Z_1,\epsilon))$, etc, in favor of the usual and more readable form $\alpha = 132$ and $\beta = 21$.

## SIMPLE GAMES (AS INFLUENCE GAMES)

In this section we consider another topic with respect to the previous section. Now we introduce simple games and influence games [17, 18, 2, 26]. Firstly, we introduce simple games. A *simple game* $\Gamma$ is given by a tuple $(N, \mathcal{W})$ where $N$ is a finite set of players and $\mathcal{W}$ is a monotonic family of subsets of $N$.

In the context of simple games, the subsets of $N$ are called *coalitions*, $N$ is the *grand coalition* and $X \in \mathcal{W}$ is a *winning coalition* (a *successful team*). Any subset of $N$ which is not a winning coalition is called a *losing coalition* (an *unsuccessful team*). A *minimal winning coalition* is a winning coalition $X$ that does not properly contain any winning coalition. That is, removing any player from $X$ results in a losing coalition. A *maximal losing coalition* is a losing coalition $X$ that is not properly contained in any other losing coalition. That is, adding any player to $X$ results in a winning coalition. $\mathcal{W}$, $\mathcal{L}$, $\mathcal{W}^m$ and $\mathcal{L}^M$ usually denote the sets of winning, losing, minimal winning and maximal losing coalitions, respectively. Any of those set families determine uniquely the game and constitute the usual forms of representation for simple games [25], although the sizes of those representations are not, in general, polynomial in the number of players.

On the other hand, before introducing formally the family of influence games we need to define a family of labeled graphs and a process of spread of influence based on the *linear threshold model* [21, 12, 22]. We use standard graph notation following [3]. As usual, given a finite set $U$, $\mathcal{P}(U)$ denotes its power set, and $|U|$ its cardinality. For any $0 \leq k \leq |U|$, $\mathcal{P}_k(U)$ denotes the subsets of $U$ with exactly $k$ elements. For a given graph $G = (V, E)$ we assume that $n = |V|$ and $m = |E|$. Also $G[S]$ denotes the subgraph induced by $S \subseteq V$ and, for a vertex $u \in V$, $N(u) = \{v \in V \mid (u, v) \in E\}$.

An *influence graph* is a tuple $(G, w, f)$, where $G = (V, E)$ is a weighted, labeled and directed graph (without loops). As usual $V$ is the set of vertices or agents, $E$ is the set of edges and $w : E \to \mathbb{N}$ is a *weight function*. Finally, $f : V \to \mathbb{N}$ is a labeling function that quantifies how influenceable each agent is. An agent $i \in V$ has *influence* over another agent $j \in V$ if and only if $(i, j) \in E$. We also consider the family of *unweighted influence graphs* $(G, f)$ in which every edge has weight 1.

Given an influence graph $(G, w, f)$ and an initial activation set $X \subseteq V$, the *spread of influence* of $X$ is the set $F(X) \subseteq V$ which is formed by the agents activated through an iterative process. We use $F_k(X)$ to denote the set of nodes activated at step $k$. Initially, at step 0, only the vertices in $X$ are activated, that is $F_0(X) = X$. The set of vertices activated at step $i > 0$ consists of all vertices for which the total weight of the edges connecting them to nodes in $F_{i-1}(X)$ meets or exceeds their labels, i.e., $F_i(X) = F_{i-1}(X) \cup \left\{v \in V \mid \sum_{\{u \in F_{i-1}(X) | (u,v) \in E\}} w((u, v)) \geq f(v)\right\}$. The process stops when no additional activation occurs and the final set of activated nodes is denoted by $F(X)$. As the number of vertices is finite, for any $i > n$, $F_i(X) = F_{i-1}(X)$. Thus, $F(X) = F_n(X)$ and we have the following result.

**Theorem 1 ([18])** *Given an influence graph $(G, w, f)$ and a set of vertices $X$, the set $F(X)$ can be computed in polynomial time.*

To finish the introduction of this Section we define influence games. An *influence game* is given by a tuple $(G, w, f, q, N)$ where $(G, w, f)$ is an influence graph, $q$ is an integer *quota*, $0 \leq q \leq |V| + 1$, and $N \subseteq V$ is the *set of players*. $X \subseteq N$ is a *successful team* if and only if $|F(X)| \geq q$, otherwise $X$ is an *unsuccessful team*.

Note that all results and definitions stated in influence games can be done for directed or undirected graphs.

## Representing Simple Games

We connect different topics –combinatorial structures, simple games and graphs with spread of influence (to define influence games)– in order to introduce a new way to represent simple games. This fact is based in the following result.

**Theorem 2 ([18])** *An influence game has associated a simple game and, viceversa, a simple game has associated an influence game.*

See the details how to construct an influence game from a simple game, and how to construct a simple game from an influence game, in [18]. Note that there exists a construction such that the influence game which define a simple game just use unweighted influence graphs. This construction just use *unweighted graphs* with unions and products. Thus, a simple game can be represented by combinatorial structures with elementary admissible operators (unions and products).

The sketch about how to generate a simple game with just combinatorial structures is described in the following five steps:

- First Step: To consider a simple game $\Gamma$.
- Second Step: To generate the corresponding influence game $\mathcal{I}_\Gamma$ as it is described in [18].
- Third Step: To characterize the influence graph, with labeled nodes, associated to the influence game $\mathcal{I}_\Gamma$. Let "$\mathcal{G}(\mathcal{I}_\Gamma)$" be such influence graph.
- Fourth Step: To define a combinatorial structure with admissible operators which corresponds to "$\mathcal{G}(\mathcal{I}_\Gamma)$". Let $\mathcal{A}[\mathcal{G}(\mathcal{I}_\Gamma)]$ be such combinatorial structure.
- Fifth Step: To consider the spread of influence over the combinatorial structure $\mathcal{A}[\mathcal{G}(\mathcal{I}_\Gamma)]$.

Each step can be computed in polynomial time whether the simple game $\Gamma$ is given by either $(N, \mathcal{W})$ or $(\mathcal{W}^m)$ (see Theorem 1 in [18]).

**Theorem 3** *It is possible to construct a simple game $\Gamma$ as a combinatorial structure. Furthermore, when $\Gamma$ is given by either $(N, \mathcal{W})$ or $(\mathcal{W}^m)$ such construction can be obtained in polynomial time.*

## MOLECULES

Combinatorial structures can also be applied to other topics like chemistry. For instance, combinatorial structures let us generate molecules from *atoms* [6, 7]. A *molecule* is an union of sets of specific cardinality of atoms, i.e., $\mathit{Molecule} := \mathsf{Union}(\mathsf{Set}(\mathit{Atom}_1, \mathit{card} = c_1), \ldots, \mathsf{Set}(\mathit{Atom}_n, \mathit{card} = c_n))$.

In the same vein, we can define a Chemical-Compound. A *chemical-compund* is a set of molecules, i.e., $\mathit{Chemical\text{-}Compound} := \mathsf{Set}(\mathsf{Union}(\mathit{Molecule}_1, \ldots, \mathit{Molecule}_n)$, where, $\forall i \in \{1, \ldots, n\}$, $\mathit{Molecule}_i := \mathsf{Union}(\mathsf{Set}(\mathit{Atom}_{i(1)}, \mathit{card} = c_{i(1)}), \ldots, \mathsf{Set}(\mathit{Atom}_{i(k_i)}, \mathit{card} = c_{i(k_i)}))$.

## Representing Molecules

From previous definitions (molecule and chemical-compund) it is clear that combinatorial structures let us represent molecules in general. For instance, water are molecules of $H_2O$, i.e., $\mathit{Water} := \mathsf{Set}(\mathit{Wat\text{-}Mol})$, where $\mathit{Wat\text{-}Mol} := \mathsf{Union}(\mathsf{Set}(H, \mathit{card} = 2), O)$ or $\mathit{Wat\text{-}Mol} := \mathsf{Union}(H, H, O)$.

On the other and, acyclic alkyne are molecules of $C_nH_{2n-2}$, i.e., $\mathit{Acyclic\text{-}Alkyne} := \mathsf{Set}(\mathit{Acy\text{-}Alk\text{-}Mol})$, where $\mathit{Acy\text{-}Alk\text{-}Mol} := \mathsf{Union}(\mathsf{Set}(C, \mathit{card} = n), \mathsf{Set}(H, \mathit{card} = 2n - 2))$, being $n$ an positive integer.

## Future Work

There is still much work to do with simple games and molecules v.s. combinatorial structures. From combinatorial structures we can study how many simple games or molecules can be generated with a specific *structure* and *size* (number of nodes). Some algorithms related with this problem are the so-called ranking, unranking, iteration or random generation of combinatorial structures [19, 20].

We can also study the limitations of these representations.

It is still open how to apply concepts over simple games and molecules to combinatorial structures: properties of players or atoms, coalitions or compounds, subclasses, etc. Reciprocally, it is also interesting to study how to apply concepts over combinatorial structures (classes) to simple games or molecules: properties over classes, operators, etc.

We ask whether it is possible to define a new concept: *Labeled* and *Unlabeled* simple game or molecule.

The algorithms to generate combinatorial structures (simple games or molecules) should be analyzed to use genetic algorithms [16, 28], (meta-)heuristic algorithms [1, 27, 24, 13] or parallel programming [4], among some algorithmic techniques.

Finally, combinatorial structures can also be applied to another different topics [29, 23].

## ACKNOWLEDGMENTS

This research project was partially supported by funds from the Spanish Ministry of Economy and Competitiveness (MINECO) and from the European Union (FEDER funds) under grant MTM2015-66818-P (MINECO/FEDER).

## REFERENCES

[1] M. Akhmedov, I. Kwee, and R. Montemanni. A divide and conquer matheuristic algorithm for the prize-collecting steiner tree problem. *Computers & Operations Research*, 70:18–25, 2016.

[2] E. Altman, T. Boulogne, R. El-Azouzi, T. Jimnez, and L. Wynter. A survey on networking games in telecommunications. *Computers & Operations Research*, 33(2):286–311, 2006. Game Theory: Numerical Methods and ApplicationsGame Theory: Numerical Methods and Applications.

[3] B. Bollobás. *Modern graph theory*, volume 184 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, NY, 1998.

[4] T.H. Cormen, C.E. Leiserson, and R.L. Rivest. *Introduction to Algorithms*. The MIT Press, The Massachusetts Institute of Technology, 1990.

[5] P. Flajolet. Mathematical methods in the analysis of algorithms and data structures. *Trends in Theoretical Computer Science*, pages 225–304, 1988.

[6] P. Flajolet and B. Salvy. Computer algebra libraries for combinatorial structures. *J. Symbolic Computation*, 20:653–671, 1995.

[7] P. Flajolet, B. Salvy, and P. Zimmermann. Lambda-upsilon-omega: The 1989 cookbook. Technical Report 1073, INRIA, 1989.

[8] P. Flajolet and R. Sedgewick. The average case analysis of algorithms: Counting and generating functions. Technical Report 1888, INRIA, 1993.

[9] P. Flajolet and J.S. Vitter. Average-case Analysis of Algorithms and Data Structures. In J. Van Leeuwen, editor, *Handbook of Theoretical Computer Science*, chapter 9. North-Holland, 1990.

[10] P. Flajolet, P. Zimmermann, and B. Van Cutsem. A calculus for the random generation of combinatorial structures. *Theoretical Computer Science*, 132(1):1–35, 1994.

[11] L.A. Goldberg and D.M. Jackson. *Combinatorial Enumeration*. John Wiley & Sons, 1983.

[12] M. Granovetter. Threshold models of collective behavior. *American Journal of Sociology*, 83(6):1420–1443, 1978.

[13] M.A. El-Sharkawi K.Y. Lee. *Modern Heuristic Optimization Techniques: Theory and Applications to Power Systems*. Wiley-IEEE Press.

[14] C. Martínez and X. Molinero. A generic approach for the unranking of labeled combinatorial classes. *Random Structures & Algorithms*, 19(3-4):472–497, 2001.

[15] C. Martínez and X. Molinero. Efficient iteration in admissible combinatorial classes. *Theoretical Computer Science*, 346(2–3):388–417, November 2005.

[16] M. Mitchell. *An Introduction to Genetic Algorithms (Complex Adaptive Systems)*. The MIT Press.

[17] X. Molinero, M. Olsen, and M. Serna. On the complexity of exchanging. *Information Processing Letters*, 116(6):437–441, 2016.

[18] X. Molinero, F. Riquelme, and M. J. Serna. Cooperation through social influence. *European Journal of Operation Research*, 242(3):960–974, May 2015.

[19] X. Molinero and J. Vives. Unranking algorithms for combinatorial structures. *International Journal of Applied Mathematics and Informatics*, 9:110–115, 2015.

[20] X. Molinero and J. Vives. Unranking algorithms for combinatorial structures. In M.V. Shitikova N.E. Mastorakis, I.Rudas and Y.S. Shmaliy, editors, *Proceedings of the International Conference on Applied Mathematics and Computational Methods in Engineering (AMCME 2015)*, pages 98–101, 2015.

[21] A.K. Nandi and H.R. Medal. Methods for removing links in a network to minimize the spread of infections. *Computers & Operations Research*, 69:10–24, 2016.

[22] T. Schelling. *Micromotives and macrobehavior*. Fels lectures on public policy analysis. W. W. Norton & Company, New York, NY, 1978.

[23] R. Sedgewick and P. Flajolet. *An Introduction to the Analysis of Algorithms*. Addison-Wesley, Reading, MA, 1996.

[24] E. Talbi. *Metaheuristics: From Design to Implementation*. Wiley.

[25] A. Taylor and W. Zwicker. *Simple games: Desirability relations, trading, pseudoweightings*. Princeton University Press, Princeton, NJ, 1999.

[26] A.D. Taylor and W.S. Zwicker. *Simple games: desirability relations, trading, and pseudoweightings*. Princeton University Press, New Jersey, USA, 1999.

[27] P. Vasant. *Meta-Heuristics Optimization Algorithms in Engineering, Business, Economics, and Finance*. IGI Global.

[28] R. Keller W. Banzhaf, P. Nordin and F. Francone. *Genetic Programming An Introduction*. San Francisco, CA: Morgan Kaufmann, 1998.

[29] S. Yazdanfar and M.Sina. Providing a real-time scheduling algorithm for multi-processor systems using the modified colonial competition algorithm. *Journal of Current Research in Science*, 3(5):8–17, 2015.