Accretive Rules in Cayley P System

Jean-Louis Giavitto & Olivier Michel

LaMI u.m.r. 8042 du CNRS, Université d'Evry Val d'Essone – GENOPOLE
Tour Evry2, 523 place des terasses de l'agora.
91000 Evry, France.
Tel: +33 1.60.87.39.04
[giavitto,michel]@lami.univ-evry.fr

— submitted to WMC'02 —

Abstract

During a discussion taking place at WMC'01, G. Paun asked the question of what can be computed only by moving symbols between membranes. In this paper we provide some elements of the answer, in a setting similar of tissue P systems, where the set of membranes is organized as a Cayley graph and using a very simple propagation process characterizing accretive growth. Our main result is to characterize the final configuration as a least fixed point and to establish two series of approximations that converge to it. All the notions introduced (Cayley graph of membranes, accretive rule and iteration) have been implemented in the `MGS` programming language and the two series can be computed in Pressburger arithmetics using the `omega` calculator in the case of Abelian Cayley graphs.

1 Introduction

P systems are new distributed parallel computing models based on the notion of a membrane structure [Pau99, Pau01]. A membrane structure is a nesting of cells represented, e.g., by a Venn diagram without intersection and with a unique superset: the skin. Objects are placed in the regions defined by the membranes and evolve following various transformations subject to some conditions: an object can evolve into another object, can pass trough a membrane or dissolve its enclosing membrane, etc. The computation is finished for instance when no object can further evolve.

During a discussion taking place at WMC'01, Gheorghe Paun asked the question of what can be computed *only by moving objects* between membranes. In its initial presentation, the P system formalism describes the topology of the membranes as a set of nested regions. Thus, the objects can be viewed as moving between the nodes of the membrane's inclusion tree. Our question can then be rephrased as: "*Starting from a set S of symbols located in a tree, what are the nodes F of the tree that are occupied by symbols after moving them following some rules $\mathcal{R}$?*" In the process of making this question more precise and amenable to some answers, we made some simplifying assumptions and some generalizations.

1. In a first approach, we are only interested by the presence or the absence of a symbol in a membrane. Then, we consider in this paper only one kind of symbol and we can simplify the representation of the content of a membrane as a boolean: *false* means that there is no symbol in the membrane and *true* that there is some.

2. Considering only the presence/absence of symbols on a finite tree is too restrictive: there is only a finite set of observable states for the system. Several natural extensions of the initial

formalism consider P systems working on graphs which are not trees [PSY01, MVPPRP01]. In this work, we follow the same path and we consider *Cayley graphs*. In this way, we can have a finite presentation for unbounded graphs of various shapes. Cayley graphs include rings, grids, regular tiling of the plane, etc. They include also n-ary trees (as the Cayley graph of free groups with n generators), which covers the case of our initial question.

3. Starting from an *arbitrary* set S of nodes occupied by symbols in a Cayley graph $\boldsymbol{G}$, there are few chances to characterize the final set F of nodes occupied after the moves allowed by a set of rules $\mathcal{R}$. It is more interesting to characterize some canonical family of starting sets S_i and try to see if the resulting set F_i can be characterized in the same way. In this study, we consider the family of *cosets* in $\boldsymbol{G}$.

4. A rule $r \in \mathcal{R}$ specifies if a symbol in a membrane must move to another membrane. Accordingly, to the assumption 1, we further simplify our problem by considering that the condition of the activation of r must take into account only the presence or the absence of symbols in the membrane and its neighbors. We name such rule: *accretive rule*.

Some justifications for this framework are given in the next section. We will show that it constitutes a tractable simplification of a more general process easily programmable in the `MGS` language [GM02]. Moreover, the problem in this form is closely linked with the problem of computing the definition domain of a systolic function [KMW67, SQ93] or the problem of computing the extension of a data-field [LC94, Gia00].

The rest of this paper is organized as follows. The next section introduces the concepts of Cayley graphs and group-based data fields in the context of the programming language `MGS`. Then we give an example to clarify the notions. Section 4 defines the notions of accretive rules, a special kind of transport rules, and explains the use of cosets to describe initial configurations. Section 5 gives a formal account of the problem and presents some results: we characterize the final set F as a least fixed point and we construct two series of approximating sets $(\mathrm{D}_n)_{n\in\mathbb{N}}$ and $(\mathrm{E}_n)_{n\in\mathbb{N}}$ such that $\mathrm{D}_n \subseteq F \subseteq \mathrm{E}_n$ and $\lim_{n\to\infty} \mathrm{D}_n = \lim_{n\to\infty} \mathrm{E}_n = F$. Finally, we show how the sets D_n and E_n can be effectively computed using Pressburger arithmetics and the `omega` calculator in the case of free Abelian Cayley graphs. Comparison with previous works and links with similar problem are given in the conclusion.

2 Cayley Shaped Membranes in `MGS`

In this section, we present the notions needed to understand the `MGS` coding of the previous computation process. `MGS` is a declarative programming language aimed at the representation and manipulation of local transformations of entities structured by *abstract topologies* [GM01c, GM02]. A set of entities organized by an abstract topology is called a *topological collection*. Topological means here that each collection type defines a neighborhood relation specifying both the notion of *locality*, *path* and *sub-collection*. A path is a finite sequence of elements e_i where e_{i+1} is a neighbor of e_i. A sub-collection B of a collection A is a subset of elements of A defined by some path and inheriting its organization from A. The *global transformation* of a topological collection C consists in the parallel application of a set of *local transformations*. A local transformation is specified by a rewriting rule r that specifies the change of a sub-collection. The application of a rewrite rule $r = \beta \Rightarrow f(\beta, ...)$ to a collection A:

1. selects a sub-collection B of A whose elements match the *path pattern* β,
2. computes a new collection C as a function f of B and its neighbors,
3. and specifies the insertion of C in place of B into A.

The collection types can range in `MGS` from totally unstructured with sets and multisets to more structured with sequences and "group-based data fields". The ability to handle in the same framework the rewriting of multisets and sequences (i.e. strings), makes `MGS` particularly suitable to implement directly several variations of P systems [GM01b].

Group-Based Data Field

Group-based data fields (GBF in short) are used to define organizations with *uniform* neighborhood. A GBF is an extension of the notion of array, where the elements are indexed by the elements of a group, called the *shape* of the GBF [GMS96, GM01a]. The elements of the group are called the *positions* of the GBF. For example:

$$\texttt{gbf}\ \textit{Grid2} \quad = \quad \langle\, \texttt{north}, \texttt{east} \,\rangle$$

defines a GBF collection type called *Grid2*, corresponding to the Von Neuman neighborhood in a classical array (a cell above, below, left or right – not diagonal). The two names `north` and `east` refer to the directions that can be followed to reach the neighbors of an element. These directions are the *generators* of the underlying group structure. The right hand side (r.h.s.) of the GBF definition gives a finite presentation of the group structure.

The list of the generators can be completed by giving equations that constraint the displacements in the shape:

$$\texttt{gbf}\ \textit{Hexagon} \quad = \quad \langle\, \texttt{east}, \texttt{north}, \texttt{northeast} \,;\, \texttt{east} + \texttt{north} = \texttt{northeast} \,\rangle$$

defines an hexagonal lattice that tiles the plane, see figure 3. Each cell has six neighbors (following the three generators and their inverses). The equation $\texttt{east} + \texttt{north} = \texttt{northeast}$ specifies that a move following `northeast` is the same as a move following the `east` direction followed by a move following the `north` direction.

For convenience, we identify the shape G of a GBF with its topological collection type (which is a presentation of the group G). A GBF g of type G can be formalized as a partial function, denoted also g, from G to some set of values: g associates a value to some positions (group elements acting as indices). An empty GBF is then the everywhere undefined function. The topology of the collections of type G is easily visualized as the Cayley graph $\boldsymbol{G}$ of G: each vertex in the Cayley graph is an element of the group G and vertex x and y are linked if there is a generator $\texttt{u}$ in the presentation of G such that $x + \texttt{u} = y$. See figure 1.

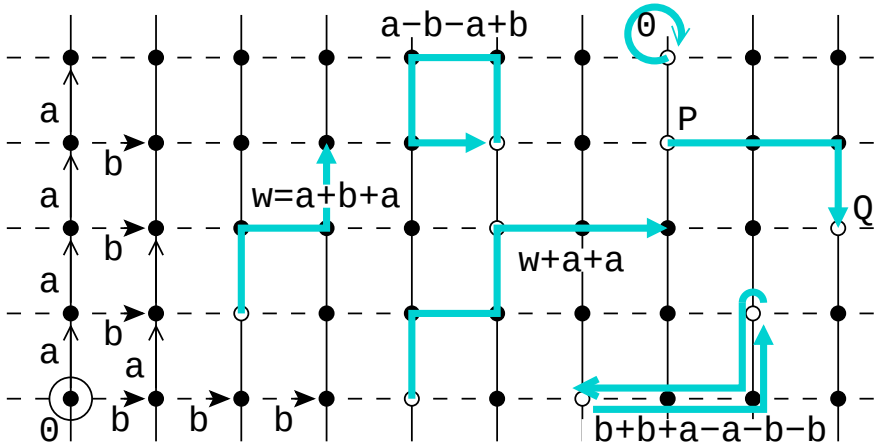

Figure 1: Graphical representation of the relationships between Cayley graphs and group theory. A vertex is a group element. An edge labeled a is a generator a of the group. A word (a product of generators) is a path. Path composition corresponds to group addition. A closed path (a cycle) is a word equal to e (the identity of the multiplication). An equation $v = w$ can be rewritten $v - w = e$ and then corresponds to a cycle in the graph. There are two kinds of cycles in the graph: the cycles that are present in all Cayley graphs and corresponding to group laws (intuitively: a backtracking path like $b + a - a - b$) and closed paths specific to the own group equations (e.g.: $a - b - a + b$). The graph connectivity (there is always a path going from P to Q) is equivalent to say that there is always a solution x to equation $P + x = Q$.

A presentation starting with ⟨ and ending with ⟩ introduces an *Abelian* organization: they are implicitly completed with the equations specifying the commutation of the generators $\mathtt{u} + \mathtt{u}' = \mathtt{u}' + \mathtt{u}$. Currently only free and Abelian groups are allowed in `MGS`: free groups with n generators correspond to n-ary trees and Abelian GBF corresponds to twisted and circular grids (the free Abelian group with n generators generalizes n-dimensional arrays). Some Abelian and non-Abelian shape are pictured in figure 2. However, all the results given here can be extended to any GBF with a shape corresponding to a group with a solvable word problem.

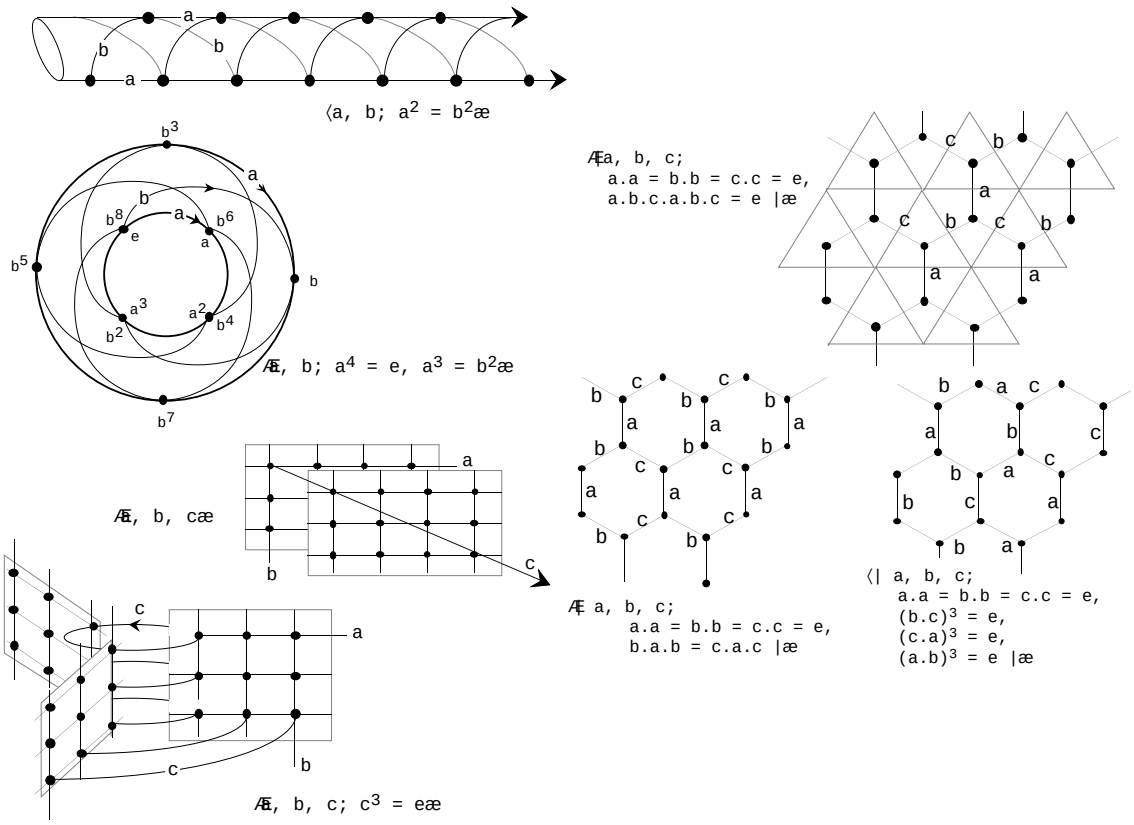

Figure 2: *Left.* Four Abelian group presentations (in multiplicative notation) and their associated graph. *Right.* Three examples of a 3-neighborhood shape (multiplicative notation). These triangular neighborhood are described by non Abelian groups.

Cayley P systems

As a matter of fact, a GBF g with a shape G represents the state of a P system on a graph [PSY01] provided that the underlying graph is the Cayley graph of some group. In this case, element of g are valuated by a multiset. Similarly, a GBF g with a shape G represents the state of a tissue P system [MVPPRP01] provided that the synapses constitute the Cayley graph $\boldsymbol{G}$ of the group G. The values of an element of g is then a couple (*neuron's state*, *multiset*).

Therefor, in this paper, a Cayley P system P is simply a set of membranes organized as a Cayley graph $\boldsymbol{G}$ together with a transformation: $P = (g, f)$ where g is a GBF with shape G and f a transformation. The membranes can be identified with the elements of G. Two membranes are neighbors if the are linked by an edge in $\boldsymbol{G}$. A symbol in a membrane can stay in the membrane or can move to a neighbor. Following the remarks in the introduction, we further simplify the state of a membrane as a boolean in the context of this work.

3 An example of Cayley P system : The Eden Model

Before formalizing the properties of Cayley P systems, let us examine an example, in order to clarify the notions and to illustrate the way of working of our systems.

We start with a simple model of growth sometimes called the Eden model (specifically, a type B Eden model [YPQ58]). The model has been used since the 1960's as a model for such things as tumor growth and growth of cities. In this model, a 2D space is partitioned in empty or occupied cells (we use the value `true` for an occupied cell and left undefined the unoccupied cells). We start with only one occupied cell. At each step, occupied cells with an empty neighbor are selected, and the corresponding empty cell is made occupied.

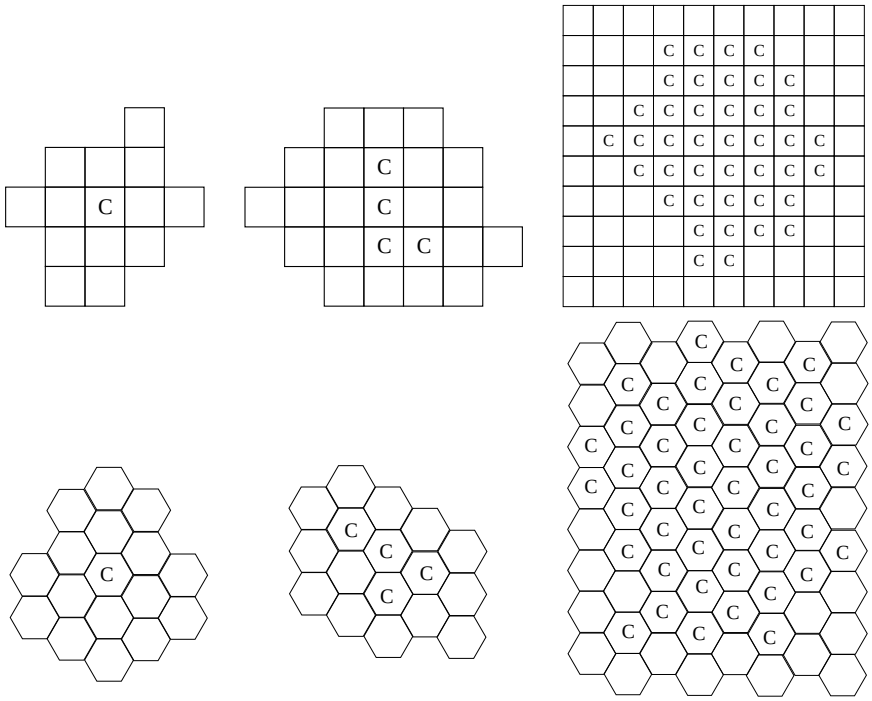

Figure 3: Eden's model on a grid and on an hexagonal mesh (initial state, and states after the 3 and the 7 time steps). *Exactly the same* transformation is used for both cases. These shapes correspond to a Cayley graph of *Grid2* and *Hexagon* with the following conventions: a vertex is represented as a face and two neighbors in the Cayley graphs share an edge in this representation. An empty cell has an undefined value. Only a part of the infinite domain is figured.

The Eden's aggregation process is simply described as the following `MGS` global transformation:

`trans` *Eden* = { x,`<undef>` $\Rightarrow$ x,`true` ; }

This global transformation is composed of only one local rule $p \Rightarrow sexp$. The expression p is a path pattern that matches a sequence of elements. The expression *sexp* computes *a sequence* of elements that replace *pointwise* the elements matched by p. To understand the path pattern, one must know that:

- The path pattern x, y means "select an x and an y, with well defined values such that x and y are neighbors". The comma operator denotes the neighborhood relationship in the left hand side (l.h.s.) of the rule and denotes the construction of a sequence in the r.h.s. An occurrence of the variable x anywhere in the rest of the rule denotes the value hold by the membrane matched by x. The expression $pos(x)$ denotes the membrane and is an element of the shape G of the GBF on which the transformation is applied.

- We assume that the boolean value `true` is used to represent an occupied cell, other cells are simply left undefined. The special symbol `<undef>` is used to match an undefined value.

Then the previous rule can be read: an occupied element x and an undefined neighbor are transformed into two occupied elements. The transformation *Eden* defines a function that can then be applied to compute the evolution of some initial state.

When applied to a collection f, the rules of a transformation T are applied synchronously, in maximal manner and to non-intersecting paths to make one evolution step. More precisely, if no rule applies, it means that there is no path matching the l.h.s. of one rule of T and then T is the identity function.

One of the advantages of the `MGS` approach, is that this transformation can apply indifferently on grid or hexagonal lattices, or *any* other collection kind. A difference with the manner in which a P systems is usually presented is that evolution rules in `MGS` are given globally and not attached to a membrane (however guards on rule can be used to dispatch the rules on specific membranes).

Cayley P Systems and Cellular Automata

It is interesting to compare transformations on GBFs with the genuine cellular automata (CA) formalism. There are several differences. The notion of GBF extends the usual square grid of CA to more general Cayley graphs. The value of a cell can be arbitrary complex (e.g. a multiset or even another GBF) and is not restricted to take a value in a finite set (however, we consider in this paper only boolean valued membranes). More importantly, the path pattern in a rule may match an arbitrary domain and not only *one* cell as it is usually the case for CA. For example the transformation:

```
trans Turn = {
  a |east> b |north-east> c |-east-north> d |east-north> e
  => a,e,b,c,d;
}
```

specify the 90°-turn of a cross in GBF GG (see illustration 4). Indeed, the path pattern x `|u>` y, where `u` is a group element, specifies that y is an `u`-neighbor of x: $pos(x) + \mathtt{u} = pos(y)$. Thus, the pattern fragment b `|north-east>` c specifies that c is at the north-west of element b.

Such behavior cannot be directly expressed in a cellular automata where a rule specifies the evolution of a single cell. Obviously such rule can be coded, but intermediate states and intermediate steps must be introduced in order for each cell to recognize the neighborhood and to synchronously trigger the rotation.

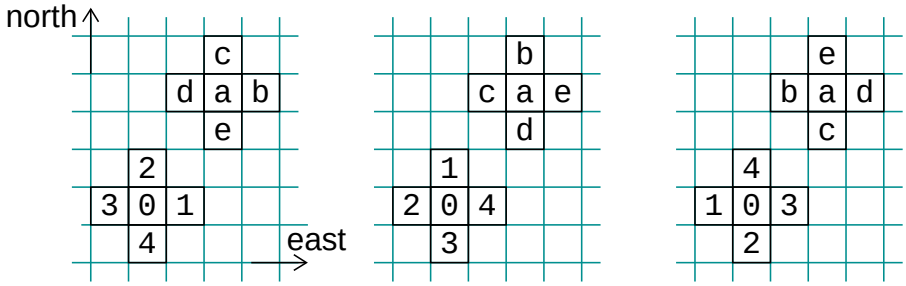

Figure 4: First and second iteration of transformation *Turn* on the GBF to the left (only defined values are pictured). In contrast with cellular automata, the evolution of a multi-cell domain can be easily specified by a single rule.

4 Accretive Rules and the Specification of a Configuration

Now, we concentrate our attention to rules f of the following form:

$$f = \quad x_1 \ \texttt{|u}_1\texttt{>}\ x_2\ \texttt{|u}_2\texttt{>}\ \ldots\ x_k\ \texttt{|u}_k\texttt{>}\ \texttt{<}\mathit{undef}\texttt{>} \quad \Rightarrow \quad x_1, x_2, \ldots, x_k, \texttt{true} \tag{1}$$

Such rules is used in the Eden's example and is representative of *accretive growth* [GGMP02] where the growing process takes place on the boundaries (this is to oppose to "intercalary growth" where the growing process is from the inside). Accretive growth is a special kind of transport rule: starting from a defined pattern $x_1\ \texttt{|u}_1\texttt{>}\ \ldots\ \texttt{|u}_{k-1}\texttt{>}\ x_k$ of already existing objects, a new object is created in an empty place. The quantity of objects hold by the membranes increases and this accounts for the growth.

This specific kind of transport rule is very important in morphogenesis and has been studied for a long time under various formalisms (Lindenmayer systems for instance or Eden's model in CA, etc.). They can be also used to model the *anisotropic* diffusion of a substance in a tissue. The Eden's rule is an *isotropic* diffusion because there is no difference made between the neighbors of the unoccupied membrane. A rule like:

$$x\ \texttt{|X> <undef>} \quad \Rightarrow \quad x, \texttt{true} \tag{2}$$

is an anisotropic diffusion in GG because the growth propagates only following the `X` direction. Anisotropic diffusion arises in biological tissue because some chemical gradient can make the process asymmetric, so as specific channel anchored in the cell membranes or when considering the effect of the extra-cellular matrix.

The l.h.s of rule f (eq. 1) specifies a set of membranes with a well-defined value and one empty membrane. We denote by R_f the set of dependencies of f:

$$\mathrm{R}_f = -\big\{\ \texttt{|u}_1\texttt{>} + \texttt{|u}_2\texttt{>} + \cdots + \texttt{|u}_k\texttt{>},\ \texttt{|u}_2\texttt{>} + \cdots + \texttt{|u}_k\texttt{>},\ \texttt{|u}_3\texttt{>} + \cdots + \texttt{|u}_k\texttt{>},\ \ldots,\ \texttt{|u}_k\texttt{>}\ \big\}$$

The expression $-A$ where A is a set denotes the inverses of the elements in A. The elements of R_f are the displacements from the position of an unocuppied membrane to the occupied ones involved by the rule f. The diameter of f is the number of elements in R_f.

Description of an Initial Configuration

We want to apply iteratively the transformation made of one rule f to some initial configuration g. The problem is to describe this initial configuration, even in the case of an unbounded initial domain.

A state of the Cayley P system is a GBF g and recall that a GBF is a partial function. In our case, this partial function associates to some membrane the value `true`. Thus, specifying g is equivalent to the specification of its definition domain. We want to take into account the group structure of g and then it is natural to use subgroups of G to specify the definition domain. however this is somewhat too restrictive and we will use *cosets*.

A coset $u \oplus H = \{u + h, h \in H\}$ is the "translation" by u of the subgroup H. In a non-Abelian group, we distinguish the right coset $u \oplus H$ and the left coset $H \oplus u$. To specify a coset we give the element u and the subgroup H. The notation $\langle a_1, a_2, \ldots, a_p\rangle : G$ defines a subgroup of G generated by $\{a_1, a_2, \ldots, a_p\}$ where the a_i are arbitrary elements of G. There is no specific equation linking the generators of the subgroup but they are subject to the equations of the enclosing group, if applicable.

An *initial configuration* is a finite union D_0 of cosets $C_1, \ldots, C_p$. The C_i are called the *base cosets* of the Cayley P system. It is easy to have specifically one element at position v in an initial configuration: it is enough to have one of the base cosets equals to $v \oplus \langle 0\rangle : G$ where 0 denotes the neutral element. Therefore, it is possible to describe with D_0 any finite set of membranes.

5 Formalization and Results

Let $P = (g, f)$ be a Cayley P system such that g of shape G is defined by the finite union of p cosets $\mathrm{D}_0 = C_1, \ldots C_p$ and the dependency set of f is $\mathrm{R}_f = \{r_1, \ldots, r_q\}$. For the sake of simplicity, we assume that G is an Abelian group. In the following, we reserve the index j to enumerate the cosets C_j and the index i to enumerate the dependences r_i.

The trajectory of P is the sequence g_n, $n \in \mathbb{N}$, where $g_0 = g$ and $g_{n+1} = f(g_n)$. Let D_n be the definition domain of g_n. As previously noted, it is equivalent to compute g_n or D_n. We are looking for a description of the state of the Cayley P systems after t evolution steps, that is, we are looking for a closed form of D_t. We also interested in a closed form of the limit domain

$$D \quad = \quad \lim_{n \to \infty} \mathrm{D}_n$$

An example. Our problem can be sketched on an example. Figure 5 illustrates the first three iterations of rule

$$h = \quad x_1 \texttt{ |east - north>} \, x_2 \texttt{ |east>} \, x_3 \texttt{ |north> <undef>} \quad \Rightarrow \quad x_1, x_2, x_3, \texttt{true} \tag{3}$$

starting from the initial configuration of shape GG:

$$\mathrm{D}_0 = C_1 \cup C_2 \quad \text{with} \quad C_1 = \langle \texttt{east} \rangle : GG \quad \text{and} \quad C_2 = \langle \texttt{north} \rangle : GG$$

The integer that appears in a membrane in the right hand side of figure 5 corresponds to the maximal length of a dependency path starting from the membrane and reaching a base coset. This integer can be thought of as the time step where the membrane value is produced (assuming a maximal rule application strategy). In this example, only one value can be produced at each time step. The membranes that have a well-defined value after 3 time steps are drawn as plain square cell. The infinite path that starts from the dotted membrane shows the beginning of an infinite dependency path: this path "jumps" over the cosets and goes to infinity, that is, this membrane will never have a defined value.

It should be obvious that

$$\begin{aligned} \mathrm{D}_n &= \mathrm{D}_{n-1} \cup (2n. \texttt{ |east> } + \texttt{ |north>}) \oplus \langle 0 \rangle : GG \\ D &= \mathrm{D}_0 \cup \texttt{ |north>} \oplus \langle 2. \texttt{ |east>} \rangle : GG \end{aligned}$$

In the previous expressions, the multiplication $n.x$ of a group element x by a positive integer n denotes the n fold sum $x + \ldots x$; if n is negative, then $n.x$ the inverse of $(-n).x$.

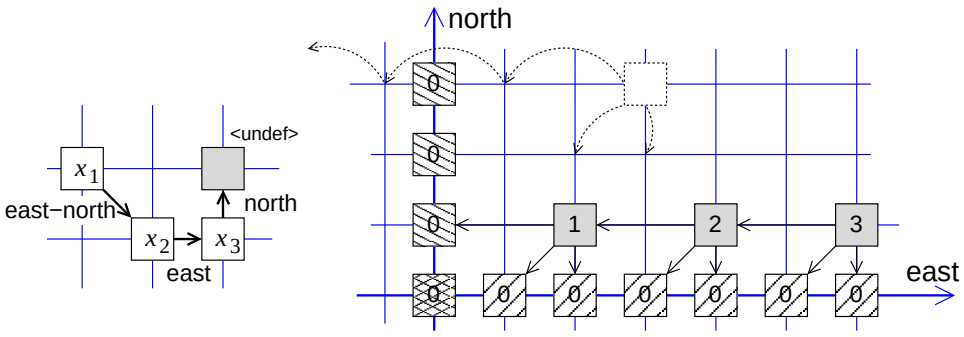

Figure 5: This schema figures a Cayley P system based on an free Abelian shape $GG = \langle \texttt{X}, \texttt{north} \rangle$. The diagram in the left hand side illustrates the path pattern of rule h (eq. 3). The right hand side illustrates the initial configuration of the Cayley P system and the first three iteration of the rule. The initial configuration is defined the cosets $\langle \texttt{east} \rangle : GG$ and $\langle \texttt{north} \rangle : GG$. The dependency set $\mathrm{R}_h = \{\texttt{-2.east}, \texttt{-north}, \texttt{-east - north}\}$. See the text for more explanation.

5.1 Definition of the lower approxiation D_n

Starting from the definition of D_n we have immediately:

$$\mathrm{D}_0 \subseteq \mathrm{D}_1 \subseteq ... \subseteq \mathrm{D}_n \subseteq ... \subseteq \mathrm{D}_\infty = D \tag{4}$$

Therefore, the sequence D_n gives a lower approximation of D. Furthermore, remark that an element u of D_{n+1} which is not an element of $\mathrm{D}_k, k \leq n$, is such that $(u + r_i) \in \mathrm{D}_n$ for all the dependencies r_i. In other word, u belongs to $\bigcap_i(\mathrm{D}_n \ominus r_i)$ where $A \ominus u = \{a - u, a \in A\}$. We can summarize this result:

$$\mathrm{D}_0 = \bigcup_j C_j \tag{5}$$

$$\mathrm{D}_{n+1} = \mathrm{D}_n \cup \bigcap_i \mathrm{D}_n \ominus r_i \tag{6}$$

5.2 Characterization of D as a least fixed point

Let u be in D_n such that $u \notin \mathrm{D}_0$, then $u + r_i \in \mathrm{D}_{n-1}$ for $1 \leq i \leq q$. Taking the limit in n, we have: if $u \in D, u \notin \mathrm{D}_0$, then $u + r_i \in D$. In other word, the set D satisfies the equation

$$D = \mathrm{D}_0 \cup \bigcap_i (D/\mathrm{D}_0) \oplus r_i \tag{7}$$

where $D/\mathrm{D}_0 = \{x \text{ such that } x \in D \wedge x \notin \mathrm{D}_0\}$. Equation 7 can be rephrased as a fixed point equation $D = \varphi(D)$ with the function φ defined by:

$$\varphi = \lambda A.\mathrm{D}_0 \cup \bigcap_i (A/\mathrm{D}_0) \oplus r_i$$

The fixpoint equation admits a least solution for the inclusion order (see any standard textbook on domain theory) that can be reached as the limit of $\varphi^n(\emptyset)$. We write this solution $\mathrm{fix}\,\varphi$. It is immediate to check that $\mathrm{D}_{n+1} \subseteq \varphi(\mathrm{D}_n)$, and then we have

$$D = \mathrm{fix}(\lambda A.\mathrm{D}_0 \cup \bigcap_i (A/\mathrm{D}_0) \oplus r_i)$$

5.3 The Greater Approximation E_n

A Geometric Interpretation. To obtain a greater approximation of D, we first interpret geometrically the property of belonging to the definition domain of g_n. To each position $u \in G$ we associate a set $\mathcal{P}_u$ of directed paths corresponding to the positions reachable from w using a sequence of dependencies. An element p of $\mathcal{P}_u$ is a word of the *monoid* $\mathcal{R}_f$ generated by R_f:

$$\mathcal{R}_f = \{\ \alpha_{i_1}.r_{i_1} + ... + \alpha_{i_k}.r_{i_k},\ \text{with } r_{i_l} \in \mathrm{R}_f \text{ and } \alpha_{i_l} \in \mathbb{N}\ \}$$

The membrane u cannot be in D if there exists a $p \in \mathcal{P}_u$ with an infinite length.

Computing a Greater Approximation E_0. If $u \in D$ is defined, then all the paths $p \in \mathcal{P}_u$ starting from u must end on a coset C_j. Amongst all these paths, there are some paths made only with r_i displacements. Let:

$$\mathrm{R}_i = \{\ -n.r_i,\ n \in \mathbb{N}\ \} \tag{8}$$

$$\mathrm{E}_0 = \mathrm{D}_0 \cup \bigcap_i \mathrm{D}_0 \oplus \mathrm{R}_i$$

The set R_i is the monoid generated by $-r_i$ (*warning:* we take the inverse of the dependency). The expression $A \oplus B$ denotes the set $\{a + b, a \in A, b \in B\}$. The set E_0 is made of the points $u \in G$

that either belong to D_0 or are such that there exists a path made only from r_i starting from u and reaching D_0. This last property is simply expressed as: $\forall i, \exists\, n_i,\ u - n.r_i \in \mathrm{D}_0$. This property holds for all $u \in D$ and then:

$$D \subseteq \mathrm{E}_0$$

Refining the Approximation E_0. The greater approximation E_0 is a little rude. We can refine them on the basis of the following remark. If $u \in D$, then we have either $u \in \mathrm{D}_0$ or $u + r_i \in D$. We can deduce that:

$$D \ \subseteq \ \mathrm{E}_1 = \ \mathrm{D}_0 \ \cup \ (\mathrm{E}_0 \ \cap \bigcap_i \mathrm{E}_0 \oplus r_i)$$

Obviously $\mathrm{E}_1 \subseteq \mathrm{E}_0$. Moreover, this construction starting from E_0 can be iterated, which introduces the sequence

$$\mathrm{E}_0 = \mathrm{D}_0 \ \cup \bigcap_i \mathrm{D}_0 \oplus \mathrm{R}_i \tag{9}$$

$$\mathrm{E}_{n+1} = \ \mathrm{D}_0 \ \cup \ (\mathrm{E}_n \ \cap \bigcap_i \mathrm{E}_n \oplus r_i) \tag{10}$$

We always have $D \subseteq \mathrm{E}_{n+1} \subseteq \mathrm{E}_n$.

Let E_∞ be the limit of E_n. For each $u \in \mathrm{E}_\infty$, we have either $u \in \mathrm{D}_0$ or $U + r_i \in \mathrm{E}_\infty$. Therefore, E_∞ is a solution of the equation (7). It should be checked that it is the *least* solution which we admit (intuitively, the element of G are equivalence classes of *finite words* of generators and then, if $x \in \mathrm{E}_\infty$ it can be checked by induction on the number of occurrences of r_i in x that $x \in D$).

5.4 Summary and a Conjecture

We can summarize the previous results by the formula:

$$\mathrm{D}_0 \ \subseteq \ ... \ \subseteq \ \mathrm{D}_n \ \subseteq \ ... \ \subseteq \ \mathrm{D}_\infty = D = \mathrm{E}_\infty \ \subseteq \ ... \ \subseteq \ \mathrm{E}_n \ \subseteq \ ... \ \subseteq \ \mathrm{E}_0 \tag{11}$$

These results can be generalized without difficulty by considering more general base case domains. That is, we may replace the coset C_i by an arbitrary set S_i in equation (5) relations (11) remain true.

A *monoid* M generated by element $u_1, ..., u_p$ of a group G is the set of elements that can be written as a positive linear combination of the u_i's. We call *comonoid* the translation of a monoid, that is, a set $x \oplus M = \{x.m, m \in M\}$ where M is a monoid. For all the examples we have worked out on free Abelian groups (that is, for membranes organized as a d-dimensional grid and indexed by $\mathbb{Z}^d$), we have checked that the limit domain D is a *finite union of comonoids*. We conjecture that this is always true. Figure 6 at the end of the paper gives six examples in $\mathbb{Z}^2$.

6 Computing the D_n and E_n in the Abelian case

Equations (5, 6, 8, 9, 10) enable the explicit construction of D_n and E_n if it is known how to compute intersection, union and product of *comonoid*.

Indeed, a coset is a special kind of comonoid and the intersection of a comonoid is either empty or a comonoid. If the sum $D \oplus M$ of a comonoid D by a monoid M is also a monoid (which is the case for Abelian shape or if the r_i commutes with all group elements), then all arguments of the intersections and unions in the previous equations are comonoids. We may then express D_n and E_n for a given n has a finite union of comonoids. It is then clear that the definition domain of g is an union of comonoids. The conjecture only says that this union is finite.

We have used the `omega calculator`, a software package [KMP+96] that enables the computation of various operations on convex polyhedra to make linear algebra in $\mathbb{Z}^n$ and represent

comonoids. Linear algebra is not enough to compute D_n and E_n because we have to compute the R_i. Fortunately, the `omega calculator` is able to determine in some cases[1] the *transitive closure* of a relation [KPRS94] which enables the computation of R_i as the transitive closure of the relation `[x, x+r`$_i$`]`. We use here the syntax of the `omega calculator` and an expression such `[`f`(x)]`, where `x` is a free variable, denotes the set $\{f(x), x \in \mathbb{Z}\}$ and an expression `[x,` f`(x)]`, defines a relation linking `x` to f`(x)`. Please refer to [KMP+96] for the `omega calculator` concepts and syntax.

Here is in example, based on the Cayley P system illustrated in figure 5. We first define the base cosets in $\mathbb{Z}^2$

```
C1 := { [n, 0] };
C2 := { [0, n] };
```

then three relations that correspond to the dependencies:

```
r1 := { [x, y] -> [x, y-1] };
r2 := { [x, y] -> [x-2, y] };
r3 := { [x, y] -> [x-1, y-1] };
```

and we need also the inverse of the dependencies:

```
ar1 := { [x, y] -> [x, y+1] };
ar2 := { [x, y] -> [x+2, y] };
ar3 := { [x, y] -> [x+1, y+1] };
```

We may now defines the D_i:

```
D0 := C1 union C2;

H1 := ar1(D0) intersection ar2(D0) intersection ar3(D0);
D1 := D0 union H1;

H2 := ar1(D1) intersection ar2(D1) intersection ar3(D1);
D2 := D1 union H2;

H3 := ar1(D2) intersection ar2(D2) intersection ar3(D2);
D3 := D2 union H3;
```

We can ask `omega` to compute a representation of D_3. The query `D3` returns:

```
{[x,0]} union {[0,y]} union {[4,1]} union {[6,1]} union {[2,1]}
```

which is what it is expected. For the approximation E_i we need to represent the monoids R_i which is done through a transitive closure:

```
AR1 := ar1*;
AR2 := ar2*;
AR3 := ar3*;
```

The definition of E_0 raises the computation of

```
E0 := AR1(D0) intersection AR2(D0) intersection AR3(D0);
```

(we have omitted the union with D_0 to avoid too complicated terms in the result). The evaluation of this definition returns

```
{[x,y]:  Exists (alpha :  0 = x+2alpha && 1 <= y && 2 <= x)}
union {[x,0]} union {[0,y]}
```

This approximation is too large, we may refine it by computing E_1:

[1] We plan to develop a dedicated library under `Mathematica` to compute these approximations systematically.

```
E1:= ar1(r1(E0) intersection E0) intersection
     ar2(r2(E0) intersection E0) intersection
     ar3(r3(E0) intersection E0);
```

The evaluation of E_1 gives:

```
{[x,1]:  Exists ( alpha :  0 = x+2alpha
                        && 4 <= x)} union {[2,1]}
```

which is also D minus D_0.

7 Conclusions

In this work, we have considered a specific transport process, called accretive growth, on a set of membranes organized as a Cayley graph. We have defined formally the trajectory of the resulting Cayley P system and characterized the final configuration. We conjecture that this configuration can be described as a finite union of so-called comonoid but we were unable to proove this assertion.

The idea to use a group structure to specify the graph underlying a tissue P system has many links with the concepts of GBF developed in the framework of the `MGS` language and the reader may refer to the references cited in the paper. To complete this conclusion, we review two related domain: cellular automata on Cayley graphs and systolic programming.

Cellular Automata on Cayley Graphs

Moving symbols between a set of cells is reminiscent of some process described in the cellular automata literature. We have pointed in section 3 the main differences. However, it is worth mentioning the work of Z. Róka on the extension of the cellular automata (CA) formalism to handle more general cell space. She considers Cayley graphs in [Rók94, Rók95b, Rók95a] to model both the cell space and the communication links between the cells (the use of Cayley graphs as intersection networks have been extensively studied, see e.g. [Hey97]). These research focus the conditions for the simulation of a CA on a given Cayley graph by another CA on another Cayley graph and to the algorithmic problem of the global synchronization of a set of cells.

There exists strong links between Cayley P system and such extension of cellular automata: in the two cases we have to study the propagation of computations in a space described by a Cayley graph. However, the mentioned work focuses on synchronization problems and establishes complexity results for various simulation. For instance, the characterization of the domain is out of the CA scope.

Systolic Programming

We recall here the terminology concerning recurrence equations. *Uniform recurrence equations* (URE) have been introduced by Karp, Miller and Winograd [KMW67]. Their model have been broadened to *affine recurrence equations* ARE. An ARE takes the following form:

$$\forall z \in D, \quad U(z) \;=\; f(U(I(z)),\, V(I'(z)),\, ...) \tag{12}$$

where D is a convex polyhedron of $\mathbb{Z}^n$ called the *domain* of the equation; z is a point of $\mathbb{Z}^n$; U, V are *variable names* indexed by z (the dimension of the index of a given variable is constant). The functions $I, I', \ldots$ are affine mappings from $\mathbb{Z}^n$ to $\mathbb{Z}$. The variable $U(I(z))$ is an argument and $U(z)$ is a result of the equation. The function f is strict. If all mappings I are translations then the system is said to be an URE. This formalism has been largely used. Indeed, there is a large corpus of mathematical results in linear algebra that can help to solve the problems encountered. One of the main problem is to characterize the definition domain of the function specified by equation (12).

This problem is very similar to the characterization of the limit domain D. Indeed, because the function f is strict, the equations defining the definition domain of U have the same formal expression that the equations defining D (assuming $\mathbb{Z}$ as the underlying group), see [Gia99].

Definition Domain of an ARE. We can review some results in this domain. Karp, Miller and Winograd [KMW67], have shown the decidability for URE on a bounded domain, without explicitly constructing the dependency graph.

However, B. Joinnault [Joi87] has shown the indecidability when the domain of the equations is not bounded. The proof relies on the coding of a Turing machine by an URE. The functions used in the specification of an URE are strict, that is, we do not have a conditional; the conditional is simulated by an adequate specification of the domain of the equations.

This result cannot be adapted in the case of Cayley P systems because the specification of the definition domain of an URE (which plays for the URE the same role as the base cosets) relies on the specification of convex polyhedra in $\mathbb{Z}^n$ and a convex polyhedron is not generally a coset in $\mathbb{Z}^n$ neither a finite union of cosets or the complementary of a finite union of cosets. In [SQ93], the indecidability result is extended to the case of *parametric bounded domain* (i.e. the domain is described by an union of finite convex polyhedron parameterized by a parameter $p \in \mathbb{Z}^m$) (for a given value of p, the domain is finite).

Dependencies beyond the affine dependences can be found in exact or approximate data flow analysis [Fea91, LC94]. More specifically for recursive structure, J.-F. Collard and A. Cohen [Coh96] have used the group structure to specify and analyze recursive computations on trees.

Acknowledgments

The authors would like to thanks the members of the "Simulation and Epigenesis" group at Genopole for stimulating discussions and biological motivations. They are also grateful to P. Prusinkiewicz, F. Delaplace and J. Cohen for numerous questions, encouragements and thoughtful remarks. This research is supported in part by the CNRS, the GDR ALP and IMPG, the University of Evry and Genopole/Evry.

References

[Coh96] Albert Cohen. Structure de données régulières et analyse de flot. Dea, ENS-Lyon, June 1996.

[Fea91] Paul Feautrier. Dataflow analysis of scalar and array references. *Int. Journal of Parallel Programming*, 20(1):23–53, February 1991.

[GGMP02] J.-L. Giavitto, C. Godin, O. Michel, and P. Prusinkiewicz. *Biological Modeling in the Genomic Context*, chapter "Computational Models for Integrative and Developmental Biology". Hermes, July 2002. (to appear).

[Gia99] J.-L. Giavitto. Scientific report for the tenure. Technical report, LRI, Université de Paris-Sud, centre d'Orsay, September 1999. Research Report 1226.

[Gia00] Jean-Louis Giavitto. A framework for the recursive definition of data structures. In *Proceedings of the 2nd Imternational ACM SIGPLAN Conference on Principles and Practice of Declarative Programming (PPDP-00)*, pages 45–55. ACM Press, September 20–23 2000.

[GM01a] J.-L. Giavitto and O. Michel. Declarative definition of group indexed data structures and approximation of their domains. In *Proceedings of the 3nd Imternational ACM SIGPLAN Conference on Principles and Practice of Declarative Programming (PPDP-01)*. ACM Press, September 2001.

[GM01b] J.-L. Giavitto and O. Michel. Mgs: Implementing a unified view on four biologically inspired computational models. In *Pre-proceedings of WMC-CdeA 2001 (Workshop on Membrane Computing, Curtea de Arges)*. Research Report 17/01 of the Universitat Rivira I Virgili, Tarragona, Spain, August 2001.

[GM01c] Jean-Louis Giavitto and Olivier Michel. Mgs: a rule-based programming language for complex objects and collections. In Mark van den Brand and Rakesh Verma, editors, *Electronic Notes in Theoretical Computer Science*, volume 59. Elsevier Science Publishers, 2001.

[GM02] J.-L. Giavitto and O. Michel. The topological structures of membrane computing. *Fundamenta Informaticae*, 49:107–129, 2002.

[GMS96] J.-L. Giavitto, O. Michel, and J. Sansonnet. Group-based fields. In *Parallel Symbolic Languages and Systems (Int. Workshop PSLS'95)*, volume LNCS 1068, pages 209–215. Springer, 1996.

[Hey97] Marie-Claude Heydemann. *Graph Symmetry*, chapter "Cayley graphs and interconnection networks", pages 167–224. Kluwer Academic Publisher, 1997.

[Joi87] Brigitte Joinnault. *Conception d'algorithmes et d'architecture systoliques.* PhD thesis, Thèse de l'Université de Rennes I, September 1987.

[KMP+96] Wayne Kelly, Vadim Maslov, William Pugh, Evan Rosser, Tatiana Shpeisman, and Dave Wonnacott. *The Omega calculator and library, version 1.1.0.* College Park, MD 20742, 18 november 1996.

[KMW67] Richard M. Karp, Raymond E. Miller, and Shmuel Winograd. The organization of computations for uniform recurrence equations. *Journal of the ACM*, 14(3):563–590, July 1967.

[KPRS94] Wayne Kelly, William Pugh, Evan Rosser, and Tatiana Shpeisman. Transitive closure of infinite graphs and its application. Technical Report UMIACS-TR-95-48, CS-TR-3457, Univ. of Maryland, College Park, MD 20742, 14 Aprils 1994.

[LC94] B. Lisper and J.-F. Collard. Extent analysis of data fields. Technical Report TRITA-IT R 94:03, Royal Institute of Technology, Sweden, January 1994.

[MVPPRP01] C. Martin-Vide, Gh. Paun, J. Pazos, and A. Rodriguez-Paton. Tissue P Systems. Technical Report TUCS tech. rep. 421, Turku Centre for Computer Science, September 2001.

[Pau99] G. Paun. Computing with membranes: An introduction. *Bulletin of the European Association for Theoretical Computer Science*, 67:139–152, February 1999.

[Pau01] G. Paun. From cells to computers: Computing with membranes (p systems). *Biosystems*, 59(3):139–158, March 2001.

[PSY01] Gh. Paun, Y. Sakakibara, and T. Yokomori. P systems on graphs of restricted forms. *Publ. Math. Debrecen*, 2001. (to appear).

[Rók94] Zsuzsanna Róka. One-way cellular automata on Cayley graphs. *Theoretical Computer Science*, 132(1–2):259–290, 26 September 1994.

[Rók95a] Zsuzsanna Róka. The firing squad synchronization problem on Cayley graphs. *Lecture Notes in Computer Science*, 969:402–??, 1995.

[Rók95b] Zsuzsanna Róka. Simulations between cellular automata on Cayley graphs. *Lecture Notes in Computer Science*, 911:483–??, 1995.

[SQ93] Y. Saouter and P. Quinton. Computability of recurrence equations. *Theoretical Computer Science*, 116(2):317–337, August 1993.

[YPQ58] Hubert P. Yockey, Robert P. Platzman, and Henry Quastler, editors. *Symposium on Information Theory in Biology.* Pergamon Press, New York, London, 1958.

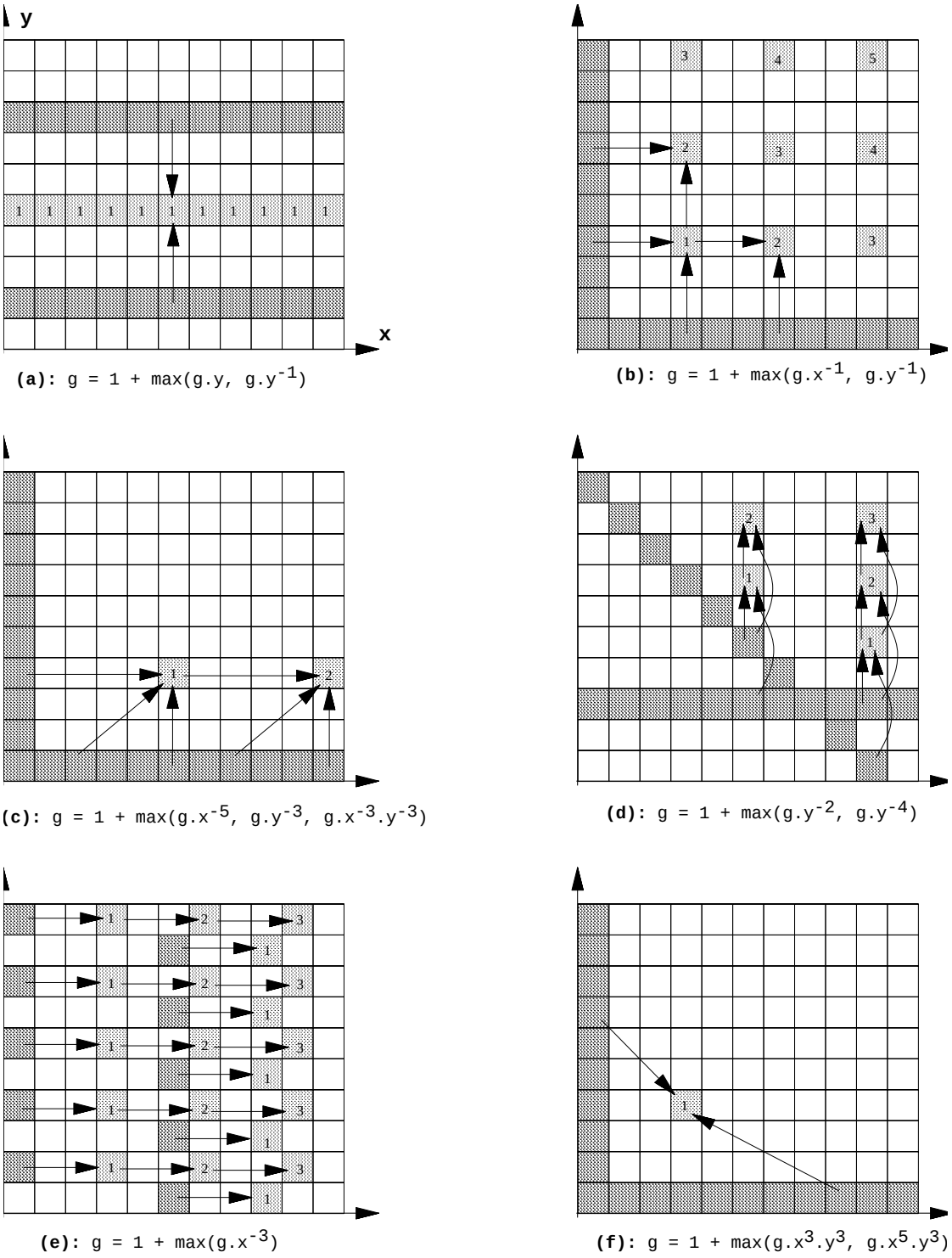

Figure 6: Six examples of GBF domains in $\mathbb{Z}^2$.
The elements of the shape $\langle x, y\rangle$ are figured by a square cell. We have presented only a part of the shape, where the left bottom square corresponds to the neutral element. We use for convenience a multiplicative notation instead of the additive notation used for the group operation in the text. The cells in gray belong to the definition domain of the GBF g. The default equation is figured but not the quantified equations. The cells of base cosets are in dark gray and without label. It is easy to recover their equations. For example, let $X = \langle x\rangle$ and $Y = \langle y\rangle$. Then, the cosets C_1 and C_2 involved in `(a)` are $y.X$ and $y^7.X$ The cosets involved in `(c)` are X and Y. Etc.