mathematics

MDPI

Article

An Algorithm for Counting the Fixed Point Orbits of an AND-OR Dynamical System with Symmetric Positive Dependency Graph

Mauro Mezzini [1,*] and Fernando L. Pelayo [2]

[1] Department of Education, Roma Tre University, 00154 Roma, Italy
[2] Escuela Superior de Ingenieria Informatica de Albacete, Computing Systems Department, University of Castilla-La Mancha, 02071 Albacete, Spain; fernandol.pelayo@uclm.es
* Correspondence: mauro.mezzini@uniroma3.it

Received: 23 July 2020; Accepted: 14 September 2020; Published: 18 September 2020

Abstract: In this paper we present an algorithm which counts the number of fixed point orbits of an AND-OR dynamical system. We further extend the algorithm in order to list all its fixed point orbits (FPOs) in polynomial time on the number of FPOs of the system.

Keywords: fixed point; AND-OR dynamical system; algorithmic complexity

1. Introduction and Preliminaries

A directed graph G is a couple $G = (V(G), E(G))$ where $V(G)$ is a set of vertices and $E(G) \subseteq V(G) \times V(G)$ is a set of directed edges. In our setting, each edge (u, v) is labeled positive or negative. The set of positive (resp. negative) edges of the graph are denoted by $E^+(G)$ (resp. $E^-(G)$). A graph is called positive (resp. negative) if all labels on the edges are positive (resp. negative). If $S \subseteq V(G)$, the subgraph of G induced by S, denoted as $G(S)$, is the graph whose vertices set is $V(G(S)) = S$ and its edge set is $E(G(S)) = \{(u, v) \in E(G) | \{u, v\} \subseteq S\}$. We denote by $G - S$ the subgraph of G induced by $V(G) \setminus S$.

A graph is undirected if for every $(u, v) \in E(G)$ then $(v, u) \in E(G)$. A graph is symmetric if it is undirected and for every $v \in V(G)$ then $(v, v) \in E(G)$. That is, a symmetric graph is an undirected graph where each vertex has a (directed) selfloop.

If (u, v) is an edge then we say that u is a predecessor of v and it is positive (resp. negative) if (u, v) is a positive (resp. negative) edge. The set of positive (resp. negative) predecessors of a vertex v is denoted as $P_G^+(v) = \{u | (u, v) \in E^+(G)\}$ (resp. $P_G^-(v) = \{u | (u, v) \in E^-(G)\}$). The set of predecessors of v is $P_G(v) = P_G^+(v) \cup P_G^-(v)$.

The neighborhood of a vertex v in a graph G, denoted as $N_G(v)$ is the set $N_G(v) = \{u | (v, u) \in E(G)\}$. The closed neighborhood of v, denoted as $N_G[v]$ is $N_G[v] = N_G(v) \cup \{v\}$. The degree of a vertex is the number of vertices in its closed neighborhood and it is denoted by $\delta_G(v) = |N_G[v]|$.

A dynamical system (DS) [1–4] or Boolean Network (BN) [5–7] $\mathcal{S}$ is a couple $\mathcal{S} = (G, F)$ where G is a directed graph called the dependency graph, F is a set of functions $F = \{f_v | v \in V(G)\}$ as defined below. We also define a $|V(G)|$-dimensional vector O of states. $O = (o_1, o_2, \ldots, o_n)$ where $o_v \in \{0, 1\}$, $v \in V(G)$. The vector O will be referred as an orbit of $\mathcal{S}$. We also consider the restriction (or the projection) of the vector O on a subset of vertices $W \subseteq V(G)$ and we refer to it as O_W. Each function $f_v : \{0, 1\}^h \to \{0, 1\}$ is taken from the set of Boolean functions, where h is the number of predecessors of v. The argument of f_v is an h-dimensional vector $x = (x_{u_1}, \ldots, x_{u_h})$ where $x_{u_i} = o_{u_i}$ when $u_i \in P_G^+(v)$, and, $x_{u_i} = \bar{o}_{u_i}$ when $u_i \in P_G^-(v)$ being $\bar{o}_{u_i}$ the Boolean negation (or the complement modulo 2) of o_{u_i}. Although f_v can be an arbitrary Boolean function on h variables, most pieces of

research on DS and BN restrict the set of Boolean functions to AND, OR, NAND and NOR. This is motivated by the following.

An AND term (resp. OR term) is the logical AND (resp. logical OR) of a set of variables (where each variable can be either complemented or not complemented). Given a set of h Boolean variables $x_1, \ldots, x_h$, an AND term in which each of the h variables appears once (in either its complemented or not complemented form) is called a minterm. Analogously, an OR term in which each of the h variables appears once (in either its complemented or not complemented form) is called a maxterm. It is well known that every Boolean function can be expressed either by disjunction of uniquely minterms or by conjunction of just maxterms, among others. Furthermore, it is well known that the Boolean algebra using the binary operators AND, OR and NOT are equivalent to the algebra obtained by using either only NAND or only NOR.

In addition, the complexity of the problems involved in a DS dramatically changes when we modify either the structure of the dependency graph, or the type or distribution of the Boolean functions involved in the system. If the vertices of a DS use a function on a set $\mathcal{F}$ then we refer to the dynamical system as an $\mathcal{F}$-DS. We also call a DS positive when its dependency graph is positive.

Given a vertex v of a DS, we may consider the state of the vertex o_v as a function of a discrete time $t \in \mathbb{N}$. Consequently, the global state O of the DS depends also on the time t. At time $t = 0$, we assign a value to each o_v and if $O(t)$ is the state of DS at time t then the state of a single vertex at time $t + 1$ will be $o_v(t + 1) = f_v(O_{P_G(v)}(t))$ where $O_{P_G(v)}$ is the restriction of O to the predecessors of v. If the states of the vertices are updated in parallel then we refer to the system as a parallel DS or PDS. In other settings, we want to update the state of exactly one vertex at time t. A permutation $\pi = \{v_0, \ldots, v_{n-1}\}$ of the vertices is given and if v_i is the vertex that is updated at time t then only v_{i+1} (indices taken modulo n) will change its state at time $t + 1$ and thus $o_{v_{i+1}}(t + 1) = f_{v_{i+1}}(O_{P_G(v_{i+1})}(t))$. In this case, we call the system a sequential DS or SDS.

Given a DS $\mathcal{S} = (G, F)$ and an orbit O, we call O a fixed point orbit (FPO) of $\mathcal{S}$ if and only if $O(t) = O(t + 1)$. When the set of functions F is clear from the context, we refer to O just as an FPO of G.

Fixed point orbits are particularly interesting since they correspond to stable states of the system. The problem of determining the existence of a fixed point orbit of a DS is called fixed point orbit existence (FPOE) problem. A number of works have explored the FPOE problem under the hypothesis that all the vertices have a selfloop. Some works put an additional restriction which requires the graph of the DS to be symmetric and/or either all the edges to be positive or all of them negative. Among others, the FPOE problem of an {AND}-DS, {OR}-DS, {NAND}-DS and {NOR}-DS is studied in [2–4,8–10]. In addition, the FPOE problem in an {AND, OR, NAND, NOR}-DS is polynomially solvable as shown in [4,11]. However, for {NAND, XNOR}-DS, {NAND, XOR}-DS,{NOR, XNOR}-DS and {NOR, XOR}-DS, the FPOE problem is NP-complete [11]. The problem of counting the fixed point orbits in a DS whose dependency graph is symmetric and all edges are positive is studied in [6] where it is proved that the problem of counting the FPOs in a $\mathcal{F}$-DS is #P-complete under the hypothesis that the function associated to each node is arbitrary.

When the constraints that require the graph to be symmetric and positive are released, then the FPOE problem will be NP-complete in {AND, OR}-DS [12,13]. It remains NP-Complete even when the indegree of each vertex is at most two [13]. It can be demonstrated that the problem remains NP-complete for {AND}-DS with all negative edges by using the transformation given in [5].

In this paper, we present an algorithm for counting the FPOs in an AND-OR symmetric positive PDS. The algorithm has been designed by using more or less standard methodologies (such as the one used also in [14–23]) and by using both a greedy strategy and recursion. The paper is organized as follows. In Section 2, we present some theoretical results about AND-OR PDS whose dependency graph is symmetric and positive. Based on them, we present an algorithm for counting all FPOs in AND-OR symmetric positive PDS and we prove its correctness. In Section 3, we further develop the algorithm given in Section 2 for building an algorithm which lists all FPOs in AND-OR symmetric PDS.

2. Algorithm for Computing the Number of FPOs in AND-OR Symmetric Positive PDS

In this section, we first determine the structure and the number of FPOs in {AND, OR}-DS whose dependency graph is symmetric and positive. It is well known that for this kind of graph the FPOE problem is polynomially solvable [2,3]. Let $\mathcal{S} = (G, F)$ be an {AND, OR}-DS where G is a symmetric positive graph. We denote by V_{OR} (resp. V_{AND}) the vertices of G such that f_v is OR (resp. AND). Furthermore, if for a vertex v we have that $f_v =$ OR (resp. $f_v =$ AND) we refer to v as an OR vertex (resp. AND vertex).

Since, in the following, we will limit our discussion to symmetric positive {AND, OR}-DS, for brevity we will often refer to them as DS without explicitly specifying that the dependency graph is symmetric positive and the set of Boolean functions $F =$ {AND, OR}.

Lemma 1. *Let $\mathcal{S}$ be a DS and let O be an FPO of $\mathcal{S}$. Let $U \subseteq V(G)$ and let $\mathcal{S}'$ be a DS whose dependency graph is $G(U)$. Then O_U is an FPO of $\mathcal{S}'$.*

Proof. Let G' be the subgraph of G induced by U and let $O' = O_U$, that is, O' is the restriction of O to the vertices in U. We prove only that if $v \in U \cap V_{\text{OR}}$ then $o'_v(t) = o'_v(t+1)$. The proof when $v \in U \cap V_{\text{AND}}$ can be obtained by applying a similar argument. Suppose first that $o'_v = 1$. In this case clearly we have that $o'_v(t) = o'_v(t+1)$. Suppose now that $o'_v = 0$. Then, since O is an FPO of $\mathcal{S}$, for each $u \in N_G(v)$ we have that $o_u = 0$. Thus, in G', for each $u \in N_{G'}(v)$ we have that $o'_u = 0$. Therefore, it follows that $o'_v(t) = o'_v(t+1)$. □

Let $A = \{A_1, \ldots, A_k\}$ be the set of the connected components of $G - V_{\text{AND}}$ and let $B = \{B_1, \ldots, B_h\}$ be the set of the connected components of $G - V_{\text{OR}}$. Let $\mathcal{B}_G$ be the symmetric bipartite graph whose vertices set is $V(\mathcal{B}_G) = A \cup B$ with bipartition (A, B) (we remark that $\mathcal{B}_G$ can be disconnected or that $A = \emptyset$ or $B = \emptyset$). There is an edge (A_i, B_j) in $E(\mathcal{B}_G)$ if and only if there is at least one edge $(u, v) \in E(G)$ with $u \in A_i$ and $v \in B_j$. We will refer to $\mathcal{B}_G$ as the AND-OR bipartition of G. Furthermore we associate to $\mathcal{S}$ an {AND, OR}-DS $\mathcal{S}' = (\mathcal{B}_G, F')$ referred to as the bipartite contraction of $\mathcal{S}$ in which $f'_a =$ OR if $a \in A$ and $f'_b =$ AND if $b \in B$.

Let us notice that $S' = (\mathcal{B}_G, F')$ can be achieved from $\mathcal{S} = (G, F)$ just computing the connected components within the sub-graphs associated to $G - V_{\text{AND}}$ and to $G - V_{\text{OR}}$. In order to do this it is sufficient with contracting all the vertices belonging to the same connected component in one.

The computational cost associated to this process is $O(n)$ being $n = |E(G)| + |V(G)|$.

For example, in Figure 1 on the left, there is an {AND, OR}-DS in which AND vertices are shaded grey and OR vertices are shaded white, whereas on the right its bipartite contraction appears.

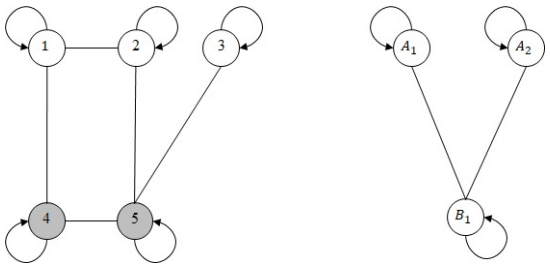


Figure 1. An {AND, OR}-DS and its bipartite contraction.

Lemma 2. *Let $\mathcal{S}$ be a DS and let A be a connected component of $G - V_{AND}$ such that $|V(A)| > 1$. If O is an FPO of $\mathcal{S}$, then for all pairs of vertices $v, u \in V(A)$ we have that $o_v = o_u$.*

Proof. Let $v, u \in V(A)$ and suppose by contradiction that $o_u \neq o_v$. Let P be a path in A connecting v to u. Then, there are two adjacent vertices, say w and z, on this path, such that $o_z \neq o_w$. Suppose, without loss of generality, that $o_z = 1$. Since G is symmetric and positive there is an edge (z, w) in G. Then O could not be an FPO since at time $t+1$ we would have that $o_w = 1$, a contradiction. □

By duality and in a similar way to Lemma 2, the following result is immediate to be proven:

Lemma 3. *Let $\mathcal{S}$ be a DS and let B be a connected component of $G - V_{OR}$ such that $|V(B)| > 1$. If O is an FPO of $\mathcal{S}$, then for all pairs of vertices $v, u \in V(B)$ we have that $o_v = o_u$.*

Now we show that FPOs of a DS are strictly related with FPOs of its corresponding bipartite contraction. In fact, we can prove the following two Lemmas.

Lemma 4. *Let $\mathcal{S} = (G, F)$ be a DS and let $\mathcal{S}' = (\mathcal{B}_G, F')$ be the bipartite contraction of $\mathcal{S}$. If O is an FPO of $\mathcal{S}$ then the orbit O' obtained from O by setting, for each $C \in V(\mathcal{B}_G)$, $o'_C = o_v$ for any $v \in V(C)$, is an FPO of $\mathcal{S}'$.*

Proof. Let O be an FPO of $\mathcal{S}$. Let $A = \{A_1, \ldots, A_k\}$ be the set of the connected components of $G - V_{\text{AND}}$ and $B = \{B_1, \ldots, B_h\}$ be the set of the connected components of $G - V_{\text{OR}}$. If $A_i \in A$, by Lemma 2, for all pairs of vertices $u, v \in A_i$ we have that $o_u = o_v$. Similarly, by Lemma 3, if $B_j \in B$, for all pairs of vertices $u, v \in B_j$ we have that $o_u = o_v$. Let us suppose by contradiction that O' is not an FPO of $\mathcal{S}'$. Then there is a vertex $C \in V(\mathcal{B}_G)$ such that $o'_C(t) \neq o'_C(t+1)$. Let us suppose that $C \in A$ (the proof of the case in which $C \in B$ is similar and omitted). By this assumption it cannot be that $o'_C(t) = 1$ because otherwise $o'_C(t+1) = 1$ since $\mathcal{B}_G$ is symmetric and positive and $f'_C =$OR. Therefore, we have that $o'_C = 0$ and C is adjacent, in $\mathcal{B}_G$, to a vertex C' such that $o'_{C'} = 1$. Then, by definition of bipartite contraction, there exists a vertex $u \in V(C)$ and a vertex $v \in C'$ such that $(v, u) \in E(G)$ and, by what said above, $o_u = 0$ and $o_v = 1$. Furthermore $u \in V_{\text{OR}}$. Then $o_u(t) \neq o_u(t+1)$ and O would not be an FPO of $\mathcal{S}$ (a contradiction to the hypothesis). □

Lemma 5. *Let $\mathcal{S} = (G, F)$ be a DS and let $\mathcal{S}' = (\mathcal{B}_G, F')$ be the bipartite contraction of $\mathcal{S}$. If O' is an FPO of $\mathcal{S}'$ then the orbit O obtained from O' by setting $o_v = o'_C$ for all $v \in V(C)$ and for all $C \in V(\mathcal{B}_G)$ is an FPO of $\mathcal{S}$.*

Proof. Let us suppose by contradiction that O is not an FPO of $\mathcal{S}$. There must be at least a vertex $v \in V(G)$ such that $o_v(t+1) \neq o_v(t)$. Suppose that $v \in V_{\text{OR}}$ and let A be the connected component of $G - V_{\text{AND}}$ containing v. Since $o_v(t+1) \neq o_v(t)$ and $v \in V_{\text{OR}}$, we must have that $o_v = 0$ and that there exists a vertex $u \in N_G(v)$ such that $o_u = 1$. Since, by construction, $o_z = 0$ for all $z \in V(A)$, we have that $u \in V_{\text{AND}}$. Let B be the connected component of $G - V_{\text{OR}}$ containing u. By construction, and since $o_v = 0$ and $o_u = 1$, we have that $o'_A = 0$, $o'_B = 1$ and (A, B) is an edge of $\mathcal{B}_G$. Then we have that $o'_A(t+1) \neq o'_A(t)$ contradicting the hypothesis that O' is an FPO of $\mathcal{S}'$. The proof in which $v \in V_{\text{AND}}$ is quite similar and therefore omitted. □

Now we can give the following Lemma which states that the number of FPOs of a DS $\mathcal{S}$ meets the number of FPOs of its bipartite contraction.

Lemma 6. *Let $\mathcal{S}$ be a DS and let $\mathcal{S}' = (\mathcal{B}_G, F')$ be the bipartite contraction of $\mathcal{S}$. The number of FPOs of $\mathcal{S}$ equals the number of FPOs of $\mathcal{S}'$.*

Proof. By Lemmas 4 and 5, there is a surjective relation from the set of FPOs of $\mathcal{S}$ and the set of FPOs of $\mathcal{S}'$. Therefore, we simply need to show that this relation is injective. That is, whenever we have two distinct FPOs O_1 and O_2 of S, then the corresponding orbits O'_1 and O'_2 in S' are distinct and vice versa. This is straightforward to prove and is left to the reader. □

The following algorithm will compute the number of FPOs of $\mathcal{S} = (G, F)$. In order to manage the base case we admit null graphs G (that is, graphs in which $V(G) = \emptyset$) where we assume that null graphs have exactly one FPO (the empty vector).

Theorem 1. *The procedure* COUNT *(Algorithm 1) correctly computes the number of FPOs in an {AND, OR}-DS with symmetric positive dependency graph.*

Proof. Let G be the symmetric positive dependency graph of an {AND, OR}-DS $\mathcal{S}$. By Lemma 6, we simply need to count the number of FPOs in the bipartite contraction $\mathcal{S}'$ of $\mathcal{S}$. We prove the correctness of the algorithm by induction on the number n of vertices of $\mathcal{B}_G$.

Base. If $n = 1$ then there is only one connected component in $\mathcal{B}_G$, that is, either all the vertices in G are OR vertices or all of them are AND vertices. By [2], there are only 2 FPOs in the DS. In this case, the algorithm will make two recursive calls with empty graphs as arguments, so returning 1 + 1 = 2. Thus, in the base case the algorithm is correct.

Induction. Let us suppose the algorithm correctly computes the number of FPOs of these DSs having graphs up to $n-1$ vertices. Let $\mathcal{B}_G$ be a graph with n vertices and let $v \in V(\mathcal{B}_G)$ be an OR vertex (v being an AND vertex requires so similar reasoning that it is omitted). In order to prove this part it is sufficient to show (1) a one-to-one correspondence between all FPOs O of the contraction of $\mathcal{S}$ to $\mathcal{B}_G$ in which $o_v = 1$, and all FPOs of the contraction of $\mathcal{S}$ to $\mathcal{B}_G - \{v\}$ and (2) a one-to-one correspondence between all FPOs O of the contraction of $\mathcal{S}$ to $\mathcal{B}_G$ in which $o_v = 0$, and all FPOs of the contraction of $\mathcal{S}$ to $\mathcal{B}_G - N_{\mathcal{B}_G}[v]$.

Proof of (1). Let $\mathcal{B}' = \mathcal{B}_G - \{v\}$ and let O' be an FPO of the contraction of $\mathcal{S}$ to $\mathcal{B}'$ (FPO of $\mathcal{S}$ over $\mathcal{B}'$, to abbreviate). Let O be an orbit of $\mathcal{B}_G$ obtained from O' by setting $O_{V(\mathcal{B}')} = O'$ and setting $o_v = 1$. We show that O is an FPO of $\mathcal{S}$ over $\mathcal{B}_G$. In fact, let $u \in N_{\mathcal{B}_G}(v)$ since $\mathcal{B}_G$ is bipartite and v is an OR vertex, u must be an AND vertex. If $o'_u = 0$, this means that $o_u(t) = o_u(t+1) = 0$. Otherwise, if $o'_u = 1$ this means that, $\forall w \in N_{\mathcal{B}'}(u)$, we have that $o'_w = 1$. Since $o_v = 1$, for all $w \in N_{\mathcal{B}_G}(u)$ we have that $o_w = 1$. All the vertices not belonging to $N_{\mathcal{B}_G}[v]$ have the same neighborhood in $\mathcal{B}_G$ and in $\mathcal{B}'$. Thus, their value will not change with time t. This proves that O is an FPO of $\mathcal{B}_G$. To finish, let us consider O, an FPO O of $\mathcal{S}$ over $\mathcal{B}_G$ in which $o_v = 1$, by Lemma 1, $O_{V(\mathcal{B}')}$ is an FPO of $\mathcal{S}$ over $\mathcal{B}'$.

Proof of (2). Let $\mathcal{B}' = \mathcal{B}_G - N_{\mathcal{B}_G}[v]$. Let O' be an FPO of $\mathcal{S}$ over $\mathcal{B}'$ and let O be an orbit of $\mathcal{B}_G$ obtained from O' by setting $O_{V(\mathcal{B}')} = O'$ and setting $o_u = 0\ \forall u \in N_{\mathcal{B}_G}[v]$. Clearly each vertex in $u \in N_{\mathcal{B}_G}(v)$ is an AND vertex, thus $o_u(t) = o_u(t+1) = 0$. v is adjacent in $\mathcal{B}_G$ to only vertices u such that $o_u = 0$ then $o_v(t) = o_v(t+1) = 0$. A vertex $w \neq v$ adjacent to a vertex in $N_{\mathcal{B}_G}(v)$ will not change its value. In fact, first we note that w is an OR vertex. Thus, if $o_w(t) = 1$ then $o_w(t) = o_w(t+1)$. Otherwise, if $o_w(t) = 0$, this means that w is adjacent in $\mathcal{B}'$ to only vertices z with $o'_z = 0$ and so remains in $\mathcal{B}_G$, since for all $u \in N_{\mathcal{B}_G}(v)$, $o_u = 0$. All the vertices neither in $N_{\mathcal{B}_G}[v]$ nor adjacent to any vertex in $N_{\mathcal{B}_G}[v]$ have their neighborhood which the same in $\mathcal{B}_G$ and $\mathcal{B}'$. Thus, their value will not change with time t. So that O is an FPO of $\mathcal{S}$ over $\mathcal{B}_G$. To finish, let us consider O, an FPO O of $\mathcal{S}$ over $\mathcal{B}_G$ in which $o_v = 0$, by Lemma 1, $O_{V(\mathcal{B}')}$ is an FPO of $\mathcal{S}$ over $\mathcal{B}'$.

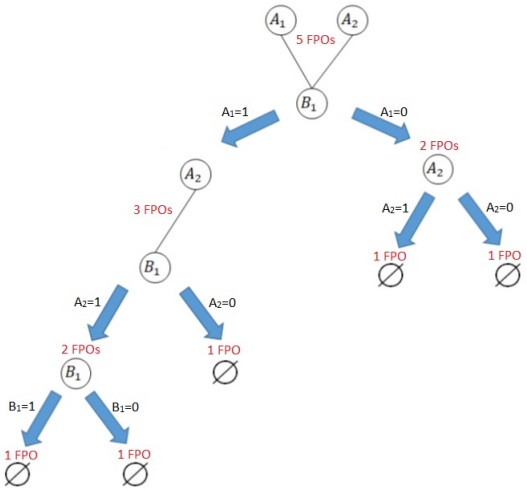
As we said in the beginning of the proof, we may analogously prove that if v is an AND vertex then there is (1) a one-to-one correspondence between all FPOs O of the contraction of $\mathcal{S}$ to $\mathcal{B}_G$ in which $o_v = 0$, and all FPOs of the contraction of $\mathcal{S}$ to $\mathcal{B}_G - \{v\}$ and (2) a one-to-one correspondence between all FPOs O of the contraction of $\mathcal{S}$ to $\mathcal{B}_G$ in which $o_v = 1$, and all FPOs of the contraction of $\mathcal{S}$ to $\mathcal{B}_G - N_{\mathcal{B}_G}$. This completes the proof. □

Figure 2 shows the recursive tree of calls of the Algorithm 1 applied to the bipartite contraction of the graph G of Figure 1. Numbers in red by the graphs compute the number of FPOs of the recursive call taking as argument the corresponding graph, so that the former call over the graph $\mathcal{B}_G$ appears on the top and outputs a total of 5 FPOs. The table on the right lists all the FPOs of $\mathcal{B}_G$ as the labels of the blue arrows show.

Algorithm 1 Computing the number of FPOs of $\mathcal{S} = (G, F)$, an {AND, OR}-DS whose dependency graph G is symmetric positive

Input: The AND-OR bipartition $\mathcal{B}_G$ of G
Output: The number of FPOs of $\mathcal{S}$

```
1: procedure COUNT(B_G)
2:     if V(B_G) = ∅ then return 1
3:     let v ∈ V(B_G)
4:     FP ←COUNT(B_G − {v})+COUNT(B_G − N_{B_G}[v])
5:     return FP
6: end procedure
```



A_1	A_2	B_1
0	0	0
0	1	0
1	0	0
1	1	0
1	1	1

Figure 2. Algorithm 1 performing over the bipartite contraction of the graph G of Figure 1. In addition, the table listing all the FPOs of $\mathcal{B}_G$.

3. Algorithm for Listing All FPO in AND-OR Symmetric PDS

In this section, we provide a lower bound on Algorithm 1 and, based on it, we present an algorithm for listing all the FPOs of an AND-OR symmetric PDS.

We have the following:

Lemma 7. *Let S be a DS and let $\mathcal{S}' = (\mathcal{B}_G, F')$ be the bipartite contraction of $\mathcal{S}$. If (A, B) is the bipartition of $\mathcal{B}_G$, then the number of FPOs of $\mathcal{S}$ is $FP \geq 2^{|A|} + 2^{|B|} - 1$.*

Proof. By Lemma 6, it is sufficient to prove that $FP \geq 2^{|A|} + 2^{|B|} - 1$ where FP is the number of FPOs in the bipartite contraction $\mathcal{S}'$ of $\mathcal{S}$. Let O be an orbit of $\mathcal{S}'$ such that for all $a \in A$ we have that $o_a = 1$. It is easy to check that if for any $b \in B$ we set $o_b = 1$ then $o_b(t) = o_b(t+1)$ since b is adjacent to vertices in A, while if $o_b = 0$ then $o_b(t) = o_b(t+1)$ since b is an AND vertex. We can build $2^{|B|}$ different FPOs. A similar argument applies if we set $o_b = 0$ for all $b \in B$. Since the FPO O in which $o_a = 1$ for all $a \in A$ and $o_b = 0$ for all $b \in B$ is present in both counts, we obtain the statement. □

From Lemma 7, the next corollary is immediately followed.

Corollary 1. *Let S be a DS and let $\mathcal{S}' = (\mathcal{B}_G, F')$ be the bipartite contraction of $\mathcal{S}$. If $n = |V(\mathcal{B}_G)|$, then the number of FPOs of $\mathcal{S}$ is not less than $2^{n/2+1}$.*

We also have the following result.

Lemma 8. *Let S be a DS and let $\mathcal{S}' = (\mathcal{B}_G, F')$ be the bipartite contraction of $\mathcal{S}$. If $\mathcal{B}_G$ is a complete bipartite graph $K_{n,m}$, then the number of FPOs of $\mathcal{S}$ equals $2^n + 2^m - 1$.*

Proof. By Lemma 6, it is sufficient to prove that $FP = 2^n + 2^m - 1$ where FP is the number of FPOs in the bipartite contraction $\mathcal{S}'$ of $\mathcal{S}$. Let (A, B) be the bipartition of $\mathcal{B}_G$ where A is the set of OR vertices. Suppose w.l.o.g. that $|A| = n$ and $|B| = m$. We show that given an FPO O of $\mathcal{S}'$, only one of the following two cases can happen: (i) $o_v = 1$ for all $v \in A$ or (ii) $o_u = 0$ for all $u \in B$. Suppose not and let O' be an FPO in which there is a vertex $v \in A$ and a vertex $u \in B$ such that $o'_v = 0$ and $o'_u = 1$. Since $\mathcal{B}_G$ is complete, v is adjacent to u so that $o'_v(t) \neq o'_v(t+1)$ and $o'_u(t) \neq o'_u(t+1)$ a contradiction to the hypothesis that O' is an FPO. Now, it is easy to see that in case (i), if we assign to each vertex in B any binary value, the orbit we obtain is a fixed point of $\mathcal{S}'$ (the same reasoning applies to case (ii)). Since there are, in case (i), 2^m of such orbits (2^n in case (ii)), the statement follows by the fact that the FPO O in which $o_v = 1$ for all $v \in A$ and $o_u = 0$ for all $u \in B$ is present in both counts. □

By Lemma 8, when the bipartite contraction of a DS is complete, listing all its FPOs requires a very simple algorithm for listing all the binary strings of a prefixed length. Therefore, in the following, we always assume that the bipartite contraction of a DS is not complete.

The Algorithm 1 can be modified in order to list in polynomial time all the FPOs of an {AND, OR}-DS whose dependency graph is symmetric positive as reported in Algorithm 2.

Algorithm 2 Listing all the FPOs of the bipartite contraction of an {AND, OR}-DS whose dependency graph is symmetric positive

Input: The AND-OR bipartite contraction $\mathcal{B}_G$ of G with bipartition (A, B) where A is the set of OR vertices and B the set of AND vertices of $\mathcal{B}_G$. An orbit O of $\mathcal{B}_G$ (initially o_v is undefined for all $v \in \mathcal{B}_G$)
Output: All FPOs of $\mathcal{B}_G$

```
1:  procedure FPO(B_G, O)
2:      if V(B_G) = ∅ then output O
3:      let v ∈ V(B_G)
4:      if v ∈ A then b ← 1 else b ← 0
5:      O' ← O
6:      o'_v ← b
7:      FPO(B_G − {v}, O')
8:      O'' ← O
9:      For all u ∈ N_{B_G}[v]
10:                 o''_u ← b
11:     FPO(B_G − N_{B_G}[v], O'')
12: end procedure
```

Theorem 2. *Let $\mathcal{S} = (G, F)$ be an {AND, OR}-DS whose dependency graph is symmetric positive. The Algorithm 2 lists all FPOs of the bipartite contraction of $\mathcal{S}$ in polynomial time using linear space.*

Proof. Let $\mathcal{S}' = (\mathcal{B}_G, F')$ be the bipartite contraction of $\mathcal{S}$ and (A, B) the bipartition of $\mathcal{B}_G$. Let $n = |A \cup B|$. The algorithm calls itself in line 7 and in line 11 and the argument, in each of the recursive call, is a graph with $n - 1$ vertices, in the worst case. The `for` cycle (lines 9-10) takes at most $O(\delta_{\mathcal{B}_G}(v))$ times. Now we can have two cases: (i) in $\mathcal{B}_G$ the degree $\delta_{\mathcal{B}_G}(v)$ of any vertex v is bounded by a constant c and (ii) $\delta_{\mathcal{B}_G}(v)$ is not bounded by constant, that is, we have that $\delta_{\mathcal{B}_G}(v) = O(n)$ so we can find a constant $0 < k < 1$ such that $\delta_{\mathcal{B}_G}(v) \leq kn$ for every $v \in V(\mathcal{B}_G)$.

Therefore, in case (i), the recursive relation of the time $T(n)$, the algorithm spends for listing all the FPOs is:

$$T(n) = T(n-1) + T(n-c) + c \leq 2T(n-1) + c \tag{1}$$

Assuming that $T(0) = 1$, clearly this recursion solves in $T(n) \leq (c+1)2^n - c$, so that the algorithm has $O(2^n)$ complexity. In case (ii), the recursive relation is

$$T(n) = T(n-1) + T(n-kn) + kn \tag{2}$$

So we have that

$$T(n) \leq 2T(n-1) + kn \tag{3}$$

The above recursion solves in $T(n) \leq k(2^{n+1} - n - 2)$ so that the algorithm has $O(2^n)$ complexity. By Corollary 1, we have that the number of FPOs is not less than $2^{n/2+1}$ and this completes the proof. □

As an example of the output of the algorithm, we refer to the table on the right hand side of Figure 2.

4. Conclusions

We presented an algorithm for counting all the FPOs of an {AND, OR}-DS whose dependency graph is symmetric positive. We gave a lower bound on that number. Furthermore, we adapted the algorithm in order to list all the FPOs of the system taking for it polynomial time on the length of the output, and linear space over the same parameter.

Author Contributions: Individual contributions are as follows. Conceptualization, M.M.; formal analysis, F.L.P.; investigation, M.M.; writing—original draft, M.M.; writing—review and editing, M.M. and F.L.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Aledo, J.A.; Martínez, S.; Valverde, J.C. Parallel Dynamical Systems over Graphs and Related Topics: A Survey. *J. Appl. Math.* **2015**, *2015*, 594294. [CrossRef]
2. Barrett, C.L.; Chen, W.Y.; Zheng, M.J. Discrete dynamical systems on graphs and Boolean functions. *Math. Comput. Simul.* **2004**, *66*, 487–497. [CrossRef]
3. Aledo, J.; Martínez, S.; Pelayo, F.; Valverde, J.C. Parallel discrete dynamical systems on maxterm and minterm Boolean functions. *Math. Comput. Model.* **2012**, *55*, 666–671. [CrossRef]
4. Aledo, J.A.; Martínez, S.; Valverde, J.C. Parallel dynamical systems over directed dependency graphs. *Appl. Math. Comput.* **2012**, *219*, 1114–1119. [CrossRef]
5. Aracena, J.; Richard, A.; Salinas, L. Maximum number of fixed points in AND–OR–NOT networks. *J. Comput. Syst. Sci.* **2014**, *80*, 1175–1190. [CrossRef]
6. Tošić, P.T.; Agha, G.A. On Computational Complexity of Counting Fixed Points in Symmetric Boolean Graph Automata. In *Unconventional Computation*; Calude, C.S., Dinneen, M.J., Păun, G., Pérez-Jímenez, M.J., Rozenberg, G., Eds.; Springer: Berlin/Heidelberg, Germany, 2005; pp. 191–205.
7. Paulevé, L.; Richard, A. Static Analysis of Boolean Networks Based on Interaction Graphs: A Survey. *Electron. Notes Theor. Comput. Sci.* **2012**, *284*, 93–104. [CrossRef]
8. Aledo, J.A.; Martinez, S.; Valverde, J.C. Parallel discrete dynamical systems on independent local functions. *J. Comput. Appl. Math.* **2013**, *237*, 335–339. [CrossRef]
9. Aledo, J.A.; Martinez, S.; Valverde, J.C. Parallel dynamical systems over special digraph classes. *Int. J. Comput. Math.* **2013**, *90*, 2039–2048. [CrossRef]
10. Aledo, J.A.; Diaz, L.G.; Martinez, S.; Valverde, J.C. On periods and equilibria of computational sequential systems. *Inf. Sci.* **2017**, *409–410*, 27–34. [CrossRef]
11. Barrett, C.L.; Hunt, M.; Marathe, M.V.; Ravi, S.S.; Rosenkrantz, D.J.; Stearns, R.E.; Tosic, P.T. *Gardens of Eden and Fixed Points in Sequential Dynamical Systems. Discrete Models: Combinatorics, Computation, and Geometry, DM-CCG 2001*; Cori, R., Mazoyer, J., Morvan, M., Mosseri, R., Eds.; Discrete Mathematics and Theoretical Computer Science: Paris, France, 2001; Volume AA, pp. 95–110.

12. Milano, M.; Roli, A. Solving the Satisfiability Problem Through Boolean Networks. In *AI*IA 99: Advances in Artificial Intelligence*; Lamma, E., Mello, P., Eds.; Springer: Berlin/Heidelberg, Germany, 2000; pp. 72–83.
13. TAMURA, T.; AKUTSU, T. Detecting a Singleton Attractor in a Boolean Network Utilizing SAT Algorithms. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.* **2009**, *92*, 493–501. [CrossRef]
14. Mezzini, M. Polynomial time algorithm for computing a minimum geodetic set in outerplanar graphs. *Theor. Comput. Sci.* **2018**, *745*, 63–74. [CrossRef]
15. Mezzini, M.; Moscarini, M. The contour of a bridged graph is geodetic. *Discret. Appl. Math.* **2016**, *204*, 213–215. [CrossRef]
16. Mezzini, M. On the geodetic iteration number of the contour of a graph. *Discret. Appl. Math.* **2016**, *206*, 211–214. [CrossRef]
17. Mezzini, M.; Moscarini, M. On the geodeticity of the contour of a graph. *Discret. Appl. Math.* **2015**, *181*, 209–220. [CrossRef]
18. Mezzini, M. Fully dynamic algorithm for chordal graphs with O(1) query-time and $O(n^2)$ update-time. *Theor. Comput. Sci.* **2012**, *445*, 82–92. [CrossRef]
19. Mezzini, M.; Moscarini, M. Simple algorithms for minimal triangulation of a graph and backward selection of a decomposable Markov network. *Theor. Comput. Sci.* **2010**, *411*, 958–966. [CrossRef]
20. Malvestuto, F.M.; Mezzini, M.; Moscarini, M. Computing simple-path convex hulls in hypergraphs. *Inf. Process. Lett.* **2011**, *111*, 231–234. [CrossRef]
21. Mezzini, M. Fast minimal triangulation algorithm using minimum degree criterion. *Theor. Comput. Sci.* **2011**, *412*, 3775–3787. [CrossRef]
22. Mezzini, M. On the complexity of finding chordless paths in bipartite graphs and some interval operators in graphs and hypergraphs. *Theor. Comput. Sci.* **2010**, *411*, 1212–1220. [CrossRef]
23. Mezzini, M. An O(mn2) Algorithm for Computing the Strong Geodetic Number in Outerplanar Graphs. *Discuss. Math. Graph Theory* **2020**. [CrossRef]