Computing Preimages and Ancestors in Reaction Systems

Roberto Barbuti$^{https://orcid.org/0000-0001-7028-4238}$, Anna Bernasconi$^{https://orcid.org/0000-0003-0263-5221}$, Roberta Gori$^{https://orcid.org/0000-0002-7424-9576}$, and Paolo Milazzo$^{https://orcid.org/0000-0002-7309-6424}$

Dipartimento di Informatica, Università di Pisa, Italy
{roberto.barbuti,anna.bernasconi,roberta.gori,paolo.milazzo}@unipi.it

Abstract. In reaction systems, preimages and n-th ancestors are sets of reactants leading to the production of a target set of products in either 1 or n steps, respectively. Many computational problems on preimages and ancestors, such as finding all minimum-cardinality n-th ancestors, computing their size, or counting them, are intractable. In this paper we propose a characterization of n-th ancestors as a Boolean formula, and we define an operator able to compute such a formula in polynomial time. Our formula can be exploited to solve all preimage and ancestors problems and, therefore, it can be directly used to study their complexity. In particular, we focus on two problems: (i) deciding whether a preimage/n-th ancestor exists (ii) finding a preimage/n-th ancestor of minimal size. Our approach naturally leads to the definition of classes of systems for which such problems can be solved in polynomial time.

Keywords: Reaction Systems, Ancestor computation, Computational Complexity, Causality Relations.

1 Introduction

Inspired by natural phenomena, many new computational formalisms have been introduced to model different aspects of biology. Basic chemical reactions inspired the reaction systems, a *qualitative* modeling formalism introduced by Ehrenfeucht and Rozenberg [16, 11]. It is based on two opposite mechanisms, namely *facilitation* and *inhibition*. Facilitation means that a reaction can occur only if all its reactants are present, while inhibition means that the reaction cannot occur if any of its inhibitors is present. A rewrite rule of a reaction system (called *reaction*) is hence a triple (R, I, P), where R, I and P are sets of objects representing *reactants*, *inhibitors* and *products*, respectively, of the modeled chemical reaction. A *reaction system* is represented by a set of reactions having such a form, together with a (finite) support set S containing all of the objects that can appear in a reaction. The state of a reaction system consists of a finite set of objects describing the biological entities that are present in the real system being modeled. The presence of an object in the state expresses the fact that the corresponding biological entity, in the real system being modeled, is present in a number of copies as high as needed. This is the *threshold supply* assumption and characterizes reaction systems.

A reaction system evolves by means of the application of its reactions. The threshold supply assumption ensures that different reactions never compete for their reactants, and hence all the applicable reactions in a step are always applied. The application of a set of reactions results in the introduction of all of their products in the next state of the system.

The main advantages of investigating reaction systems is that they have a clean computational model allowing precise formal analisys and they can be considered as reference for other computing system (e.g. [17, 1]).

Computational complexity of some problems related to the dynamics of reaction systems has been extensively studied (e.g. in [21, 20, 17, 14, 15]). In [14, 15], Dennunzio et al. introduced the concept of preimage and n-th ancestor. Roughly speaking, a n-th ancestor is a set of objects that lead to the production of a target set of objects after n evolution steps, while a preimage is a 1-th ancestor. The authors studied the complexity of several problems related to n-th ancestors by defining reductions between well known hard problems and the corresponding n-th ancestor problem. They proved that finding a minimal size preimage or ancestor, computing their size, or counting them are all intractable problems.

In this paper we propose a *constructive method* to reason on preimages and n-th ancestors. Indeed, we define a formula able to characterize all n-th ancestors of a given set of objects. Such a formula is obtained by revising the idea of formula based predictor introduced in [3, 2, 4, 6, 5, 7]. A formula based predictor is a logic formula that exactly characterizes all states leading to a given product in a given number of steps. It allows the study of all causal dependencies of one object from the others and therefore enhances previous works on causality in reaction systems [12], systems biology [18, 8, 9] and natural computing [13].

Following the same approach, here we define a n-ancestors formula that fully characterizes all n-th ancestors. Moreover, we define an operator able to compute the formula in polynomial time. To exploit this formula to solve problems as deciding the existence of n-th ancestors or the computation of a minimal size preimage or ancestor, one more step is needed. Such a step is a logic transformation of the formula into a Disjunctive Normal Form (DNF), possibily minimized with respect to the number of terms (i.e., conjunctions) or to the number of propositional symbols occurring in it.

The proposed approach allows us to study the complexity of the preimage and n-th ancestor problems in a constructive way, and to identify classes of reaction systems for which these problems can be solved in polynomial time.

The paper is organized as follows. Section 2 introduces (Closed) reaction systems, preimage and n-th ancestor. Section 3 presents some preliminary notions. The definition of n-th ancestors formulas is given in Section 4 together with an effective operator to compute them. In Section 5 we bound the complexity of the

n-th ancestors formula. Finally, Section 6 lists some conditions under which the existence and minimal size of ancestors can be computed in polynomial time.

2 Closed Reaction Systems

In this section we recall the basic definition of reaction system [16, 11]. Let S be a finite set of symbols, called objects. A *reaction* is formally a triple (R, I, P) with $R, I, P \subseteq S$, composed of *reactants* R, *inhibitors* I, and *products* P. Reactants and inhibitors $R \cup I$ of a reaction are collectively called *resources* of such a reaction, and we assume them to be disjoint ($R \cap I = \emptyset$), otherwise the reaction would never be applicable. The set of all possible reactions over a set S is denoted by $\mathrm{rac}(S)$. Finally, a *reaction system* is a pair $\mathcal{A} = (S, A)$, where S is a finite support set, and $A \subseteq \mathrm{rac}(S)$ is a set of reactions.

The state of a reaction system is described by a set of objects. Let $a = (R_a, I_a, P_a)$ be a reaction and T a set of objects. The result $\mathrm{res}_a(T)$ of the application of a to T is either P_a, if T separates R_a from I_a (i.e. $R_a \subseteq T$ and $I_a \cap T = \emptyset$), or the empty set $\emptyset$ otherwise. The application of multiple reactions at the same time occurs without any competition for the used reactants (*threshold supply assumption*). Therefore, each reaction which is not inhibited can be applied, and the result of the application of multiple reactions is cumulative. Formally, given a reaction system $\mathcal{A} = (S, A)$, the result of application of $\mathcal{A}$ to a set $T \subseteq S$ is defined as $\mathrm{res}_{\mathcal{A}}(T) = \mathrm{res}_A(T) = \bigcup_{a \in A} \mathrm{res}_a(T)$.

An important feature of reaction systems is the assumption about the *non-permanency* of objects: the objects carried over to the next step are only those produced by reactions. All the other objects vanish, even if they are not involved in any reaction.

Given a initial set D_0 the semantics of a *Closed* reaction system can be simply defined as the *result sequence*, $\delta = D_1, \ldots, D_n$ where each set D_i, for $i \geq 1$, is obtained from the application of reactions $\mathcal{A}$ to the state obtained at the previous step D_{i-1} ; formally $D_i = res_{\mathcal{A}}(D_{i-1})$ for all $1 \leq i < n$. The sequence of states of the reaction system coincides with the result sequence $\delta = D_1, \ldots, D_n$. In [14, 15], the authors introduce the idea of *preimage* and n-th *ancestor*. For simplicity, we define them for a single product s.

Definition 1. *Let $\mathcal{A} = (S, A)$ be a r.s. and $s \in S$. A set D_0 is a n-th* ancestor *of s if $s \in res^{(n)}_{\mathcal{A}}(D_0)$. D_0 is called a* preimage *of s if D_0 is a 1-th ancestor of s.*

The same concepts can be naturally extended to sets of products.

3 Causal Predicates in Reaction Systems

In our formulas, we use objects of reaction systems as propositional symbols. Formally, we introduce the set F_S of propositional formulas on S defined in the standard way: $S \cup \{true, false\} \subseteq F_S$ and $\neg f_1, f_1 \vee f_2, f_1 \wedge f_2 \in F_S$ if $f_1, f_2 \in F_S$. The propositional formulas F_S are interpreted with respect to subsets of the objects $C \subseteq S$. Intuitively, $s \in C$ denotes the presence of element s and therefore the truth of the corresponding propositional symbol. The complete definition of the formula satisfaction relation is as follows.

Definition 2. *Let $C \subseteq S$ for a given set of objects S. Given a propositional formula $f \in F_S$, the* satisfaction relation *$C \models f$ is inductively defined as follows:*

$C \models s$ *iff* $s \in C$, $\quad C \models true$,
$C \models \neg f'$ *iff* $C \not\models f'$, $\quad C \models f_1 \vee f_2$ *iff either* $C \models f_1$ *or* $C \models f_2$,
$C \models f_1 \wedge f_2$ *iff* $C \models f_1$ *and* $C \models f_2$.

In the following, $\equiv$ stands for the logical equivalence on propositional formulas F_S. Moreover, given a formula $f \in F_S$, with $atom(f)$ we denote the set of propositional symbols that appear in f.

Given a formula f, a Disjunctive Normal Form (DNF) of f can be computed by applying the following procedure:

1. Put the negations next to the atomic objects using De Morgan's laws;
2. Put the conjunctions within the disjunctions using the distributive law;
3. Simplify the obtained formula using the the idempotent, negation, domination and negation laws.

Alternatively, we can construct the complete DNF of f by constructing the truth table of f and representing with a conjunction all rows that have a truth value 1. It is worth noting that both methods are exponential in the worst case. Indeed, in the first method the application of the distributive laws (step 2) can be exponential; while in the second method the construction of the truth table is exponential in the number of variables of f.

Any DNF formulation of the formula allows us to efficiently solve problems such as determining the existence of a preimage or of a n-th ancestor (see Section 6 for more details) or to find the minimal-cardinality preimage or n-th ancestor. However, altough not strictly necessary, it can be convenient to consider a compact DNF representation of f so that it can be more easily verified. This requires a

Logic minimization step: further simplify the formula, in exact or heuristic way, in order to derive a DNF minimal with respect to the number of terms (i.e., conjunctions) or to the number of literals occurring in it, or any other given cost metric.

This last step is computationally expensive, as logic minimization is an NP-hard problem[1] [19, 22]. However, since near minimum solutions are sufficient, the logic minimization step can be performed applying heuristic methods to produce solutions that are near to the optimum in a relatively short time. In particular, in our setting, we are interested in deriving a compact logic expression containing only *essential* propositional symbols, i.e., symbols on which the expression actually depends. Thus, in this context, we can apply heuristic techniques (e.g.,

[1] More precisely, the decision version of the problem of finding a minimal DNF representation of a Boolean function f starting from its truth table is NP-complete, while it becomes NP^{NP}-complete starting from a DNF for f.

the ESPRESSO heuristic minimizer [10]) to produce near minimal *prime* and *irredundant* DNF formulas, i.e., DNF where each conjunction corresponds to a prime implicant[2] of the function represented by the expression (primality), and no conjunction can be deleted without changing the function represented by the expression (irredendancy). Indeed, non essential propositional symbols cannot appear in any prime implicant of a given Boolean function.

Let us denote with $min(f)$ the DNF obtained after the exact or heuristic logic minimization step. For any formula $f \in F_S$, $min(f)$ is equivalent to f and is minimal with respect to the number of terms (i.e., conjunctions) or to the number of literals occurring in it, or to any other chosen cost function. Thus, for any reasonable cost function we have $f \equiv min(f)$ and $atom(min(f)) \subseteq atom(f)$ and there exists no formula f' such that $f' \equiv_l f$ and $atom(f') \subset atom(min(f))$.

The causes of an object in a reaction system are defined by a propositional formula on the set of objects S. First of all we define the *applicability predicate* of a reaction a as a propositional logic formula on S describing the requirements for applicability of a, namely that all reactants have to be present and inhibitors have to be absent. This is represented by the conjunction of all atomic formulas representing reactants and the negations of all atomic formulas representing inhibitors of the considered reaction.

Definition 3. *Let $a = (R, I, P)$ be a reaction with $R, I, P \subseteq S$ for a set of objects S. The* applicability predicate *of a, denoted by $ap(a)$, is defined as follows:* $ap(a) = \left(\bigwedge_{s_r \in R} s_r\right) \wedge \left(\bigwedge_{s_i \in I} \neg s_i\right)$.

The *causal predicate* of a given object s is a propositional formula on S representing the conditions for the production of s in one step, namely that at least one reaction having s as a product has to be applicable.

Definition 4. *Let $\mathcal{A} = (S, A)$ be a r.s. and $s \in S$. The* causal predicate *of s in $\mathcal{A}$, denoted by $cause(s, \mathcal{A})$ (or $cause(s)$, when $\mathcal{A}$ is clear from the context), is defined as follows*[3]*:* $cause(s, \mathcal{A}) = \bigvee_{\{(R,I,P) \in A \mid s \in P\}} ap\left((R, I, P)\right)$.

We introduce a simple reaction system as running example.

Example 5. Let $\mathcal{A}_1 = (\{A, \dots, G\}, \{a_1, a_2, a_3\})$ be a reaction system with

$$a_1 = (\{A\}, \{\}, \{B\}) \qquad a_2 = (\{C, D\}, \{\}, \{E, F\}) \qquad a_3 = (\{G\}, \{B\}, \{E\}).$$

The *applicability predicates* of the reactions are $ap(a_1) = A$, $ap(a_2) = C \wedge D$ and $ap(a_3) = G \wedge \neg B$. Thus, the *causal predicates* of the objects are

$$cause(A) = cause(C) = cause(D) = cause(G) = false,$$
$$cause(B) = A,\ cause(F) = C \wedge D,\ cause(E) = (G \wedge \neg B) \vee (C \wedge D).$$

[2] A prime implicant of a Boolean function f is a conjunction of literals, that implies f and s. t. removing any literal results in a new conjunction that does not imply f.

[3] We assume that $cause(s) = false$ if there is no $(R, I, P) \in A$ such that $s \in P$.

4 Characterizing the n-th ancestors

We aim to define a formula characterizing all the initial sets D_0 that lead to the production of a given product $s \in S$ after exactly n steps. Note that the formula for the n-th ancestor of a set of products $\{s_1, s_2, ..., s_m\} \subseteq S$ can be obtained by combining in conjunction all the n-ancestors formulas for each s_i with $i \in \{1, ..., m\}$ (see Corollary 10).

We base our new definitions on the well know notions of *formula based* and *specialized formula predictors*, originally presented in [3, 2, 4, 5], that characterize all causes of an object in a given number of steps. Following this approach, all n-th ancestors of an object s are characterized by a propositional formula f, i.e., they are all initial sets D_0 that satisfy f according to the satisfaction relation defined in Def. 2.

Definition 6 (n-Ancestors Formula). *Let $\mathcal{A} = (\mathcal{S}, \mathcal{A})$ be a r.s., $s \in S$ and $f \in F_S$ a propositional formula. We say that formula f* is a n-ancestors formula *of s if it holds that $D_0 \models f \Leftrightarrow s \in D_n$.*

Note that if f is a n-ancestors formula of s and $f' \equiv f$ then also f' is a n-ancestors formula of s. Among all the equivalent formulas, it is convenient to choose the one containing the minimal number of propositional symbols, so that they do not contain inessential objects. This is formalized by the following approximation order on F_S.

Definition 7 (Approximation Order). *Given $f_1, f_2 \in F_S$ we say that $f_1 \sqsubseteq_f f_2$ if and only if $f_1 \equiv f_2$ and $atom(f_1) \subseteq atom(f_2)$.*

It can be shown that there exists a *unique equivalence class* of n-ancestors formulas of s that is minimal w.r.t. the order $\sqsubseteq_f$.

We now define an operator `Anc` that allows n-ancestors formulas to be effectively computed.

Definition 8. *Let $\mathcal{A} = (S, A)$ be a r.s. and $s \in S$. We define a function* $\texttt{Anc} : S \times \mathbb{N} \to F_S$ *as follows:* $\texttt{Anc}(s, n) = \texttt{Anc}_a(cause(s), n-1)$*, where the auxiliary function* $\texttt{Anc}_a : F_S \times \mathbb{N} \to F_S$ *is recursively defined as follows:*

$$\begin{array}{ll}
\texttt{Anc}_a(s, 0) = s & \texttt{Anc}_a(s, i) = \texttt{Anc}_a(cause(s), i-1) \quad \textit{if } i > 0 \\
\texttt{Anc}_a((f'), i) = (\texttt{Anc}_a(f', i)) & \texttt{Anc}_a(f_1 \vee f_2, i) = \texttt{Anc}_a(f_1, i) \vee \texttt{Anc}_a(f_2, i) \\
\texttt{Anc}_a(\neg f', i) = \neg\texttt{Anc}_a(f', i) & \texttt{Anc}_a(f_1 \wedge f_2, i) = \texttt{Anc}_a(f_1, i) \wedge \texttt{Anc}_a(f_2, i) \\
\texttt{Anc}_a(true, i) = true & \texttt{Anc}_a(false, i) = false
\end{array}$$

The function $\texttt{Anc}(\_, n)$ gives a n-ancestors formula that, in general, is not in DNF form and may not be minimal w.r.t. to $\sqsubseteq_f$. For this purpose we could apply heuristic techniques to produce *prime* and *irredundant* quasi minimal DNF, that are guaranteed to be minimal w.r.t. to $\sqsubseteq_f$.

Theorem 9. *Let $\mathcal{A} = (S, A)$ be a r.s.. For any object $s \in S$,*

- $\texttt{Anc}(s, n)$ *is the* n-ancestors formula *of s;*
- $min(\texttt{Anc}(s, n))$ *is the* n-ancestors formula *of s and is* minimal *w.r.t. $\sqsubseteq_f$.*

The proof of the previous result can be obtained by revisiting the proof of Theorem 4.4 and Corollary 4.7 in [3]. The previous result extends naturally to sets as follows.

Corollary 10. *Let $\mathcal{A} = (S, A)$ be a r.s.. Given a set of objects $\{s_1, ..., s_m\} \subseteq S$,*

- $\bigwedge_{i\in\{1,...,m\}} \texttt{Anc}(s_i, n)$ *is a* n-ancestors formula *of* $\{s_1, ..., s_m\}$*;*
- *min* $\left(\bigwedge_{i\in\{1,...,m\}} \texttt{Anc}(s_i, n)\right)$ *is a* n-ancestors formula *of* $\{s_1, ..., s_m\}$ *and it is* minimal *w.r.t.* $\sqsubseteq_f$*.*

These constructive (minimal) characterizations of preimages and n-th ancestors can be exploited for solving computational problems studied in [14, 15]. In particular, in Section 6 we will use n-ancestors formulas for studying the complexity of checking the existence of preimages and n-th ancestors, and of computing minimal preimages and n-th ancestors.

Example 11. Let us consider again the reaction system $\mathcal{A}_1$ of Ex. 5. Assume we are interested in the 1-ancestors formula of E. Hence, we calculate it by applying the function Anc:

$$\begin{aligned}\texttt{Anc}(E, 1) &= \texttt{Anc}_a\big((G \wedge \neg B) \vee (C \wedge D), 0\big)\\ &= \big(\texttt{Anc}_a(G, 0) \wedge \neg\texttt{Anc}_a(B, 0)\big) \vee \big(\texttt{Anc}_a(C, 0) \wedge \texttt{Anc}_a(D, 0)\big)\\ &= (G \wedge \neg B) \vee (C \wedge D)\end{aligned}$$

An initial set D_0 satisfies $\texttt{Anc}(E, 1)$ iff the execution of the reaction system starting from D_0 leads to the production of object E after 1 step. Furthermore, in this case the obtained formula is also minimal given that $min(\texttt{Anc}(E, 1)) = \texttt{Anc}(E, 1)$ since $\texttt{Anc}(E, 1)$ is already in minimal DNF. The 1-ancestors (or preimages) of E are the sets D_0 satisfying $\texttt{Anc}(E, 1)$. They are all possible sets containing either G but not B, or both C and D. Note that the 2-ancestor formula of E is equal to $false$. Indeed,

$$\begin{aligned}\texttt{Anc}(E, 2) &= \texttt{Anc}_a\big((G \wedge \neg B) \vee (C \wedge D), 1\big)\\ &= \big(\texttt{Anc}_a(G, 1) \wedge \neg\texttt{Anc}_a(B, 1)\big) \vee \big(\texttt{Anc}_a(C, 1) \wedge \texttt{Anc}_a(D, 1)\big)\\ &= (\texttt{Anc}_a(false, 0) \wedge \neg\texttt{Anc}_a(A, 0)) \vee (\texttt{Anc}_a(false, 0) \wedge \texttt{Anc}_a(false, 0))\\ &= (false \wedge \neg A) \vee (false \wedge false)\\ &\equiv false\end{aligned}$$

This means that there does not exist any 2nd ancestor of E, that is no D_0 can lead to E in two steps. Of course, also any n-ancestors formula of E with $n > 2$ is equal to $false$. Therefore, we can conclude that there does not exist any n-th ancestor of E for any $n > 2$.

Example 12. Let us consider now the reaction system
$\mathcal{A}_2 = (\{A, \ldots, L\}, \{a_1, \ldots, a_8\})$ with the following reaction rules:

$a_1 = (\{A\}, \{B\}, \{C\})$	$a_2 = (\{C\}, \{\}, \{E, I\})$	$a_3 = (\{G, B\}, \{\}, \{D\})$
$a_4 = (\{B\}, \{\}, \{B\})$	$a_5 = (\{H, B\}, \{\}, \{D\})$	$a_6 = (\{E, D\}, \{\}, \{F\})$
$a_7 = (\{I\}, \{\}, \{G\})$	$a_8 = (\{L\}, \{\}, \{H\})$	

Assume we are interested in the 1-ancestors of F. We obtain the 1-ancestors formula $\texttt{Anc}(F,1) = E \wedge D$, expressing that any set containing $\{E, D\}$ is a 1-ancestors (preimage) of F. Looking for the 2-nd ancestors of F, we obtain

$$\begin{aligned}\texttt{Anc}(F,2) &= \texttt{Anc}_a\big((E \wedge D),1\big) = \big(\texttt{Anc}_a(E,1) \wedge \texttt{Anc}_a(D,1)\big)\\ &= \texttt{Anc}_a(C,0) \wedge \texttt{Anc}_a((G \wedge B) \vee (H \wedge B),0)\\ &= C \wedge ((\texttt{Anc}_a(G,0) \wedge \texttt{Anc}_a(B,0)) \vee (\texttt{Anc}_a(H,0) \wedge \texttt{Anc}_a(B,0)))\\ &= C \wedge ((G \wedge B) \vee (H \wedge B)).\end{aligned}$$

Note that $min(\texttt{Anc}(F,2)) = (C \wedge G \wedge B) \vee (C \wedge H \wedge B)$ in this case is simply obtained by applying the distributive law that gives an already minimized DNF.

The 2-ancestors formula expresses that any set containing either $\{C, G, F\}$ or $\{C, H, B\}$ is a 2-nd ancestors of F. Instead, as regards 3-ancestors we have

$$\begin{aligned}\texttt{Anc}(F,3) &= \texttt{Anc}_a\big((E \wedge D),2\big) = \big(\texttt{Anc}_a(E,2) \wedge \texttt{Anc}_a(D,2)\big)\\ &= \texttt{Anc}_a(C,1) \wedge \texttt{Anc}_a((G \wedge B) \vee (H \wedge B)),1)\\ &= \texttt{Anc}_a(C,1) \wedge ((\texttt{Anc}_a(G,1) \wedge \texttt{Anc}_a(B,1) \vee (\texttt{Anc}_a(H,1) \wedge \texttt{Anc}_a(B,1)))\\ &= \texttt{Anc}_a(A \wedge \neg B,0) \wedge ((\texttt{Anc}_a(I,0) \wedge \texttt{Anc}_a(B,0) \vee (\texttt{Anc}_a(L,0) \wedge \texttt{Anc}_a(B,0)))\\ &= A \wedge \neg B \wedge ((I \wedge B) \vee (L \wedge B))\end{aligned}$$

This time $min(\texttt{Anc}(F,3)) = false$ therefore, we can be sure that it does not exist any n-th ancestor of F for any $n > 2$.

5 Analysis of the structure of n-ancestors formulas

In this section we analyze the structure of the formula resulting from our operator, giving an upper bound to its size and to its depth, i.e., the levels of nesting of AND and OR operators.

To this aim, given a reaction system $\mathcal{A} = (S, A)$, we first define two auxiliary notions: the maximum number of products and inhibitors in a rule of the system, denoted $cp(A)$, and the maximum number of rules sharing a product, denoted $mp(A)$. In the formal definition, $|S|$ indicates the cardinality of the set S.

Definition 13. *Given a reaction system $\mathcal{A} = (S, A)$, let*

$$\begin{aligned}&cp(A) = max\{|R| + |I| \mid (R, I, P) \in A\},\\ &mp(A) = max\{|p(A, s)| \mid s \in S\} \textit{ where } p(A, s) = \{(R, I, P) \in A \mid s \in P\}.\end{aligned}$$

First observe that for each $s \in S$ the size of the formula $cause(s)$ in terms of number of literals is at most $cp(A) \cdot mp(A)$. Computing the n-ancestors formula requires n steps. At the first step, the formula computed by our operator is $cause(s)$, for some s, whose maximal size is at most $cp(A) \cdot mp(A)$. At the second step, each one of the $cp(A) \cdot mp(A)$ literals of the previous formula has to be substituted with its causes, obtaining a new formula whose size is at most $(cp(A) \cdot mp(A))^2$, and so on. Hence, the size of the resulting formula $\texttt{Anc}(s, n)$ is at most $(cp(A) \cdot mp(A))^n$ for each $s \in S$, namely the size of $\texttt{Anc}(s, n)$ is polynomial in $cp(A)$ and $mp(A)$, as long as n has a constant value. Therefore, the size of the n-ancestors formula for the set product $\{s_1, s_2, ..., s_m\} \subseteq S$ is $m \cdot (cp(A) \cdot mp(A))^n$ which is polynomial in m, $cp(A)$ and $mp(A)$.

Let us now evaluate the depth of the formula. To this aim, the idea is to measure the level of nesting of $\wedge$-$\vee$ operators. Intuitively, formula A is level 0, formulas $A \wedge B \wedge C$, $A \wedge (B \wedge C)$ and $A \vee B$ are level 1, formulas $A \wedge (B \wedge C) \wedge (C \wedge D)$ and $A \vee (B \wedge C)$ are level 2, and so on.

Definition 14. *Let $f \in F_S$, we call* nesting level *of f the maximum depth of its representation through a AND-OR tree.*

In order to bound the nesting level of $\mathtt{Anc}(s, n)$, we define the following:

Definition 15. *Given a reaction system $\mathcal{A} = (S, A)$. We define*

$$c(A) = \begin{cases} 1 & \text{if } cp(A) > 1; \\ 0 & \text{otherwise.} \end{cases} \qquad p(A) = \begin{cases} 1 & \text{if } mp(A) > 1; \\ 0 & \text{otherwise.} \end{cases}$$

The next result bounds the nesting level of the formula $\mathtt{Anc}(s, n)$, for $s \in S$.

Theorem 16. *Let $\mathcal{A} = (S, A)$ be a reaction system. For each $s \in S$, the nesting level of the formula $\mathtt{Anc}(s, n)$, characterizing the n-th ancestors of s, is at most $n \cdot (c(A) + p(A))$.*

Proof. Follows immediately from the definitions of $\mathtt{Anc}$ and of *cause*. $\square$

For a set $\{s_1, s_2, ..., s_m\} \subseteq S$, we have the following result.

Corollary 17. *Let $\mathcal{A} = (S, A)$ be a reaction system. The nesting level of the formula $\bigwedge_{i \in \{1,...,m\}} \mathtt{Anc}(s_i, n)$, characterizing the n-th ancestors of $\{s_1, s_2, ..., s_m\}$, is at most $1 + n \cdot (c(A) + p(A))$.*

6 Complexity of the existence and minimal size ancestors

Here we focus on the problem of existence and minimal size n-th ancestors. The first problem consists in establishing whether a n-th ancestor of a given product exists, while the second one deals with finding a n-th ancestor with a minimal number of objects. We apply our constructive characterization to prove that in some particular cases these problems can be solved in polynomial time. It is worth noting that starting from a formula in DNF, both problems can be solved in polynomial time. Indeed, the satisfability of a formula in DNF can be checked in a time that is linear in the size of the formula[4]; analogously, an ancestor of minimal size can be found in linear time scanning the DNF form in order to select the conjunction with the minimal number of positive literals. Transforming a formula into DNF may require exponential time. Therefore, our idea is to identify conditions that guarantee that our operator returns a formula that it is already in DNF. From Corollary 17 it follows that the size of such a formula is polynomial.

The first result is related to preimages.

[4] A formula in DNF is satisfiable if and only if at least one of its conjunctions is satisfiable; and a conjunction is satisfiable if and only if it does not contain both a symbol x and its complement $\neg x$.

Theorem 18. *Let $\mathcal{A} = (S, A)$ be a reaction system. For an object $s \in S$, the existence and minimal size of the preimage can be solved in polynomial time.*

Proof. The formula $\texttt{Anc}(s, 1)$ is, by definition, a DNF. □

We now investigate syntactical conditions under which existence and minimal size n-th ancestor, with $n > 1$, can be computed in polynomial time. The first condition we introduce is *linear dependency.*

Definition 19. *Let $\mathcal{A} = (S, A)$ be a r.s.. The* n-linear dependency *of an object s_2 from an object s_1, denoted $s_1 \overset{n}{\hookrightarrow} s_2$, is recursively defined as follows:*

1. $s_1 \overset{1}{\hookrightarrow} s_2$ iff $p(A, s) = 1$ and either $(\{s_1\}, \emptyset, \{s_2\}) \in A$ or $(\emptyset, \{s_1\}, \{s_2\}) \in A$;
2. $s_1 \overset{n}{\hookrightarrow} s_2$ with $n > 1$ iff there exists $s_3 \in S$ such that $s_1 \overset{1}{\hookrightarrow} s_3$ and $s_3 \overset{n-1}{\hookrightarrow} s_2$.

Intuitively, s_2 is n-linearly dependent from s_1 if it can be produced from s_1 in n steps in a unique way and by producing a single element at each step.

The second property states when an object is *n-linearly produced.*

Definition 20. *Let $\mathcal{A} = (S, A)$ be a reaction system. An object s_2 is* n-linearly produced *in $\mathcal{A}$ iff*

– *$|p(A, s_2)| = 0$, or*
– *there exists $s_1 \in S$ such that $s_1 \overset{1}{\hookrightarrow} s_2$ and s_1 is* $(n-1)$-linearly produced.

An object is linearly produced *when it is* n-linearly produced *for all n.*

Theorem 21. *Let $\mathcal{A} = (S, A)$ be a reaction system. If $\forall s \in S.|p(A, s)| \leq 1$, and for every rule $(R, I, P) \in A$ all the objects in I are n-linearly produced, then, for any $s \in S$, the existence and minimal size of the n-th ancestors of s can be solved in polynomial time.*

Proof. (Sketch) Since for every object $s \in S$ we have $|p(A, s)| \leq 1$, we know that no $\vee$ operator is introduced in the computation of the n-ancestors formula. Moreover, every object used as inhibitor is n-linearly produced, thus every negated atom, in the computation of n steps, is replaced by a single literal. As a consequence, a negation of a conjunctive formula (which could be transformed in a disjunctive formula) never occurs. Hence, the n-ancestors formula of s is simply a conjunction of literals of nesting level 1, a particular case of DNF. □

The next result extends the previous one to sets of objects.

Corollary 22. *Let $\mathcal{A} = (S, A)$ be a reaction system. If all the conditions of Theorem 21 are satisfied, then, for any set $\{s_1, s_2, ..., s_m\} \subseteq S$, the existence and minimal size of the n-th ancestors can be solved in polynomial time.*

Proof. (Sketch) The n-ancestors formula of a set of objects is, by definition, a conjunctive formula of nesting level 1. Hence, the whole n-ancestors formula of $\{s_1, s_2, ..., s_m\}$ is still a conjunctive formula of nesting level 1. □

Example 23. Let $\mathcal{A}_3 = (\{A, \dots, G\}, \{a_1, \dots, a_6\})$ be a reaction system with

$$a_1 = (\{A, B\}, \{\}, \{C\}) \qquad a_2 = (\{D, E\}, \{F\}, \{A\}) \qquad a_3 = (\{G\}, \{\}, \{F\})$$
$$a_4 = (\{B\}, \{\}, \{G\}) \qquad a_5 = (\{C, B\}, \{\}, \{D\}) \qquad a_6 = (\{E\}, \{\}, \{E\})$$

Since every object is produced by at most one rule (i.e. $\forall s \in S.|p(A, s)| \leq 1$) and the only inhibitor F is (linearly) produced by a reaction with a single reactant which, in turn, is (linearly) produced by a reaction with a single reactant, then the conditions of Theorem 21 and Corollary 22 are satisfied. Indeed, if we compute, for instance, $\texttt{Anc}(A, 2)$ we obtain $C \wedge D \wedge E \wedge \neg G$ that is in DNF.

Conditions expressed by Theorem 21 and Corollary 22 are not a characterization of all reaction systems having a n-ancestors formula in DNF. Weaker conditions can be found as, for instance, in the following proposition.

Proposition 24. *Let $\mathcal{A} = (S, A)$ be a reaction system. If*

- *there exists $\overline{s} \in S$ such that $|p(A, s)| \geq 2$, $\forall s \in S/\{\overline{s}\}$, $|p(A, s)| \leq 1$ and $\forall (R, I, P) \in A$ $\overline{s} \notin R$, and*
- *for each rule $(R, I, P) \in A$ all the objects in I are n-linearly produced,*

then, for any object $s \in S$, the existence and minimal size of the n-th ancestors can be solved in polynomial time.

Proof. (Skech) Since there is only $\overline{s} \in S$ that is produced by more than one rule, only $cause(\overline{s})$ can contain the $\vee$ operator. Moreover $\overline{s}$ cannot appear as reactant in any rule. Thus it cannot be introduced in the computation of the n-ancestors formula of any other object. Further, every object used as inhibitor is n-linearly produced, thus every negated atom, in the computation of n steps, is replaced by a literal. Then, the n-ancestors formula of s is a DNF. $\square$

7 Conclusions and Future Works

We proposed a constructive characterization of all n-ancestors of a set product, computed using an effective operator and exploited it to study the complexity of the existence and minimal size of the n-th ancestors and we found that they can be solved in polynomial time for reaction systems satisfying some syntactic conditions. As future work, we plan to apply our results to real systems. Moreover, we intend to investigate weaker syntactical conditions (corresponding to richer classes of reaction systems for which the considered problems are polynomial) and whether other computational problems could be solved in polynomial time by exploiting our n-ancestors formula, possibly transformed into Conjunctive Normal Form.

References

1. Barbuti, R., Bove, P., Gori, R., Levi, F., Milazzo, P.: Simulating gene regulatory networks using reaction systems (2018), to appear in Proc. of the 27th Int. Workshop on Concurrency, Specification and Programming, CS&P'18

2. Barbuti, R., Gori, R., Levi, F., Milazzo, P.: Specialized predictor for reaction systems with context properties. In: Proc. of the 24th Int. Workshop on Concurrency, Specification and Programming, CS&P. pp. 31–43 (2015)
3. Barbuti, R., Gori, R., Levi, F., Milazzo, P.: Investigating dynamic causalities in reaction systems. Theoretical Computer Science **623**, 114–145 (2016)
4. Barbuti, R., Gori, R., Levi, F., Milazzo, P.: Specialized predictor for reaction systems with context properties. Fundamenta Informaticae **147**(2-3), 173–191 (2016)
5. Barbuti, R., Gori, R., Levi, F., Milazzo, P.: Generalized contexts for reaction systems: definition and study of dynamic causalities. Acta Informatica **55**(3), 227–267 (2018)
6. Barbuti, R., Gori, R., Milazzo, P.: Multiset patterns and their application to dynamic causalities in membrane systems. In: 18th Int. Conference on Membrane Computing (CMC18). pp. 54–73. LNCS 10725, Springer (2017)
7. Barbuti, R., Gori, R., Milazzo, P.: Predictors for flat membrane systems. Theoretical Computer Science. **736**, 79–102 (2018)
8. Bodei, C., Gori, R., Levi, F.: An analysis for causal properties of membrane interactions. Electronic Notes in Theoretical Computer Science. **299**, 15–31 (2013)
9. Bodei, C., Gori, R., Levi, F.: Causal static analysis for brane calculi. Theoretical Computer Science **587**, 73–103 (2015)
10. Brayton, R.K., Sangiovanni-Vincentelli, A.L., McMullen, C.T., Hachtel, G.D.: Logic Minimization Algorithms for VLSI Synthesis. Kluwer Academic Publishers, Norwell, MA, USA (1984)
11. Brijder, R., Ehrenfeucht, A., Main, M.G., Rozenberg, G.: A tour of reaction systems. International Journal of Foundations of Computer Science **22**(7), 1499–1517 (2011)
12. Brijder, R., Ehrenfeucht, A., Rozenberg, G.: A note on causalities in reaction systems. ECEASST **30** (2010)
13. Busi, N.: Causality in membrane systems. In: Membrane Computing, 8th International Workshop, WMC 2007, Thessaloniki, Greece, June 25-28, 2007 Revised Selected and Invited Papers. pp. 160–171 (2007)
14. Dennunzio, A., Formenti, E., Manzoni, L., Porreca, A.E.: Ancestors, descendants, and gardens of eden in reaction systems. Theoretical Computer Science **608**, 16–26 (2015)
15. Dennunzio, A., Formenti, E., Manzoni, L., Porreca, A.E.: Preimage problems for reaction systems. In: Int. Conference on Language and Automata Theory and Applications (LATA 2015). pp. 537–548. Springer (2015)
16. Ehrenfeucht, A., Rozenberg, G.: Reaction systems. Fundamenta informaticae **75**(1-4), 263–280 (2007)
17. Formenti, E., Manzoni, L., Porreca, A.E.: Fixed points and attractors of reaction systems. In: Conference on Computability in Europe. pp. 194–203. Springer (2014)
18. Gori, R., Levi, F.: Abstract interpretation based verification of temporal properties for bioambients. Information Computation **208**(8), 869–921 (2010)
19. Hassoun, S., Sasao, T. (eds.): Logic Synthesis and Verification. Kluwer Academic Publishers (2002)
20. Salomaa, A.: Functional constructions between reaction systems and propositional logic. Int. J. Found. Comput. Sci. **24**(1), 147–160 (2013)
21. Salomaa, A.: Minimal and almost minimal reaction systems. Natural Computing **12**(3), 369–376 (2013)
22. Umans, C., Villa, T., Sangiovanni-Vincentelli, A.L.: Complexity of two-level logic minimization. IEEE Trans. on CAD of Integrated Circuits and Systems **25**(7), 1230–1246 (2006)