Natural Computing (2024) 23:189–204
https://doi.org/10.1007/s11047-024-09980-7

Pure reaction automata

Rocco Ascone[1] · **Giulia Bernardini**[1] · **Enrico Formenti**[2] · **Francesco Leiter**[1] · **Luca Manzoni**[1]

Accepted: 28 February 2024 / Published online: 29 May 2024

Abstract
This work introduces the new class of pure reaction automata, as well as a new update manner, called maximal reactive manner, that can also be applied to standard reaction automata. Pure reaction automata differ from the standard model in that they don't have permanence: the entities that are not consumed by the reactions happening at a certain state are not conserved in the result states. We prove that the set of languages accepted by the new class under the maximal reactive manner contains the set of languages accepted by standard reaction automata under the same manner or under the maximal parallel manner. We also prove that a strict subclass of pure reaction automata can compute any partial recursive function.

Keywords Reaction systems · Reaction automata · Formal languages · Computability

1 Introduction

Reaction systems (RS) are a growing and now established computational model, introduced by Ehrenfeucht and Rozenberg (2004, 2007), Brijder et al. (2011a), that takes inspiration from the chemical reactions occurring inside living cells: a set of *entities* or chemical species is transformed by one or more reactions and, like in real life, a reaction for which all *reactants* are present and all *inhibitors* are absent will generate all the expected *products*. Reaction Systems have some characteristics that distinguish them from other bio-inspired models. First of all, there is no permanence: an entity that is not used by any reaction will not remain in the system, but it will disappear. Second, there is no conflict: even if two reactions have the same reactants both of them will be enabled if their respective inhibitors are not present.

✉ Rocco Ascone
rocco.ascone@phd.units.it

✉ Francesco Leiter
francesco.leiter@studenti.units.it

Giulia Bernardini
giulia.bernardini@units.it

Enrico Formenti
enrico.formenti@univ-cotedazur.fr

Luca Manzoni
lmanzoni@units.it

[1] University of Trieste, Trieste, Italy

[2] Université Côte d'Azur, CNRS, I3S, Nice, France

Reaction systems have been successfully employed as a modelling tool in several areas, see for instance (Corolli et al. 2012; Azimi et al 2014; Barbuti et al. 2021). Concerning theoretical aspects, their properties have been studied from the point of view of the complexity of the dynamics (Formenti et al. 2014a, b, 2015; Ehrenfeucht et al. 2017; Barbuti et al. 2018a; Holzer and Rauch 2021; Teh and Lim 2022; Ascone et al. 2024; from the causality perspective (Brijder et al. 2010; Barbuti et al. 2016, 2018b); by introducing in a natural way additional restrictions and extensions (Brijder et al. 2011b; Salomaa 2017; Azimi 2017; Bottoni et al. 2019; Manzoni et al. 2020); by classifying and simulating them (Manzoni et al. 2014; Teh and Atanasiu 2017, 2020); and by relating them with other computational models (Kleijn et al. 2011; Păun et al. 2013; Dutta et al. 2019).

However, reaction systems are limited by the fact that their states are subsets of a finite set of entities. Hence, their dynamics can only contain a finite number of distinct configurations, making computational universality unattainable for them. A natural way to preserve the fundamentals of reaction systems while making them a universal computational model is to allow multiplicity for the entities. More than a decade ago, the first significant step in this direction was taken by Okubo et al. (2012b) with the introduction of *Reaction Automata* and with the successive streamline of studies on the topic (see Okubo et al. 2012a; Okubo 2014; Okubo and Yokomori 2015, 2018; Yokomori and Okubo 2021; Okubo et al. 2022). In reaction automata, a state is a *multiset* of entities (thus allowing an infinite

set of possible states) and reactions are modified to require multisets of reactants and products. A significant change introduced by this model is that now there is competition between reactions, requiring the model to specify a policy to decide which reactions are allowed to take place.

A well-studied model that resembles reaction automata are P automata (see, e.g., Csuhaj-Varjú and Vaszil 2002; Freund et al. 2003; Csuhaj-Varjú et al. 2006, 2009), the automata-like version of P systems, where multisets of *objects* are subdivided into multiple regions (or *membranes*). Those objects evolve according to a series of rules (usually *antiport rules*) that move the objects between the different regions and rewrite them. These rules are applied according to a given computational model, like sequential or maximally parallel (where conflicts are possible). One main difference between P automata and reaction automata is the "spatial" component of the former, where the subdivision of the space in regions is essential to perform the computation. The latter can be seen as a "degenerate" case of P automata, where only one region is present and only rules of a particular kind are admitted.

In this work, we continue the study of reaction automata by providing several new results and connections to other models. First of all, we define the *maximally reactive manner*, a new criterion on how to select which reactions, among the competing ones, will take place. We also introduce a new kind of reaction automata, called *pure* reaction automata. They differ from those introduced by Yokomori and Okubo (2021) in how the outcome of a reaction is defined. While classical reaction automata have permanence (*i.e.*, the entities that are not used are preserved), in pure reaction automata the entities that are not used by any reaction are lost. This makes pure reaction automata more similar, in this aspect, to reaction systems, where non-permanence is a defining characteristic.

The introduction of a new manner and a new model of reaction automata raises the question of its computation power compared to the already existing models. We prove that the set of languages accepted by pure reaction automata working in a maximal reactive manner contains the set of languages accepted by standard reaction automata working in the same manner as well as the set of languages accepted by reaction automata working in a maximal parallel manner (a manner already investigated by Yokomori and Okubo (2021)).

We also introduce a new research direction for reaction automata by looking at them as devices to compute partial functions—similarly to what has been already done for chemical reaction networks in Chen et al. (2014), Clamons et al. (2020). From this new point of view, we show that a restricted class of pure reaction automata can compute any recursive function from $\mathbb{N}^k$ to $\mathbb{N}$.

The paper is structured as follows. In Sect. 2 we give preliminary notions on reaction automata. In Sect. 3 we define and explore the computational power of pure reaction automata as language acceptors, while in Sect. 4 we change the perspective by exploring the computational power of a restricted class of pure reaction automata as devices for computing partial recursive functions. In Sect. 5 we provide a summary of our results and some directions for future research.

2 Preliminaries

Let S be a finite alphabet. We denote by S^* the set of words over S, that is, all finite sequences of elements of S, with $\varepsilon \in S^*$ denoting the empty word consisting of zero letters. A *multiset* over S is defined as a function $V : S \to \mathbb{N}$ such that $V(a) \in \mathbb{N}$ is the multiplicity of $a \in S$ in the multiset. By $S^\#$ we denote the set of all multisets over S. Given V and W two multisets over S, we define a partial order and four operations as follows:

- Inclusion: $V \leq W$ if $V(a) \leq W(a)$, for each $a \in S$;
- Sum: $(V + W)(a) := V(a) + W(a)$, for each $a \in S$;
- Intersection: $(V \cap W)(a) := \min\{V(a), W(a)\}$, for each $a \in S$;
- Difference: $(V - W)(a) := V(a) - W(a)$, for each $a \in S$ (only defined for $W \leq V$);
- Symmetric difference: $(V \triangle W)(a) := (V + W)(a) - (V \cap W)(a)$, for each $a \in S$.

Furthermore, we define a relation between the letters of S and a multiset: given $V \in S^\#$ and $a \in S$, $a \in V$ if and only if $V(a) \geq 1$, i.e., a letter is an element of V if and only if its multiplicity is at least one. This allows us to define the *set* underlying a multiset $V \in S^\#$ as the set of letters with nonzero multiplicity:

$$\mathrm{set}(V) := \{a \in S \mid a \in V\}.$$

The following properties follow immediately for any $V, W \in S^\#$:

$$\mathrm{set}(V + W) = \mathrm{set}(V) \cup \mathrm{set}(W), \quad \mathrm{set}(V \cap W) = \mathrm{set}(V) \cap \mathrm{set}(W).$$

Moreover, if $V \leq W$, then $\mathrm{set}(V) \subseteq \mathrm{set}(W)$, while the converse is not true: e.g., for $V = \{a, a, b\}$ and $W = \{a, b, c\}$, we have that $\mathrm{set}(V) \subseteq \mathrm{set}(W)$ but $V \not\leq W$.

A set $U \subseteq S$ can be seen as a multiset V_U such that $V_U(a) = 1$ if a is in U and $V_U(a) = 0$ otherwise. In particular, for each symbol $a \in S$, we will often denote the multiset $V_{\{a\}}$ simply by a. We will denote the empty multiset by $0 \in S^\#$.

The total number of elements in a multiset $V \in S$ is defined as $\|V\| := \sum_{a \in S} V(a)$.

Remark 1 The following map:

$$S^{\#} \longrightarrow \mathbb{N}^{|S|}$$
$$V \longmapsto (V(a_1), \dots, V(a_n))$$

is an isomorphism of monoids; we will denote such isomorphism by $S^{\#} \cong \mathbb{N}^{|S|}$.

2.1 Reaction automata

In this section, we recall the definition of reaction automata given by Yokomori and Okubo (2021) and we introduce a new policy for enabling reactions.

Definition 2 *(Reaction)* Given an alphabet of reactants S, a *reaction over* S is a triple $\mathbf{a} = (R_\mathbf{a}, I_\mathbf{a}, P_\mathbf{a})$, where $R_\mathbf{a} \in S^{\#}$ is the multiset of *reactants*, $I_\mathbf{a} \subseteq S$ is the set of *inhibitors* and $P_\mathbf{a} \in S^{\#}$ is the multiset of *products*. The set of all reactions over S is denoted by $\mathrm{rac}(S)$.

A crucial assumption in the reaction system model is that if a reactant is present at a certain *state* $T \in S^{\#}$, then its quantity is always enough for all the reactions that use it to take place, provided the respective inhibitors are not present. In other words, reactions do not conflict even if they share some resources. This assumption is no longer in place in the model of reaction automata, for which there is only a certain quantity of each of the reactants. It is therefore necessary to specify a criterion (called a *manner*) that decides which of several conflicting reactions take place. In this paper, we focus on two manners, provided in Definition 3. Before defining such manners, we need to introduce a few operations and relations among reactions.

Let $\mathbf{a} = (R_\mathbf{a}, I_\mathbf{a}, P_\mathbf{a})$, $\mathbf{b} = (R_\mathbf{b}, I_\mathbf{b}, P_\mathbf{b}) \in \mathrm{rac}(S)$; we define the sum of the two reactions $\mathbf{a} + \mathbf{b} := (R_\mathbf{a} + R_\mathbf{b}, I_\mathbf{a} \cup I_\mathbf{b}, P_\mathbf{a} + P_\mathbf{b})$. Furthermore, we define a partial order over all possible reactions over S: $\mathbf{a} \leq_\mathrm{r} \mathbf{b}$ if and only if $R_\mathbf{a} \leq R_\mathbf{b}$ and $I_\mathbf{a} \subseteq I_\mathbf{b}$. In other words, a reaction is greater than another when it is more restrictive, i.e., it requires more reactants and there are more elements capable of disabling it. With this relation, we get that $\mathbf{a} =_\mathrm{r} \mathbf{b}$ if and only if $R_\mathbf{a} = R_\mathbf{b}$ and $I_\mathbf{a} = I_\mathbf{b}$, but we do not impose any conditions on the products: in particular, it could hold $\mathbf{a} =_\mathrm{r} \mathbf{b}$ with $P_\mathbf{a} \neq P_\mathbf{b}$. We remark that this partial order is different from the one proposed by Ehrenfeucht and Rozenberg (2009) for reactions within reaction systems.

Given a finite set $\mathbf{A} \subseteq \mathrm{rac}(S)$, we denote by $\langle \mathbf{A} \rangle$ the abelian semigroup generated by the elements of $\mathbf{A}$:

$$\langle \mathbf{A} \rangle := \{\lambda_1 \mathbf{a}_1 + \cdots + \lambda_n \mathbf{a}_n \mid \mathbf{a}_i \in \mathbf{A}, \lambda_i \in \mathbb{N} \quad \forall i = 1, \dots, n\}.$$

Note that any element of $\langle \mathbf{A} \rangle$ is a reaction that can be interpreted as a multiset of reactions from $\mathbf{A}$, where the coefficients $\lambda_1, \dots, \lambda_n$ give the multiplicity of each reaction. We will thus denote a multiset of reactions that are enabled in a certain state as a single reaction from $\langle \mathbf{A} \rangle$.

Definition 3 *(Manners)* Let $\mathbf{a} = (R_\mathbf{a}, I_\mathbf{a}, P_\mathbf{a}) \in \mathrm{rac}(S)$ and $T \in S^{\#}$, we say that $\mathbf{a}$ is *enabled* in T if $R_\mathbf{a} \leq T$ and $I_\mathbf{a} \cap \mathrm{set}(T) = \emptyset$. Given $\mathbf{A}$ a finite set of reactions over S and $\mathbf{a} \in \langle \mathbf{A} \rangle$ enabled in T, then:

1. $\mathbf{a}$ is enabled in T in a *maximally parallel manner* (*mp*) if there exists no $\mathbf{c} \in \langle \mathbf{A} \rangle$ such that $\mathbf{a} + \mathbf{c}$ is enabled in T, i.e., $\mathbf{a}$ is maximal w.r.t. addition.
2. $\mathbf{a}$ is enabled in T in a *maximally reactive manner* (*mr*) if there exists no $\mathbf{b} \in \langle \mathbf{A} \rangle$ such that $\mathbf{a} <_\mathrm{r} \mathbf{b}$ and $\mathbf{b}$ is enabled in T, i.e., $\mathbf{a}$ is maximal w.r.t the partial order $\leq_\mathrm{r}$.

We denote by $\mathrm{En}^X_\mathbf{A}(T)$ the set of reactions from $\langle \mathbf{A} \rangle$ enabled in T in manner $X \in \{mp, mr\}$.

Remark 4 The notion of *mp* manner given in Definition 3 is equivalent to the one given by Yokomori and Okubo (2021). Indeed, given $\alpha, \beta \in \mathbf{A}^{\#}$, consider the corresponding elements $\mathbf{a}, \mathbf{b} \in \langle \mathbf{A} \rangle$, identified by $\alpha \mapsto \sum_{\mathbf{d} \in \mathbf{A}} \alpha(\mathbf{d})\mathbf{d} \in \langle \mathbf{A} \rangle$. If $\beta > \alpha$ is enabled by T, then $\mathbf{c} = \mathbf{b} - \mathbf{a}$ is such that $\mathbf{a} + \mathbf{c} = \mathbf{b}$ is enabled by T; the viceversa is obtained in a similar way.

Remark 5 At first sight, the definitions of manners *mp* and *mr* are hard to tell apart. However, the two criteria are distinct, and in particular, *mr* is stronger than *mp* in the sense that $\mathrm{En}^{mr}_\mathbf{A}(T) \subseteq \mathrm{En}^{mp}_\mathbf{A}(T)$ for any $\mathbf{A} \subseteq \mathrm{rac}(S)$ and any state T. In other words, if a reaction $\mathbf{a} \in \langle \mathbf{A} \rangle$ is enabled in a maximally reactive manner, it is also enabled in a maximally parallel manner. Indeed, the existence of $\mathbf{c} \in \langle \mathbf{A} \rangle$ s.t. $\mathbf{a} + \mathbf{c}$ is enabled in T would imply that a reaction greater than $\mathbf{a}$ is enabled in T, leading to a contradiction. The converse is not always true: see Example 7.

In reaction systems, the state resulting from a set of reactions is simply defined as the union of the products of all the reactions. In reaction automata, the reactants that are not consumed by the reactions that take place remain in the resulting states. We make this concept precise in Definition 6.

Definition 6 *(Result)* The *result* of a set of reactions $\mathbf{A}$ on a state T in a manner X is a set of states, denoted by $\mathrm{Res}^X_\mathbf{A}(T)$, defined as follows:

$$\mathrm{Res}^X_\mathbf{A}(T) = \{P_\mathbf{a} + (T - R_\mathbf{a}) \mid \mathbf{a} = (R_\mathbf{a}, I_\mathbf{a}, P_\mathbf{a}) \in \mathrm{En}^X_\mathbf{A}(T)\}.$$

When $\mathrm{En}^X_{\mathbf{A}}(T) = \emptyset$, we define $\mathrm{Res}^X_{\mathbf{A}}(T) = \{T\}$, that is, if no multiset of reactions from $\langle \mathbf{A} \rangle$ is enabled in T, then T remains unchanged.

Example 7 Let $S = \{w_1, w_2, \heartsuit\}$, $\mathbf{A} = \{\mathbf{a}_1 = (w_1, \emptyset, \heartsuit), \mathbf{a}_2 = (w_1 + w_2, \emptyset, w_2)\}$, and consider a state $T = w_1 + w_2$. Then the set of reactions enabled in T in a maximally parallel manner is $\mathrm{En}^{mp}_{\mathbf{A}}(T) = \{\mathbf{a}_1, \mathbf{a}_2\}$, as $\nexists \mathbf{c} \in \langle \mathbf{A} \rangle$ such that $\mathbf{a}_1 + \mathbf{c}$ is enabled in T; and the set of reactions enabled in T in a maximally reactive manner is $\mathrm{En}^{mr}_{\mathbf{A}}(T) = \{\mathbf{a}_2\}$, because $\mathbf{a}_2 >_r \mathbf{a}_1$ and thus $\mathbf{a}_1$ is not *mr*-enabled. The corresponding results in the two manners are thus $\mathrm{Res}^{mp}_{\mathbf{A}}(T) = \{\heartsuit + w_2, w_2\}$, $\mathrm{Res}^{mr}_{\mathbf{A}}(T) = \{w_2\}$.

We are now in a position to define reaction automata.

Definition 8 (Yokomori and Okubo (2021)) A *reaction automaton* (RA) $\mathcal{A}$ is a five-tuple $\mathcal{A} = (S, \Sigma, \mathbf{A}, D_0, f)$, where S is a finite set of reactants, called the *background set* of $\mathcal{A}$; $\Sigma \subseteq S$ is the *input alphabet* of $\mathcal{A}$; $\mathbf{A} \subseteq \mathrm{rac}(S)$ is a finite set of reactions over S; $D_0 \in S^{\#}$ is the *initial multiset*; and $f \in S$ is a special symbol which indicates the final state.

Definition 9 Consider a reaction automaton $\mathcal{A} = (S, \Sigma, \mathbf{A}, D_0, f)$, a word $w = w_1 \cdots w_n \in \Sigma^*$ and a manner $X \in \{mp, mr\}$. An *interactive process in $\mathcal{A}$ with input w in manner X* is an infinite sequence $\pi = D_0, \ldots, D_i, \ldots$ where

$$\begin{cases} D_{i+1} \in \mathrm{Res}^X_{\mathbf{A}}(w_{i+1} + D_i) & \text{for } 0 \le i \le n-1 \\ D_{i+1} \in \mathrm{Res}^X_{\mathbf{A}}(D_i) & \text{for } i \ge n. \end{cases}$$

By $\mathrm{IP}_X(\mathcal{A}, w)$ we denote the set of all interactive processes in $\mathcal{A}$ with input w in manner X. We say that a process π *strongly accepts* w if there exists $m \ge n = |w|$ such that $f \in D_m$ and $\mathrm{En}^X_{\mathbf{A}}(D_m) = \emptyset$ (see also Example 11). By $\mathrm{AIP}_X(\mathcal{A}, w)$ we denote the set of all processes $\pi \in \mathrm{IP}_X(\mathcal{A}, w)$ such that π strongly accepts w. The *language strongly accepted* by $\mathcal{A}$ is defined as

$$L_X(\mathcal{A}) = \{w \in \Sigma^* \mid \mathrm{AIP}_X(\mathcal{A}, w) \neq \emptyset\}.$$

The set of languages strongly accepted by reaction automata working in manner X is denoted by $\mathcal{RA}_X$: $L \in \mathcal{RA}_X$ if and only if there exists a reaction automaton working in manner X that strongly accepts L.

We say that a process π *weakly accepts* w if there exists $m \ge n = |w|$ such that $f \in D_m$ and $D_k = D_m$ for all $k \ge m$ (see also Example 12). By $\mathrm{AIP}^{\mathrm{w}}_X(\mathcal{A}, w)$ we denote the set of all processes $\pi \in \mathrm{IP}_X(\mathcal{A}, w)$ such that π weakly accepts w. The *language weakly accepted* by $\mathcal{A}$ is defined as

$$L^{\mathrm{w}}_X(\mathcal{A}) = \{w \in \Sigma^* \mid \mathrm{AIP}^{\mathrm{w}}_X(\mathcal{A}, w) \neq \emptyset\}.$$

The set of languages weakly accepted by reaction automata working in manner X is denoted by $\mathcal{RA}^{\mathrm{w}}_X$.

We will sometimes represent an interactive process π with the following "arrow notation", which extends the notation proposed by Yokomori and Okubo (2021):

$$\pi : D_0 \xrightarrow[w_1]{\mathbf{a}_1} D_1 \xrightarrow[w_2]{\mathbf{a}_2} D_2 \xrightarrow[w_3]{\mathbf{a}_3} \cdots D_{n-1} \xrightarrow[w_n]{\mathbf{a}_n} D_n \xrightarrow{\mathbf{a}_{n+1}} D_{n+1} \xrightarrow{\mathbf{a}_{n+2}} \cdots$$

where $D_{i-1} \xrightarrow[w_i]{\mathbf{a}_i} D_i$ means w_i is the input letter at state D_{i-1}, $\mathbf{a}_i \in \langle \mathbf{A} \rangle$ is the reaction enabled in $D_{i-1} + w_i$ which takes place, and $D_i \in \mathrm{Res}^X_{\mathbf{A}}(w_i + D_{i-1})$.

Example 10 If $w = \varepsilon$ the empty word, then $|w| = n = 0$, thus an interactive process accepting strongly ε is of the form $D_0 \xrightarrow{\mathbf{a}_1} D_1 \xrightarrow{\mathbf{a}_2} D_2 \xrightarrow{\mathbf{a}_3} \cdots D_m \rightarrow D_m \rightarrow D_m \cdots$, where D_m does not enable any reaction.

Note that in Definition 9 we call *strong* the acceptance condition proposed by Yokomori and Okubo (2021); the weak notion of acceptance will be needed for the definition of pure reaction automata (see Sect. 3).

In Proposition 13 we show that the weak acceptance criterion extends the strong criterion in the following sense: given an automaton that strongly accepts a certain language, it is always possible to construct another automaton that weakly accepts the same language. Examples 11 and 12 provide intuition on this result before we formally prove it in Proposition 13: Example 11 provides an automaton accepting words with the strong criterion, while Example 12 constructs another automaton that accepts exactly the same language as in Example 11 but using the weak criterion.

Example 11 Given an input alphabet $\Sigma = \{a, b\}$, a background set $S = \{a, b, s_0, s_1, f\}$ and a set of reactions $\mathbf{A} = \{\mathbf{a}_1 = (s_0 + a, \emptyset, s_1), \mathbf{a}_2 = (s_1 + b, \emptyset, s_0), \mathbf{a}_3 = (s_0, \emptyset, f), \mathbf{a}_4 = (b, \emptyset, b)\}$, let $\mathcal{A} = (S, \Sigma, \mathbf{A}, s_0, f)$ be a reaction automaton working in *mr* manner that accepts words with the strong criterion. We show that the language strongly accepted by $\mathcal{A}$ is $L_{mr}(\mathcal{A}) = \{(ab)^n : n \in \mathbb{N}\}$. Consider the input word $w = abab \in \Sigma^*$: we obtain the following process.

$$s_0 \xrightarrow[a]{\mathbf{a}_1} s_1 \xrightarrow[b]{\mathbf{a}_2} s_0 \xrightarrow[a]{\mathbf{a}_1} s_1 \xrightarrow[b]{\mathbf{a}_2} s_0 \xrightarrow{\mathbf{a}_3} f.$$

The computation stops since $\mathrm{En}^{mr}_{\mathbf{A}}(f) = \emptyset$. A trivial extension of this argument proves that $\{(ab)^n : n \in \mathbb{N}\} \subseteq L_{mr}(\mathcal{A})$.

To show the other inclusion, observe that a word $w \notin \{(ab)^n : n \in \mathbb{N}\}$ if and only if at least one of the following cases happens: (i) *aa* occurs in w; (ii) *bb* occurs in w; (iii) w starts with b; (iv) w ends with a. In case (i), when the second consecutive a is fed to the process, it ends up in the state $s_1 + a$: the computation stops because $\mathrm{En}^{mr}_{\mathbf{A}}(s_1 + a) = \emptyset$, but

since $f \notin s_1 + a$, w is not strongly accepted. In case (ii), after reading the second consecutive b, the process reaches the state $s_0 + b$: then $s_0 + b \xrightarrow{\mathbf{a}_3+\mathbf{a}_4} b + f$ and $\mathrm{En}^{mr}_{\mathbf{A}}(b+f) = \{\mathbf{a}_4\}$, thus the process keeps looping in this state and w is not strongly accepted because $\mathrm{En}^{mr}_{\mathbf{A}}(b+f) \neq \emptyset$. The same happens in case (iii), when b is added to the initial state s_0. Finally, in case (iv), after reading the last a, the process ends up in state s_1: the computation stops because $\mathrm{En}^{mr}_{\mathbf{A}}(s_1) = \emptyset$, but since $f \notin s_1$, w is not strongly accepted. We can thus conclude that $L_{mr}(\mathcal{A}) = \{(ab)^n : n \in \mathbb{N}\}$.

Note that using the weak notion of acceptance instead of the strong one for the reaction automaton $\mathcal{A}$ of Example 11 would imply $\{(ab)^n : n \in \mathbb{N}\} \subsetneq L^{\mathrm{w}}_{mr}(\mathcal{A})$, since for example the string $w = b \notin \{(ab)^n : n \in \mathbb{N}\}$ is weakly accepted by the following process:

$$s_0 \xrightarrow[b]{\mathbf{a}_3+\mathbf{a}_4} b + f \xrightarrow{\mathbf{a}_4} b + f \xrightarrow{\mathbf{a}_4} \cdots .$$

Example 12 shows that it is anyway possible to construct another RA $\mathcal{B}$ such that $L^{\mathrm{w}}_{mr}(\mathcal{B}) = \{(ab)^n : n \in \mathbb{N}\}$.

Example 12 Given an input alphabet $\Sigma = \{a, b\}$ and a background set $S = \{a, b, s_0, s_1, \spadesuit, \clubsuit, f\}$ let $\mathcal{B} = (S, \Sigma, \mathbf{B}, s_0 + \clubsuit, f)$ be a reaction automaton working in *mr* manner and using the weak acceptance criterion, with $\mathbf{B}$ consisting of the following reactions:

$$\begin{aligned}
\mathbf{a}^{\spadesuit}_1 &= (s_0 + a, \{\spadesuit\}, s_1 + \spadesuit) & \mathbf{a}^{\clubsuit}_1 &= (s_0 + a, \{\clubsuit\}, s_1 + \clubsuit)\\
\mathbf{a}^{\spadesuit}_2 &= (s_1 + b, \{\spadesuit\}, s_0 + \spadesuit) & \mathbf{a}^{\clubsuit}_2 &= (s_1 + b, \{\clubsuit\}, s_0 + \clubsuit)\\
\mathbf{a}^{\spadesuit}_3 &= (s_0, \{\spadesuit\}, f + \spadesuit) & \mathbf{a}^{\clubsuit}_3 &= (s_0, \{\clubsuit\}, f + \clubsuit)\\
\mathbf{a}^{\spadesuit}_4 &= (b, \{\spadesuit\}, b + \spadesuit) & \mathbf{a}^{\clubsuit}_4 &= (b, \{\clubsuit\}, b + \clubsuit)\\
\mathbf{r}^{\spadesuit} &= (\spadesuit, \emptyset, 0) & \mathbf{r}^{\clubsuit} &= (\clubsuit, \emptyset, 0).
\end{aligned}$$

We show that $\mathcal{B}$ weakly accepts the same language that is strongly accepted by $\mathcal{A}$ in Example 11. Consider $w = ab \in \{(ab)^n : n \in \mathbb{N}\}$. We obtain the following process:

$$s_0 + \clubsuit \xrightarrow[a]{\mathbf{a}^{\spadesuit}_1+\mathbf{r}^{\clubsuit}} s_1 + \spadesuit \xrightarrow[b]{\mathbf{a}^{\clubsuit}_2+\mathbf{r}^{\spadesuit}} s_0 + \clubsuit \xrightarrow{\mathbf{a}^{\spadesuit}_3+\mathbf{r}^{\clubsuit}} f + \spadesuit \xrightarrow{\mathbf{r}^{\spadesuit}} f.$$

The computation stops since $\mathrm{En}^{mr}_{\mathbf{B}}(f) = \emptyset$, thus $\mathrm{Res}^{mr}_{\mathbf{B}}(f) = \{f\}$ and w is weakly accepted. Consider now the string $w = aa \notin \{(ab)^n : n \in \mathbb{N}\}$. We obtain the following process:

$$s_0 + \clubsuit \xrightarrow[a]{\mathbf{a}^{\spadesuit}_1+\mathbf{r}^{\clubsuit}} s_1 + \spadesuit \xrightarrow[a]{\mathbf{r}^{\spadesuit}} s_1 + a$$

and the computation stops since $\mathrm{En}^{mr}_{\mathbf{B}}(s_1 + a) = \emptyset$; aa is not weakly accepted because $f \notin s_1 + a$. Similarly, it is easy to see that any string in case (i) of Example 11 is not weakly accepted. Consider the string $w = b \notin \{(ab)^n : n \in \mathbb{N}\}$. We obtain the following process:

$$s_0 + \clubsuit \xrightarrow[b]{\mathbf{a}^{\spadesuit}_3+\mathbf{a}^{\spadesuit}_4+\mathbf{r}^{\clubsuit}} b + f + \spadesuit \xrightarrow{\mathbf{a}^{\spadesuit}_4+\mathbf{r}^{\spadesuit}} b + f + \clubsuit \xrightarrow{\mathbf{a}^{\clubsuit}_4+\mathbf{r}^{\spadesuit}} b + f + \spadesuit \to \cdots$$

The states $b + f + \clubsuit$ and $b + f + \spadesuit$ keep alternating, so the string is not weakly accepted. Similarly, one can prove that any string starting with b (case (iii) of Example 11) is not weakly accepted. Working out the other cases in a similar fashion, it is easy to show that $L^{\mathrm{w}}_{mr}(\mathcal{B}) = \{(ab)^n : n \in \mathbb{N}\}$.

In Example 12 we were able to simulate the processes of $\mathcal{A}$ using the following rule: for any step m, if $\mathrm{En}^{X}_{\mathbf{A}}(D_m) = \emptyset$ then the computation stops also in $\mathcal{B}$; if, instead, $\mathrm{En}^{X}_{\mathbf{A}}(D_m) \neq \emptyset$, then the computation in $\mathcal{B}$ keeps alternating between states containing $\spadesuit$ and states containing $\clubsuit$. The proof of the following Proposition 13 relies on a generalization of this argument.

Proposition 13 *Given any reaction automaton $\mathcal{A} = (S, \Sigma, \mathbf{A}, D_0, f)$ working in a manner $X \in \{mp, mr\}$, there exists a reaction automaton $\mathcal{B}$ working in manner X such that $L_X(\mathcal{A}) = L^{\mathrm{w}}_X(\mathcal{B})$.*

Proof Let $\mathcal{A} = (S, \Sigma, \mathbf{A}, D_0, f)$ and let $\spadesuit, \clubsuit \notin S$ be two symbols that are not in the alphabet of $\mathcal{A}$. We define $\mathcal{B} = (S \cup \{\spadesuit, \clubsuit\}, \Sigma, \mathbf{B}, D_0 + \clubsuit, f)$ the reaction automaton such that $\mathbf{B} := \mathbf{A}^{\spadesuit} \cup \mathbf{A}^{\clubsuit} \cup \{\mathbf{r}^{\spadesuit}, \mathbf{r}^{\clubsuit}\}$, where

$$\begin{aligned}
\mathbf{A}^{\spadesuit} &= \{\mathbf{a}^{\spadesuit} = (R, I \cup \{\spadesuit\}, P + \spadesuit) \mid \mathbf{a} = (R, I, P) \in \mathbf{A}\}\\
\mathbf{A}^{\clubsuit} &= \{\mathbf{a}^{\clubsuit} = (R, I \cup \{\clubsuit\}, P + \clubsuit) \mid \mathbf{a} = (R, I, P) \in \mathbf{A}\}\\
\mathbf{r}^{\spadesuit} &= (\spadesuit, \emptyset, 0)\\
\mathbf{r}^{\clubsuit} &= (\clubsuit, \emptyset, 0).
\end{aligned}$$

Given $\mathbf{a} = \sum_{j=1}^{n} \lambda_j \mathbf{a}_j \in \langle \mathbf{A} \rangle$ we define $c_{\mathbf{a}} = \sum_{j=1}^{n} \lambda_j$, $\mathbf{a}^{\spadesuit} = \sum_{j=1}^{n} \lambda_j \mathbf{a}^{\spadesuit}_j \in \langle \mathbf{A}^{\spadesuit} \rangle$ and similarly $\mathbf{a}^{\clubsuit} = \sum_{j=1}^{n} \lambda_j \mathbf{a}^{\clubsuit}_j \in \langle \mathbf{A}^{\clubsuit} \rangle$.

We want to prove that $\mathcal{B}$ weakly accepts the same language that is strongly accepted by $\mathcal{A}$. Given a process $\pi \in \mathrm{IP}_X(\mathcal{A}, w)$

$$\pi : D_0 \xrightarrow[w_1]{\mathbf{a}_1} D_1 \xrightarrow[w_2]{\mathbf{a}_2} D_2 \xrightarrow[w_3]{\mathbf{a}_3} \ldots D_{n-1} \xrightarrow[w_n]{\mathbf{a}_n} D_n \xrightarrow{\mathbf{a}_{n+1}} D_{n+1} \cdots$$

if n is even, there is a one-to-one correspondence with the process $\bar{\pi} \in \mathrm{IP}_X(\mathcal{B}, w)$

$$\begin{aligned}
\bar{\pi} : D_0 + \clubsuit \xrightarrow[w_1]{\mathbf{a}^{\spadesuit}_1+\mathbf{r}^{\clubsuit}} D_1 + c_{\mathbf{a}_1} \spadesuit \xrightarrow[w_2]{\mathbf{a}^{\clubsuit}_2+c_{\mathbf{a}_1}\mathbf{r}^{\spadesuit}} D_2 + c_{\mathbf{a}_2} \clubsuit \xrightarrow[w_3]{\mathbf{a}^{\spadesuit}_3+c_{\mathbf{a}_2}\mathbf{r}^{\clubsuit}} \cdots\\
\cdots D_{n-1} + c_{\mathbf{a}_{n-1}} \spadesuit \xrightarrow[w_n]{\mathbf{a}^{\clubsuit}_n+c_{\mathbf{a}_{n-1}}\mathbf{r}^{\spadesuit}} D_n + c_{\mathbf{a}_n} \clubsuit \xrightarrow{\mathbf{a}^{\spadesuit}_{n+1}+c_{\mathbf{a}_n}\mathbf{r}^{\clubsuit}} \cdots .
\end{aligned}$$

If n is odd, a process corresponding to π can be obtained similarly. Since $\spadesuit, \clubsuit$ are not present in any of the reactant

sets of $\mathbf{A}^{\spadesuit}$ and $\mathbf{A}^{\clubsuit}$, the reaction $\mathbf{r}^{\spadesuit}$ (respectively, $\mathbf{r}^{\clubsuit}$) is enabled as many times as the multiplicity of ♠ (respectively, ♣) in any given state; this is independent of the manner $X \in \{mp, mr\}$ we are working with. In particular, any process in $\mathrm{IP}_X(\mathcal{B}, w)$ ends in a state D_m s.t. $D_k = D_m$ for all $k > m$ if and only if at some step no reactions are enabled (as otherwise, ♠ and ♣ would keep alternating). Therefore $\mathrm{AIP}_X(\mathcal{A}, w)$ is in a bijection with $\mathrm{AIP}^{\mathrm{w}}_X(\mathcal{B}, w)$. We conclude that $L_X(\mathcal{A}) = L^{\mathrm{w}}_X(\mathcal{B})$. □

Corollary 14 *$\mathcal{RA}_{mr} \subseteq \mathcal{RA}^{\mathrm{w}}_{mr}$ and $\mathcal{RA}_{mp} \subseteq \mathcal{RA}^{\mathrm{w}}_{mp}$.*

Proof Follows directly from Proposition 13. □

Corollary 15 *Every recursively enumerable language is weakly accepted by a reaction automaton working in a maximally parallel manner.*

Proof Follows from Proposition 13 and (Okubo et al. 2012b, Corollary 1). □

We thus proved that the weak acceptance criterion extends the strong criterion. This will be useful in the next section to demonstrate the computational power of pure reaction automata.

3 Pure reaction automata

In this section, we introduce a different kind of reaction automata, which differs from the model introduced by Yokomori and Okubo (2021) by how the result of a reaction is defined: instead of transferring the reactants that are not consumed by the reactions in the result states, we define the next states to consist only of the products of the reactions, similar to what is done in reaction systems. We make this concept formal in Definition 16.

Definition 16 *(Pure result)* The *pure result* of a finite set of reactions $\mathbf{A}$ on a state T in a manner X is

$$\widehat{\mathrm{Res}}^{X}_{\mathbf{A}}(T) = \{P_{\mathbf{a}} \mid \mathbf{a} = (R_{\mathbf{a}}, I_{\mathbf{a}}, P_{\mathbf{a}}) \in \mathrm{En}^{X}_{\mathbf{A}}(T)\},$$

and we define $\widehat{\mathrm{Res}}^{X}_{\mathbf{A}}(T) = \{0\}$ when $\mathrm{En}^{X}_{\mathbf{A}}(T) = \emptyset$.

Example 17 Let $S = \{w_1, w_2, \heartsuit\}$, $\mathbf{A} = \{\mathbf{a}_1 = (w_1, \emptyset, \heartsuit), \mathbf{a}_2 = (w_1 + w_2, \emptyset, w_2)\}$ and consider a state $T = w_1 + w_2$ as in Example 7. Recall that $\mathrm{Res}^{mp}_{\mathbf{A}}(T) = \{\heartsuit + w_2, w_2\}$ (since $\mathrm{En}^{mp}_{\mathbf{A}}(T) = \{\mathbf{a}_1, \mathbf{a}_2\}$); in contrast, the pure result of T in *mp* manner is $\widehat{\mathrm{Res}}^{mp}_{\mathbf{A}}(T) = \{\heartsuit, w_2\}$. In particular, reaction $\mathbf{a}_1$ does not consume the reactant w_2 that is present in T: the pure result of $\mathbf{a}_1$ only consists of ♡ and w_2 is lost, while in Example 7 w_2 was preserved in the result.

We name this new kind of reaction automata *Pure Reaction Automata* (PRA). We define interactive processes in pure reaction automata in much the same way as standard reaction automata, as specified by Definition 18.

Definition 18 Let $\mathcal{M} = (S, \Sigma, \mathbf{A}, D_0, f)$ be a PRA, $w = w_1 \cdots w_n \in \Sigma^*$ and $X \in \{mp, mr\}$. An interactive process in $\mathcal{M}$ with input w in manner X is an infinite sequence $\pi = D_0, \ldots, D_i, \ldots$ where

$$\begin{cases} D_{i+1} \in \widehat{\mathrm{Res}}^{X}_{\mathbf{A}}(w_{i+1} + D_i) & \text{for } 0 \le i \le n-1 \\ D_{i+1} \in \widehat{\mathrm{Res}}^{X}_{\mathbf{A}}(D_i) & \text{for } i \ge n. \end{cases}$$

Exactly as for reaction automata, we say that π weakly accepts w if there exists $m \ge n = |w|$ such that $f \in D_m$ and $D_k = D_m$ for all $k \ge m$. We also define $\mathrm{IP}_X(\mathcal{M}, w)$, $\mathrm{AIP}^{\mathrm{w}}_X(\mathcal{M}, w)$, and $L^{\mathrm{w}}_X(\mathcal{M})$ in the same way as for reaction automata. The set of languages weakly accepted by PRA working in manner X is denoted by $\mathcal{PRA}^{\mathrm{w}}_X$.

Example 19 Given an input alphabet $\Sigma = \{a, b\}$, a background set $S = \{a, b, s_0, s_1, f\}$ and a set of reactions $\mathbf{A} = \{\mathbf{a}_1 = (s_0, \{b\}, s_1), \mathbf{a}_2 = (s_1 + b, \emptyset, s_0), \mathbf{a}_3 = (s_0, \{a, b\}, f), \mathbf{a}_4 = (f, \emptyset, f)\}$, let $\mathcal{M} = (S, \Sigma, \mathbf{A}, s_0, f)$ be a pure reaction automaton working in *mr* manner. We show that the language (weakly) accepted by $\mathcal{M}$ is $L^{\mathrm{w}}_{mr}(\mathcal{M}) = \{(ab)^n : n \in \mathbb{N}\}$. Consider the input word $w = abab \in \Sigma^*$: we obtain the following process.

$$s_0 \xrightarrow[a]{\mathbf{a}_1} s_1 \xrightarrow[b]{\mathbf{a}_2} s_0 \xrightarrow[a]{\mathbf{a}_1} s_1 \xrightarrow[b]{\mathbf{a}_2} s_0 \xrightarrow{\mathbf{a}_3} f \xrightarrow{\mathbf{a}_4} f \xrightarrow{\mathbf{a}_4} \cdots.$$

Therefore the string *abab* is accepted. Note that the pure result of reaction $\mathbf{a}_1$ applied at states $s_0 + a$ consists only of s_1, even if a is not consumed by the reaction. We also remark that the step $s_0 \xrightarrow{\mathbf{a}_3} f$ generates f instead of s_1 because $\mathbf{a}_3 >_{\mathrm{r}} \mathbf{a}_1$ and thus $\mathrm{En}^{mr}_{\mathbf{B}}(s_0) = \{\mathbf{a}_3\}$. A trivial extension of this argument proves that $\{(ab)^n : n \in \mathbb{N}\} \subseteq L^{\mathrm{w}}_{mr}(\mathcal{M})$. Consider now the string $w = aa \notin \{(ab)^n : n \in \mathbb{N}\}$, as in Example 12. We obtain the following process:

$$s_0 \xrightarrow[a]{\mathbf{a}_1} s_1 \xrightarrow[a]{} 0 \to 0 \to \cdots$$

and the computation stops since $\mathrm{En}^{mr}_{\mathbf{B}}(s_1 + a) = \mathrm{En}^{mr}_{\mathbf{B}}(0) = \emptyset$, thus *aa* is not accepted. Working out the other cases listed in Example 11 similarly, it is easy to see that $L^{\mathrm{w}}_{mr}(\mathcal{M}) = \{(ab)^n : n \in \mathbb{N}\}$.

Note that a non-pure automaton with the same sets of reactions, initial state and final element would not accept, e.g., the string $ab \in \{(ab)^n : n \in \mathbb{N}\}$ since the only interactive process with input ab is:

$$s_0 \xrightarrow[a]{\mathbf{a}_1} s_1 + a \xrightarrow[b]{\mathbf{a}_2} s_0 + a \xrightarrow{\mathbf{a}_1} s_1 + a \to s_1 + a \cdots$$

and the computation stops without accepting ab since, as previously remarked, $\mathrm{En}^{mr}_{\mathbf{B}}(s_1 + a) = \emptyset$.

Theorem 20 *Given any reaction automaton $\mathcal{A} = (S, \Sigma, \mathbf{A}, D_0, f)$ working in a maximally reactive manner, there exists a pure reaction automaton $\mathcal{M}$ working in a maximally reactive manner such that $L^{\mathrm{w}}_{mr}(\mathcal{A}) = L^{\mathrm{w}}_{mr}(\mathcal{M})$.*

Proof We define a pure reaction automaton $\mathcal{M} = (S \cup S', \Sigma, \mathbf{A}', D_0, f)$ operating in a maximally reactive manner such that:

- $S' = \{x' \mid x \in S\}$ is a set in a bijection with the elements of S (a "copy" of S). From now on, given a multiset X over S, X' will be naturally defined as the multiset consisting of the copies of the elements of X.
- Σ is the same input alphabet as $\mathcal{A}$;
- $\mathbf{A}' = \mathbf{A}_p \cup \mathbf{A}_c$, where $\mathbf{A}_p = \{(x, \emptyset, x + x') \mid x \in S\}$ and $\mathbf{A}_c = \{(R + R', I', P + P') \mid (R, I, P) \in \mathbf{A}\}$.
- D_0 is the same initial state as $\mathcal{A}$;
- $f \in S$ is the same final element as $\mathcal{A}$.

Claim 21 *For any state W of $\mathcal{A}$, it holds $W \in \mathrm{Res}^{mr}_{\mathbf{A}}(V)$ if and only if $W + W' \in \widehat{\mathrm{Res}}^{mr}_{\mathbf{A}'}(V + V')$.*

Proof Given a reaction $\mathbf{a} \in \mathbf{A}$, we denote the corresponding reaction in $\mathbf{A}_c$ by $\mathbf{a}'$. We prove the two implications.

$\Rightarrow$) If $W \in \mathrm{Res}^{mr}_{\mathbf{A}}(V)$ then $\exists \mathbf{a} = (R_{\mathbf{a}}, I_{\mathbf{a}}, P_{\mathbf{a}}) \in \mathrm{En}^{mr}_{\mathbf{A}}(V)$ such that $P_{\mathbf{a}} + V - R_{\mathbf{a}} = W$. Consider $\mathbf{a}'' := \mathbf{a}' + (V - R_{\mathbf{a}}, \emptyset, V - R_{\mathbf{a}} + V' - R'_{\mathbf{a}}) = (V + R'_{\mathbf{a}}, I'_{\mathbf{a}}, W + W') \in \langle \mathbf{A}' \rangle$. Clearly, $V + V'$ enables $\mathbf{a}''$; we want to show that it is mr enabled. Suppose for a contradiction that there exists $\mathbf{b}'' \in \langle \mathbf{A}' \rangle$ such that $\mathbf{b}'' >_{\mathrm{r}} \mathbf{a}''$ and $\mathbf{b}''$ is enabled by $V + V'$, then $\mathbf{b}''$ is of the form $\mathbf{b}'' = \mathbf{b}' + (R, \emptyset, R + R')$ for some $\mathbf{b} = (R_{\mathbf{b}}, I_{\mathbf{b}}, P_{\mathbf{b}}) \in \langle \mathbf{A} \rangle$ and some $R \in S^{\#}$. Looking at the reactants, we have that $V + V' \geq R_{\mathbf{b}} + R'_{\mathbf{b}} + R \geq V + R'_{\mathbf{a}}$, which implies $R_{\mathbf{b}} + R = V \Rightarrow R_{\mathbf{b}} \leq V$ and $R'_{\mathbf{b}} \geq R'_{\mathbf{a}}$. Furthermore, looking at the inhibitors, we obtain that $I'_{\mathbf{b}} \supseteq I'_{\mathbf{a}}$, thus $\mathbf{b} >_{\mathrm{r}} \mathbf{a}$ and $\mathbf{b}$ is enabled by V. Since $\mathbf{a}$ is mr enabled we get a contradiction, thus $\mathbf{a}'' \in \mathrm{En}^{mr}_{\mathbf{A}'}(V + V')$, hence $W + W' \in \widehat{\mathrm{Res}}^{mr}_{\mathbf{A}'}(V + V')$.

$\Leftarrow$) If $W + W' \in \widehat{\mathrm{Res}}^{mr}_{\mathbf{A}'}(V + V')$ then $\exists \mathbf{a}'' = \mathbf{a}' + (R, \emptyset, R + R') \in \mathrm{En}^{mr}_{\mathbf{A}'}(V + V')$, for some $\mathbf{a} \in \langle \mathbf{A} \rangle$ and $R \in S^{\#}$, such that $P_{\mathbf{a}} + R = W$. We notice that $V - R_{\mathbf{a}} = R$, as otherwise for any $x \in V - R_{\mathbf{a}} - R$ the reaction $\mathbf{a}'' + (x, \emptyset, x + x')$ would be enabled by $V + V'$ and would be strictly greater than $\mathbf{a}''$, in contradiction to the fact that $\mathbf{a}''$ is mr enabled. Clearly, $\mathbf{a} \in \langle \mathbf{A} \rangle$ is enabled by V. We now prove it is also mr enabled. Suppose for a contradiction that there exists $\mathbf{b} \in \langle \mathbf{A} \rangle$ such that $\mathbf{b} >_{\mathrm{r}} \mathbf{a}$ and $\mathbf{b}$ is enabled by V, then $\mathbf{b}'' := \mathbf{b}' + (V - R_{\mathbf{b}}, \emptyset, V - R_{\mathbf{b}} + V' - R'_{\mathbf{b}})$ would be enabled by $V + V'$ and $\mathbf{b}'' >_{\mathrm{r}} \mathbf{a}''$, a contradiction. Finally we can conclude that $\mathbf{a} \in \mathrm{En}^{mr}_{\mathbf{A}}(V)$, hence $P_{\mathbf{a}} + V - R_{\mathbf{a}} = W \in \mathrm{Res}^{mr}_{\mathbf{A}}(V)$. □

Let us now make some considerations about the workings of $\mathcal{M}$. Consider any state V, let R be s.t. $(R, I, P) \in \mathbf{A}$ for some I and P and $(R \cup R') \subseteq V + V'$; then, for any $x \in R$, the corresponding reaction $(x, \emptyset, x + x') \in \mathbf{A}_p$ cannot be mr enabled, as it is smaller than $(R + R', I', P + P') \in \mathbf{A}_c$. Thus a reaction $(x, \emptyset, x + x') \in \mathbf{A}_p$ will only be applied in two cases: either (i), the multiplicity of x in V is greater than that of x', thus there is at least one "spare" x that will not be used by any reaction from $\mathbf{A}_c$; or (ii), there is no $(R \cup R') \subseteq V + V'$ such that $(R, I, P) \in \mathbf{A}$. Note that case (i) can only happen in the initial state D_0, in which there are no elements from S', or when x is added to a state because it is a letter of an input word: this is because every state other than D_0 is the product of some reaction in $\langle \mathbf{A}' \rangle$, thus, by definition, it is of the form $P + P'$ for some $P \in S^{\#}$. Case (ii) ensures that if x is not consumed by any reaction from $\langle \mathbf{A}_c \rangle$, then it is conserved in the next state by a reaction from $\mathbf{A}_p$, simulating what would happen in the reaction automaton $\mathcal{A}$ when a reactant is not consumed by any reaction from $\mathbf{A}$.

Consider now an input word $w = w_1 \cdots w_n \in \Sigma^*$. We notice that when a letter w_i of w is added to the $(i-1)$-th state of a process in $\mathcal{M}$, it immediately reacts through the corresponding reaction $(w_i, \emptyset, w_i + w'_i) \in \mathbf{A}_p$ because it occurs case (i) above. This helps us to find the desired correspondence between processes of $\mathcal{A}$ and processes of $\mathcal{M}$ accepting w. Let $\pi = D_0, D_1, \ldots, D_i, \ldots \in \mathrm{IP}_{mr}(\mathcal{A}, w)$: the corresponding process $E_0, E_1 \ldots, E_n \in \mathrm{IP}_{mr}(\mathcal{M}, w)$ is obtained as follows. $E_1 := D_0 + D'_0 + w_1 + w'_1 \in \widehat{\mathrm{Res}}^{mr}_{\mathbf{A}'}(D_0 + w_1)$ since the only reactions that are enabled are those in $\langle \mathbf{A_p} \rangle$. In the next steps we have, by Claim 21 and the observations here above:

$$\begin{cases} E_{k+1} = D_k + D'_k + w_{k+1} + w'_{k+1} \in \widehat{\mathrm{Res}}^{mr}_{\mathbf{A}'}(E_k + w_{k+1}) & \forall k = 1, \ldots, n-1 \\ E_{k+1} = D_k + D'_k \in \widehat{\mathrm{Res}}^{mr}_{\mathbf{A}'}(E_k) & \forall k \geq n. \end{cases}$$

Hence the processes of $\mathcal{M}$ mimic those of $\mathcal{A}$ with one step of delay, i.e., $\pi' := D_0, E_1, \ldots, E_i, \ldots \in \mathrm{IP}_{mr}(\mathcal{M}, w)$. If $\pi \in \mathrm{AIP}^{\mathrm{w}}_{mr}(\mathcal{A}, w)$ then there exists $m \geq |w|$ such that

$D_k = D_m$ for all $k \geq m$ and $f \in D_m$; this holds true if and only if $E_k = E_{m+1}$ for all $k \geq m+1$ and $f \in E_{m+1}$. Therefore $\pi \in \mathrm{AIP}^{\mathrm{w}}_{mr}(\mathcal{A}, w)$ if and only if $\pi' \in \mathrm{AIP}^{\mathrm{w}}_{mr}(\mathcal{M}, w)$. Since the accepting condition is the same, and a state maintaining f in $\mathcal{A}$ must correspond to a state maintaining f in $\mathcal{M}$, we have that $\mathrm{AIP}^{\mathrm{w}}_{mr}(\mathcal{A}, w)$ is in a natural bijection with $\mathrm{AIP}^{\mathrm{w}}_{mr}(\mathcal{M}, w)$, and hence we obtain the thesis. □

We next prove in Theorem 24 that for any reaction automaton working in *mp* manner, there exists a pure reaction automaton working in *mr* manner that weakly accepts the same languages. Example 22 shows that this result cannot be achieved using the same construction as in the proof of Theorem 20, thus we will provide a more involved reduction, an example of which is laid out in Example 23.

Example 22 Given $\Sigma = \{w_1, w_2\}$, $S = \{w_1, w_2, \heartsuit\}$ and $\mathbf{A} = \{\mathbf{a}_1 = (w_1, \varnothing, \heartsuit), \mathbf{a}_2 = (w_1 + w_2, \varnothing, w_2)\}$, let $\mathcal{A} = (S, \Sigma, \mathbf{A}, w_1, \heartsuit)$ be a reaction automaton working in *mp* manner. Consider the pure reaction automaton $\mathcal{M} = (S \cup S', \Sigma, \mathbf{A}', w_1, \heartsuit)$ as in the proof of Theorem 20, thus with $\mathbf{A}' = \{\mathbf{a}'_1, \mathbf{a}'_2\} \cup \{\mathbf{a}_x = (x, \varnothing, x + x') \mid x \in S\}$. Then w_2 belongs to the language weakly accepted by $\mathcal{A}$ in a maximally parallel manner, i.e., $w_2 \in L^{\mathrm{w}}_{mp}(\mathcal{A})$ (see Definition 9), but it does not belong to the language accepted by $\mathcal{A}$ in a maximally reactive manner, i.e., $w_2 \notin L^{\mathrm{w}}_{mr}(\mathcal{M})$.

Indeed, as seen in Example 7, the set of reactions from $\mathcal{A}$ enabled in state $T = w_1 + w_2$ in a maximally parallel manner is $\mathrm{En}^{mp}_{\mathbf{A}}(w_1 + w_2) = \{\mathbf{a}_1, \mathbf{a}_2\}$, thus the result is $\mathrm{Res}^{mp}_{\mathbf{A}}(w_1 + w_2) = \{\heartsuit + w_2, w_2\}$ and w_2 is accepted by the interactive process $D_0 + w_2 = w_1 + w_2 \xrightarrow{\mathbf{a}_1} \heartsuit + w_2 \to \heartsuit + w_2 \to \cdots \in \mathrm{AIP}^{\mathrm{w}}(w_2, \mathcal{A})$.

In contrast, since the set of reactions enabled in $T = w_1 + w_2$ in a maximally recative manner is $\mathrm{En}^{mr}_{\mathbf{A}}(w_1 + w_2) = \{\mathbf{a}_2\}$ (see Example 7), then the result is $\mathrm{Res}^{mr}_{\mathbf{A}}(w_1 + w_2) = \{w_2\}$, thus by Claim 21 we obtain that the set of reactions from $\mathcal{M}$ enabled in $T + T' = w_1 + w'_1 + w_2 + w'_2$ in a maximally reactive manner is $\mathrm{En}^{mr}_{\mathbf{A}'}(w_1 + w'_1 + w_2 + w'_2) = \{\mathbf{a}'_2\}$ and the pure result is $\widehat{\mathrm{Res}}^{mr}_{\mathbf{A}}(w_1 + w'_1 + w_2 + w'_2) = \{w_2 + w'_2\}$. Therefore there is only one process with input w_2, namely: $D_0 + w_2 = w_1 + w_2 \xrightarrow{\mathbf{a}_{w_1} + \mathbf{a}_{w_2}} w_1 + w'_1 + w_2 + w'_2 \xrightarrow{\mathbf{a}'_2} w_2 + w'_2 \xrightarrow{\mathbf{a}_{w_2}} w_2 + w'_2 \xrightarrow{\mathbf{a}_{w_2}} \cdots$; since $\heartsuit \notin w_1 + w'_2$, this is not an accepting process, thus w_2 is not in the language weakly accepted by $\mathcal{M}$ in a maximally reactive manner, i.e., $w_2 \notin L^{\mathrm{w}}_{mr}(\mathcal{M})$.

Example 23 Consider $\mathcal{A} = (S, \Sigma, \mathbf{A}, w_1, \heartsuit)$ the RA from Example 22. We make two copies of each element of the background set, the i-th copy being in a natural correspondence with reaction $\mathbf{a}_i$: $S^i = \{w^i_1, w^i_2, \heartsuit^i\}$ for $i \in \{1, 2\}$. Let $\mathbf{A}'$ consist of the following reactions over $S \cup S^1 \cup S^2$:

$$\mathbf{a}'_1 := (w_1 + w^1_1, \varnothing, \heartsuit + \heartsuit^1 + \heartsuit^2)$$
$$\mathbf{a}'_2 := (w_1 + w_2 + w^2_1 + w^2_2, \varnothing, w_2 + w^1_2 + w^2_2)$$
$$\mathbf{a}_x := (x, \varnothing, x + x^1 + x^2) \quad \text{for all } x \in S.$$

We define the pure reaction automaton $\mathcal{M}' = (S \cup S^1 \cup S^2, \Sigma, \mathbf{A}', w_1, \heartsuit)$. We have that $\mathrm{En}^{mr}_{\mathbf{A}'}(w_1 + w^1_1 + w^2_1 + w_2 + w^1_2 + w^2_2) = \{\mathbf{a}'_2, \mathbf{a}'_1 + \mathbf{a}_{w_2}\}$ since $\mathbf{a}'_2$ and $\mathbf{a}'_1 + \mathbf{a}_{w_2}$ are not comparable and are both maximal. Therefore we obtain the following accepting process with input w_2:

$$D_0 + w_2 = w_1 + w_2 \xrightarrow{\mathbf{a}_{w_1} + \mathbf{a}_{w_2}} w_1 + w^1_1 + w^2_1 + w_2 + w^1_2 + w^2_2 \xrightarrow{\mathbf{a}'_1 + \mathbf{a}_{w_2}} \heartsuit + \heartsuit^1 + \heartsuit^2 + w_2 + w^1_2 + w^2_2 \xrightarrow{\mathbf{a}_\heartsuit + \mathbf{a}_{w_2}} \heartsuit + \heartsuit^1 + \heartsuit^2 + w_2 + w^1_2 + w^2_2 \xrightarrow{\mathbf{a}_\heartsuit + \mathbf{a}_{w_2}} \cdots.$$

We have obtained that $w_2 \in L^{\mathrm{w}}_{mr}(\mathcal{M}')$. In the proof of the following theorem, we will extend this construction.

Theorem 24 *Given a reaction automaton $\mathcal{A} = (S, \Sigma, \mathbf{A}, D_0, f)$ working in a maximally parallel manner, there exists a pure reaction automaton $\mathcal{M}$ working in a maximally reactive manner such that $L^{\mathrm{w}}_{mp}(\mathcal{A}) = L^{\mathrm{w}}_{mr}(\mathcal{M})$.*

Proof Let $\mathbf{A} = \{\mathbf{a}_1, \ldots, \mathbf{a}_k\}$. We make $k = |\mathbf{A}|$ copies of each element of the background set, the i-th copy being in a natural correspondence with reaction $\mathbf{a}_i$: $S^i = \{x^i \mid x \in S\}$ for each $i = 1, 2, \ldots, |\mathbf{A}|$. We define a pure reaction automaton $\mathcal{M} = (S \cup S', \Sigma, \mathbf{A}', D_0, f)$ working in a maximally reactive manner such that:

- $S' := S^1 \cup \cdots \cup S^k$;
- Σ is the same input alphabet as $\mathcal{A}$;
- $\mathbf{A}' := \mathbf{A}_p \cup \mathbf{A}_c$, where

$$\mathbf{A}_p := \{(x, \varnothing, x + x^1 + x^2 + \cdots + x^k) \mid x \in S\}$$
$$\mathbf{A}_c := \{\mathbf{a}'_i := (R_{\mathbf{a}_i} + R^i_{\mathbf{a}_i}, I^i_{\mathbf{a}_i}, P_{\mathbf{a}_i} + P^1_{\mathbf{a}_i} + \cdots + P^k_{\mathbf{a}_i}) \mid \mathbf{a}_i = (R_{\mathbf{a}_i}, I_{\mathbf{a}_i}, P_{\mathbf{a}_i}) \in \mathbf{A}\}$$

- D_0 is the same initial state as $\mathcal{A}$;
- f is the same final element as $\mathcal{A}$.

Furthermore, for any $T \in S^{\#}$ we define $T' := \sum_{j=1}^{k} T^j \in (S')^{\#}$, where T^j consists of the j-th copy of every element of T. In particular, x' denotes $x^1 + x^2 + \cdots + x^k$ for any $x \in S$.

Claim 25 $W \in \mathrm{Res}^{mp}_{\mathbf{A}}(V)$ *iff* $W + W' \in \widehat{\mathrm{Res}}^{mr}_{\mathbf{A}'}(V + V')$.

Proof We prove the two implications.

$\Rightarrow$) If $W \in \mathrm{Res}^{mp}_{\mathbf{A}}(V)$ then $\exists \mathbf{a} = \sum_{j=1}^{k} \lambda_j \mathbf{a}_j \in \mathrm{En}^{mp}_{\mathbf{A}}(V)$ such that $P_{\mathbf{a}} + V - R_{\mathbf{a}} = W$. Consider $\mathbf{a}'' := \sum_{j=1}^{k} \lambda_j \mathbf{a}'_j + (V - R_{\mathbf{a}}, \varnothing, V - R_{\mathbf{a}} + V' - R'_{\mathbf{a}}) \in \langle \mathbf{A}' \rangle$, then $V + V'$ enables $\mathbf{a}''$; we want to show that it is *mr* enabled. Let $\mathbf{b}'' \in \langle \mathbf{A}' \rangle$ such that $\mathbf{b}'' >_{\mathrm{r}} \mathbf{a}''$ and suppose for a contradiction that $\mathbf{b}''$ is enabled by $V + V'$. We have that $\mathbf{b}'' = \sum_{j=1}^{k} \mu_j \mathbf{a}'_j + (R, \varnothing, R + R')$ for some $\mathbf{b} = \sum_{j=1}^{k} \mu_j \mathbf{a}_j \in \langle \mathbf{A} \rangle$ and some $R \in S^{\#}$, thus looking at the reactants we obtain $R = V - R_{\mathbf{b}}$. Furthermore, since $\mathbf{b}'' >_{\mathrm{r}} \mathbf{a}''$, there exists i such that $\lambda_i < \mu_i$, thus $\mathbf{a} + (\mu_i - \lambda_i)\mathbf{a}_i \leq_{\mathrm{r}} \mathbf{b}$ is enabled by V, a contradiction since $\mathbf{a}$ is enabled in maximally parallel manner. Finally, we can conclude that $\mathbf{a}'' \in \mathrm{En}^{mr}_{\mathbf{A}'}(V + V')$, hence $W + W' \in \widehat{\mathrm{Res}}^{mr}_{\mathbf{A}'}(V + V')$.

$\Leftarrow$) If $W + W' \in \widehat{\mathrm{Res}}^{mr}_{\mathbf{A}'}(V + V')$ then $\exists \mathbf{a}'' = \mathbf{a}' + (R, \varnothing, R + R') \in \mathrm{En}^{mr}_{\mathbf{A}'}(V + V')$, for some $\mathbf{a} \in \langle \mathbf{A} \rangle$ and $R \in S^{\#}$, such that $P_{\mathbf{a}} + R = W$. Note that $V - R_{\mathbf{a}} = R$, as otherwise, for any $x \in V - R_{\mathbf{a}} - R$, the reaction $\mathbf{a}'' + (x, \varnothing, x + x')$ would be enabled by $V + V'$ and strictly greater than $\mathbf{a}''$, in contradiction to the fact that $\mathbf{a}$ is maximally enabled. We immediately remark that $\mathbf{a} \in \langle \mathbf{A} \rangle$ is enabled by V. We want to prove that is *mp* enabled. Let $\mathbf{c} \in \langle \mathbf{A} \rangle$ such that $\mathbf{a} + \mathbf{c}$ is enabled by V, then $(\mathbf{a} + \mathbf{c})'' >_{\mathrm{r}} \mathbf{a}''$ is enabled by $V + V'$, a contradiction. Finally we can conclude that $\mathbf{a} \in \mathrm{En}^{mp}_{\mathbf{A}}(V)$, hence $P_{\mathbf{a}} + V - R_{\mathbf{a}} = P_{\mathbf{a}} + R = W \in \mathrm{Res}^{mp}_{\mathbf{A}}(V)$. □

We conclude as in Theorem 20. □

Corollary 26 $\mathcal{RA}_{mr} \subseteq \mathcal{RA}^{\mathrm{w}}_{mr} \subseteq \mathcal{PRA}^{\mathrm{w}}_{mr}$ *and* $\mathcal{RA}_{mp} \subseteq \mathcal{RA}^{\mathrm{w}}_{mp} \subseteq \mathcal{PRA}^{\mathrm{w}}_{mr}$.

Proof Follows directly from Theorems 20 and 24 and Corollary 14. □

Corollary 27 *Every recursively enumerable language is weakly accepted by a pure reaction automaton working in a maximally reactive manner.*

Proof Follows directly from Corollary 15 and Theorem 24. □

4 Computing functions with pure reaction automata

In this section, we introduce a new angle to investigate the computing power of pure reaction automata. Inspired by existing work on chemical reaction networks (see Chen et al. 2014; Clamons et al. 2020) we will show that PRA can be seen as machines to compute partial functions from $\mathbb{N}^k$ into $\mathbb{N}$. For more details on partial recursive functions and related computability issues, we refer the reader to Rogers (1987).

Definition 28 Given a partial function $\phi : \mathbb{N}^n \rightharpoonup \mathbb{N}$, consider an alphabet $\Sigma := \{a_1, \ldots, a_n\}$ so that $\Sigma^{\#} \cong \mathbb{N}^n$. A PRA $\mathcal{M} = (S, \Sigma, \mathbf{A}, D_0, f)$ *computes* ϕ in a manner $X \in \{mp, mr\}$ if:

- when $\phi(x_1, \ldots, x_n)$ is defined, it holds $\phi(x_1, \ldots, x_n) = y$ if and only if when the linear combination $x_1 a_1 + \cdots + x_n a_n$ is added to the initial state D_0 for any process $D_0 + x_1 a_1 + \cdots + x_n a_n, \ldots, D_k, \ldots$ there exists $k \in \mathbb{N}$ such that $D_k = D_m \forall m \geq k, yf \leq D_k, (y+1)f \not\leq D_k$;
- $\phi(x_1, \ldots, x_n)$ is undefined if and only if all processes starting from $D_0 + a_1 + \cdots + x_n a_n$ satisfy $\forall k \in \mathbb{N} : D_k \neq D_{k+1}$, i.e., none of the processes stabilizes.

In other words, a PRA computes a partial function ϕ if and only if, interpreting $(x_1, \ldots, x_n) \in \mathbb{N}^n$ as the coefficients of a linear combination of the elements in Σ and adding this combination to the initial state, one of the following happens: if ϕ is not defined for $(x_1, \ldots, x_n)$, no process that starts from there stabilizes; otherwise, any process stabilizes on a state that contains f with multiplicity y, where $\phi(x_1, \ldots, x_n) = y$.

We will focus on a special class of PRA, specified in Definition 29, such that there is exactly one process that can start from each possible initial state. Throughout this section, we assume that all PRA operate in a maximally reactive manner.

Definition 29 Given $\mathcal{M} = (S, \Sigma, \mathbf{A}, D_0, S_f)$ a PRA working in a manner $X \in \{mp, mr\}$, we say that $\mathcal{M}$ is *deterministic* (DPRA) if and only if for any reachable state V, the pure result $\widehat{\mathrm{Res}}^{X}_{\mathbf{A}}(V)$ consists of one element only.

The next lemmas provide a few examples of partial functions that can be computed with DPRA working in a maximally reactive manner.

Lemma 30 *There exists a* DPRA *computing the sum function* $\phi : \mathbb{N}^n \to \mathbb{N}, \phi(x_1, x_2) := x_1 + x_2$.

Proof Let $\Sigma := \{a_1, a_2\}$, $S := \{a_1, a_2, f\}$ and $\mathbf{A} := \{\mathbf{b}_1 = (a_1, \varnothing, f), \mathbf{b}_2 = (a_2, \varnothing, f), \mathbf{b}_3 = (f, \varnothing, f)\}$. $\mathcal{M} := (S, \Sigma, \mathbf{A}, 0, f)$ is clearly deterministic and it computes ϕ as, for any $(x_1, x_2) \in \mathbb{N}^2$, adding $x_1a_1 + x_2a_2$ to the initial state 0 gives rise to the process

$$x_1a_1 + x_2a_2 \xrightarrow{x_1\mathbf{b}_1 + x_2\mathbf{b}_2} (x_1 + x_2)f \xrightarrow{(x_1+x_2)\mathbf{b}_3} (x_1 + x_2)f \longrightarrow \cdots.$$

□

Lemma 31 *There exists a* DPRA *computing the difference function* $\phi : \mathbb{N}^n \rightharpoonup \mathbb{N}, \phi(x_1, x_2) := x_1 - x_2$.

Proof Clearly, $\phi(x_1, x_2)$ is defined if and only if $x_1 \geq x_2$. Let $\Sigma := \{a_1, a_2\}$, $S := \{a_1, a_2, a_1', a_2', \diamond, f\}$ and let $\mathbf{A}$ consist of the following six reactions:

$\mathbf{b}_1 = (a_1, \varnothing, a_1')$
$\mathbf{b}_2 = (a_2, \varnothing, a_2' + \diamond)$
$\mathbf{b}_3 = (a_2', \varnothing, a_2)$
$\mathbf{b}_4 = (a_1', \varnothing, f)$
$\mathbf{b}_5 = (a_1' + a_2' + \diamond, \varnothing, 0)$
$\mathbf{b}_6 = (f, \varnothing, f)$.

We claim that $\mathcal{M} := (S, \Sigma, \mathbf{A}, 0, f)$ computes ϕ. Indeed, let $(x_1, x_2) \in \mathbb{N}^2$ such that $x_1 \geq x_2$, then adding $x_1a_1 + x_2a_2$ to the initial state 0 gives rise to the process

$$\begin{aligned} x_1a_1 + x_2a_2 &\xrightarrow{x_1\mathbf{b}_1 + x_2\mathbf{b}_2} x_1a_1' + x_2a_2' + x_2\diamond \\ &\xrightarrow{(x_1-x_2)\mathbf{b}_4 + x_2\mathbf{b}_5} (x_1 - x_2)f \xrightarrow{(x_1-x_2)\mathbf{b}_6} (x_1 - x_2)f \longrightarrow \cdots. \end{aligned}$$

On the other hand, if $(x_1, x_2) \in \mathbb{N}^2$ is such that $x_1 < x_2$, then the process becomes

$$\begin{aligned} x_1a_1 + x_2a_2 &\xrightarrow{x_1\mathbf{b}_1 + x_2\mathbf{b}_2} x_1a_1' + x_2a_2' + x_2\diamond \\ &\xrightarrow{(x_2-x_1)\mathbf{b}_3 + x_1\mathbf{b}_5} (x_2 - x_1)a_2 \\ &\xrightarrow{(x_2-x_1)\mathbf{b}_2} (x_2 - x_1)a_2' + (x_2 - x_1)\diamond \\ &\xrightarrow{(x_2-x_1)\mathbf{b}_3} (x_2 - x_1)a_2 \longrightarrow \cdots, \end{aligned}$$

thus the process gets stuck in an unstable configuration. Note that the proof relies heavily on maximal reactivity: $\mathbf{b}_5 >_r \mathbf{b}_3 + \mathbf{b}_4$ since $a_1' + a_2' + \diamond > a_1' + a_2'$. The role of the element $\diamond$ is precisely to make $\mathbf{b}_5$ maximal in this case. □

We will prove that the class of functions that DPRA operating in *mr* can compute is universal, in the sense of Church-Turing's Thesis. We start with the following basic lemma.

Lemma 32 *The constant function equal to* 0, *the successor function and the projection functions can be computed by a* DPRA.

Proof *Constant function.* The constant function $C_0 : \mathbb{N}^k \to \mathbb{N}, C_0(x_1, \ldots, x_k) := 0 \ \forall (x_1, \ldots, x_k) \in \mathbb{N}^k$ is computed by the DPRA $\mathcal{M} = (\Sigma \cup \{f\}, \Sigma, \varnothing, 0, f)$. It is straightforward to verify that $x_1a_1 + \cdots + x_ka_k$ gives rise to the process $x_1a_1 + \cdots + x_ka_k \to 0 \to 0 \to \cdots$.

Successor function. The successor function $S : \mathbb{N} \to \mathbb{N}$, $S(x) := x + 1 \ \forall x \in \mathbb{N}$, is computed by the DPRA $\mathcal{M} = (S, \Sigma, \mathbf{A}, \rhd, f)$, where $S = \Sigma \cup \{\rhd, f\}$, $\Sigma = \{a\}$ and $\mathbf{A} = \{(\rhd, \varnothing, f), (a, \varnothing, f), (f, \varnothing, f)\}$. It is straightforward to verify that $\rhd + xa$ gives rise to the process $\rhd + xa \to (x + 1)f \to (x + 1)f \to \cdots$.

Projection functions. Given $k, n \in \mathbb{N}$ with $1 \leq n \leq k$, the projection function $P_n^k : \mathbb{N}^k \to \mathbb{N}$, $P_n^k(x_1, \ldots, x_k) := x_n \ \forall (x_1, \ldots, x_k) \in \mathbb{N}^k$, is computed by the DPRA $\mathcal{M} = (S, \Sigma, \mathbf{A}, 0, f)$ where $S = \Sigma \cup \{f\}$, $\Sigma = \{a_1, \ldots, a_k\}$ and $\mathbf{A} = \{(a_n, \varnothing, f), (f, \varnothing, f)\}$. It is straightforward to verify that $x_1a_1 + \cdots + x_ka_k$ gives rise to the process $x_1a_1 + \cdots + x_ka_k \to x_nf \to x_nf \to \cdots$. □

Note that so far we have not made use of inhibitors in the reactions; however, they will become crucial in proving the following results.

Remark 33 If a process within a DPRA reaches the state $D_k = 0$, then $D_{k+1} = 0$. Furthermore, by determinism, if $D_k = D_{k+1}$ then $D_{k+m} = D_k$ for all $m \in \mathbb{N}$, thus to decide whether the process stabilizes it suffices to find the smallest $k \in \mathbb{N}$ such that $D_k = D_{k+1}$.

Definition 34 (*Normalized* DPRA) A DPRA $\mathcal{M} = (S, \Sigma, \mathbf{A}, D_0, f)$ operating in a maximally reactive manner which computes a partial function ϕ is called *normalized* if the following two conditions hold:

1. there exists $h \in S$ such that, whenever $\phi(x_1, \ldots, x_k) = y$, the process $D_0 + x_1a_1 + \cdots + x_na_n, \ldots, D_k, \ldots$ stabilizes in a state of the type $P + h + yf$, for some $P \in S^{\#}$;
2. h and f are not present in the multiset of reactants of any reaction in $\mathbf{A}$.

A normalized DPRA computing ϕ will be denoted by $\mathcal{M}(\phi, h, f)$.

The following lemma proves that to determine the computational power of DPRA as function acceptors, it suffices to study normalized DPRA.

Lemma 35 *Given any* DPRA *computing a partial function* ϕ, *there exists a normalized* DPRA *computing* ϕ.

Proof Given $\mathcal{M} = (S, \Sigma, \mathbf{A}, D_0, f)$ computing ϕ, $\Sigma = \{a_1, \ldots, a_n\}$, $S = \Sigma \cup \{s_1, \ldots, s_m, f\}$, we define $\widehat{\mathcal{M}} = (\widehat{S}, \Sigma, \mathbf{A}', \rhd + \diamond + \spadesuit, g)$ as follows:

- $\widehat{S} := S^0 \cup S^1 \cup S^2 \cup S^3 \cup \Sigma \cup \{\rhd, g, g^1, h, \square, \diamond, \spadesuit, \#^1, \#^3\}$, where $S^i := \{s^i : s \in S\}$ for all $i \in \{0, 1, 2, 3\}$.
- Σ is the same input alphabet as $\mathcal{M}$;
- $\mathbf{A}' := \mathbf{A}_{in} \cup \mathbf{A}^0 \cup \mathbf{A}_{\spadesuit} \cup \mathbf{A}_f$, where

$$\mathbf{A}_{in} := \begin{cases} (a, \varnothing, a^0 + a^1 + \#^1) & \text{for each } a \in \Sigma, \\ (\rhd, \varnothing, D_0^0 + D_0^1 + \|D_0\|\#^1 + \spadesuit) & \\ (\diamond, \{\square\}, \diamond) & \end{cases}$$

$$\mathbf{A}^0 := \begin{cases} (R^0, I^0 \cup S^1 \cup \{\square\}, P^0 + P^1 + \|P^1\|\#^1) & \text{for each } (R, I, P) \in \mathbf{A} \\ (R^0, I^0 \cup S^2 \cup \{\square\}, P^0 + P^2) & \text{for each } (R, I, P) \in \mathbf{A} \end{cases}$$

$$\mathbf{A}_{\spadesuit} := \begin{cases} (\#^1 + a^1, \{\square\}, \#^3 + a^3) & \text{for each } a \in S \\ (a^2, \{\square\}, \spadesuit) & \text{for each } a \in S \\ (a^3, \{\square\}, \spadesuit) & \text{for each } a \in S \\ (a^2 + a^3 + \#^3, \{\square\}, 0) & \text{for each } a \in S \end{cases}$$

$$\mathbf{A}_f := \begin{cases} (\#^1 + f^1, \{\spadesuit, \square\}, g) & \\ (\diamond, S^2 \cup \{\spadesuit, \square, \rhd\}, \square) & \\ (g, \varnothing, g + g^1) & \\ (\square, \varnothing, \square + h) & \end{cases}$$

- $\rhd + \diamond + \spadesuit$ is the new initial state.
- g is the new final element.

Before analyzing the processes within $\mathcal{M}$, let us give an intuition about the role of each group of reactions and each symbol in $\widehat{S}$. The reactions of type $\mathbf{A}^0$ mimic the reactions of $\mathcal{M}$: at every step of the process, they produce two copies of the state that would have been reached by the process in $\mathcal{M}$. One of the copies always consists of elements from S^0, the second copy consists of elements from S^1 in the first step, from S^2 in the second step, and they keep alternating because the two groups of reactions in $\mathbf{A}^0$ are inhibited by elements from S^1 and S^2, respectively. We call the steps in which elements from S^1 are produced *of type 1*; the steps in which elements from S^2 are produced are *of type 2*. In steps of type 1, the reactions from the first group also produce the symbol $\#^1$ with a multiplicity equal to the number of elements from S^1 that are produced; $\#^1$ is thus a counter for the elements from S^1 in the next state.

The first two groups of reactions from $\mathbf{A}_{in}$ only happen at the beginning of the computation (indeed $\rhd$ symbolizes starting the computation), their role being to produce two copies, consisting of elements from S^0 and S^1, respectively, of all the input elements and a copy of the initial state of $\mathcal{M}$, so as to allow the reactions from $\mathbf{A}^0$ to take place; they also produce the symbol $\spadesuit$, which must be produced as long as the computation has not reached a stationary state. The last reaction produces the symbol $\diamond$, whose role is explained later, at every step, until there is a signal for the computation to stop.

The reactions in the first group of $\mathbf{A}_{\spadesuit}$ have the role of transforming every element from S^1 in the current state into its copy from S^3 in the next state, producing also a counter $\#^3$ that will have the same multiplicity as $\#^1$. The rest of the reactions are used to compare the elements from S^2 with those from S^3 in the current state: the symbol $\spadesuit$ is produced as long as they differ, and it is no longer produced as soon as they are equal. This comparing mechanism heavily relies on maximal reactivity, because it always holds $(a^2 + a^3 + \#^3, \{\square\}, 0) >_{\mathrm{r}} (a^2, \{\square\}, \spadesuit) + (a^3, \{\square\}, \spadesuit)$ for any $a \in S$: this implies that if the elements from S^2 are a copy of those from S^3, none of the reactions from the second and third group of $\mathbf{A}_{\spadesuit}$ will be maximally enabled (note that $\#^3$ will be present in the exact same quantity as the multiplicity of the elements from S^3) thus $\spadesuit$ will not be produced. The role of the $\#^3$ counters is really important since they give a way to prioritize the last group of reactions in $\mathbf{A}_{\spadesuit}$.

The reactions from $\mathbf{A}_f$ are needed to produce the last states at the end of the computation. The first two take place as soon as $\spadesuit$ is no longer present in a state, thus when the computation needs to stop. If f was generated by the last reaction when the process stops in the original automaton $\mathcal{M}$, then f^1 is present as well (together with the right multiplicity of the counter $\#^1$), thus the first reaction from $\mathbf{A}_f$ takes place: indeed, it is always greater than the first reaction from $\mathbf{A}_{\spadesuit}$ (because its set of inhibitors also contains $\spadesuit$), thus it is enabled in a maximal reactive manner. The second reaction from $\mathbf{A}_f$ is maximally enabled as well (it is strictly greater than the last reaction from $\mathbf{A}_{in}$) thus it produces $\square$, while $\diamond$ is no longer produced. The role of $\square$ is to disable all the reactions from $\mathbf{A}^0$; the role of $\diamond$ is to enable the reaction that produces $\square$ whenever needed. Finally, the last two

reactions from $\mathbf{A}_f$ are enabled, and they produce h and y copies of $g + g^1$.

Let us now analyze the processes within $\widehat{\mathcal{M}}$. Given $\pi = D_0 + x_1 a_1 + \cdots + x_n a_n, \ldots, D_k, \ldots$ a process in $\mathcal{M}$, we denote $X_0 := D_0 + x_1 a_1 + \cdots + x_n a_n$. We obtain the following process for $\widehat{\mathcal{M}}$:

$$\begin{aligned}
&\rhd + \spadesuit + \diamond + x_1 a_1 + \cdots + x_n a_n \\
&\quad \longrightarrow X_0^0 + X_0^1 + \|X_0\|^{\#^1} + \spadesuit + \diamond \\
&\quad \longrightarrow D_1^0 + D_1^2 + X_0^3 + \|X_0\|^{\#^3} + \diamond \\
&\quad \longrightarrow D_2^0 + D_2^1 + \|D_2\|^{\#^1} + \|X_0 \triangle D_1\| \spadesuit + \diamond \\
&\quad \longrightarrow \cdots
\end{aligned}$$

Suppose π stabilizes, then by Remark 33 there exists a k such that $D_k = D_{k+1}$ but $D_{k-1} \neq D_k$. We first consider the case in which $D_k \neq 0$.

If k is odd, we have:

$$\begin{aligned}
&D_k^0 + D_k^2 + D_{k-1}^3 + \|D_{k-1}\|^{\#^3} + \diamond \longrightarrow \\
&\qquad \longrightarrow D_k^0 + D_k^1 + \|D_k\|^{\#^1} + \|D_k \triangle D_{k-1}\| \spadesuit + \diamond \\
&\qquad \longrightarrow D_k^0 + D_k^2 + D_k^3 + \|D_k\|^{\#^3} + \diamond \\
&\qquad \longrightarrow D_k^0 + D_k^1 + \|D_k\|^{\#^1} + \underbrace{\|D_k \triangle D_k\|}_{=0} \spadesuit + \diamond \\
&\qquad \longrightarrow \square + D_k^0 + D_k^2 + D_k^3 - yf^3 + \|D_k - yf\|^{\#^3} + yg + yg^1 + \diamond \\
&\qquad \longrightarrow \square + h + yg + yg^1 \longrightarrow \square + h + yg + yg^1 \longrightarrow \cdots .
\end{aligned}$$

If k is even, we obtain the same process as above, except it starts from the second row. Note that if $D_k = 0$, thus $D_{k-1} \neq 0$, then the computation in $\mathcal{M}$ stops returning the value $y = 0$. We divide again two cases:

- if k is odd then $D_{k-1}^3 + \|D_{k-1}\|^{\#^3} + \diamond \to \|k-1\| \spadesuit + \square \to \square + h \to \square + h \to \cdots$;
- if k is even then $\|D_{k-1} \triangle D_{k-2}\| \spadesuit + \diamond \to \diamond \to \square \to \square + h \to \square + h \to \cdots$;

Recall that Condition 1 of Definition 34 requires the state on which the process stabilizes to be of the form $P + h + yf$. In all the previous cases, this condition is satisfied with $P = \square + yg \in \widehat{S}^{\#}$ and $f = g^1$.

If, instead, the process π in $\mathcal{M}$ does not stabilize (and in particular, we have that $D_k \neq 0$ for all k), then $\spadesuit$ will never disappear, thus $\square$ will never be generated, and thus the corresponding process in $\widehat{\mathcal{M}}$ will never stabilize as well. □

Building on Lemma 35, we can prove that DPRA compute a much larger class of partial functions.

Lemma 36 *Given $\phi : \mathbb{N}^k \rightharpoonup \mathbb{N}$ and $\psi_i : \mathbb{N}^n \rightharpoonup \mathbb{N}$ for all $i \in \{1, \ldots, k\}$, all computable by* DPRA, *the composition function $\kappa : \mathbb{N}^n \rightharpoonup \mathbb{N}$ defined by*

$$\kappa(x_1, \ldots, x_n) := \phi(\psi_1(x_1, \ldots, x_n), \ldots, \psi_k(x_1, \ldots, x_n))$$

is also computable by a DPRA.

Proof Let $\Sigma = \{a_1, \ldots, a_n\}$ and $\Sigma^N = \{b_1^N, \ldots, b_k^N\}$ be two disjoint sets such that $\Sigma^{\#} \cong \mathbb{N}^n$ and $(\Sigma^N)^{\#} \cong \mathbb{N}^k$; we additionally define $\Sigma^i = \{a_j^i : a_j \in \Sigma\}$ for all $i \in \{1, \ldots, k\}$. We consider $\mathcal{N} := \mathcal{N}(\phi, h^N, f^N) = (S^N, \Sigma^N, \mathbf{A}^N, D_0^N, f^N)$ the normalized DPRA computing ϕ and $\mathcal{M}^i := \mathcal{M}(\psi_i, h^i, f^i) = (S^i, \Sigma^i, \mathbf{A}^i, D_0^i, f^i)$ the normalized DPRA computing ψ_i for all $i = 1, \ldots, k$; furthermore, we assume that the respective background sets $S^N, S^1, \ldots, S^k$ are disjoint, and we will label the corresponding reactions and elements with the same apex. We denote $\mathcal{K} := (S, \Sigma, \mathbf{A}, D_0, f^N)$ the DPRA defined by:

- $S := S^N \cup S^1 \cup \cdots \cup S^k \cup \Sigma \cup \{\rhd^M, \rhd^N, \square^M, \square^N\}$, where $\rhd^M, \rhd^N, \square^M, \square^N$ are new elements;
- $\Sigma := \{a_1, \ldots, a_n\}$ is the input alphabet;
- $\mathbf{A} := \mathbf{A}_{in} \cup \mathbf{A}_c \cup \mathbf{A}_{\square}^N \cup \mathbf{A}_{\square}^1 \cup \cdots \cup \mathbf{A}_{\square}^k$, where

$$\mathbf{A}_{in} := \begin{cases} (\rhd^M, \varnothing, D_0^1 + \cdots + D_0^k), \\ (\rhd^N, \varnothing, D_0^N), \\ (\square^N, \varnothing, \square^N) \\ (a_i, \varnothing, a_i^1 + \cdots + a_i^k) \qquad \text{for each } i \in \{1, \ldots, n\} \end{cases}$$

$$\mathbf{A}_c := \begin{cases} (h^1 + \cdots + h^k + \square^N, \{\square^M\}, \rhd^N + \square^M) \\ (f^i, \square^N, b_i^N) \qquad \text{for each } i \in \{1, \ldots, k\} \end{cases}$$

$$\mathbf{A}_{\square}^N := \{ (R_a^N, I_a^N \cup \{\square^N\}, P_a^N) \text{ for each } (R_a^N, I_a^N, P_a^N) \in \mathbf{A}^N$$

$$\mathbf{A}_{\square}^i := \{ (R_a^i, I_a^i \cup \{\square^M\}, P_a^i) \text{ for each } (R_a^i, I_a^i, P_a^i) \in \mathbf{A}^i \} \quad \forall i \in \{1, \ldots, k\};$$

- $D_0 := \rhd^M + \square^N$ is the initial state;
- f^N is the same final element as $\mathcal{N}$.

We want to analyse the process starting with $\rhd^M + \square^N + x_1a_1 + \cdots + x_na_n$. At the first step, only the reactions in $\mathbf{A}_{in}$ are enabled; thus k copies of each of the input elements are generated (one for each of the alphabets Σ^i), as well as the initial states $D_0^1, \ldots, D_0^k$ of $\mathcal{M}^1, \ldots, \mathcal{M}^k$. The element $\square^N$ is preserved. Now each of the $\mathcal{M}^i$ is simulated by the reactions in $\mathbf{A}_{\square}^i$, and $\square^N$ is preserved by the third reaction of $\mathbf{A}_{in}$ until the reaction $(h^1 + \cdots + h^k + \square^N, \{\square^M\}, \rhd^N + \square^M) >_{\mathrm{r}} (\square^N, \varnothing, \square^N)$ from $\mathbf{A}_c$ becomes enabled, i.e., when the computation of all $\mathcal{M}^i$ terminates and produces all the elements h^i (note that here it is crucial to operate in a *mr* manner). Recall that by Condition 2 from Definition 34, none of the elements h^i are present in any reactant multiset, thus no combination of the reactions from $\mathbf{A}_{\square}^i$ can be greater than the first reaction from $\mathbf{A}_c$. When this reaction takes place, $\square^N$ is replaced by $\rhd^N + \square^M$, thus the reactions $(\rhd^N, \varnothing, D_0^N)$ and $(f^i, \square^N, b_i^N)$ produce the initial state and the input for $\mathcal{N}$, while $\square^M$ will block any reactions in $\mathbf{A}_{\square}^1, \ldots, \mathbf{A}_{\square}^k$.

Finally, in the successive computation, there will be no element of $S^1 \cup \cdots \cup S^k$ left, and $\mathcal{N}$ will start its computing process using, as desired, the outputs of the other k machines as inputs. □

Lemma 37 *Given $\phi : \mathbb{N}^{k+2} \rightharpoonup \mathbb{N}$ and $\psi : \mathbb{N}^k \rightharpoonup \mathbb{N}$, both computable by* DPRA, *the primitive recursion $\rho : \mathbb{N}^{k+1} \rightharpoonup \mathbb{N}$ defined by*

$$\rho(0, x_1, \ldots, x_k) := \psi(x_1, \ldots, x_k)$$

and

$$\rho(y+1, x_1, \ldots, x_k) := \phi(y, \rho(y, x_1, \ldots, x_k), x_1, \ldots, x_k) \quad \forall y \geq 1$$

is also computable by a DPRA.

Proof Let $\Sigma_k = \{a_1, \ldots, a_k\}$ such that $\Sigma_k^{\#} \cong \mathbb{N}^k$ and let $a, a_0 \notin \Sigma_k$ be two extra symbols. We define two additional alphabets $\Sigma^M := \{a_1^M, \ldots, a_k^M\}$ and $\Sigma^N = \{a^N, a_0^N, a_1^N, \ldots, a_k^N\}$, so that $(\Sigma^N)^{\#} \cong \mathbb{N}^{k+2}$.

We consider $\mathcal{N} := \mathcal{N}(\phi, h^N, f) = (S^N, \Sigma^N, \mathbf{A}^N, D_0^N, f)$ the normalized DPRA computing ϕ and $\mathcal{M} := \mathcal{M}(\psi, h^M, f) = (S^M, \Sigma^M, \mathbf{A}^M, D_0^M, f)$ the normalized DPRA computing ψ; furthermore, we assume that the respective background sets S^N, S^M are disjoint except for the final element, i.e., $S^N \cap S^M = \{f\}$; we will label the corresponding reactions and elements with the same apex. We define $\mathcal{R} := (S, \Sigma, \mathbf{A}, D_0, f')$ the DPRA such that:

- $S := S^N \cup S^M \cup \Sigma \cup \Sigma'_k \cup \{\rhd, \rhd^M, \rhd^N, \square^M, \square^N, \square_{in}^M, \square_{in}^N, \circlearrowleft^N, \#, \#', f'\}$, where $\Sigma'_k = \{a'_1, \ldots, a'_k\}$;
- $\Sigma := \{a, a_1, \ldots, a_k\}$ is the input alphabet;
- $\mathbf{A} := \mathbf{A}_{in} \cup \mathbf{A}_m \cup \mathbf{A}_c \cup \mathbf{A}_{\square}^N \cup \mathbf{A}_{\square}^M$, where

$$\mathbf{A}_{in} := \begin{cases} (\rhd, \varnothing, \rhd^M) \\ (\rhd^M, \{\square^M, \square_{in}^M\}, D_0^M + \square_{in}^M) \\ (\rhd^N, \{\square^N, \square_{in}^N\}, D_0^N + \square_{in}^N) \\ (f', \{\square^N, \square_{in}^N\}, a_0^N) \\ (\#', \{\square^N, \square_{in}^N\}, a^N) \\ (a_i', \{\square^M, \square_{in}^M\}, a_i^M) & \text{for each } i \in \{1, \ldots, k\} \\ (a_i', \{\square^N, \square_{in}^N\}, a_i^N) & \text{for each } i \in \{1, \ldots, k\} \end{cases}$$

$$\mathbf{A}_m := \begin{cases} (a_i, \varnothing, a_i + a_i') & \text{for each } i \in \{1, \ldots, k\} \\ (a, \varnothing, a) \\ (\square^N, \varnothing, \square^N) \\ (\square^M, \varnothing, \square^M) \\ (\square_{in}^N, \varnothing, \square_{in}^N) \\ (\square_{in}^M, \varnothing, \square_{in}^M) \\ (\#, \varnothing, \# + \#') \\ (f, \varnothing, f') \end{cases}$$

$$\mathbf{A}_c := \begin{cases} (h^M + a, \varnothing, \circlearrowleft^N) \\ (h^N + a, \varnothing, \circlearrowleft^N + \square^N) \\ (\circlearrowleft^N + \square^N, \varnothing, \square^M + \rhd^N) \\ (\circlearrowleft^N + \square_{in}^N + \square^N, \varnothing, \# + \#' + \rhd^N) \end{cases}$$

$$\mathbf{A}_{\square}^N := \{(R_a^N, I_a^N \cup \{\square^N\}, P_a^N) : (R_a^N, I_a^N, P_a^N) \in \mathbf{A}^N\}$$

$$\mathbf{A}_{\square}^M := \{(R_a^M, I_a^M \cup \{\square^M\}, P_a^M) : (R_a^M, I_a^M, P_a^M) \in \mathbf{A}^M\};$$

- $D_0 := \rhd + \square^N$ is the new initial state;
- f' is the new final element.

First, we note that the reactions in $\mathbf{A}_m$ are inhibitorless. Thus, they will occur, maintaining the respective elements and generating a copy of them using the alphabet Σ'_k, whenever their reactants are present, as they do not conflict with any other reaction. The elements from Σ'_k will be later translated into the corresponding elements from Σ^M or Σ^N, depending on the phase of the process, by the last two reaction groups from $\mathbf{A}_{in}$.

We now explain how the machine works. In the initial state $\rhd + \square^N + ya + x_1a_1 + \cdots + x_ka_k$, the reaction

$$(\rhd, \varnothing, \rhd^M) + y(a, \varnothing, a) + \sum_{i=1}^{k} x_i(a_i, \varnothing, a_i + a_i')$$

is maximally enabled; in particular, it generates $\rhd^M$, which in the next step enables the generation of the initial state of $\mathcal{M}$. Moreover, the element $\square_{in}^M$ is generated and later preserved via the reaction $(\square_{in}^M, \varnothing, \square_{in}^M) \in \mathbf{A}_m$, preventing the initial state of $\mathcal{M}$ from being generated multiple times. The process that would take place in $\mathcal{M}$ is then simulated with the reactions from $\mathbf{A}_{\square}^M$; whenever $\mathcal{M}$ terminates, it produces a single element h^M. We remark that ya is preserved in any state via $(a, \varnothing, a) \in \mathbf{A}_m$, then we have to consider two cases:

$y = 0$. No reaction from $\mathbf{A}_c$ nor $\mathbf{A}^N_\square$ is enabled, so the system reaches a stable state and the result is the same as the result given by $\mathcal{M}$ with input $(x_1, \dots, x_k)$.

$y \geq 1$. The reaction $(h^M + a, \emptyset, \circlearrowleft^N)$ is enabled (being strictly greater than one instance of $(a, \emptyset, a)$, thus reducing the multiplicity of a, which is initially equal to y, by 1); in the next step, the reaction $(\circlearrowleft^N + \square^N, \emptyset, \square^M + \triangleright^N)$ will take place, thus eliminating $\square^N$ and introducing $\square^M$, whose role is to block all reactions from $\mathbf{A}^M_\square$ (simulating $\mathcal{M}$) and to enable the reactions from $\mathbf{A}^N_\square$ (simulating $\mathcal{N}$). At this point, the initial state of $\mathcal{N}$ is generated by reactions in $\mathbf{A}_{in}$, receiving as input $(0, \psi(x_1, \dots, x_k), x_1, \dots, x_k)$, as the reaction $(f', \{\square_N, \square^N_{in}\}, a^N_0)$ transfers the output of $\mathcal{M}$ to the second coordinate of ϕ, and since # is not present, and consequently #′ neither, the first coordinate is 0. The process will continue until $\mathcal{N}$ eventually produces an output. If a is no longer present ($y - 1 = 0$), the output will be $\phi(0, \psi(x_1, \dots, x_k), x_1, \dots, x_k)$, as desired; if a is still present ($y > 1$), the reaction $(h^N + a, \emptyset, \circlearrowleft^N + \square^N)$ will take place, and as a consequence, the whole simulation of $\mathcal{N}$ will be reinitialized by $\square^N$. The only reactions that will take place are those from $\mathbf{A}_m$ maintaining and copying the input, the reaction $(f, \emptyset, f')$ saving the output of $\mathcal{N}$, and the reaction $(\circlearrowleft^N + \square^N_{in} + \square^N, \emptyset, \# + \#' + \triangleright^N)$ that will reinitialize $\mathcal{N}$ in the next step. Since it will receive the multiplicity of #′ as the first coordinate, the multiplicity of f' as the second one and then the rest of the saved input, the machine will then calculate $\phi(1, \phi(0, \psi(x_1, \dots, x_k), x_1, \dots, x_k), x_1, \dots, x_k)$. This process will repeat until the occurrences of a are exhausted (recall that its multiplicity decreases by 1 at every iteration), so we have proved that $\mathcal{R}$ computes ρ, the primitive recursion operator of ϕ and ψ.

□

Lemma 38 *Given* $\phi : \mathbb{N}^{k+1} \rightharpoonup \mathbb{N}$, *computable by a* DPRA, *the minimization operator* $\mu_\phi : \mathbb{N}^k \rightharpoonup \mathbb{N}$ *defined by*

$$\mu_\phi(x_1, \dots, x_k) := \begin{cases} z & \text{if } \phi(z, x_1, \dots, x_k) = 0 \text{ and } \phi(y, x_1, \dots, x_k) > 0 \text{ for all } y < z \\ \perp & \text{otherwise} \end{cases}$$

where $\perp$ *means that the operator is undefined, is also computable by a* DPRA.

Proof We consider, with the same notation as in the previous lemmas, $\mathcal{N} := \mathcal{N}(\phi, h^N, f^N) = (S^N, \Sigma^N, \mathbf{A}^N, D^N_0, f^N)$ the normalized DPRA computing ϕ, with $\Sigma^N = \{a^N_0, a^N_1, \dots, a^N_k\}$. We define $\mathcal{M} := (S, \Sigma, \mathbf{A}, D_0, f')$ the DPRA given by:

- $S := S^N \cup \Sigma \cup \{\triangleright, \square^N, \square^N_{in}, f, f'\}$;
- $\Sigma := \{a_1, \dots, a_k\}$ is the new input alphabet;
- $\mathbf{A} := \mathbf{A}_{in} \cup \mathbf{A}_m \cup \mathbf{A}_c \cup \mathbf{A}^N_\square$, where

$$\mathbf{A}_{in} := \begin{cases} (\triangleright, \{\square^N, \square^N_{in}\}, D^N_0 + \square^N_{in}) \\ (f, \{\square^N, \square^N_{in}\}, a^N_0) \\ (a_i, \{\square^N, \square^N_{in}\}, a^N_i) \qquad \text{for each } i \in \{1, \dots, k\} \end{cases}$$

$$\mathbf{A}_m := \begin{cases} (a_i, \emptyset, a_i) \qquad \text{for each } i \in \{1, \dots, k\} \\ (f', \emptyset, f + f') \\ (\square^N_{in}, \emptyset, \square^N_{in}) \end{cases}$$

$$\mathbf{A}_c := \begin{cases} (h^N + f^N, \emptyset, \square^N) \\ (\square^N + \square^N_{in}, \emptyset, f + f' + \triangleright) \end{cases}$$

$$\mathbf{A}^N_\square := \{(R^N_a, I^N_a \cup \{\square^N\}, P^N_a) : (R^N_a, I^N_a, P^N_a) \in \mathbf{A}^N\};$$

- $D_0 := \triangleright$ is the new initial state;
- f' is the new final element.

Much like in previous proofs, the reactions from $\mathbf{A}_m$ maintain the input elements. We now analyze the process of $\mathcal{M}$ starting from the state $\triangleright + x_1 a_1 + \cdots + x_k a_k$. At the beginning of the computation, using the first reaction from $\mathbf{A}_{in}$, the initial state of $\mathcal{N}$ is generated, receiving an input with 0 as the first coordinate: indeed, the value of the first coordinate is given by the multiplicity of a^N_0, which at the first step is not produced because f is not present in the initial state. Then $\mathcal{N}$ is simulated with the reactions from $\mathbf{A}^N_\square$. If $\mathcal{N}$ terminates and produces h^N, there are two possible cases:

1. f^N is not present. Then, since neither f' is present, the state remains unchanged, and hence the result is 0.
2. f^N is present. This indicates that $\mathcal{N}$ has returned a non-zero value, so the reaction $(h^N + f^N, \emptyset, \square^N)$ takes place, inhibiting every reaction from $\mathbf{A}^N_\square$ and enabling $(\square^N + \square^N_{in}, \emptyset, f' + f + \triangleright)$ in the next step. Since the former reaction is greater than $(\square^N_{in}, \emptyset, \square^N_{in})$, the element $\square^N_{in}$ is not generated and the reaction generating the initial state of $\mathcal{N}$ is enabled again, receiving as input $(1, x_1, \dots, x_k)$, because now f is present with multiplicity 1, thus a^N_0 is generated with multiplicity 1. This process is iterated until $\mathcal{N}$ eventually outputs 0: at each itera-

tion, the multiplicity of f' is increased by 1 (as well as those of f, via the second reaction from $\mathbf{A}_m$, and of a_0^N via the second reaction from $\mathbf{A}_{in}$), thus when the process terminates, the result is precisely the multiplicity of f'.

We proved that $\mathcal{M}$ computes the minimization operator of ϕ. □

We have arrived at the main result of this section.

Theorem 39 *The class of partial functions computed by deterministic pure reaction automata operating in a maximally reactive manner coincides with the class of general recursive functions.*

Proof By Lemma 32, deterministic pure reaction automata can compute the basic functions: namely, the constant function equal to 0, the successor function and the projection functions. Furthermore, the partial functions computed by deterministic pure automata are closed under composition (Lemma 36), primitive recursion (Lemma 37) and minimization (Lemma 38), therefore they coincide with the class of general recursive functions. □

5 Conclusions

In this work, we introduced and studied a new criterion, the *maximally reactive manner*, for selecting the reactions that take place in a computation step of a reaction automaton. We also defined a new variant of reaction automata, the *pure* reaction automata, where there is no permanence, mimicking (in this aspect) the behaviour of reaction systems. We studied the relation between pure and classical reaction automata working in a maximally reactive manner, showing that the absence of permanence is not a strong limitation: for every reaction automaton working in a maximally parallel (or maximally reactive) manner recognizing a certain language, there always exists a pure reaction automaton working in a maximally reactive manner recognizing the same language. When seen as devices for computing partial functions, *deterministic* pure reaction automata working in a maximally reactive manner are able to compute all general recursive functions.

An interesting direction for future research is to further investigate the relation between the different manners since the choice of one or the other can potentially change the computational power of a reaction automaton (pure or not). The choice of different manners could also give rise to interesting results for the chemical version of reaction automata, in which the set of inhibitors is constrained to be empty. The role of determinism is also to be further explored since it can be another factor that impacts the computational power of reaction automata, possibly also determining which kinds of functions can be computed under time and space (size of the multisets) constraints. Reaction systems were also explored with additional extensions and restrictions, like adding a duration for reactions or forcing each reaction to have only one reactant or one inhibitor. Similar questions can be asked about reaction automata: what is their effect on the computational power? Can we still obtain universality in all cases?

Author Contributions LM and FL conceived the research work; FL and RA proved the main results; GB, RA and LM wrote the manuscript; LM and GB supervised the work; EF, RA, GB and LM reviewed the manuscript.

Funding Open access funding provided by Università degli Studi di Trieste within the CRUI-CARE Agreement. No funding available.

Data availability No Data associated with the manuscript.

Declarations

Conflict of interest No Conflict of interest to declare.

References

Ascone R, Bernardini G, Manzoni L (2024) Fixed points and attractors of reactantless and inhibitorless reaction systems. Theor Comput Sci 984(114):322. https://doi.org/10.1016/j.tcs.2023.114322

Azimi S (2017) Steady states of constrained reaction systems. Theor Comput Sci 701:20–26

Azimi S, Iancu B, Petre I (2014) Reaction system models for the heat shock response. Fund Inform 131(3–4):299–312. https://doi.org/10.3233/FI-2014-1016

Barbuti R, Gori R, Levi F et al (2016) Investigating dynamic causalities in reaction systems. Theor Comput Sci 623:114–145. https://doi.org/10.1016/j.tcs.2015.11.041

Barbuti R, Gori R, Levi F et al (2018) Generalized contexts for reaction systems: definition and study of dynamic causalities. Acta Inform 55:227–267

Barbuti R, Bove P, Gori R et al (2021) Encoding threshold Boolean networks into reaction systems for the analysis of gene regulatory networks. Fund Inform 179(2):205–225. https://doi.org/10.3233/FI-2021-2021

Barbuti R, Bernasconi A, Gori R, et al (2018a) Computing preimages and ancestors in reaction systems. In: International conference on theory and practice of natural computing. Springer, pp 23–35

Bottoni P, Labella A, Rozenberg G (2019) Reaction systems with influence on environment. J Membr Comput 1:3–19
Brijder R, Ehrenfeucht A, Main M et al (2011) A tour of reaction systems. Int J Found Comput Sci 22(07):1499–1517
Brijder R, Ehrenfeucht A, Rozenberg G (2011) Reaction systems with duration. Comput Coop Life 6610:191–202. https://doi.org/10.1007/978-3-642-20000-7_16
Brijder R, Ehrenfeucht A, Rozenberg G (2010) A note on causalities in reaction systems. In: Electronic communications of the EASST 30
Chen H, Doty D, Soloveichik D (2014) Deterministic function computation with chemical reaction networks. Nat Comput 13(4):517–534. https://doi.org/10.1007/S11047-013-9393-6
Clamons S, Qian L, Winfree E (2020) Programming and simulating chemical reaction networks on a surface. J R Soc Interface 17(166):20190,790. https://doi.org/10.1098/rsif.2019.0790
Corolli L, Maj C, Marini F et al (2012) An excursion in reaction systems: from computer science to biology. Theor Comput Sci 454:95–108. https://doi.org/10.1016/j.tcs.2012.04.003
Csuhaj-Varjú E, Ibarra OH, Vaszil G (2006) On the computational complexity of P automata. Nat Comput 5:109–126
Csuhaj-Varjú E, Marion O, Vaszil G (2009) The Oxford handbook of membrane computing, Oxford University Press, chap P Automata
Csuhaj-Varjú E, Vaszil G (2002) P automata or purely communicating accepting p systems. In: Workshop on membrane computing, Springer, pp 219–233
Dutta S, Jankowski A, Rozenberg G et al (2019) Linking reaction systems with rough sets. Fund Inform 165(3–4):283–302
Ehrenfeucht A, Rozenberg G (2007) Reaction systems. Fund Inform 75(1–4):263–280
Ehrenfeucht A, Kleijn J, Koutny M et al (2017) Evolving reaction systems. Theor Comput Sci 682:79–99. https://doi.org/10.1016/j.tcs.2016.12.031
Ehrenfeucht A, Rozenberg G (2004) Basic notions of reaction systems. In: 8th international conference on developments in language theory (DLT). Lecture Notes in Computer Science, vol 3340. Springer, pp 27–29. https://doi.org/10.1007/978-3-540-30550-7_3
Ehrenfeucht A, Rozenberg G (2009) Introducing time in reaction systems. Theor Comput Sci 410(4):310–322. https://doi.org/10.1016/j.tcs.2008.09.043, computational Paradigms from Nature
Formenti E, Manzoni L, Porreca AE (2015) On the complexity of occurrence and convergence problems in reaction systems. Nat Comput 14:185–191
Formenti E, Manzoni L, Porreca AE (2014a) Cycles and global attractors of reaction systems. In: Descriptional complexity of formal systems: 16th international workshop (DCFS), Springer, pp 114–125. https://doi.org/10.1007/978-3-319-09704-6_11
Formenti E, Manzoni L, Porreca AE (2014b) Fixed points and attractors of reaction systems. In: 10th conference on computability in Europe (CiE). Language, life, limits, Springer, pp 194–203. https://doi.org/10.1007/978-3-319-08019-2_20
Freund R, Martín-Vide C, Obtułowicz A, et al (2003) On three classes of automata-like p systems. In: 7th international conference on developments in language theory, DLT 2003 Szeged, Hungary, July 7–11, 2003 Proceedings 7, Springer, pp 292–303
Holzer M, Rauch C (2021) On the computational complexity of reaction systems, revisited. In: Computer Science–Theory and Applications: 16th International Computer Science Symposium in Russia, CSR 2021, Sochi, Russia, June 28–July 2, 2021, Proceedings 16, Springer, pp 170–185
Kleijn J, Koutny M, Rozenberg G (2011) Modelling reaction systems with petri nets. In: BioPPN-2011, 2nd International Workshop on Biological Processes and Petri Nets, Newcastle University
Manzoni L, Pocas D, Porreca AE (2014) Simple reaction systems and their classification. Int J Found Comput Sci 25(04):441–457. https://doi.org/10.1142/S012905411440005X
Manzoni L, Porreca AE, Rozenberg G (2020) Facilitation in reaction systems. J Membr Comput 2(3):149–161
Okubo F (2014) Reaction automata working in sequential manner. RAIRO-Theor Inform Appl-Informatique Théorique et Applications 48(1):23–38
Okubo F, Yokomori T (2018) The computing power of determinism and reversibility in chemical reaction automata. Springer, Cham, pp 279–298. https://doi.org/10.1007/978-3-319-73216-9_13
Okubo F, Kobayashi S, Yokomori T (2012) On the properties of language classes defined by bounded reaction automata. Theor Comput Sci 454:206–221
Okubo F, Fujioka K, Yokomori T (2022) Chemical reaction regular grammars. New Gener Comput 40(2):659–680. https://doi.org/10.1007/S00354-022-00160-8
Okubo F, Kobayashi S, Yokomori T (2012) Reaction automata. Theor Comput Sci 429:247–257. https://doi.org/10.1016/j.tcs.2011.12.045, magic in Science
Okubo F, Yokomori T (2015) Recent developments on reaction automata theory: a survey. In: Recent advances in natural computing: selected results from the IWNC 7 symposium, Springer, pp 1–22
Păun G, Pérez-Jiménez MJ, Rozenberg G (2013) Bridging membrane and reaction systems-further results and research topics. Fund Inform 127(1–4):99–114
Rogers H (1987) Theory of recursive functions and effective computability. MIT Press, Cambridge
Salomaa A (2017) Minimal reaction systems: duration and blips. Theor Comput Sci 682:208–216
Teh WC, Atanasiu A (2017) Irreducible reaction systems and reaction system rank. Theor Comput Sci 666:12–20
Teh WC, Atanasiu A (2020) Simulation of reaction systems by the strictly minimal ones. J Membr Comput 2:162–170
Teh WC, Lim J (2022) Evolvability of reaction systems and the invisibility theorem. Theor Comput Sci 924:17–33. https://doi.org/10.1016/j.tcs.2022.03.039
Yokomori T, Okubo F (2021) Theory of reaction automata: a survey. J Membr Comput 3(1):63–85. https://doi.org/10.1007/S41965-021-00070-6