Demands in Reaction Systems

Lucie Ciencialová[1,*,†], Luděk Cienciala[1,†]

[1]Institute of Computer Science, Faculty of Philosophy and Science in Opava, Silesian University in Opava, Opava, Czech Republic

Abstract

Reaction systems provide a qualitative computational model inspired by the functioning of biochemical reaction networks, where the presence or absence of objects determines the applicability of reactions. In this paper, we introduce reaction systems with demands, an extension in which enabled reactions are equipped with demand vectors expressing requirements and preferences concerning the context provided by the environment in the following computation step. We define several strategies for combining the demands of simultaneously enabled reactions and study the resulting interactive processes. Furthermore, we introduce extended reaction systems with demands by distinguishing terminal and nonterminal symbols and by imposing a forced halting condition, which allows the model to be used as a language-generating formalism. We define languages generated by halting interactive processes and extend the model to distributed reaction systems with demands, where several reaction system components cooperate under a common environment. Finally, we investigate the computational power of distributed reaction systems with demands and show that every right-linear simple matrix language can be generated by such a system. This establishes a connection between reaction systems with demand-driven environments and matrix grammars and demonstrates the ability of the proposed model to represent nontrivial language-generating computations.

Keywords

Reaction Systems, Reactions with Demands, Distributed Reaction Systems with Demands, Right-linear Matrix Grammars

1. Introduction

Reaction systems, introduced by Ehrenfeucht and Rozenberg [1, 2], provide a qualitative model of computation inspired by the functioning of biochemical reaction networks. In contrast to many quantitative formalisms, reaction systems focus on the presence or absence of biochemical entities rather than on their multiplicities. A reaction is enabled whenever all its reactants are present and none of its inhibitors occur in the current state, while the products of all enabled reactions collectively determine the next state of the system.

Since their introduction, reaction systems have been extensively studied from both theoretical and application-oriented perspectives. Numerous extensions have been proposed to enhance their expressive power or to capture additional aspects of biological and computational processes. These include, among others, reaction systems with delays, reversible reaction systems, stochastic variants, and distributed reaction systems.

A particularly interesting research direction concerns the role of the environment in the evolution of reaction systems. In the classical model, the environment influences the computation solely by supplying a context at each computation step. Several authors have investigated richer interactions between the system and its environment. Context automata were introduced to regulate the sequence of contexts provided to distributed reaction systems [3], while evaluating environments assign values or polarities to environmental objects, allowing the environment to influence which reactions are preferred or enabled under different conditions. These extensions demonstrate that enriching the interaction between a reaction system and its environment significantly increases the modelling capabilities of the formalism.

ITAT'26: Information Technologies – Applications and Theory, September 25–29, 2026, Biele Karpaty, Slovakia

*Corresponding author.

†These authors contributed equally.

lucie.ciencialova@fpf.slu.cz (L. Ciencialová); ludek.cienciala@fpf.slu.cz (L. Cienciala)

0000-0002-0877-7063 (L. Ciencialová); 0000-0001-9473-5945 (L. Cienciala)



© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073

published 2026-11-09

CEUR-WS.org/Vol-4323/paper29.pdf

In this paper, we propose a different approach to modelling the interaction between a reaction system and its environment. Instead of allowing the environment to evaluate or prioritize reactions, we let the reactions themselves express their preferences regarding the future context. To this end, every reaction is equipped with a demand vector, whose components specify whether the reaction requires the presence of an object, prefers its absence, or is indifferent to it in the context of the next computation step. Since several reactions may be enabled simultaneously, we introduce several demand evaluation strategies that combine the demands of individual reactions into a single demand vector representing the behaviour of the whole reaction system.

Building upon this framework, we introduce distributed reaction systems with demands. Besides supporting cooperation among several reaction system components, the proposed model distinguishes between terminal and nonterminal symbols, allowing it to serve as a language-generating device. We define the corresponding notion of interactive processes and generated languages, and investigate the computational power of the model. In particular, we prove that the family of right-linear matrix languages is included in the family of languages generated by distributed reaction systems with demands.

2. Basic Notions and Notations

Throughout the paper we assume the reader to be familiar with the basics of formal language theory; for more details consult [4].

An alphabet V is a finite nonempty set. For an alphabet V, V^* denotes the set of words (strings) over V, including the empty word, λ. A language L is a subset of V^*. For a finite set V, $card(V)$ denotes the number of elements of V; if V is the emptyset, then $card(V) = 0$.

A generative grammar is denoted by $G = (N, T, P, S)$ where N and T denote the set of nonterminals and the set of terminals, P is the set of productions, and S is the start symbol. For details on regular, linear, and context-free grammars consult [4]. The family of regular languages is denoted by REG and the family of recursively enumerable languages is denoted by RE.

A right-linear simple matrix grammar, [5], of degree n is an $(n + 3)$-tuple of the form $G = (N_1, \dots, N_n, T, S, M)$ where $N_1, \dots, N_n$ and T are pairwise disjoint alphabets, $S \notin (T \cup N)$, where $N = \bigcup_{i=1}^{n} N_i$, and M consists of matrices of the following forms:

1. $(S \to x)$, $x \in T^*$.
2. $(S \to A_1 \dots A_n)$, $A_i \in N_i$, $1 \leq i \leq n$.
3. $(A_1 \to x_1 B_1, \dots, A_n \to x_n B_n)$, $A_i, B_i \in N_i$, $x_i \in T^*$, $1 \leq i \leq n$.
4. $(A_1 \to x_1, \dots, A_n \to x_n)$, $A_i \in N_i$, $x_i \in T^*$, $1 \leq i \leq n$.

Let $V = N \cup T \cup \{S\}$. For $u, v \in V^*$ we write $u \Longrightarrow v$ if one of the following cases holds: (1) $u = S$ and $(S \to v) \in M$, (2) $u = x_1 A_1 \dots x_n A_n$, $v = x_1 w_1 B_1 \dots x_n w_n B_n$ where $(A_1 \to w_1 B_1, \dots, A_n \to w_n B_n) \in M$, (3) $u = x_1 A_1 \dots x_n A_n$, $v = x_1 w_1 \dots x_n w_n$ where $(A_1 \to w_1, \dots, A_n \to w_n) \in M$, $A_i, B_i \in N_i$, $v, x_i, w_i \in T^*$.

The language generated by G is $L(G) = \{z \in T^* \mid S \Longrightarrow^* z\}$, where $\Longrightarrow^*$ denotes the reflexive transitive closure of $\Longrightarrow$.

For right-linear simple matrix grammars (with erasing rules) the following normal form is known: matrix of type (1) can be omitted, in matrices of type (3) we have $x_i \in T \cup \{\lambda\}$, $1 \leq i \leq n$, and in matrices of type (4) we have $x_i = \lambda$, $1 \leq i \leq n$.

RSM_n denotes the family of languages $L(G)$ for right-linear simple matrix grammars G of degree at most n, $n \geq 1$. The union of language families RSM_n, for all $n \geq 1$ is denoted by RSM_*.

It is known that $RSM_n \subset RSM_{n+1}$, $n \geq 1$. Furthermore, $REG \subset RSM_*$, and RSM_* and CF are incomparable [5].

3. Reaction Systems with Demand

We first recall the notion of a reaction system [1, 2].

A reaction system is a pair $\mathsf{A} = (S, A)$, where S is a finite nonempty set, called the *background set* of A, and A is a finite set of reactions defined over S.

Each reaction $b \in A$ is a triplet $b = (R_b, I_b, P_b)$, where R_b, I_b, and P_b are nonempty subsets of S satisfying $R_b \cap I_b = \emptyset$. The sets R_b, I_b, and P_b are referred to as the *reactants*, *inhibitors*, and *products* of the reaction b, respectively.

A reaction $b \in A$ is said to be *enabled* by a set $T \subseteq S$, denoted by $en_b(T)$, if $R_b \subseteq T$ and $I_b \cap T = \emptyset$. The result of applying b to T, denoted by $res_b(T)$, is equal to P_b whenever b is enabled by T; otherwise, $res_b(T) = \emptyset$. The result of the entire reaction set A on T is defined as $res_A(T) = \bigcup_{b \in A} res_b(T)$.

The dynamics of a reaction system are described by the notion of an interactive process.

Let $\mathsf{A} = (S, A)$ be a reaction system. An interactive process in A is a pair $\pi = (\gamma, \varphi)$ of finite sequences, where $\gamma = c_0, c_1, \ldots, c_{n-1}$ and $\varphi = d_1, \ldots, d_n$ for some $n \geq 1$, such that $c_0, \ldots, c_{n-1}, d_1, \ldots, d_n \subseteq S$, $d_1 = res_A(c_0)$, and $d_i = res_A(d_{i-1} \cup c_{i-1})$ for every $2 \leq i \leq n$.

The sequences γ and φ are called the *context sequence* and the *result sequence* of π, respectively. The initial context c_0 represents the state from which the interactive process starts, while the contexts $c_1, \ldots, c_{n-1}$ represent the successive inputs provided by the environment during the computation.

In order to extend reaction systems by introducing demands on future contexts, we first establish a representation of subsets of the background set by binary vectors.

Let $S = \{a_1, a_2, \ldots, a_n\}$ be a finite nonempty background set with a fixed ordering. For every subset $x \subseteq S$, we define its characteristic vector $\chi(x) \in \{0,1\}^{card(S)}$ such that for every $1 \leq i \leq card(S)$

$$\chi(x)_i = \begin{cases} 1 & \text{if } a_i \in x, \\ 0 & \text{otherwise.} \end{cases}$$

Thus, every subset of S can be represented as a binary vector describing the presence or absence of individual objects.

Example 1. *Let $S = \{a, b, c, d\}$.*

For the subset $x_1 = \{a, c\}$ we obtain $\chi(x_1) = (1, 0, 1, 0)$, since only the first and third objects are present. For the empty set $x_2 = \emptyset$, $\chi(x_2) = (0, 0, 0, 0)$. Finally, for $x_3 = S$ we have $\chi(x_3) = (1, 1, 1, 1)$.

The objective of the extension introduced in this paper is to allow reactions not only to produce objects but also to express preferences regarding the context of the following computation step.

Definition 1. *Let $D = \{0, +, -\}$ denote the set of demand symbols. A reaction system with demand is a pair $\mathsf{A} = (S, A)$, where S is the background set and A is a finite set of reactions with demand.*

A reaction with demand is a quadruple $b = (R_b, I_b, P_b, dem_b)$, where $R_b, I_b, P_b \subseteq S$ satisfy $R_b \cap I_b = \emptyset$. As in classical reaction systems, the sets R_b, I_b, and P_b are called reactants, inhibitors, and products, respectively, and, in contrast to classical reaction systems, P_b is allowed to be empty. The additional component $dem_b \in D^{card(S)}$ is called the demand vector of reaction b.

The meaning of symbols in the demand vector is as follows:

- $dem_{b_i} = +$ expresses the requirement (inclusive demand, desire) that object a_i should occur in the context of the next computation step,
- $dem_{b_i} = -$ expresses exclusion or aversion to the presence of object a_i,
- $dem_{b_i} = 0$ expresses neutrality or indifference with respect to object a_i.

Consequently, the demand vector specifies the desired properties of the context used in the subsequent step of the computation.

The operational semantics of reaction systems with demand remains identical to classical reaction systems.

A reaction b is enabled in a state $x \subseteq S$, denoted by $en_b(x)$, iff $R_b \subseteq x$ and $I_b \cap x = \emptyset$.

The result of reaction b on x is defined as $res_b(x) = \begin{cases} P_b & \text{if } en_b(x), \\ \emptyset & \text{otherwise.} \end{cases}$

The set of enabled reactions of A in state x is $en_A(x) = \{b \in A \mid en_b(x)\}$.

The global result of the reaction system is then defined by $res_A(x) = \bigcup_{b \in A} res_b(x)$.

Unlike classical reaction systems, enabled reactions also contribute their demand vectors. Before defining the resulting demand of the whole system, we first introduce an auxiliary structure accumulating demands from enabled reactions.

For $x \subseteq S$, let $sumd_A(x) = (M_1, M_2, \ldots, M_{card(S)})$ be the demand accumulation vector, where $M_i = \bigcup_{b \in en_A(x)} \{dem_{b_i}\}$. Hence, M_i contains all symbols appearing at position i in the demand vectors of enabled reactions.

The resulting demand of the reaction system is denoted by $dem_A(x) \in D^{card(S)}$.

Several evaluation strategies may be used.

Strong mode This mode accepts only unanimous demands. For every position i

$$dem_A(x)_i = \begin{cases} + & \text{if } M_i = \{+\}, \\ - & \text{if } M_i = \{-\}, \\ 0 & \text{otherwise.} \end{cases}$$

Thus, a requirement or exclusion is accepted only if all enabled reactions agree.

Weak mode In the weak mode positive requests dominate neutral and negative ones.

$$dem_A(x)_i = \begin{cases} + & \text{if } + \in M_i, \\ - & \text{if } - \in M_i \text{ and } + \notin M_i, \\ 0 & \text{otherwise.} \end{cases}$$

Positive weak mode In this mode conflicting positive and negative requirements cancel each other.

$$dem_A(x)_i = \begin{cases} + & \text{if } + \in M_i \text{ and } - \notin M_i, \\ - & \text{if } - \in M_i \text{ and } + \notin M_i, \\ 0 & \text{otherwise.} \end{cases}$$

Compare mode The compare mode is based on the frequency of positive and negative requirements rather than on $sumd_A(x)$.

Let $p_i = card(\{b \in en_A(x) \mid dem_b(i) = +\})$ and $n_i = card(\{b \in en_A(x) \mid dem_b(i) = -\})$. Then

$$dem_A(x)_i = \begin{cases} + & \text{if } p_i > n_i, \\ - & \text{if } n_i > p_i, \\ 0 & \text{otherwise.} \end{cases}$$

After obtaining the demand vector of the system, the next context is generated according to this demand. We denote by $Ctx(A, x)$ the set of all contexts satisfying the demand vector $dem_A(x)$.

The interactive process follows the classical definition of reaction systems, however, the contexts are provided externally but they are based on demand vectors. An interactive process in A is a pair $\pi = (\gamma, \varphi)$ where $\gamma = c_0, c_1, \ldots, c_{n-1}$ and $\varphi = d_1, d_2, \ldots, d_n$ such that $d_1 = res_A(c_0)$ and for $2 \leq i \leq n$ $d_i = res_A(d_{i-1} \cup c_{i-1})$. The context $c_i \in Ctx(A, d_{i-1} \cup c_{i-1})$ for $i > 0$.

Since the symbol 0 expresses neutrality, a demand vector may correspond to several possible contexts.

Example 2. *Let $S = \{a, b, c, d\}$ and $dem_A(x) = (+, 0, -, +)$. Then objects a and d must occur in the context, object c must be absent, while object b is unrestricted. Hence the admissible contexts are $\{a, d\}$ and $\{a, b, d\}$.*

Let $dem_A(x) = (+, 0, 0, -)$. Then object a must be present and object d must be absent, while objects b and c are optional. Therefore the possible contexts are $\{a\}$, $\{a, b\}$, $\{a, c\}$, and $\{a, b, c\}$.

The notion of an interactive process introduced above naturally leads to the notion of a halting interactive process.

An interactive process $\pi = (\gamma, \varphi)$ is said to be *halting* if for the selected context c_n no reaction of A is enabled in the state $d_n \cup c_n$, i.e., $en_A(d_n \cup c_n) = \emptyset$.

We associate a set of words (language) with every halting interactive process.

Let $X \subseteq S$. By $Ord(X)$ we denote the set of all words over the alphabet S that contain each element of X exactly once. Equivalently, $Ord(X)$ is the set of all permutations of the elements of X. In particular, $Ord(\emptyset) = \{\lambda\}$.

Definition 2. *Let $\pi = (\gamma, \varphi)$ be a halting interactive process of a reaction system* A *and let $\varphi = d_1, d_2, \ldots, d_n$ be its result sequence. The language associated with π, denoted by $L(\mathsf{A}, \pi)$, is defined as*

$$L(\mathsf{A}, \pi) = \{w_1 w_2 \cdots w_n \mid w_i \in Ord(d_i),\ 1 \leq i \leq n\}.$$

In other words, every word of $L(\pi)$ is obtained by choosing an arbitrary ordering of the elements of each result set d_i and concatenating the selected orderings in the order determined by the computation.

Example 3. *Let $\varphi = \big(\{a, b\}, \{c\}, \{d, e\}\big)$. Then $Ord(\{a, b\}) = \{ab, ba\}$, $Ord(\{c\}) = \{c\}$, $Ord(\{d, e\}) = \{de, ed\}$, and therefore*

$$L(\pi) = \{abcde,\ abced,\ bacde,\ baced\}.$$

Definition 3. *Let* A *be a reaction system with demand and let $c_0 \subseteq S$ be an initial context.*

The language associated with a pair (A, c_0), denoted by $L(\mathsf{A}, c_0)$, is the union of languages generated by all halting interactive processes whose initial context is c_0.

The demand vectors provide a mechanism for controlling the evolution of the computation by restricting the contexts that may occur in subsequent steps. In this way, the order in which reactions become enabled can be regulated without modifying the underlying reaction system semantics. As shown in the following example, demand vectors can be used to enforce a cyclic activation of reactions and thereby generate the language $\{(abc)^n \mid n \geq 0\}$.

Example 4. *Consider the reaction system with demand* $\mathsf{A} = (S, A)$ *over the background set $S = \{a, b, c\}$, where $A = \{r_1, r_2, r_3\}$ and*

$$\begin{aligned} r_1 &= (\{a, c\}, \{b\}, \{a\}, (0, +, -)), \\ r_2 &= (\{a, b\}, \{c\}, \{b\}, (-, 0, +)), \\ r_3 &= (\{b, c\}, \{a\}, \{c\}, (0, -, 0)). \end{aligned}$$

Let the initial context be $c_0 = \{a, c\}$ and assume the positive weak mode.

In the initial state $\{a, c\}$ only reaction r_1 is enabled, hence $d_1 = res_A(\{a, c\}) = \{a\}$. The corresponding demand vector is $(0, +, -)$, which allows contexts containing b and forbids c, i.e., $c_1 \in \{\{b\}, \{a, b\}\}$. In both cases the next state is $\{a, b\}$.

In state $\{a, b\}$ only reaction r_2 is enabled, hence $d_2 = \{b\}$. Its demand vector is $(-, 0, +)$, requiring presence of c and absence of a, thus $c_2 \in \{\{c\}, \{b, c\}\}$, and the next state is $\{b, c\}$.

In state $\{b, c\}$ only reaction r_3 is enabled, hence $d_3 = \{c\}$. The demand vector $(0, -, 0)$ allows any context without b, i.e., $c_3 \in \{\emptyset, \{a\}, \{c\}, \{a, c\}\}$.

If $c_3 \in \{\{a\}, \{a, c\}\}$, the system returns to state $\{a, c\}$ and the computation continues. Otherwise, if $c_3 \in \{\emptyset, \{c\}\}$, no reaction is enabled and the computation halts.

Each full cycle produces the word abc, and therefore the language generated by A is

$$L(\mathsf{A}, \{a, c\}) = \{(abc)^n \mid n \geq 1\}.$$

4. Extended Reaction Systems

Before introducing the formal definition of an extended reaction system with demands, we briefly motivate the need for such a refinement.

In classical reaction systems, all elements of the background set S play a uniform role: they may simultaneously participate in the computation as reactants, inhibitors, and products, and there is no intrinsic distinction between symbols that structure the computation and symbols that represent its observable output. However, when reaction systems are used as generative devices, this uniformity becomes restrictive. In particular, it is often desirable to distinguish between auxiliary symbols that control the evolution of the system and symbols that form the final generated word or language.

Moreover, in the presence of demand-driven context selection, the computation may continue indefinitely unless a principled termination condition is imposed. To address this, we introduce a separation of the background set into nonterminal and terminal symbols, and we additionally define a notion of forced halting that guarantees that computations contributing to the generated language end in configurations consisting solely of terminal symbols.

Let $\mathrm{A} = (S, A)$ be a reaction system with demands and let $N \subset S$ be a set of nonterminal symbols. The set of terminal symbols is defined as $T = S \setminus N$.

An interactive process $\pi = (\gamma, \varphi)$ with $\gamma = c_0, c_1, \ldots, c_{n-1}$ and $\varphi = d_1, d_2, \ldots, d_n$ is said to be *forced halting* if its last configuration contains no nonterminal symbols, i.e.,

$$(c_n \cup d_n) \cap N = \emptyset.$$

In other words, the computation is forced to terminate at a step where the current context together with the products of each component consists exclusively of terminal symbols.

We define the mapping $\sigma_T : 2^S \to 2^{T^*}$ as follows.

For every $g \subseteq S$,

$$\sigma_T(g) = Ord(g \cap T),$$

where $Ord(g \cap T)$ denotes the set of all words containing each terminal symbol from $g \cap T$ exactly once in an arbitrary order. In particular, $\sigma_T(g) = \{\lambda\}$ if $g \cap T = \emptyset$.

5. Distributed Reaction Systems with Demands and Their Associated Languages

We extend the notion of reaction systems with demands to a distributed setting. A distributed reaction system with demands consists of several reaction system components that share a common background set and interact with a common environment. During each computation step, every component evaluates its enabled reactions independently.

The environment, however, provides a single context shared by all components. Therefore, the local demands of individual components have to be combined into a single global demand determining the admissible contexts for the next computation step. In the remainder of this paper, we consider only the weak mode of constructing the global demand vector.

We define the concept of a distributed reaction system and its operation, with slight modifications of the notions and notations originally introduced in [3] and then in [6].

Definition 4. *A distributed reaction system with demands (a DRSD, for short) is a pair* $\Delta = (S, \mathrm{A})$ *where* S *is a finite nonempty set, called the background set of* Δ*, and* $\mathrm{A} = (A_1, \ldots, A_n)$*.* A_i*,* $1 \leq i \leq n$*, is a finite nonempty set of reactions over* S*, called the* i*th component of* Δ*.*

Notice that (S, A_i) is a reaction system; if no confusion arises, then A_i can also be called the ith reaction system of Δ.

We extend the notion of the demand accumulation vector to DRSD, where the accumulated demands are computed from the reactions enabled in the individual components. For a distributed reaction system $\mathsf{A} = (A_1, A_2, \dots, A_k)$ and a sequence of sets $X = (x_1, x_2, \dots, x_k)$, where $x_j \subseteq S$ for $j = 1, \dots, k$, let

$$sumd_\Delta(X) = (M_1, M_2, \dots, M_{card(S)})$$

be the demand accumulation vector, where

$$M_i = \bigcup_{j=1}^{k} \bigcup_{b \in en_{A_j}(x_j)} \{dem_{b_i}\},$$

for every $i = 1, \dots, card(S)$.

The demand vector $dem_\Delta(X)$ and the context set $Ctx(\Delta, X)$ are defined analogously to the corresponding notions for reaction systems.

Let $\Delta = (S, \mathsf{A})$ be a distributed reaction system with demands, where $\mathsf{A} = (A_1, \dots, A_k)$. The current state of Δ is represented by a pair (c, D), where $c \subseteq S$ is the current context supplied by the environment and $D = (d_1, d_2, \dots, d_k)$, where $d_i \subseteq S$ is the result produced by the ith component in the preceding computation step, for $1 \leq i \leq k$.

Let (c, D) and (c', D') be two states of Δ, where $D = (d_1, \dots, d_k)$ and $D' = (d'_1, \dots, d'_k)$. We denote by D_c a vector with components $d_i \cup c,\ 1 \leq i \leq k$. We say that (c', D') is a *successor state* of (c, D), denoted by $(c, D) \Longrightarrow (c', D')$, if the following conditions are satisfied:

1. For every component A_i,
$$d'_i = res_{A_i}(d_i \cup c).$$
2. The context c' belongs to the set of contexts determined by the global demand of the distributed reaction system, i.e.,
$$c' \in Ctx(\Delta, D_c),$$
where $dem_\Delta(D_c)$ is obtained from the demand vectors of all components according to the selected demand combination strategy.

An interactive process in Δ is a pair $\pi = (\gamma, \varphi)$ of finite sequences, where $\gamma = c_0, c_1, \dots, c_{n-1}$ and $\varphi = D_1, D_2, \dots, D_n$ for some $n \geq 1$, such that: $(c_i, D_i) \Longrightarrow (c_{i+1}, D_{i+1}),\ 0 \leq i \leq n-1$

We denote by Q_Δ the set of all states of the distributed reaction system with demands Δ.

In the following, we assign languages to distributed reaction systems with demands.

Definition 5. *Let $\Delta = (S, \mathsf{A})$, $\mathsf{A} = (A_1, \dots, A_n)$, $n \geq 1$, is a distributed reaction system with demands over S with $N \subset S$ and $T = S \setminus N$ We define mappings ϕ, ϕ_i as follows.*

1. *Mapping $\phi : Q_\Delta \to \underbrace{T^* \times \cdots \times T^*}_{n \text{ times}}$ is defined as follows:*
   for each state $s = (c, (d_1, \dots, d_n))$, let $\phi(s) = (\sigma_T(d_1), \dots, \sigma_T(d_n))$.
2. *Mapping $\phi_i : Q_\Delta \to T^*$, $1 \leq i \leq n$, is defined as follows:*
   for each state $s = (c, (d_1, \dots, d_n))$, let $\phi_i(s) = \sigma_T(d_i)$.

Definition 6. *Let $\Delta = (S, \mathsf{A})$, $\mathsf{A} = (A_1, \dots, A_n)$, $n \geq 1$, is a distributed reaction system with demands over S with $N \subset S$ and $T = S \setminus N$. Let alphabet Σ and mappings ϕ, ϕ_i, $1 \leq i \leq n$ be defined as in Definition 5.*

Let $\pi : s_0 \Longrightarrow \cdots \Longrightarrow s_m$, $m \geq 1$, be a interactive process with forced halting of Δ, where $s_j = (c_j, (d_{j,1}, \dots, d_{j,n}))$, $1 \leq j \leq m$, is a state of Δ.

We define $H(\pi) = (\phi_1(s_0) \dots \phi_1(s_m)), \dots, (\phi_n(s_0) \dots \phi_n(s_m))$.

The language of Δ over T is defined as follows:

$$L(\Delta, T) = \{\phi_1(s_0) \dots \phi_1(s_m) \dots \phi_n(s_0) \dots \phi_n(s_m) \mid \pi : s_0 \Longrightarrow \cdots \Longrightarrow s_m \in \Pi_\Delta, m \geq 1\}.$$

The language of A_i over T, $1 \leq i \leq n$, is

$$L(A_i, T) = \{\phi_i(s_0) \dots \phi_i(s_m) \mid \pi : s_0 \Longrightarrow \cdots \Longrightarrow s_m \in \Pi_\Gamma, m \geq 1\}.$$

The following theorem shows that distributed reaction systems with demands are sufficiently powerful to generate all right-linear simple matrix languages. The proof is based on a constructive transformation of a right-linear simple matrix grammar in normal form into a distributed reaction system with demands. The construction uses individual components to simulate the parallel application of the rules within matrices, while demand vectors control the selection of matrices in consecutive computation steps.

Theorem 1. *For every right-linear simple matrix language $L \subseteq T^*$ we can give a distributed reaction system with demands $\Delta = (S, \mathsf{A})$ such that $L = L(\Delta, T)$ holds.*

Proof. Let L be a right-linear language. It is well known that every right-linear language is generated by a right-linear matrix grammar in the normal form. Let $G = (N, T, \mathsf{M}, S_0)$ be such a grammar generating L.

Without loss of generality, we assume that G contains exactly one initial matrix of type 1,$m_1 = (S_0 \to A_1 A_2 \cdots A_n)$.

Let $\mathsf{M} = \{m_1, m_2, \ldots, m_p\}$ be the set of all matrices of G.

We denote by

$$M = \{M_1, M_2, \ldots, M_p\}$$

the set of matrix symbols corresponding to the matrices in M, where each symbol m_i uniquely represents the matrix M_i. For every matrix $m_i \in \mathsf{M}$, we define $next(M_i) = \{ M_j \in M \mid m_j$ may be applied immediately after $m_i \}$.

We construct a distributed reaction system with demands $\Delta = (S, \mathsf{A})$, where $\mathsf{A} = (A_1, \ldots, A_n, A_{n+1})$, consisting of $n + 1$ components.

The background set S consists of

- all terminal symbols,
- all nonterminal symbols of the grammar,
- one additional symbol corresponding to every matrix of G.
- auxiliary nonterminals U, X

The matrix symbols are used solely to control the order in which matrices are simulated. We fix an arbitrary ordering of the elements of S, which determines the positions of the corresponding entries in all demand vectors.

The first component simulates the first rule of every matrix. For the unique initial matrix $M_1 = (S_0 \to A_1 A_2 \cdots A_n)$ the component contains the reaction $((\{M_1\}, M \setminus \{M_1\}, \emptyset), dem)$ where the demand vector dem is defined as follows.

For every matrix symbol corresponding to a matrix in $next(m_1)$, the corresponding entry of d is equal to 0, whereas all remaining matrix positions contain $-$. Furthermore, the entries corresponding to the nonterminals $A_1, \ldots, A_n$ are set to $+$. All remaining positions of d are equal to $-$.

For every matrix of type 2 $M_i = (A_1 \to x_1 B_1,\ A_2 \to x_2 B_2,\ \ldots,\ A_n \to x_n B_n)$, the jth component, $1 \leq j \leq n$, contains the reaction $(\{M_i, A_j\},\ M \setminus \{M_i\},\ f(x_j),\ dem)$, where $f : T \cup \{\lambda\} \to 2^T$ is defined by $f(a) = \{a\} \quad (a \in T), \qquad f(\lambda) = \emptyset$, and the demand vector dem is defined as follows.

The value on the position corresponding to the nonterminal B_j is $+$, the positions corresponding to the matrix symbols of all matrices in $next(M_i)$ has value 0, and all other positions of d are equal to $-$.

For every matrix of type 3 $M_i = (A_1 \to \lambda,\ A_2 \to \lambda,\ \ldots,\ A_n \to \lambda)$, the jth component, $1 \leq j \leq n$, contains the reaction $(\{M_i, A_j\},\ M \setminus \{M_i\},\ \emptyset,\ dem)$, where the demand vector dem consists entirely of the symbol $-$.

The choice of the matrix applied in the next step of the derivation is nondeterministic. Therefore, the reaction system uses demand vectors to communicate to the environment that the next context is expected to contain exactly one matrix symbol corresponding to one of the applicable matrices. More precisely, the demand vector contains the value 0 at the positions corresponding to the matrix symbols in $next(M_i)$.

If the environment provides a context containing either several different matrix symbols or no matrix symbol at all, the computation enters an infinite loop. To detect such an invalid context, we introduce an additional symbol U.

For every pair of matrices $M_i, M_j \in M$, $i \neq j$, we add to every component the reaction $(\{M_i, M_j\}, \{U\}, \{U\}, dem)$.

Furthermore, for every nonterminal symbol $A_i \in N$, every component contains the reaction $(\{A_i\}, M, \{U\}, dem)$.

The demand vector dem is the same for both types of reactions and is defined as follows: all its entries are equal to $-$ except the entry corresponding to the symbol U, which is equal to $+$.

Finally, the infinite loop is ensured by adding the reaction $(\{U\}, \{X\}, \{U\}, dem)$, where dem is the same demand vector as above.

X is an auxiliary symbol that ensures the non-emptiness of the inhibitor set.

The initial context of Δ is the context containing the symbol corresponding to the initial matrix M_1.

By the construction of the components of Δ and the associated demand vectors, every forced halting interactive process of Δ uniquely determines a sequence of matrices constituting a terminating derivation of G. Conversely, every terminating derivation of G gives rise to a unique forced halting interactive process of Δ. Hence, there is a one-to-one correspondence between the forced halting interactive processes of Δ and the terminating derivations of G. Consequently, the forced halting interactive processes of Δ are exactly those computations that generate the words of L.

Moreover, the construction of the reactions guarantees that any computation step producing an invalid context either violates the demand constraints or enters the infinite loop generated by the auxiliary symbol U. Hence, such computations do not contribute to the language of Δ.

Let T be the set of terminal symbols of G. For every forced halting interactive process

$$\pi = (\gamma, \varphi)$$

with result sequence

$$\varphi = D_1, \ldots, D_m,$$

the generated words are obtained by applying the mapping σ_T to the results of the individual components and concatenating the obtained words in the order of the computation steps.

Thus,

$$L(\Delta, T) = \{w \mid w \text{ is generated by a forced halting interactive process of } \Delta\}$$

contains exactly the terminal words generated by the corresponding terminating derivations of G.

Consequently,

$$L(\Delta, T) = L(G).$$

□

6. Conclusion

In this paper, we introduced reaction systems with demands, an extension of the classical reaction system model in which reactions are able not only to produce objects but also to express preferences concerning future environmental contexts. The introduced demand vectors provide a mechanism for influencing the interaction between a reaction system and its environment without changing the basic semantics of reaction application. We defined several strategies for combining demands of simultaneously enabled reactions and investigated the corresponding interactive processes.

To enable the use of reaction systems with demands as language-generating devices, we introduced extended reaction systems with demands, where the background set is divided into terminal and nonterminal symbols and computations are required to terminate in configurations containing only terminal symbols. Based on this extension, we defined languages generated by halting interactive processes.

Furthermore, we extended the model to distributed reaction systems with demands, where several reaction system components operate in parallel and share a common environment. The global demand of the system is obtained by combining the demands generated by individual components, which allows the environment to coordinate the behaviour of the whole distributed system. We considered the weak mode of demand combination and defined the corresponding notion of generated language.

The main theoretical result shows that distributed reaction systems with demands are able to generate every right-linear simple matrix language. The construction demonstrates that demand vectors can be used to control the selection of derivation steps and to simulate the parallel application of matrix rules. This result establishes a link between reaction systems with demand-driven environments and matrix grammars and confirms the expressive power of the proposed model.

Several directions remain open for future research. One natural continuation is the investigation of the generative power of reaction systems with demands under other demand evaluation strategies, such as strong mode, positive weak mode, or compare mode. Another interesting direction is the characterization of the exact language families generated by restricted variants of distributed reaction systems with demands, including systems with bounded numbers of components or limited background sets. Further research may also focus on relationships with other computational models inspired by biological processes and on possible applications of demand-based interaction in modelling adaptive systems.

Acknowledgments

This work was supported by Silesian University in Opava within the project IGS/1/2026 and SGS/9/2024.

Declaration on Generative AI

During the preparation of this work, the authors used ChatGPT-4o in order to: Grammar and spelling check. After using these tool(s)/service(s), the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] A. Ehrenfeucht, G. Rozenberg, Basic notions of reaction systems, in: C. Calude, E. Calude, M. J. Dinneen (Eds.), Developments in Language Theory, 8th International Conference, DLT 2004, Auckland, New Zealand, December 13-17, 2004, Proceedings, volume 3340 of *Lecture Notes in Computer Science*, Springer, 2004, pp. 27–29.

[2] A. Ehrenfeucht, G. Rozenberg, Reaction systems, Fundam. Informaticae 75 (2007) 263–280.

[3] A. Meski, M. Koutny, W. Penczek, Model Checking for Temporal-Epistemic Properties of Distributed Reaction Systems, Tecnical Report Series CS-TR- 1526, Newcastle University, 2019.

[4] J. E. Hopcroft, R. Motwani, J. D. Ullman, Introduction to automata theory, languages, and computation, 3rd Edition, Pearson international edition, Addison-Wesley, 2007.

[5] O. H. Ibarra, Simple matrix languages, Inf. Control. 17 (1970) 359–394. doi:10.1016/S0019-9958(70)80034-1.

[6] L. Ciencialová, L. Cienciala, E. Csuhaj-Varjú, Language classes of extended distributed reaction systems, International Journal of Foundations of Computer Science 36 (2025) 1017–1040. doi:10.1142/S0129054123460024.