Asynchrony and Persistence in Reaction Systems

Maciej Koutny [a] Marta Pietkiewicz-Koutny [a] Alex Yakovlev [b]

[a] *School of Computing, Newcastle University, Newcastle upon Tyne, NE1 7RU, United Kingdom*

[b] *School of Engineering, Newcastle University, Newcastle upon Tyne, NE1 7RU, United Kingdom*

Abstract

Reaction systems are a model of interactive computation, where the interaction between a system — itself built up of a number of reactions — and its environment is modeled through context sequences provided by the environment. The standard execution semantics of reaction systems is synchronous, i.e., at each computational step all the enabled reactions are executed. In this paper, we 'de-synchronise' such an execution model by allowing only a subset of enabled reactions to be executed. We then study the resulting asynchronous model assuming two fundamental execution policies. The first one allows any subset of reactions to be executed, and the second one draws each subset from a pre-defined pool. We also introduce and discuss the notion of persistence of reactions and sets of reactions in the resulting models of asynchronous reaction systems. In particular, we demonstrate that reaction persistence can be implemented.

Key words: reaction system, interactive computation, context sequence, interactive process, synchronous execution, asynchronous execution, traceability, persistence, interrupt

1 Introduction

The history of computer science offers us many useful lessons and instructive examples of posing some basic questions, such as, what modelling paradigms

Email addresses: maciej.koutny@ncl.ac.uk (Maciej Koutny), marta.koutny@ncl.ac.uk (Marta Pietkiewicz-Koutny), alex.yakovlev@ncl.ac.uk (Alex Yakovlev).

best reflect the nature of computational processes. Such questions are asked every time we face with new limitations of the existing models as we advance in our ability to build computing systems with better performance, higher complexity, lower power etc. These limitations are typically caused by the ever-increasing scope of applications of computing in society and by the progress achieved in technologies used to construct computing systems. Both these aspects, changes in applications and underlying technologies, and the response of engineers to address them in software and hardware development, set a scene for innovation in modelling computing systems. An important computational paradigm, which manifests itself in hardware and software products vividly, is the notion of interaction. The vision that this notion is basic to computer science has been expressed by pioneers of interactive computation, cf. Milner's work [1].

Why is interaction basic? Because it views computations in terms of basic structural and behavioural relationships between the actors and data objects involved in computations. It is these relationships that ultimately determine the whole 'fabric' of computational processes in space and in time. It is evident that lessons learnt from nature where we can see exemplars of interacting agents and data objects, are highly fruitful in formulating models of interactive computations. Reaction systems are particularly prominent here. They describe computational processes in their basic form, namely, reactions are agents which interact with each other by exchanging reactants and inhibitors, expressed as elements of sets [2].

The original motivation behind reaction systems was to model interactions between biochemical reactions taking place in the living cell (see, e.g., [3–6]). The model of reaction systems explicitly formalised two basic mechanisms underpinning such interactions, viz. facilitation and inhibition: the product of one reaction may contain reactants of another reaction (hence facilitating it), and this product may also contain inhibitors of yet another reaction (hence inhibiting it). The dynamic behaviour of a reaction system is formalised through interactive processes, where a transition from the current state to its successor state is determined by: the transformations implied by the reactions of the reaction system, and the contribution of the environment in the form of a context set. Thus reaction systems are *open systems* with their behaviour (interactive processes) being influenced by the environment. Since their introduction, research on reaction systems was guided by biological motivations (see, e.g., [6–10]) as well as the curiosity and need to understand the essence underlying computations. As a result, reaction systems have now become an inspiring model of interactive computation (see, e.g., [11–24]).

Today's definition and treatment of reaction systems as collections of interacting actors and data objects is based on a rather restricted view of spatio-temporal aspects of this interaction. Basically, their interaction and evolution

in time is mediated by the central clocking mechanism, which determines the moments when all reactions must update their state. This method of interaction or rather policy of execution, is called here as '(fully or globally) synchronous'. There are certain reasons behind this method, i.e., mostly to keep the basic model maximally simple.

Traditional reaction systems have a number of underlining principles that govern them in their operation. They are: (i) maximum concurrency, (ii) complete renewal of state (no permanency), (iii) both promotion and inhibition, (iv) 0/1 (binary) resource availability, (v) no contention for resources. Although these principles have many advantages in keeping the model simple and elegant, they can be a limiting factor on the way of advancing the basic reaction system model to capture computations that are more asynchronous. Being asynchronous is usually being closer to real-life, from both the needs of applications as well as from the standpoint of implementation technologies. This paper tackles the idea of *de-synchronising* reaction systems, for which two main motivations are as follows: (i) spatial distribution of computational resources and effects of locality and availability of resources in real-world environments; and (ii) impracticality of creating a global mechanism or controller for clocking reactions all in-step. Similar sort of motivations lie behind the developments of models of asynchronous systems in very large scale integrated circuits [25]. Asynchrony would typically allow enabled actions in such systems to execute in either order, retain the state of enabled actions while other actions are executed, involve fine-grained causality between elementary events and permit arbitration for shared resources. There have been examples of developing de-synchronisation mechanisms for VLSI circuits designed initially as globally synchronous [26]. While they contain some useful ideas in principle, the problem that we are facing here, that of de-synchronizing reaction systems has its own generic features and therefore deserves a special theoretical consideration.

First and foremost, we see de-synchronisation as a measure to enhance the capability of the reaction system model to be a foundational model for interactive computing. Hence, the notion of interaction is crucial here. To perform de-synchronisation we need to address this problem looking at three main aspects: behaviour, structure and correctness.

In terms of behaviour, a clear notion of the execution of a reaction is required, i.e., in order to interact with other reactions a reaction has to be equipped with the means of letting other reactions know about its occurrence or progress. In terms of structure, a clear notion of locality and its levels of application, i.e., granularity issue, has to be defined. In terms of correctness, it is important to define a form of traceability of a reaction or a group thereof and its or their safe (undisturbed) execution.

This paper addresses these three issues systematically, in a step-wise manner, by following an important guiding principle:

> *... do as much as possible using the entities present in the states rather than relying on reaction 'management' structures ...*

The importance of this principle stems from our intuitive desire to support the inherently distributed nature of reaction systems, thereby rendering them maximally self-managed. The truly asynchronous model should therefore underpin interaction not only between the reaction system and its environment but also transcend into the whole manner of interactions between reactions themselves.

What is the key property of a reaction system that should tie together its distributed nature of interactions in the absence of global synchronisation, and yet lead it to a coherent self-managed operation? We believe that the key of such property is persistence of reactions. This belief is based on our prior experience in developing models of asynchronous systems [27].

Persistence in system behaviour is of paramount importance in real life. Two examples: 'a non-persistent digital circuit installed on an airplane can lead to a disaster' and 'a given species of animal may not survive a change in changing environmental conditions'. Here, we aim at capturing in a fundamental (or 'minimal') way the notion of persistence by considering it in the realm of reaction systems. Two important issues are necessary for defining persistence in reaction systems.

Firstly, in order to approach the definition of persistence we require to advance our understanding of reaction execution. Indeed, the semantical treatment of reaction systems does not support an explicit notion of executing a reaction. This is something which distinguishes reaction systems from other formal models such as Petri nets, not to mention process algebras. It is therefore crucial to re-assess the meaning of persistence as behaviour-related.

Secondly, persistence is not necessarily a global property. For example, the environment may be triggering different activities with different periods, and only then we need to verify that the system (or its part) is persistent. Persistence (and other such properties) is also context-dependent. If arbitrary contexts are allowed, almost no systems are persistent. Therefore persistence should be investigated relative to context sequences. For example, in Petri nets persistence is action-based (cf. Landweber's work [28]). In logic circuits, persistence is linked to logic gate switching, thus also involves actions and their enabledness when the circuit changes its states [27]. What unites those approaches is a clear notion of traceability of actions in states. In terms of reaction systems, this basically addresses a question, what reaction was or could be assumed to happen when we have observed two consecutive states?

Finally, it is important when talking about practical aspects of different periodicity of triggers (of reactions), which may be implemented by context sequences, to state that it is important even in the synchronous case.

In the synchronous setup (all actions which can execute do so) 'all reactions are trivially persistent', i.e., 'synchrony is like fairness for everyone' (we are not waiting with execution of any enabled actions/reactions). In the asynchronous setup this is no longer the case, and sometimes we can only state that it is possible that a given reaction has been executed.

Closely linked to context-dependency, and gradually restricting the context of reaction triggering, we define two levels of de-synchronisation. We follow here a route of defining different strata of globally asynchronous and locally synchronous (GALS) reaction systems, inspired by our previous work with Petri net models of GALS in [29]. To this end, we move from the notions of traceability and persistence of individual reactions to those of sets of reactions.

This study does by no means address all the challenges and interesting problems on the way to defining asynchronous reaction systems. Since our journey begins here at the fully synchronous reaction systems, and we prefer a step-by-step de-synchronisation we will leave the problem of de-synchronisation for distributed reaction systems [30,31] for further study.

Furthermore, we realise that the realm of the original definition of a reaction system, motivated by biological reasons [3–6], requires the fulfilment of the property of non-permanency. Therefore, a challenging modelling issue on the way of true de-synchronisation, is to find a way of interleaving reactions that can be enabled concurrently within the same reaction system. In this work, we methodologically 'avoid' resolving this issue purely *within* the biological realm of the reaction system itself. The issue is therefore approached in this paper by the potentially 'non-biological' means, i.e., the so-called asynchrony management techniques, such as incorporation of additional entities into the original reaction system. Other possible ways are left for the future work.

The paper is organised in the following way. The next section provides main definitions relating to the basic (synchronous) reaction systems. Section 3 introduces fully de-synchronised reaction systems. In the resulting model, we introduce and discuss the notion of reaction persistence, and also outline the mechanisms for action interruption and suspension. Section 4 restricts the full asynchrony of reaction systems, by assuming that at each computational step, the set of actions scheduled for execution comes from a pre-defined pool. We also extend the notion of reaction persistence to sets of reactions. Section 5 concludes the paper.

2 Reaction Systems

In this section we recall some basic notions concerning reaction systems [5,24].

Let S be a nonempty finite *background set* comprising *entities*. A subset of S is sometimes called a *state*. In the original biochemical interpretation, entities represent, e.g., atoms, ions, and molecules that may be present in the system states. A *reaction* over S is a triple $b = (R, I, P)$ such that R, I, and P are nonempty subsets of S with $R \cap I = \varnothing$. The sets R and P are called the *reactant / inhibitor / product* sets of b, and can be denoted by R_b / I_b / P_b, respectively.

Throughout this paper we assume S is a fixed background set, A is a fixed set of reactions over S, and $\mathcal{A} = (S, A)$ is a fixed *reaction system*.

A reaction $b \in A$ is *enabled* at $X \subseteq S$ if $R_b \subseteq X$ and $I_b \cap X = \varnothing$, i.e., if all the reactants are present and all inhibitors are absent. The *result* of applying a subset of reactions $B \subseteq A$ to $X \subseteq S$ is $res_B(X) = \bigcup\{P_b \mid b \in B \cap en(X)\}$, where $en(X)$ is the set of all reactions *enabled* at X. Note that when $b \in B$ is enabled by X, it contributes its product P_b to the successor state; otherwise it does not contribute anything. The dynamic behaviour of reaction systems is expressed in the following way.

An *interactive process* in $\mathcal{A}$ is a pair $\pi = (\gamma, \delta)$ of sequences of subsets of S, $\gamma = C_0 \dots C_n$ and $\delta = D_0 \dots D_n$ $(n \geq 1)$, such that, for every $1 \leq i \leq n$, $D_i = res_A(D_{i-1} \cup C_{i-1})$. Then γ is the *context sequence*, δ is the *result sequence*, and the sequence $W_0 \dots W_n$ (where $W_i = C_i \cup D_i$, for all i) is the *state sequence* of π. Note that π is fully determined by γ and D_0.

To formalise the notion of *simulation* between processes of reaction systems, for a pair of sequences of subsets of the background set, $\pi = (C_0 \dots C_n, D_0 \dots D_n)$, and a subset X of S, we denote by $\pi \cap X$ the pair of sequences $(C_0 \cap X \dots C_n \cap X, D_0 \cap X \dots D_n \cap X)$. Intuitively, $\pi \cap X$ is the *projection* of π onto X.

The *transition system* of $\mathcal{A}$ is $tr_{\mathcal{A}} = (2^S, \{(X, C, res_A(X) \cup C) \mid X, C \subseteq S\})$, where the second component is the set of arcs labelled by contexts thrown in by the environment. Note that $tr_{\mathcal{A}}$ has no initial state, and that it is deterministic, i.e., , for each state X and each context C there is exactly one arc labelled by C outgoing from X.

3 Full de-synchronisation

Basic de-synchronisation removes the assumption that all the enabled reactions are executed at each computation stage of an interactive process. The change to the original definition is syntactically slight but semantically far reaching.

Definition 3.1 *An* asynchronous interactive process *in $\mathcal{A}$ is a pair $\pi = (\gamma, \delta)$ of sequences of subsets of S, $\gamma = C_0 \dots C_n$ and $\delta = D_0 \dots D_n$ ($n \geq 1$), such that, for every $1 \leq i \leq n$, $D_i = res_{A_i}(D_{i-1} \cup C_{i-1})$ for some nonempty $A_i \subseteq A$. We denote this by $\pi \in asynproc(\gamma, D_0)$.*

Thus $asynproc(\gamma, D_0)$ comprises all asynchronous interactive processes derived from D_0 and the context sequence γ.

Definition 3.1 allows any nonempty subset of reactions to be 'active' at any point during the execution. Also, intuitively, in 'interesting' cases one might be using some kind of *execution policy*, with an appropriate notion of *control enabledness*. The assumption that A_i is nonempty intuitively means that one cannot suppress a system totally. Also this is consistent with the notion of policy in Petri nets [32].

Each interactive process in $\mathcal{A}$ is also an asynchronous interactive process. The converse does not hold in general as the following example shows.

Example 3.2 *Let $\mathcal{A} = (\{\mathrm{X}, \mathrm{Y}\}, \{b, c\})$ be a reaction system, where:*

$$b = (\{\mathrm{X}\}, \{\mathrm{Y}\}, \{\mathrm{X}\}) \quad \textit{and} \quad c = (\{\mathrm{X}\}, \{\mathrm{Y}\}, \{\mathrm{Y}\}) \; .$$

Then $\pi = (\varnothing\varnothing\varnothing, \{\mathrm{X}\}\{\mathrm{X}\}\{\mathrm{X}, \mathrm{Y}\})$ is an asynchronous interactive process in $\mathcal{A}$ (with $A_1 = \{b\}$ and $A_2 = \{b, c\}$). However, π is not *an interactive process in any reaction system $\mathcal{A}' = (S, A')$. The reason is that this would mean that $res_{A'}(\{\mathrm{X}\}) = \{\mathrm{X}\}$ and, at the same time, $res_{A'}(\{\mathrm{X}\}) = \{\mathrm{X}, \mathrm{Y}\}$. having said that, we show below that it is possible to see π as part of an interactive process in which two additional entities are used.*

Although π in Example 3.2 is not an interactive process, it can be simulated by a *projection* of an interactive process. To simulate the above π as an interactive process, we can use the process $\pi' = (\{\mathrm{TRG}_b\}\{\mathrm{TRG}_b, \mathrm{TRG}_c\}\varnothing, \{\mathrm{X}\}\{\mathrm{X}\}\{\mathrm{X}, \mathrm{Y}\})$ of $\mathcal{A}' = (\{\mathrm{X}, \mathrm{Y}, \mathrm{TRG}_b, \mathrm{TRG}_c\}, \{b', c'\})$, where $b' = (\{\mathrm{X}, \mathrm{TRG}_b\}, \{\mathrm{Y}\}, \{\mathrm{X}\})$ and $c' = (\{\mathrm{X}, \mathrm{TRG}_c\}, \{\mathrm{Y}\}, \{\mathrm{Y}\})$. Using trigger entities in a context sequence we can select which enabled reactions are allowed to be executed. The simulation follows from the fact that $\pi = \pi' \cap \{\mathrm{X}, \mathrm{Y}\}$.

In general, all asynchronous interactive processes can be simulated by the standard interactive processes. Basically, for each reaction b we introduce a

unique 'trigger' entity TRG_b and add it to the entity set R_b. Then, referring to Definition 3.1, for each $0 < i \leq n$, we add to C_{i-1} the set $\{\text{TRG}_b \mid b \in A_i\}$. The adjusted system is executed according to the standard (synchronous) semantics 'hiding' the trigger entities.

The described simulation is not exact, as it requires the introduction of additional entities. However, after projecting away the added entities, the simulating process becomes identical to the process being simulated, as shown in the following proposition.

Proposition 3.3 *Let $\pi = (C_0 \dots C_n, D_0 \dots D_n)$ be an asynchronous interactive process in $\mathcal{A} = (S, A)$. Moreover, let $A_i \subseteq A$ be such that $D_i = res_{A_i}(D_{i-1} \cup C_{i-1})$, for every $1 \leq i \leq n$. Then $\pi' = (C'_0 \dots C'_n, D_0 \dots D_n)$ is an interactive process in the reaction system $\mathcal{A}' = (S', A')$, assuming that:*

$$\begin{aligned} S' &= S \uplus \{\text{TRG}_b \mid b \in A\} \\ A' &= \{b' = (R_b \cup \{\text{TRG}_b\}, I_b, P_b) \mid b \in A\} \\ C'_i &= C_i \cup \{\text{TRG}_b \mid b \in A_{i+1}\} \quad \textit{for} \;\; 0 \leq i < n \\ C'_n &= C_n \; . \end{aligned}$$

Moreover, $\pi = \pi' \cap S$.

Proof Below, $W_i = C_i \cup D_i$ and $W'_i = C'_i \cup D_i$, for every $0 \leq i \leq n$. We also observe that, for every $b' \in A'$, we have $R_b, I_b, P_b \subseteq S$ and $P_{b'} = P_b$.

From the assumptions made and the definition of the result function we have, for every $1 \leq i \leq n$, $D_i = res_{A_i}(W_{i-1}) = \bigcup\{P_b \mid b \in A_i \cap en(W_{i-1})\}$, where $A_i \subseteq A$. We intend to prove that, for every $1 \leq i \leq n$, $D_i = res_{A'}(W'_{i-1})$, and so we need to show that, for every $1 \leq i \leq n$, $res_{A_i}(W_{i-1}) = res_{A'}(W'_{i-1})$. Recalling that, for every $b \in A$, we have exactly one reaction $b' = (R_b \cup \{\text{TRG}_b\}, I_b, P_b) \in A'$, we take $1 \leq i \leq n$ and proceed as follows.

To show the $(\subseteq)$ inclusion, we observe that:

$$\begin{aligned} d \in res_{A_i}(W_{i-1}) &\iff d \in \bigcup\{P_b \mid b \in A_i \cap en(W_{i-1})\} \\ &\iff \exists b \in A_i : \; R_b \subseteq W_{i-1} \;\wedge\; I_b \cap (W_{i-1}) = \varnothing \;\wedge\; d \in P_b \; . \end{aligned}$$

Since $I_b \subseteq S$ and $S \cap \{\text{TRG}_b \mid b \in A\} = \varnothing$ and $P_{b'} = P_b$ and $b \in A_i \subseteq A$ (so $\text{TRG}_b \in C'_{i-1}$), we have that

$$\begin{aligned} &\exists b' \in A' : R_b \cup \{\text{TRG}_b\} \subseteq W_{i-1} \cup \{\text{TRG}_b\} \subseteq W'_{i-1} \;\wedge\; I_b \cap (W'_{i-1}) = \varnothing \;\wedge\; d \in P_{b'} \\ &\iff d \in \bigcup\{P_{b'} \mid b' \in A' \cap en(W'_{i-1})\} \iff d \in res_{A'}(W'_{i-1}) \; . \end{aligned}$$

To show the ($\supseteq$) inclusion, we observe that:

$$\begin{aligned}
d \in res_{A'}(W'_{i-1}) &\iff d \in \bigcup\{P_{b'} \mid b' \in A' \cap en(W'_{i-1})\} \\
&\iff \exists b' \in A' : b' \in en(W'_{i-1}) \ \wedge \ d \in P_{b'} = P_b \\
&\iff \exists b' \in A' : R_b \cup \{\mathrm{TRG}_b\} \subseteq W'_{i-1} \ \wedge \ I_b \cap (W'_{i-1}) = \varnothing \ \wedge \ d \in P_b \\
&\iff \exists b \in A_i : R_b \cup \{\mathrm{TRG}_b\} \subseteq W_{i-1} \cup \{\mathrm{TRG}_b \mid b \in A_i\} \ \wedge \\
&\qquad I_b \cap (W_{i-1} \cup \{\mathrm{TRG}_b \mid b \in A_i\}) = \varnothing \ \wedge \ d \in P_b \ .
\end{aligned}$$

Since $R_b \subseteq S$, $I_b \subseteq S$, and $\{\mathrm{TRG}_b \mid b \in A_i\} \cap S = \varnothing$, we have:

$$\begin{aligned}
&\exists b \in A_i : R_b \subseteq W_{i-1} \ \wedge \ I_b \cap (W_{i-1}) = \varnothing \ \wedge \ d \in P_b \\
&\iff d \in \bigcup\{P_b \mid b \in A_i \cap en(W_{i-1})\} \iff d \in res_{A_i}(W_{i-1}) \ .
\end{aligned}$$

Hence the result holds. □

We could introduce an alternative simulation using additional entities blocking reactions through suitably extended inhibitor sets of reactions. In fact, we could combine these two approaches, so that some reactions would be triggered and some blocked by the context.

We have shown that individual asynchronous interactive processes can be simulated by projected interactive processes. The same simulation cannot be done at the transition system level since, after erasing the additional entities, the resulting transition system of a reaction system becomes nondeterministic. That is, in general, there will be more than one arc labelled by the same (projected) context outgoing from a given (projected) state.

The capture of asynchronous interactive process from Definition 3.1 can be made more concise after introducing the notation $X \to Y$ which means that $X, Y \subseteq S$ are such that $Y = res_B(X)$, for a nonempty $B \subseteq A$ (note that there can be more than one set B with such a property).

Corollary 3.4 *Let $\pi = (\gamma, \delta)$, where $\gamma = C_0 \ldots C_n$ and $\delta = D_0 \ldots D_n$ ($n \geq 1$) are two sequences of subsets of S. Then $\pi \in asynproc(\gamma, D_0)$ if and only if $D_{i-1} \cup C_{i-1} \to D_i$, for every $1 \leq i \leq n$.*

3.1 Tracing reactions in asynchronous interactive processes

One of the main aims of this paper is to define the notion of persistence in the context of reaction systems. To discuss persistence, we first look at more basic issue which is:

in an asynchronous interactive process, at which point can we assert that a given reaction has occurred?

The above is a problem of incomplete information (we do not know which sets A_i of active reactions were actually used), and our only information is the state information. This is dramatically different from Petri nets view, for example, where behaviours are typically modelled by execution sequences that are based on transitions (i.e., reactions).

Tracing (or discovering) that a reaction has occurred on the basis of the transformation of system state is a fundamental problem in the analysis of system behaviour (it is know, e.g., under the name of opacity in security [33]). Here we propose two natural captures (weak and strong) of this concept in the setting of reaction systems. In what follows, $\mathit{reactsets}(X,Y) = \{B \subseteq A \mid \mathit{res}_B(X) = Y\}$ is the set of *possibly activated reaction sets*, for $X, Y \subseteq S$ satisfying $X \to Y$.

Definition 3.5 *Let $X, Y \subseteq S$ be such that $X \to Y$, and $b \in \mathit{en}(X)$. Then b* occurred weakly *(*occurred strongly*) in $X \to Y$ if $b \in \bigcup \mathit{reactsets}(X,Y)$ (resp. $b \in \bigcap \mathit{reactsets}(X,Y)$). We then denote $b \in \mathit{weakocc}(X,Y)$ (resp. $b \in \mathit{strongocc}(X,Y)$).*

Thus, if $b \in \mathit{weakocc}(X,Y)$ then b *may* have occurred in the transformation $X \to Y$ since all its products are in the successor state (this can be seen as *effect-based* tracing). If $b \in \mathit{strongocc}(X,Y)$ then b *must* have occurred in the transformation $X \to Y$ (this can be seen as *reaction-based* tracing). One could also introduce other notions of occurrence in-between the weak and strong ones. For example, p-occurrence where p is the proportion of those sets in $\mathit{reactsets}(X,Y)$ which comprise b. Intuitively, this captures the contribution of b to $\mathit{reactsets}(X,Y)$.

Theorem 3.6 *Let $\mathcal{A} = (S,A)$ be a reaction system and $X, Y \subseteq S$ be such that $X \to Y$. Then $\mathit{strongocc}(X,Y) \subseteq \{b \in \mathit{en}(X) \mid P_b \subseteq Y\} = \mathit{weakocc}(X,Y)$.*

Proof The inclusion $\mathit{strongocc}(X,Y) \subseteq \mathit{weakocc}(X,Y)$ follows directly from Definition 3.5. We will prove that $\mathit{weakocc}(X,Y) = \{b \in \mathit{en}(X) \mid P_b \subseteq Y\}$.

Recall that from $X \to Y$ we know that there is $B \subseteq A$ such that

$$Y = \mathit{res}_B(X) = \bigcup \{P_b \mid b \in B \cap \mathit{en}(X)\} \; . \tag{1}$$

To show the ($\subseteq$) inclusion, we observe that:

$$
\begin{aligned}
& b \in \mathit{weakocc}(X,Y) \iff b \in \bigcup \mathit{reactsets}(X,Y) \ \wedge \ b \in \mathit{en}(X) \\
& \iff \exists B \subseteq A : \ b \in B \ \wedge \ \mathit{res}_B(X) = Y \ \wedge \ b \in \mathit{en}(X) \\
& \iff \exists B \subseteq A : \ b \in B \ \wedge \ Y = \bigcup \{P_b \mid b \in B \cap \mathit{en}(X)\} \ \wedge \ b \in \mathit{en}(X) \\
& \implies \exists B \subseteq A : \ b \in B \ \wedge \ b \in \mathit{en}(X) \ \wedge \ P_b \subseteq Y \\
& \implies b \in \{e \in \mathit{en}(X) \mid P_e \subseteq Y\} \ .
\end{aligned}
$$

To show the ($\supseteq$) inclusion, we take $b \in \{b \in \mathit{en}(X) \mid P_b \subseteq Y\}$ and observe that from $X \to Y$ we have that there exists $B' \subseteq A$ such that $Y = \mathit{res}_{B'}(X)$. Hence there is $B' \subseteq A$ such that:

$$
Y = \mathit{res}_{B'}(X) =_{\mathit{Eq.(1)}} \bigcup \{P_{b'} \mid b' \in B' \cap \mathit{en}(X)\} \ \wedge \ b \in \mathit{en}(X) \ \wedge \ P_b \subseteq Y \ .
$$

Consequently, there is $B = B' \cup \{b\} \subseteq A$ such that:

$$
Y = \mathit{res}_B(X) = \bigcup \{P_b \mid b \in B \cap \mathit{en}(X)\} \ \wedge \ b \in \mathit{en}(X) \ \wedge \ b \in B \ .
$$

The above is equivalent to $b \in \mathit{weakocc}(X,Y)$. □

The inclusion in Theorem 3.6 cannot be reversed, as shown in the next example.

Example 3.7 *Let* $\mathcal{A} = (\{\mathrm{W}, \mathrm{X}, \mathrm{Y}, \mathrm{Z}\}, \{b, c\})$ *be a reaction system, where*

$$
b = (\{\mathrm{X}\}, \{\mathrm{W}\}, \{\mathrm{Z}\}) \quad \textit{and} \quad c = (\{\mathrm{X}\}, \{\mathrm{Y}\}, \{\mathrm{Z}\}) \ .
$$

Then, for $X = \{\mathrm{X}\}$ *and* $Y = \{\mathrm{Z}\}$*, we have:*

$$
\mathit{reactsets}(X,Y) = \mathit{reactsets}(\{\mathrm{X}\}, \{\mathrm{Z}\}) = \{\{b\}, \{c\}, \{b, c\}\}
$$

as $\mathit{res}_{\{b\}}(X) = \mathit{res}_{\{c\}}(X) = \mathit{res}_{\{b,c\}}(X) = \{\mathrm{Z}\}$*. Hence* $\mathit{weakocc}(X,Y) = \{b, c\}$ *whereas* $\mathit{strongocc}(X,Y) = \varnothing$*.*

One can make the two notions of reaction occurrence collapse by introducing reaction fingerprinting. Basically, for each reaction b we can introduce a unique 'fingerprint' entity FPR_b and add it to the product set P_b. In the resulting reaction system, $\mathit{strongocc}(X,Y) = \mathit{weakocc}(X,Y)$, whenever $X \to Y$.

Proposition 3.8 *Let* $\mathcal{A} = (S, A)$ *and* $\mathcal{A}' = (S', A')$ *be reaction systems such that:*

$$
\begin{aligned}
S' &= S \uplus \{\mathrm{FPR}_b \mid b \in A\} \\
A' &= \{b' = (R_b, I_b, P_b \cup \{\mathrm{FPR}_b\}) \mid b \in A\} \ .
\end{aligned}
$$

Moreover, let $X, Y \subseteq S'$ be such that $X \to Y$ (in $\mathcal{A}'$). Then $strongocc(X,Y) = weakocc(X,Y)$ (in $\mathcal{A}'$).

Proof As, by Theorem 3.6, we have $strongocc(X,Y) \subseteq weakocc(X,Y)$, all we need to show is that $weakocc(X,Y) \subseteq strongocc(X,Y)$.

To the contrary, suppose that $b' \in weakocc(X,Y) \setminus strongocc(X,Y)$. From $b' \in weakocc(X,Y)$ we have that $b' \in \bigcup reactsets(X,Y)$ and $b' \in en(X)$. This is equivalent to

$$\exists B \subseteq A' : b' \in B \ \wedge \ res_B(X) = Y \ \wedge \ b' \in en(X)$$

which, in turn, is equivalent to

$$\exists B \subseteq A' : b' \in B \ \wedge \ Y = \bigcup \{P_c \cup \{\mathrm{FPR}_c\} \mid c' \in B \cap en(X)\} \ \wedge \ b' \in en(X) \ .$$

Hence there is $B \subseteq A'$ such that $b' \in B$ and $P_b \cup \{\mathrm{FPR}_b\} \subseteq Y$ and $b' \in en(X)$.

Consequently, $b' \in weakocc(X,Y)$ implies $\mathrm{FPR}_b \in Y$. Now, if at the same time $b' \notin strongocc(X,Y)$, then there must be $B' \subseteq A'$ such that $b' \notin B'$ and

$$Y = res_{B'}(X) = \bigcup \{P_c \cup \{\mathrm{FPR}_c\} \mid c' \in B' \cap en(X)\} \ .$$

This and $P_d \cap \{\mathrm{FPR}_c \mid c \in A\} = \varnothing$, for every $d \in A$, implies $\mathrm{FPR}_b \notin Y$, yielding a contradiction. □

We also note that, crucially, the behaviours of reaction systems $\mathcal{A}$ and $\mathcal{A}'$ in Proposition 3.8 are strongly linked as the asynchronous interactive processes of the latter projection simulate the asynchronous interactive processes of the former.

3.2 Persistent reactions in asynchronous interactive processes

Persistence in system behaviour is of paramount importance in real life. For example, a non-persistent digital circuit installed on an airplane can lead to a disaster, and a given species of animal may not survive a change in changing environmental conditions. Here we aim at capturing in a fundamental (or 'minimal') way the notion of persistence by considering it in the realm of reaction systems. It is therefore crucial to re-assess the meaning of persistence as a behaviour-related notion.

To start with, the definition of persistence we prefer to use in the case of reaction systems is based on a reformulation of the standard Petri net definition:

> *if $X \to Y$ and b is enabled at X and has not occurred in the transformation from X to Y, then b is still enabled at Y.*

We re-interpret the above as follows: [1]

if $X \to Y$ and b is enabled at X but not at Y, then b has occurred in the transformation from X to Y.

The definition of reaction persistence within a process is based on more fundamental reaction persistence between two states (a move which can be understood as one step in a process).

Definition 3.9 *Let $X, Y, Z \subseteq S$ be such that $X \to Y$, and $b \in A$. Then b is* weakly persistent *(*strongly persistent*) reaction w.r.t. X, Y, Z if $b \in en(X)$ and $b \notin en(Y \cup Z)$ implies $b \in weakocc(X, Y)$ (resp. $b \in strongocc(X, Y)$). We then denote $b \in weakpers(X, Y, Z)$ (resp. $b \in strongpers(X, Y, Z)$).*

Note that, in the above definition, the set Z plays the role of a context set.

We lift this to the level of whole processes with persistence being defined w.r.t. a given context sequence and the initial state of a reaction system.

Definition 3.10 *Let $\gamma = C_0 \dots C_n$ $(n \geq 1)$ be a sequence of subsets of S and $D_0 \subseteq S$. Then $b \in A$ is* weakly persistent *(resp.* strongly persistent*) reaction w.r.t. γ and D_0 if, for every $\pi = (\gamma, D_0 \dots D_n) \in asynproc(\gamma, D_0)$ and for every $1 \leq i \leq n$, $b \in weakpers(C_{i-1} \cup D_{i-1}, D_i, C_i)$ (resp. $b \in strongpers(C_{i-1} \cup D_{i-1}, D_i, C_i)$). We then denote $b \in weakpers(\gamma, D_0)$ (resp. $b \in strongpers(\gamma, D_0)$).*

One might ask whether it is at all possible to implement persistence in the realm of reaction system. The answer is positive, as shown in the next example.

Example 3.11 *Let $\mathcal{A} = (\{\text{GO}, \text{DONE}\}, \{b_{pers}, c_{supp}\})$ be a reaction system, where:*

$$b_{pers} = (\{\text{GO}\}, \{\text{DONE}\}, \{\text{DONE}\}) \quad \textit{and} \quad c_{supp} = (\{\text{GO}\}, \{\text{DONE}\}, \{\text{GO}\}) \; .$$

It is also assumed that the context never throws in DONE. [2] *Then the persistent behaviour of b_{pers} is achieved by including* GO *in the context. This enables reactions b_{pers} and c_{supp} (provided that* DONE *is not present), and c_{supp} may be executed until b_{pers} is executed (note that we need to activate a nonempty set of reactions A_i at every step). For instance, we have the following asynchronous*

[1] Clearly, both interpretations are equivalent in the setting of, for example, Petri nets.

[2] Note that such a property of context sequences (and other properties of context sequences considered in this paper) can easily be described by logic formulas (c.f. [34]).

interactive process:

$$\gamma = \{\text{GO}\} \quad \varnothing \quad \varnothing \quad \varnothing \quad \{\text{GO}\} \quad \varnothing \quad \varnothing \quad \{\text{GO}\} \quad \varnothing \quad \varnothing$$
$$\delta = \quad \varnothing \quad \{\text{GO}\} \; \{\text{GO}\} \; \{\text{GO}\} \; \{\text{GO}\} \; \{\text{GO}\} \; \{\text{DONE}\} \quad \varnothing \quad \{\text{GO}\} \; \{\text{GO}, \text{DONE}\}$$

Note that the successive executions of b_{pers} are separated by at least one state in which it is not enabled. Also, observe that at $(C_6, D_6) = (\varnothing, \text{DONE})$ none of the reactions is enabled, so any nonempty subset of $\{b_{pers}, c_{supp}\}$ can play the role of A_7, leading, in all cases, to the empty product set $D_7 = \varnothing$. Then, only the context can enable the reactions. In this example, the reactions can only be disabled from within the reaction system.

Formally, we have:

Proposition 3.12 *Let $\mathcal{A}$ be the reaction system in Example 3.11, $D_0 \subseteq \{\text{GO}, \text{DONE}\}$ and $\gamma = C_0 \dots C_n$ ($n \geq 1$) be a sequence of subsets of $\{\text{GO}\}$. Then $b_{pers} \in strongpers(\gamma, D_0)$.*

Proof Let $\pi = (\gamma, \delta) = (C_0 \dots C_n, D_0 \dots D_n) \in asynproc(\gamma, D_0)$ and $W_i = C_i \cup D_i$, for every $0 \leq i \leq n$.

The possible states of π, represented as (C, D) pairs are:

$$\kappa_1 = (\varnothing, \varnothing) \qquad \kappa_2 = (\{\text{GO}\}, \varnothing) \qquad \kappa_3 = (\varnothing, \{\text{GO}\})$$
$$\kappa_4 = (\{\text{GO}\}, \{\text{GO}\}) \qquad \kappa_5 = (\varnothing, \{\text{DONE}\}) \qquad \kappa_6 = (\{\text{GO}\}, \{\text{DONE}\})$$
$$\kappa_7 = (\varnothing, \{\text{GO}, \text{DONE}\}) \qquad \kappa_8 = (\{\text{GO}\}, \{\text{GO}, \text{DONE}\}) \; .$$

Let $1 \leq i \leq n$. From the assumptions we have the following four cases:
Case 1: $W_{i-1} = \varnothing$ and $en(W_{i-1}) = \varnothing$.
Case 2: $W_{i-1} = \{\text{GO}\}$ and $en(W_{i-1}) = \{b_{pers}, c_{supp}\}$.
Case 3: $W_{i-1} = \{\text{DONE}\}$ and $en(W_{i-1}) = \varnothing$.
Case 4: $W_{i-1} = \{\text{GO}, \text{DONE}\}$ and $en(W_{i-1}) = \varnothing$.

To show now that $b_{pers} \in strongpers(\gamma, D_0)$ we use the following succession of equivalences obtained from the application of the introduced definitions:

$$b_{pers} \in strongpers(\gamma, D_0) \iff$$
$$\forall 1 \leq i \leq n : b_{pers} \in strongpers(W_{i-1}, D_i, C_i) \iff$$
$$\forall 1 \leq i \leq n : b_{pers} \in en(W_{i-1}) \setminus en(W_i) \implies b_{pers} \in strongocc(W_{i-1}, D_i) \iff$$
$$\forall 1 \leq i \leq n : b_{pers} \in en(W_{i-1}) \setminus en(W_i) \implies b_{pers} \in \bigcap reactsets(W_{i-1}, D_i) \iff$$
$$\forall 1 \leq i \leq n : b_{pers} \in en(W_{i-1}) \setminus en(W_i) \implies b_{pers} \in \bigcap \{B \subseteq A \mid res_B(W_{i-1}) = D_i\} \; .(*)$$

Looking at the implication in the last formula, we observe that in Cases 1,3,4 the implication holds as the antecedent is false.

Now we consider Case 2, where $en(W_{i-1}) = \{b_{pers}, c_{supp}\}$.

We have $b_{pers} \in en(W_{i-1})$ for $(C_{i-1}, D_{i-1}) = \kappa_2, \kappa_3, \kappa_4$.

We have $b_{pers} \notin en(W_i)$ for $(C_i, D_i) = \kappa_1, \kappa_5, \kappa_6, \kappa_7, \kappa_8$.

We observe that $\kappa_1 = (\varnothing, \varnothing)$ cannot succeed any of $\kappa_2, \kappa_3, \kappa_4$ for any of the three candidates for a set of active reactions B ($\{b_{pers}\}$, $\{c_{supp}\}$, $\{b_{pers}, c_{supp}\}$), as both b_{pers} and c_{supp} are enabled at $\kappa_2, \kappa_3, \kappa_4$ and they will contribute either GO or DONE to D_i. In the remaining possible successor states $(\kappa_5, \kappa_6, \kappa_7, \kappa_8)$, we have DONE $\in D_i$, and therefore we need to exclude $\{c_{supp}\}$ as a candidate for a set of active reactions B, leaving as the candidates $\{b_{pers}\}$ and $\{b_{pers}, c_{supp}\}$. So $reactsets(W_{i-1}, D_i) = \{\{b_{pers}\}, \{b_{pers}, c_{supp}\}\}$, and $b_{pers} \in \bigcap reactsets(W_{i-1}, D_i) = \{b_{pers}\}$. Hence the implication in formula $(*)$ also holds in Case 2. As a consequence, $b_{pers} \in strongpers(\gamma, D_0)$ □

A consequence of Proposition 3.12 is that the strong persistence obtained there is a *general* property, pointing at *robustness* of the implementation of reaction persistence introduced in Example 3.11. We will formulate it as Theorem 4.12 in the next section.

Example 3.11 presents an implementation of reaction persistence based on two closely coupled reactions. We hypothesise that no implementation exists based on a single reaction unless the behaviour of c_{supp} is embedded in the context sequences.

3.3 Interrupting and suspending reactions

By modifying Example 3.11, we can show the cases of non-persistent reactions (b_{inter} in Example 3.13 and b_{susp} in Example 3.14) and obtain implementations of two important control notions. The first one is *reaction interruption.*

Example 3.13 *Let* $\mathcal{A} = (\{\text{GO}, \text{DONE}, \text{STOP}\}, \{b_{inter}, c_{supp}\})$ *be a reaction system, where:*

$$b_{inter} = (\{\text{GO}\}, \{\text{DONE}, \text{STOP}\}, \{\text{DONE}\})$$

$$c_{supp} = (\{\text{GO}\}, \{\text{DONE}, \text{STOP}\}, \{\text{GO}\}) .$$

Moreover, the context is not allowed to throw in DONE*. In this case, when* STOP *is thrown in by the context, both* b_{inter} *and* c_{supp} *become disabled. Hence,* b_{inter} *is no longer persistent as we can have, for example,* $\{\text{GO}\} \to \{\text{STOP}, \text{GO}\}$ *with* $B = \{c_{supp}\}$ *as the activated reaction set (the only possible set to accom-*

plish this transformation). More precisely, b_{inter} is enabled at $\{\text{GO}\}$ and not at $\{\text{STOP}, \text{GO}\}$, but b_{inter} clearly has not occurred (even weakly) in the transformation $\{\text{GO}\} \to \{\text{STOP}, \text{GO}\}$. From $(\{\text{STOP}\}, \{\text{GO}\})$ only the context can activate the reactions. In this example, the context or b_{inter} can disable reactions.

The second is *reaction suspension.*

Example 3.14 *Let $\mathcal{A} = (\{\text{GO}, \text{DONE}, \text{STOP}\}, \{b_{susp}, c_{supp}\})$ be a reaction system, where:*

$$b_{susp} = (\{\text{GO}\}, \{\text{DONE}, \text{STOP}\}, \{\text{DONE}\})$$
$$c_{supp} = (\{\text{GO}\}, \{\text{DONE}\}, \{\text{GO}\}) .$$

Moreover, the context is not allowed to throw in DONE. *In this case, when* STOP *is thrown in by the context, an enabled reaction b_{susp} is disabled, but not the supporting reaction c_{supp}. And, as long as the environment keeps throwing in* STOP *the reaction b_{susp} is suspended. However, if this is no longer the case, b_{susp} is immediately re-enabled, without help of the context.*

4 GALS reaction systems

The unconstrained treatment of asynchrony in Section 3 might seem to be too 'generous'. However, it was motivated by the fact that one may not know anything *how* the 'management' of asynchrony works. In such a case, we need to work defensively and assume that any subset of reactions may be scheduled for execution at any time. We will now bring into our considerations a 'minimal' information about practical management of asynchrony, by simply restricting sets of reactions which can be activated in a single execution.

Throughout this section we assume that $\mathbb{A} \subseteq 2^A \setminus \{\varnothing\}$ is a fixed nonempty set of nonempty reaction sets covering A (i.e., $\bigcup \mathbb{A} = A$). For $X, Y \subseteq S$, we denote $X \to_{\mathbb{A}} Y$ if there is $B \in \mathbb{A}$ such that $Y = res_B(X)$. Also, for every reaction $b \in A$, $\mathbb{A}_b = \{B \in \mathbb{A} \mid b \in B\}$ is the set of those sets in $\mathbb{A}$ to which b belongs. Note that it may happen that $\mathbb{A}_b = \mathbb{A}_{b'}$, for distinct b and b'.

Intuitively, each set in $\mathbb{A}$ is a 'synchronous island' of reactions. These islands can overlap, which means that we consider here more general model than the standard GALS.
When $\mathbb{A} = 2^A \setminus \{\varnothing\}$, the resulting model is that of unconstrained de-synchronisation discussed in Section 3 (i.e., $X \to_{\mathbb{A}} Y$ if and only if $X \to Y$), and when $\mathbb{A} = \{A\}$, the resulting model is the original synchronous model of reaction systems.

Definition 4.1 *Let $\gamma = C_0 \dots C_n$ $(n \geq 1)$ be a sequence of subsets of S. An*

asynchronous interactive process *in* $\mathcal{A}$ *w.r.t.* γ *and* $\mathbb{A}$ *is a pair* $\pi = (\gamma, \delta)$*, where* $\delta = D_0 \dots D_n$ *is a sequence of subsets of* S *such that, for every* $1 \leq i \leq n$*,* $D_{i-1} \cup C_{i-1} \rightarrow_{\mathbb{A}} D_i$*. We denote this by* $\pi \in asynproc_{\mathbb{A}}(\gamma, D_0)$*.*

It immediately follows that:

Proposition 4.2 $asynproc_{\mathbb{A}'}(\gamma, D_0) \subseteq asynproc_{\mathbb{A}}(\gamma, D_0)$*, for every nonempty set of reaction sets* $\mathbb{A}' \subseteq \mathbb{A}$*.*

Projection simulation by the synchronous model at the level of individual processes is possible by making copies of reactions and introducing triggers corresponding to the reaction sets in $\mathbb{A}$, similarly as in Proposition 3.3.

Proposition 4.3 *Let* $\pi = (C_0 \dots C_n, D_0 \dots D_n) \in asynproc_{\mathbb{A}}(\gamma, D_0)$*. Moreover, let* $A_i \in \mathbb{A}$ *be such that* $D_i = res_{A_i}(D_{i-1} \cup C_{i-1})$*, for every* $1 \leq i \leq n$*. Then* $\pi' = (C'_0 \dots C'_n, D_0 \dots D_n)$ *is an interactive process in the reaction system* $\mathcal{A}' = (S', A')$*, assuming that* TRG_B *is a fresh entity, for each* $B \in \mathbb{A}$*, as well as:*

$$
\begin{aligned}
S' &= S \uplus \{\mathrm{TRG}_B \mid B \in \mathbb{A}\} \\
A' &= \{(R_b \cup \{\mathrm{TRG}_B\}, I_b, P_b) \mid b \in A \wedge B \in \mathbb{A}_b\} \\
C'_i &= C_i \cup \{\mathrm{TRG}_{A_{i+1}}\} \quad \textit{for} \;\; 0 \leq i < n \\
C'_n &= C_n \; .
\end{aligned}
$$

Moreover, $\pi = \pi' \cap S$*.*

Proof Similar to that of Proposition 3.3. □

4.1 Tracing reaction sets

Tracing individual reactions and defining persistence of individual reactions introduced in the previous section works in the case of GALS reaction systems as well. In this and the following sub-section we also lift these notions to sets of reactions. We first concentrate on defining the notion of occurrence of a set of reactions.

In what follows, $reactsets_{\mathbb{A}}(X, Y) = \{B \in \mathbb{A} \mid res_B(X) = Y\}$ is the set of *possibly activated reaction sets*, for $X, Y \subseteq S$ satisfying $X \rightarrow_{\mathbb{A}} Y$.

Definition 4.4 *Let* $X, Y \subseteq S$ *be such that* $X \rightarrow_{\mathbb{A}} Y$*, and* $E \subseteq en(X)$ *be a nonempty set of reactions. Then* E occurred weakly *(*occurred strongly*) in* $X \rightarrow_{\mathbb{A}} Y$ *if there is* $B \in reactsets_{\mathbb{A}}(X, Y)$ *such that* $E \subseteq B$ *(resp.* $E \subseteq \bigcap reactsets_{\mathbb{A}}(X, Y)$*).*
We then denote $E \in weakocc_{\mathbb{A}}(X, Y)$ *(resp.* $E \in strongocc_{\mathbb{A}}(X, Y)$*).*

Proposition 4.5 *$weakocc_{\mathbb{A}'}(X,Y) \subseteq weakocc_{\mathbb{A}}(X,Y)$ and $strongocc_{\mathbb{A}}(X,Y) \subseteq strongocc_{\mathbb{A}'}(X,Y)$, for all $\varnothing \neq \mathbb{A}' \subseteq \mathbb{A}$ and $X \rightarrow_{\mathbb{A}'} Y$.*

Proof Follows directly from Definition 4.4. □

Theorem 4.6 *Let $\mathcal{A} = (S,A)$ be a reaction system and $X,Y \subseteq S$ be such that $X \rightarrow_{\mathbb{A}} Y$. Then $strongocc_{\mathbb{A}}(X,Y) \subseteq weakocc_{\mathbb{A}}(X,Y)$.*

Proof Follows directly from Definition 4.4. □

For $\mathbb{A} = 2^A \setminus \{\varnothing\}$ and $b \in A$, we have $\{b\} \in weakocc_{\mathbb{A}}(X,Y) \iff b \in weakocc(X,Y)$ and $\{b\} \in strongocc_{\mathbb{A}}(X,Y) \iff b \in strongocc(X,Y)$. This, and Example 3.7 means that the inclusion in Theorem 4.6 cannot be reversed.

The technique of fingerprinting introduced in the previous section works in the case of GALS reaction systems as well. We omit the details.

The next result relates the occurrence of a set of reactions $E \subseteq A$ with the occurrences of the reactions it contains.

Theorem 4.7 *Let $\mathcal{A} = (S,A)$ be a reaction system and $X,Y \subseteq S$ be such that $X \rightarrow_{\mathbb{A}} Y$. Then the following hold, for every $E \subseteq A$:*

(1) $E \in strongocc_{\mathbb{A}}(X,Y)$ iff $\{e\} \in strongocc_{\mathbb{A}}(X,Y)$, for every $e \in E$.
(2) $E \in weakocc_{\mathbb{A}}(X,Y)$ implies $\{e\} \in weakocc_{\mathbb{A}}(X,Y)$, for every $e \in E$.

Proof (1) $\{e\} \in strongocc_{\mathbb{A}}(X,Y)$, for every $e \in E$, means that $e \in en(X)$ and $\{e\} \subseteq \bigcap reactsets_{\mathbb{A}}(X,Y)$, for every $e \in E$. This, in turn, is equivalent to $E \subseteq en(X)$ and $E \subseteq \bigcap reactsets_{\mathbb{A}}(X,Y)$. The last formula, according to Definition 4.4, means that $E \in strongocc_{\mathbb{A}}(X,Y)$.

(2) $E \in weakocc_{\mathbb{A}}(X,Y)$ means that $E \subseteq en(X)$ and that there is $B \in reactsets_{\mathbb{A}}(X,Y)$ such that $E \subseteq B$. Then, for every $e \in E$, $e \in en(X)$ and $\{e\} \subseteq B$, and so $\{e\} \in weakocc_{\mathbb{A}}(X,Y)$. □

The following example shows that one needs to require the strong occurrence of all the individual reactions in Theorem 4.7(1).

Example 4.8 *Let $\mathcal{A} = (\{\mathrm{X},\mathrm{Y},\mathrm{Z}\},\{b,c\})$ and $\mathbb{A} = \{A_1,A_2\}$, where $A_1 = \{b\}$, $A_2 = \{b,c\}$, and:*

$$b = (\{\mathrm{X}\},\{\mathrm{Z}\},\{\mathrm{X},\mathrm{Y}\}) \qquad c = (\{\mathrm{X}\},\{\mathrm{Z}\},\{\mathrm{X}\})\ .$$

Then we have $reactsets_{\mathbb{A}}(\{\mathrm{X}\},\{\mathrm{X},\mathrm{Y}\}) = \{\{b\},\{b,c\}\}$. It therefore follows that $\{b\} \in strongocc_{\mathbb{A}}(\{\mathrm{X}\},\{\mathrm{X},\mathrm{Y}\})$, but $\{b,c\},\{c\} \notin strongocc_{\mathbb{A}}(\{\mathrm{X}\},\{\mathrm{X},\mathrm{Y}\})$. Note also that $\{b,c\},\{c\} \in weakocc_{\mathbb{A}}(\{\mathrm{X}\},\{\mathrm{X},\mathrm{Y}\})$.

The implication in Theorem 4.7(2) cannot in general be reversed, as the next example shows. However, it can be reversed in special cases; for example, if $\bigcup Z \in \mathbb{A}$ for every nonempty $Z \subseteq \mathbb{A}$.

Example 4.9 *Let* $\mathcal{A} = (\{\mathrm{X}, \mathrm{Y}\}, \{b, c, d\})$ *and* $\mathbb{A} = \{A_1, A_2\}$, *where* $A_1 = \{b, d\}$, $A_2 = \{c, d\}$, *and:*

$$b = (\{\mathrm{X}\}, \{\mathrm{Y}\}, \{\mathrm{X}\}) \qquad c = (\{\mathrm{X}\}, \{\mathrm{Y}\}, \{\mathrm{Y}\}) \qquad d = (\{\mathrm{X}\}, \{\mathrm{Y}\}, \{\mathrm{X}, \mathrm{Y}\}) \; .$$

Then $\{b\}, \{c\} \in weakocc_{\mathbb{A}}(\{\mathrm{X}\}, \{\mathrm{X}, \mathrm{Y}\})$, *but* $\{b, c\} \notin weakocc_{\mathbb{A}}(\{\mathrm{X}\}, \{\mathrm{X}, \mathrm{Y}\})$.

4.2 Strong persistence of reaction sets

We adapt the previous definitions, concentrating only on the strong persistence of reaction sets.

Definition 4.10 *Let* $X, Y, Z \subseteq S$ *be such that* $X \rightarrow_{\mathbb{A}} Y$, *and* $E \subseteq A$ *be a nonempty set of reactions. Then* E *is* strongly persistent *w.r.t.* $X, Y, Z, \mathbb{A}$ *if* $E \subseteq en(X)$ *and* $E \not\subseteq en(Y \cup Z)$ *implies* $E \in strongocc_{\mathbb{A}}(X, Y)$.
We then denote $E \in strongpers_{\mathbb{A}}(X, Y, Z)$.

We can lift this to the level of whole processes (or computations) with persistence being defined w.r.t. a given context sequence and an initial state of the system.

Definition 4.11 *Let* $\gamma = C_0 \dots C_n$ $(n \geq 1)$ *be a sequence of subsets of* S *and* $D_0 \subseteq S$. *Then a nonempty set of reactions* $E \subseteq A$ *is* strongly persistent *reaction set w.r.t.* γ *and* D_0 *and* $\mathbb{A}$ *if, for every* $\pi = (\gamma, D_0 \dots D_n) \in asynproc_{\mathbb{A}}(\gamma, D_0)$ *and for every* $1 \leq i \leq n$, $E \in strongpers_{\mathbb{A}}(C_{i-1} \cup D_{i-1}, D_i, C_i)$.
We then denote $E \in strongpers_{\mathbb{A}}(\gamma, D_0)$.

We can now formulate a result showing the robustness of implementation of persistent reactions introduced in Example 3.11.

Theorem 4.12 *Let* $\mathcal{A} = (S \uplus \{\mathrm{GO}, \mathrm{DONE}\}, A \uplus \{b_{pers}, c_{supp}\})$ *be a reaction system such that* $\mathrm{DONE} \notin P_d$, *for every* $d \in A$ *and:*

$$b_{pers} = (\{\mathrm{GO}\}, \{\mathrm{DONE}\}, \{\mathrm{DONE}\}) \quad \textit{and} \quad c_{supp} = (\{\mathrm{GO}\}, \{\mathrm{DONE}\}, \{\mathrm{GO}\}) \; .$$

Moreover, let D_0 *be a subset of* $S \cup \{\mathrm{GO}, \mathrm{DONE}\}$, $\gamma = C_0 \dots C_n$ $(n \geq 1)$ *be a sequence of subsets of* $S \cup \{\mathrm{GO}\}$, *and:*

$$\mathbb{A} = \{B \subseteq A \cup \{b_{pers}, c_{supp}\} \mid b_{pers} \in B \vee c_{supp} \in B\} \; .$$

Then $\{b_{pers}\} \in strongpers_{\mathbb{A}}(\gamma, D_0)$.

Proof Follows directly from Proposition 3.12. □

A nonempty set of reactions E is *bundled* if $E \subseteq B$ or $E \cap B = \varnothing$, for every $B \in \mathbb{A}$. As we already mentioned, each set of $\mathbb{A}$ behaves like a 'synchronous island'. Therefore, in the definition of a bundle we want to ensure that at each execution step of an asynchronous interactive (GALS) process all the enabled reactions of a bundle are activated together, or none of them is activated.

In the rest of this section, we show how to implement strong persistence for all bundled sets of reactions.

Example 4.13 *Let* $\mathcal{A} = (\{\text{GO}, \text{GO}', \text{DONE}\}, \{b_{pers}, b'_{pers}, c_{supp}\})$ *be a reaction system, where:*

$$\begin{aligned} b_{pers} &= (\{\text{GO}\}, \{\text{DONE}\}, \{\text{DONE}\}) \\ b'_{pers} &= (\{\text{GO}'\}, \{\text{DONE}\}, \{\text{DONE}\}) \\ c_{supp} &= (\{\text{GO}, \text{GO}'\}, \{\text{DONE}\}, \{\text{GO}, \text{GO}'\}) \,. \end{aligned} \tag{2}$$

Moreover, suppose that $E = \{b_{pers}, b'_{pers}\}$ *is bundled, the context never throws in* DONE*, and* $b_{pers}, b'_{pers} \in B$ *or* $c_{supp} \in B$*, for every* $B \in \mathbb{A}$*. The reaction set* E *is then strongly persistent even though neither* $\{b_{pers}\}$ *nor* $\{b'_{pers}\}$ *is.*

Formally, we have the following result.

Proposition 4.14 *Let* $\mathcal{A}$*,* E *and* $\mathbb{A}$ *be as in Example 4.13. Moreover, suppose that* $D_0 \subseteq \{\text{GO}, \text{GO}', \text{DONE}\}$*, and* $\gamma = C_0 \dots C_n$ *(*$n \geq 1$*) is a sequence of subsets of* $\{\text{GO}, \text{GO}'\}$*. Then* $\{b_{pers}, b'_{pers}\} \in strongpers_{\mathbb{A}}(\gamma, D_0)$*. Moreover, in general, neither* $\{b_{pers}\} \in strongpers_{\mathbb{A}}(\gamma, D_0)$ *nor* $\{b'_{pers}\} \in strongpers_{\mathbb{A}}(\gamma, D_0)$ *holds.*

Proof The proof showing that the reaction set $E = \{b_{pers}, b'_{pers}\}$ is strongly persistent is similar to that of Proposition 3.12.

To show the second part of the proposition we now show that there is a process $\pi \in asynproc_{\mathbb{A}}(\gamma, D_0)$ such that $\{b_{pers}\} \notin strongpers_{\mathbb{A}}(\gamma, D_0)$ (for $\{b'_{pers}\}$ the proof is symmetric).
Let us take $\pi = (C_0C_1, D_0D_1)$, where $C_0 = \{\text{GO}\}$ and $D_0 = C_1 = D_1 = \varnothing$. We observe that $\pi \in asynproc_{\mathbb{A}}(C_0C_1, D_0)$. Indeed, we have $C_0 \cup D_0 \to_{\mathbb{A}} D_1$ as $res_{\{c_{supp}\}}(\{\text{GO}\}) = \varnothing$ and $\{c_{supp}\} \in \mathbb{A}$.
In this case, $\{b_{pers}\} \subseteq en(C_0 \cup D_0)$ and $\{b_{pers}\} \not\subseteq en(C_1 \cup D_1)$. On the other hand, $\{b_{pers}\} \notin strongocc_{\mathbb{A}}(C_0 \cup D_0, D_1)$, and so $\{b_{pers}\} \notin strongpers_{\mathbb{A}}(C_0C_1, D_0)$. □

A slight modification of Example 4.13 ensures that not only $\{b_{pers}, b'_{pers}\}$ is persistent, but also $\{b_{pers}\}$. In a way, the next example indicates that persistence can be introduced in a modular way.

Example 4.15 *Let* $\mathcal{A} = (\{\text{GO}, \text{GO}', \text{DONE}\}, \{b_{pers}, b'_{pers}, c_{supp}, d_{supp}\})$ *be a reaction system, where everything is as in Example 4.13 except an additional reaction* $d_{supp} = (\{\text{GO}\}, \{\text{DONE}\}, \{\text{GO}\})$ *and an additional assumption that* $b_{pers} \in B$ *or* $d_{supp} \in B$*, for every* $B \in \mathbb{A}$*. Then both* E *and* $\{b_{pers}\}$ *are strongly persistent, but* $\{b'_{pers}\}$ *is not.*

The implementation of reaction set persistence introduced in Example 4.13 and discussed later assumed a two-element reaction set. This, however, can be easily generalised to any number of reactions, in the following way.

Example 4.16 *Let* $n \geq 1$ *and* $\mathcal{A} = (S, A)$ *be a reaction system, where:*

$$\begin{aligned} S &= \{\text{GO}_1, \ldots, \text{GO}_n, \text{DONE}\} \\ A &= \{b^1_{pers}, \ldots, b^n_{pers}, c_{supp}\} \\ b^i_{pers} &= (\{\text{GO}_i\}, \{\text{DONE}\}, \{\text{DONE}\}) \qquad (1 \leq i \leq n) \\ c_{supp} &= (\{\text{GO}_1, \ldots, \text{GO}_n\}, \{\text{DONE}\}, \{\text{GO}_1, \ldots, \text{GO}_n\}) \; . \end{aligned}$$

Moreover, the set $E = \{b^1_{pers}, \ldots, b^n_{pers}\}$ *is bundled, the context never throws in* DONE*, and* $E \subseteq B$ *or* $\{c_{supp}\} \subseteq B$*, for every* $B \in \mathbb{A}$*. The reaction set* E *is then strongly persistent.*

The robustness of the implementation of strong persistence introduced in Example 4.16 is confirmed by the following result.

Theorem 4.17 *Let* $\mathcal{A} = (S \uplus \{\text{GO}_1, \ldots, \text{GO}_n, \text{DONE}\}, A \uplus \{b^1_{pers}, \ldots, b^n_{pers}, c_{supp}\})$ *be a reaction system such that* $b^1_{pers}, \ldots, b^n_{pers}, c_{supp}$ *are as in Example 4.16. Moreover, suppose that:*

(1) $E = \{b^1_{pers}, \ldots, b^n_{pers}\}$ *is bundled;*
(2) $\text{DONE} \notin P_d$*, for every* $d \in A$*;*
(3) $E \subseteq B$ *or* $c_{supp} \in B$*, for every* $B \in \mathbb{A}$*;*
(4) D_0 *is a subset of* $S \cup \{\text{GO}_1, \ldots, \text{GO}_n, \text{DONE}\}$*; and*
(5) $\gamma = C_0 \ldots C_n$ *(*$n \geq 1$*) is a sequence of subsets of* $S \cup \{\text{GO}_1, \ldots, \text{GO}_n\}$*.*

Then $E \in strongpers_{\mathbb{A}}(\gamma, D_0)$*.*

Proof Follows from a straightforward generalisation of the proof of Proposition 3.12. □

5 Concluding remarks

In this paper, we propose an approach for the de-synchronisation of the execution of reaction systems. At every step of the execution, we allow a subset of the enabled reactions to be activated instead of the full set of enabled reactions, as in the standard synchronous model of reaction systems. This gives rise to the novel notion of an asynchronous interactive process. In the context of such processes, we introduce mechanisms to capture the occurrence of single reactions and sets of reactions. This, in turn, forms the basis for defining persistence in the context of reaction systems — the property linked to many desirable features of real-world dynamic systems.

The present paper introduced new fundamental concepts and solutions into the realm of reaction systems. They have a potential of driving the development of further notions and results of far reaching applicability. In what follows, we outline exemplars of ideas based on the technical content of this paper, and so adding to the points made in the general discussion contained in the introduction.

In Section 3.1, just after Definition 3.5 which introduced the notions of weak and strong occurrence, it was mentioned that one might consider the notion of p-occurrence to capture the likelihood of a reaction being executed when moving between two system states. This idea can be taken much further in order to support *quantitative* analysis of notions relying on traceability of reactions. For example, in the setup of Definition 4.4 one could assume that each potential set of activated reactions $B \in \mathbb{A}$ comes with a known or estimated non-negative (or even dynamically changing) probability weight $w(B)$. Then one might define the probability of occurrence of E as:

$$p_E = \frac{\sum_{B \in reactsets_{\mathbb{A}}(X,Y) \wedge E \subseteq B} w(B)}{\sum_{B \in reactsets_{\mathbb{A}}(X,Y)} w(B)} \ .$$

Note that the strong occurrence would then correspond to $p_E = 1$. We envisage that the above can give rise to a fully fledged treatment of probability in reaction systems (note that a similar notion of probability cannot be defined for the synchronous reactions systems).

The concrete reaction systems implementing concepts of persistence, interruption and suspension introduced in Examples 3.11, 3.13, 3.14, and 4.16 should be seen as *patterns* of achieving the corresponding behavioural properties. Looking at the first example, if a persistent reaction b_{pers} was supposed to be derived from a reaction $b = (\{\mathrm{X}\}, \varnothing, \{\mathrm{Y}\})$, then one would have

$$b_{pers} = (\{\mathrm{X}, \mathrm{GO}\}, \{\mathrm{DONE}\}, \{\mathrm{Y}, \mathrm{DONE}\}) \quad \textit{and} \quad c_{supp} = (\{\mathrm{X}, \mathrm{GO}\}, \{\mathrm{DONE}\}, \{\mathrm{X}, \mathrm{GO}\}) \ .$$

And, clearly, the scheme would have worked if the context would never throw in DONE.

This leads to a general point that in the above examples the behaviour of the context can be seen as linked to the concept of *local variable*. More precisely, what really matters in Example 3.11 is that only an occurrence of b_{pers} can generate DONE, and so 'reset' the corresponding 'local variable'. For this to work, it suffices to ensure syntactically that DONE can only be generated by b_{pers} (intuitively, this corresponds to treating some of the molecules as representations of the values of Boolean variables). We envisage that the above discussion will lead to a fully fledged development and embedding of programming language concepts in the domain of reaction systems.

Persistence is not necessarily a global property. For example, the environment may be triggering different activities with different periods, and only then we need to verify that the system (or its part) is persistent. It would therefore be interesting to pursue the analysis of periodicity of persistence in reaction systems. Note that the results of such a study would also be relevant in the synchronous case.

In this paper, context sequences were considered individually. In future, we plan to change this by introducing a way of specifying context sequences through suitably modified context controllers [35]. In particular, the aim would be to indicate which reactions can go ahead and which not, and in this way controlling asynchrony (and so introducing asynchrony management). Moreover, such a control could be exercised by taking into account the current state of the reaction system. The intuition behind the first option is that context sequences and activation of reactions are generated purely externally, without considering the current state of a reaction system.

For example, an asynchronous controller might be introduced as a labelled directed graph $\mathcal{E} = (Q, V)$ such that Q is a finite nonempty set of states and $V \subseteq Q \times (2^S \times (2^A \setminus \{\varnothing\})) \times Q$ is a set of labelled arcs specifying the context sets to be thrown in as well as the sets of reactions to be activated. Following [35], one would also require a non-blocking nature of the controller, by assuming that for every state q, there is at least one labelled arc of the form $(q, (C, B), q')$. Such an asynchronous controller could then be put to work in tandem with reaction system $\mathcal{A} = (S, A)$.

In particular, an interactive process in $\mathcal{A}$ controlled by $\mathcal{E}$ and initiated at $(q, X) \in Q \times 2^S$ could be defined as a pair $\pi = (C_0, \ldots, C_n, D_0, \ldots, D_n)$ of finite sequences of sets of molecules for which one could find a path in $\mathcal{E}$ of the form:

$$(q, (C_0, B_0), q_0), (q_0, (C_1, B_1), q_1), \ldots, (q_{n-1}, (C_n, B_n), q_n)$$

so that $D_0 = res_{B_0}(X)$ and $D_i = res_{B_i}(C_{i-1} \cup D_{i-1})$, for $i = 1, \ldots, n$.

Crucially, an introduction of asynchronous controllers would advance the formalism presented in this paper (where context sequences can come from an arbitrary set) to more a formalism amenable to the mechanisms of automated property verification; for example, by suitably adapting techniques from [16,30].

A further step would be to allow asynchronous controllers to 'inspect' the current state of the reaction systems generalising the notion of state-aware context controllers of [35].

Acknowledgments

We are grateful to the anonymous reviewers for their careful reading and suggestions which have helped us to improve the presentation and to clarify our ideas.

The research in this paper was partially supported by the Polish National Agency for Academic Exchange under Grant No. PPI/APM/2018/1/00036/U/001 and by the UK EPSRC Grant No. EP/N031768/1 "Event-based parallel computing - partially ordered event-triggered systems (POETS)".

References

[1] R. Milner, Turing, computing and communication, in: D. Goldin, S. Smolka, P. Wegner (Eds.), Interactive Computation: The New Paradigm, Springer Berlin Heidelberg, 2006, pp. 1–8.

[2] A. Ehrenfeucht, I. Petre, G. Rozenberg, Reaction systems: a model of computation inspired by the functioning of the living cell, techreport 1161, Turku Centre for Computer Science (2016).

[3] A. Ehrenfeucht, G. Rozenberg, Reaction systems, Fundam. Inform. 75 (1-4) (2007) 263–280.

[4] A. Ehrenfeucht, J. Kleijn, M. Koutny, G. Rozenberg, Reaction systems: A natural computing approach to the functioning of living cells, in: A Computable Universe, World Scientific, 2012, pp. 189–208.

[5] A. Ehrenfeucht, I. Petre, G. Rozenberg, Reaction systems: A model of computation inspired by the functioning of the living cell, in: The Role of Theory in Computer Science - Essays Dedicated to Janusz Brzozowski, World Scientific, 2017, pp. 1–32.

[6] A. Ehrenfeucht, J. Kleijn, M. Koutny, G. Rozenberg, Qualitative and Quantitative Aspects of a Model for Processes Inspired by the Functioning of the Living Cell, John Wiley & Sons, Ltd, 2012, Ch. 16, pp. 303–321.

[7] L. Corolli, C. Maj, F. Marini, D. Besozzi, G. Mauri, An excursion in reaction systems: From computer science to biology, Theor. Comput. Sci. 454 (2012) 95–108.

[8] I. Petre, A. Mizera, C. L. Hyder, A. Meinander, A. Mikhailov, R. I. Morimoto, L. Sistonen, J. E. Eriksson, R. Back, A simple mass-action model for the eukaryotic heat shock response and its mathematical validation, Natural Computing 10 (1) (2011) 595–612.

[9] S. Azimi, B. Iancu, I. Petre, Reaction system models for the heat shock response, Fundam. Inform. 131 (3-4) (2014) 299–312.

[10] R. Barbuti, P. Bove, R. Gori, F. Levi, P. Milazzo, Simulating gene regulatory networks using reaction systems, Vol. 2240 of CEUR Workshop Proceedings, 2018.

[11] P. Bottoni, A. Labella, G. Rozenberg, Reaction systems with influence on environment, Journal of Membrane Computing 1 (1) (2019) 3–19.

[12] P. Bottoni, A. Labella, G. Rozenberg, Networks of reaction systems, Int. J. Found. Comput. Sci. to appear.

[13] R. Barbuti, R. Gori, F. Levi, P. Milazzo, Generalized contexts for reaction systems: definition and study of dynamic causalities, Acta Inf. 55 (3) (2018) 227–267.

[14] M. Hirvensalo, On probabilistic and quantum reaction systems, Theor. Comput. Sci. 429 (2012) 134–143.

[15] A. Salomaa, Functions and sequences generated by reaction systems, Theor. Comput. Sci. 466 (2012) 87–96.

[16] A. Meski, W. Penczek, G. Rozenberg, Model checking temporal properties of reaction systems, Inf. Sci. 313 (2015) 22–42.

[17] A. Ehrenfeucht, M. G. Main, G. Rozenberg, A. T. Brown, Stability and chaos in reaction systems, Int. J. Found. Comput. Sci. 23 (5) (2012) 1173.

[18] A. Salomaa, Functional constructions between reaction systems and propositional logic, Int. J. Found. Comput. Sci. 24 (1) (2013) 147–160.

[19] E. Formenti, L. Manzoni, A. E. Porreca, Fixed points and attractors of reaction systems, Vol. 8493 of Lecture Notes in Computer Science, Springer, 2014, pp. 194–203.

[20] A. Ehrenfeucht, G. Rozenberg, Events and modules in reaction systems, Theor. Comput. Sci. 376 (1-2) (2007) 3–16.

[21] E. Formenti, L. Manzoni, A. E. Porreca, On the complexity of occurrence and convergence problems in reaction systems, Natural Computing 14 (1) (2015) 185–191.

[22] A. Ehrenfeucht, G. Rozenberg, Zoom structures and reaction systems yield exploration systems, Int. J. Found. Comput. Sci. 25 (3) (2014) 275–306.

[23] A. Ehrenfeucht, J. Kleijn, M. Koutny, G. Rozenberg, Minimal reaction systems, Trans. Computational Systems Biology 14 (2012) 102–122.

[24] A. Ehrenfeucht, J. Kleijn, M. Koutny, G. Rozenberg, Evolving reaction systems, Theor. Comput. Sci. 682 (2017) 79–99.

[25] A. Yakovlev, P. Vivet, M. Renaudin, Advances in asynchronous logic: from principles to GALS & noc, recent industry applications, and commercial CAD tools, in: E. Macii (Ed.), Design, Automation and Test in Europe, DATE 13, Grenoble, France, March 18-22, 2013, EDA Consortium San Jose, CA, USA / ACM DL, 2013, pp. 1715–1724.

[26] J. Cortadella, A. Kondratyev, L. Lavagno, C. P. Sotiriou, Desynchronization: Synthesis of asynchronous circuits from synchronous specifications, IEEE Trans. on CAD of Integrated Circuits and Systems 25 (10) (2006) 1904–1921.

[27] A. Yakovlev, M. Kishinevsky, A. Kondratyev, L. Lavagno, M. Pietkiewicz-Koutny, On the models for asynchronous circuit behaviour with OR causality, Formal Methods in System Design 9 (3) (1996) 189–233.

[28] L. H. Landweber, E. L. Robertson, Properties of conflict-free and persistent petri nets, J. ACM 25 (3) (1978) 352Ű–364.

[29] J. Fernandes, M. Koutny, L. Mikulski, M. Pietkiewicz-Koutny, D. Sokolov, A. Yakovlev, Persistent and nonviolent steps and the design of GALS systems, Fundam. Inform. 137 (1) (2015) 143–170.

[30] A. Meski, M. Koutny, W. Penczek, Verification of linear-time temporal properties for reaction systems with discrete concentrations, Fundam. Inform. 154 (1-4) (2017) 289–306.

[31] A. Meski, M. Koutny, W. Penczek, Reaction mining for reaction systems, Vol. 10867 of Lecture Notes in Computer Science, Springer, 2018, pp. 131–144.

[32] P. Darondeau, M. Koutny, M. Pietkiewicz-Koutny, A. Yakovlev, Synthesis of nets with step firing policies, Fundam. Inform. 94 (3-4) (2009) 275–303.

[33] J. W. Bryans, M. Koutny, L. Mazaré, P. Y. A. Ryan, Opacity generalised to transition systems, Int. J. Inf. Sec. 7 (6) (2008) 421–435.

[34] R. Barbuti, R. Gori, F. Levi, P. Milazzo, Specialized predictor for reaction systems with context properties, Fundam. Inform. 147 (2-3) (2016) 173–191.

[35] J. Kleijn, M. Koutny, L. Mikulski, G. Rozenberg, Reaction systems, transition systems, and equivalences, Vol. 11011 of Lecture Notes in Computer Science, Springer, 2018, pp. 63–84.