Towards Bridging Time and Causal Reversibility

Marco Bernardo and Claudio Antares Mezzina(✉)

Dipartimento di Scienze Pure e Applicate, Università di Urbino, Urbino, Italy
claudio.mezzina@uniurb.it

Abstract. Causal consistent reversibility blends causality and reversibility. For a concurrent system, it says that an action can be undone provided this has no consequences, thereby making it possible to bring the system back to a past consistent state. Time reversibility is considered instead in the performance evaluation field. A continuous-time Markov chain is time reversible if its behavior remains the same when the direction of time is reversed. We try to bridge these two theories by showing the conditions under which both causal consistent reversibility and time reversibility can be achieved in the setting of a stochastic process algebra.

1 Introduction

The interest into computation reversibility dates back to the 60's, when it was observed that irreversible computations cause heat dissipation into circuits [16]. This suggested that low energy consumption could be achieved by resorting to *reversible computing*, in which there is no information loss [3]. Nowadays, reversible computing has several applications ranging from biochemical reactions [29,30] and parallel discrete-event simulation [27,32] to robotics [22], control theory [33], fault tolerant systems [6,17,35,36], and program debugging [7,20].

In a reversible system, we can observe two directions of computation: a *forward* one, coinciding with the normal way of computing, and a *backward* one, which is able to undo the effects of the forward one. In the literature, there exist different meanings of reversibility. For instance, in a Petri net model reversibility means that one can always reach the initial marking [2], while in distributed systems it amounts to the capability of returning to a past consistent state [5]. In contrast, in the performance evaluation field, reversibility is intended as time reversibility and is instrumental to develop efficient analysis methods [13].

Our focus is on the relationship between *causal consistent reversibility* and *time reversibility*, from a process algebraic perspective. On the one hand, quantitative aspects have been disregarded in the setting of causal consistent reversibility. On the other hand, the theory of time reversibility has been applied to concurrent systems without explicitly taking causality into account.

In this paper, instead, we aim at bridging these two theories, by showing how causal consistent reversibility and time reversibility can be jointly obtained.

Published by Springer Nature Switzerland AG 2020
A. Gotsman and A. Sokolova (Eds.): FORTE 2020, LNCS 12136, pp. 22–38, 2020.
https://doi.org/10.1007/978-3-030-50086-3_2

To this purpose, we consider a stochastic process calculus in which every action is equipped with a positive real number expressing the rate at which the action is executed. As is well known in the literature [10], the stochastic process underlying the calculus turns out to be a continuous-time Markov chain (CTMC) [14].

The contribution of this paper is twofold. Firstly, we apply for the first time the technique of [28] to a stochastic process calculus. In particular, we provide forward and backward operational semantic rules – featuring forward and backward actions and rates – and then we show that the resulting calculus is causal consistent reversible. This is accomplished by importing from the reduction semantics setting of [5] the notion of concurrent transitions, which is new in the structural operational semantics framework of [28].

Secondly, after observing that the CTMC underlying the calculus is stationary, we show that time reversibility can be achieved by using, in the operational semantic rules, backward rates equal to the corresponding forward rates. This is quite different from the approaches followed for example in [8,25], where time reversibility is addressed a posteriori, as we instead obtain a calculus in which time reversibility can be guaranteed by construction.

This paper is organized as follows. In Sects. 2 and 3 we recall background information about causal consistent reversibility and time reversibility, respectively. Then, in Sect. 4 we provide and illustrate our results about the integration of these two forms of reversibility in the considered stochastic process calculus. Finally, in Sect. 5 we conclude with some directions for future work.

2 Causal Consistent Reversibility

Reversibility in a system means the possibility of reverting the last performed action. In a sequential system, this is very simple as there exists just one last action. In a concurrent system, the situation is more complex as there is no clear definition of last action. Indeed, there might be several concurrent last actions. One could resort to timestamps to decide which action is the last one, but having synchronised clocks in a distributed system is rather difficult.

A good approximation is to consider as last action each action that has not caused any other action yet. This is at the basis of the so called *causal consistent reversibility* [5], which relates reversibility with causality. Intuitively, the definition says that, in a concurrent system, any action can be undone provided that all of its consequences, if any, are undone beforehand.

In the process algebra literature, there are two reversible variants of CCS [26]. The first one in time order, RCCS [5], uses stack-based memories attached to processes to record all the actions executed by the processes themselves. In contrast, [28] proposes a general method, of which CCSK is a result, to reverse calculi whose operational semantic rules are expressed in the path format [1]. The basic idea of this method is to make all the operators of the calculus static and to univocally identify each executed action with a communication key. Note that, since dynamic operators such as prefixing and choice are forgetful by definition, making them static avoids information loss during a reduction.

Despite these two approaches may seem different, they have been shown to be equivalent in terms of labeled transition system isomorphism [18]. The approach of [5] is more suitable for systems whose operational semantics is given in terms of reduction semantics, hence its application is to be preferred in the case of very expressive calculi [4,19] as well as programming languages [21,24]. On the other hand, the approach of [28] is very handy when it comes to deal with labeled transition systems and CCS-like calculi, which is the case of this paper.

For example, given the process $P + Q$, from $P \xrightarrow{\alpha} P'$ we derive $P + Q \xrightarrow{\alpha} P'$. If we assume the possibility of reverting action α, i.e., $P' \xrightarrow{\alpha_r} P$, we have that P' gets back to a state in which the information about the choice operator and Q is lost. To avoid this, in [28] $+Q$ is treated as a dead decoration of process P'. In this way, the use of explicit memories of [5] is avoided because the necessary information is syntactically maintained within processes.

3 Time Reversibility

In the field of performance evaluation, a different notion of reversibility, called *time reversibility*, is considered. We illustrate it in the specific case of continuous-time Markov chains, which are discrete-state stochastic processes characterized by the *memoryless property* [14].

A *stochastic process* describes the evolution of some random phenomenon over time through a set of random variables, one for each time instant. A stochastic process $X(t)$ taking values into a discrete state space $\mathcal{S}$ for $t \in \mathbb{R}_{\geq 0}$ is a *continuous-time Markov chain (CTMC)* iff for all $n \in \mathbb{N}$, time instants $t_0 < t_1 < \cdots < t_n < t_{n+1} \in \mathbb{R}_{\geq 0}$, and states $s_0, s_1, \ldots, s_n, s_{n+1} \in \mathcal{S}$ it holds that $\Pr\{X(t_{n+1}) = s_{n+1} \mid X(t_i) = s_i, 0 \leq i \leq n\} = \Pr\{X(t_{n+1}) = s_{n+1} \mid X(t_n) = s_n\}$, i.e., the probability of moving from one state to another does not depend on the particular path that has been followed in the past to reach the current state, hence that path can be forgotten.

A CTMC is *irreducible* iff each of its states can be reached from every other state. A state $s \in \mathcal{S}$ is *recurrent* iff the CTMC will eventually return to s with probability 1, in which case s is called *positive recurrent* iff the expected number of steps until the CTMC returns to it is finite. A CTMC is *ergodic* iff it is irreducible and all of its states are positive recurrent; ergodicity coincides with irreducibility in the case that the CTMC has finitely many states.

A CTMC can be represented as a labeled transition system or as a state-indexed matrix. In the first case, each transition is labeled with some probabilistic information describing the evolution from its source state to its target state. In the second case, the same information is stored into an entry, indexed by those two states, of a matrix. The value of this probabilistic information is, in general, a function of the time at which the state change takes place.

For the sake of simplicity, we restrict ourselves to *time-homogeneous* CTMCs, in which conditional probabilities of the form $\Pr\{X(t + t') = s' \mid X(t) = s\}$ do not depend on t, so that the considered information is simply a positive real number. This is called the *rate* at which the CTMC moves from state s to

state s' and uniquely characterizes the exponentially distributed time taken by the considered move. It can be shown that the sojourn time in any state $s \in \mathcal{S}$ is exponentially distributed with rate given by the sum of the rates of the moves of s. The average sojourn time in s is the inverse of such a sum and the probability of moving from s to s' is proportional to the corresponding rate.

Every time-homogeneous, ergodic CTMC $X(t)$ is *stationary*, which means that $(X(t_i + t'))_{1 \leq i \leq n}$ has the same joint distribution as $(X(t_i))_{1 \leq i \leq n}$ for all $n \in \mathbb{N}_{\geq 1}$ and $t_1 < \cdots < t_n, t' \in \mathbb{R}_{\geq 0}$. Specifically, $X(t)$ has a unique *steady-state probability distribution* π that for all $s \in \mathcal{S}$ fulfills $\pi(s) = \lim_{t \to \infty} \Pr\{X(t) = s \mid X(0) = s'\}$ for any $s' \in \mathcal{S}$. These probabilities can be computed by solving the linear system of *global balance equations* $\pi \cdot \mathbf{Q} = \mathbf{0}$ subject to $\sum_{s \in \mathcal{S}} \pi(s) = 1$ and $\pi(s) \in \mathbb{R}_{>0}$ for all $s \in \mathcal{S}$. The *infinitesimal generator matrix* $\mathbf{Q}$ contains for each pair of distinct states the rate of the corresponding move, which is 0 in the absence of a direct move between them, with $q_{s,s} = -\sum_{s' \neq s} q_{s,s'}$ for all $s \in \mathcal{S}$ so that every row of $\mathbf{Q}$ sums up to 0.

Due to state space explosion and numerical stability problems [34], the calculation of the solution of the global balance equation system is not always feasible. However, it can be tackled in the case that the behavior of the considered CTMC remains the same when the direction of time is reversed. A CTMC $X(t)$ is *time reversible* iff $(X(t_i))_{1 \leq i \leq n}$ has the same joint distribution as $(X(t' - t_i))_{1 \leq i \leq n}$ for all $n \in \mathbb{N}_{\geq 1}$ and $t_1 < \cdots < t_n, t' \in \mathbb{R}_{\geq 0}$, in which case $X(t)$ and its reversed version $X^{\mathrm{r}}(t) = X(t' - t)$ are stochastically identical; in particular, $X(t)$ and $X^{\mathrm{r}}(t)$ share the same steady-state probability distribution π if any. In order for a stationary CTMC $X(t)$ to be time reversible, it is necessary and sufficient that the *partial balance equations* $\pi(s) \cdot q_{s,s'} = \pi(s') \cdot q_{s',s}$ are satisfied for all $s, s' \in \mathcal{S}$ such that $s \neq s'$ or, equivalently, that $q_{s_1,s_2} \cdot \ldots \cdot q_{s_{n-1},s_n} \cdot q_{s_n,s_1} = q_{s_1,s_n} \cdot q_{s_n,s_{n-1}} \cdot \ldots \cdot q_{s_2,s_1}$ for all $n \in \mathbb{N}_{\geq 2}$ and distinct $s_1, \ldots, s_n \in \mathcal{S}$ [13].

Time reversibility of CTMC-based compositional models of concurrent systems has been investigated in [8]. More precisely, conditions relying on the conservation of total exit rates of states and of rate products around cycles are examined, which support the hierarchical and compositional reversal of stochastic process algebra terms. These conditions also lead to the efficient calculation of steady-state probability distributions in a product form typical of queueing theory [15], thus avoiding the need of solving the global balance equation system. More recently, in [25] similar conditions have been employed to characterize the class of ρ-reversible stochastic automata. Under certain constraints, the joint steady-state probability distribution of the composition of two such automata is the product of the steady-state probability distributions of the two automata.

4 Integrating Causal and Time Reversibility

In this section, we integrate the two concepts of causal consistent reversibility and time reversibility recalled in the previous two sections. To do so, we start with a simple calculus called RMPC – Reversible Markovian Process Calculus, in which actions are paired with rates, whose syntax and semantics are inspired by the

approach of [28]. Then, we show that the reversibility induced by RMPC is causal consistent by importing the notion of concurrent transitions from [5]. Finally, we exhibit the conditions under which time reversibility is achieved too and we compare our setting, in which time reversibility is ensured by construction, with those of [8,25].

4.1 Syntax and Semantics for RMPC

The syntax of RMPC is shown in Table 1. A *forward process* P can be one of the following: the idle process $\mathbf{0}$; the prefixed process $(a, \lambda).P$, which is able to perform an action a at rate λ and then continues as process P; the nondeterministic choice $P + Q$ between processes P and Q; or the cooperation $P \parallel_L Q$, indicating that processes P and Q execute in parallel and must synchronise only on actions prescribed by the set L.

A *reversible process* R is built on top of forward processes. As in [28], the syntax of reversible processes differs from the one of forward processes by the fact that in the former each prefix (a, λ) can be decorated with a *communication key* i thus becoming $(a, \lambda)[i]$. A process of the form $(a, \lambda)[i].R$ expresses that in the past the process synchronised with the environment and this synchronisation was identified by key i. Keys are thus attached only to already executed actions.

Table 1. Syntax of RMPC forward/standard/initial processes and reversible processes

$$\begin{array}{l} P, Q \ ::= \mathbf{0} \mid (a, \lambda).P \mid P + Q \mid P \parallel_L Q \\ R, S \ ::= \ P \mid (a, \lambda)[i].R \mid R + S \mid R \parallel_L S \end{array}$$

Let $\mathcal{A}$ be the set of actions such that $a, b \in \mathcal{A}$, $\mathcal{R} = \mathbb{R}_{>0}$ be the set of rates such that $\lambda, \mu \in \mathcal{R}$, and $\mathcal{K}$ be the set of keys such that $i, j \in \mathcal{K}$. Let $\mathcal{L} = \mathcal{A} \times \mathcal{R} \times \mathcal{K}$ be the set of labels each formed by an action, a rate, and a communication key. We let ℓ and its decorated versions range over $\mathcal{L}$. Moreover, given a forward label $\ell = (a, \lambda)[i]$, we denote by $\overline{\ell} = (a, \overline{\lambda})[i]$ the corresponding backward label. Finally, $\mathcal{P}$ is the set of processes generated by the production for R in Table 1.

Definition 1 (standard process). *Process $R \in \mathcal{P}$ is standard, written $\mathtt{std}(R)$, iff it can be derived from the production for P in Table 1.*

Definition 2 (process key). *The set of keys of process $R \in \mathcal{P}$, written $\mathtt{key}(R)$, is inductively defined as follows:*

$$\begin{array}{ll} \mathtt{key}(P) = \emptyset & \mathtt{key}((a, \lambda)[i].R) = \{i\} \cup \mathtt{key}(R) \\ \mathtt{key}(R + S) = \mathtt{key}(R) \cup \mathtt{key}(S) & \mathtt{key}(R \parallel_L S) = \mathtt{key}(R) \cup \mathtt{key}(S) \end{array}$$

The semantics for RMPC is defined as a labeled transition system $(\mathcal{P}, \mathcal{L}, \mapsto)$. Like in [28], the transition relation $\mapsto \subseteq \mathcal{P} \times \mathcal{L} \times \mathcal{P}$ is given by $\rightarrow \cup \rightsquigarrow$, where

Table 2. Structural operational semantic rules for RMPC

$$\textsc{Act1}\ \frac{\texttt{std}(R)}{(a,\lambda).R \xrightarrow{(a,\lambda)[i]} (a,\lambda)[i].R} \qquad \textsc{Act1}^\bullet\ \frac{\texttt{std}(R)}{(a,\lambda)[i].R \overset{(a,\overline{\lambda})[i]}{\rightsquigarrow} (a,\lambda).R}$$

$$\textsc{Act2}\ \frac{R \xrightarrow{(b,\mu)[j]} R' \quad j \neq i}{(a,\lambda)[i].R \xrightarrow{(b,\mu)[j]} (a,\lambda)[i].R'} \qquad \textsc{Act2}^\bullet\ \frac{R \overset{(b,\overline{\mu})[j]}{\rightsquigarrow} R' \quad j \neq i}{(a,\lambda)[i].R \overset{(b,\overline{\mu})[j]}{\rightsquigarrow} (a,\lambda)[i].R'}$$

$$\textsc{Cho}\ \frac{R \xrightarrow{(a,\lambda)[i]} R' \qquad \texttt{std}(S)}{R + S \xrightarrow{(a,\lambda)[i]} R' + S} \qquad \textsc{Cho}^\bullet\ \frac{R \overset{(a,\overline{\lambda})[i]}{\rightsquigarrow} R' \qquad \texttt{std}(S)}{R + S \overset{(a,\overline{\lambda})[i]}{\rightsquigarrow} R' + S}$$

$$\textsc{Par}\ \frac{R \xrightarrow{(a,\lambda)[i]} R' \quad a \notin L \quad i \notin \texttt{key}(S)}{R \parallel_L S \xrightarrow{(a,\lambda)[i]} R' \parallel_L S} \qquad \textsc{Par}^\bullet\ \frac{R \overset{(a,\overline{\lambda})[i]}{\rightsquigarrow} R' \quad a \notin L \quad i \notin \texttt{key}(S)}{R \parallel_L S \overset{(a,\overline{\lambda})[i]}{\rightsquigarrow} R' \parallel_L S}$$

$$\textsc{Coo}\ \frac{R \xrightarrow{(a,\lambda)[i]} R' \quad S \xrightarrow{(a,\mu)[i]} S' \quad a \in L}{R \parallel_L S \xrightarrow{(a,\lambda\cdot\mu)[i]} R' \parallel_L S'} \qquad \textsc{Coo}^\bullet\ \frac{R \overset{(a,\overline{\lambda})[i]}{\rightsquigarrow} R' \quad S \overset{(a,\overline{\mu})[i]}{\rightsquigarrow} S' \quad a \in L}{R \parallel_L S \overset{(a,\overline{\lambda}\cdot\overline{\mu})[i]}{\rightsquigarrow} R' \parallel_L S'}$$

the *forward transition relation* $\longrightarrow$ and the *backward transition relation* $\rightsquigarrow$ are the least relations respectively induced by the forward rules in the left part and the backward rules in the right part of Table 2.

Rule Act1 deals with prefixed processes of the form $(a,\lambda).P$, with P written as R subject to $\texttt{std}(R)$. In addition to transforming the action prefix into a transition label, it generates a fresh key i, which is bound to the action (a,λ) thus yielding the label $(a,\lambda)[i]$. As we can note, the prefix is not discarded by the application of the rule, instead it becomes a key-storing decoration in the target process. Rule Act1$^\bullet$ reverts the action $(a,\lambda)[i]$ of the process $(a,\lambda)[i].R$ provided that R is a standard process, which ensures that $(a,\lambda)[i]$ is the only past action that is left to undo. One of the main design choices of the entire framework is how the rate $\overline{\lambda}$ of the backward action is calculated. For the time being, we leave it unspecified in Act1$^\bullet$ as the value of this rate is not necessary to prove the causal consistency part of reversibility, but as we will see later on it is important in the proof of time reversibility.

The presence of rules Act2 and Act2$^\bullet$ is motivated by the fact that rule Act1 does not discard the executed prefix from the process it generates. In particular, rule Act2 allows a prefixed process $(a,\lambda)[i].R$ to execute if R can itself execute, provided that the action performed by R picks a key j different from i. Rule Act2$^\bullet$ simply propagates the execution of backward actions from inner subprocesses that are not standard as long as key uniqueness is preserved.

Unlike the classical rules of the choice operator [26], rule Cho does not discard the context, i.e., the part of the process that has not contributed to the action. More in detail, if the process $R + S$ does an action, say $(a,\lambda)[i]$, and

becomes R', then the entire process becomes $R' + S$. In this way, the information about $+\,S$ is preserved. Furthermore, since S is a standard process because of the premise $\mathtt{std}(S)$, it will never execute even if it is present in the process $R' + S$. Hence, the $+\,S$ can be seen as a decoration, or a dead context, of process R. Note that, in order to apply rule Cho, at least one of the two processes has to be standard, meaning that it is impossible for two non-standard processes to execute if they are composed by a choice operator. Rule Cho$^\bullet$ has precisely the same structure as rule Cho, but uses the backward transition relation. For both rules, we omit their symmetric variants in which it is S to move.

The semantics of cooperation is inspired by [11]. Rule Par allows process R within $R \parallel_L S$ to individually perform an action $(a, \lambda)[i]$, provided that $a \notin L$. Rule Coo allows R and S to cooperate through any action in the set L, provided that the communication key is the same on both sides. For the sake of simplicity, the rate of the cooperation action is assumed to be the product of the rates of the two involved actions [9]. Rules Par$^\bullet$ and Coo$^\bullet$ respectively have the same structure as Par and Coo; the symmetric variants of Par and Par$^\bullet$ are omitted.

Not all the processes generated by the grammar in Table 1 are meaningful as, e.g., there might be several action prefixes sharing the same key in a sequential process, i.e., a process without occurrences of the cooperation operator. We only consider processes that are initial or reachable in the following sense:

Definition 3 (initial and reachable process). *Process $R \in \mathcal{P}$ is initial iff $\mathtt{std}(R)$ holds. Process $R \in \mathcal{P}$ is reachable iff it is initial or can be derived from an initial one via finitely many applications of the rules for $\longrightarrow$ in Table 2.*

4.2 Properties Preliminary to Reversibility

A basic property to satisfy in order for RMPC to be reversible is the so called loop lemma [5,28], which will be exploited to establish both causal consistent reversibility and time reversibility. This property states that each transition of a reachable process can be undone. Formally:

Lemma 1 (loop lemma). *Let $R \in \mathcal{P}$ be a reachable process. Then $R \xrightarrow{(a,\lambda)[i]} S$ iff $S \overset{(a,\overline{\lambda})[i]}{\rightsquigarrow} R$.*

Proof. We proceed by induction on the depth of the derivation of $R \xrightarrow{(a,\lambda)[i]} S$ (resp., $S \overset{(a,\overline{\lambda})[i]}{\rightsquigarrow} R$), by noticing that for each forward (resp., backward) rule there exists a corresponding backward (resp., forward) one. ∎

The lemma generalizes as follows. For any sequence σ of $n \in \mathbb{N}_{>0}$ labels $\ell_1, \ldots, \ell_n$, let $R \xrightarrow{\sigma} S$ be the corresponding *forward* sequence of transitions $R \xrightarrow{\ell_1} R_1 \xrightarrow{\ell_2} \cdots \xrightarrow{\ell_n} S$ and $\overline{\sigma}$ be the corresponding *backward* sequence such that, for each ℓ_i occurring in σ, it holds that $R_{i-1} \xrightarrow{\ell_i} R_i$ iff $R_i \overset{\overline{\ell_i}}{\rightsquigarrow} R_{i-1}$. A direct consequence of the loop lemma is the following:

Corollary 1. *Let $R \in \mathcal{P}$ be a reachable process. Then $R \overset{\sigma}{\longmapsto} S$ iff $S \overset{\overline{\sigma}}{\longmapsto} R$.*

4.3 Causal Consistent Reversibility for RMPC

In order to prove the causal consistent reversibility of RMPC, we borrow some machinery from [5] that needs to be adapted as the reversing method of [28] we are using is different from the one of [5]. In particular, we import the notion of concurrent transitions.

Given a transition $\theta : R \overset{\ell}{\mapsto} S$ with $R, S \in \mathcal{P}$ reachable processes, we call R the *source* of θ and S its *target*. Two transitions are said to be *coinitial* if they have the same source, and *cofinal* if they have the same target. A sequence of pairwise composable transitions is called a *computation*, where *composable* means that the target of any transition in the sequence is the source of the next transition. We let θ and its decorated variants range over transitions, while ω and its decorated variants range over computations. If θ is a forward transition, i.e., $\theta : R \overset{\ell}{\rightarrow} S$, we denote its backward version $S \overset{\overline{\ell}}{\rightarrow} R$ as $\overline{\theta}$. The notions of source, target, and composability extend naturally to computations. We indicate with ϵ_R the empty computation whose source is R and with $\omega_1; \omega_2$ the composition of two composable computations ω_1 and ω_2.

Before specifying when two transitions are concurrent, we need to define the set of causes – identified by keys – of a given communication key.

Definition 4 (causal set). *Let $R \in \mathcal{P}$ be a reachable process and $i \in \mathtt{key}(R)$. The causal set $\mathtt{cau}(R, i)$ is inductively defined as follows for $j \neq i$:*

$$\begin{aligned}
&\mathtt{cau}((a, \lambda)[i].R, i) = \emptyset \\
&\mathtt{cau}((a, \lambda)[j].R, i) = \{j\} \cup \mathtt{cau}(R, i) \\
&\mathtt{cau}(R + S, i) = \mathtt{cau}'(R, i) \cup \mathtt{cau}'(S, i) \\
&\mathtt{cau}(R \parallel_L S, i) = \mathtt{cau}'(R, i) \cup \mathtt{cau}'(S, i)
\end{aligned}$$

where $\mathtt{cau}'(R, i) = \mathtt{cau}(R, i)$ if $i \in \mathtt{key}(R)$ and $\mathtt{cau}'(R, i) = \emptyset$ otherwise.

If $i \in \mathtt{key}(R)$, then $\mathtt{cau}(R, i)$ represents the set of keys in R that caused i, with $\mathtt{cau}(R, i) \subset \mathtt{key}(R)$ since $i \notin \mathtt{cau}(R, i)$ and keys not causally related to i are not considered. A key j causes i if it appears syntactically before i in R or, said otherwise, i is inside the scope of j. We are now in place to define what we mean by concurrent transitions:

Definition 5 (concurrent transitions). *Two coinitial transitions θ_1 and θ_2 from a reachable process $R \in \mathcal{P}$ are in conflict iff one of the following holds:*

1. $\theta_1 : R \xrightarrow{(a,\lambda)[j]} S_1$ *and* $\theta_2 : R \overset{(b,\overline{\mu})[i]}{\rightsquigarrow} S_2$ *with* $i \in \mathtt{cau}(S_1, j)$.
2. $R = R_1 + R_2$ *with* θ_k *deriving from* $R_k \xrightarrow{(a_k,\lambda_k)[i_k]} S_k$ *for* $k = 1, 2$.

Two coinitial transitions are concurrent when they are not in conflict.

The first condition above just tells that a forward transition is in conflict with a backward one whenever the latter tries to undo a cause of the key of the

former. The second condition deems as conflictual two transitions respectively generated by the two subprocesses of a choice operator. Figure 1 shows two related examples. In the first case, the process $(a, \lambda)[i].(b, \mu).P$ can perform two transitions: a backward one and a forward one. They meet the first condition of Definition 5 as the backward transition removes the key i that is in the causal set of j. In the second case, we have that process $(a, \lambda).P + (a, \lambda).P$ is able to trigger two forward transitions. Since they arise from the same choice operator, they are in conflict according to the second condition of Definition 5.

Remark 1. It is worth noting that in a stochastic process calculus like RMPC the semantic treatment of the choice operator is problematic [10] because a process of the form $(a, \lambda).P + (a, \lambda).P$ should produce either a single a-transition whose rate is $\lambda + \lambda$, or two a-transitions each having rate λ that do not collapse into a single one. In our reversible framework, two distinct transitions are generated thanks to the fact that the key associated with the executed action is stored into the derivative process too, as shown in the bottom part of Fig. 1.

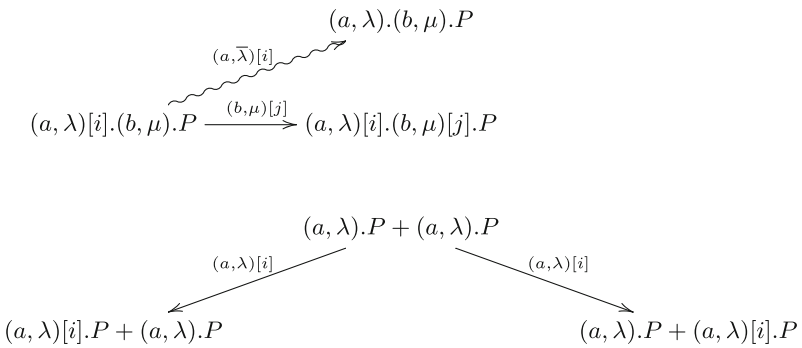

Fig. 1. Examples of conflicting transitions

Concurrent transitions can commute, while conflicting ones cannot. Formally:

Lemma 2 (diamond lemma). *Let $\theta_1 : R \overset{\ell_1}{\longmapsto} S_1$ and $\theta_2 : R \overset{\ell_2}{\longmapsto} S_2$ be two coinitial transitions from a reachable process $R \in \mathcal{P}$. If θ_1 and θ_2 are concurrent, then there exist two cofinal transitions $\theta_2/\theta_1 : S_1 \overset{\ell_2}{\longmapsto} S$ and $\theta_1/\theta_2 : S_2 \overset{\ell_1}{\longmapsto} S$.*

Proof. By case analysis on the form of θ_1 and θ_2. ∎

We are now in a position to show that reversibility in our framework is causally consistent. Following [23], we first define a notion of *causal equivalence* between computations that abstracts from the order of concurrent transitions. We formalize $\asymp$ as the least equivalence relation over computations that is closed under composition and obeys the following rules:

$$\theta_1; \theta_2/\theta_1 \asymp \theta_2; \theta_1/\theta_2 \qquad \theta; \overline{\theta} \asymp \epsilon_{\mathtt{source}(\theta)} \qquad \overline{\theta}; \theta \asymp \epsilon_{\mathtt{target}(\overline{\theta})}$$

Equivalence $\asymp$ states that if we have two concurrent transitions, then the two computations obtained by swapping the order of their execution are the same, and that any computation composed by a transition followed by its inverse is equivalent to the empty computation. The proof of causal consistency relying on $\asymp$ follows that of [5], although the arguments are different due to the fact that the notion of concurrent transitions is formalized differently.

The following lemma says that, up to causal equivalence, one can always reach for the maximum freedom of choice among transitions, meaning that it is possible to undo all the executed actions and then restart.

Lemma 3 (rearranging lemma). *For any computation ω there exist two forward computations ω_1 and ω_2 such that $\omega \asymp \overline{\omega_1};\omega_2$.*

Proof. By induction on the length of ω and on the distance (intended as number of transitions) between the beginning of ω and the earliest pair of consecutive transitions in ω such that the former is forward while the latter is backward. The analysis uses both the loop lemma (Lemma 1) and the diamond lemma (Lemma 2). ■

The following lemma says that if two computations ω_1 and ω_2 are coinitial and cofinal and ω_2 is made of forward transitions only, then in ω_1 there are some transitions that are later undone. This computation is thus causally equivalent to a forward one in which the undone transitions do not take place at all.

Lemma 4 (shortening lemma). *Let ω_1 and ω_2 be coinitial and cofinal computations, with ω_2 forward. Then there exists a forward computation ω_1' of length at most that of ω_1 such that $\omega_1' \asymp \omega_1$.*

Proof. The proof is by induction on the length of ω_1, using the diamond lemma (Lemma 2) and the rearranging lemma (Lemma 3). In the proof, the forward computation ω_2 is the main guideline for shortening ω_1 into a forward computation. Indeed, the proof relies crucially on the fact that ω_1 and ω_2 share the same source and the same target and that ω_2 is a forward computation. ■

Theorem 1 (causal consistency). *Let ω_1 and ω_2 be coinitial computations. Then $\omega_1 \asymp \omega_2$ iff ω_1 and ω_2 are cofinal too.*

Proof. The 'if' direction follows by definition of causal equivalence and computation composition. The 'only if' direction exploits the diamond lemma (Lemma 2), the rearranging lemma (Lemma 3), and the shortening lemma (Lemma 4). ■

With Theorem 1 we have proved that the notion of causal equivalence characterises a space for admissible rollbacks that are (i) consistent in the sense that they do not lead to previously unreachable states and (ii) flexible enough to allow undo operations to be rearranged. This implies that the states reached by a backward computation could be reached by performing forward computations only. We can therefore conclude that RMPC is causal consistent reversible.

4.4 Time Reversibility for RMPC

The rules in Table 2 associate with any initial process $R \in \mathcal{P}$ a labeled transition system $[\![R]\!] = (\mathcal{P}, \mathcal{L}, \mapsto, R)$. To investigate time reversibility, we have to derive from $[\![R]\!]$ the CTMC $\mathcal{M}[\![R]\!]$ underlying R and we have to specify how each backward rate $\overline{\lambda}$ is obtained from the corresponding forward rate λ.

First of all, we observe that every non-terminal state of $[\![R]\!]$ has infinitely many outgoing transitions. The reason is that rules Act1 and Act2 generate a transition for each possible admissible key, with the key being part of both the label and the derivative process term. On the one hand, this is useful for avoiding the generation of a single (a, λ)-transition in the case of a process like $(a, \lambda).P + (a, \lambda).P$ whose overall exit rate is $\lambda + \lambda$; even if the key is the same, two different states $(a, \lambda)[i].P + (a, \lambda).P$ and $(a, \lambda).P + (a, \lambda)[i].P$ are reached. On the other hand, it requires considering transition bundles to build $\mathcal{M}[\![R]\!]$.

We call *transition bundle* a set of transitions departing from the same state and labeled with the same action/rate but different keys, whose target states are syntactically identical up to keys. Formally, we denote by $\equiv_{\mathcal{K}}$ the least equivalence relation over $\mathcal{P}$ induced by $(a, \lambda)[i].S \equiv_{\mathcal{K}} (a, \lambda)[j].S$. We then define the CTMC underlying an initial process $R \in \mathcal{P}$ as the labeled transition system $\mathcal{M}[\![R]\!] = (\mathcal{P}/\equiv_{\mathcal{K}}, \mathcal{A} \times \mathcal{R}, \mapsto_{\mathcal{K}}, [R]_{\equiv_{\mathcal{K}}})$ where:

- $\mathcal{P}/\equiv_{\mathcal{K}}$ is the quotient set of $\equiv_{\mathcal{K}}$ over $\mathcal{P}$, i.e., the set of classes of processes that are equivalent to each other according to $\equiv_{\mathcal{K}}$.
- $[R]_{\equiv_{\mathcal{K}}}$ is the equivalence class of R with respect to $\equiv_{\mathcal{K}}$, which simply is the singleton set $\{R\}$ as R is initial and hence contains no keys.
- $\mapsto_{\mathcal{K}} \subseteq \mathcal{P}/\equiv_{\mathcal{K}} \times (\mathcal{A} \times \mathcal{R}) \times \mathcal{P}/\equiv_{\mathcal{K}}$ is the transition relation given by $\rightarrow_{\mathcal{K}} \cup \rightsquigarrow_{\mathcal{K}}$ such that $[R]_{\equiv_{\mathcal{K}}} \xrightarrow{(a,\lambda)}_{\mathcal{K}} [R']_{\equiv_{\mathcal{K}}}$ whenever $R \xrightarrow{(a,\lambda)[i]} R'$ for some $i \in \mathcal{K}$ and $[R]_{\equiv_{\mathcal{K}}} \overset{(a,\overline{\lambda})}{\rightsquigarrow}_{\mathcal{K}} [R']_{\equiv_{\mathcal{K}}}$ whenever $R \overset{(a,\overline{\lambda})[i]}{\rightsquigarrow} R'$ for some $i \in \mathcal{K}$.

When moving from $[\![R]\!]$ to $\mathcal{M}[\![R]\!]$, individual states are replaced by classes of states that are syntactically identical up to keys in the same positions, moreover keys are removed from transition labels. As a consequence, every state of $\mathcal{M}[\![R]\!]$ turns out to have finitely many outgoing transitions. We also note that $\mathcal{M}[\![R]\!]$ is an *action-labeled* CTMC, as each of its transitions is labeled with both a rate and an action.

As a preliminary step towards time reversibility, we have to show that $\mathcal{M}[\![R]\!]$ is stationary. It holds that $\mathcal{M}[\![R]\!]$ is even ergodic thanks to the loop lemma.

Lemma 5. *Let $R \in \mathcal{P}$ be an initial process. Then $\mathcal{M}[\![R]\!]$ is time homogeneous and ergodic.*

Proof. The time homogeneity of $\mathcal{M}[\![R]\!]$ is a straightforward consequence of the fact that its rates simply are positive real numbers, not time-dependent functions. The ergodicity of $\mathcal{M}[\![R]\!]$ stems from the fact that the graph representation of the considered CTMC is a finite, strongly connected component due to Corollary 1. ■

We exploit once more the loop lemma to derive that, in the case that $\overline{\lambda} = \lambda$, the steady-state probability distribution of $\mathcal{M}[\![R]\!]$ is the uniform distribution, from which time reversibility will immediately follow.

Lemma 6. *Let $R \in \mathcal{P}$ be an initial process, $\mathcal{S}$ be the set of states of $\mathcal{M}[\![R]\!]$, and $n = |\mathcal{S}|$. If every backward rate is equal to the corresponding forward rate, then the steady-state probability distribution π of $\mathcal{M}[\![R]\!]$ satisfies $\pi(s) = 1/n$ for all $s \in \mathcal{S}$.*

Proof. If $n = 1$, i.e., *R is equal to $\mathbf{0}$ or to the cooperation of several processes whose initial actions have to synchronize but are different from each other, then trivially $\pi(s) = 1/n = 1$ for the only state $s \in \mathcal{S}$.*
Suppose now that $n \geq 2$. From Lemma 5, it follows that $\mathcal{M}[\![R]\!]$ has a unique steady-state probability distribution π. Due to Lemma 1, the global balance equation for an arbitrary $s \in \mathcal{S}$ is as follows:

$$\pi(s) \cdot \sum_{s \xrightarrow{(a,\lambda)}_{\mathcal{K}} s'} \lambda \;=\; \sum_{s' \xrightarrow{(a,\overline{\lambda})}_{\mathcal{K}} s} \pi(s') \cdot \overline{\lambda}$$

Since every backward rate is equal to the corresponding forward rate, the global balance equation for s boils down to:

$$\pi(s) \cdot \sum_{s \xrightarrow{(a,\lambda)}_{\mathcal{K}} s'} \lambda \;=\; \sum_{s' \xrightarrow{(a,\lambda)}_{\mathcal{K}} s} \pi(s') \cdot \lambda$$

Since the two summations have the same number of summands, the equation above is satified when $\pi(s) = \pi(s')$ for all $s' \in \mathcal{S}$ reached by a transition from s. All global balance equations are thus satisfied when $\pi(s) = 1/n$ for all *$s \in \mathcal{S}$.* ■

Theorem 2 (time reversibility). *Let $R \in \mathcal{P}$ be an initial process. If every backward rate is equal to the corresponding forward rate, then $\mathcal{M}[\![R]\!]$ is time reversible.*

Proof. Let $\mathcal{S}$ be the set of states of $\mathcal{M}[\![R]\!]$ and $n = |\mathcal{S}|$. From Lemma 5, it follows that $\mathcal{M}[\![R]\!]$ has a unique steady-state probability distribution π. To avoid trivial cases, suppose $n \geq 2$ and consider $s, s' \in \mathcal{S}$ with $s \neq s'$ connected by a transition. Due to Lemma 1, the partial balance equation for s and s' is as follows:

$$\pi(s) \cdot \sum_{s \xrightarrow{(a,\lambda)}_{\mathcal{K}} s'} \lambda \;=\; \pi(s') \cdot \sum_{s' \xrightarrow{(a,\overline{\lambda})}_{\mathcal{K}} s} \overline{\lambda}$$

Since every backward rate is equal to the corresponding forward rate, the partial balance equation for s and s' boils down to:

$$\pi(s) \cdot \sum_{s \xrightarrow{(a,\lambda)}_{\mathcal{K}} s'} \lambda \;=\; \pi(s') \cdot \sum_{s' \xrightarrow{(a,\lambda)}_{\mathcal{K}} s} \lambda$$

Since the two summations have the same number of summands and $\pi(s) = \pi(s') = 1/n$ *due to Lemma 6, the equation above is satified. The result then follows from the fact that* s *and* s' *are two arbitrary distinct states connected by a transition.* ■

The main difference between our approach to time reversibility and the ones of [8,25] is twofold. Firstly, our approach is part of a more general framework in which also causal consistent reversibility is addressed. Secondly, our approach is inspired by the idea of [28] of developing a formalism in which it is possible to express models whose reversibility is guaranteed by construction, instead of building a posteriori the time-reversed version of a certain model like in [8] or verifying a posteriori whether a given model is time reversible or not like in [25].

It is worth noting that these methodological differences do not prevent us from adapting to our setting some results from [8,25], although a few preliminary observations about notational differences are necessary.

Both [8] and [25] make a distinction between active actions, each of which is given a rate, and passive actions, each of which is given a weight, with the constraint that, in case of synchronization, the rate of the active action is multiplied by the weight of the corresponding passive action. In RMPC there is no such distinction, however the same operation, i.e., multiplication, is applied to the rates of two synchronizing actions. A passive action can thus be rendered as an action with rate 1, while a set of alternative passive actions can be rendered as a set of actions whose rates sum up to 1. Moreover, in [25] synchronization is enforced between any active-passive pair of identical actions, whereas in RMPC the cooperation operator is enriched with an explicit synchronization set L, which yields as a special case the aforementioned synchronization discipline when L is equal to the set $\mathcal{A}$ of all the actions. We can therefore conclude that our cooperation operator is a generalization of those used in [8,25], hence the recalled notational differences do not hamper result transferral.

In [8] the compositionality of a CTMC-based stochastic process calculus is exploited to prove the reversed compound agent theorem (RCAT), which establishes the conditions under which the time-reversed version of the cooperation of two processes is equal to the cooperation of the time-reversed versions of those two processes. The application of RCAT leads to product-form solution results from stochastic process algebraic models, including a new different proof of Jackson theorem for product-form queueing networks [12].

In [25] the notion of ρ-reversibility is introduced for stochastic automata, which are essentially action-labeled CTMCs. Function ρ is a state permutation that ensures (i) for each action the equality of the total exit rate of any state s and $\rho(s)$ and (ii) the conservation of action-related rate products across cycles when considering states in the forward direction and their ρ-counterparts in the backward direction. For any ergodic ρ-reversible automaton, it turns out that $\pi(s) = \pi(\rho(s))$ for every state s. Moreover, the synchronization inspired by [31] of two ρ-reversible stochastic automata is still ρ-reversible and, in case of ergodicity, under certain conditions the steady-state probability of any compound state is the product of the steady-state probabilities of the two constituent states.

Our time reversibility result for RMPC can be rephrased in the setting of [25] in terms of ρ-reversibility with ρ being the identity function over states. As a consequence, the following two results stem from Theorem 2 of the present paper and, respectively, Theorems 2 and 3 of [25]:

Corollary 2 (time reversibility closure). *Let $R, S \in \mathcal{P}$ be initial processes and $L \subseteq \mathcal{A}$. If every backward rate is equal to the corresponding forward rate, then $\mathcal{M}[\![R \parallel_L S]\!]$ is time reversible too.*

Corollary 3 (product form). *Let $R, S \in \mathcal{P}$ be initial processes and $L \subseteq \mathcal{A}$. If every backward rate is equal to the corresponding forward rate and the set of states $\mathcal{S}$ of $\mathcal{M}[\![R \parallel_L S]\!]$ is equal to $\mathcal{S}_R \times \mathcal{S}_S$ where $\mathcal{S}_R$ is the set of states of $\mathcal{M}[\![R]\!]$ and $\mathcal{S}_S$ is the set of states of $\mathcal{M}[\![S]\!]$, then $\pi(r, s) = \pi_R(r) \cdot \pi_S(s)$ for all $(r, s) \in \mathcal{S}_R \times \mathcal{S}_S$.*

The product form result above avoids the calculation of the global balance equations over $\mathcal{M}[\![R \parallel_L S]\!]$, as $\pi(r, s)$ can simply be obtained by multiplying $\pi_R(r)$ with $\pi_S(s)$. However, the condition $\mathcal{S} = \mathcal{S}_R \times \mathcal{S}_S$ requires to check that every state in the full Cartesian product is reachable from $R \parallel_L S$. This means that no compound state is such that its constituent states enable some action, but none of the enabled actions can be executed due to the constraints imposed by the synchronization set L. The condition $\mathcal{S} = \mathcal{S}_R \times \mathcal{S}_S$ implies that $\mathcal{M}[\![R \parallel_L S]\!]$ is ergodic over the full Cartesian product of the two original state spaces, which is the condition used in [25]. Although implicit in the statement of the corollary, the time reversibility of $\mathcal{M}[\![R \parallel_L S]\!]$ is essential for the product form result.

5 Conclusions

Different interpretations of reversibility are present in the literature. In this paper, we have started our research quest towards bridging causal consistent reversibility [5] – developed in concurrency theory – and time reversibility [13] – originated in the field of stochastic processes. We have accomplished this by introducing the stochastic process calculus RMPC, whose syntax and semantics follow the approach of [28], thus paving the way to concurrent system models that are both causal consistent reversible and time reversible by construction. Based on time reversibility, we have also adapted from [25] a product form result that enables the efficient calculation of performance measures.

There are several lines of research that we plan to undergo, ranging from the application of our results to examples and case studies modeled with RMPC to the development of further theoretical results. For instance, we would like to investigate other conditions under which time reversibility is achieved, in addition to the one relying on the equality of forward and backward rates.

Moreover, we observe that the syntax of RMPC does not include recursion. From the point of view of the ergodicity of the underlying CTMC, this is not a problem because every forward transition has the corresponding backward transition by construction. However, there might be situations in which recursion

is necessary to appropriately describe the behavior of a system. Because of the use of communication keys, a simple process of the form $P \triangleq (a, \lambda).P$, whose standard labeled transition system features a single state with a self-looping transition, produces a sequence of infinitely many distinct states even if we resort to transition bundles. Our claim is that the specific cooperation operator that we have considered may require a mechanism lighter than communications keys to keep track of past actions, which may avoid the generation of an infinite state space in the presence of recursion.

Acknowledgement. We would like to thank Andrea Marin for the valuable discussions on time reversibility. The second author has been partially supported by the French ANR project *DCore* ANR-18-CE25-0007 and by the Italian INdAM – GNCS project 2020 *Reversible Concurrent Systems: From Models to Languages*.

References

1. Baeten, J.C.M., Verhoef, C.: A congruence theorem for structured operational semantics with predicates. In: Best, E. (ed.) CONCUR 1993. LNCS, vol. 715, pp. 477–492. Springer, Heidelberg (1993). https://doi.org/10.1007/3-540-57208-2_33
2. Barylska, K., Koutny, M., Mikulski, L., Piatkowski, M.: Reversible computation vs. reversibility in Petri nets. Sci. Comput. Program. **151**, 48–60 (2018)
3. Bennett, C.H.: Logical reversibility of computations. IBM J. Res. Dev. **17**, 525–532 (1973)
4. Cristescu, I., Krivine, J., Varacca, D.: A compositional semantics for the reversible π-calculus. In: Proceedings of LICS 2013, pp. 388–397. IEEE-CS Press (2013)
5. Danos, V., Krivine, J.: Reversible communicating systems. In: Gardner, P., Yoshida, N. (eds.) CONCUR 2004. LNCS, vol. 3170, pp. 292–307. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-28644-8_19
6. Danos, V., Krivine, J.: Transactions in RCCS. In: Abadi, M., de Alfaro, L. (eds.) CONCUR 2005. LNCS, vol. 3653, pp. 398–412. Springer, Heidelberg (2005). https://doi.org/10.1007/11539452_31
7. Giachino, E., Lanese, I., Mezzina, C.A.: Causal-consistent reversible debugging. In: Gnesi, S., Rensink, A. (eds.) FASE 2014. LNCS, vol. 8411, pp. 370–384. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54804-8_26
8. Harrison, P.G.: Turning back time in Markovian process algebra. Theoret. Comput. Sci. **290**(3), 1947–1986 (2003)
9. Hillston, J.: The nature of synchronisation. In: Proceedings of PAPM 1994, pp. 51–70. University of Erlangen, Technical Report 27–4 (1994)
10. Hillston, J.: A Compositional Approach to Performance Modelling. Cambridge University Press, Cambridge (1996)
11. Hoare, C.A.R.: Communicating Sequential Processes. Prentice Hall, Upper Saddle River (1985)
12. Jackson, J.R.: Jobshop-like queueing systems. Manage. Sci. **10**(1), 131–142 (1963)
13. Kelly, F.P.: Reversibility and Stochastic Networks. Wiley, Chichester (1979)
14. Kemeny, J.G., Snell, J.L.: Finite Markov Chains. Van Nostrand, New York (1960)
15. Kleinrock, L.: Queueing Systems. Wiley, New York (1975)
16. Landauer, R.: Irreversibility and heat generated in the computing process. IBM J. Res. Dev. **5**, 183–191 (1961)

17. Lanese, I., Lienhardt, M., Mezzina, C.A., Schmitt, A., Stefani, J.-B.: Concurrent flexible reversibility. In: Felleisen, M., Gardner, P. (eds.) ESOP 2013. LNCS, vol. 7792, pp. 370–390. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-37036-6_21
18. Lanese, I., Medić, D., Mezzina, C.A.: Static versus dynamic reversibility in CCS. Acta Informatica (2019)
19. Lanese, I., Mezzina, C.A., Stefani, J.-B.: Reversing higher-order π. In: Gastin, P., Laroussinie, F. (eds.) CONCUR 2010. LNCS, vol. 6269, pp. 478–493. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15375-4_33
20. Lanese, I., Nishida, N., Palacios, A., Vidal, G.: CauDEr: a causal-consistent reversible debugger for erlang. In: Gallagher, J.P., Sulzmann, M. (eds.) FLOPS 2018. LNCS, vol. 10818, pp. 247–263. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-90686-7_16
21. Lanese, I., Nishida, N., Palacios, A., Vidal, G.: A theory of reversibility for Erlang. J. Log. Algeb. Meth. Program. **100**, 71–97 (2018)
22. Laursen, J.S., Ellekilde, L.P., Schultz, U.P.: Modelling reversible execution of robotic assembly. Robotica **36**(5), 625–654 (2018)
23. Lévy, J.J.: An algebraic interpretation of the $\lambda\beta$K-calculus; and an application of a labelled λ-calculus. Theoret. Comput. Sci. **2**(1), 97–114 (1976)
24. Lienhardt, M., Lanese, I., Mezzina, C.A., Stefani, J.-B.: A reversible abstract machine and its space overhead. In: Giese, H., Rosu, G. (eds.) FMOODS/FORTE -2012. LNCS, vol. 7273, pp. 1–17. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-30793-5_1
25. Marin, A., Rossi, S.: Quantitative analysis of concurrent reversible computations. In: Sankaranarayanan, S., Vicario, E. (eds.) FORMATS 2015. LNCS, vol. 9268, pp. 206–221. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-22975-1_14
26. Milner, R.: Communication and Concurrency. Prentice Hall, Upper Saddle River (1989)
27. Perumalla, K.S., Park, A.J.: Reverse computation for rollback-based fault tolerance in large parallel systems - evaluating the potential gains and systems effects. Cluster Comput. **17**(2), 303–313 (2014)
28. Phillips, I.C.C., Ulidowski, I.: Reversing algebraic process calculi. J. Logic Algeb. Program. **73**(1–2), 70–96 (2007)
29. Phillips, I., Ulidowski, I., Yuen, S.: A reversible process calculus and the modelling of the ERK signalling pathway. In: Glück, R., Yokoyama, T. (eds.) RC 2012. LNCS, vol. 7581, pp. 218–232. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36315-3_18
30. Michele Pinna, G.: Reversing steps in membrane systems computations. In: Gheorghe, M., Rozenberg, G., Salomaa, A., Zandron, C. (eds.) CMC 2017. LNCS, vol. 10725, pp. 245–261. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-73359-3_16
31. Plateau, B.: On the stochastic structure of parallelism and synchronization models for distributed algorithms. In: Proceedings of SIGMETRICS 1985, pp. 147–154. ACM Press (1985)
32. Schordan, M., Oppelstrup, T., Jefferson, D.R., Barnes Jr., P.D.: Generation of reversible C++ code for optimistic parallel discrete event simulation. New Gener. Comput. **36**(3), 257–280 (2018)
33. Siljak, H., Psara, K., Philippou, A.: Distributed antenna selection for massive MIMO using reversing Petri nets. IEEE Wirel. Commun. Lett. **8**(5), 1427–1430 (2019)

34. Stewart, W.J.: Introduction to the Numerical Solution of Markov Chains. Princeton University Press, Princeton (1994)
35. Vassor, M., Stefani, J.-B.: Checkpoint/rollback vs causally-consistent reversibility. In: Kari, J., Ulidowski, I. (eds.) RC 2018. LNCS, vol. 11106, pp. 286–303. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-99498-7_20
36. de Vries, E., Koutavas, V., Hennessy, M.: Communicating transactions. In: Gastin, P., Laroussinie, F. (eds.) CONCUR 2010. LNCS, vol. 6269, pp. 569–583. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15375-4_39