Causal Reversibility in Nondeterministic Process Calculi Extended with Time or Probabilities

Marco Bernardo[a], Claudio A. Mezzina[a], Andrea Esposito[a]

[a] *Dipartimento di Scienze Pure e Applicate, Università di Urbino, Italy*

Abstract

In addition to forward computations, a reversible system also features backward computations along which the effects of forward ones can be undone. This is accomplished by reverting executed actions starting from the last one. Since the last performed action may not be uniquely identifiable in a concurrent setting, Danos and Krivine proposed causal reversibility: an executed action can be undone provided that all of its consequences have been undone already. Phillips and Ulidowski then showed how to define nondeterministic process calculi that meet causal reversibility by construction. Lanese, Phillips, and Ulidowski subsequently classified the basic properties that ensure causal reversibility. In this paper we investigate the extent to which those techniques apply to reversible nondeterministic process calculi that include quantitative aspects. Firstly, we consider the introduction of time described via numeric delays with action execution separated from time passing like in the calculus of Moller and Tofts, where actions can be lazy or eager and time is subject to time determinism and time additivity. Secondly, we address the introduction of probabilities like in the calculus of Hansson and Jonsson, in which action execution and probabilistic choices alternate. We show that both resulting reversible calculi satisfy causal reversibility provided that suitable variants of the aforementioned techniques are developed to guarantee the proper forward and backward interplay of nondeterminism and quantitative aspects. The use of the former calculus is illustrated on a timeout mechanism, whereas the use of the latter is exemplified on quantum teleportation.

1. Introduction

In the 1960's it was observed that irreversible computations cause heat dissipation into circuits because any logically irreversible manipulation of information, such as bit erasure or computation path merging, must be accompanied by a corresponding entropy increase [42, 4]. Therefore, any logically reversible computation, in which no information is canceled, may be potentially carried out without releasing further heat, as later verified in [12] and given a physical foundation in [30]. This suggested that low energy consumption could be achieved by resorting to *reversible computing*, in which there is no information loss because computation can go not only forward, but also backward by undoing the effects of the forward direction when needed. Nowadays, reversible computing has several applications such as biochemical reaction modeling [56, 57], parallel discrete-event simulation [53, 62], fault-tolerant systems [23, 69, 43, 68], robotics [47], wireless communications [66], program debugging [33, 45], and distributed algorithms [71, 8].

Reversibility in a computing system has to do with the possibility of reverting executed actions starting from the last one. In a sequential computing system this is very simple because at every step only one action is executed, hence the only challenge is how to store the information needed to reverse each executed action. As far as concurrent and distributed systems are concerned, a critical aspect of reversibility is that there may not be a total order over executed actions, hence the last performed action may not be uniquely identifiable. This led to the introduction of the notion of *causal reversibility* [22], according to which a previously executed action can be undone provided that all of its consequences, if any, have been undone beforehand. It is worth noting that the concept of causality is used in place of the concept of time to decide whether an action can be undone or not.

In a pure nondeterministic process algebraic setting, two distinct approaches have been developed to deal with causal reversibility. The *dynamic* approach of [22, 41], which is adequate for very expressive calculi and programming languages, attaches external stack-based memories to process terms where all the executed actions and discarded alternatives are stored. A single transition relation is present, with transitions being labeled with forward or backward actions. In contrast, the *static* approach of [55], which is convenient for handling labeled transition systems and basic process calculi, makes all process algebraic operators static – in particular action prefix and choice – so that executed actions and discarded alternatives are kept within the syntax. Two separate transition relations are generated, a forward one and a backward one, which require the introduction of communication keys to identify actions synchronizing with each other. The two approaches have been shown to be equivalent in terms of labeled transition system isomorphism [44] and the basic properties they exploit to ensure causal reversibility have been systematically studied in [46].

When quantitative aspects of system behavior are considered, other notions of reversibility may come into play. This is the case with *time reversibility* for stochastic processes like continuous-time Markov chains [39]. It ensures that the stochastic behavior of a *shared-resource* computing system remains the same when the direction of time is reversed and is instrumental to develop performance evaluation methods that cope with state space explosion and numerical instability problems. In [9] we have jointly investigated causal reversibility and time reversibility in the setting of a Markovian process calculus defined according to the static approach of [55], where every action is *integrated* with a positive real number quantifying the rate of the corresponding exponentially distributed random duration. Later on we have also examined the relationships between the two forms of reversibility directly on labeled transition system models [8].

Therefore the approach of [55, 46], originally developed for nondeterministic calculi, smoothly applies to *stochastically timed* calculi too. In this paper, which is an integrated and revised version of [10, 11], we show that this is no longer the case when mixing nondeterminism and deterministic time or probabilities. After recalling how the approach works for nondeterministic calculi (Section 2), we provide the necessary adaptations in terms of (*i*) operational semantic rules with respect to those in [55] and (*ii*) properties ensuring causal reversibility with respect to those in [46].

1.1. Reversibility, Nondeterminism, and Real Time

The first contribution of the paper (Section 3) is to address the reversibility of nondeterministic calculi for *real-time* systems. An example of application of such reversible calculi is given by transactional systems based on a micro-service architecture, where compensation mechanisms triggered by timeouts are used to maintain system consistency in the absence of a global clock [31].

Unlike [9], where time flows stochastically and is integrated within actions, here time flows *deterministically* and is described *orthogonally* to actions, i.e., through a delay prefix operator separate from action prefix. In the rich literature of deterministically timed process calculi – timed CSP [61], temporal CCS [51], timed CCS [72], real-time ACP [3], urgent LOTOS [14], CIPA [1], TPL [38], ATP [52], TIC [59], and PAFAS [21] – the differences are due to the following time-related options:

- *Durationless actions* versus *durational actions.* In the first case, actions are instantaneous events and time passes in between them; hence, functional behavior and time are *orthogonal.* In the second case, every action takes a certain amount of time to be performed and time passes only due to action execution; hence, functional behavior and time are *integrated.*

- *Relative time* versus *absolute time.* Assume that timestamps are associated with the events observed during system execution. In the first case, each timestamp refers to the time instant of the previous observation. In the second case, all timestamps refer to the starting time of the system execution.

- *Global clock* versus *local clocks.* In the first case, a single clock governs time passing. In the second case, several clocks associated with the various system parts elapse independently, although defining a unique notion of global time.

In addition, there are several different interpretations of action execution in terms of whether and when the execution can be delayed:

- *Eagerness*: actions must be performed as soon as they become enabled, i.e., without any delay, thereby implying that their execution is *urgent*.
- *Laziness*: after getting enabled, actions can be delayed arbitrarily long before they are executed.
- *Maximal progress*: enabled actions can be delayed unless they are independent of the external environment, in which case their execution is urgent.

The two major combinations of the aforementioned options yield the *two-phase functioning principle*, according to which actions are durationless, time is relative, and there is a single global clock, and the *one-phase functioning principle*, according to which actions are durational, time is absolute, and several local clocks are present. In [20] the two principles have been investigated under the various action execution interpretations through a bisimilarity-preserving encoding from the latter principle to the former, whilst the inverse encoding was provided in [6] along with a further pair of encodings for the case of stochastic delays.

Here we focus on the two-phase functioning principle, yielding action transitions separated from delay transitions like in temporal CCS [51], where time passing is governed by:

- *Time determinism*: time cannot decide which process is selected in a nondeterministic choice or advances in a parallel composition.
- *Time additivity*: a single delay transition can be split into several ones whose durations sum up to the duration of the original transition, as well as several delay transitions can be merged into a single one whose duration is the sum of the durations of the original ones.

We develop RTPC, a reversible timed process calculus inspired by [51] with lazy/eager actions whose syntax and semantics adhere to the static approach of [55], and show that it meets causal reversibility through notions of [22] and the technique of [46]. As for the design of the calculus and the achievement of causal reversibility, it turns out that the following adaptations are necessary with respect to [55, 22, 46]:

- Similar to executed actions, which have to be decorated with communication keys so as to know who synchronized with whom when building the backward transition relation [55], elapsed delays have to be decorated with keys to ensure that all subprocesses of an alternative or parallel composition go back in time in a well-paired way according to time determinism.
- Conflicting transitions, from which concurrent transitions [22] are then derived, and causal equivalence [22], which is needed to identify computations that differ for the order of concurrent action transitions, have to be extended with additional conditions specific to delay transitions.
- Backward transitions independence, one of the properties studied in [46] to ensure causal reversibility, does not hold in general due to time additivity. Furthermore, the semantic rules implementing laziness and maximal progress have to be carefully designed to guarantee the validity of the parabolic lemma [22, 46], another property supporting causal reversibility.

1.2. Reversibility, Nondeterminism, and Probabilities

The second contribution of the paper (Section 4) is to address the reversibility of untimed calculi featuring nondeterminism and probabilities and show that the approach of [55, 46] has to be adapted in a way different from timed calculi. An example of application of such reversible calculi is randomized dining philosophers [48], in which a philosopher may revoke the choice of the first chopstick if the second one is not available (possibly within a short amount of time like in [8]). A different example is given by speculative consumers [58], where to boost parallelism a consumer can probabilistically predict a value on the basis of which to launch a computation, which has to be undone if the guessed value is different from the one sent later by the producer. Yet another example is the smart contract rollback vulnerability [19]; for instance, in a lottery a smart contract makes a probabilistic choice to draft the winning ticket, but an attacker may try to revert the transaction in the case that the purchased ticket is different from the winning one [26].

There are several probabilistic state-transition models that can be used as a basis for a reversible calculus. A limited form of nondeterminism is allowed within *reactive* models [34], which correspond to Markov decision processes [25] and Rabin probabilistic automata [60]. Like in *generative* models [34], which correspond to action-labeled discrete-time Markov chains [40], each transition is labeled with an action and an execution probability, but probabilities are enforced only among transitions labeled with the same action. Therefore, in every state a nondeterministic selection is made among transitions labeled with different actions, then a probabilistic selection takes place inside the set of transitions labeled with that action.

Internal nondeterminism, i.e., nondeterministic choices among transitions labeled with the same action, is supported by Segala simple probabilistic automata [63]. In this model every transition is labeled only with an action and goes from a state to a probability distribution over states. In every state the transition to be executed is selected nondeterministically, then the reached state is selected probabilistically among those in the support of the target probability distribution of the chosen transition. This combination of probability and nondeterminism is called *non-alternating*.

By contrast, in the *alternating* model of [36] states are divided into nondeterministic and probabilistic, with transitions being classified as action transitions, which are labeled with an action and go from a nondeterministic state to a probabilistic one, and probabilistic transitions, which are labeled with a probability and go from a probabilistic state to a nondeterministic one. A more flexible variant, called *non-strictly alternating* model [54], admits action transitions between two nondeterministic states too.

Both the non-alternating model and the alternating one – whose relationships have been studied in [65] – encompass nondeterministic models, generative models, and reactive models as special cases. Branching bisimulation semantics [35] plays a fundamental role in reversible systems, as it turns out to coincide with back-and-forth variants of weak bisimilarity [24, 7, 29, 26, 27, 28]. Here we adopt the non-strictly alternating model because in [2] a probabilistic branching bisimulation congruence has been developed for it along with equational and logical characterizations and a polynomial-time decision procedure. In the non-alternating model, for which branching bisimilarity has been just defined in [64], weak variants of bisimulation semantics are more involved because, to achieve transitivity, they require that a single transition be matched by a convex combination of several transitions, corresponding to the use of randomized schedulers; decision procedures can be found in [16, 67].

We develop RPPC, a reversible probabilistic process calculus inspired by [37, 2] whose syntax and semantics adhere to the static approach of [55], and show that it meets causal reversibility through notions of [22] and the technique of [46]. As for the design of the calculus and the achievement of causal reversibility, it turns out that the following adaptations are necessary with respect to [55, 22, 46]:

- Similar to executed actions, which have to be decorated with communication keys so as to know who synchronized with whom when building the backward transition relation [55], probabilistic selections have to be decorated with keys to avoid wrong pairings in the backward direction when they have been performed on both sides of a nondeterministic choice or a parallel composition. Unlike time, which elapses in the same way across all alternative and parallel subprocesses, probabilistic selections can be made independently by parallel subprocesses in certain circumstances.

- To comply with the adopted model, in which the forward transitions departing from a state are all either action transitions or probabilistic transitions, like in the forward-only calculus of [2] probabilistic selections have to be made before nondeterministic ones when going forward. As a consequence, a probabilistic selection cannot resolve nondeterministic choices or decide which process advances in a parallel composition. This adds a technical challenge to the definition of the operational semantic rules with respect to [55], as nondeterministic selections – including those among concurrent actions – have to be revoked before probabilistic ones when going backward.

- Conflicting transitions, from which concurrent transitions [22] are then derived, and causal equivalence [22], which is needed to identify computations that differ for the order of concurrent transitions, have to be extended with additional conditions specific to probabilistic transitions.

- The square property and the parabolic lemma, two of the properties studied in [46] to ensure causal reversibility, have to be revised to deal with extended squares including probabilistic transitions too.

$$\boxed{\begin{array}{lcl} F, G & ::= & \underline{0} \mid a\,.\,F \mid F + G \mid F \parallel_L G \\ R, S & ::= & F \mid a[i]\,.\,R \mid R + S \mid R \parallel_L S \end{array}}$$

Table 1: Syntax of forward processes (top) and reversible processes (bottom) of RPC ($. > + > \parallel_L$)

2. Background: Causal Reversibility in Nondeterministic Process Calculi

In this section we recall the static approach of [55] by presenting the syntax and semantics of RPC – *Reversible Process Calculus*, whose operators are a mix of those of CCS [50] and CSP [15] (Section 2.1). Then we show how to prove the causal reversibility of RPC based on the approach of [46] and some concepts taken from [22] (Section 2.2). This will constitute the basis for developing causally reversible process calculi in which time (Section 3) or probabilities (Section 4) are added to nondeterminism.

2.1. Syntax and Semantics

Given a countable set $\mathcal{A}$ of actions – ranged over by a, b – including an unobservable action denoted by τ, the syntax of RPC is shown in Table 1 along with operator precedence. A standard *forward process* F is one of the following: the terminated process $\underline{0}$; the action-prefixed process $a\,.\,F$, which is able to perform action a and then continue as process F; the nondeterministic choice $F + G$, which is resolved based on the actions initially executable by processes F and G; or the parallel composition $F \parallel_L G$, indicating that processes F and G execute in parallel and must synchronize only on actions in $L \subseteq \mathcal{A} \setminus \{\tau\}$.

A *reversible process* R is built on top of forward processes. The syntax of reversible processes differs from the one of forward processes due to the fact that, in the former, each action prefix is decorated with a *communication key* i belonging to a countable set $\mathcal{K}$. A process of the form $a[i]\,.\,R$ expresses that in the past the process synchronized with the environment on a and this synchronization was identified by key i. Keys are thus attached only to executed actions and are necessary to remember who synchronized with whom when undoing actions; keys could be omitted in the absence of parallel composition.

We denote by $\mathcal{P}$ the set of processes generated by the second production in Table 1, while we use predicate $\texttt{std}(\_)$ to identify the standard forward processes that can be derived from the first production in the same table. For example, $a\,.\,b\,.\,\underline{0}$ is a standard forward process that can perform action a and then execute b, while $a[i]\,.\,b\,.\,\underline{0}$ is a non-standard reversible process that can either undo action a, or perform action b. Note that $a\,.\,b[j]\,.\,\underline{0}$ is not in $\mathcal{P}$ because a future action cannot precede a past one in the description of a process.

Let $\mathcal{L} = \mathcal{A} \times \mathcal{K}$ be ranged over by ℓ. The semantics for RPC is the labeled transition system $(\mathcal{P}, \mathcal{L}, \longmapsto_{\mathtt{a}})$, where the transition relation $\longmapsto_{\mathtt{a}} \subseteq \mathcal{P} \times \mathcal{L} \times \mathcal{P}$ is given by the union of the *forward transition relation* $\longrightarrow_{\mathtt{a}}$ and the *backward transition relation* $\dashrightarrow_{\mathtt{a}}$. In the definitions of the transition relations, we make use of the set $\texttt{key}_{\mathtt{a}}(R)$ of action keys in a process $R \in \mathcal{P}$:

$$\begin{array}{rcl} \texttt{key}_{\mathtt{a}}(F) & = & \emptyset \\ \texttt{key}_{\mathtt{a}}(a[i]\,.\,R) & = & \{i\} \cup \texttt{key}_{\mathtt{a}}(R) \\ \texttt{key}_{\mathtt{a}}(R + S) & = & \texttt{key}_{\mathtt{a}}(R) \cup \texttt{key}_{\mathtt{a}}(S) \\ \texttt{key}_{\mathtt{a}}(R \parallel_L S) & = & \texttt{key}_{\mathtt{a}}(R) \cup \texttt{key}_{\mathtt{a}}(S) \end{array}$$

as well as of predicate $\texttt{npa}(\_)$ to establish whether the considered process $R \in \mathcal{P}$ contains no past actions (note that $\texttt{std}(R)$ ensures $\texttt{npa}(R)$):

$$\begin{array}{rcl} \texttt{npa}(F) & = & \texttt{true} \\ \texttt{npa}(a[i]\,.\,R) & = & \texttt{false} \\ \texttt{npa}(R + S) & = & \texttt{npa}(R) \land \texttt{npa}(S) \\ \texttt{npa}(R \parallel_L S) & = & \texttt{npa}(R) \land \texttt{npa}(S) \end{array}$$

The action transition relations $\longrightarrow_{\mathtt{a}} \subseteq \mathcal{P} \times \mathcal{L} \times \mathcal{P}$ and $\dashrightarrow_{\mathtt{a}} \subseteq \mathcal{P} \times \mathcal{L} \times \mathcal{P}$ are the least relations respectively induced by the forward rules in the left part of Table 2 and by the backward rules in the right part of the same table. Note that the rules in the two columns are perfectly symmetric.

Rule Act1 handles processes of the form $a\,.\,F$, where F is written as R subject to $\texttt{std}(R)$. In addition to transforming the prefixing action a into a transition label, it generates a key i that is bound to action a

(Act1) $\dfrac{\mathtt{std}(R)}{a\,.\,R \xrightarrow{a[i]}_{\mathtt{a}} a[i]\,.\,R}$	(Act1$^\bullet$) $\dfrac{\mathtt{std}(R)}{a[i]\,.\,R \overset{a[i]}{\dashrightarrow}_{\mathtt{a}} a\,.\,R}$
(Act2) $\dfrac{R \xrightarrow{b[j]}_{\mathtt{a}} R' \quad j \neq i}{a[i]\,.\,R \xrightarrow{b[j]}_{\mathtt{a}} a[i]\,.\,R'}$	(Act2$^\bullet$) $\dfrac{R \overset{b[j]}{\dashrightarrow}_{\mathtt{a}} R' \quad j \neq i}{a[i]\,.\,R \overset{b[j]}{\dashrightarrow}_{\mathtt{a}} a[i]\,.\,R'}$
(Cho) $\dfrac{R \xrightarrow{a[i]}_{\mathtt{a}} R' \quad \mathtt{npa}(S)}{R + S \xrightarrow{a[i]}_{\mathtt{a}} R' + S}$	(Cho$^\bullet$) $\dfrac{R \overset{a[i]}{\dashrightarrow}_{\mathtt{a}} R' \quad \mathtt{npa}(S)}{R + S \overset{a[i]}{\dashrightarrow}_{\mathtt{a}} R' + S}$
(Par) $\dfrac{R \xrightarrow{a[i]}_{\mathtt{a}} R' \quad a \notin L \quad i \notin \mathtt{key}_{\mathtt{a}}(S)}{R \parallel_L S \xrightarrow{a[i]}_{\mathtt{a}} R' \parallel_L S}$	(Par$^\bullet$) $\dfrac{R \overset{a[i]}{\dashrightarrow}_{\mathtt{a}} R' \quad a \notin L \quad i \notin \mathtt{key}_{\mathtt{a}}(S)}{R \parallel_L S \overset{a[i]}{\dashrightarrow}_{\mathtt{a}} R' \parallel_L S}$
(Coo) $\dfrac{R \xrightarrow{a[i]}_{\mathtt{a}} R' \quad S \xrightarrow{a[i]}_{\mathtt{a}} S' \quad a \in L}{R \parallel_L S \xrightarrow{a[i]}_{\mathtt{a}} R' \parallel_L S'}$	(Coo$^\bullet$) $\dfrac{R \overset{a[i]}{\dashrightarrow}_{\mathtt{a}} R' \quad S \overset{a[i]}{\dashrightarrow}_{\mathtt{a}} S' \quad a \in L}{R \parallel_L S \overset{a[i]}{\dashrightarrow}_{\mathtt{a}} R' \parallel_L S'}$

Table 2: Operational semantic rules for RPC action transitions

thus yielding the label $a[i]$. Unlike the classical rule [50], according to [55] the prefix is not discarded by the application of this rule, instead it becomes a key-storing part of the target process that is necessary to offer again that action after rolling back. Rule Act1$^\bullet$ reverts the action $a[i]$ of the process $a[i]\,.\,R$ provided that R is a standard process, which ensures that $a[i]$ is the only past action that is left to undo.

The presence of rule Act2 – which is absent in [50] – 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 process $a[i]\,.\,R$ to execute if R itself can execute, provided that the action performed by R picks a key j different from i so that all the action prefixes in a sequence are decorated with distinct keys. Likewise, rule Act2$^\bullet$ propagates the execution of backward actions from inner subprocesses that are not standard as long as key uniqueness is preserved, so that past actions are overall undone from the most recent one to the least recent one.

Unlike the classical rules for nondeterministic choice [50], according to [55] rule Cho does not discard the part of the overall process that has not contributed to the executed action. If process R does an action, say $a[i]$, and becomes R', then the entire process $R + S$ becomes $R' + S$ as the information about $+\,S$ is necessary for offering again the original choice after rolling back. Once the choice is made, only the non-standard process R' can proceed further, with process S – which is standard – constituting a dead context of R'. Note that, in order to apply rule Cho, at least one between R and S must contain no past actions, meaning that it is impossible for two processes containing past actions to execute if they are composed by a choice operator. Rule Cho$^\bullet$ has precisely the same structure as rule Cho, but deals with the backward transition relation; if R' is standard, then the dead context S will come into play again. The symmetric variants of Cho and Cho$^\bullet$, in which it is S to move, are omitted.

The semantics of parallel composition is inspired by [15]. Rule Par allows process R within $R \parallel_L S$ to individually perform an action $a[i]$ provided that $a \notin L$. It is also checked that the executing action is bound to a fresh key $i \notin \mathtt{key}_{\mathtt{a}}(S)$, thus ensuring the uniqueness of communication keys across parallel composition too. Rule Coo instead allows both R and S to move by synchronizing on any action in the set L as long as the communication key is the same on both sides, i.e., $i \in \mathtt{key}_{\mathtt{a}}(R') \cap \mathtt{key}_{\mathtt{a}}(S')$. The resulting cooperation action has the same name and the same key. 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.

Example 2.1. To illustrate the need of communication keys, consider for instance the standard forward process $(a\,.\,F_1 \parallel_\emptyset a\,.\,F_2) \parallel_{\{a\}} (a\,.\,F_3 \parallel_\emptyset a\,.\,F_4)$, which may evolve to $(a[i]\,.\,F_1 \parallel_\emptyset a[j]\,.\,F_2) \parallel_{\{a\}} (a[i]\,.\,F_3 \parallel_\emptyset a[j]\,.\,F_4)$ after doing a forward $a[i]$-transition in which $a\,.\,F_1$ and $a\,.\,F_3$ synchronize followed by a forward $a[j]$-transition in which $a\,.\,F_2$ and $a\,.\,F_4$ synchronize. When going backward, in the absence of communication keys i and j

we could not know that the a preceding F_1 (resp. F_2) synchronized with the a preceding F_3 (resp. F_4). ■

Process syntax prevents future actions from preceding past ones. However, this is not the only necessary limitation, because not all the processes generated by the considered grammar are semantically meaningful. On the one hand, in the case of a choice at least one of the two subprocesses has to contain no past actions, hence for instance $a[i] \,.\, \underline{0} + b[j] \,.\, \underline{0}$ is not admissible as it indicates that both branches have been selected. On the other hand, key uniqueness must be enforced within processes containing past actions, so for example $a[i] \,.\, b[i] \,.\, \underline{0}$ and $a[i] \,.\, \underline{0} \,\|_{\emptyset}\, b[i] \,.\, \underline{0}$ are not admissible either. In the following we thus consider only *reachable processes*, whose set we denote by $\mathbb{P}$. They include processes from which a computation can start, i.e., standard forward processes, as well as processes that can be derived from the previous ones via finitely many applications of the semantic rules in Table 2. Given a reachable process $R \in \mathbb{P}$, if $\mathtt{npa}(R)$ then $\mathtt{key}_{\mathtt{a}}(R) = \emptyset$ while $\mathtt{key}_{\mathtt{a}}(R') \neq \emptyset$ for every other process R' reachable from R as any action labeling a transition along the path from R to R' has been equipped with a key inside R'.

2.2. Causal Reversibility

The causal reversibility of RPC is the capability of each reachable process of backtracking *correctly*, i.e., without encountering previously inaccessible states, and *flexibly*, i.e., along any path that is causally equivalent to the one undertaken in the forward direction. The investigation of causal reversibility is conducted through the technique of [46] by making use of some notions taken from [22].

A necessary condition for reversibility is the *loop property* [22, 55, 46]. It establishes that each executed action can be undone and that each undone action can be redone. Therefore, when considering the states associated with two arbitrary reachable processes, it must be the case that either there is no transition between them, or there is a pair of identically labeled transitions such that one is a forward transition from the first to the second state while the other is a backward transition from the second to the first state.

Proposition 2.2. [loop property] Let $R, S \in \mathbb{P}$ and $a[i] \in \mathcal{L}$. Then $R \xrightarrow{a[i]}_{\mathtt{a}} S$ iff $S \overset{a[i]}{\dashrightarrow}_{\mathtt{a}} R$.

Proof By induction on the depth of the derivation of either transition and then by case analysis on the last applied rule, as each direct rule in Table 2 has a corresponding reverse rule in the same table and vice versa. ■

Given a transition $\theta : R \overset{\ell}{\longmapsto}_{\mathtt{a}} S$ with $R, S \in \mathbb{P}$ and $\ell \in \mathcal{L}$, we call R the *source* of θ and S its *target*. If θ is a forward transition, i.e., $\theta : R \xrightarrow{\ell}_{\mathtt{a}} S$, we denote by $\overline{\theta} : S \overset{\ell}{\dashrightarrow}_{\mathtt{a}} R$ the corresponding backward transition. Two transitions are said to be *coinitial* if they have the same source and *cofinal* if they have the same target. Two transitions are *composable* when the target of the first transition coincides with the source of the second transition. A finite sequence of pairwise composable transitions is called a *path*. We use ε for the empty path and ω to range over paths, with $|\omega|$ denoting the length of ω, i.e., the number of transitions constituting ω. When ω is a forward path, we denote by $\overline{\omega}$ the corresponding backward path, where the order of the transitions is reversed. The notions of source, target, coinitiality, cofinality, and composability naturally extend to paths. We indicate with $\omega_1\omega_2$ the composition of the two paths ω_1 and ω_2 when they are composable, i.e., when the target of the last transition of ω_1 coincides with the source of the first transition of ω_2.

Before specifying when two transitions are concurrent [22], we need to present the notion of process context along with the set of causes – identified by action keys – that lead to a given communication key. A *process context* is a process with a hole $\bullet$ in it, which is generated by the following grammar:

$$\mathcal{C} \ ::= \ \bullet \mid a[i] \,.\, \mathcal{C} \mid R + \mathcal{C} \mid \mathcal{C} + R \mid R \,\|_L\, \mathcal{C} \mid \mathcal{C} \,\|_L\, R$$

We write $\mathcal{C}[R]$ to denote the process obtained by replacing the hole in $\mathcal{C}$ with R.

The *causal set* $\mathtt{cau}(R, i)$ of $R \in \mathbb{P}$ until $i \in \mathcal{K}$ is inductively defined as follows:

$$\begin{array}{rcl}
\mathtt{cau}(F, i) & = & \emptyset \\
\mathtt{cau}(a[j] \,.\, R, i) & = & \begin{cases} \{j\} \cup \mathtt{cau}(R, i) & \text{if } j \neq i \text{ and } i \in \mathtt{key}_{\mathtt{a}}(R) \\ \emptyset & \text{otherwise} \end{cases} \\
\mathtt{cau}(R + S, i) & = & \mathtt{cau}(R, i) \cup \mathtt{cau}(S, i) \\
\mathtt{cau}(R \,\|_L\, S, i) & = & \mathtt{cau}(R, i) \cup \mathtt{cau}(S, i)
\end{array}$$

If $i \in \mathtt{key}_{\mathtt{a}}(R)$ then $\mathtt{cau}(R, i)$ represents the set of keys in R that caused i, with $\mathtt{cau}(R, i) \subsetneq \mathtt{key}_{\mathtt{a}}(R)$ as on the one hand $i \notin \mathtt{cau}(R, i)$ and on the other hand keys that are not causally related to i are not considered. A key j causes i if it appears syntactically before i in R; equivalently, i is inside the scope of j.

We are now in a position to recall, for coinitial transitions, what we mean by concurrent transitions on the basis of the notion of conflicting transitions. The first condition below tells that a forward action transition is in conflict with a coinitial backward one whenever the latter tries to undo a cause of the key of the former, while the second one deems as conflictual two action transitions respectively generated by the two subprocesses of a choice.

Definition 2.3. [conflicting and concurrent transitions] Two coinitial transitions θ_1 and θ_2 from a process $R \in \mathbb{P}$ are *in conflict* if one of the following conditions holds, otherwise they are said to be *concurrent*:

1. $\theta_1 : R \xrightarrow{a[i]}_{\mathtt{a}} S_1$ and $\theta_2 : R \overset{b[j]}{\dashrightarrow}_{\mathtt{a}} S_2$ with $j \in \mathtt{cau}(S_1, i)$.

2. $R = \mathcal{C}[F_1 + F_2]$ with θ_k deriving from $F_k \xrightarrow{a_k[i_k]}_{\mathtt{a}} S_k$ for $k = 1, 2$. ■

We prove causal reversibility by exploiting the technique of [46], according to which causal reversibility stems from the *square property* – which amounts to concurrent transitions being confluent, hence they can commute while conflicting ones cannot – *backward transitions independence* – which generalizes the concept of backward determinism used for reversible sequential languages [74] – and *past well foundedness* – which ensures that reachable processes have a finite past.

Lemma 2.4. [square property] Let $\theta_1 : R \xmapsto{\ell_1}_{\mathtt{a}} S_1$ and $\theta_2 : R \xmapsto{\ell_2}_{\mathtt{a}} S_2$ be two coinitial transitions from a process $R \in \mathbb{P}$. If θ_1 and θ_2 are concurrent, then there exist two cofinal transitions $\theta_2' : S_1 \xmapsto{\ell_2}_{\mathtt{a}} S$ and $\theta_1' : S_2 \xmapsto{\ell_1}_{\mathtt{a}} S$ with $S \in \mathbb{P}$.

Proof The proof is by case analysis on the directions of θ_1 and θ_2. We distinguish three cases according to whether the two transitions are both forward, both backward, or one forward and the other backward:

- If θ_1 and θ_2 are both forward, since θ_1 and θ_2 are concurrent, by virtue of condition 2 of Definition 2.3 the two forward transitions cannot originate from a choice operator. They must thus be generated by a parallel composition, but not through rule Coo because θ_1 and θ_2 must have different keys and hence cannot synchronize. Without loss of generality, we assume that $R = R_1 \parallel_L R_2$ with $R_1 \xrightarrow{a[i]}_{\mathtt{a}} S_1$, $R_2 \xrightarrow{b[j]}_{\mathtt{a}} S_2$, $a, b \notin L$, and $i \neq j$. By applying rule Par we have that $R_1 \parallel_L R_2 \xrightarrow{a[i]}_{\mathtt{a}} S_1 \parallel_L R_2 \xrightarrow{b[j]}_{\mathtt{a}} S_1 \parallel_L S_2$ as well as $R_1 \parallel_L R_2 \xrightarrow{b[j]}_{\mathtt{a}} R_1 \parallel_L S_2 \xrightarrow{a[i]}_{\mathtt{a}} S_1 \parallel_L S_2$.

- If θ_1 and θ_2 are both backward, then we proceed like in the previous case, with the two coinitial transitions not originating from a choice operator because they go backward. Rule Par$^\bullet$ is then employed.

- If θ_1 is forward and θ_2 is backward, since θ_1 and θ_2 are concurrent, by virtue of condition 1 of Definition 2.3 it holds that θ_2 cannot remove any cause of θ_1. Since any subprocess of a choice or a parallel composition cannot perform both a forward transition and a backward transition without preventing the backward one from removing a cause of the forward one, and in the case of a choice only one of the two subprocesses can perform transitions after the choice has been made (as would be in our case in which we consider a backward transition), without loss of generality we assume that $R = R_1 \parallel_L R_2$ with $R_1 \xrightarrow{a[i]}_{\mathtt{a}} S_1$, $R_2 \overset{b[j]}{\dashrightarrow}_{\mathtt{a}} S_2$, $a, b \notin L$, and $i \neq j$. By applying rules Par and Par$^\bullet$ we have that $R_1 \parallel_L R_2 \xrightarrow{a[i]}_{\mathtt{a}} S_1 \parallel_L R_2 \overset{b[j]}{\dashrightarrow}_{\mathtt{a}} S_1 \parallel_L S_2$ as well as $R_1 \parallel_L R_2 \overset{b[j]}{\dashrightarrow}_{\mathtt{a}} R_1 \parallel_L S_2 \xrightarrow{a[i]}_{\mathtt{a}} S_1 \parallel_L S_2$. ■

Lemma 2.5. [backward transitions independence] Let $R \in \mathbb{P}$. Then any two coinitial backward transitions $\theta_1 : R \overset{\ell_1}{\dashrightarrow}_{\mathtt{a}} S_1$ and $\theta_2 : R \overset{\ell_2}{\dashrightarrow}_{\mathtt{a}} S_2$ are concurrent.

Proof By Definition 2.3 it is not possible for two backward transitions to be in conflict. ■

Lemma 2.6. [past well foundedness] Let $R_0 \in \mathbb{P}$. Then there is no infinite sequence of backward transitions such that $R_i \overset{\ell_i}{\dashrightarrow}_{\mathtt{a}} R_{i+1}$ for all $i \in \mathbb{N}$.

Proof By induction on $|\mathtt{key}_{\mathtt{a}}(R_0)|$, as every backward transition connects two different processes and hence decreases by one the (finite) number of past actions in R_0. ■

Following [22, 49], we also recall a notion of *causal equivalence* over paths, which abstracts from the order of concurrent action transitions. In this way, paths obtained by swapping the order of their concurrent action transitions are identified with each other. Moreover, the composition of a transition with its inverse is identified with the empty path.

Definition 2.7. [causal equivalence] *Causal equivalence* $\asymp$ is the smallest equivalence relation over paths traversing processes in $\mathbb{P}$ that is closed under composition and satisfies the following clauses:

1. $\theta_1\theta_2' \asymp \theta_2\theta_1'$ for every two coinitial concurrent action transitions $\theta_1 : R \xmapsto{a[i]}_{\mathtt{a}} R_1$ and $\theta_2 : R \xmapsto{b[j]}_{\mathtt{a}} R_2$ and every two cofinal action transitions $\theta_2' : R_1 \xmapsto{b[j]}_{\mathtt{a}} S$ and $\theta_1' : R_2 \xmapsto{a[i]}_{\mathtt{a}} S$.
2. $\theta\overline{\theta} \asymp \varepsilon$ and $\overline{\theta}\theta \asymp \varepsilon$ for every action transition θ. ■

The further property below, called *parabolic lemma* in [46], states that every path can be seen as a backward path followed by a forward path. As observed in [22], up to causal equivalence one can always reach for the maximum freedom of choice among transitions by going backward and only then going forward (not the other way around). Intuitively, one could depict computations as parabolas: the system first draws potential energy from its memory, by undoing all the executed actions, and then restarts.

Lemma 2.8. [parabolic lemma] For each path ω traversing processes in $\mathbb{P}$ there exist two forward paths ω_1 and ω_2 traversing processes in $\mathbb{P}$ such that $\omega \asymp \overline{\omega_1}\omega_2$ and $|\omega_1| + |\omega_2| \leq |\omega|$.

Proof A consequence of the square property and backward transitions independence [46]. ■

We conclude by obtaining a property called *causal consistency* in [46], which establishes that being coinitial and cofinal is necessary and sufficient in order for two paths to be causally equivalent, i.e., to contain concurrent action transitions in different orders (swap) or to be one the empty path and the other a transition followed by its reverse (cancelation).

Theorem 2.9. [causal consistency] Let ω_1 and ω_2 be two paths traversing processes in $\mathbb{P}$. Then $\omega_1 \asymp \omega_2$ iff ω_1 and ω_2 are both coinitial and cofinal.

Proof A consequence of past well foundedness and the parabolic lemma [46]. ■

Theorem 2.9 shows that causal equivalence characterizes over RPC a space for admissible rollbacks that are (*i*) correct as they do not lead to processes not reachable by some forward path and (*ii*) flexible enough to allow undo operations to be rearranged with respect to the order in which the undone concurrent action transitions were originally performed. This implies that the processes reached by any backward path could be reached by performing forward paths only. We can thus conclude that RPC meets causal reversibility.

3. Causal Reversibility in Nondeterministic and Real-Time Process Calculi

In this section we present the syntax and semantics of RTPC – *Reversible Timed Process Calculus*, which are inspired by those of temporal CCS [51] and tailored for a reversible setting according to the static approach of [55] (Section 3.1). Then we illustrate how RTPC works by means of an example based on a timeout mechanism (Section 3.2). Finally we investigate the causal reversibility of RTPC (Section 3.3).

F, G	$::=$	$\underline{0} \mid a \,.\, F \mid (n) \,.\, F \mid F + G \mid F \parallel_L G$
R, S	$::=$	$F \mid a[i] \,.\, R \mid (n)^{[i]} \,.\, R \mid R + S \mid R \parallel_L S$

Table 3: Syntax of forward processes (top) and reversible processes (bottom) of RTPC ($. > + > \parallel_L$)

3.1. Syntax and Semantics

The syntax of RTPC is shown in Table 3, featuring the delay prefix operator. We assume *time determinism* [51], i.e., time passing cannot solve choices or decide which process advances, hence the same amount of time must pass in all subprocesses of a nondeterministic choice or parallel composition. Under *eagerness*, the execution of all actions is urgent and $\underline{0}$ cannot let time pass. Under *laziness*, action execution can be delayed arbitrarily long and $\underline{0}$ lets time pass. Under *maximal progress*, only the execution of τ is urgent, because τ cannot be involved in synchronizations, and $\underline{0}$ lets time pass.

The delay-prefixed process $(n) \,.\, F$ lets $n \in \mathbb{N}_{>0}$ time units pass and then continues as process F. Its reversible variant $(n)^{[i]} \,.\, R$ means that n time units elapsed in the past. Here communication key i is needed to ensure time determinism in the backward direction, so that all subprocesses go back in time in the same way; keys could be omitted from delays in the absence of nondeterministic choice and parallel composition.

We denote by $\mathcal{P}_\mathtt{t}$ the set of processes generated by the second production in Table 3, while we use predicate $\mathtt{std}(\_)$ to identify the standard forward processes that can be derived from the first production in the same table. For example, $a \,.\, (5) \,.\, b \,.\, \underline{0}$ is a standard forward process that can perform action a, let 5 time units pass, and then execute b, while $a[i] \,.\, (5)^{[j]} \,.\, b \,.\, \underline{0}$ is a non-standard reversible process that can either go back by 5 time units and undo action a, or perform action b. Note that $a \,.\, (5)^{[j]} \,.\, b \,.\, \underline{0}$ and $a \,.\, (5) \,.\, b[j] \,.\, \underline{0}$ are not in $\mathcal{P}_\mathtt{t}$ because a future action or delay cannot precede a past one in the description of a process.

Let $\mathcal{A}_\mathcal{K} = \mathcal{A} \times \mathcal{K}$ and $\mathbb{N}_\mathcal{K} = \mathbb{N}_{>0} \times \mathcal{K}$, with $\mathcal{L}_\mathtt{t} = \mathcal{A}_\mathcal{K} \cup \mathbb{N}_\mathcal{K}$ ranged over by ℓ. The semantics for RTPC is the labeled transition system $(\mathcal{P}'_\mathtt{t}, \mathcal{L}_\mathtt{t}, \longmapsto_\mathtt{t})$, where $\mathcal{P}'_\mathtt{t}$ is obtained from $\mathcal{P}_\mathtt{t}$ by adding $\langle n^i \rangle \,.\, R$ to the second production in Table 3; in the following $\delta(n, i)$ denotes $(n)^{[i]}$ or $\langle n^i \rangle$, with the use of the latter being explained later on. The transition relation $\longmapsto_\mathtt{t} \subseteq \mathcal{P}'_\mathtt{t} \times \mathcal{L}_\mathtt{t} \times \mathcal{P}'_\mathtt{t}$ is given by $\longmapsto_\mathtt{t} \;=\; \longrightarrow_\mathtt{t} \cup \dashrightarrow_\mathtt{t}$ where in turn the *forward transition relation* is given by $\longrightarrow_\mathtt{t} \;=\; \longrightarrow_\mathtt{a} \cup \longrightarrow_\mathtt{d}$ and the *backward transition relation* is given by $\dashrightarrow_\mathtt{t} \;=\; \dashrightarrow_\mathtt{a} \cup \dashrightarrow_\mathtt{d}$, with subscript $\mathtt{a}$ denoting action transitions and subscript $\mathtt{d}$ denoting delay transitions. In the definitions of the transition relations, we make use of the set $\mathtt{key}_\mathtt{a}(R)$ of action keys in a process $R \in \mathcal{P}'_\mathtt{t}$ as well as of predicate $\mathtt{npa}(\_)$ to establish whether the considered process contains no past actions, whose definitions are extended as follows with respect to Section 2.1:

$$\begin{array}{rcl} \mathtt{key}_\mathtt{a}(\delta(n, i) \,.\, R) & = & \mathtt{key}_\mathtt{a}(R) \\ \mathtt{npa}(\delta(n, i) \,.\, R) & = & \mathtt{npa}(R) \end{array}$$

The *action transition relations* $\longrightarrow_\mathtt{a} \subseteq \mathcal{P}'_\mathtt{t} \times \mathcal{A}_\mathcal{K} \times \mathcal{P}'_\mathtt{t}$ and $\dashrightarrow_\mathtt{a} \subseteq \mathcal{P}'_\mathtt{t} \times \mathcal{A}_\mathcal{K} \times \mathcal{P}'_\mathtt{t}$, whose union we denote by $\longmapsto_\mathtt{a}$, are the least relations respectively induced by the forward rules in the left part of Table 4 and by the backward rules in the right part of the same table. The only new rules with respect to Table 2 are ACT3 and ACT3$^\bullet$, which play a propagating role in an elapsed delay context analogous to the one of rules ACT2 and ACT2$^\bullet$ in an executed action context. Note that executed actions and elapsed delays are not required to exhibit different keys.

The *delay transition relations* $\longrightarrow_\mathtt{d} \subseteq \mathcal{P}'_\mathtt{t} \times \mathbb{N}_\mathcal{K} \times \mathcal{P}'_\mathtt{t}$ and $\dashrightarrow_\mathtt{d} \subseteq \mathcal{P}'_\mathtt{t} \times \mathbb{N}_\mathcal{K} \times \mathcal{P}'_\mathtt{t}$, whose union we denote by $\longmapsto_\mathtt{d}$, are the least relations respectively induced by the forward rules in the left part of Table 5 and by the backward rules in the right part of the same table. Note that the rules in the two columns are perfectly symmetric.

As for the three idling rules, it is necessary to use specific combinations. None of them is present under eagerness. Rules IDLING1 and IDLING2 encode laziness: $\underline{0}$ can let time pass and every action can be delayed arbitrarily long. Rules IDLING1 and IDLING3 encode maximal progress: $\underline{0}$ can let time pass and every action other than τ can be delayed arbitrarily long. Rules IDLING1$^\bullet$, IDLING2$^\bullet$, and IDLING3$^\bullet$ play the corresponding roles in the backward direction. Note that the three forward rules introduce a *dynamic* delay prefix $\langle n^i \rangle$ in the target process, which is then removed from the source process by the three backward rules. The need for $\langle n^i \rangle$ in case of idling will be illustrated in Example 3.12.

(Act1)	$\dfrac{\mathtt{std}(R)}{a\,.\,R \xrightarrow{a[i]}_{\mathtt{a}} a[i]\,.\,R}$	(Act1$^{\bullet}$)	$\dfrac{\mathtt{std}(R)}{a[i]\,.\,R \overset{a[i]}{\dashrightarrow}_{\mathtt{a}} a\,.\,R}$
(Act2)	$\dfrac{R \xrightarrow{b[j]}_{\mathtt{a}} R' \quad j \neq i}{a[i]\,.\,R \xrightarrow{b[j]}_{\mathtt{a}} a[i]\,.\,R'}$	(Act2$^{\bullet}$)	$\dfrac{R \overset{b[j]}{\dashrightarrow}_{\mathtt{a}} R' \quad j \neq i}{a[i]\,.\,R \overset{b[j]}{\dashrightarrow}_{\mathtt{a}} a[i]\,.\,R'}$
(Act3)	$\dfrac{R \xrightarrow{b[j]}_{\mathtt{a}} R'}{\delta(n,i)\,.\,R \xrightarrow{b[j]}_{\mathtt{a}} \delta(n,i)\,.\,R'}$	(Act3$^{\bullet}$)	$\dfrac{R \overset{b[j]}{\dashrightarrow}_{\mathtt{a}} R'}{\delta(n,i)\,.\,R \overset{b[j]}{\dashrightarrow}_{\mathtt{a}} \delta(n,i)\,.\,R'}$
(Cho)	$\dfrac{R \xrightarrow{a[i]}_{\mathtt{a}} R' \quad \mathtt{npa}(S)}{R + S \xrightarrow{a[i]}_{\mathtt{a}} R' + S}$	(Cho$^{\bullet}$)	$\dfrac{R \overset{a[i]}{\dashrightarrow}_{\mathtt{a}} R' \quad \mathtt{npa}(S)}{R + S \overset{a[i]}{\dashrightarrow}_{\mathtt{a}} R' + S}$
(Par)	$\dfrac{R \xrightarrow{a[i]}_{\mathtt{a}} R' \quad a \notin L \quad i \notin \mathtt{key}_{\mathtt{a}}(S)}{R \parallel_L S \xrightarrow{a[i]}_{\mathtt{a}} R' \parallel_L S}$	(Par$^{\bullet}$)	$\dfrac{R \overset{a[i]}{\dashrightarrow}_{\mathtt{a}} R' \quad a \notin L \quad i \notin \mathtt{key}_{\mathtt{a}}(S)}{R \parallel_L S \overset{a[i]}{\dashrightarrow}_{\mathtt{a}} R' \parallel_L S}$
(Coo)	$\dfrac{R \xrightarrow{a[i]}_{\mathtt{a}} R' \quad S \xrightarrow{a[i]}_{\mathtt{a}} S' \quad a \in L}{R \parallel_L S \xrightarrow{a[i]}_{\mathtt{a}} R' \parallel_L S'}$	(Coo$^{\bullet}$)	$\dfrac{R \overset{a[i]}{\dashrightarrow}_{\mathtt{a}} R' \quad S \overset{a[i]}{\dashrightarrow}_{\mathtt{a}} S' \quad a \in L}{R \parallel_L S \overset{a[i]}{\dashrightarrow}_{\mathtt{a}} R' \parallel_L S'}$

Table 4: Operational semantic rules for RTPC action transitions

The pairs of rules Delay1 and Delay1$^{\bullet}$, Delay2 and Delay2$^{\bullet}$, and Delay3 and Delay3$^{\bullet}$ are respectively the delay counterparts of the pairs of rules Act1 and Act1$^{\bullet}$, Act2 and Act2$^{\bullet}$, and Act3 and Act3$^{\bullet}$. We remind that executed actions and elapsed delays are allowed to share the same keys.

Rules TAdd1 and TAdd2, which are variants of rule Delay1, encode *time additivity* [51]. The former *merges* several consecutive delays in the transition label so that they elapse as a single delay, while the latter *splits* a single delay within the process into several shorter delays that elapse consecutively. For instance, process $(2)\,.\,(1)\,.\,\underline{0}$ can evolve to $(2)^{[i]}\,.\,(1)\,.\,\underline{0}$ via a 2-delay transition generated by Delay1, $(2)^{[i]}\,.\,(1)^{[j]}\,.\,\underline{0}$ via a 3-delay transition generated by TAdd1, or $\langle 1^{i}\rangle\,.\,(1)\,.\,(1)\,.\,\underline{0}$ via a 1-delay transition generated by TAdd2. Rules TAdd1$^{\bullet}$ and TAdd2$^{\bullet}$ respectively undo the merging and splitting operations over delays. As for splitting, note that elapsed delays are treated as dynamic. If this were not the case, then for example instead of $\langle 1^{i}\rangle\,.\,(1)\,.\,(1)\,.\,\underline{0}$, to which only TAdd2$^{\bullet}$ is applicable leading to $(2)\,.\,(1)\,.\,\underline{0}$, we would have $(1)^{[i]}\,.\,(1)\,.\,(1)\,.\,\underline{0}$, to which also Delay1$^{\bullet}$ is applicable erroneously leading to $(1)\,.\,(1)\,.\,(1)\,.\,\underline{0}$.

Rules TCho1 and TCoo encode *time determinism* [51]. Time advances in the same way with the same key in the two subprocesses of a choice or a parallel composition, without making any decision. Rule TCho2 is necessary for dealing with the case in which the nondeterministic choice has already been resolved due to the execution of an action by R. Rules TCho1$^{\bullet}$, TCoo$^{\bullet}$, and TCho2$^{\bullet}$ play the corresponding roles in the backward direction. The symmetric variants of TCho2 and TCho2$^{\bullet}$, in which it is S to move, are omitted.

Example 3.1. To illustrate the need of communication keys also for delays (new with respect to [55]), consider the standard forward process $(n)\,.\,\underline{0} \parallel_{\emptyset} (n)\,.\,\underline{0}$, which may evolve to $(n)^{[i]}\,.\,\underline{0} \parallel_{\emptyset} (n)^{[i]}\,.\,\underline{0}$ after that delay n has elapsed. When going backward under laziness or maximal progress, in the absence of key i it may happen that $\underline{0}$ in either subprocess lets n time units pass backward – due to Idling1$^{\bullet}$ – with this pairing with undoing delay n in the other subprocess – due to Delay1$^{\bullet}$ – which results in a process where one delay n can elapse forward again whereas the other one cannot. The presence of keys forces the idling of either $\underline{0}$ to synchronize with the idling of the other $\underline{0}$. The same situation would take place with $+$ in lieu of $\parallel_{\emptyset}$. By contrast, the problem does not show up under eagerness. In that case, elapsed delays can be uniformly decorated, for example with † like in [9]. ■

(Idling1) $\underline{0} \xrightarrow{(n)^{[i]}}_{\mathtt{d}} \langle n^i \rangle . \underline{0}$	(Idling1$^{\bullet}$) $\langle n^i \rangle . \underline{0} \overset{(n)^{[i]}}{\dashrightarrow}_{\mathtt{d}} \underline{0}$
(Idling2) $\dfrac{\mathtt{std}(R)}{a . R \xrightarrow{(n)^{[i]}}_{\mathtt{d}} \langle n^i \rangle . a . R}$	(Idling2$^{\bullet}$) $\dfrac{\mathtt{std}(R)}{\langle n^i \rangle . a . R \overset{(n)^{[i]}}{\dashrightarrow}_{\mathtt{d}} a . R}$
(Idling3) $\dfrac{\mathtt{std}(R) \quad a \neq \tau}{a . R \xrightarrow{(n)^{[i]}}_{\mathtt{d}} \langle n^i \rangle . a . R}$	(Idling3$^{\bullet}$) $\dfrac{\mathtt{std}(R) \quad a \neq \tau}{\langle n^i \rangle . a . R \overset{(n)^{[i]}}{\dashrightarrow}_{\mathtt{d}} a . R}$
(Delay1) $\dfrac{\mathtt{std}(R)}{(n) . R \xrightarrow{(n)^{[i]}}_{\mathtt{d}} (n)^{[i]} . R}$	(Delay1$^{\bullet}$) $\dfrac{\mathtt{std}(R)}{(n)^{[i]} . R \overset{(n)^{[i]}}{\dashrightarrow}_{\mathtt{d}} (n) . R}$
(Delay2) $\dfrac{R \xrightarrow{(n)^{[j]}}_{\mathtt{d}} R'}{a[i] . R \xrightarrow{(n)^{[j]}}_{\mathtt{d}} a[i] . R'}$	(Delay2$^{\bullet}$) $\dfrac{R \overset{(n)^{[j]}}{\dashrightarrow}_{\mathtt{d}} R'}{a[i] . R \overset{(n)^{[j]}}{\dashrightarrow}_{\mathtt{d}} a[i] . R'}$
(Delay3) $\dfrac{R \xrightarrow{(m)^{[j]}}_{\mathtt{d}} R' \quad j \neq i}{\delta(n,i) . R \xrightarrow{(m)^{[j]}}_{\mathtt{d}} \delta(n,i) . R'}$	(Delay3$^{\bullet}$) $\dfrac{R \overset{(m)^{[j]}}{\dashrightarrow}_{\mathtt{d}} R' \quad j \neq i}{\delta(n,i) . R \overset{(m)^{[j]}}{\dashrightarrow}_{\mathtt{d}} \delta(n,i) . R'}$
(TAdd1) $\dfrac{\mathtt{std}(R) \quad R \xrightarrow{(m)^{[j]}}_{\mathtt{d}} R' \quad j \neq i}{(n) . R \xrightarrow{(n+m)^{[i]}}_{\mathtt{d}} (n)^{[i]} . R'}$	(TAdd1$^{\bullet}$) $\dfrac{\mathtt{std}(R') \quad R \overset{(m)^{[j]}}{\dashrightarrow}_{\mathtt{d}} R' \quad j \neq i}{(n)^{[i]} . R \overset{(n+m)^{[i]}}{\dashrightarrow}_{\mathtt{d}} (n) . R'}$
(TAdd2) $\dfrac{\mathtt{std}(R) \quad n = n_1 + n_2}{(n) . R \xrightarrow{(n_1)^{[i]}}_{\mathtt{d}} \langle n_1^i \rangle . (n_2) . R}$	(TAdd2$^{\bullet}$) $\dfrac{\mathtt{std}(R) \quad n = n_1 + n_2}{\langle n_1^i \rangle . (n_2) . R \overset{(n_1)^{[i]}}{\dashrightarrow}_{\mathtt{d}} (n) . R}$
(TCho1) $\dfrac{\mathtt{npa}(R+S) \quad R \xrightarrow{(n)^{[i]}}_{\mathtt{d}} R' \quad S \xrightarrow{(n)^{[i]}}_{\mathtt{d}} S'}{R + S \xrightarrow{(n)^{[i]}}_{\mathtt{d}} R' + S'}$	(TCho1$^{\bullet}$) $\dfrac{\mathtt{npa}(R+S) \quad R \overset{(n)^{[i]}}{\dashrightarrow}_{\mathtt{d}} R' \quad S \overset{(n)^{[i]}}{\dashrightarrow}_{\mathtt{d}} S'}{R + S \overset{(n)^{[i]}}{\dashrightarrow}_{\mathtt{d}} R' + S'}$
(TCho2) $\dfrac{\neg\mathtt{npa}(R) \quad \mathtt{npa}(S) \quad R \xrightarrow{(n)^{[i]}}_{\mathtt{d}} R'}{R + S \xrightarrow{(n)^{[i]}}_{\mathtt{d}} R' + S}$	(TCho2$^{\bullet}$) $\dfrac{\neg\mathtt{npa}(R) \quad \mathtt{npa}(S) \quad R \overset{(n)^{[i]}}{\dashrightarrow}_{\mathtt{d}} R'}{R + S \overset{(n)^{[i]}}{\dashrightarrow}_{\mathtt{d}} R' + S}$
(TCoo) $\dfrac{R \xrightarrow{(n)^{[i]}}_{\mathtt{d}} R' \quad S \xrightarrow{(n)^{[i]}}_{\mathtt{d}} S'}{R \parallel_L S \xrightarrow{(n)^{[i]}}_{\mathtt{d}} R' \parallel_L S'}$	(TCoo$^{\bullet}$) $\dfrac{R \overset{(n)^{[i]}}{\dashrightarrow}_{\mathtt{d}} R' \quad S \overset{(n)^{[i]}}{\dashrightarrow}_{\mathtt{d}} S'}{R \parallel_L S \overset{(n)^{[i]}}{\dashrightarrow}_{\mathtt{d}} R' \parallel_L S'}$

Table 5: Operational semantic rules for RTPC delay transitions

$$\begin{tikzcd}
& (2)\,.\,a\,.\,\underline{0} + (2)\,.\,b\,.\,\underline{0} \arrow[dl, bend right, "(1)^{[i]}"'] \arrow[dr, bend left, "(2)^{[k]}"] & \\
\langle 1^i\rangle\,.\,(1)\,.\,a\,.\,\underline{0} + \langle 1^i\rangle\,.\,(1)\,.\,b\,.\,\underline{0} \arrow[ur, dashed, "(1)^{[i]}"'] \arrow[d, bend right, "(1)^{[j]}"'] & & (2)^{[k]}.\,a\,.\,\underline{0} + (2)^{[k]}.\,b\,.\,\underline{0} \arrow[ul, dashed, "(2)^{[k]}"] \\
\langle 1^i\rangle\,.\,(1)^{[j]}.\,a\,.\,\underline{0} + \langle 1^i\rangle\,.\,(1)^{[j]}.\,b\,.\,\underline{0} \arrow[u, dashed, bend right, "(1)^{[j]}"'] & &
\end{tikzcd}$$

Figure 1: Time additivity and loop property for delay transitions

In the following we consider only *reachable processes*, whose set we denote by $\mathbb{P}_{\rm t}$, which include processes from which a computation can start, i.e., standard forward processes, as well as processes that can be derived from the previous ones via finitely many applications of the semantic rules in Tables 4 and 5. Note that processes like $(1)^{[i]}.\,(1)^{[i]}.\,(1)\,.\,\underline{0}$ and $((1)^{[i]}.\,\underline{0})\,\#\,((1)^{[j]}.\,\underline{0})$ where $\# \in \{+, \|_L\}$ and $i \neq j$ are not in $\mathbb{P}_{\rm t}$.

We conclude by showing the validity of time determinism and time additivity. Since Delay1 creates a fresh key (whereas Delay1$^\bullet$ uses the existing one), time determinism holds in the forward direction up to the keys associated with the considered delay. This is formalized via syntactical substitutions of delay keys; $R\{^j/_i\}$ stands for the process obtained from R by replacing key i with key j. As for time additivity, two distinct processes may be reached in the forward direction because elapsed delays are kept within processes. Dually, in the backward direction, the starting processes may be different. This is illustrated below.

Example 3.2. The delay transitions of the labeled transition system in Figure 1 are built as follows:

- The first 1-delay forward transition on the left is obtained from rules TAdd2 and TCho1, with the reverse rules being employed for the corresponding backward transition.
- The second 1-delay forward transition on the left is obtained from rules Delay3 and TCho1, with the reverse rules being employed for the corresponding backward transition.
- The 2-delay forward transition on the right is obtained from rules Delay1 and TCho1, with the reverse rules being employed for the corresponding backward transition.
- No combination of rules yields a 2-delay backward transition on the left from $\langle 1^i\rangle\,.\,(1)^{[j]}.\,a\,.\,\underline{0} + \langle 1^i\rangle\,.\,(1)^{[j]}.\,b\,.\,\underline{0}$ to $(2)\,.\,a\,.\,\underline{0} + (2)\,.\,b\,.\,\underline{0}$, i.e., no backward merging is possible after a forward splitting.
- No combination of rules yields a 1-delay backward transition from $(2)^{[k]}.\,a\,.\,\underline{0} + (2)^{[k]}.\,b\,.\,\underline{0}$ on the right to $\langle 1^i\rangle\,.\,(1)\,.\,a\,.\,\underline{0} + \langle 1^i\rangle\,.\,(1)\,.\,b\,.\,\underline{0}$ on the left, i.e., no backward splitting is possible.

The two consecutive 1-delay forward transitions on the left and the 2-delay forward transition on the right depart from the same process but reach two different processes. In contrast, the two consecutive 1-delay backward transitions on the left and the 2-delay backward transition on the right reach the same process although they depart from two different processes.

The labeled transition system would be similar if the initial process were $(1)\,.\,(1)\,.\,a\,.\,\underline{0} + (1)\,.\,(1)\,.\,b\,.\,\underline{0}$: on the left $\langle 1^i\rangle$ would become $(1)^{[i]}$ while on the right $(2)^{[k]}.\,a\,.\,\underline{0} + (2)^{[k]}.\,b\,.\,\underline{0}$ would become $(1)^{[i]}.\,(1)^{[j]}.\,a\,.\,\underline{0} + (1)^{[i]}.\,(1)^{[j]}.\,b\,.\,\underline{0}$, which would already be present on the left (i.e., there would be 3 states in lieu of 4). The conceptual difference is that, in the forward direction, instead of a delay splitting from $(2)\,.\,a\,.\,\underline{0} + (2)\,.\,b\,.\,\underline{0}$ – the two consecutive 1-delay forward transitions on the left – we would have a delay merging via rule TAdd1 from $(1)\,.\,(1)\,.\,a\,.\,\underline{0} + (1)\,.\,(1)\,.\,b\,.\,\underline{0}$ – the 2-delay forward transition on the right. ■

Proposition 3.3. [time determinism] Let $R, S_1, S_2 \in \mathbb{P}_{\mathrm{t}}$, $n \in \mathbb{N}_{>0}$, $i_1, i_2 \in \mathcal{K}$, and $j \in \mathcal{K}$ be not occurring associated with past delays in S_1 and S_2. Then:

- If $R \xrightarrow{(n)^{[i_1]}}_{\mathtt{d}} S_1$ and $R \xrightarrow{(n)^{[i_2]}}_{\mathtt{d}} S_2$, then $S_1\{^j/_{i_1}\} = S_2\{^j/_{i_2}\}$.
- If $R \overset{(n)^{[i_1]}}{\dashrightarrow}_{\mathtt{d}} S_1$ and $R \overset{(n)^{[i_2]}}{\dashrightarrow}_{\mathtt{d}} S_2$, then $i_1 = i_2$ and $S_1 = S_2$.

Proof A straightforward consequence of rules Delay1 (which generates a transition for each possible key), TCho1, and TCoo in the forward direction and their reverse variants in the backward direction. ■

Proposition 3.4. [time additivity] Let $R, R', S, S' \in \mathbb{P}_{\mathrm{t}}$, $n, h, m_1, \dots, m_h \in \mathbb{N}_{>0}$ be such that $\sum_{1 \leq l \leq h} m_l = n$, and $i, i_1, \dots, i_h \in \mathcal{K}$. Then:

- $R \xrightarrow{(n)^{[i]}}_{\mathtt{d}} S$ iff $R \xrightarrow{(m_1)^{[i_1]}}_{\mathtt{d}} \dots \xrightarrow{(m_h)^{[i_h]}}_{\mathtt{d}} S'$.
- $R \overset{(n)^{[i]}}{\dashrightarrow}_{\mathtt{d}} S$ iff $R' \overset{(m_1)^{[i_1]}}{\dashrightarrow}_{\mathtt{d}} \dots \overset{(m_h)^{[i_h]}}{\dashrightarrow}_{\mathtt{d}} S$.

Proof A straightforward consequence of rules TAdd1 and TAdd2 in the forward direction and their reverse variants in the backward direction. ■

3.2. Use Case: A Timeout Mechanism

Timeout mechanisms are crucial for ensuring system responsiveness and avoiding deadlocks in distributed environments and fault-tolerant systems. Industrially relevant programming languages for developing distributed software, such as Erlang, Elixir, and Akka, often utilize timeout primitives to prevent operations from becoming indefinitely blocked.

Following [51] we can define a timeout operator Timeout(F, G, t). It allows process F to communicate with the environment within t time units. After this time has passed and F has not communicated yet, process G takes control. The operator can be rendered in RTPC as Timeout$(F, G, t) = F + (t) \,.\, \tau \,.\, G$ under maximal progress, i.e., when τ is urgent while all the other actions are lazy.

As a matter of fact, the Erlang programming language provides a timeout facility on blocking receive. For example, let us consider the following snippet of Erlang code in which two processes A and B execute in parallel:

```
1 process_A() ->
2   receive
3     X -> handleMsg()
4     after 50 ->
5       handleTimeout()
6   end end.
7 process_B(Pid) ->
8   timer:sleep(50),
9   Pid! Msg  end.
10
11 PidA = spawn(?MODULE, process_A, []),
12 spawn(?MODULE, process_B, [PidA]).
```

Process A (lines 1–6) awaits a message from the environment (e.g., from process B); if a message is received within 50 ms, then process A calls function handleMsg(), otherwise it calls function handleTimeout(). Process B (lines 7–9) sleeps for 50 ms and then sends a message to Pid (e.g., the identifier of process A).

The translation into RTPC of the code for the two processes is as follows:

$$\begin{array}{rcl} A & = & \textsc{Timeout}(msg \,.\, F, G, 50) \quad = \quad msg \,.\, F + (50) \,.\, \tau \,.\, G \\ B & = & (50) \,.\, msg \,.\, \underline{0} \end{array}$$

where F encodes handleMsg() and G encodes handleTimeout(). If we run the two processes in parallel (mimicking lines 11–12), we might have the following forward execution under maximal progress:

$$\begin{array}{rl} A \parallel_{\{msg\}} B & \xrightarrow{(50)^{[i]}}_{\mathtt{d}} \quad (\langle 50^i \rangle \,.\, msg \,.\, F + (50)^{[i]} . \, \tau \,.\, G) \parallel_{\{msg\}} ((50)^{[i]} . \, msg \,.\, \underline{0}) \\ & \xrightarrow{\tau[j]}_{\mathtt{a}} \quad (\langle 50^i \rangle \,.\, msg \,.\, F + (50)^{[i]} . \, \tau[j] \,.\, G) \parallel_{\{msg\}} ((50)^{[i]} . \, msg \,.\, \underline{0}) \end{array}$$

at which point G takes over because B delayed the execution of *msg* and τ was urgent. If process B wants to revert its behavior, e.g., by going back by 50 ms, it cannot do it alone as it has to wait for process A to first undo $\tau[j]$ and then undo $\langle 50^i \rangle$ on the left branch and $(50)^{[i]}$ on the right branch together with B. This is clearly a causally consistent backward computation as no new process is encountered along the way.

$$\begin{tikzcd}
 & b \,.\, a \,.\, \underline{0} & & & a \,.\, (n) \,.\, \underline{0} \\
b[j] \,.\, a \,.\, \underline{0} \arrow[ur, dashed, "{b[j]}"] \arrow[r, "{a[i]}"] & b[j] \,.\, a[i] \,.\, \underline{0} & & a[i] \,.\, (n) \,.\, \underline{0} \arrow[ur, dashed, "{a[i]}"] \arrow[r, "{(n)^{[j]}}"] & a[i] \,.\, (n)^{[j]} . \, \underline{0}
\end{tikzcd}$$

Figure 2: Examples of conflicting transitions: conditions 1 (left) and 5 (right)

3.3. Causal Reversibility

As for the loop property, Proposition 2.2 smoothly extends to delay transitions because, as can be seen in Figure 1, a backward merging after a forward splitting and a backward splitting are not possible.

Proposition 3.5. [loop property] Let $R, S \in \mathbb{P}_{\rm t}$ and $\ell \in \mathcal{L}_{\rm t}$. Then $R \xrightarrow{\ell}_{\rm t} S$ iff $S \overset{\ell}{\dashrightarrow}_{\rm t} R$.

Proof By induction on the depth of the derivation of either transition and then by case analysis on the last applied rule, as each direct rule in Tables 4 and 5 has a corresponding reverse rule in the same tables and vice versa. ■

As for paths, when ω is a forward or backward path we denote by $\mathtt{time}(\omega)$ the duration of ω, i.e., the sum of the labels of its delay transitions. With respect to Section 2.2, the syntax of process contexts is enriched with the following case:

$$(n)^{[i]} . \mathcal{C}$$

and the definition of the causal set $\mathtt{cau}(R, i)$ for $R \in \mathbb{P}_{\rm t}$ and action key $i \in \mathcal{K}$ is extended as follows:

$$\mathtt{cau}(\delta(n, j) \,.\, R, i) \;=\; \mathtt{cau}(R, i)$$

with no need to add j to $\mathtt{cau}(R, i)$ due to conditions 5 and 6 of the forthcoming Definition 3.6, which can be expressed that way as time passing is identical across all alternative and parallel subprocesses.

With respect to Definition 2.3, the notion of conflicting transitions has four further conditions specific to this timed setting. The first one (number 3 below) views as conflictual two coinitial delay transitions regardless of their directions, provided that they are labeled with different delays when going in the same direction. The second one (number 4 below) considers as conflictual a forward action transition and a forward delay transition that are coinitial, whereas a similar situation cannot show up in the backward direction because, if a delay can be undone in all subprocesses, then no action can be undone (both cases will be illustrated in Example 3.12). The third one (number 5 below) regards as conflictual a forward delay transition and a backward action transition that are coinitial and originated from the same subprocess. In the fourth one (number 6 below) the action transition is forward and the delay transition is backward. Figure 2 illustrates the first and the fifth cases.

Definition 3.6. [conflicting and concurrent transitions] Two coinitial transitions θ_1 and θ_2 from a process $R \in \mathbb{P}_{\rm t}$ are *in conflict* if one of the following conditions holds, otherwise they are said to be *concurrent*:

1. $\theta_1 : R \xrightarrow{a[i]}_{\mathtt{a}} S_1$ and $\theta_2 : R \overset{b[j]}{\dashrightarrow}_{\mathtt{a}} S_2$ with $j \in \mathtt{cau}(S_1, i)$.

2. $R = \mathcal{C}[F_1 + F_2]$ with θ_k deriving from $F_k \xrightarrow{a_k[i_k]}_{\mathtt{a}} S_k$ for $k = 1, 2$.

3. $\theta_1 : R \overset{(n)^{[i]}}{\longmapsto}_{\mathtt{d}} S_1$ and $\theta_2 : R \overset{(m)^{[j]}}{\longmapsto}_{\mathtt{d}} S_2$ with $n \neq m$ when going in the same direction.

4. $\theta_1 : R \xrightarrow{a[i]}_{\mathtt{a}} S_1$ and $\theta_2 : R \xrightarrow{(m)^{[j]}}_{\mathtt{d}} S_2$.

5. $R = \mathcal{C}[a[i] \,.\, (n) \,.\, F]$ with θ_1 deriving from $(n) \,.\, F \xrightarrow{(m)^{[j]}}_{\mathtt{d}} S_1$ and θ_2 deriving from $a[i] \,.\, (n) \,.\, F \overset{a[i]}{\dashrightarrow}_{\mathtt{a}} S_2$.

6. $R = \mathcal{C}[\delta(n, i) \,.\, a \,.\, F]$ with θ_1 deriving from $a \,.\, F \xrightarrow{a[j]}_{\mathtt{a}} S_1$ and θ_2 deriving from $\delta(n, i) \,.\, a \,.\, F \overset{(n)^{[i]}}{\dashrightarrow}_{\mathtt{d}} S_2$.

■

The proof of the square property in this time setting is an extension of the one in Lemma 2.4.

Lemma 3.7. [square property] Let $\theta_1 : R \xmapsto{\ell_1}_{\mathrm{t}} S_1$ and $\theta_2 : R \xmapsto{\ell_2}_{\mathrm{t}} S_2$ be two coinitial transitions from a process $R \in \mathbb{P}_{\mathrm{t}}$. If θ_1 and θ_2 are concurrent, then there exist two cofinal transitions $\theta'_2 : S_1 \xmapsto{\ell_2}_{\mathrm{t}} S$ and $\theta'_1 : S_2 \xmapsto{\ell_1}_{\mathrm{t}} S$ with $S \in \mathbb{P}_{\mathrm{t}}$.

Proof The proof is by case analysis on the directions of θ_1 and θ_2. We distinguish three cases according to whether the two transitions are both forward, both backward, or one forward and the other backward:

- If θ_1 and θ_2 are both forward, there are three subcases:
  - If their labels are actions, since θ_1 and θ_2 are concurrent, by virtue of condition 2 of Definition 3.6 the two forward transitions cannot originate from a choice operator. They must thus be generated by a parallel composition, but not through rule Coo because θ_1 and θ_2 must have different keys and hence cannot synchronize. Without loss of generality, we assume that $R = R_1 \parallel_L R_2$ with $R_1 \xrightarrow{a[i]}_{\mathsf{a}} S_1$, $R_2 \xrightarrow{b[j]}_{\mathsf{a}} S_2$, $a, b \notin L$, and $i \neq j$. By applying rule Par we have that $R_1 \parallel_L R_2 \xrightarrow{a[i]}_{\mathsf{a}} S_1 \parallel_L R_2 \xrightarrow{b[j]}_{\mathsf{a}} S_1 \parallel_L S_2$ as well as $R_1 \parallel_L R_2 \xrightarrow{b[j]}_{\mathsf{a}} R_1 \parallel_L S_2 \xrightarrow{a[i]}_{\mathsf{a}} S_1 \parallel_L S_2$.
  - If their labels are delays, by virtue of condition 3 of Definition 3.6 the two forward transitions cannot be concurrent, hence this subcase does not apply.
  - If one label is an action and the other is a delay, by virtue of condition 4 of Definition 3.6 the two forward transitions cannot be concurrent, hence this subcase does not apply either.
- If θ_1 and θ_2 are both backward, there are again three subcases:
  - The first one is similar to the first one of the previous case, with the two coinitial transitions not originating from a choice operator because they go backward. Rule Par$^\bullet$ is then employed.
  - The second one does not apply because we are considering coinitial transitions that go backward.
  - The third one does not apply either because no action can be undone if a delay can be undone in all subprocesses.
- If θ_1 is forward and θ_2 is backward, there are once more three subcases:
  - If their labels are actions, since θ_1 and θ_2 are concurrent, by virtue of condition 1 of Definition 3.6 it holds that θ_2 cannot remove any cause of θ_1. Since any subprocess of a choice or a parallel composition cannot perform both a forward transition and a backward transition without preventing the backward one from removing a cause of the forward one, and in the case of a choice only one of the two subprocesses can perform transitions after the choice has been made (as would be in our case in which we consider a backward transition), without loss of generality we assume that $R = R_1 \parallel_L R_2$ with $R_1 \xrightarrow{a[i]}_{\mathsf{a}} S_1$, $R_2 \overset{b[j]}{\dashrightarrow}_{\mathsf{a}} S_2$, $a, b \notin L$, and $i \neq j$. By applying rules Par and Par$^\bullet$ we have that $R_1 \parallel_L R_2 \xrightarrow{a[i]}_{\mathsf{a}} S_1 \parallel_L R_2 \overset{b[j]}{\dashrightarrow}_{\mathsf{a}} S_1 \parallel_L S_2$ as well as $R_1 \parallel_L R_2 \overset{b[j]}{\dashrightarrow}_{\mathsf{a}} R_1 \parallel_L S_2 \xrightarrow{a[i]}_{\mathsf{a}} S_1 \parallel_L S_2$.
  - If their labels are delays, by virtue of condition 3 of Definition 3.6 the two transitions cannot be concurrent, hence this subcase does not apply.
  - If one label is an action and the other is a delay, since θ_1 and θ_2 are concurrent, by virtue of conditions 5 and 6 of Definition 3.6 the two transitions cannot be generated by the same subprocess. Hence, without loss of generality we assume that $R = R_1 \parallel_L R_2$ with $a \notin L$. There are two further subcases:

- $*$ Suppose that $R_1 \xrightarrow{(m)^{[j]}}_{\mathtt{d}} S_1$ and $R_2 \overset{a[i]}{\dashrightarrow}_{\mathtt{a}} S_2$. Under eagerness the latter transition would prevail at the level of R and hence this subcase would not apply. Under laziness the former transition would be possible at the level of R if R_2 could perform an identically labeled forward delay transition via idling. But then at the level of R we would have a forward $(m)^{[j]}$-transition and a backward $a[i]$-transition generated by the same subprocess R_2, hence again this subcase would not apply.
- $*$ If $R_1 \overset{(m)^{[j]}}{\dashrightarrow}_{\mathtt{a}} S_1$ and $R_2 \xrightarrow{a[i]}_{\mathtt{d}} S_2$ then we reason like in the previous subcase. ■

Unlike Lemma 2.5, backward transitions independence holds in this timed setting as long as at least one of the two coinitial backward transitions is not a delay transition.

Lemma 3.8. [backward transitions independence] Let $R \in \mathbb{P}_{\mathrm{t}}$. Then two coinitial backward transitions $\theta_1 : R \overset{\ell_1}{\dashrightarrow}_{\mathrm{t}} S_1$ and $\theta_2 : R \overset{\ell_2}{\dashrightarrow}_{\mathrm{t}} S_2$ are concurrent provided that at least one of them is not a delay transition.

Proof By Definition 3.6 it is not possible for two backward transitions to be in conflict except when they are both delay transitions (see condition 3). ■

As far as past well foundedness is concerned, with respect to Lemma 2.6 we observe that, under laziness and maximal progress, the adoption of the dynamic delay prefix $\langle n^i \rangle$ in rules Idling1, Idling2, Idling3, Idling1$^\bullet$, Idling2$^\bullet$, Idling3$^\bullet$ avoids the generation of backward self-loops, from which infinite sequences of backward transitions would be obtained.

Lemma 3.9. [past well foundedness] Let $R_0 \in \mathbb{P}_{\mathrm{t}}$. Then there is no infinite sequence of backward transitions such that $R_i \overset{\ell_i}{\dashrightarrow}_{\mathrm{t}} R_{i+1}$ for all $i \in \mathbb{N}$.

Proof By induction on $|\mathtt{key}_{\mathtt{a}}(R_0)| + \mathit{delay}(R_0)$, where $\mathit{delay}(R_0)$ is the number of past delays in R_0, i.e., the number of delays decorated with different keys. Indeed, every backward transition connects two different processes and hence decreases by one the (finite) number of past actions or delays in R_0. ■

With respect to Definition 2.7, the notion of causal equivalence in this timed setting cannot swap delay transitions due to time determinism. On the other hand, it can equate with the empty path any path made out of forward (resp. backward) delay transitions followed by a path made out of backward (resp. forward) delay transitions returning to the origin provided that both paths have the same duration, which is the third clause below. This can be seen by taking for example the delay path $\xrightarrow{(1)^{[i]}}_{\mathtt{d}} \xrightarrow{(1)^{[j]}}_{\mathtt{d}} \overset{(1)^{[j]}}{\dashrightarrow}_{\mathtt{d}} \overset{(1)^{[i]}}{\dashrightarrow}_{\mathtt{d}}$ on the left of Figure 1.

Definition 3.10. [causal equivalence] *Causal equivalence* $\asymp_{\mathrm{t}}$ is the smallest equivalence relation over paths traversing processes in $\mathbb{P}_{\mathrm{t}}$ that is closed under composition and satisfies the following clauses:

1. $\theta_1 \theta'_2 \asymp_{\mathrm{t}} \theta_2 \theta'_1$ for every two coinitial concurrent action transitions $\theta_1 : R \xmapsto{a[i]}_{\mathtt{a}} R_1$ and $\theta_2 : R \xmapsto{b[j]}_{\mathtt{a}} R_2$ and every two cofinal action transitions $\theta'_2 : R_1 \xmapsto{b[j]}_{\mathtt{a}} S$ and $\theta'_1 : R_2 \xmapsto{a[i]}_{\mathtt{a}} S$.
2. $\theta \overline{\theta} \asymp_{\mathrm{t}} \varepsilon$ and $\overline{\theta} \theta \asymp_{\mathrm{t}} \varepsilon$ for every transition θ.
3. $\omega_1 \overline{\omega_2} \asymp_{\mathrm{t}} \varepsilon$ and $\overline{\omega_2} \omega_1 \asymp_{\mathrm{t}} \varepsilon$ for every two coinitial and cofinal forward paths ω_1 and ω_2 with delay transitions only such that $\mathtt{time}(\omega_1) = \mathtt{time}(\omega_2)$. ■

Unlike Lemma 2.8, in this timed setting the parabolic lemma has to be proven directly. It does not stem from the square property and backward transitions independence as the latter does not hold for all coinitial backward transitions, as is the case with coinitial backward delay transitions.

Lemma 3.11. [parabolic lemma] For each path ω traversing processes in $\mathbb{P}_{\rm t}$ there exist two forward paths ω_1 and ω_2 traversing processes in $\mathbb{P}_{\rm t}$ such that $\omega \asymp_{\rm t} \overline{\omega_1}\omega_2$ and $|\omega_1| + |\omega_2| \leq |\omega|$.

Proof We proceed by induction on the number $d(\omega)$ of discording backward transitions, i.e., of backward transitions that are preceded by – but not necessarily adjacent to – forward transitions within path ω:

- If $d(\omega) = 0$ then ω is formed by a backward path followed by a forward one (each possibly empty).
- If $d(\omega) > 0$ then we take in ω the leftmost discording backward transition $\overline{\theta_2}$ such that there exists a subpath $\theta_1\overline{\theta_2}$ in ω with θ_1 and θ_2 being forward, so that $\omega = \overline{\omega_1}\omega'\theta_1\overline{\theta_2}\omega_2$ with ω_1 and ω' being forward and, like ω_2, possibly empty. If $\theta_1 = \theta_2$ then by applying clause 2 of Definition 3.10 we have that $\overline{\omega_1}\omega'\theta_1\overline{\theta_2}\omega_2 \asymp_{\rm t} \overline{\omega_1}\omega'\omega_2$ with $|\overline{\omega_1}\omega'\omega_2| < |\omega|$ and $d(\overline{\omega_1}\omega'\omega_2) < d(\omega)$ so that the induction hypothesis applies to $\overline{\omega_1}\omega'\omega_2$, otherwise there are three cases:
  - If θ_1 and $\overline{\theta_2}$ are two action transitions, then $\overline{\theta_1}$ and $\overline{\theta_2}$ are coinitial – because the target of θ_1 has to coincide with the source of $\overline{\theta_2}$ within ω – and hence concurrent by virtue of Lemma 3.8. By applying Lemma 3.7 we have that there exist two cofinal backward transitions $\overline{\theta_2'}$ and $\overline{\theta_1'}$ such that θ_1 and $\overline{\theta_2'}$ are coinitial and $\overline{\theta_2}$ and θ_1' are cofinal. Then, by applying clause 1 of Definition 3.10, we have that $\overline{\omega_1}\omega'\theta_1\overline{\theta_2}\omega_2 \asymp_{\rm t} \overline{\omega_1}\omega'\overline{\theta_2'}\theta_1'\omega_2$ with $|\overline{\omega_1}\omega'\overline{\theta_2'}\theta_1'\omega_2| = |\omega|$. If $\omega' = \varepsilon$ and $\omega_2 = \varepsilon$ or starts with a forward transition, then $d(\overline{\omega_1}\omega'\overline{\theta_2'}\theta_1'\omega_2) < d(\omega)$ so that the induction hypothesis applies to $\overline{\omega_1}\omega'\overline{\theta_2'}\theta_1'\omega_2$, otherwise we consider the next leftmost discording pair, of which $\overline{\theta_2'}$ or θ_1' is part depending on whether $\omega' \neq \varepsilon$ or not.
  - If θ_1 and $\overline{\theta_2}$ are two delay transitions, then the time elapsed with θ_1 is reverted due to time additivity by a sequence of backward delay transitions $\overline{\theta_2\omega_2'}$ where $\overline{\omega_2'}\omega_2'' = \omega_2$; that time cannot remain because, if n time units elapse forward with a single delay transition, we cannot go back by less than n time units (see Example 3.2). From $\texttt{time}(\theta_1) = \texttt{time}(\overline{\theta_2\omega_2'})$ and clause 3 of Definition 3.10 it follows that $\overline{\omega_1}\omega'\theta_1\overline{\theta_2\omega_2'}\omega_2'' \asymp_{\rm t} \overline{\omega_1}\omega'\omega_2''$ with $|\overline{\omega_1}\omega'\omega_2''| < |\omega|$ and $d(\overline{\omega_1}\omega'\omega_2'') < d(\omega)$ so that the induction hypothesis applies to $\overline{\omega_1}\omega'\omega_2''$.
  - If one is an action transition and the other is a delay transition, then we would have two coinitial backward transitions $\overline{\theta_1}$ and $\overline{\theta_2}$ of different type, but this is not possible because no action can be undone if a delay can be undone in all subprocesses and hence this case does not apply. ■

Example 3.12. If rules Idling1, Idling2, Idling3, Idling1$^\bullet$, Idling2$^\bullet$, Idling3$^\bullet$ did not respectively introduce and retract dynamic delay prefixes of the form $\langle n^i \rangle$, then the parabolic lemma would not hold.

Consider for instance the process $R = a \,.\, (n) \,.\, \underline{0} \,\|_\emptyset (n) \,.\, \underline{0}$. Under eagerness it can initially perform only $R \xrightarrow{a[i]}_{\tt a} a[i] \,.\, (n) \,.\, \underline{0} \,\|_\emptyset (n) \,.\, \underline{0} = S_1$ followed by $S_1 \xrightarrow{(n)^{[j]}}_{\tt d} a[i] \,.\, (n)^{[j]} .\, \underline{0} \,\|_\emptyset (n)^{[j]} .\, \underline{0} = S_1'$ up to delay splitting.

Under laziness it can also perform $R \xrightarrow{(n)^{[j]}}_{\tt d} \langle n^j \rangle \,.\, a \,.\, (n) \,.\, \underline{0} \,\|_\emptyset (n)^{[j]} .\, \underline{0} = S_2$ up to delay splitting because in R the execution of action a on the left can be postponed via Idling2 by as many time units as there are in delay (n) on the right where Delay1 applies, then TCoo is used. If we keep going forward, we obtain $S_2 \xrightarrow{a[i]}_{\tt a} \langle n^j \rangle \,.\, a[i] \,.\, (n) \,.\, \underline{0} \,\|_\emptyset (n)^{[j]} .\, \underline{0} = S_2'$ followed by $S_2' \xrightarrow{(n)^{[k]}}_{\tt d} \langle n^j \rangle \,.\, a[i] \,.\, (n)^{[k]} \,.\, \underline{0} \,\|_\emptyset (n)^{[j]} .\, \langle n^k \rangle \,.\, \underline{0} = S_2''$ up to delay splitting because in S_2' the subprocess $\underline{0}$ on the right can let via Idling1 as many time units pass as there are in delay (n) on the left. Note that the square property does not hold because S_1' is different from S_2' and S_2''. Indeed, the two initial transitions of R are in conflict according to condition 4 of Definition 3.6; moreover, unlike S_2', from S_1' delay $(n)^{[j]}$ can be undone whereas action $a[i]$ cannot.

When going backward, by virtue of the presence of dynamic delays $\langle n^j \rangle$ and $\langle n^k \rangle$ all the transitions above are undone in the reverse order and the same states are traversed thanks to Idling1$^\bullet$ and Idling2$^\bullet$. However, if dynamic delays were not adopted by the aforementioned idling rules, so that on the lazy side we would end up in $a[i] \,.\, (n)^{[k]} \,.\, \underline{0} \,\|_\emptyset (n)^{[j]} .\, \underline{0}$, then, observing that $(n)^{[k]}$ and $(n)^{[j]}$ have different keys and hence cannot be undone together via TCoo$^\bullet$, either $(n)^{[k]}$ is undone via Idling1$^\bullet$ applied to the subprocess $\underline{0}$ on the right, or $(n)^{[j]}$ is undone via Idling1$^\bullet$ applied to the subprocess $\underline{0}$ on the left. In the latter case, the new state $a[i] \,.\, (n)^{[k]} \,.\, \underline{0} \,\|_\emptyset (n) \,.\, \underline{0}$ is encountered, from which it is only possible to redo $(n)^{[j]}$ via Idling1

F, G	$::=$	$\underline{0} \mid a \,.\, F \mid F \,{}_p\!\oplus\, G \mid F + G \mid F \parallel_L G$
R, S	$::=$	$F \mid a[i] \,.\, R \mid R \,{}_{[i]p}\!\oplus\, S \mid R \,{}_p\!\oplus_{[i]}\, S \mid R + S \mid R \parallel_L S$

Table 6: Syntax of forward processes (top) and reversible processes (bottom) of RPPC ($. > {}_p\oplus > + > \parallel_L$)

applied to the subprocess $\underline{0}$ on the left; note that $(n)^{[k]}$ cannot be undone because $(n) \,.\, \underline{0}$ cannot let time pass backward. The presence of this new state would violate the parabolic lemma. In particular, the path traversing R and then S_2, S_2', and S_2'' without dynamic delays and finally the new state could not be causally equivalent to anyone composed of a backward path followed by a forward path. ■

Similar to Theorem 2.9, we conclude by showing that RTPC meets causal consistency.

Theorem 3.13. [causal consistency] Let ω_1 and ω_2 be two paths traversing processes in $\mathbb{P}_{\rm t}$. Then $\omega_1 \asymp_{\rm t} \omega_2$ iff ω_1 and ω_2 are both coinitial and cofinal.

Proof A consequence of past well foundedness and the parabolic lemma [46]. ■

4. Causal Reversibility in Nondeterministic and Probabilistic Process Calculi

In this section we present the syntax and semantics of RPPC – *Reversible Probabilistic Process Calculus*, which are inspired by those of probabilistic CCS [37, 2] and tailored for a reversible setting according to the static approach of [55] (Section 4.1). Then we illustrate how RPPC works by means of an example based on quantum teleportation (Section 4.2). Finally we investigate the causal reversibility of RPPC (Section 4.3).

4.1. Syntax and Semantics

The syntax of RPPC is shown in Table 6, featuring the probabilistic choice operator. While in [37] there is a strict alternation between nondeterministic processes like $N = \sum_{h \in H} a_h \,.\, P_h$ and probabilistic processes like $P = \bigoplus_{h \in H} \langle p_h \rangle \,.\, N_h$, as in [2] we stipulate for our more liberal syntax that probabilistic choices have to be *resolved before* nondeterministic ones when going forward – hence the latter have to be *revoked before* the former when going backward – so that every process either executes actions or makes probabilistic selections and no consecutive probabilistic transitions are possible. Thus, similar to time determinism, a probabilistic selection cannot resolve nondeterministic choices or decide who advances in a parallel composition. When the subprocesses of a nondeterministic choice or a parallel composition make a probabilistic selection, the corresponding probabilities are multiplied at the level of the entire process.

In the probabilistic choice $F \,{}_p\!\oplus\, G$, process F is selected with probability $p \in \mathbb{R}_{]0,1[}$ while process G is selected with probability $1 - p$. Its two reversible variants $R \,{}_{[i]p}\!\oplus\, S$ and $R \,{}_p\!\oplus_{[i]}\, S$ indicate that in the past a probabilistic selection was made in favor of the left or the right subprocess, respectively. Here communication key i is needed to avoid wrong pairings of probabilistic selections in the backward direction; keys could be omitted from probabilities in the absence of nondeterministic choice and parallel composition. Unlike the timed setting, where time passing is identical across all alternative and parallel subprocesses, probabilistic selections can be made independently by parallel subprocesses like in $a_1 \,.\, (b \,.\, \underline{0} \,{}_p\!\oplus\, c \,.\, \underline{0}) \parallel_\emptyset a_2 \,.\, (d \,.\, \underline{0} \,{}_q\!\oplus\, e \,.\, \underline{0})$, a fact that will require the definition of the set of probabilistic selection keys occurring in a process.

We denote by $\mathcal{P}_{\rm p}$ the set of processes generated by the second production in Table 6, while we use predicate std(_) to identify the standard forward processes that can be derived from the first production in the same table. For example, $a \,.\, (b \,.\, \underline{0} \,{}_{0.5}\!\oplus\, c \,.\, \underline{0})$ is a standard forward process that can execute action a and then probabilistically selects between doing action b or doing action c, while $a[i] \,.\, (b \,.\, \underline{0} \,{}_{[j]0.5}\!\oplus\, c \,.\, \underline{0})$ is a reversible process that can either undo the probabilistic selection in favor of b and then action a, or perform action b. Note that $a \,.\, (b \,.\, \underline{0} \,{}_{[j]0.5}\!\oplus\, c \,.\, \underline{0})$ and $a \,.\, (b[i] \,.\, \underline{0} \,{}_{0.5}\!\oplus\, c \,.\, \underline{0})$ are not in $\mathcal{P}_{\rm p}$ as a future action or probabilistic selection cannot precede a past one in the description of a process.

Let $\mathcal{A}_\mathcal{K} = \mathcal{A} \times \mathcal{K}$ and $\mathbb{R}_\mathcal{K} = \mathbb{R}_{]0,1[} \times \mathcal{K}$, with $\mathcal{L}_{\rm p} = \mathcal{A}_\mathcal{K} \cup \mathbb{R}_\mathcal{K}$ ranged over by ℓ. The semantics for RPPC is the labeled transition system $(\mathcal{P}_{\rm p}, \mathcal{L}_{\rm p}, \longmapsto_{\rm p})$. The transition relation $\longmapsto_{\rm p} \subseteq \mathcal{P}_{\rm p} \times \mathcal{L}_{\rm p} \times \mathcal{P}_{\rm p}$ is given by

(Act1)	$\dfrac{\mathtt{std}(R)}{a\,.\,R \xrightarrow{a[i]}_{\mathtt{a}} a[i]\,.\,R}$	(Act1$^\bullet$)	$\dfrac{\mathtt{std}(R)}{a[i]\,.\,R \overset{a[i]}{\dashrightarrow}_{\mathtt{a}} a\,.\,R}$
(Act2)	$\dfrac{R \xrightarrow{b[j]}_{\mathtt{a}} R' \quad j \neq i}{a[i]\,.\,R \xrightarrow{b[j]}_{\mathtt{a}} a[i]\,.\,R'}$	(Act2$^\bullet$)	$\dfrac{R \overset{b[j]}{\dashrightarrow}_{\mathtt{a}} R' \quad j \neq i}{a[i]\,.\,R \overset{b[j]}{\dashrightarrow}_{\mathtt{a}} a[i]\,.\,R'}$
(Act3)	$\dfrac{R \xrightarrow{b[j]}_{\mathtt{a}} R'}{R \,{}_{[i]p}{\oplus}\, S \xrightarrow{b[j]}_{\mathtt{a}} R' \,{}_{[i]p}{\oplus}\, S}$	(Act3$^\bullet$)	$\dfrac{R \overset{b[j]}{\dashrightarrow}_{\mathtt{a}} R'}{R \,{}_{[i]p}{\oplus}\, S \overset{b[j]}{\dashrightarrow}_{\mathtt{a}} R' \,{}_{[i]p}{\oplus}\, S}$
(Cho)	$\dfrac{R \xrightarrow{a[i]}_{\mathtt{a}} R' \quad \mathtt{npa}(S) \quad S \not\longrightarrow_{\mathrm{p}}}{R + S \xrightarrow{a[i]}_{\mathtt{a}} R' + S}$	(Cho$^\bullet$)	$\dfrac{R \overset{a[i]}{\dashrightarrow}_{\mathtt{a}} R' \quad \mathtt{npa}(S) \quad S \not\longrightarrow_{\mathrm{p}}}{R + S \overset{a[i]}{\dashrightarrow}_{\mathtt{a}} R' + S}$
(Par)	$\dfrac{R \xrightarrow{a[i]}_{\mathtt{a}} R' \quad a \notin L \quad i \notin \mathtt{key}_{\mathtt{a}}(S) \quad S \not\longrightarrow_{\mathrm{p}}}{R \parallel_L S \xrightarrow{a[i]}_{\mathtt{a}} R' \parallel_L S}$	(Par$^\bullet$)	$\dfrac{R \overset{a[i]}{\dashrightarrow}_{\mathtt{a}} R' \quad a \notin L \quad i \notin \mathtt{key}_{\mathtt{a}}(S) \quad S \not\longrightarrow_{\mathrm{p}}}{R \parallel_L S \overset{a[i]}{\dashrightarrow}_{\mathtt{a}} R' \parallel_L S}$
(Coo)	$\dfrac{R \xrightarrow{a[i]}_{\mathtt{a}} R' \quad S \xrightarrow{a[i]}_{\mathtt{a}} S' \quad a \in L}{R \parallel_L S \xrightarrow{a[i]}_{\mathtt{a}} R' \parallel_L S'}$	(Coo$^\bullet$)	$\dfrac{R \overset{a[i]}{\dashrightarrow}_{\mathtt{a}} R' \quad S \overset{a[i]}{\dashrightarrow}_{\mathtt{a}} S' \quad a \in L}{R \parallel_L S \overset{a[i]}{\dashrightarrow}_{\mathtt{a}} R' \parallel_L S'}$

Table 7: Operational semantic rules for RPPC action transitions

$\longmapsto_{\mathrm{p}} \, = \, \longrightarrow_{\mathrm{p}} \cup \dashrightarrow_{\mathrm{p}}$ where in turn the *forward transition relation* is given by $\longrightarrow_{\mathrm{p}} \, = \, \longrightarrow_{\mathtt{a}} \cup \longrightarrow_{\mathtt{p}}$ and the *backward transition relation* is given by $\dashrightarrow_{\mathrm{p}} \, = \, \dashrightarrow_{\mathtt{a}} \cup \dashrightarrow_{\mathtt{p}}$, with subscript $\mathtt{a}$ denoting action transitions and subscript $\mathtt{p}$ denoting probabilistic transitions. In the definitions of the transition relations, we make use of the set $\mathtt{key}_{\mathtt{a}}(R)$ of action keys and of the set $\mathtt{key}_{\mathtt{p}}(R)$ of probabilistic selection keys occurring in $R \in \mathcal{P}_{\mathrm{p}}$:

$$\begin{array}{rcl@{\qquad\qquad}rcl}
\mathtt{key}_{\mathtt{a}}(F) & = & \emptyset & \mathtt{key}_{\mathtt{p}}(F) & = & \emptyset \\
\mathtt{key}_{\mathtt{a}}(a[i]\,.\,R) & = & \{i\} \cup \mathtt{key}_{\mathtt{a}}(R) & \mathtt{key}_{\mathtt{p}}(a[i]\,.\,R) & = & \mathtt{key}_{\mathtt{p}}(R) \\
\mathtt{key}_{\mathtt{a}}(R \,{}_{[i]p}{\oplus}\, S) & = & \mathtt{key}_{\mathtt{a}}(R) & \mathtt{key}_{\mathtt{p}}(R \,{}_{[i]p}{\oplus}\, S) & = & \{i\} \cup \mathtt{key}_{\mathtt{p}}(R) \\
\mathtt{key}_{\mathtt{a}}(R \,{}_{p}{\oplus}_{[i]}\, S) & = & \mathtt{key}_{\mathtt{a}}(S) & \mathtt{key}_{\mathtt{p}}(R \,{}_{p}{\oplus}_{[i]}\, S) & = & \{i\} \cup \mathtt{key}_{\mathtt{p}}(S) \\
\mathtt{key}_{\mathtt{a}}(R + S) & = & \mathtt{key}_{\mathtt{a}}(R) \cup \mathtt{key}_{\mathtt{a}}(S) & \mathtt{key}_{\mathtt{p}}(R + S) & = & \mathtt{key}_{\mathtt{p}}(R) \cup \mathtt{key}_{\mathtt{p}}(S) \\
\mathtt{key}_{\mathtt{a}}(R \parallel_L S) & = & \mathtt{key}_{\mathtt{a}}(R) \cup \mathtt{key}_{\mathtt{a}}(S) & \mathtt{key}_{\mathtt{p}}(R \parallel_L S) & = & \mathtt{key}_{\mathtt{p}}(R) \cup \mathtt{key}_{\mathtt{p}}(S)
\end{array}$$

as well as of predicate $\mathtt{npa}(\_)$ to establish whether the considered process contains no past actions, whose definition is extended as follows with respect to Section 2.1:

$$\begin{array}{rcl}
\mathtt{npa}(R \oplus_{[i]p} S) & = & \mathtt{npa}(R) \\
\mathtt{npa}(R \oplus_{p[i]} S) & = & \mathtt{npa}(S)
\end{array}$$

The *action transition relations* $\longrightarrow_{\mathtt{a}} \subseteq \mathcal{P}_{\mathrm{p}} \times \mathcal{A}_{\mathcal{K}} \times \mathcal{P}_{\mathrm{p}}$ and $\dashrightarrow_{\mathtt{a}} \subseteq \mathcal{P}_{\mathrm{p}} \times \mathcal{A}_{\mathcal{K}} \times \mathcal{P}_{\mathrm{p}}$, whose union we denote by $\longmapsto_{\mathtt{a}}$, are the least relations respectively induced by the forward rules in the left part of Table 7 and by the backward rules in the right part of the same table. The only new rules with respect to Table 2 are Act3 and Act3$^\bullet$, along with their omitted symmetric variants for $R \,{}_{p}{\oplus}_{[i]}\, S$ where S has been selected with probability $1 - p$, which play a propagating role in a resolved probabilistic choice context analogous to the one of rules Act2 and Act2$^\bullet$ in an executed action context. Note that executed actions and resolved probabilistic choices are not required to exhibit different keys. Another difference with respect to Table 2 is the presence of clause $S \not\longrightarrow_{\mathrm{p}}$, meaning that S has no probabilistic forward transitions, in rules Cho and Par as well as their reverse variants. This stems from the fact that, when going forward, probabilistic choices have to be resolved before nondeterministic ones, including those arising from action interleaving.

The *probabilistic transition relations* $\longrightarrow_{\mathtt{p}} \subseteq \mathcal{P}_{\mathrm{p}} \times \mathbb{R}_{\mathcal{K}} \times \mathcal{P}_{\mathrm{p}}$ and $\dashrightarrow_{\mathtt{p}} \subseteq \mathcal{P}_{\mathrm{p}} \times \mathbb{R}_{\mathcal{K}} \times \mathcal{P}_{\mathrm{p}}$, whose union we denote by $\longmapsto_{\mathtt{p}}$, are the least relations respectively induced by the forward rules in the left part of Table 8

(PSEL1) $\dfrac{\mathtt{std}(R) \quad \mathtt{std}(S) \quad R \not\rightarrow_{\mathtt{p}}}{R\,{}_{p}{\oplus}\, S \xrightarrow{(p)^{[i]}}_{\mathtt{p}} R\,{}_{[i]p}{\oplus}\, S}$	(PSEL1$^{\bullet}$) $\dfrac{\mathtt{std}(R) \quad \mathtt{std}(S) \quad R \not\rightarrow_{\mathtt{p}}}{R\,{}_{[i]p}{\oplus}\, S \overset{(p)^{[i]}}{\dashrightarrow}_{\mathtt{p}} R\,{}_{p}{\oplus}\, S}$
(PSEL2) $\dfrac{\begin{array}{c}\mathtt{std}(R) \quad \mathtt{std}(S) \\ R \xrightarrow{(q)^{[j]}}_{\mathtt{p}} R' \quad i \notin \mathtt{key}_{\mathtt{p}}(R')\end{array}}{R\,{}_{p}{\oplus}\, S \xrightarrow{(p \cdot q)^{[i]}}_{\mathtt{p}} R'\,{}_{[i]p}{\oplus}\, S}$	(PSEL2$^{\bullet}$) $\dfrac{\begin{array}{c}\mathtt{std}(R') \quad \mathtt{std}(S) \\ R \overset{(q)^{[j]}}{\dashrightarrow}_{\mathtt{p}} R' \quad i \notin \mathtt{key}_{\mathtt{p}}(R)\end{array}}{R\,{}_{[i]p}{\oplus}\, S \overset{(p \cdot q)^{[i]}}{\dashrightarrow}_{\mathtt{p}} R'\,{}_{p}{\oplus}\, S}$
(PSEL3) $\dfrac{\neg\mathtt{npa}(R) \quad R \xrightarrow{(q)^{[j]}}_{\mathtt{p}} R' \quad j \neq i}{R\,{}_{[i]p}{\oplus}\, S \xrightarrow{(q)^{[j]}}_{\mathtt{p}} R'\,{}_{[i]p}{\oplus}\, S}$	(PSEL3$^{\bullet}$) $\dfrac{\neg\mathtt{npa}(R') \quad R \overset{(q)^{[j]}}{\dashrightarrow}_{\mathtt{p}} R' \quad j \neq i}{R\,{}_{[i]p}{\oplus}\, S \overset{(q)^{[j]}}{\dashrightarrow}_{\mathtt{p}} R'\,{}_{[i]p}{\oplus}\, S}$
(PSEL4) $\dfrac{R \xrightarrow{(q)^{[j]}}_{\mathtt{p}} R'}{a[i]\,.\,R \xrightarrow{(q)^{[j]}}_{\mathtt{p}} a[i]\,.\,R'}$	(PSEL4$^{\bullet}$) $\dfrac{R \overset{(q)^{[j]}}{\dashrightarrow}_{\mathtt{p}} R'}{a[i]\,.\,R \overset{(q)^{[j]}}{\dashrightarrow}_{\mathtt{p}} a[i]\,.\,R'}$
(PCHO1) $\dfrac{\begin{array}{c}R \xrightarrow{(p)^{[i]}}_{\mathtt{p}} R' \quad i \notin \mathtt{key}_{\mathtt{p}}(S) \quad S \not\rightarrow_{\mathtt{p}} \\ \mathtt{npa}(S)\end{array}}{R + S \xrightarrow{(p)^{[i]}}_{\mathtt{p}} R' + S}$	(PCHO1$^{\bullet}$) $\dfrac{\begin{array}{c}R \overset{(p)^{[i]}}{\dashrightarrow}_{\mathtt{p}} R' \quad i \notin \mathtt{key}_{\mathtt{p}}(S) \quad S \not\rightarrow_{\mathtt{p}} \\ \mathtt{npa}(S)\end{array}}{R + S \overset{(p)^{[i]}}{\dashrightarrow}_{\mathtt{p}} R' + S}$
(PCHO2) $\dfrac{R \xrightarrow{(p)^{[i]}}_{\mathtt{p}} R' \quad S \xrightarrow{(q)^{[i]}}_{\mathtt{p}} S'}{R + S \xrightarrow{(p \cdot q)^{[i]}}_{\mathtt{p}} R' + S'}$	(PCHO2$^{\bullet}$) $\dfrac{R \overset{(p)^{[i]}}{\dashrightarrow}_{\mathtt{p}} R' \quad S \overset{(q)^{[i]}}{\dashrightarrow}_{\mathtt{p}} S'}{R + S \overset{(p \cdot q)^{[i]}}{\dashrightarrow}_{\mathtt{p}} R' + S'}$
(PPAR) $\dfrac{R \xrightarrow{(p)^{[i]}}_{\mathtt{p}} R' \quad i \notin \mathtt{key}_{\mathtt{p}}(S) \quad S \not\rightarrow_{\mathtt{p}}}{R \parallel_L S \xrightarrow{(p)^{[i]}}_{\mathtt{p}} R' \parallel_L S}$	(PPAR$^{\bullet}$) $\dfrac{\begin{array}{c}R \overset{(p)^{[i]}}{\dashrightarrow}_{\mathtt{p}} R' \quad i \notin \mathtt{key}_{\mathtt{p}}(S) \quad S \not\rightarrow_{\mathtt{p}} \\ \mathtt{npa}(S) \vee \neg\mathtt{npa}(R)\end{array}}{R \parallel_L S \overset{(p)^{[i]}}{\dashrightarrow}_{\mathtt{p}} R' \parallel_L S}$
(PCOO) $\dfrac{R \xrightarrow{(p)^{[i]}}_{\mathtt{p}} R' \quad S \xrightarrow{(q)^{[i]}}_{\mathtt{p}} S'}{R \parallel_L S \xrightarrow{(p \cdot q)^{[i]}}_{\mathtt{p}} R' \parallel_L S'}$	(PCOO$^{\bullet}$) $\dfrac{R \overset{(p)^{[i]}}{\dashrightarrow}_{\mathtt{p}} R' \quad S \overset{(q)^{[i]}}{\dashrightarrow}_{\mathtt{p}} S'}{R \parallel_L S \overset{(p \cdot q)^{[i]}}{\dashrightarrow}_{\mathtt{p}} R' \parallel_L S'}$

Table 8: Operational semantic rules for RPPC probabilistic transitions

and by the backward rules in the right part of the same table. Although each backward probabilistic transition is labeled with the same probability as the corresponding forward transition, this probabilistic value is meaningful only in the forward direction. The rules in the two columns are perfectly symmetric apart from PPAR and PPAR$^{\bullet}$ as the latter features one additional premise.

Rules PSEL1 and PSEL2 handle probabilistic selections between standard processes. The former rule describes the case in which R has no probabilistic choices, hence the probability of selecting R is simply p. The latter rule describes the case in which R contains probabilistic choices, so that p is multiplied by the probability of selecting R'. To enable reversibility, in both rules the probability associated with the operator is decorated with a unique key and the subprocess that has not been selected is not discarded. Rules PSEL1$^{\bullet}$ and PSEL2$^{\bullet}$ are the backward counterparts. The symmetric variants for $R\,{}_{p}{\oplus}_{[i]}\, S$, in which S has been selected with probability $1 - p$, are omitted. For processes like $\underline{0}\,{}_{0.5}{\oplus}\, \underline{0}$ two distinct transitions $\underline{0}\,{}_{0.5}{\oplus}\, \underline{0} \xrightarrow{(0.5)^{[i]}}_{\mathtt{p}} \underline{0}\,{}_{[i]0.5}{\oplus}\, \underline{0}$ and $\underline{0}\,{}_{0.5}{\oplus}\, \underline{0} \xrightarrow{(0.5)^{[i]}}_{\mathtt{p}} \underline{0}\,{}_{0.5}{\oplus}_{[i]}\, \underline{0}$ are generated thanks to decorations in distinct positions within the two target processes, thus avoiding to resort to multisets of probabilistic transitions [37].

Rules PSEL3 and PSEL3$^{\bullet}$ propagate probabilistic selections in the context of resolved probabilistic choices followed by executed actions; their symmetric variants are omitted. Rules PSEL4 and PSEL4$^{\bullet}$ propagate probabilistic selections in the context of executed actions. We remind that executed actions and resolved probabilistic choices are allowed to share the same keys.

Rule PCHO1 represents a probabilistic selection made within a nondeterministic choice by R alone as $S \not\to_{\mathtt{p}}$, provided that S contains no past actions and key i does not occur in S, with PCHO1$^{\bullet}$ being its backward counterpart. Their symmetric variants are omitted. Remember that, as a consequence of the fact that probabilistic choices have to be resolved before nondeterministic choices when going forward, nondeterministic choices have to be revoked before probabilistic choices when going backward. For instance, $(a \,.\, \underline{0} \,{}_{p}\!\oplus\, b \,.\, \underline{0}) + c \,.\, \underline{0}$ first resolves the probabilistic choice thus becoming e.g. $(a \,.\, \underline{0} \,{}_{[i]p}\!\oplus\, b \,.\, \underline{0}) + c \,.\, \underline{0}$ and then can perform e.g. c thus evolving to $(a \,.\, \underline{0} \,{}_{[i]p}\!\oplus\, b \,.\, \underline{0}) + c[j] \,.\, \underline{0}$. Here the probabilistic choice cannot be revoked – otherwise $(a \,.\, \underline{0} \,{}_{p}\!\oplus\, b \,.\, \underline{0}) + c[j] \,.\, \underline{0}$ would be reached that cannot be encountered when going forward – thanks to the $\mathtt{npa}$ constraint. Note that rule CHO$^{\bullet}$ is applicable instead, because $a \,.\, \underline{0} \,{}_{[i]p}\!\oplus\, b \,.\, \underline{0} \not\to_{\mathtt{p}}$.

Rule PCHO2 expresses instead the fact that, since a probabilistic selection cannot decide which subprocess is chosen in a nondeterministic choice, if the two subprocesses of a nondeterministic choice make a probabilistic selection with the same key then the two selections are synchronized and the corresponding probabilities are multiplied. Rule PCHO2$^{\bullet}$ plays the corresponding role in the backward direction.

Likewise, rule PPAR represents a probabilistic selection made within a parallel composition by R alone as $S \not\to_{\mathtt{p}}$, provided that key i does not occur in S, with PPAR$^{\bullet}$ being its backward counterpart. Their symmetric variants are omitted. Note that PPAR$^{\bullet}$ additionally requires $\mathtt{npa}(S) \vee \neg\mathtt{npa}(R)$. The $\mathtt{npa}(S)$ part stems from the fact that nondeterministic choices, including those between two interleaving actions in a parallel composition, have to be revoked before probabilistic choices when going backward. As an example, $(a \,.\, \underline{0} \,{}_{p}\!\oplus\, b \,.\, \underline{0}) \parallel_{\emptyset} c \,.\, \underline{0}$ first resolves the probabilistic choice thus becoming e.g. $(a \,.\, \underline{0} \,{}_{[i]p}\!\oplus\, b \,.\, \underline{0}) \parallel_{\emptyset} c \,.\, \underline{0}$ and then can perform e.g. c thus evolving to $(a \,.\, \underline{0} \,{}_{[i]p}\!\oplus\, b \,.\, \underline{0}) \parallel_{\emptyset} c[j] \,.\, \underline{0}$. Here the probabilistic choice cannot be revoked – otherwise $(a \,.\, \underline{0} \,{}_{p}\!\oplus\, b \,.\, \underline{0}) \parallel_{\emptyset} c[j] \,.\, \underline{0}$ would be reached that cannot be encountered when going forward – thanks to the $\mathtt{npa}(S)$ part. The $\neg\mathtt{npa}(R)$ part is needed when two probabilistic choices are preceded by two interleaving actions like in $a_1 \,.\, (b \,.\, \underline{0} \,{}_{p}\!\oplus\, c \,.\, \underline{0}) \parallel_{\emptyset} a_2 \,.\, (d \,.\, \underline{0} \,{}_{q}\!\oplus\, e \,.\, \underline{0})$. While going forward this can reach e.g. $a_1[i_1] \,.\, (b \,.\, \underline{0} \,{}_{[j_1]p}\!\oplus\, c \,.\, \underline{0}) \parallel_{\emptyset} a_2[i_2] \,.\, (d \,.\, \underline{0} \,{}_{q}\!\oplus_{[j_2]}\, e \,.\, \underline{0})$, from which it must be possible to revoke either probabilistic choice.

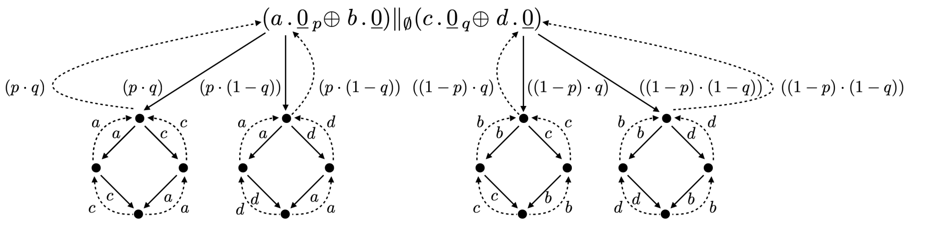
Rule PCOO represents instead the fact that, since a probabilistic selection cannot decide which subprocess advances in a parallel composition, if the two subprocesses of a parallel composition make a probabilistic selection with the same key then the two selections are synchronized and the corresponding probabilities are multiplied. Rule PCOO$^{\bullet}$ plays the corresponding role in the backward direction.

Example 4.1. To illustrate the need of communication keys also for probabilistic choices (new with respect to [55]), consider the standard forward process $(a \,.\, \underline{0} \,{}_{p}\!\oplus\, b \,.\, \underline{0}) \parallel_{\emptyset} (c \,.\, \underline{0} \,{}_{q}\!\oplus\, d \,.\, \underline{0})$, which may evolve to one of the following four reversible processes:

- $(a \,.\, \underline{0} \,{}_{[i]p}\!\oplus\, b \,.\, \underline{0}) \parallel_{\emptyset} (c \,.\, \underline{0} \,{}_{[i]q}\!\oplus\, d \,.\, \underline{0})$ with probability $p \cdot q$.
- $(a \,.\, \underline{0} \,{}_{[i]p}\!\oplus\, b \,.\, \underline{0}) \parallel_{\emptyset} (c \,.\, \underline{0} \,{}_{q}\!\oplus_{[i]}\, d \,.\, \underline{0})$ with probability $p \cdot (1 - q)$.
- $(a \,.\, \underline{0} \,{}_{p}\!\oplus_{[i]}\, b \,.\, \underline{0}) \parallel_{\emptyset} (c \,.\, \underline{0} \,{}_{[i]q}\!\oplus\, d \,.\, \underline{0})$ with probability $(1 - p) \cdot q$.
- $(a \,.\, \underline{0} \,{}_{p}\!\oplus_{[i]}\, b \,.\, \underline{0}) \parallel_{\emptyset} (c \,.\, \underline{0} \,{}_{q}\!\oplus_{[i]}\, d \,.\, \underline{0})$ with probability $(1 - p) \cdot (1 - q)$.

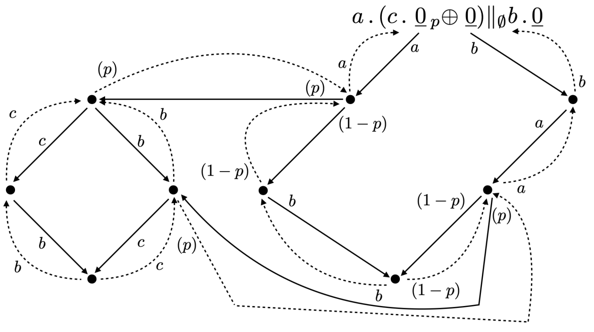
When going backward, in the absence of communication key i we could not know which subprocess of the probabilistic choice on the left of $\parallel_{\emptyset}$ was combined with which subprocess of the probabilistic choice on the right of $\parallel_{\emptyset}$.

It may be argued that what really matters is the position of the key with respect to the probabilistic parameter, hence a uniform decoration – e.g., ${}_{\langle p}\!\oplus$ and ${}_{p\rangle}\!\oplus$ – within every occurrence of the probabilistic choice operator should suffice. However, consider the standard forward process $a \,.\, (b \,.\, \underline{0} \,{}_{p}\!\oplus\, c \,.\, \underline{0}) \;\#\; (d \,.\, \underline{0} \,{}_{q}\!\oplus\, e \,.\, \underline{0})$ where $\# \in \{+, \parallel_{\emptyset}\}$. The only initial option is resolving the probabilistic choice on the right, thus reaching for instance $a \,.\, (b \,.\, \underline{0} \,{}_{p}\!\oplus\, c \,.\, \underline{0}) \;\#\; (d \,.\, \underline{0} \,{}_{[i]q}\!\oplus\, e \,.\, \underline{0})$. Then suppose that a is executed, which can only be followed by resolving the probabilistic choice on the left, which yields for instance $a[j] \,.\, (b \,.\, \underline{0} \,{}_{p}\!\oplus_{[k]}\, c \,.\, \underline{0}) \;\#\; (d \,.\, \underline{0} \,{}_{[i]q}\!\oplus\, e \,.\, \underline{0})$. Now, in the backward direction, the two probabilistic selections cannot be undone together – otherwise the



$(a\,.\,\underline{0}\,{}_p\!\oplus\, b\,.\,\underline{0}) \Vert_\emptyset (c\,.\,\underline{0}\,{}_q\!\oplus\, d\,.\,\underline{0})$
$(p\cdot q)$
$(p\cdot(1-q))$
$((1-p)\cdot q)$
$((1-p)\cdot(1-q))$

Figure 3: Labeled transition system underlying $(a\,.\,\underline{0}\,{}_p\!\oplus\, b\,.\,\underline{0}) \Vert_\emptyset (c\,.\,\underline{0}\,{}_q\!\oplus\, d\,.\,\underline{0})$



$a\,.\,(c\,.\,\underline{0}\,{}_p\!\oplus\,\underline{0}) \Vert_\emptyset b\,.\,\underline{0}$
(p)
$(1-p)$

Figure 4: Labeled transition system underlying $a\,.\,(c\,.\,\underline{0}\,{}_p\!\oplus\,\underline{0}) \Vert_\emptyset b\,.\,\underline{0}$

inconsistent $a[j]\,.\,(b\,.\,\underline{0}\,{}_p\!\oplus\, c\,.\,\underline{0}) \# (d\,.\,\underline{0}\,{}_q\!\oplus\, e\,.\,\underline{0})$ would be reached – because $i \neq k$ and hence PCho2$^\bullet$ and PCoo$^\bullet$ are not applicable. ■

In the following we consider only *reachable processes*, whose set we denote by $\mathbb{P}_\mathrm{p}$, which include processes from which a computation can start, i.e., standard forward processes, as well as processes that can be derived from the previous ones via finitely many applications of the semantic rules in Tables 7 and 8. Note that processes like $F_1\,{}_{[i]p}\!\oplus\, F_2\,{}_{[i]q}\!\oplus\, F_3$ and $(\underline{0}\,{}_p\!\oplus_{[i]}\, a[j]\,.\,F) \# (a[j]\,.\,(F_1\,{}_{[i]q}\!\oplus\, F_2))$ for $\# \in \{+, \Vert_{\{a\}}\}$ are not in $\mathbb{P}_\mathrm{p}$.

We conclude by discussing some properties specific to this probabilistic setting. First of all, we observe that forward probabilistic transitions across a parallel composition are combined with each other into single transitions by rule PCoo in the same way as nested probabilistic choices yield a single forward probabilistic transition by rule PSel2 and its symmetric variant. Every state reached by a forward probabilistic transition cannot thus have forward probabilistic transitions. Therefore, a labeled transition system cannot feature squares composed of forward probabilistic transitions, whereas it can contain squares made out of forward action transitions due to the interleaving of concurrent actions. As an example consider again process $(a\,.\,\underline{0} \oplus_p b\,.\,\underline{0}) \Vert_\emptyset (c\,.\,\underline{0} \oplus_q d\,.\,\underline{0})$, whose underlying labeled transition system is depicted in Figure 3 up to keys.

Secondly, when focusing only on the forward transition relation $\longrightarrow_\mathrm{p} = \longrightarrow_\mathrm{a} \cup \longrightarrow_\mathrm{p}$, the labeled transition system is consistent with the non-strict variant [54] of the alternating model [37]. This means that in every state all the forward transitions are either action transitions, in which case the state is nondeterministic, or probabilistic transitions, in which case the state is probabilistic. The alternation is not strict because, while a probabilistic state can reach via forward transitions only nondeterministic states due to rule PSel2 and its symmetric variant as well as rule PCoo, a nondeterministic state can reach via forward transitions either probabilistic states or other nondeterministic states. In contrast, a state can have both backward action transitions and backward probabilistic transitions. This may happen when transitions arising from probabilistic selections are involved in squares along with action transitions stemming from the interleaving of concurrent actions. For instance consider $a\,.\,(c\,.\,\underline{0}\,{}_p\!\oplus\,\underline{0}) \Vert_\emptyset b\,.\,\underline{0}$, whose underlying labeled transition system is depicted in Figure 4, and look at the bottommost state of the rectangle on the right, which features a backward b-transition as well as a backward $(1-p)$-transition, or the rightmost state of the square on the left, which features a backward b-transition as well as a backward p-transition.

The interested reader is referred to Appendix A to see how syntax and semantics change in the case of strict alternation between action transitions and probabilistic transitions.

4.2. Use Case: Quantum Teleportation

A quantum bit, or qubit, is a physical system with two basis states conventionally denoted by $|0\rangle$ and $|1\rangle$, which correspond to one-bit classical values. According to quantum theory, a general state of a quantum system is a superposition, i.e., linear combination, of basis states. For instance, a qubit has state $\alpha|0\rangle + \beta|1\rangle$ where $\alpha, \beta \in \mathbb{C}$ satisfy $|\alpha|^2 + |\beta|^2 = 1$. Measuring its value destroys superposition and yields 0 with probability $p = |\alpha|^2$ and 1 with probability $1 - p = |\beta|^2$, leaving the system in state $|0\rangle$ or $|1\rangle$ respectively.

We can thus see a qubit $|\psi\rangle$ as a process that can synchronize on an action z (for zero) or o (for one), which in RPPC can be modeled as follows when omitting trailing $\underline{0}$'s:

$$\begin{array}{rcl} |\psi\rangle(z, o, p) & = & |0\rangle(z) \;{}_{p}\!\oplus |1\rangle(o) \\ |0\rangle(z) & = & z \\ |1\rangle(o) & = & o \end{array}$$

This model is parametric with respect to the two actions z and o, corresponding to the two classical values 0 and 1, as well as the probability p of qubit $|\psi\rangle$ to be in state $|0\rangle$.

Tensor product is used to represent systems made out of several qubits. For example, a 2-qubit system has basis states $|00\rangle$, $|01\rangle$, $|10\rangle$, $|11\rangle$. Its general state is $\alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle$ where $|\alpha|^2 + |\beta|^2 + |\gamma|^2 + |\delta|^2 = 1$. Measuring the first (resp. second) qubit only yields 0 with probability $|\alpha|^2 + |\beta|^2$ (resp. $|\alpha|^2 + |\gamma|^2$) and 1 with probability $|\gamma|^2 + |\delta|^2$ (resp. $|\beta|^2 + |\delta|^2$), leaving the system in state $\frac{1}{\sqrt{|\alpha|^2+|\beta|^2}}(\alpha|00\rangle + \beta|01\rangle)$ or state $\frac{1}{\sqrt{|\gamma|^2+|\delta|^2}}(\gamma|10\rangle + \delta|11\rangle)$ (resp. in state $\frac{1}{\sqrt{|\alpha|^2+|\gamma|^2}}(\alpha|00\rangle + \gamma|10\rangle)$ or state $\frac{1}{\sqrt{|\beta|^2+|\delta|^2}}(\beta|01\rangle + \delta|11\rangle)$) respectively. If both qubits are measured simultaneously, then the possible results are 0, 1, 2, 3 with corresponding probabilities $|\alpha|^2$, $|\beta|^2$, $|\gamma|^2$, $|\delta|^2$ and states $|00\rangle$, $|01\rangle$, $|10\rangle$, $|11\rangle$, respectively.

In RPPC we can represent a 2-qubit entanglement as follows:

$$|\xi_1\xi_2\rangle(z_1, z_2, o_1, o_2, p_1, q_2, q_2') = z_1 \,.\, (z_2 \,.\, \mathit{zero} \;{}_{q_2}\!\oplus o_2 \,.\, \mathit{one}) \;{}_{p_1}\!\oplus o_1 \,.\, (z_2 \,.\, \mathit{two} \;{}_{q_2'}\!\oplus o_2 \,.\, \mathit{three})$$

where $p_1 \cdot q_2 = |\alpha|^2$, $p_1 \cdot (1 - q_2) = |\beta|^2$, $(1 - p_1) \cdot q_2' = |\gamma|^2$, $(1 - p_1) \cdot (1 - q_2') = |\delta|^2$.

Having recalled this, let us model one of the basic gates of quantum circuits. The Hadamard gate performs a rotation of a qubit, mapping the two qubit basis states $|0\rangle$ and $|1\rangle$ to two superposition states with equal probability of the computational basis stases $|0\rangle$ and $|1\rangle$ themselves. Its RPPC model is:

$$H(z_i, o_i, z_u, o_u) = z_i \,.\, (z_u \;{}_{\frac{1}{2}}\!\oplus o_u) + o_i \,.\, (z_u \;{}_{\frac{1}{2}}\!\oplus o_u)$$

where subscript i indicates the input of the gate while u its output. For example, we can express the application of the Hadamard gate to qubit $|0\rangle$, i.e., $|0\rangle(z) \parallel_L H(z, o, z_u, o_u)$ where $L = \{z, o\}$, as follows:

$$z \parallel_L z \,.\, (z_u \;{}_{\frac{1}{2}}\!\oplus o_u) + o \,.\, (z_u \;{}_{\frac{1}{2}}\!\oplus o_u) \xrightarrow{z[1]}_{\mathtt{a}} z[1] \parallel_L z[1] \,.\, (z_u \;{}_{\frac{1}{2}}\!\oplus o_u) + o \,.\, (z_u \;{}_{\frac{1}{2}}\!\oplus o_u)$$

Note that the forward behavior is given by $z_u \;{}_{\frac{1}{2}}\!\oplus o_u$, which is exactly the application of H to $|0\rangle$. Hadamard gate is a unitary operator, i.e., the application of a gate H to its inverse $H^\dagger$ gives back the original input. To some extent, we can model this behavior by exploiting the built-in reverse semantics of RPPC:

$$z[1] \parallel_L z[1] \,.\, (z_u \;{}_{\frac{1}{2}}\!\oplus o_u) + o \,.\, (z_u \;{}_{\frac{1}{2}}\!\oplus o_u) \overset{z[1]}{\dashrightarrow}_{\mathtt{a}} z \parallel_L z \,.\, (z_u \;{}_{\frac{1}{2}}\!\oplus o_u) + o \,.\, (z_u \;{}_{\frac{1}{2}}\!\oplus o_u)$$

A limitation is that we cannot have $HH = HH^\dagger$, but this is due to the fact our calculus is a general purpose one, hence it does not have built-in operators specific to quantum computing like [32, 73, 17, 18]. Also, it does not have operators to adjust amplitudes (e.g., probabilities) at run time. Nonetheless, as done in [70], with RPPC we can faithfully model the quantum teleportation protocol [5].

Quantum teleportation is a protocol that transfers an unknown quantum state from a location to another without physically moving the particle that carries it. It works by combining a shared entangled state between a sender, Alice, and a receiver, Bob, with classical communication. First, Alice and Bob share a maximally entangled pair of qubits, forming a quantum channel. Alice then performs a joint Bell-basis measurement on the qubit containing the unknown state and her half of the entangled pair, which projects Bob's qubit into a state related to the original one by a specific Pauli transformation. Alice sends the two-bit outcome of her measurement to Bob via a classical channel, then Bob uses this information to apply the correct transformation to his qubit, perfectly recovering the original state. The process destroys Alice's original qubit, ensuring compliance with the no-cloning theorem, and does not enable faster-than-light communication because the classical bits still travel on a classical communication channel, which means at or below the speed of light.

The setup of a maximal entangled state is done by a third participant, which we will call EPR from Einstein-Podolsky-Rosen. In RPPC we can model its behavior as follows:

$$\begin{array}{rcl} E & = & set \,.\, E_1 \\ E_1 & = & h \,.\, E_2 \\ E_2 & = & q_A \,.\, E_3 \\ E_3 & = & q_B \end{array}$$

EPR first sets the quantum state, then performs a Hadamard transformation on it, and finally sends the first qubit to Alice and the second one to Bob. Note that all these operations are abstracted away and described by using the corresponding actions *set*, h, q_A, and q_B.

Alice's behavior is represented as follows:

$$\begin{array}{rcl} A & = & q_A \,.\, A_1 \\ A_1 & = & cnot_{q_A} \,.\, A_2 \\ A_2 & = & h \,.\, A_3 \\ A_3 & = & z_1 \,.\, (z_2 \,.\, zero \;{}_{\frac{1}{2}}\!\oplus\; o_2 \,.\, one) \;{}_{\frac{1}{2}}\!\oplus\; o_1 \,.\, (z_1 \,.\, two \;{}_{\frac{1}{2}}\!\oplus\; o_2 \,.\, three) \end{array}$$

Alice first receives her qubit and then entangles it with the one she wants to send to Bob by applying the CNOT and Hadamard transformations. In standard quantum teleportation the sender's (Alice's) Bell-state measurement has four equally likely outcomes, corresponding to the fact that in A_3 each combination of the final states $|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$, i.e., *zero*, *one*, *two*, and *three*, has probability $\frac{1}{4}$ to happen.

After receiving his qubit from EPR, Bob awaits the measurement of Alice and, according to the received value, applies different transformation on his qubit:

$$\begin{array}{rcl} B & = & q_B \,.\, B_1 \\ B_1 & = & zero \,.\, id + one \,.\, x_G + two \,.\, z_G + three \,.\, z_G \,.\, x_G \end{array}$$

where *id* is the identity (i.e., empty operation), x_G represents the Pauli X-gate, and z_G the Pauli Z-gate.

We can thus model the entire quantum teleportation protocol as follows:

$$QTP \;=\; E \,\|_{L_1} (A \,\|_{L_2} B)$$

where $L_1 = \{q_A, q_B\}$ and $L_2 = \{one, two, three, four\}$. Initially it sets up the quantum state and communicate it to Alice and Bob as follows:

$$QTP \xrightarrow{set[1]}_{\mathtt{a}} \xrightarrow{h[2]}_{\mathtt{a}} \xrightarrow{q_A[3]}_{\mathtt{a}} \xrightarrow{q_B[4]}_{\mathtt{a}} set[1] \,.\, h[2] \,.\, q_A[3] \,.\, q_B[4] \,\|_{L_1} (q_A[3] \,.\, A_1 \,\|_{L_2} q_B[4] \,.\, B_1) = QTP_1$$

Bob is now blocked on receiving a value from Alice, hence the only one allowed to move forward is Alice:

$$QTP_1 \xrightarrow{cnot_{q_A}[5]}_{\mathtt{a}} \xrightarrow{h[6]}_{\mathtt{a}} set[1] \,.\, h[2] \,.\, q_A[3] \,.\, q_B[4] \,\|_{L_1} (q_A[3] \,.\, cnot_{q_A}[5] \,.\, h[6] \,.\, A_3 \,\|_{L_2} q_B[4] \,.\, B_1) = QTP_2$$

Once Alice has performed the measurement, she communicates the value to Bob; for instance, if the measurement returns state *one* then:

$$\begin{array}{l} QTP_2 \xrightarrow{(\frac{1}{4})^{[7]}}_{\mathtt{p}} \xrightarrow{z_1[8]}_{\mathtt{a}} \xrightarrow{o_2[9]}_{\mathtt{a}} set[1] \,.\, h[2] \,.\, q_A[3] \,.\, q_B[4] \,\|_{L_1} \\ \quad (q_A[3] \,.\, cnot_{q_A}[5] \,.\, h[6] \,.\, (z_1[8] \,.\, (z_2 \,.\, zero \;{}_{\frac{1}{2}}\!\oplus_{[7]}\; o_2[9] \,.\, one) \;{}_{[7]\frac{1}{2}}\!\oplus\; o_1 \,.\, (z_1 \,.\, two \;{}_{\frac{1}{2}}\!\oplus\; o_2 \,.\, three)) \,\|_{L_2} \\ \qquad q_B[4] \,.\, B_1) = QTP_3 \end{array}$$

at which point Alice and Bob can synchronize according to the aforementioned measurement:

$$\begin{array}{l} QTP_3 \xrightarrow{one[10]}_{\mathtt{a}} \xrightarrow{x_G[11]}_{\mathtt{a}} set[1] \,.\, h[2] \,.\, q_A[3] \,.\, q_B[4] \,\|_{L_1} \\ \quad (q_A[3] \,.\, cnot_{q_A}[5] \,.\, h[6] \,.\, (z_1[8] \,.\, (z_2 \,.\, zero \;{}_{\frac{1}{2}}\!\oplus_{[7]}\; o_2[9] \,.\, one[10]) \;{}_{[7]\frac{1}{2}}\!\oplus\; o_1 \,.\, (z_1 \,.\, two \;{}_{\frac{1}{2}}\!\oplus\; o_2 \,.\, three)) \,\|_{L_2} \\ \qquad q_B[4] \,.\, (zero \,.\, id + one[10] \,.\, x_G[11] + two \,.\, z_G + three \,.\, x_G \,.\, z_G)) = QTP_4 \end{array}$$

We conclude by recalling that the measurement process and the communication channel may be faulty. If Bob detects corrupted classical bits, the RPPC semantics allows the protocol to get back to the state where the wrong bits have been calculated (QTP_2) or transmitted (not modeled above). Since we will prove that RPPC meets causal reversibility, a new state is reached that is guaranteed to be correct, in the sense that it could have been reached by a forward computation in the case that the corruption had not happened.

4.3. Causal Reversibility

As for the loop property, Proposition 2.2 extends to probabilistic transitions as can be seen in Figures 3 and 4, but the proof has to take into account that PPar and PPar$^\bullet$ are not perfectly symmetric.

Proposition 4.2. [loop property] Let $R, S \in \mathbb{P}_{\rm p}$ and $\ell \in \mathcal{L}_{\rm p}$. Then $R \xrightarrow{\ell}_{\rm p} S$ iff $S \overset{\ell}{\dashrightarrow}_{\rm p} R$.

Proof By induction on the depth of the derivation of either transition and then by case analysis on the last applied rule, as each direct rule in Tables 7 and 8 has a corresponding reverse rule in the same tables and vice versa. To be precise, PPar and PPar$^\bullet$ are not perfectly symmetric as the latter features one additional premise, hence we develop the proof in the case that $R \xrightarrow{\ell}_{\rm p} S$ with PPar being the last applied rule (the case about $S \overset{\ell}{\dashrightarrow}_{\rm p} R$ and PPar$^\bullet$ is simpler as PPar premises are less demanding).

Consider $R_1 \parallel_L R_2$ and let $R_1 \xrightarrow{(p)^{[i]}}_{\rm p} S_1$ with $i \notin \mathtt{key}_{\mathtt{p}}(R_2)$ and $R_2 \not\rightarrow_{\rm p}$, so that the application of PPar yields $R_1 \parallel_L R_2 \xrightarrow{(p)^{[i]}}_{\rm p} S_1 \parallel_L R_2$. By applying the induction hypothesis to $R_1 \xrightarrow{(p)^{[i]}}_{\rm p} S_1$ we obtain $S_1 \overset{(p)^{[i]}}{\dashrightarrow}_{\rm p} R_1$. In order to apply PPar$^\bullet$ to $S_1 \parallel_L R_2$, we also need to have $\mathtt{npa}(R_2) \vee \neg\mathtt{npa}(S_1)$. This is indeed the case because otherwise $\neg\mathtt{npa}(R_2) \wedge \mathtt{npa}(S_1)$ would hold instead, hence $\neg\mathtt{npa}(R_2) \wedge \mathtt{npa}(R_1)$ would hold too meaning that the probabilistic selection in R_1, which is not preceded by any executed action, has been resolved after (instead of before) the execution of an action in R_2. ■

With respect to Section 2.2, the syntax of process contexts is enriched with the following two cases:

$$\mathcal{C}\ {}_{[i]p}\!\oplus R \mid R\ {}_{p}\!\oplus_{[i]} \mathcal{C}$$

and the definition of causal set $\mathtt{cau}(R, i)$ for $R \in \mathbb{P}_{\rm p}$ and action or probabilistic selection key $i \in \mathcal{K}$ under the assumption $\mathtt{key}_{\mathtt{a}}(R) \cap \mathtt{key}_{\mathtt{p}}(R) = \emptyset$ is extended as follows due to condition 1 of the forthcoming Definition 4.3:

$$\begin{array}{rcl}
\mathtt{cau}(F, i) & = & \emptyset \\
\mathtt{cau}(a[j] \,.\, R, i) & = & \left\{ \begin{array}{ll} \{j\} \cup \mathtt{cau}(R, i) & \text{if } j \neq i \text{ and } i \in \mathtt{key}_{\mathtt{a}}(R) \cup \mathtt{key}_{\mathtt{p}}(R) \\ \emptyset & \text{otherwise} \end{array} \right. \\
\mathtt{cau}(R\ {}_{[j]p}\!\oplus S, i) & = & \left\{ \begin{array}{ll} \{j\} \cup \mathtt{cau}(R, i) & \text{if } j \neq i \text{ and } i \in \mathtt{key}_{\mathtt{a}}(R) \cup \mathtt{key}_{\mathtt{p}}(R) \\ \emptyset & \text{otherwise} \end{array} \right. \\
\mathtt{cau}(R\ {}_{p}\!\oplus_{[j]} S, i) & = & \left\{ \begin{array}{ll} \{j\} \cup \mathtt{cau}(S, i) & \text{if } j \neq i \text{ and } i \in \mathtt{key}_{\mathtt{a}}(S) \cup \mathtt{key}_{\mathtt{p}}(S) \\ \emptyset & \text{otherwise} \end{array} \right. \\
\mathtt{cau}(R + S, i) & = & \mathtt{cau}(R, i) \cup \mathtt{cau}(S, i) \\
\mathtt{cau}(R \parallel_L S, i) & = & \mathtt{cau}(R, i) \cup \mathtt{cau}(S, i)
\end{array}$$

With respect to Definition 2.3, the notion of conflicting transitions generalizes the first condition to probabilistic transitions (with abuse of notation, the key is made explicit next to ℓ below) and has one further condition (number 3 below) that is the probabilistic counterpart of the second one, as the probabilistic choice operator replaces the nondeterministic one. Since probabilistic choices have to be resolved before nondeterministic ones and the latter have to be revoked before the former, there can never be conflicts between action transitions and probabilistic transitions. To be precise, as shown in Figure 4 there can be coinitial backward transitions such that some are action transitions and the others are probabilistic transitions, but they are originated by distinct subprocesses composed in parallel.

Definition 4.3. [conflicting and concurrent transitions] Two coinitial transitions θ_1 and θ_2 from a process $R \in \mathbb{P}_{\rm p}$ are *in conflict* if one of the following conditions holds, otherwise they are said to be *concurrent*:

1. $\theta_1 : R \xrightarrow{\ell_1[i]}_{\rm p} S_1$ and $\theta_2 : R \overset{\ell_2[j]}{\dashrightarrow}_{\rm p} S_2$ with $j \in \mathtt{cau}(S_1, i)$.

2. $R = \mathcal{C}[F_1 + F_2]$ with θ_k deriving in R from $F_k \xrightarrow{a_k[i_k]}_{\mathtt{a}} S_k$ for $k = 1, 2$.

3. $R = \mathcal{C}[F_1\ {}_{p}\!\oplus F_2]$ with θ_k deriving in R from $F_k \xrightarrow{(p_k)^{[i_k]}}_{\rm p} S_k$ for $k = 1, 2$. ■

The proof of the square property in this probabilistic setting is an extension of the one in Lemma 2.4. In particular, we have to deal with the fact that probabilistic choices take precedence over nondeterministic ones in the forward direction. Hence, even if from a process there are two transitions coming from the two subprocesses of a parallel composition, we have to determine whether either reached process can directly perform the other transition to close the square, or it has to first resolve probabilistic choices. Once such choices have been resolved, the remaining transition can be done thus closing the square. For example, consider $a\,.\,(c\,.\,\underline{0}\,{}_p\!\oplus\,\underline{0}) \,\|_\emptyset\, b\,.\,\underline{0}$ depicted in Figure 4. It can initially perform an a-action and a b-action, but if a is done then a probabilistic choice is enabled. Hence, before doing b, the process has to first resolve the probabilistic choice and only then it can proceed with b. On the other hand, if the process does b first, then it can immediately do a, but in order to reach the same process as the path starting with a it also has to resolve the probabilistic choice. As an example of analogous though larger square, consider the one originated from $a_1\,.\,(b\,.\,\underline{0}\,{}_p\!\oplus\, c\,.\,\underline{0}) \,\|_\emptyset\, a_2\,.\,(d\,.\,\underline{0}\,{}_q\!\oplus\, e\,.\,\underline{0})$, where both probabilistic choices are preceded by a non-synchronizing action. These cases are respectively handled by the second and the third clauses below.

Lemma 4.4. [square property] Let $\theta_1 : R \xmapsto{\ell_1}_{\mathrm{p}} S_1$ and $\theta_2 : R \xmapsto{\ell_2}_{\mathrm{p}} S_2$ be two coinitial transitions from a process $R \in \mathbb{P}_{\mathrm{p}}$. If θ_1 and θ_2 are concurrent, then one of the following clauses holds:

1. If $S_1 \not\rightarrow_{\mathrm{p}}$ and $S_2 \not\rightarrow_{\mathrm{p}}$ then there exist two cofinal transitions $\theta_2' : S_1 \xmapsto{\ell_2}_{\mathrm{p}} S$ and $\theta_1' : S_2 \xmapsto{\ell_1}_{\mathrm{p}} S$ with $S \in \mathbb{P}_{\mathrm{p}}$.

2. If $S_1 \rightarrow_{\mathrm{p}}$ and $S_2 \not\rightarrow_{\mathrm{p}}$ then there exist two cofinal paths $\omega_2' : S_1 \xmapsto{\ell_p}_{\mathrm{p}} S_1' \xmapsto{\ell_2}_{\mathrm{p}} S$ and $\omega_1' : S_2 \xmapsto{\ell_1}_{\mathrm{p}} S_2' \xmapsto{\ell_p}_{\mathrm{p}} S$ with $S_1', S_2', S \in \mathbb{P}_{\mathrm{p}}$.

3. If $S_1 \rightarrow_{\mathrm{p}}$ and $S_2 \rightarrow_{\mathrm{p}}$ then there exist two cofinal paths $\omega_2' : S_1 \xmapsto{\ell_p}_{\mathrm{p}} S_p^1 \xmapsto{\ell_2}_{\mathrm{p}} S_q^1 \xmapsto{\ell_q}_{\mathrm{p}} S$ and $\omega_1' : S_2 \xmapsto{\ell_q}_{\mathrm{p}} S_q^2 \xmapsto{\ell_1}_{\mathrm{p}} S_p^2 \xmapsto{\ell_p}_{\mathrm{p}} S$ with $S_p^1, S_p^2, S_q^1, S_q^2, S \in \mathbb{P}_{\mathrm{p}}$.

Proof The proof is by case analysis on the direction of θ_1 and θ_2. We distinguish three cases according to whether the two transitions are both forward, both backward, or one forward and the other backward:

- If θ_1 and θ_2 are both forward, there are three subcases:
  - If their labels are actions, since θ_1 and θ_2 are concurrent, by virtue of condition 2 of Definition 4.3 the two forward transitions cannot originate from a nondeterministic choice operator. They must thus be generated by a parallel composition, but not through rule Coo because θ_1 and θ_2 must have different keys and hence cannot synchronize. Without loss of generality, we assume that $R = R_1 \,\|_L\, R_2$ with $R_1 \xrightarrow{a[i]}_{\mathtt{a}} S_1$, $R_2 \xrightarrow{b[j]}_{\mathtt{a}} S_2$, $a, b \notin L$, and $i \neq j$. There are three further subcases:
    * If neither S_1 nor S_2 has a probabilistic transition, then by applying rule Par we have that $R_1 \,\|_L\, R_2 \xrightarrow{a[i]}_{\mathtt{a}} S_1 \,\|_L\, R_2 \xrightarrow{b[j]}_{\mathtt{a}} S_1 \,\|_L\, S_2$ as well as $R_1 \,\|_L\, R_2 \xrightarrow{b[j]}_{\mathtt{a}} R_1 \,\|_L\, S_2 \xrightarrow{a[i]}_{\mathtt{a}} S_1 \,\|_L\, S_2$, hence clause 1 holds.
    * If S_1 has a probabilistic transition – say $S_1 \xrightarrow{(p)^{[k]}}_{\mathrm{p}} S_1'$ – whereas S_2 has not, the corresponding probabilistic choice has to be resolved before letting R_2 do its b-transition. By applying rules Par and PPar we have that $R_1 \,\|_L\, R_2 \xrightarrow{a[i]}_{\mathtt{a}} S_1 \,\|_L\, R_2 \xrightarrow{(p)^{[k]}}_{\mathrm{p}} S_1' \,\|_L\, R_2$. Since S_1' cannot have a probabilistic transition, by applying rule Par we have that $S_1' \,\|_L\, R_2 \xrightarrow{b[j]}_{\mathtt{a}} S_1' \,\|_L\, S_2$. On the other hand, by applying rules Par twice and PPar we have that $R_1 \,\|_L\, R_2 \xrightarrow{b[j]}_{\mathtt{a}} R_1 \,\|_L\, S_2 \xrightarrow{a[i]}_{\mathtt{a}} S_1 \,\|_L\, S_2 \xrightarrow{(p)^{[k]}}_{\mathrm{p}} S_1' \,\|_L\, S_2$, hence clause 2 holds.

        * If both S_1 and S_2 have a probabilistic transition – say $S_1 \xrightarrow{(p)^{[k]}}_{\mathtt{p}} S'_1$ and $S_2 \xrightarrow{(q)^{[h]}}_{\mathtt{p}} S'_2$ – then similar to the previous subcase by applying rules Par, PPar, Par, and PPar we have that $R_1 \parallel_L R_2 \xrightarrow{a[i]}_{\mathtt{a}} S_1 \parallel_L R_2 \xrightarrow{(p)^{[k]}}_{\mathtt{p}} S'_1 \parallel_L R_2 \xrightarrow{b[j]}_{\mathtt{a}} S'_1 \parallel_L S_2 \xrightarrow{(q)^{[h]}}_{\mathtt{p}} S'_1 \parallel_L S'_2$ and $R_1 \parallel_L R_2 \xrightarrow{b[j]}_{\mathtt{a}} R_1 \parallel_L S_2 \xrightarrow{(q)^{[h]}}_{\mathtt{p}} R_1 \parallel_L S'_2 \xrightarrow{a[i]}_{\mathtt{a}} S_1 \parallel_L S'_2 \xrightarrow{(p)^{[k]}}_{\mathtt{p}} S'_1 \parallel_L S'_2$, hence clause 3 holds.

    – The subcase in which their labels are probabilities does not apply because the two forward transitions can originate from neither a choice operator by virtue of condition 3 of Definition 4.3 and rule PCho2, nor from a parallel composition due to rule PCoo.
    – The subcase in which one label is an action and the other is a probability does not apply either because a forward action transition cannot be generated until the probabilistic choice associated with the forward probabilistic transition has been resolved.

- If θ_1 and θ_2 are both backward, there are again three subcases:

    – If their labels are actions, the two coinitial transitions cannot originate from a nondeterministic choice operator because they go backward. They must thus be generated by a parallel composition, but not through rule Coo$^\bullet$ because θ_1 and θ_2 must have different keys and hence cannot synchronize. Without loss of generality, we assume that $R = R_1 \parallel_L R_2$ with $R_1 \overset{a[i]}{\dashrightarrow}_{\mathtt{a}} S_1$, $R_2 \overset{b[j]}{\dashrightarrow}_{\mathtt{a}} S_2$, $a, b \notin L$, and $i \neq j$; note that $S_1 \not\to_{\mathtt{p}}$ and $S_2 \not\to_{\mathtt{p}}$ because otherwise S_1 could not have a forward a-transition and S_2 could not have a forward b-transition. By applying rule Par$^\bullet$ we have that $R_1 \parallel_L R_2 \overset{a[i]}{\dashrightarrow}_{\mathtt{a}} S_1 \parallel_L R_2 \overset{b[j]}{\dashrightarrow}_{\mathtt{a}} S_1 \parallel_L S_2$ as well as $R_1 \parallel_L R_2 \overset{b[j]}{\dashrightarrow}_{\mathtt{a}} R_1 \parallel_L S_2 \overset{a[i]}{\dashrightarrow}_{\mathtt{a}} S_1 \parallel_L S_2$, hence clause 1 holds.
    – If their labels are probabilities, the two backward transitions can be generated only when $R = R_1 \parallel_L R_2$ with $R_1 \overset{(p)^{[i]}}{\dashrightarrow}_{\mathtt{p}} S_1$, $R_2 \overset{(q)^{[j]}}{\dashrightarrow}_{\mathtt{p}} S_2$, $i \neq j$, and rule PPar$^\bullet$ being applicable, which implies that $(\mathtt{npa}(R_2) \vee \neg \mathtt{npa}(R_1)) \wedge R_2 \not\to_{\mathtt{p}}$ holds for the first transition and $(\mathtt{npa}(R_1) \vee \neg \mathtt{npa}(R_2)) \wedge R_1 \not\to_{\mathtt{p}}$ holds for the second one; note that $S_1 \to_{\mathtt{p}}$ and $S_2 \to_{\mathtt{p}}$. The only situation in which both transitions can be generated is when $R_i \not\to_{\mathtt{p}}$ and $\neg \mathtt{npa}(R_i)$ – otherwise $i = j$ and rule PCoo$^\bullet$ would have been applied – for $i \in \{1, 2\}$. By applying rule PPar$^\bullet$ we have that $R_1 \parallel_L R_2 \overset{(p)^{[i]}}{\dashrightarrow}_{\mathtt{p}} S_1 \parallel_L R_2$ (resp. $R_1 \parallel_L R_2 \overset{(q)^{[j]}}{\dashrightarrow}_{\mathtt{p}} R_1 \parallel_L S_2$). From $\neg \mathtt{npa}(R_1)$ (resp. $\neg \mathtt{npa}(R_2)$) it follows that S_1 (resp. S_2) has at least one backward action transition, say $S_1 \overset{a[k]}{\dashrightarrow}_{\mathtt{a}} S'_1$ (resp. $S_2 \overset{b[h]}{\dashrightarrow}_{\mathtt{a}} S'_2$). By applying rule Par$^\bullet$ we have that $S_1 \parallel_L R_2 \overset{a[k]}{\dashrightarrow}_{\mathtt{a}} S'_1 \parallel_L R_2$ (resp. $R_1 \parallel_L S_2 \overset{b[h]}{\dashrightarrow}_{\mathtt{a}} R_1 \parallel_L S'_2$). By applying rules PPar$^\bullet$ and Par$^\bullet$ again we have that $S'_1 \parallel_L R_2 \overset{(q)^{[j]}}{\dashrightarrow}_{\mathtt{p}} S'_1 \parallel_L S_2 \overset{b[h]}{\dashrightarrow}_{\mathtt{a}} S'_1 \parallel_L S'_2$ (resp. $R_1 \parallel_L S'_2 \overset{(p)^{[i]}}{\dashrightarrow}_{\mathtt{p}} S_1 \parallel_L S'_2 \overset{a[k]}{\dashrightarrow}_{\mathtt{a}} S'_1 \parallel_L S'_2$), hence clause 3 holds.
    – If one label is an action and the other is a probability, the two backward transitions can be generated only when $R = R_1 \parallel_L R_2$ with $R_1 \overset{a[i]}{\dashrightarrow}_{\mathtt{a}} S_1$, $R_2 \overset{(p)^{[j]}}{\dashrightarrow}_{\mathtt{p}} S_2$, $R_1 \not\to_{\mathtt{p}}$ otherwise the backward transition of R_2 would not be possible, and $\neg \mathtt{npa}(R_2)$ – say $S_2 \overset{b[k]}{\dashrightarrow}_{\mathtt{a}} S'_2$ – in order for PPar$^\bullet$ to be applicable; note that $S_1 \not\to_{\mathtt{p}}$ – otherwise S_1 could not have a forward a-transition – and $S_2 \to_{\mathtt{p}}$. By applying rules Par$^\bullet$, PPar$^\bullet$, and Par$^\bullet$ we have that $R_1 \parallel_L R_2 \overset{a[i]}{\dashrightarrow}_{\mathtt{a}} S_1 \parallel_L R_2 \overset{(p)^{[j]}}{\dashrightarrow}_{\mathtt{p}} S_1 \parallel_L S_2 \overset{b[k]}{\dashrightarrow}_{\mathtt{a}} S_1 \parallel_L S'_2$, while by applying rules PPar$^\bullet$ and Par$^\bullet$ twice we have that $R_1 \parallel_L R_2 \overset{(p)^{[j]}}{\dashrightarrow}_{\mathtt{p}} R_1 \parallel_L S_2 \overset{b[k]}{\dashrightarrow}_{\mathtt{a}} R_1 \parallel_L S'_2 \overset{a[i]}{\dashrightarrow}_{\mathtt{a}} S_1 \parallel_L S'_2$, hence clause 2 holds.

- If θ_1 is forward and θ_2 is backward, there are once more three subcases:
  - If their labels are actions, since θ_1 and θ_2 are concurrent, by virtue of condition 1 of Definition 4.3 it holds that θ_2 cannot remove any cause of θ_1. Since any subprocess of a choice or a parallel composition cannot perform both a forward transition and a backward transition without preventing the backward one from removing a cause of the forward one, and in the case of a choice only one of the two subprocesses can perform transitions after the choice has been made (as would be in our case in which we consider a backward transition), without loss of generality we assume that $R = R_1 \,\|_L\, R_2$ with $R_1 \xrightarrow{a[i]}_{\mathtt{a}} S_1$, $R_2 \overset{b[j]}{\dashrightarrow}_{\mathtt{a}} S_2$, $a, b \notin L$, and $i \neq j$; note that $S_2 \not\rightarrow_{\mathtt{p}}$ because otherwise S_2 could not have a forward b-transition. There are two further subcases:
    - $*$ If S_1 does not have a probabilistic transition, by applying rules Par and Par$^\bullet$ we have that $R_1 \,\|_L\, R_2 \xrightarrow{a[i]}_{\mathtt{a}} S_1 \,\|_L\, R_2 \overset{b[j]}{\dashrightarrow}_{\mathtt{a}} S_1 \,\|_L\, S_2$, while by applying rules Par$^\bullet$ and Par we have that $R_1 \,\|_L\, R_2 \overset{b[j]}{\dashrightarrow}_{\mathtt{a}} R_1 \,\|_L\, S_2 \xrightarrow{a[i]}_{\mathtt{a}} S_1 \,\|_L\, S_2$, hence clause 1 holds.
    - $*$ If S_1 has a probabilistic transition, say $S_1 \xrightarrow{(p)^{[k]}}_{\mathtt{p}} S_1'$, then by applying rules Par and PPar we have that $R_1 \,\|_L\, R_2 \xrightarrow{a[i]}_{\mathtt{a}} S_1 \,\|_L\, R_2 \xrightarrow{(p)^{[k]}}_{\mathtt{p}} S_1' \,\|_L\, R_2$. Since S_1' cannot have a probabilistic transition, by applying rule Par$^\bullet$ we have that $S_1' \,\|_L\, R_2 \overset{b[j]}{\dashrightarrow}_{\mathtt{a}} S_1' \,\|_L\, S_2$. On the other hand, by applying rules Par$^\bullet$, Par, and PPar we have that $R_1 \,\|_L\, R_2 \overset{b[j]}{\dashrightarrow}_{\mathtt{a}} R_1 \,\|_L\, S_2 \xrightarrow{a[i]}_{\mathtt{a}} S_1 \,\|_L\, S_2 \xrightarrow{(p)^{[k]}}_{\mathtt{p}} S_1' \,\|_L\, S_2$, hence clause 2 holds.
  - The subcase in which their labels are probabilities does not apply because otherwise, due to Proposition 4.2, there would be two consecutive probabilistic transitions in the same direction.
  - If one label is an action and the other is a probability, the two transitions can be generated only when $R = R_1 \,\|_L\, R_2$ with $R_1 \xrightarrow{a[i]}_{\mathtt{a}} S_1$, $R_2 \overset{(p)^{[j]}}{\dashrightarrow}_{\mathtt{p}} S_2$ (note that $R_1 \not\rightarrow_{\mathtt{p}}$ so that the backward transition of R_2 is possible), and $\neg\mathtt{npa}(R_2)$ – say $S_2 \overset{b[k]}{\dashrightarrow}_{\mathtt{a}} S_2'$ – in order for PPar$^\bullet$ to be applicable; note that $S_2 \rightarrow_{\mathtt{p}}$. There are two further subcases:
    - $*$ If $S_1 \not\rightarrow_{\mathtt{p}}$ then by applying rules Par, PPar$^\bullet$, and Par$^\bullet$ we have that $R_1 \,\|_L\, R_2 \xrightarrow{a[i]}_{\mathtt{a}} S_1 \,\|_L\, R_2 \overset{(p)^{[j]}}{\dashrightarrow}_{\mathtt{p}} S_1 \,\|_L\, S_2 \overset{b[k]}{\dashrightarrow}_{\mathtt{a}} S_1 \,\|_L\, S_2'$, while by applying rules PPar$^\bullet$, Par$^\bullet$, and Par we have that $R_1 \,\|_L\, R_2 \overset{(p)^{[j]}}{\dashrightarrow}_{\mathtt{p}} R_1 \,\|_L\, S_2 \overset{b[k]}{\dashrightarrow}_{\mathtt{a}} R_1 \,\|_L\, S_2' \xrightarrow{a[i]}_{\mathtt{a}} S_1 \,\|_L\, S_2'$, hence clause 2 holds.
    - $*$ If $S_1 \rightarrow_{\mathtt{p}}$, say $S_1 \xrightarrow{(q)^{[h]}}_{\mathtt{p}} S_1'$, then by applying rules Par, PPar, PPar$^\bullet$, and Par$^\bullet$ we have that $R_1 \,\|_L\, R_2 \xrightarrow{a[i]}_{\mathtt{a}} S_1 \,\|_L\, R_2 \xrightarrow{(q)^{[h]}}_{\mathtt{p}} S_1' \,\|_L\, R_2 \overset{(p)^{[j]}}{\dashrightarrow}_{\mathtt{p}} S_1' \,\|_L\, S_2 \overset{b[k]}{\dashrightarrow}_{\mathtt{a}} S_1' \,\|_L\, S_2'$, while by applying rules PPar$^\bullet$, Par$^\bullet$, Par, and PPar we have that $R_1 \,\|_L\, R_2 \overset{(p)^{[j]}}{\dashrightarrow}_{\mathtt{p}} R_1 \,\|_L\, S_2 \overset{b[k]}{\dashrightarrow}_{\mathtt{a}} R_1 \,\|_L\, S_2' \xrightarrow{a[i]}_{\mathtt{a}} S_1 \,\|_L\, S_2' \xrightarrow{(q)^{[h]}}_{\mathtt{p}} S_1' \,\|_L\, S_2'$, hence clause 3 holds. ■

Backward transitions independence and past well foundedness are proven like in Lemmas 2.5 and 2.6 respectively.

Lemma 4.5. [backward transitions independence] Let $R \in \mathbb{P}_{\mathrm{p}}$. Then two coinitial backward transitions $\theta_1 : R \overset{\ell_1}{\dashrightarrow}_{\mathrm{p}} S_1$ and $\theta_2 : R \overset{\ell_2}{\dashrightarrow}_{\mathrm{p}} S_2$ are concurrent.

Proof By Definition 4.3 it is not possible for two backward transitions to be in conflict. ■

Lemma 4.6. [past well foundedness] Let $R_0 \in \mathbb{P}_{\mathrm{p}}$. Then there is no infinite sequence of backward transitions such that $R_i \overset{\ell_i}{\dashrightarrow}_{\mathrm{p}} R_{i+1}$ for all $i \in \mathbb{N}$.

Proof By induction on $|\mathtt{key}_{\mathtt{a}}(R_0)| + |\mathtt{key}_{\mathtt{p}}(R_0)|$, as every backward transition connects two different processes and hence decreases by one the (finite) number of past actions or probabilistic selections in R_0. ■

With respect to Definition 2.7, the notion of causal equivalence in this probabilistic setting cannot swap two concurrent forward action transitions (resp. backward probabilistic transitions) before all probabilistic choices (resp. nondeterministic choice) have been resolved (resp. revoked). More precisely, after opening a square, it may be the case that, in order to close it, the process has to first resolve or revoke some choices. This is rendered by the third and fourth clauses in the definition below. For example, if we consider again process $a\,.\,(c\,.\,\underline{0}\;{}_p\!\oplus\;\underline{0}) \parallel_\emptyset b\,.\,\underline{0}$ in Figure 4, we have that the forward action transitions a and b are concurrent and coinitial. If we take the left path, then after doing a the process has to resolve the probabilistic choice. Suppose it decides for the right branch, which is labeled with $(1-p)$. Then a process is reached in which b can be done. On the other hand, if we take the right path, we have that the process can do b followed by a but then again, in order to close the square, it has to resolve the probabilistic choice; if it decides for $(1-p)$ the same process as the left path is reached. Therefore the two paths can be considered as causally equivalent. Something similar happens in the larger square originating from $a_1\,.\,(b\,.\,\underline{0}\;{}_p\!\oplus\;c\,.\,\underline{0}) \parallel_\emptyset a_2\,.\,(d\,.\,\underline{0}\;{}_q\!\oplus\;e\,.\,\underline{0})$.

In the third clause below, although the labels are reminiscent of the order of action and probabilistic transitions in the case of a forward square, also the case of a backward square is included, which starts with a backward action transition and a backward probabilistic transition that are coinitial and concurrent. Likewise, in the fourth clause below, also the cases of a backward square and a mixed square are included. The former starts with two coinitial concurrent backward probabilistic transitions, while the latter starts with a forward/backward action transition and a backward/forward probabilistic transition that are coinitial and concurrent. The example processes are again $a\,.\,(c\,.\,\underline{0}\;{}_p\!\oplus\;\underline{0}) \parallel_\emptyset b\,.\,\underline{0}$ and $a_1\,.\,(b\,.\,\underline{0}\;{}_p\!\oplus\;c\,.\,\underline{0}) \parallel_\emptyset a_2\,.\,(d\,.\,\underline{0}\;{}_q\!\oplus\;e\,.\,\underline{0})$.

Definition 4.7. [causal equivalence] *Causal equivalence* $\asymp_{\mathrm{p}}$ *is the smallest equivalence relation over paths traversing processes in* $\mathbb{P}_{\mathrm{p}}$ *that is closed under composition and satisfies the following clauses:*

1. $\theta_1\theta_2' \asymp_{\mathrm{p}} \theta_2\theta_1'$ for every two coinitial concurrent action transitions $\theta_1 : R \xmapsto{a[i]}_{\mathtt{a}} R_1$ and $\theta_2 : R \xmapsto{b[j]}_{\mathtt{a}} R_2$ and every two cofinal action transitions $\theta_2' : R_1 \xmapsto{b[j]}_{\mathtt{a}} S$ and $\theta_1' : R_2 \xmapsto{a[i]}_{\mathtt{a}} S$.

2. $\theta\overline{\theta} \asymp_{\mathrm{p}} \varepsilon$ and $\overline{\theta}\theta \asymp_{\mathrm{p}} \varepsilon$ for every transition θ.

3. $\theta_1\theta_p\theta_2' \asymp_{\mathrm{p}} \theta_2\theta_1'\theta_p'$ for every two coinitial concurrent transitions $\theta_1 : R \xmapsto{\ell_1}_{\mathrm{p}} R_1$ and $\theta_2 : R \xmapsto{\ell_2}_{\mathrm{p}} R_2$, every two transitions $\theta_p : R_1 \xmapsto{\ell_p}_{\mathrm{p}} R_1'$ and $\theta_1' : R_2 \xmapsto{\ell_1}_{\mathrm{p}} R_2'$, and every two cofinal transitions $\theta_2' : R_1' \xmapsto{\ell_2}_{\mathrm{p}} S$ and $\theta_p' : R_2' \xmapsto{\ell_p}_{\mathrm{p}} S$.

4. $\theta_1\theta_p\theta_2'\theta_q' \asymp_{\mathrm{p}} \theta_2\theta_q\theta_1'\theta_p'$ for every two coinitial concurrent transitions $\theta_1 : R \xmapsto{\ell_1}_{\mathrm{p}} R_1$ and $\theta_2 : R \xmapsto{\ell_2}_{\mathrm{p}} R_2$, every two transitions $\theta_p : R_1 \xmapsto{\ell_p}_{\mathrm{p}} R_1'$ and $\theta_q : R_2 \xmapsto{\ell_q}_{\mathrm{p}} R_2'$, every two transitions $\theta_2' : R_1' \xmapsto{\ell_2}_{\mathrm{p}} R_1''$ and $\theta_1' : R_2' \xmapsto{\ell_1}_{\mathrm{p}} R_2''$, and every two cofinal transitions $\theta_q' : R_1'' \xmapsto{\ell_q}_{\mathrm{p}} S$ and $\theta_p' : R_2'' \xmapsto{\ell_p}_{\mathrm{p}} S$. ■

Unlike Lemma 2.8, in this probabilistic setting the parabolic lemma has to be proven directly because the square property of Lemma 4.4 and the causal equivalence of Definition 4.7 are extensions of those in [46] in which probabilistic transitions can be additionally present inside squares. They yield a significant modification of the original notion of square composed of 4 action transitions only, as here a square can also be composed of 6 or 8 transitions with 2 or 4 of them being probabilistic, respectively.

Another difference is that $|\omega_1| + |\omega_2| \leq |\omega|$ does not necessarily hold. If we consider again the rectangle on the right of Figure 4, the path ω starting from the leftmost state and composed of a forward b-transition followed by a backward $(1-p)$-transition is causally equivalent to $\overline{\omega_1}\omega_2$ where $\overline{\omega_1}$ is composed of a backward $(1-p)$-transition followed by a backward a-transition and ω_2 is composed of a forward b-transition followed by a forward a-transition. Indeed, $\overline{\omega_1}\omega_2 = \overline{(1-p)}\,\overline{a}\,b\,a \asymp_{\mathrm{p}} b\,\overline{(1-p)}\,\overline{a}\,a \asymp_{\mathrm{p}} b\,\overline{(1-p)}$ due to clauses 3 and 2 of Definition 4.7. Note that $|\omega_1| + |\omega_2| = 2 + 2 = 4 \not\leq 2 = |\omega|$.

Lemma 4.8. [parabolic lemma] For each path ω traversing processes in $\mathbb{P}_\mathrm{p}$ there exist two forward paths ω_1 and ω_2 traversing processes in $\mathbb{P}_\mathrm{p}$ such that $\omega \asymp_\mathrm{p} \overline{\omega_1}\omega_2$ and $|\omega_1| + |\omega_2| \leq k \cdot |\omega|$ with $k \in \{1, 2, 3\}$.

Proof We proceed by induction on the number $d(\omega)$ of discording backward transitions, i.e., of backward transitions that are preceded by – but not necessarily adjacent to – forward transitions within path ω:

- If $d(\omega) = 0$ then ω is formed by a backward path followed by a forward one (each possibly empty).
- If $d(\omega) > 0$ then we take in ω the leftmost discording backward transition $\overline{\theta_2}$ such that there exists a subpath $\theta_1\overline{\theta_2}$ in ω with θ_1 and θ_2 being forward, so that $\omega = \overline{\omega_1}\omega'\theta_1\overline{\theta_2}\omega_2$ with ω_1 and ω' being forward and, like ω_2, possibly empty. If $\theta_1 = \theta_2$ then by applying clause 2 of Definition 4.7 we have that $\overline{\omega_1}\omega'\theta_1\overline{\theta_2}\omega_2 \asymp_\mathrm{p} \overline{\omega_1}\omega'\omega_2$ with $|\overline{\omega_1}\omega'\omega_2| < |\omega|$ and $d(\overline{\omega_1}\omega'\omega_2) < d(\omega)$ so that the induction hypothesis applies to $\overline{\omega_1}\omega'\omega_2$, otherwise, after observing that $\overline{\theta_1}$ and $\overline{\theta_2}$ are coinitial – because the target of θ_1 has to coincide with the source of $\overline{\theta_2}$ within ω – and hence concurrent by virtue of Lemma 4.5, there are three cases:

  – If θ_1 and $\overline{\theta_2}$ are two action transitions, by applying clause 1 of Lemma 4.4 we have that there exist two cofinal backward transitions $\overline{\theta'_2}$ and $\overline{\theta'_1}$ such that θ_1 and $\overline{\theta'_2}$ are coinitial and $\overline{\theta_2}$ and θ'_1 are cofinal. Then, by applying clause 1 of Definition 4.7, we have that $\overline{\omega_1}\omega'\theta_1\overline{\theta_2}\omega_2 \asymp_\mathrm{p} \overline{\omega_1}\omega'\overline{\theta'_2}\theta'_1\omega_2$ with $|\overline{\omega_1}\omega'\overline{\theta'_2}\theta'_1\omega_2| = |\omega|$. If $\omega' = \varepsilon$ and $\omega_2 = \varepsilon$ or starts with a forward transition, then $d(\overline{\omega_1}\omega'\overline{\theta'_2}\theta'_1\omega_2) < d(\omega)$ so that the induction hypothesis applies to $\overline{\omega_1}\omega'\overline{\theta'_2}\theta'_1\omega_2$, otherwise we consider the next leftmost discording pair, of which $\overline{\theta'_2}$ or θ'_1 is part depending on whether $\omega' \neq \varepsilon$ or not.

  – If θ_1 and $\overline{\theta_2}$ are two probabilistic transitions, by applying clause 3 of Lemma 4.4 we have that there exist two cofinal backward paths $\overline{\omega'_2} = \overline{\theta_p}\overline{\theta'_2}\overline{\theta_q}$ and $\overline{\omega'_1} = \overline{\theta'_q}\overline{\theta'_1}\overline{\theta'_p}$ such that θ_1 and $\overline{\theta_p}$ are coinitial and $\overline{\theta_2}$ and θ'_q are cofinal. Then, by applying clauses 2 and 4 of Definition 4.7, we have that $\overline{\omega_1}\omega'\theta_1\overline{\theta_2}\omega_2 \asymp_\mathrm{p} \overline{\omega_1}\omega'\overline{\theta_p}\theta_p\theta_1\overline{\theta_2}\overline{\theta'_q}\theta'_q\omega_2 \asymp_\mathrm{p} \overline{\omega_1}\omega'\overline{\theta_p}\overline{\theta'_2}\overline{\theta_q}\theta'_p\theta'_1\theta'_q\omega_2$ with $|\overline{\omega_1}\omega'\overline{\theta_p}\overline{\theta'_2}\overline{\theta_q}\theta'_p\theta'_1\theta'_q\omega_2| \leq 3 \cdot |\omega|$. If $\omega' = \varepsilon$ and $\omega_2 = \varepsilon$ or starts with a forward transition, then $d(\overline{\omega_1}\omega'\overline{\theta_p}\overline{\theta'_2}\overline{\theta_q}\theta'_p\theta'_1\theta'_q\omega_2) < d(\omega)$ so that the induction hypothesis applies to $\overline{\omega_1}\omega'\overline{\theta_p}\overline{\theta'_2}\overline{\theta_q}\theta'_p\theta'_1\theta'_q\omega_2$, otherwise we consider the next leftmost discording pair, of which $\overline{\theta_p}$ or θ'_q is part depending on whether $\omega' \neq \varepsilon$ or not.

  – If one is an action transition and the other is a probabilistic transition, then we have two subcases:

    ∗ If θ_1 is a probabilistic transition and $\overline{\theta_2}$ is an action transition, by applying clause 2 of Lemma 4.4 we have that there exist two cofinal backward paths $\overline{\omega'_2} = \overline{\theta_p}\overline{\theta'_2}$ and $\overline{\omega'_1} = \overline{\theta'_1}\overline{\theta'_p}$ such that θ_1 and $\overline{\theta_p}$ are coinitial and $\overline{\theta_2}$ and θ'_1 are cofinal. Then, by applying clauses 2 and 3 of Definition 4.7, we have that $\overline{\omega_1}\omega'\theta_1\overline{\theta_2}\omega_2 \asymp_\mathrm{p} \overline{\omega_1}\omega'\overline{\theta_p}\theta_p\theta_1\overline{\theta_2}\omega_2 \asymp_\mathrm{p} \overline{\omega_1}\omega'\overline{\theta_p}\overline{\theta'_2}\theta'_p\theta'_1\omega_2$ – note that $\overline{\theta_p}$ does not change – with $|\overline{\omega_1}\omega'\overline{\theta_p}\overline{\theta'_2}\theta'_p\theta'_1\omega_2| \leq 2 \cdot |\omega|$. If $\omega' = \varepsilon$ and $\omega_2 = \varepsilon$ or starts with a forward transition, then $d(\overline{\omega_1}\omega'\overline{\theta_p}\overline{\theta'_2}\theta'_p\theta'_1\omega_2) < d(\omega)$ so that the induction hypothesis applies to $\overline{\omega_1}\omega'\overline{\theta_p}\overline{\theta'_2}\theta'_p\theta'_1\omega_2$, otherwise we consider the next leftmost discording pair, of which $\overline{\theta_p}$ or θ'_1 is part depending on whether $\omega' \neq \varepsilon$ or not.

    ∗ If θ_1 is an action transition and $\overline{\theta_2}$ is a probabilistic transition, by applying clause 2 of Lemma 4.4 we have that there exist two cofinal backward paths $\overline{\omega'_2} = \overline{\theta'_2}\overline{\theta_p}$ and $\overline{\omega'_1} = \overline{\theta'_p}\overline{\theta'_1}$ such that θ_1 and $\overline{\theta'_2}$ are coinitial and $\overline{\theta_2}$ and θ'_p are cofinal. Then, by applying clauses 2 and 3 of Definition 4.7, we have that $\overline{\omega_1}\omega'\theta_1\overline{\theta_2}\omega_2 \asymp_\mathrm{p} \overline{\omega_1}\omega'\theta_1\overline{\theta_2}\overline{\theta'_p}\theta'_p\omega_2 \asymp_\mathrm{p} \overline{\omega_1}\omega'\overline{\theta'_2}\overline{\theta_p}\theta'_1\theta'_p\omega_2$ – note that θ'_p does not change – with $|\overline{\omega_1}\omega'\overline{\theta'_2}\overline{\theta_p}\theta'_1\theta'_p\omega_2| \leq 2 \cdot |\omega|$. If $\omega' = \varepsilon$ and $\omega_2 = \varepsilon$ or starts with a forward transition, then $d(\overline{\omega_1}\omega'\overline{\theta'_2}\overline{\theta_p}\theta'_1\theta'_p\omega_2) < d(\omega)$ so that the induction hypothesis applies to $\overline{\omega_1}\omega'\overline{\theta'_2}\overline{\theta_p}\theta'_1\theta'_p\omega_2$, otherwise we consider the next leftmost discording pair, of which $\overline{\theta'_2}$ or θ'_p is part depending on whether $\omega' \neq \varepsilon$ or not. ■

Similar to Theorem 2.9, we conclude by showing that RPPC meets causal consistency.

Theorem 4.9. [causal consistency] Let ω_1 and ω_2 be two paths traversing processes in $\mathbb{P}_{\rm p}$. Then $\omega_1 \asymp_{\rm p} \omega_2$ iff ω_1 and ω_2 are both coinitial and cofinal.

Proof A consequence of past well foundedness and the parabolic lemma [46]. ■

5. Conclusions

In this paper we have investigated whether the causal reversibility techniques for nondeterministic process calculi of [55, 46], which are known to be applicable to stochastically timed process calculi too [9], scale to deterministic time and probabilities. Specifically, we have shown how to adapt them – in terms of process syntax, operational semantics, loop property, square property, backward transitions independence, past well foundedness, and parabolic lemma – in order to obtain causal consistency in nondeterministic process calculi enriched with deterministic time or probabilities. In both cases, it has been necessary to extend – with additional conditions specific to the considered quantitive aspects – the instrumental notions of conflicting/concurrent transitions and causal equivalence borrowed from [22].

In the case of the reversible version of a timed process calculus like temporal CCS [51], in which time passing is separated from action execution, we have addressed the three cases of eagerness, laziness, and maximal progress and shown how to achieve causal reversibility under time determinism and time additivity. This has required decorating elapsed delays with communication keys to ensure timed determinism as well as introducing dynamic elapsed delays for a correct handling of time additivity in addition to laziness and maximal progress. With respect to the reversible timed setting of [13], which builds on TPL [38] featuring laziness and unitary delays only, we have dealt with eagerness too and, most importantly, we have described time via arbitrary numeric delays subject to time additivity, which is technically more challenging. As a future investigation, we would like to address dense time.

In the case of the reversible version of a probabilistic process calculus like the one in [37, 2], in which action execution alternates with probabilistic selections, we have shown how to achieve causal reversibility under the constraint that probabilistic choices are resolved before nondeterministic ones in the forward direction and hence revoked after the latter in the backward direction. This has required decorating probabilistic selections with communication keys – with the notable difference with respect to delays that probabilistic selections can be made independently by distinct processes composed in parallel – as well as extending square property, causal equivalence, and parabolic lemma to situations in which more than four transitions, possibly of different kinds, are involved. As a future direction, similar to the reversible stochastically timed setting relying on continuous-time Markov chains of [9, 8], in this setting based on discrete-time Markov chains we would like to study time reversibility [39] and its possible relationships with causal reversibility.

For both cases, we plan to develop behavioral equivalences in the bisimulation style based on previous definitions and results in [24, 55, 9], which could lay the basis for the verification of reversible processes. Another interesting direction to pursue is causal reversibility in a nondeterministic process calculus extended with both deterministic time and probabilities.

Acknowledgments. This research has been supported by the PRIN 2020 project *NiRvAna – Noninterference and Reversibility Analysis in Private Blockchains*, the PRIN 2022 project *DeKLA – Developing Kleene Logics and Their Applications*, and the INdAM-GNCS 2024 project *MARVEL – Modelli Composizionali per l'Analisi di Sistemi Reversibili Distribuiti.*

Appendix A. Strictly Alternating RPPC

The syntax of the standard forward processes for RPPC$_{\text{sa}}$, a variant of RPPC complying with the strictly alternating model of [37], is as follows:

$$\begin{array}{rcl} N, M & ::= & \underline{0} \mid a \,.\, P \mid N + M \mid N \parallel_L M \\ P, Q & ::= & \bigoplus_{h \in H} \langle p_h \rangle N_h \mid P \parallel_L Q \end{array}$$

where the first line refers to nondeterministic processes while the second one refers to probabilistic processes. Moreover, H is a finite and non-empty index set, $p_h \in \mathbb{R}_{]0,1]}$ for all $h \in H$, and $\sum_{h \in H} p_h = 1$.

We observe that parallel composition applies to both nondeterministic processes and probabilistic processes, action prefix and nondeterministic choice characterize nondeterministic processes, and probabilistic choice characterizes probabilistic processes. The strict alternation arises from the fact that the continuation of an action prefix is a probabilistic process and the continuation of every summand within a probabilistic choice is a nondeterministic process.

The syntax of the corresponding reversible processes is the following:

$$\begin{array}{rcl} RN, RM & ::= & N \mid a[i] \,.\, RP \mid RN + RM \mid RN \parallel_L RM \\ RP, RQ & ::= & P \mid \langle p_z \rangle^{[i]} RN_z \bigoplus_{h \in H \setminus \{z\}} \langle p_h \rangle RN_h \mid RP \parallel_L RQ \end{array}$$

where $z \in H$ singles out the summand that has been selected in a probabilistic choice.

The related operational semantic rules are shown in Table A.9 for action transitions and in Table A.10 for probabilistic transitions. As can be noted, they are fewer than those in Tables 7 and 8, respectively, and also simpler because they use only `std` and $\mathtt{key}_{\mathtt{a}}$ and do not need negative premises on probabilistic transitions. We denote by $\mathbb{P}_{\text{p,sa}}$ the set of reachable processes.

The only subtlety is related to rules $\text{P}\small{\text{AR}}_{\text{sa}}$ and $\text{P}\small{\text{AR}}^{\bullet}_{\text{sa}}$. In the former rule, since RN can do an action and hence is nondeterministic, RM is nondeterministic too and hence has to be made probabilistic in the derivative process, which would simply be accomplished by transforming it into $\langle 1 \rangle RM$ if it were a standard forward process [37]. Function `mkpr` used in rule $\text{P}\small{\text{AR}}_{\text{sa}}$ is inductively defined over $\mathbb{P}_{\text{p,sa}}$ as follows:

$$\begin{array}{rcl} \mathtt{mkpr}(N) & = & \langle 1 \rangle N \\ \mathtt{mkpr}(a[i] \,.\, RP) & = & a[i] \,.\, \mathtt{mkpr}(RP) \\ \mathtt{mkpr}(RN + RM) & = & \left\{ \begin{array}{ll} \mathtt{mkpr}(RN) + RM & \text{if } \mathtt{std}(RM) \\ RN + \mathtt{mkpr}(RM) & \text{if } \mathtt{std}(RN) \end{array} \right. \\ \mathtt{mkpr}(RN \parallel_L RM) & = & \mathtt{mkpr}(RN) \parallel_L \mathtt{mkpr}(RM) \\ \mathtt{mkpr}(P) & = & P \\ \mathtt{mkpr}(\langle p_z \rangle^{[i]} RN_z \bigoplus_{h \in H \setminus \{z\}} \langle p_h \rangle RN_h) & = & \langle p_z \rangle^{[i]} \mathtt{mkpr}(RN_z) \bigoplus_{h \in H \setminus \{z\}} \langle p_h \rangle RN_h \\ \mathtt{mkpr}(RP \parallel_L RQ) & = & \mathtt{mkpr}(RP) \parallel_L \mathtt{mkpr}(RQ) \end{array}$$

In the latter rule, since RN can undo an action and hence is probabilistic, RM is probabilistic too and hence has to be made nondeterministic in the derivative process. Function `mknd` used in rule $\text{P}\small{\text{AR}}^{\bullet}_{\text{sa}}$ is inductively defined over over $\mathbb{P}_{\text{p,sa}}$ as follows by taking into account the clause $\mathtt{mkpr}(N) = \langle 1 \rangle N$:

$$\begin{array}{rcl} \mathtt{mknd}(N) & = & N \\ \mathtt{mknd}(a[i] \,.\, RP) & = & a[i] \,.\, \mathtt{mknd}(RP) \\ \mathtt{mknd}(RN + RM) & = & \left\{ \begin{array}{ll} \mathtt{mknd}(RN) + RM & \text{if } \mathtt{std}(RM) \\ RN + \mathtt{mknd}(RM) & \text{if } \mathtt{std}(RN) \end{array} \right. \\ \mathtt{mknd}(RN \parallel_L RM) & = & \mathtt{mknd}(RN) \parallel_L \mathtt{mknd}(RM) \\ \mathtt{mknd}(\langle 1 \rangle N) & = & N \\ \mathtt{mknd}(\langle p_z \rangle^{[i]} RN_z \bigoplus_{h \in H \setminus \{z\}} \langle p_h \rangle RN_h) & = & \langle p_z \rangle^{[i]} \mathtt{mknd}(RN_z) \bigoplus_{h \in H \setminus \{z\}} \langle p_h \rangle RN_h \\ \mathtt{mknd}(RP \parallel_L RQ) & = & \mathtt{mknd}(RP) \parallel_L \mathtt{mknd}(RQ) \end{array}$$

Example Appendix A.1. Consider the strictly alternating process $a \,.\, \langle 1 \rangle \underline{0} \parallel_{\emptyset} b \,.\, \langle 1 \rangle \underline{0}$. If a is executed first, then we have the following sequence of forward transitions where `mkpr` is used twice:

$$\begin{array}{l} a \,.\, \langle 1 \rangle \underline{0} \parallel_{\emptyset} b \,.\, \langle 1 \rangle \underline{0} \xrightarrow{a[i]}_{\mathtt{a}} a[i] \,.\, \langle 1 \rangle \underline{0} \parallel_{\emptyset} \langle 1 \rangle b \,.\, \langle 1 \rangle \underline{0} \xrightarrow{(1)^{[j]}}_{\mathtt{p}} a[i] \,.\, \langle 1 \rangle^{[j]} \underline{0} \parallel_{\emptyset} \langle 1 \rangle^{[j]} b \,.\, \langle 1 \rangle \underline{0} \\ \qquad \xrightarrow{b[h]}_{\mathtt{a}} a[i] \,.\, \langle 1 \rangle^{[j]} \langle 1 \rangle \underline{0} \parallel_{\emptyset} \langle 1 \rangle^{[j]} b[h] \,.\, \langle 1 \rangle \underline{0} \xrightarrow{(1)^{[k]}}_{\mathtt{p}} a[i] \,.\, \langle 1 \rangle^{[j]} \langle 1 \rangle^{[k]} \underline{0} \parallel_{\emptyset} \langle 1 \rangle^{[j]} b[h] \,.\, \langle 1 \rangle^{[k]} \underline{0} \end{array}$$

In the backward direction, it is `mknd` to be used twice. ■

$$(\textsc{Act}1_{\mathrm{sa}})\ \frac{\mathtt{std}(RP)}{a\,.\,RP \xrightarrow{a[i]}_{\mathtt{a}} a[i]\,.\,RP}$$

$$(\textsc{Act}1^{\bullet}_{\mathrm{sa}})\ \frac{\mathtt{std}(RP)}{a[i]\,.\,RP \overset{a[i]}{\dashrightarrow}_{\mathtt{a}} a\,.\,RP}$$

$$(\textsc{Act}2_{\mathrm{sa}})\ \frac{RP \xrightarrow{b[j]}_{\mathtt{a}} RP' \quad j \neq i}{a[i]\,.\,RP \xrightarrow{b[j]}_{\mathtt{a}} a[i]\,.\,RP'}$$

$$(\textsc{Act}2^{\bullet}_{\mathrm{sa}})\ \frac{RP \overset{b[j]}{\dashrightarrow}_{\mathtt{a}} RP' \quad j \neq i}{a[i]\,.\,RP \overset{b[j]}{\dashrightarrow}_{\mathtt{a}} a[i]\,.\,RP'}$$

$$(\textsc{Act}3_{\mathrm{sa}})\ \frac{RN_z \xrightarrow{b[j]}_{\mathtt{a}} RN'_z \quad z \in H \quad \forall h \in H \setminus \{z\}.\,\mathtt{std}(RN_h)}{\langle p_z\rangle^{[i]} RN_z \displaystyle\bigoplus_{h \in H\setminus\{z\}} \langle p_h\rangle RN_h \xrightarrow{b[j]}_{\mathtt{a}} \langle p_z\rangle^{[i]} RN'_z \displaystyle\bigoplus_{h \in H\setminus\{z\}} \langle p_h\rangle RN_h}$$

$$(\textsc{Act}3^{\bullet}_{\mathrm{sa}})\ \frac{RN_z \overset{b[j]}{\dashrightarrow}_{\mathtt{a}} RN'_z \quad z \in H \quad \forall h \in H \setminus \{z\}.\,\mathtt{std}(RN_h)}{\langle p_z\rangle^{[i]} RN_z \displaystyle\bigoplus_{h \in H\setminus\{z\}} \langle p_h\rangle RN_h \overset{b[j]}{\dashrightarrow}_{\mathtt{a}} \langle p_z\rangle^{[i]} RN'_z \displaystyle\bigoplus_{h \in H\setminus\{z\}} \langle p_h\rangle RN_h}$$

$$(\textsc{Cho}_{\mathrm{sa}})\ \frac{RN \xrightarrow{a[i]}_{\mathtt{a}} RN' \quad \mathtt{std}(RM)}{RN + RM \xrightarrow{a[i]}_{\mathtt{a}} RN' + RM}$$

$$(\textsc{Cho}^{\bullet}_{\mathrm{sa}})\ \frac{RN \overset{a[i]}{\dashrightarrow}_{\mathtt{a}} RN' \quad \mathtt{std}(RM)}{RN + RM \overset{a[i]}{\dashrightarrow}_{\mathtt{a}} RN' + RM}$$

$$(\textsc{Par}_{\mathrm{sa}})\ \frac{RN \xrightarrow{a[i]}_{\mathtt{a}} RN' \quad a \notin L \quad i \notin \mathtt{key}_{\mathtt{a}}(RM)}{RN \parallel_L RM \xrightarrow{a[i]}_{\mathtt{a}} RN' \parallel_L \mathtt{mkpr}(RM)}$$

$$(\textsc{Par}^{\bullet}_{\mathrm{sa}})\ \frac{RN \overset{a[i]}{\dashrightarrow}_{\mathtt{a}} RN' \quad a \notin L \quad i \notin \mathtt{key}_{\mathtt{a}}(RM)}{RN \parallel_L RM \overset{a[i]}{\dashrightarrow}_{\mathtt{a}} RN' \parallel_L \mathtt{mknd}(RM)}$$

$$(\textsc{Coo}_{\mathrm{sa}})\ \frac{RN \xrightarrow{a[i]}_{\mathtt{a}} RN' \quad RM \xrightarrow{a[i]}_{\mathtt{a}} RM' \quad a \in L}{RN \parallel_L RM \xrightarrow{a[i]}_{\mathtt{a}} RN' \parallel_L RM'}$$

$$(\textsc{Coo}^{\bullet}_{\mathrm{sa}})\ \frac{RN \overset{a[i]}{\dashrightarrow}_{\mathtt{a}} RN' \quad RM \overset{a[i]}{\dashrightarrow}_{\mathtt{a}} RM' \quad a \in L}{RN \parallel_L RM \overset{a[i]}{\dashrightarrow}_{\mathtt{a}} RN' \parallel_L RM'}$$

Table A.9: Operational semantic rules for RPPC$_{\mathrm{sa}}$ action transitions

$$(\textsc{PSel1}_{\mathrm{sa}})\ \frac{z \in H \quad \forall h \in H.\, \mathtt{std}(RN_h)}{\bigoplus_{h \in H} \langle p_h \rangle RN_h \xrightarrow{(p_z)^{[i]}}_{\mathtt{p}} \langle p_z \rangle^{[i]} RN_z \bigoplus_{h \in H \setminus \{z\}} \langle p_h \rangle RN_h}$$

$$(\textsc{PSel1}^{\bullet}_{\mathrm{sa}})\ \frac{z \in H \quad \forall h \in H.\, \mathtt{std}(RN_h)}{\langle p_z \rangle^{[i]} RN_z \bigoplus_{h \in H \setminus \{z\}} \langle p_h \rangle RN_h \overset{(p_z)^{[i]}}{\dashrightarrow}_{\mathtt{p}} \bigoplus_{h \in H} \langle p_h \rangle RN_h}$$

$$(\textsc{PSel2}_{\mathrm{sa}})\ \frac{RN_z \xrightarrow{(q)^{[j]}}_{\mathtt{p}} RN'_z \quad z \in H \quad \forall h \in H \setminus \{z\}.\, \mathtt{std}(RN_h) \quad j \neq i}{\langle p_z \rangle^{[i]} RN_z \bigoplus_{h \in H \setminus \{z\}} \langle p_h \rangle RN_h \xrightarrow{(q)^{[j]}}_{\mathtt{p}} \langle p_z \rangle^{[i]} RN'_z \bigoplus_{h \in H \setminus \{z\}} \langle p_h \rangle RN_h}$$

$$(\textsc{PSel2}^{\bullet}_{\mathrm{sa}})\ \frac{RN_z \overset{(q)^{[j]}}{\dashrightarrow}_{\mathtt{p}} RN'_z \quad z \in H \quad \forall h \in H \setminus \{z\}.\, \mathtt{std}(RN_h) \quad j \neq i}{\langle p_z \rangle^{[i]} RN_z \bigoplus_{h \in H \setminus \{z\}} \langle p_h \rangle RN_h \overset{(q)^{[j]}}{\dashrightarrow}_{\mathtt{p}} \langle p_z \rangle^{[i]} RN'_z \bigoplus_{h \in H \setminus \{z\}} \langle p_h \rangle RN_h}$$

$$(\textsc{PSel3}_{\mathrm{sa}})\ \frac{RP \xrightarrow{(q)^{[j]}}_{\mathtt{p}} RP'}{a[i] \,.\, RP \xrightarrow{(q)^{[j]}}_{\mathtt{p}} a[i] \,.\, RP'}$$

$$(\textsc{PSel3}^{\bullet}_{\mathrm{sa}})\ \frac{RP \overset{(q)^{[j]}}{\dashrightarrow}_{\mathtt{p}} RP'}{a[i] \,.\, RP \overset{(q)^{[j]}}{\dashrightarrow}_{\mathtt{p}} a[i] \,.\, RP'}$$

$$(\textsc{PCho}_{\mathrm{sa}})\ \frac{RN \xrightarrow{(q)^{[j]}}_{\mathtt{p}} RN' \quad \mathtt{std}(RM)}{RN + RM \xrightarrow{(q)^{[j]}}_{\mathtt{p}} RN' + RM}$$

$$(\textsc{PCho}^{\bullet}_{\mathrm{sa}})\ \frac{RN \overset{(q)^{[j]}}{\dashrightarrow}_{\mathtt{p}} RN' \quad \mathtt{std}(RM)}{RN + RM \overset{(q)^{[j]}}{\dashrightarrow}_{\mathtt{p}} RN' + RM}$$

$$(\textsc{PCoo}_{\mathrm{sa}})\ \frac{RP \xrightarrow{(p)^{[i]}}_{\mathtt{p}} RP' \quad RQ \xrightarrow{(q)^{[i]}}_{\mathtt{p}} RQ'}{RP \,\|_L\, RQ \xrightarrow{(p \cdot q)^{[i]}}_{\mathtt{p}} RP' \,\|_L\, RQ'}$$

$$(\textsc{PCoo}^{\bullet}_{\mathrm{sa}})\ \frac{RP \overset{(p)^{[i]}}{\dashrightarrow}_{\mathtt{p}} RP' \quad RQ \overset{(q)^{[i]}}{\dashrightarrow}_{\mathtt{p}} RQ'}{RP \,\|_L\, RQ \overset{(p \cdot q)^{[i]}}{\dashrightarrow}_{\mathtt{p}} RP' \,\|_L\, RQ'}$$

Table A.10: Operational semantic rules for $\mathrm{RPPC}_{\mathrm{sa}}$ probabilistic transitions

References

[1] L. Aceto and D. Murphy. Timing and causality in process algebra. *Acta Informatica*, 33:317–350, 1996.

[2] S. Andova, S. Georgievska, and N. Trcka. Branching bisimulation congruence for probabilistic systems. *Theoretical Computer Science*, 413:58–72, 2012.

[3] J.C.M. Baeten and J.A. Bergstra. Real time process algebra. *Formal Aspects of Computing*, 3:142–188, 1991.

[4] C.H. Bennett. Logical reversibility of computation. *IBM Journal of Research and Development*, 17:525–532, 1973.

[5] C.H. Bennett, G. Brassard, C. Crépeau, R. Jozsa, A. Peres, and W.K. Wootters. Teleporting an unknown quantum state via dual classical and Einstein-Podolsky-Rosen channels. *Physical Review Letters*, 70:1895–1899, 1993.

[6] M. Bernardo, F. Corradini, and L. Tesei. Timed process calculi with deterministic or stochastic delays: Commuting between durational and durationless actions. *Theoretical Computer Science*, 629:2–39, 2016.

[7] M. Bernardo and A. Esposito. Modal logic characterizations of forward, reverse, and forward-reverse bisimilarities. In *Proc. of the 14th Int. Symp. on Games, Automata, Logics, and Formal Verification (GANDALF 2023)*, volume 390 of *EPTCS*, pages 67–81, 2023.

[8] M. Bernardo, I. Lanese, A. Marin, C.A. Mezzina, S. Rossi, and C. Sacerdoti Coen. Causal reversibility implies time reversibility. In *Proc. of the 20th Int. Conf. on the Quantitative Evaluation of Systems (QEST 2023)*, volume 14287 of *LNCS*, pages 270–287. Springer, 2023.

[9] M. Bernardo and C.A. Mezzina. Bridging causal reversibility and time reversibility: A stochastic process algebraic approach. *Logical Methods in Computer Science*, 19(2):6:1–6:27, 2023.

[10] M. Bernardo and C.A. Mezzina. Causal reversibility for timed process calculi with lazy/eager durationless actions and time additivity. In *Proc. of the 21st Int. Conf. on Formal Modeling and Analysis of Timed Systems (FORMATS 2023)*, volume 14138 of *LNCS*, pages 15–32. Springer, 2023.

[11] M. Bernardo and C.A. Mezzina. Reversibility in process calculi with nondeterminism and probabilities. In *Proc. of the 21st Int. Coll. on Theoretical Aspects of Computing (ICTAC 2024)*, volume 15373 of *LNCS*, pages 251–271. Springer, 2024.

[12] A. Bérut, A. Arakelyan, A. Petrosyan, S. Ciliberto, R. Dillenschneider, and E. Lutz. Experimental verification of Landauer's principle linking information and thermodynamics. *Nature*, 483:187–189, 2012.

[13] L. Bocchi, I. Lanese, C.A. Mezzina, and S. Yuen. revTPL: The reversible temporal process language. *Logical Methods in Computer Science*, 20(1):11:1–11:35, 2024.

[14] T. Bolognesi and F. Lucidi. LOTOS-like process algebras with urgent or timed interactions. In *Proc. of the 4th Int. Conf. on Formal Description Techniques for Distributed Systems and Communication Protocols (FORTE 1991)*, volume C-2 of *IFIP Transactions*, pages 249–264, 1991.

[15] S.D. Brookes, C.A.R. Hoare, and A.W. Roscoe. A theory of communicating sequential processes. *Journal of the ACM*, 31:560–599, 1984.

[16] S. Cattani and R. Segala. Decision algorithms for probabilistic bisimulation. In *Proc. of the 13th Int. Conf. on Concurrency Theory (CONCUR 2002)*, volume 2421 of *LNCS*, pages 371–385. Springer, 2002.

[17] L. Ceragioli, F. Gadducci, G. Lomurno, and G. Tedeschi. Quantum bisimilarity via barbs and contexts: Curbing the power of non-deterministic observers. *Proc. of the ACM on Programming Languages*, 8:43:1–43:29, 2024.

[18] L. Ceragioli, F. Gadducci, G. Lomurno, and G. Tedeschi. Effect semantics for quantum process calculi. In *Proc. of the 35th Int. Conf. on Concurrency Theory (CONCUR 2024)*, volume 311 of *LIPIcs*, pages 16:1–16:22, 2024.

[19] K. Chatterjee, A.K. Goharshady, and A. Pourdamghani. Probabilistic smart contracts: Secure randomness on the blockchain. In *Proc. of the 1st IEEE Int. Conf. on Blockchain and Cryptocurrency (ICBC 2019)*, pages 403–412. IEEE-CS Press, 2019.

[20] F. Corradini. Absolute versus relative time in process algebras. *Information and Computation*, 156:122–172, 2000.

[21] F. Corradini, W. Vogler, and L. Jenner. Comparing the worst-case efficiency of asynchronous systems with PAFAS. *Acta Informatica*, 38:735–792, 2002.

[22] V. Danos and J. Krivine. Reversible communicating systems. In *Proc. of the 15th Int. Conf. on Concurrency Theory (CONCUR 2004)*, volume 3170 of *LNCS*, pages 292–307. Springer, 2004.

[23] V. Danos and J. Krivine. Transactions in RCCS. In *Proc. of the 16th Int. Conf. on Concurrency Theory (CONCUR 2005)*, volume 3653 of *LNCS*, pages 398–412. Springer, 2005.

[24] R. De Nicola, U. Montanari, and F. Vaandrager. Back and forth bisimulations. In *Proc. of the 1st Int. Conf. on Concurrency Theory (CONCUR 1990)*, volume 458 of *LNCS*, pages 152–165. Springer, 1990.

[25] C. Derman. *Finite State Markovian Decision Processes*. Academic Press, 1970.

[26] A. Esposito, A. Aldini, and M. Bernardo. Noninterference analysis of reversible probabilistic systems. In *Proc. of the 44th Int. Conf. on Formal Techniques for Distributed Objects, Components, and Systems (FORTE 2024)*, volume 14678 of *LNCS*, pages 39–59. Springer, 2024.

[27] A. Esposito, A. Aldini, and M. Bernardo. Noninterference analysis of stochastically timed reversible systems. In *Proc. of the 45th Int. Conf. on Formal Techniques for Distributed Objects, Components, and Systems (FORTE 2025)*, volume 15732 of *LNCS*, pages 75–95. Springer, 2025.

[28] A. Esposito, A. Aldini, and M. Bernardo. Noninterference analysis of deterministically timed reversible systems. In *Proc. of the 2nd Int. Joint Conf. on Quantitative Evaluation of Systems and Formal Modeling and Analysis of Timed Systems (QEST+FORMATS 2025)*, volume 16143 of *LNCS*, pages 293–313. Springer, 2025.

[29] A. Esposito, A. Aldini, M. Bernardo, and S. Rossi. Noninterference analysis of reversible systems: An approach based on branching bisimilarity. *Logical Methods in Computer Science*, 21(1):6:1–6:28, 2025.

[30] M.P. Frank. Physical foundations of Landauer's principle. In *Proc. of the 10th Int. Conf. on Reversible Computation (RC 2018)*, volume 11106 of *LNCS*, pages 3–33. Springer, 2018.

[31] H. Garcia-Molina and K. Salem. Sagas. In *Proc. of the 13th ACM Int. Conf. on Management of Data (SIGMOD 1987)*, pages 249–259. ACM Press, 1987.

[32] S.J. Gay and R. Nagarajan. Communicating quantum processes. In *Proc. of the 32nd ACM Symp. on Principles of Programming Languages (POPL 2005)*, pages 145–157. ACM Press, 2005.

[33] E. Giachino, I. Lanese, and C.A. Mezzina. Causal-consistent reversible debugging. In *Proc. of the 17th Int. Conf. on Fundamental Approaches to Software Engineering (FASE 2014)*, volume 8411 of *LNCS*, pages 370–384. Springer, 2014.

[34] R.J. van Glabbeek, S.A. Smolka, and B. Steffen. Reactive, generative and stratified models of probabilistic processes. *Information and Computation*, 121:59–80, 1995.

[35] R.J. van Glabbeek and W.P. Weijland. Branching time and abstraction in bisimulation semantics. *Journal of the ACM*, 43:555–600, 1996.

[36] H. Hansson. *Time and Probability in Formal Design of Distributed Systems.* PhD Thesis, 1992.

[37] H. Hansson and B. Jonsson. A calculus for communicating systems with time and probabilities. In *Proc. of the 11th IEEE Real-Time Systems Symp. (RTSS 1990)*, pages 278–287. IEEE-CS Press, 1990.

[38] M. Hennessy and T. Regan. A process algebra for timed systems. *Information and Computation*, 117:221–239, 1995.

[39] F.P. Kelly. *Reversibility and Stochastic Networks.* John Wiley & Sons, 1979.

[40] J.G. Kemeny and J.L. Snell. *Finite Markov Chains.* Van Nostrand, 1960.

[41] J. Krivine. A verification technique for reversible process algebra. In *Proc. of the 4th Int. Workshop on Reversible Computation (RC 2012)*, volume 7581 of *LNCS*, pages 204–217. Springer, 2012.

[42] R. Landauer. Irreversibility and heat generation in the computing process. *IBM Journal of Research and Development*, 5:183–191, 1961.

[43] I. Lanese, M. Lienhardt, C.A. Mezzina, A. Schmitt, and J.-B. Stefani. Concurrent flexible reversibility. In *Proc. of the 22nd European Symp. on Programming (ESOP 2013)*, volume 7792 of *LNCS*, pages 370–390. Springer, 2013.

[44] I. Lanese, D. Medić, and C.A. Mezzina. Static versus dynamic reversibility in CCS. *Acta Informatica*, 58:1–34, 2021.

[45] I. Lanese, N. Nishida, A. Palacios, and G. Vidal. CauDEr: A causal-consistent reversible debugger for Erlang. In *Proc. of the 14th Int. Symp. on Functional and Logic Programming (FLOPS 2018)*, volume 10818 of *LNCS*, pages 247–263. Springer, 2018.

[46] I. Lanese, I. Phillips, and I. Ulidowski. An axiomatic theory for reversible computation. *ACM Trans. on Computational Logic*, 25(2):11:1–11:40, 2024.

[47] J.S. Laursen, L.-P. Ellekilde, and U.P. Schultz. Modelling reversible execution of robotic assembly. *Robotica*, 36:625–654, 2018.

[48] D. Lehmann and M.O. Rabin. On the advantage of free choice: A symmetric and fully distributed solution to the dining philosophers problem. In *Proc. of the 8th ACM Symp. on Principles of Programming Languages (POPL 1981)*, pages 133–138. ACM Press, 1981.

[49] J.-J. Lévy. An algebraic interpretation of the $\lambda\beta$K-calculus; and an application of a labelled λ-calculus. *Theoretical Computer Science*, 2:97–114, 1976.

[50] R. Milner. *Communication and Concurrency.* Prentice Hall, 1989.

[51] F. Moller and C. Tofts. A temporal calculus of communicating systems. In *Proc. of the 1st Int. Conf. on Concurrency Theory (CONCUR 1990)*, volume 458 of *LNCS*, pages 401–415. Springer, 1990.

[52] X. Nicollin and J. Sifakis. The algebra of timed processes ATP: Theory and application. *Information and Computation*, 114:131–178, 1994.

[53] K.S. Perumalla and A.J. Park. Reverse computation for rollback-based fault tolerance in large parallel systems – Evaluating the potential gains and systems effects. *Cluster Computing*, 17:303–313, 2014.

[54] A. Philippou, I. Lee, and O. Sokolsky. Weak bisimulation for probabilistic systems. In *Proc. of the 11th Int. Conf. on Concurrency Theory (CONCUR 2000)*, volume 1877 of *LNCS*, pages 334–349. Springer, 2000.

[55] I. Phillips and I. Ulidowski. Reversing algebraic process calculi. *Journal of Logic and Algebraic Programming*, 73:70–96, 2007.

[56] I. Phillips, I. Ulidowski, and S. Yuen. A reversible process calculus and the modelling of the ERK signalling pathway. In *Proc. of the 4th Int. Workshop on Reversible Computation (RC 2012)*, volume 7581 of *LNCS*, pages 218–232. Springer, 2012.

[57] G.M. Pinna. Reversing steps in membrane systems computations. In *Proc. of the 18th Int. Conf. on Membrane Computing (CMC 2017)*, volume 10725 of *LNCS*, pages 245–261. Springer, 2017.

[58] P. Prabhu, G. Ramalingam, and K. Vaswani. Safe programmable speculative parallelism. In *Proc. of the 31st ACM Conf. on Programming Language Design and Implementation (PLDI 2010)*, pages 50–61. ACM Press, 2010.

[59] J. Quemada, D. de Frutos, and A. Azcorra. TIC: A timed calculus. *Formal Aspects of Computing*, 5:224–252, 1993.

[60] M.O. Rabin. Probabilistic automata. *Information and Control*, 6:230–245, 1963.

[61] G.M. Reed and A.W. Roscoe. A timed model for communicating sequential processes. *Theoretical Computer Science*, 58:249–261, 1988.

[62] M. Schordan, T. Oppelstrup, D.R. Jefferson, and P.D. Barnes Jr. Generation of reversible C++ code for optimistic parallel discrete event simulation. *New Generation Computing*, 36:257–280, 2018.

[63] R. Segala. *Modeling and Verification of Randomized Distributed Real-Time Systems.* PhD Thesis, 1995.

[64] R. Segala and N.A. Lynch. Probabilistic simulations for probabilistic processes. In *Proc. of the 5th Int. Conf. on Concurrency Theory (CONCUR 1994)*, volume 836 of *LNCS*, pages 481–496. Springer, 1994.

[65] R. Segala and A. Turrini. Comparative analysis of bisimulation relations on alternating and non-alternating probabilistic

models. In *Proc. of the 2nd Int. Conf. on the Quantitative Evaluation of Systems (QEST 2005)*, pages 44–53. IEEE-CS Press, 2005.

[66] H. Siljak, K. Psara, and A. Philippou. Distributed antenna selection for massive MIMO using reversing Petri nets. *IEEE Wireless Communication Letters*, 8:1427–1430, 2019.

[67] A. Turrini and H. Hermanns. Polynomial time decision algorithms for probabilistic automata. *Information and Computation*, 244:134–171, 2015.

[68] M. Vassor and J.-B. Stefani. Checkpoint/rollback vs causally-consistent reversibility. In *Proc. of the 10th Int. Conf. on Reversible Computation (RC 2018)*, volume 11106 of *LNCS*, pages 286–303. Springer, 2018.

[69] E. de Vries, V. Koutavas, and M. Hennessy. Communicating transactions. In *Proc. of the 21st Int. Conf. on Concurrency Theory (CONCUR 2010)*, volume 6269 of *LNCS*, pages 569–583. Springer, 2010.

[70] Y. Wang. Reversible quantum process algebra, 2021. arXiv:1501.05260.

[71] B. Ycart. The philosophers' process: An ergodic reversible nearest particle system. *Annals of Applied Probability*, 3:356–363, 1993.

[72] Wang Yi. CCS + time = an interleaving model for real time systems. In *Proc. of the 18th Int. Coll. on Automata, Languages and Programming (ICALP 1991)*, volume 510 of *LNCS*, pages 217–228. Springer, 1991.

[73] M. Ying, Y. Feng, R. Duan, and Z. Ji. An algebra of quantum processes. *ACM Trans. on Computational Logic*, 10(3):19:1–19:36, 2009.

[74] T. Yokoyama and R. Glück. A reversible programming language and its invertible self-interpreter. In *Proc. of the 13th ACM Workshop on Partial Evaluation and Semantics-based Program Manipulation (PEPM 2007)*, pages 144–153. ACM Press, 2007.