The Correctness of Concurrencies in (Reversible) Concurrent Calculi

Clément Aubert

The Correctness of Concurrencies in (Reversible) Concurrent Calculi

Clément Aubert [1,]

School of Computer and Cyber Sciences, Augusta University, Georgia, USA

Abstract

This article designs a general principle to check the correctness of the definition of concurrency (a.k.a. independence) of events for concurrent calculi. Concurrency relations are central in process algebras, but also two-sided: they are often defined independently on composable and on coinitial transitions, and no criteria exists to assess whether they "interact correctly". This article starts by examining how reversibility can provide such a correctness of concurrencies criteria, and its implications. It then defines, for the first time, a syntactical definition of concurrency for CCSK, a reversible declension of the calculus of communicating systems. To do so, according to our criteria, requires to define concurrency relations for all types of transitions along two axis: direction (forward or backward) and concomitance (coinitial or composable). Our definition is uniform thanks to proved transition systems and satisfy our sanity checks: square properties, sideways diamonds, but also the reversible checks (reverse diamonds and causal consistency). We also prove that our formalism is either equivalent to or a refinement of pre-existing definitions of concurrency for reversible systems. We conclude by discussing additional criteria and possible future works.

Keywords: Formal Semantics, Process Algebra, Concurrency, Reversibility

2020 MSC: 68N19, 68Q85

Email address: caubert@augusta.edu (Clément Aubert)

[1]This work has benefited from the support of the Augusta University Research Assistance Grants Program.

Preamble

Following Lars Kristiansen [1], we tried to give priority to readability, particularly in this Preamble, to reach a broader audience: reversibility is not a topic on its own, it is a tool that can bring enlightenment to diverse fields, and we hope that this preamble will help the reader unfamiliar with reversibility but curious about concurrency to understand how this tool can be leveraged with benefits.

A concurrent program is by nature extremely hard to debug [2], but its correctness can be evaluated by writing a specification, and then checking that the program matches it [3]. Expressing those specifications requires a formal language, that abstracts away irrelevant details and focus on the program's observable behaviour. Process algebras provide such a high-level description of interactions, communications, and synchronizations between a collection of independent processes that allows to model a vast range of situations. A central element of those algebras is to define when two events (generally associated to the transitions that triggered them) are independent, or *concurrent*. By duality, events that are not concurrent are dubbed dependent, or *causally related*.[2] Being able to distinguish between events those that are causally related and those that are not is one of the crucial contributions of process algebras, as this mechanism allows to sidestep many of the difficulties one has to face when debugging concurrent programs.

But how can one guarantee that those definitions of concurrency and causal relations are "the right ones"? Since the are defined by duality (two events are concurrent iff they are not causally related), it suffices to define and check only one of the two notion. For concurrency, written $\smile$, a standard correctness criteria is expressed in terms of "diamonds":

[2] As a matter of fact, the order is often swapped: dependency is the primitive relation, and concurrency is defined by duality.

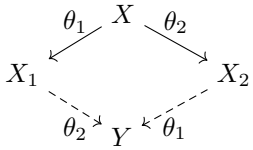
$$\forall t_1 : X \xrightarrow{\theta_1} X_1, t_2 : X \xrightarrow{\theta_2} X_2 \text{ with } t_1 \smile t_2, \exists Y \text{ s.t. } X_1 \xrightarrow{\theta_2} Y \text{ and } X_2 \xrightarrow{\theta_1} Y.$$

(Square Property)

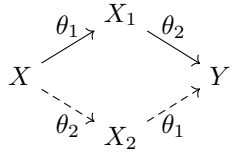
$$\forall t_1 : X \xrightarrow{\theta_1} X_1, t_2 : X_1 \xrightarrow{\theta_2} Y \text{ with } t_1 \smile t_2, \exists X_2 \text{ s.t. } X \xrightarrow{\theta_2} X_2 \xrightarrow{\theta_1} Y.$$

(Sidways Diamond)

They intuitively means that any two transitions t_1, t_2 that are independent (i.e., that are in the $\smile$ relation) can be combined differently without altering the end-result. (Square Property) expresses this fact for *coinitial* transitions: it states that simultaneous transitions starting from the same state can "later on" agree if they are concurrent. (Sidways Diamond) expresses it for *composable* transitions: subsequent transitions, that follow each other, can be swapped if they are concurrent. Graphically, we can represent them as follows:

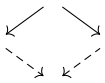


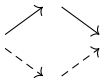
(Square Property)



(Sidways Diamond)

More succinctly:





Looking more carefully, one may observe that requiring both diamonds to hold actually requires *two* definitions of concurrency: one on coinitial transitions ($\smile_i$), and one on composable transitions ($\smile_c$). *If* they are related, and if they are, *how*, is generally overlooked: in the vast literature on process algebras, one can find systems where only one notion is defined, but to my knowledge those that defines the two do not have a formal criteria to assess whenever they interact correctly. At best, the same definition is be used for both notions of concurrencies [4], which seems to prevent the need for a formal criteria. One reason for this lack of criteria, we assume, is that "interacting correctly" is difficult to define: how should composable and coinitial transitions relate w.r.t. concurrency?

One of the goal of this paper is to convince the reader that reversibility provides an excellent method to answer that question. It has been, to the best of our knowledge, completely overlooked, despite its simplicity and universal applicability. The starting point is the *loop lemma*, that states that any transition in a reversible system $t : X \xrightarrow{\theta} Y$ can be reversed[3] as $t^\bullet : Y \overset{\theta}{\rightsquigarrow} X$ with $(t^\bullet)^\bullet = t$. From there, a correctness criterion linking $\smile_i$ and $\smile_c$ can easily be formulated:

$$\begin{aligned}&(t_1 \smile_i t_2 \text{ for } t_1 : X \xrightarrow{\theta_1} X_1, t_2 : X \xrightarrow{\theta_2} X_2)\\ \iff &(t_1^\bullet \smile_c t_2 \text{ for } t_1^\bullet : X_1 \overset{\theta_1}{\rightsquigarrow} X, t_2 : X \xrightarrow{\theta_2} X_2)\end{aligned}$$

(Correctness of Concurrencies)

However, this correctness uses a definition of $\smile_i$ on *forward* coinitial transitions, and a definition of $\smile_c$ on *backward then forward* composable transitions. Looking more closely, defining both $\smile_i$ and $\smile_c$ on reversible systems requires to split each definition in four, depending on the directions of the transitions:

	coinitial	Composable
Both forward	$\smile_i^f$ ↙ ↘	$\smile_c^f$ ↗ ↘
Both backward	$\smile_i^b$ ⇜ ⇝	$\smile_c^b$ ⇗ ⇝
Forward then backward	$\smile_i^{fb}$ ↙ ⇝	$\smile_c^{fb}$ ↗ ⇝
Backward then forward	$\smile_i^{bf}$ ⇜ ↘	$\smile_c^{bf}$ ⇗ ↘

Our (Correctness of Concurrencies) to relate coinitial and composable concurrencies seems to come at the high cost of having to define *eight different notions of concurrency* (not to mention the additional diamonds we now have to prove—we come back to this later). Luckily, three principles can be leveraged to limit the burden considerably:

[3]In general, the label θ is not altered by this reversing, but the rest of our discussion in this preamble would still be valid if it was, albeit probably less digest.

Concurrencies should be symmetric That is,

$$\begin{aligned}&(t_1 \smile_i^{fb} t_2 \text{ for } t_1 : X \xrightarrow{\theta_1} X_1, t_2 : X \overset{\theta_2}{\rightsquigarrow} X_2)\\ \iff &(t_2 \smile_i^{bf} t_1 \text{ for } t_2 : X \overset{\theta_2}{\rightsquigarrow} X_2, t_1 : X \xrightarrow{\theta_1} X).\end{aligned}$$

Concurrencies should be direction-agnostic That is,

$$\begin{aligned}&(t_1 \smile_c^{f} t_2 \text{ for } t_1 : X \xrightarrow{\theta_1} X_1, t_2 : X_1 \xrightarrow{\theta_2} X_2)\\ \iff &(t_2^\bullet \smile_c^{b} t_1^\bullet \text{ for } t_2^\bullet : X_2 \overset{\theta_2}{\rightsquigarrow} X_1, t_1^\bullet : X_1 \overset{\theta_1}{\rightsquigarrow} X).\end{aligned}$$

Correctness of Concurrencies The criteria we presented earlier can be instantiated as:

$$\begin{aligned}&(t_1 \smile_i^{f} t_2 \text{ for } t_1 : X \xrightarrow{\theta_1} X_1, t_2 : X \xrightarrow{\theta_2} X_2)\\ \iff &(t_1^\bullet \smile_c^{bf} t_2 \text{ for } t_1^\bullet : X_1 \overset{\theta_1}{\rightsquigarrow} X, t_2 : X \xrightarrow{\theta_2} X_2)\end{aligned}$$

$$\begin{aligned}&(t_1 \smile_i^{fb} t_2 \text{ for } t_1 : X \xrightarrow{\theta_1} X_1, t_2 : X \overset{\theta_2}{\rightsquigarrow} X_2)\\ \iff &(t_1^\bullet \smile_c^{b} t_2 \text{ for } t_1^\bullet : X_1 \overset{\theta_1}{\rightsquigarrow} X, t_2 : X \overset{\theta_2}{\rightsquigarrow} X_2)\end{aligned}$$

$$\begin{aligned}&(t_1 \smile_i^{bf} t_2 \text{ for } t_1 : X \overset{\theta_1}{\rightsquigarrow} X_1, t_2 : X \xrightarrow{\theta_2} X_2)\\ \iff &(t_1^\bullet \smile_c^{f} t_2 \text{ for } t_1^\bullet : X_1 \xrightarrow{\theta_1} X, t_2 : X \xrightarrow{\theta_2} X_2)\end{aligned}$$

$$\begin{aligned}&(t_1 \smile_i^{b} t_2 \text{ for } t_1 : X \overset{\theta_1}{\rightsquigarrow} X_1, t_2 : X \overset{\theta_2}{\rightsquigarrow} X_2)\\ \iff &(t_1^\bullet \smile_c^{fb} t_2 \text{ for } t_1^\bullet : X_1 \xrightarrow{\theta_1} X, t_2 : X \overset{\theta_2}{\rightsquigarrow} X_2)\end{aligned}$$

Writing e.g., $\smile_i^{f} \iff \smile_c^{bf}$ to express that $\smile_i^{f}$ and $\smile_c^{bf}$ can be mutually defined, our three principles give:

$$\smile_i^{f} \iff \smile_c^{bf} \tag{1}$$

$$\smile_i^{b} \iff \smile_c^{fb} \tag{2}$$

$$\smile_i^{fb} \iff \smile_i^{bf} \iff \smile_c^{f} \iff \smile_c^{b} \tag{3}$$

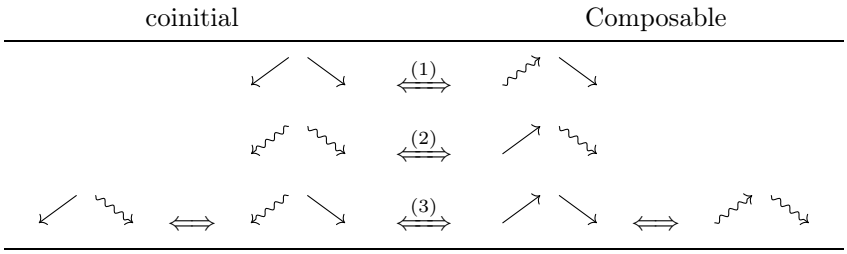


Table 1: Concurrencies for Reversible Systems

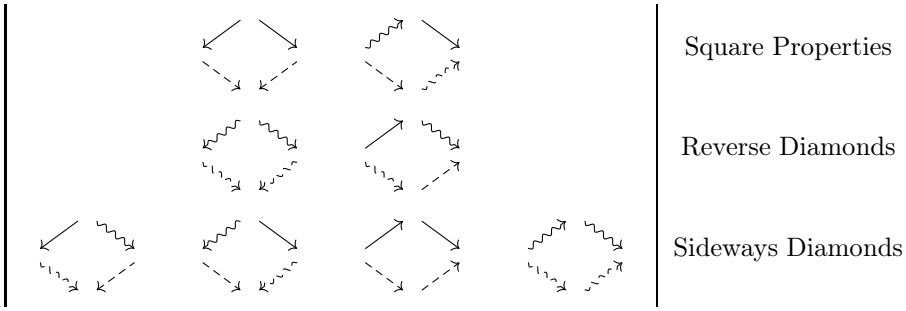


Table 2: Diamonds and Squares for Reversible Systems

Hence, we need to define only one relation on each line (1), (2) and (3) to define concurrency for the eight possible cases. We represent those equivalences graphically in Table 1.

Defining three relations instead of eight and letting the principles we laid out earlier guarantee that they interact correctly saves us some burden, but we still have to address our initial question: how can we make sure that those (now numerous) definitions of concurrency are "the right ones"? A natural strategy is to decline our diamonds ((Square Property) and (Sidways Diamond)) to reversible systems to account for all the possible situations, as presented in Table 2.[4]

Satisfying all those diamonds is, in our opinion, an excellent indication that

[4]There has been some variations in the naming of those properties. From a rewriting perspective, they are all forms of (local) confluence, but the concurrency community has used diverse namings. The name "reverse diamond" was coined very early in the study of reversible systems [5, Proposition 5.10][6, Definition 2.3] and seemed the best fit for this property that does not exist in forward-only systems.

the concurrency relations were properly defined and behave as expected. Furthermore, using the logical principles we presented, proving only one of each (Square Properties), (Reverse Diamonds) and (Sideways Diamonds) is enough to obtain them all.

We believe this article to be the first one to identify and clearly lay out this criteria to guarantee the correctness of the definitions of concurrency. For systems endowed with only a definition for coinitial or composable transitions, our criteria can also be used to provide a definition for the missing one. An added beauty is that it allows to mutually define concurrency relations between coinitial and composable transitions, and between forward and backward transitions, making both worlds interact in harmony. It is enabled by the study of reversibility, which has repeatedly contributed to a better understanding of notions applicable to the forward-only world too.

This article illustrates those general principles for a particular concurrent system, CCSK. It introduces a single definition for $\smile_c^f$, $\smile_c^b$ and $\smile_c^{fb}$ and $\smile_c^{bf}$ by providing a direction-agnostic definition of $\smile_c$, and prove three of the required diamonds. We believe the general applicability of the principles exposed in this preamble goes far beyond the particular case of CCSK, or of process algebras for that matter, and hope that it will inspire researchers in other fields to leverage reversibility to obtain sound, logical, notions.

1. Introduction: Reversibility, Concurrency–Interplays

Concurrency Theory is being reshaped by reversibility: fine distinctions between causality and causation [5] contradicted Milner's expansion laws [7, Example 4.11], and the study of causal models for reversible computation led to novel correction criteria for causal semantics—both reversible and irreversible [8]. "Traditional" equivalence relations have been captured syntactically [9], while original observational equivalences were developed [7]: reversibility triggered a global reconsideration of established theories and tools, with the clear intent of providing actionable methods for reversible systems [10], novel axiomatic

foundations [11] and original non-interleaving models [8, 12, 13].

Two Formalisms extend with reversible features the Calculus of Communicating Systems (CCS) [14], which is the godfather of π-calculus [15], among others formalisms. Reversible CCS (RCCS) [16] and CCS with keys (CCSK) [5] are similarly the source of most [8, 17, 18, 19]—if not all—of later formalism developed to enhance reversible systems with some respect (rollback operator, name-passing abilities, probabilistic features, ...). Even if those two systems share a lot of similarities [20], they diverge in some respects that are not fully understood—typically, it seems that different notions of "contexts with history" led to establish the existence of congruences for CCSK [7, Proposition 4.9] or the impossibility thereof for RCCS [21, Theorem 2]. However, they also share some shortcomings, and we offer to tackle one of them for CCSK, by providing a syntactical definition of concurrency that is easy to manipulate and that satisfies the usual sanity checks, in addition to our (Correctness of Concurrencies).

Reversible Concurrency is of course a central notion in the study of RCCS and CCSK, as it enables the definition of causal consistency—a principle that, intuitively, states that backward reductions can undo an action only if its consequences have already been undone—and to obtain models where concurrency and causation are decorrelated [5]. As such, it has been studied from multiple angles, but, in our opinion, never in a fully satisfactory manner. In CCSK, sideways and reverse diamonds properties were proven using conditions on keys and "joinable" transitions [5, Propositions 5.10 and 5.19], but to our knowledge no "definitive" definition of concurrency was proposed. Ad-hoc definitions relying on memory inclusion [22, Definition 3.1.1] or disjointness [16, Definition 7] for RCCS, and semantical notions for both RCCS [9, 12, 23] and CCSK [6, 13, 24] have been proposed, but, to our knowledge, none of these have ever been

1. compared to each other,
2. compared to pre-existing forward-only definitions of concurrency,
3. proven to satisfy our (Correctness of Concurrencies).

Our Contribution introduces the first syntactical definition of concurrency for CCSK (Sect. 3.1), by extending the "universal" concurrency developed for forward-only CCS [25], that leveraged *proved* transition systems [26]. Our definition of dependency (Sect. 3.3) is almost identical to the one used for proved forward-only systems, and our definition of concurrency is simple enough to be applicable to all types of transitions along the two axis: direction (forward or backward) and concomitance (coinitial or composable). The square properties, sideways and reverse diamonds are proven in a very similar fashion, and gives all the squares of Table 2 easily (Sect. 4.2). We furthermore establish the correctness of this definition by proving other expected reversible properties, among which causal consistency (Sect. 4.3). We then discuss how proved transition systems can be adapted to other reversible systems (RCCS [16, 22] and its "identified" declensions [21]) , and how our definition of concurrency relates to pre-existing ones, including one coming from reversible π-calculus (Sect. 5). In essence, we prove that our technique gives a notion of concurrency that either match or subsumes existing definitions, that sometimes lack a notion of concurrency for transitions of opposite directions. Finally, we sketch some additional criteria our definition should ideally fulfill, and how to approach them (Sect. 6). We then briefly conclude (Sect. 7).

Changelog. This article extends and improves a conference publication [27] and its preliminary technical report [28]. In particular, it:

- Clarifies in the Preamble the general applicability and methodology behind our method,
- Clarifies the interplay and differences between our definition of concurrency and the forward-only one (Sect. 3.4),
- Streamlines and clarifies the proofs of all the results,
- Contains more details about the "universality" of our approach,
- Proves the additional Coinitial Propagation of Independence property (Corollary 1), which in turns gives the Independence of Diamonds (Corollary 2),

- Generally improves the exposition and narrative.

2. Finite and Reversible Process Calculi

We begin by recalling the pre-existing material required to detail our contribution: the finite fragment of CCS, its proved transition system, and then CCSK.

2.1. A Proved Transition System For CCS

We briefly recall the (forward-only) finite fragment of the calculus of communicating system (simply called CCS) following a standard presentation [29], and then its proved transition system [25]. Proved transition systems [25, 26, 30, 31, 32, 33] enrich the transition labels with prefixes that describe parts of their derivation, to keep track of their dependencies or lack thereof.

We recall here a variation on an earlier formalism [34] that accommodated CCS with replication and enabled a definition of causality that agreed with pre-existing causal semantics of CCS and CCS with recursion [25, Theorem 1]. This system includes information about sums [25, footnote 2], but diverge in its definitions of dependencies and concurrencies: our definition of dependency needs to account for the permanence of the sum operator, and our concurrency relation accounts for internal (i.e., τ-) transitions, omitted from that work [25, Definition 3] but present in older articles [32, Definition 2.3].

Definition 1 ((Co-)names and labels)**.** Let $\mathsf{N} = \{a, b, c, \dots\}$ be a set of *names* and $\overline{\mathsf{N}} = \{\overline{a}, \overline{b}, \overline{c}, \dots\}$ its set of *co-names*. The set of *labels* L is $\mathsf{N} \cup \overline{\mathsf{N}} \cup \{\tau\}$, and we use α, β (resp. λ) to range over L (resp. $\mathsf{L}\backslash\{\tau\}$). A bijection $\overline{\cdot} : \mathsf{N} \to \overline{\mathsf{N}}$, whose inverse is also written $\overline{\cdot}$, gives the *complement* of a name, and we let $\overline{\tau} = \tau$ for commodity.

Definition 2 (Operators)**.** CCS processes are defined as usual:

$$
\begin{array}{lll}
P, Q ::= 0 & & \text{(Inactive process)} \\
\quad \alpha.P & & \text{(Prefix)} \\
\quad P\backslash\alpha & & \text{(Restriction)}
\end{array}
$$

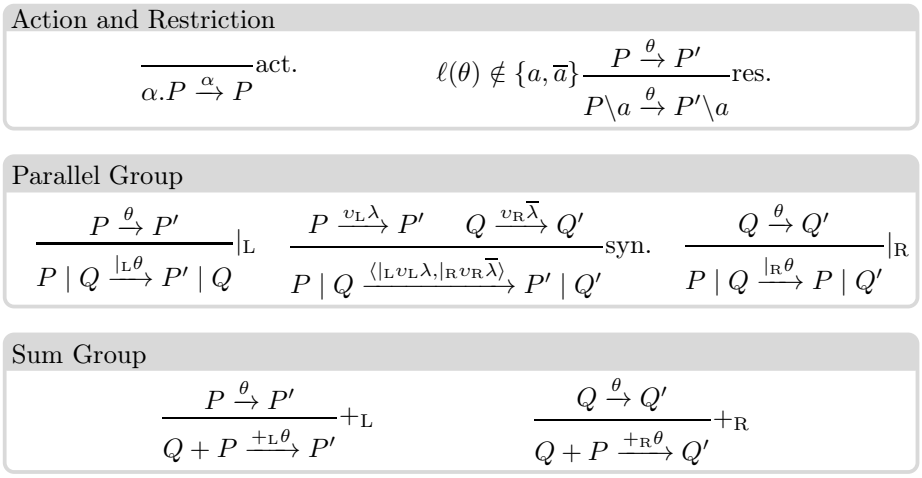


Figure 1: Rules of the proved labeled transition system (LTS) for CCS

$P + Q$ (Sum)

$P \mid Q$ (Parallel composition)

The inactive process 0 is omitted when preceded by a prefix, and the binding power of the operators [35, p. 68], from highest to lowest, is $\backslash\alpha$, $\alpha.$, $\mid$ and $+$, so that e.g., $\alpha.P + Q\backslash\alpha \mid P + a$ is to be read as $(\alpha.P) + (((Q\backslash\alpha) \mid P) + (a.0))$. In a process $P \mid Q$ (resp. $P + Q$), we call P and Q *threads* (resp. *branches*).

Definition 3 (Enhanced labels)**.** Let υ, υ_{L} and υ_{R} range over strings in the set $\{|_{\mathrm{L}}, |_{\mathrm{R}}, +_{\mathrm{L}}, +_{\mathrm{R}}\}^*$, *enhanced labels* are defined as

$$\theta \coloneqq \upsilon\alpha \parallel \upsilon\langle |_{\mathrm{L}}\upsilon_{\mathrm{L}}\alpha, |_{\mathrm{R}}\upsilon_{\mathrm{R}}\overline{\alpha}\rangle$$

We write E the set of enhanced labels, and define $\ell : \mathsf{E} \to \mathsf{L}$:

$$\ell(\upsilon\alpha) = \alpha \qquad \ell(\upsilon\langle |_{\mathrm{L}}\upsilon_{\mathrm{L}}\alpha, |_{\mathrm{R}}\upsilon_{\mathrm{R}}\overline{\alpha}\rangle) = \tau$$

The *proved* labeled transition system for CCS, $\xrightarrow{\theta}$, is reminded in Fig. 1.

2.2. CSSK: A "Keyed" Reversible Concurrent Calculus

CCSK captures uncontrolled reversibility using two symmetric LTS—one for forward transitions, one for backward transitions—that manipulate *keys* mark-

ing executed prefixes, to guarantee that reverting synchronizations cannot be done without both parties agreeing. We borrow the syntax to the latest paper on the topic [7], which slightly differs [7, Remark 4.2] with the classical definition [5]. However, those changes have no impact since we refrain from using CCSK's newly introduced structural congruence, but discuss it in Sect. 6.

Definition 4 (Keys, prefixes and CCSK processes)**.** Let $\mathsf{K} = \{m, n, \dots\}$ be a set of *keys*, we let k range over them. Prefixes are of the form $\alpha[k]$—we call them *keyed labels*—or α. CCSK processes are CCS processes where the prefix can also be of the form $\alpha[k]$, we let X, Y range over them.

The forward LTS for CCSK, $\xrightarrow{\alpha[k]}$, is given in Fig. 2—with key and std defined in Definition 5. The reverse LTS $\overset{\alpha[k]}{\rightsquigarrow}$ is the exact symmetric of $\xrightarrow{\alpha[k]}$ [7, Figure 2] (it can also be read from Fig. 3), and we write $X \xrightarrow{\alpha[k]}\!\!\!\!\rightarrow Y$ if $X \overset{\alpha[k]}{\rightsquigarrow} Y$ or $X \xrightarrow{\alpha[k]} Y$. For all three types of arrows, we sometimes omit the label and keys when they are not relevant, and mark with * their transitive closures. As usual, we restrict ourselves to reachable processes, defined below.

Definition 5 (Standard and reachable processes)**.** The set of keys in X, $\mathrm{key}(X)$, is defined inductively:

$$\mathrm{key}(0) = \emptyset \qquad \mathrm{key}(P + Q) = \mathrm{key}(P) \cup \mathrm{key}(Q)$$
$$\mathrm{key}(\alpha.P) = \mathrm{key}(P) \qquad \mathrm{key}(P|Q) = \mathrm{key}(P) \cup \mathrm{key}(Q)$$
$$\mathrm{key}(\alpha[k].P) = \{k\} \cup \mathrm{key}(P)$$

We say that X is *standard* and write $\mathrm{std}(X)$ iff $\mathrm{key}(X) = \emptyset$—that is, if X is a CCS process. If there exists a process O_X s.t. $\mathrm{std}(O_X)$ and $O_X \twoheadrightarrow^* X$, then X is *reachable*.

Lemma 1 (Loop lemma)**.** *For all $t : X \xrightarrow{\theta} X'$, there exists a unique $t^\bullet : X' \overset{\theta}{\rightsquigarrow} X$, and conversely. Furthermore, $(t^\bullet)^\bullet = t$.*

Proof. This was proven for CCSK at its inception [5, Prop. 5.1] and simply follows from the fact that each rule in Fig. 2 has an inverse. □

Action, Prefix and Restriction

$$\mathrm{std}(X)\ \frac{}{\alpha.X \xrightarrow{\alpha[k]} \alpha[k].X}\ \text{act.} \qquad k \neq k'\ \frac{X \xrightarrow{\beta[k]} X'}{\alpha[k'].X \xrightarrow{\beta[k]} \alpha[k'].X'}\ \text{pre.}$$

$$\alpha \notin \{a, \overline{a}\}\ \frac{X \xrightarrow{\alpha[k]} X'}{X\backslash a \xrightarrow{\alpha[k]} X'\backslash a}\ \text{res.}$$

Parallel Group

$$k \notin \mathrm{key}(Y)\ \frac{X \xrightarrow{\alpha[k]} X'}{X \mid Y \xrightarrow{\alpha[k]} X' \mid Y}\ |_{\mathrm{L}} \qquad k \notin \mathrm{key}(X)\ \frac{Y \xrightarrow{\alpha[k]} Y'}{X \mid Y \xrightarrow{\alpha[k]} X \mid Y'}\ |_{\mathrm{R}}$$

$$\frac{X \xrightarrow{\lambda[k]} X' \qquad Y \xrightarrow{\overline{\lambda}[k]} Y'}{X \mid Y \xrightarrow{\tau[k]} X' \mid Y'}\ \text{syn.}$$

Sum Group

$$\mathrm{std}(Y)\ \frac{X \xrightarrow{\alpha[k]} X'}{X + Y \xrightarrow{\alpha[k]} X' + Y}\ +_{\mathrm{L}} \qquad \mathrm{std}(X)\ \frac{Y \xrightarrow{\alpha[k]} Y'}{X + Y \xrightarrow{\alpha[k]} X + Y'}\ +_{\mathrm{R}}$$

Figure 2: Rules of the forward labeled transition system (LTS) for CCSK

3. A New Causal Semantics for CCSK

We begin our contribution with a simple definition of a proved transition system for CCSK and its causal semantics. Enhanced keyed labels let us easily define a notion of causality for CCSK with "built-in" reversibility, as *the exact same definition will be used for forward and backward transitions*. We discuss this design choice in more detail in Sect. 3.4, after proving with Lemma 3 that the past does not matter (when it is not involved).

3.1. Proved Labeled Transition System for CCSK

Enhanced *keyed* labels differ with enhanced labels (Definition 3) only in the fact that their labels must be keyed. We will abuse the notation and write them the same way:

Definition 6 (Enhanced keyed labels)**.** Let υ, υ_{L} and υ_{R} range over strings in $\{|_{\mathrm{L}}, |_{\mathrm{R}}, +_{\mathrm{L}}, +_{\mathrm{R}}\}^*$, *enhanced keyed labels* are defined as

$$\theta \coloneqq \upsilon\alpha[k] \parallel \upsilon\langle |_{\mathrm{L}}\upsilon_{\mathrm{L}}\alpha[k], |_{\mathrm{R}}\upsilon_{\mathrm{R}}\overline{\alpha}[k]\rangle$$

We write E the set of enhanced keyed labels, and define $\ell : \mathsf{E} \to \mathsf{L}$ and $\not\kappa : \mathsf{E} \to \mathsf{K}$:

$$\ell(\upsilon\alpha[k]) = \alpha \qquad \ell(\upsilon\langle |_{\mathrm{L}}\upsilon_{\mathrm{L}}\alpha[k], |_{\mathrm{R}}\upsilon_{\mathrm{R}}\overline{\alpha}[k]\rangle) = \tau$$
$$\not\kappa(\upsilon\alpha[k]) = k \qquad \not\kappa(\upsilon\langle |_{\mathrm{L}}\upsilon_{\mathrm{L}}\alpha[k], |_{\mathrm{R}}\upsilon_{\mathrm{R}}\overline{\alpha}[k]\rangle) = k$$

We present in Fig. 3 the rules for the *proved* forward and backward LTS for CCSK. The rules $|_{\mathrm{R}}$, $|^{\bullet}_{\mathrm{R}}$, $+_{\mathrm{R}}$ and $+^{\bullet}_{\mathrm{R}}$ are omitted but can easily be inferred. This LTS has its derivation in bijection with CCSK's original LTS:

Lemma 2 (Adequacy of the proved labeled transition system)**.** *The transition $X \xrightarrow{\alpha[k]} X'$ can be derived using Fig. 2 iff $X \xrightarrow{\theta} X'$ with $\not\kappa(\theta) = m$ and $\ell(\theta) = \alpha$ can be derived using Fig. 3.*

Proof. The proof is by induction on the length of the derivation: since the only axiom rules (act. and act$^{\bullet}$.) are identical, it easily follow by inspection of the remaining rules of Fig 2 and 3. □

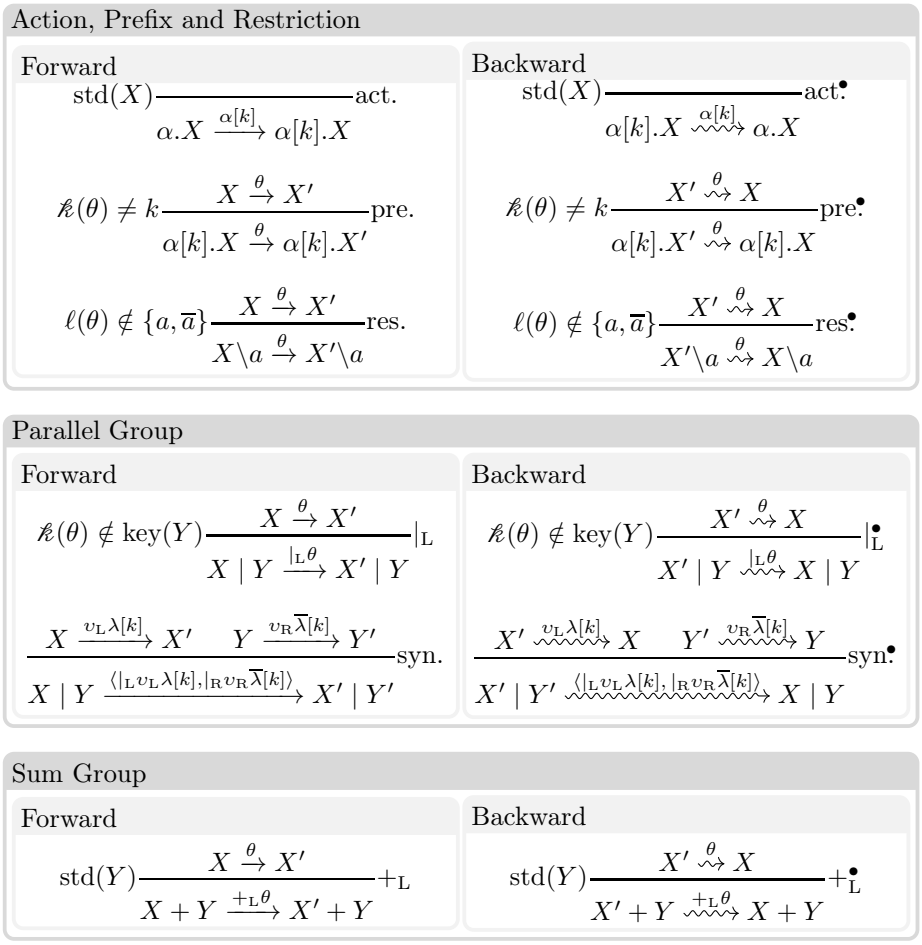


Figure 3: Rules of the *proved* LTS for CCSK

Definition 7 (Transitions and traces)**.** In a *transition* $t : X \xrightarrow{\theta} X'$, X is the *source*, and X' is the *target* of t. Two transitions are *coinitial* (resp. *cofinal*) if they have the same source (resp. target). Transitions t_1 and t_2 are *composable* if the target of t_1 is the source of t_2. Two transitions are *concomitant*[5] if they are either coinitial or composable.

[5]For lack of a more canonical term. *Adjacent* was also suggested by Ivan Lanese, and "joinable" was also used [5, p. 84], but for concomitant *concurrent* transitions.

A sequence of pairwise composable transitions $t_1; \cdots ; t_n$ is called a *trace*, denoted T, and ϵ is the empty trace.

Note that following Lemma 2, the Loop lemma trivially holds for the system presented in Fig. 3, and we write similarly $t^\bullet : X \xrightarrow{\theta} X'$ the reverse of $t : X' \overset{\theta}{\rightsquigarrow} X$, and reciprocally.

3.2. The Past Does Not Matter (When It Is Not Involved)

In Sect. 4, we will need to use the fact that the pre. rule is transparent from the perspective of enhanced keyed labels, as no "memory" of its usage is stored in the label of the transition. This lets us show that as long as a transition does not reverse a particular action, its presence or absence does not affect derivability or the label (Lemma 3). To make this more formal, we begin by introducing a function that "removes" a keyed label.

Definition 8 (Removal function)**.** Given a label α and a key k, we define *the removal function* $\mathrm{rm}_{\alpha[k]}$ by

$$\mathrm{rm}_{\alpha[k]}(0) = 0 \qquad \mathrm{rm}_{\alpha[k]}(X \mid Y) = \mathrm{rm}_{\alpha[k]}(X) \mid \mathrm{rm}_{\alpha[k]}(Y)$$

$$\mathrm{rm}_{\alpha[k]}(\beta.X) = \beta.X \qquad \mathrm{rm}_{\alpha[k]}(X + Y) = \mathrm{rm}_{\alpha[k]}(X) + \mathrm{rm}_{\alpha[k]}(Y)$$

$$\mathrm{rm}_{\alpha[k]}(X\backslash a) = (\mathrm{rm}_{\alpha[k]}\, X)\backslash a$$

$$\mathrm{rm}_{\alpha[k]}(\beta[k'].X) = \begin{cases} X & \text{if } \alpha = \beta \text{ and } k = k' \\ \beta[k'].\, \mathrm{rm}_{\alpha[k]}(X) & \text{otherwise} \end{cases}$$

We define *the removal function of a label and its complement* by

$$\mathrm{rm}_k^\alpha = \begin{cases} \mathrm{rm}_{\alpha[k]} \circ \mathrm{rm}_{\overline{\alpha}[k]} & \text{if } \alpha \in \mathsf{L}\backslash\{\tau\}, \\ \mathrm{rm}_{\tau[k]} & \text{otherwise} \end{cases}.$$

The function $\mathrm{rm}_{\alpha[k]}$ simply looks for an occurrence of $\alpha[k]$ and removes it: as there is at most one such occurence in a reachable processs, there is no need for a recursive call when it is found. This function preserves derivability of transitions that do not involve the key removed:

Lemma 3. *For all X, α, k, and θ with $\not\kappa(\theta) \neq k$, if $k \notin \mathrm{key}(\mathrm{rm}_k^\alpha(X))$,*[6] *then* $X \overset{\theta}{\twoheadrightarrow} Y \iff \mathrm{rm}_k^\alpha(X) \overset{\theta}{\twoheadrightarrow} \mathrm{rm}_k^\alpha(Y)$.

Proof. We reason by the number of occurrences of k in X, which is the same as the number of occurences of k in Y, since $\not\kappa(\theta) \neq k$. As keys occur at most twice, attached to complementary names, in reachable processes [7, Lemma 3.4], we know that we have only three cases to consider: 0, 1 and 2.

0 **occurence** Then there is nothing to prove, as $\mathrm{rm}_k^\alpha(X) = X$ and $\mathrm{rm}_k^\alpha(Y) = Y$.

1 **occurence** Since $k \notin \mathrm{key}(\mathrm{rm}_k^\alpha(X))$, we know that the key k is attached to α or $\overline{\alpha}$. We suppose without loss of generality that it is attached to α, and start by proving the left-to-right direction of the implication. This means that the derivation of $X \overset{\theta}{\twoheadrightarrow} Y$ is of the form

$$\dfrac{\begin{array}{c}\vdots\,\pi_1\\ X' \xrightarrow{\theta'} Y'\end{array}}{\begin{array}{c}\alpha[k].X' \xrightarrow{\theta'} \alpha[k].Y'\\ \vdots\,\pi_2\\ X \xrightarrow{\theta} Y\end{array}}\mathrm{pre.} \quad \text{or} \quad \dfrac{\begin{array}{c}\vdots\,\pi_1^\bullet\\ X' \overset{\theta'}{\rightsquigarrow} Y'\end{array}}{\begin{array}{c}\alpha[k].X' \overset{\theta'}{\rightsquigarrow} \alpha[k].Y'\\ \vdots\,\pi_2^\bullet\\ X \overset{\theta}{\rightsquigarrow} Y\end{array}}\mathrm{pre}^\bullet$$

depending on the direction of the transition.

To obtain the derivation of $\mathrm{rm}_k^\alpha(X) \xrightarrow{\theta} \mathrm{rm}_k^\alpha(Y)$, it suffices to "skip" the pre. rule: since it does not alter the enhanced keyed label θ', composing π_1 and π_2 (where $\alpha[k].X'$ and $\alpha[k].Y'$ have been replaced by X' and Y') yields a correct derivation of $\mathrm{rm}_k^\alpha(X) \xrightarrow{\theta} \mathrm{rm}_k^\alpha(Y)$ (where k does not occur, since it was not occuring in π_1 or π_2). The same reasoning can be used to obtain the derivation of $\mathrm{rm}_k^\alpha(X) \overset{\theta}{\rightsquigarrow} \mathrm{rm}_k^\alpha(Y)$.

[6] This cumbersome condition is here to prevent k from occurring in X attached to a different label. In practice, we will always remove $\alpha[k]$ from processes where we know it occurs, so that this condition will always be vacuously true, since the same key cannot be attached to labels that are not complement of each others [7, Lemma 3.4].

Action

$$\alpha[k] \lessdot \theta \quad \forall \alpha, k, \theta$$

Sum Group

$$+_{\mathrm{L}}\theta \lessdot +_{\mathrm{R}}\theta'$$

$$+_{\mathrm{R}}\theta \lessdot +_{\mathrm{L}}\theta'$$

$$+_d\theta \lessdot +_d\theta' \quad \text{if } \theta \lessdot \theta'$$

Parallel Group

$$|_d\theta \lessdot |_d\theta' \quad \text{if } \theta \lessdot \theta'$$

$$\langle \theta_{\mathrm{L}}, \theta_{\mathrm{R}} \rangle \lessdot \theta \quad \text{if } \exists d \text{ s.t.} \theta_d \lessdot \theta$$

$$\theta \lessdot \langle \theta_{\mathrm{L}}, \theta_{\mathrm{R}} \rangle \quad \text{if } \exists d \text{ s.t.} \theta \lessdot \theta_d$$

$$\langle \theta_{\mathrm{L}}, \theta_{\mathrm{R}} \rangle \lessdot \langle \theta'_{\mathrm{L}}, \theta'_{\mathrm{R}} \rangle \quad \text{if } \exists d \text{ s.t.} \theta_d \lessdot \theta'_d$$

For $d \in \{\mathrm{L}, \mathrm{R}\}$.

Figure 4: Dependency Relation on Enhanced Keyed Labels

For the right-to-left direction of the implication, it suffices to introduce a pre. or pre$^\bullet_\bullet$ rule in the derivation of $\mathrm{rm}^\alpha_k(X) \xrightarrow{\theta}\!\!\!\!\rightarrow \mathrm{rm}^\alpha_k(Y)$. We know by hypothesis that $k \notin \mathrm{key}(\mathrm{rm}^\alpha_k(X))$, and since $\not\kappa(\theta) \neq k$, $k \notin \mathrm{key}(\mathrm{rm}^\alpha_k(Y))$ as well. Hence, the side condition of pre. or pre$^\bullet_\bullet$ is always met, and the rule can be applied at any point in the derivation to obtain the desired transition.

2 **occurences** Then it suffices to apply the reasoning above twice, to the pre. or pre$^\bullet_\bullet$ rules that introduce $\alpha[k]$ and $\overline{\alpha}[k]$, to obtain the desired transition. □

3.3. Dependency and Concurrency for CCSK

Definition 9 (Dependency relation)**.** The *dependency relation* $\lessdot$ on enhanced keyed labels is induced by the axioms of Fig. 4.

Claim 1. *The dependency relation $\lessdot$ is reflexive, neither symmetric nor anti-symmetric, and not transitive.*

Proof. We prove each property separately:

Reflexive This proceeds by induction on the structure of θ: if θ is $\alpha[k]$, then it is immediate by definition. Otherwise, it proceeds easily by induction on the main operator of θ.

Not symmetric For instance, $a[m] \lessdot |_{\mathrm{L}} b[n]$, but $|_{\mathrm{L}} b[n] \lessdot a[m]$ does not hold.

Not antisymmetric For instance, $a[m] \lessdot b[n]$ and $b[n] \lessdot a[m]$ both hold, and yet $a[m] \neq b[n]$.

Not transitive For instance,

$$|_{\mathrm{L}}a[n_1] \lessdot \langle |_{\mathrm{L}}b[m], |_{\mathrm{R}}\overline{b}[m] \rangle \qquad \text{and} \qquad \langle |_{\mathrm{L}}b[m], |_{\mathrm{R}}\overline{b}[m] \rangle \lessdot |_{\mathrm{R}}c[n_2]$$

both hold, and yet $|_{\mathrm{L}}a[n_1] \lessdot |_{\mathrm{R}}c[n_2]$ does not hold. □

Definition 10 (Concurrency relation)**.** Two enhanced keyed labels θ_1 and θ_2 are *concurrent*, written $\theta_1 \smile \theta_2$ iff neither $\theta_1 \lessdot \theta_2$ nor $\theta_2 \lessdot \theta_1$.

Claim 2. *The concurrency relation $\smile$ is irreflexive and symmetric.*

Proof. Irreflexivity follows from the fact that $\lessdot$ is reflexive, symmetry is immediate by definition. □

Definition 11 (Composable concurrency)**.** Let $t_1 : X_1 \xrightarrow{\theta_1}\!\!\!\!\rightarrow X_2$ and $t_2 : X_2 \xrightarrow{\theta_2}\!\!\!\!\rightarrow X_3$ be two composable transitions, t_1 *is concurrent with* t_2 *($t_1 \smile_c t_2$)* iff $\theta_1 \smile \theta_2$.

Coinitial concurrency is then defined using composable concurrency and the Loop lemma:

Definition 12 (Coinitial concurrency)**.** Let $t_1 : X \xrightarrow{\theta_1}\!\!\!\!\rightarrow Y_1$ and $t_2 : X \xrightarrow{\theta_2}\!\!\!\!\rightarrow Y_2$ be two coinitial transitions, t_1 *is concurrent with* t_2 *($t_1 \smile_i t_2$)* iff $t_1^\bullet \smile_c t_2$.

To our knowledge, this is the first time coinitial concurrency is defined *from* composable concurrency (not just concomitantly), hence enforcing the (Correctness of Concurrencies). While the axiomatic approach discussed coinitial concurrency [11, Section 5], it primarily studied independence relations that could be defined in any way, and did not connect these two notions of concurrency. However, our system is somehow a degenerate case, since both concurrencies are actually a property of *the enhanced key labels*, and not of the traces:

Lemma 4 (Concurrencies are trace-insensible)**.** *For all* $t_1 : X_1 \xrightarrow{\theta_1}\!\!\!\!\rightarrow X_2$ *and* $t_3 : X_1' \xrightarrow{\theta_1}\!\!\!\!\rightarrow X_2'$,

1. *For all* $t_2 : X_2 \xrightarrow{\theta_2} X_3$ *and* $t_4 : X_2' \xrightarrow{\theta_2} X_3'$, $t_1 \smile_c t_2 \iff t_3 \smile_c t_4$.

2. *For all* $t_2 : X_1 \xrightarrow{\theta_2} X_3$ *and* $t_4 : X_1' \xrightarrow{\theta_2} X_3'$, $t_1 \smile_i t_2 \iff t_3 \smile_i t_4$.

Proof. The proof is immediate:

1. $t_1 \smile_c t_2 \iff \theta_1 \smile \theta_2 \iff t_3 \smile_c t_4$ by Definitions 11 and 12, and
2. $t_1 \smile_i t_2 \iff \theta_1 \smile \theta_2 \iff t_3 \smile_i t_4$ by Definitions 11 and 12. □

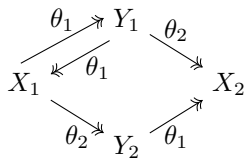
Lemma 4 makes it clear that all the needed information is in the labels, and that the actual processes (or their actual traces) involved are irrelevant. As a corollary, we can ease the notation:

Notation 1. For t_1 and t_2 two concomitant traces with labels θ_1 and θ_2, we will simply write $\theta_1 \smile \theta_2$ for $t_1 \smile_c t_2$ or $t_1 \smile_i t_2$.

We can also obtain Coinitial Propagation of Independence [11, Defitinion 4.2] as a simple corollary:

Corollary 1 (Coinitial Propagation of Independence (CPI))**.** *For all* $t_1 : X_1 \xrightarrow{\theta_1} Y_1$, $t_2 : X_1 \xrightarrow{\theta_2} Y_2$, $t_3 : Y_1 \xrightarrow{\theta_2} X_2$ *and* $t_4 : Y_2 \xrightarrow{\theta_1} X_2$ *with* $t_1 \smile_i t_2$ *then* $t_3 \smile_c t_1^\bullet$.

In picture, we have



Proof. This is immediate:

$$t_1 \smile_i t_2 \implies \theta_1 \smile \theta_2 \implies t_3 \smile_c t_1^\bullet$$

□

Example 1. Consider the following trace, dependencies, and concurrent enhanced keyed labels:

$$
\begin{aligned}
&(a.\overline{b}) \mid (b+c)\\
&\xrightarrow{|_{\mathrm{L}}a[m]} a[m].\overline{b} \mid b+c\\
&\xrightarrow{|_{\mathrm{L}}\overline{b}[n]} a[m].\overline{b}[n] \mid b+c\\
&\xrightarrow{|_{\mathrm{R}}+_{\mathrm{R}}c[n']} a[m].\overline{b}[n] \mid b+c[n']\\
&\rightsquigarrow a[m].\overline{b} \mid b+c[n']\\
&\rightsquigarrow a[m].\overline{b} \mid b+c\\
&\xrightarrow{\langle |_{\mathrm{L}}\overline{b}[n], |_{\mathrm{R}}+_{\mathrm{L}}b[n]\rangle} a[m].\overline{b}[n] \mid b[n]+c
\end{aligned}
$$

And we have, e.g.,

$$|_{\mathrm{L}}a[m] \lessdot |_{\mathrm{L}}\overline{b}[n]$$

as $a[m] \lessdot \overline{b}[n]$,

$$|_{\mathrm{R}}+_{\mathrm{R}}c[n'] \lessdot \langle |_{\mathrm{L}}\overline{b}[n], |_{\mathrm{R}}+_{\mathrm{L}}b[n]\rangle$$

as $+_{\mathrm{R}}c[n'] \lessdot +_{\mathrm{L}}b[n]$, and

$$|_{\mathrm{L}}\overline{b}[n] \smile |_{\mathrm{R}}+_{\mathrm{R}}c[n']$$

since labels prefixed by $|_{\mathrm{L}}$ and $|_{\mathrm{R}}$ are never causes of each others.

3.4. Discussion

This may be a good moment to pause and reflect on this definition of concurrency we will be using. Originally, in CCS [25, p. 311], the dependency relation $\lessdot$ on labels had to be parametrized by the trace, since a transition with a label θ_1 could not be the cause of a transition with a label θ_2 unless it happened *before* it.

We would then have [25, Definition 3], given a trace T,

$$\theta_1 \lessdot_T \theta_2 \iff \begin{cases} \theta_1 \lessdot \theta_2 \\ \theta_1 \text{ happens before } \theta_2 \text{ in } T \end{cases}$$

Without this constraint, one could e.g., decide that b is a dependency of a in the CCS trace $a.b \xrightarrow{a} b \xrightarrow{b} 0$, since, after all, $b \preccurlyeq a$. But of course this would not make much sense, due to the temporal order of those transitions. Hence, $\lessdot_T$ would be considered instead of $\lessdot$, and the *causal dependency* $\preccurlyeq_T$ would be defined as the symmetric and transitive closure of $\lessdot_T$ [32, Definition 2.2][25, Definition 3].

This transitive closure was important, too. Without it, you could for instance conclude that in a CCS trace [25, footnote 2]

$$b.a|\overline{a}.c \xrightarrow{|_{\mathrm{L}}b} a|\overline{a}.c \xrightarrow{\langle |_{\mathrm{L}}a, |_{\mathrm{R}}\overline{a}\rangle} 0|c \xrightarrow{|_{\mathrm{R}}c} 0|0 \tag{4}$$

it was the case that the transitions whose labels are $|_{\mathrm{L}}b$ and $|_{\mathrm{R}}c$ are independent, since neither $|_{\mathrm{L}}b \lessdot |_{\mathrm{R}}c$ nor $|_{\mathrm{R}}c \lessdot |_{\mathrm{L}}b$ would hold.

But, in our definition, we do not parametrize the dependency relation by the trace, and we do not need its transitive closure. Why? There are two reasons, and neither are caused by reversibility, curiously enough.

The first one is that we are interested in concomitant (Definition 7) transitions. This is fairly standard, as the diamonds are concerned only with that types of transitions, and as only "local" permutations will be considered. Wondering wether $|_{\mathrm{L}}b$ and $|_{\mathrm{R}}c$ are independent in (4) makes no sense, as they are not concomitant, and will never be, since neither can be permuted with the transition labeled $\langle |_{\mathrm{L}}a, |_{\mathrm{R}}\overline{a}\rangle$.

The second reason is that dependencies is simply a tool to define concurrency, and we are not focused on capturing "the right" notion. It is acceptable if we consider a to be a dependency of b and b to be a dependency of a in $a.b \xrightarrow{a} b \xrightarrow{b} 0$: what matters is that we detect that there is *some dependency* between those two traces, e.g., that they are not concurrent. Stated differently, the temporal order is needed for causality but not for concurrency, as we have $\theta_1 \smile \theta_2$ iff neither $\theta_1 \lessdot \theta_2$ *nor* $\theta_2 \lessdot \theta_1$ (Definition 10, this is also the case in one of our inspiration paper [25, Definition 4]). This disjunction allows to discard the temporal order: suddenly, we do not care about which happened first.

As we wrote, those design choices are *not* caused by reversibility, but, as it turns out, they play really well together. In CCSK, when the CCS trace $a.b \xrightarrow{a} b \xrightarrow{b} 0$ is executed and then reversed, we obtain:

$$a.b \xrightarrow{a[m]} a[m].b \xrightarrow{b[n]} a[m].b[n] \tag{5}$$

$$a[m].b[n] \xrightarrow{\!\!\!\!\!\!\rightsquigarrow\!\!\!\!\!\!}{}^{b[n]} a[m].b \xrightarrow{\!\!\!\!\!\!\rightsquigarrow\!\!\!\!\!\!}{}^{a[m]} a.b \tag{6}$$

Again, one could argue that $b[n] \lessdot a[m]$ makes no sense for (5) but is correct w.r.t. (6), due to the temporal order. We explained why this does not matter when the focus in on concurrency, but it is also interesting to remark that deciding that $b[n]$ is a dependency of $a[m]$ regardless of their temporal order *makes our definition of dependency independent from the direction of the transition.*

We can also observe that our dependency relation matches the forward-only definition for action and parallel composition, but not for sum: while the original system [25, Definition 2] requires only $+_d\theta \lessdot \theta'$ if $\theta \lessdot \theta'$, this definition would not capture faithfully the dependencies in our system where the sum operator is preserved after a reduction. This is also the reason why our dependency relation is reflexive, while their is not (Claim 1).

All in all, our design choices allow to use only one definition of dependency to define concurrencie*s*, instead of having to take the transitive closure of a tertiary conflict or causality relation (as briefly discussed in Sect. 5.1.2). Our definitions are also direction-insensitive[7] and "identifiers agnostic": by that we mean that any identifying mechanism, not only the key mechanism of CCSK, could work with it, as we discuss further in Sect. 5.

4. Diamonds, Squares and Consistency

4.1. Preliminary: Decomposing Transitions

To prove the required properties, we need an intuitive and straightforward lemma (Lemma 5) that decomposes a concurrent trace involving two threads into one trace involving one thread while maintaining concurrency. That is, we prove that a trace of the form

$$X \mid Y \xrightarrow{|_{\mathrm{L}}\theta} X' \mid Y \xrightarrow{|_{\mathrm{L}}\theta'} X'' \mid Y$$

with $|_{\mathrm{L}}\theta \smile |_{\mathrm{L}}\theta'$ can be decomposed into a trace

$$X \xrightarrow{\theta} X' \xrightarrow{\theta'} X''$$

with $\theta \smile \theta'$. A similar lemma is also needed to decompose traces involving two branches Lemma 6). In both cases, the lemma is cumbersome to spell out, but easy to prove by simple case analysis.

[7] As we discuss in Sect. 5, many existing definitions proceeds by case ("If transitions are forward, ..., if they are backward, ..."), sometimes "forgetting" about transitions of opposite directions (Sect. 5.2.3). Our definition does not make such distinction, and is adequate for any combination of forward and bacward transitions.

Lemma 5 (Decomposing concurrent parallel transitions)**.** *Let* $i \in \{1,2\}$ *and* $\theta_i \in \{|_{\mathrm{L}}\theta_i', |_{\mathrm{R}}\theta_i'', \langle |_{\mathrm{L}}\theta_i', |_{\mathrm{R}}\theta_i''\rangle\}$*, define* the left projection on enhanced keyed labels π_{L} *as:*

$$\pi_{\mathrm{L}}(\theta_i) = \begin{cases} \theta_i' & \text{if } \theta = |_{\mathrm{L}}\theta_i' \text{ or if } \theta_i = \langle |_{\mathrm{L}}\theta_i', |_{\mathrm{R}}\theta_i''\rangle) \\ \textit{undefined} & \textit{otherwise} \end{cases}$$

and extend it to processes as

$$\pi_{\mathrm{L}}(X) = \begin{cases} X_{\mathrm{L}} & \text{if } X = X_{\mathrm{L}} \mid X_{\mathrm{R}} \\ \textit{undefined} & \textit{otherwise} \end{cases}$$

We define similarly the right projection on keyed labels π_{R} *and extend it to processes.*

Whenever $T : X_{\mathrm{L}} \mid X_{\mathrm{R}} \xrightarrow{\theta_1}\!\!\!\!\rightarrow Y_{\mathrm{L}} \mid Y_{\mathrm{R}} \xrightarrow{\theta_2}\!\!\!\!\rightarrow Z_{\mathrm{L}} \mid Z_{\mathrm{R}}$ *with* $\theta_1 \smile \theta_2$*, then for* $d \in \{\mathrm{L}, \mathrm{R}\}$*, if* $\pi_d(\theta_1)$ *and* $\pi_d(\theta_2)$ *are both defined, then there exist a trace*

$$\pi_d(T) : \pi_d(X_{\mathrm{L}} \mid X_{\mathrm{R}}) \xrightarrow{\pi_d(\theta_1)}\!\!\!\!\rightarrow \pi_d(Y_{\mathrm{L}} \mid Y_{\mathrm{R}}) \xrightarrow{\pi_d(\theta_2)}\!\!\!\!\rightarrow \pi_d(Z_{\mathrm{L}} \mid Z_{\mathrm{R}})$$

and $\pi_d(\theta_1) \smile \pi_d(\theta_2)$*.*

Proof. The trace $\pi_d(T)$ exists by virtue of the rule $|_d$, syn. or their reverses. What remains to prove is that $\pi_d(\theta_1) \smile \pi_d(\theta_2)$ holds.

The proof is by case on θ_1 and θ_2, but always follows the same pattern. As we know that both $\pi_d(\theta_1)$ and $\pi_d(\theta_2)$ need to be defined, there are 7 cases:

$$\begin{array}{llll} \theta_1 = |_{\mathrm{L}}\theta_1' \text{ and } \theta_2 = |_{\mathrm{L}}\theta_2' & & \theta_1 = |_{\mathrm{R}}\theta_1' \text{ and } \theta_2 = |_{\mathrm{R}}\theta_2' \\ \theta_1 = |_{\mathrm{R}}\theta_1' \text{ and } \theta_2 = \langle |_{\mathrm{L}}\theta_2', |_{\mathrm{R}}\theta_2''\rangle & & \theta_1 = \langle |_{\mathrm{L}}\theta_1', |_{\mathrm{R}}\theta_1''\rangle \text{ and } \theta_2 = |_{\mathrm{R}}\theta_2' \\ \theta_1 = |_{\mathrm{L}}\theta_1' \text{ and } \theta_2 = \langle |_{\mathrm{L}}\theta_2', |_{\mathrm{R}}\theta_2''\rangle & & \theta_1 = \langle |_{\mathrm{L}}\theta_1', |_{\mathrm{R}}\theta_1''\rangle \text{ and } \theta_2 = |_{\mathrm{L}}\theta_2' \\ \theta_1 = \langle |_{\mathrm{L}}\theta_1', |_{\mathrm{R}}\theta_1''\rangle \text{ and } \theta_2 = \langle |_{\mathrm{L}}\theta_2', |_{\mathrm{R}}\theta_2''\rangle & & \end{array}$$

By symmetry, we can bring this number down to three:

(case letter)	**a)**	**b)**	**c)**
θ_1	$\|_{\mathrm{L}}\theta_1'$	$\langle \|_{\mathrm{L}}\theta_1', \|_{\mathrm{R}}\theta_1''\rangle$	$\langle \|_{\mathrm{L}}\theta_1', \|_{\mathrm{R}}\theta_1''\rangle\}$
θ_2	$\|_{\mathrm{L}}\theta_2'$	$\|_{\mathrm{L}}\theta_2'$	$\langle \|_{\mathrm{L}}\theta_2', \|_{\mathrm{R}}\theta_2''\rangle\}$

In each case, assume $\pi_{\mathrm{L}}(\theta_1) = \theta'_1 \smile \theta'_2 = \pi_{\mathrm{L}}(\theta_2)$ does not hold. Then it must be the case that either $\theta'_1 \lessdot \theta'_2$ or $\theta'_2 \lessdot \theta'_1$, and since both can be treated the same way thanks to symmetry, we only need to detail the following three cases:

a) If $\theta'_1 \lessdot \theta'_2$, then it is immediate that $\theta_1 = |_{\mathrm{L}}\theta'_1 \lessdot |_{\mathrm{L}}\theta'_2 = \theta_2$, contradicting $\theta_1 \smile \theta_2$.

b) If $\theta'_1 \lessdot \theta'_2$, then $|_{\mathrm{L}}\theta'_1 \lessdot |_{\mathrm{L}}\theta'_2$ and $\langle |_{\mathrm{L}}\theta'_1, |_{\mathrm{R}}\theta''_1\rangle \lessdot |_{\mathrm{L}}\theta'_2$, from which we can deduce $\theta_1 \lessdot \theta_2$, contradicting $\theta_1 \smile \theta_2$.

c) If $\theta'_1 \lessdot \theta'_2$, then $|_{\mathrm{L}}\theta'_1 \lessdot |_{\mathrm{L}}\theta'_2$ and $\langle |_{\mathrm{L}}\theta'_1, |_{\mathrm{R}}\theta''_1\rangle \lessdot \langle |_{\mathrm{L}}\theta'_2, |_{\mathrm{R}}\theta'_2\rangle$, from which we can deduce $\theta_1 \lessdot \theta_2$, contradicting $\theta_1 \smile \theta_2$.

Hence, in all cases, assuming that $\pi_d(\theta_1) \smile \pi_d(\theta_2)$ does not hold leads to a contradiction. □

Lemma 6 (Decomposing concurrent sum transitions)**.** *Let $i \in \{1,2\}$ and $\theta_i \in \{+_{\mathrm{L}}\theta'_i, +_{\mathrm{R}}\theta''_i\}$, define* the left summand of enhanced keyed labels π_{L} *as:*

$$\rho_{\mathrm{L}}(\theta_i) = \begin{cases} \theta'_i & \text{if } \theta = +_{\mathrm{L}}\theta'_i \\ \textit{undefined} & \textit{otherwise} \end{cases}$$

and extend it to processes as

$$\rho_{\mathrm{L}}(X) = \begin{cases} X_{\mathrm{L}} & \text{if } X = X_{\mathrm{L}} + X_{\mathrm{R}} \\ \textit{undefined} & \textit{otherwise} \end{cases}$$

We define similarly the right summand of keyed labels ρ_{R} *and extend it to processes.*

Whenever $T : X_{\mathrm{L}} + X_{\mathrm{R}} \xrightarrow{\theta_1}\!\!\!\!\rightarrow Y_{\mathrm{L}} + Y_{\mathrm{R}} \xrightarrow{\theta_2}\!\!\!\!\rightarrow Z_{\mathrm{L}} + Z_{\mathrm{R}}$ with $\theta_1 \smile \theta_2$, then for $d \in \{\mathrm{L},\mathrm{R}\}$, if $\rho_d(\theta_1)$ and $\rho_d(\theta_2)$ are both defined, then there exists a trace

$$\rho_d(T) : \rho_d(X_{\mathrm{L}} + X_{\mathrm{R}}) \xrightarrow{\rho_d(\theta_1)}\!\!\!\!\rightarrow \rho_d(Y_{\mathrm{L}} + Y_{\mathrm{R}}) \xrightarrow{\rho_d(\theta_2)}\!\!\!\!\rightarrow \rho_d(Z_{\mathrm{L}} + Z_{\mathrm{R}})$$

and $\rho_d(\theta_1) \smile \rho_d(\theta_2)$.

Proof. The trace $\rho_d(T)$ exists by virtue of the rule $+_d$ or its reverse. What remains to prove is that $\rho_d(\theta_1) \smile \rho_d(\theta_2)$ holds.

The proof is by case on θ_1 and θ_2, but always follows the same pattern. As we know that both $\rho_d(\theta_1)$ and $\rho_d(\theta_2)$ need to be defined, there are 2 cases:

θ_1	$+_{\mathrm{L}}\theta'_1$	$+_{\mathrm{R}}\theta'_1$
θ_2	$+_{\mathrm{L}}\theta'_2$	$+_{\mathrm{R}}\theta'_2$

For $d \in \{\mathrm{L}, \mathrm{R}\}$, assume $\rho_d(\theta_1) = \theta'_1 \smile \theta'_2 = \rho_d(\theta_2)$ does not hold, then it is immediate to note that $\theta_1 \smile \theta_2$ cannot hold either, a contradiction. □

4.2. Diamonds and Squares: Concurrency in Action

Our goal in this section is to prove our Main Theorem, that states that for all $X \xrightarrow{\theta_1}\!\!\!\!\rightarrow X_1$ and $X \xrightarrow{\theta_2}\!\!\!\!\rightarrow X_2$ with $\theta_1 \smile \theta_2$, there exist $X_1 \xrightarrow{\theta_2}\!\!\!\!\rightarrow Y$ and $X_2 \xrightarrow{\theta_1}\!\!\!\!\rightarrow Y$. This statements, because we enjoy the Loop lemma and enforce the (Correctness of Concurrencies), is equivalent to stating that we enjoy all the diamonds listed in Table 2. This is one of the main technical goal of this paper, the other interesting properties, discussed in Sect. 4.3, following almost for free thanks to the the axiomatic approach to reversible computation [11].

To obtain this result, we first prove one of the (Sideways Diamonds) (actually, *the* (Sidways Diamond)) with Lemma 7, one of the (Reverse Diamonds) (Lemma 8) and finally one of the Square Properties (Lemma 9). Our Main Theorem is then a trivial consequence of those lemmas. Interestingly, all three proofs are almost identical, except for some very subtle points that we highlight.

Lemma 7 (Sideways diamond)**.** *For all $X \xrightarrow{\theta_1} X_1 \xrightarrow{\theta_2} Y$ with $\theta_1 \smile \theta_2$, there exists X_2 s.t. $X \xrightarrow{\theta_2} X_2 \xrightarrow{\theta_1} Y$.*

In short, the proof proceeds by induction on the length of the deduction for the derivation for $X \xrightarrow{\theta_1} X_1$, using Lemmas 5 and 6 to enable the induction hypothesis if θ_1 is not a prefix. The proof requires a particular care when X is not standard, more particularly if the last rule is pre., but Lemma 3 provides just what is needed to deal with this case.

Proof. The proof proceeds by induction on the length of the deduction for the derivation for $X \xrightarrow{\theta_1} X_1$.

Length 1 In this case, the derivation is a single application of act., and θ_1 is of the form $\alpha[k]$. But $\alpha[k] \smile \theta_2$ cannot hold, as $\alpha[k] \lessdot \theta_2$ always holds, and this case is vacuously true.

Length > 1 We proceed by case on the last rule.

pre. There exists α, k, X' and X_1' s.t. $X = \alpha[k].X' \xrightarrow{\theta_1} \alpha[k].X_1' = X_1$ and $\not\kappa(\theta_1) \neq k$. As $\alpha[k].X_1' \xrightarrow{\theta_2} Y$ we know that $\not\kappa(\theta_2) \neq k$ [7, Lemma 3.4]. Furthermore, since k occurs attached to α in X_1 and since X_1 makes a *forward* transition to reach Y, we know that $k \notin \mathrm{key}(\mathrm{rm}_k^\alpha(X_1)) \cup \mathrm{key}(\mathrm{rm}_k^\alpha(Y))$. Hence, we can apply Lemma 3 twice to obtain

$$\mathrm{rm}_k^\alpha(\alpha[k].X') = X' \xrightarrow{\theta_1} \mathrm{rm}_k^\alpha(\alpha[k].X_1') = X_1' \xrightarrow{\theta_2} \mathrm{rm}_k^\alpha(Y)$$

As $\theta_1 \smile \theta_2$ by hypothesis, we can use the induction hypothesis to obtain that there exists X_2 s.t. $X' \xrightarrow{\theta_2} X_2 \xrightarrow{\theta_1} \mathrm{rm}_k^\alpha(Y)$. Since $\not\kappa(\theta_2) \neq k$, we can append pre. to the derivation of $X' \xrightarrow{\theta_2} X_2$ to obtain $\alpha[k].X' = X \xrightarrow{\theta_2} \alpha[k].X_2$. Using Lemma 3 one last time, we obtain that $\mathrm{rm}_k^\alpha(\alpha[k].X_2) = X_2 \xrightarrow{\theta_1} \mathrm{rm}_k^\alpha(Y)$ implies $\alpha[k].X_2 \xrightarrow{\theta_1} Y$, which concludes this case.

res. This is immediate by induction hypothesis.

$|_\mathrm{L}$ There exists X_L, X_R, θ_1', X_{1_L}, and Y_L, Y_R s.t. $X \xrightarrow{\theta_1} X_1 \xrightarrow{\theta_2} Y$ is

$$X_\mathrm{L} \mid X_\mathrm{R} \xrightarrow{|_\mathrm{L}\theta_1'} X_{1_\mathrm{L}} \mid X_\mathrm{R} \xrightarrow{\theta_2} Y_\mathrm{L} \mid Y_\mathrm{R}.$$

Then, $X_\mathrm{L} \xrightarrow{\theta_1'} X_{1_\mathrm{L}}$ and the proof proceeds by case on θ_2:

θ_2 **is** $|_\mathrm{R}\theta_2'$ Then $X_\mathrm{R} \xrightarrow{\theta_2'} Y_\mathrm{R}$, $X_{1_\mathrm{L}} = Y_\mathrm{L}$ and the occurrences of the rules $|_\mathrm{L}$ and $|_\mathrm{R}$ can be swapped to obtain

$$X_\mathrm{L} \mid X_\mathrm{R} \xrightarrow{|_\mathrm{R}\theta_2'} X_\mathrm{L} \mid Y_\mathrm{R} \xrightarrow{|_\mathrm{L}\theta_1'} Y_\mathrm{L} \mid Y_\mathrm{R}.$$

θ_2 **is** $|_\mathrm{L}\theta_2'$ Then, $X_\mathrm{L} \xrightarrow{\theta_1'} X_{1_\mathrm{L}} \xrightarrow{\theta_2'} Y_\mathrm{L}$ and $X_\mathrm{R} = Y_\mathrm{R}$. As $|_\mathrm{L}\theta_1' = \theta_1 \smile \theta_2 = |_\mathrm{L}\theta_2'$, it is the case that $\theta_1' \smile \theta_2'$ in $X_\mathrm{L} \xrightarrow{\theta_1'} X_{1_\mathrm{L}} \xrightarrow{\theta_2'} Y_\mathrm{L}$ by Lemma 5, and we can use induction to obtain X_2 s.t. $X_\mathrm{L} \xrightarrow{\theta_2'} X_2 \xrightarrow{\theta_1'} Y_\mathrm{L}$, from which it is immediate to obtain $X_\mathrm{L} \mid X_\mathrm{R} \xrightarrow{|_\mathrm{L}\theta_2'} X_2 \mid X_\mathrm{R} \xrightarrow{|_\mathrm{L}\theta_1'} Y_\mathrm{L} \mid X_\mathrm{R} = Y_\mathrm{L} \mid Y_\mathrm{R}$.

θ_2 **is** $\langle |_{\mathrm{L}}\theta_{2_{\mathrm{L}}}, |_{\mathrm{R}}\theta_{2\mathrm{R}}\rangle$ Since $|_{\mathrm{L}}\theta_1' = \theta_1 \smile \theta_2 = \langle |_{\mathrm{L}}\theta_{2_{\mathrm{L}}}, |_{\mathrm{R}}\theta_{2\mathrm{R}}\rangle$, we have that $\theta_1' \smile \theta_{2_{\mathrm{L}}}$ in $X_{\mathrm{L}} \xrightarrow{\theta_1'} X_{1_{\mathrm{L}}} \xrightarrow{\theta_{2_{\mathrm{L}}}} Y_{\mathrm{L}}$ by Lemma 5. Hence, we can use induction to obtain $X_{\mathrm{L}} \xrightarrow{\theta_{2_{\mathrm{L}}}} X_2 \xrightarrow{\theta_1'} Y_{\mathrm{L}}$. Since we also have that $X_{\mathrm{R}} \xrightarrow{\theta_{2\mathrm{R}}} Y_{\mathrm{R}}$, we can compose both traces using first syn., then $|_{\mathrm{L}}$ to obtain

$$X_{\mathrm{L}} \mid X_{\mathrm{R}} \xrightarrow{\langle |_{\mathrm{L}}\theta_{2_{\mathrm{L}}}, |_{\mathrm{R}}\theta_{2\mathrm{R}}\rangle} X_2 \mid Y_{\mathrm{R}} \xrightarrow{|_{\mathrm{L}}\theta_1'} Y_{\mathrm{L}} \mid Y_{\mathrm{R}}.$$

$|_{\mathrm{R}}$ This is symmetric to $|_{\mathrm{L}}$.

syn. There exists $X_{\mathrm{L}}, X_{\mathrm{R}}, \theta_{1\mathrm{L}}, \theta_{1\mathrm{L}}, X_{1_{\mathrm{L}}}, X_{1_{\mathrm{R}}}, Y_{\mathrm{L}}$ and Y_{R} s.t. $X \xrightarrow{\theta_1} X_1 \xrightarrow{\theta_2} Y$ is

$$X_{\mathrm{L}} \mid X_{\mathrm{R}} \xrightarrow{\langle |_{\mathrm{L}}\theta_{1\mathrm{L}}, |_{\mathrm{R}}\theta_{1\mathrm{R}}\rangle} X_{1_{\mathrm{L}}} \mid X_{1_{\mathrm{R}}} \xrightarrow{\theta_2} Y_{\mathrm{L}} \mid Y_{\mathrm{R}}.$$

Then, $X_{\mathrm{L}} \xrightarrow{\theta_{1\mathrm{L}}} X_{1_{\mathrm{L}}}$, $X_{\mathrm{R}} \xrightarrow{\theta_{1\mathrm{R}}} X_{1_{\mathrm{R}}}$ and the proof proceeds by case on θ_2:

θ_2 **is** $|_{\mathrm{R}}\theta_{2\mathrm{R}}$ Then $X_{1_{\mathrm{R}}} \xrightarrow{\theta_{2\mathrm{R}}} Y_{\mathrm{R}}$, $X_{1_{\mathrm{L}}} = Y_{\mathrm{L}}$ and $\langle |_{\mathrm{L}}\theta_{1\mathrm{L}}, |_{\mathrm{R}}\theta_{1\mathrm{R}}\rangle \smile |_{\mathrm{R}}\theta_{2\mathrm{R}}$. Then by Lemma 5 there exists $X_{\mathrm{R}} \xrightarrow{\theta_{1\mathrm{R}}} X_{1_{\mathrm{R}}} \xrightarrow{\theta_{2\mathrm{R}}} Y_{\mathrm{R}}$ and $\theta_{1\mathrm{R}} \smile \theta_{2\mathrm{R}}$. We can then use the induction hypothesis to obtain $X_{\mathrm{R}} \xrightarrow{\theta_{2\mathrm{R}}} X_{2_{\mathrm{R}}} \xrightarrow{\theta_{1\mathrm{R}}} Y_{\mathrm{R}}$ from which it is immediate to obtain

$$X_{\mathrm{L}} \mid X_{\mathrm{R}} \xrightarrow{|_{\mathrm{R}}\theta_{2\mathrm{R}}} X_{\mathrm{L}} \mid X_{2_{\mathrm{R}}} \xrightarrow{\langle |_{\mathrm{L}}\theta_{2_{\mathrm{L}}}, |_{\mathrm{R}}\theta_{1\mathrm{R}}\rangle} X_{1_{\mathrm{L}}} \mid Y_{\mathrm{R}} = Y_{\mathrm{L}} \mid Y_{\mathrm{R}}.$$

θ_2 **is** $|_{\mathrm{L}}\theta_{2_{\mathrm{L}}}$ This is symmetric to $|_{\mathrm{R}}\theta_{2\mathrm{R}}$.

θ_2 **is** $\langle |_{\mathrm{L}}\theta_{2_{\mathrm{L}}}, |_{\mathrm{R}}\theta_{2\mathrm{R}}\rangle$ This case is essentially a combination of the two previous cases. Since $\langle |_{\mathrm{L}}\theta_{1\mathrm{L}}, |_{\mathrm{R}}\theta_{1\mathrm{R}}\rangle = \theta_1 \smile \theta_2 = \langle |_{\mathrm{L}}\theta_{2_{\mathrm{L}}}, |_{\mathrm{R}}\theta_{2\mathrm{R}}\rangle$, Lemma 5 gives the two traces

$$X_{\mathrm{L}} \xrightarrow{\theta_{1\mathrm{L}}} X_{1_{\mathrm{L}}} \xrightarrow{\theta_{2_{\mathrm{L}}}} Y_{\mathrm{L}} \qquad \text{and} \qquad X_{\mathrm{R}} \xrightarrow{\theta_{1\mathrm{R}}} X_{1_{\mathrm{R}}} \xrightarrow{\theta_{2\mathrm{R}}} Y_{\mathrm{R}}$$

and $\theta_{1\mathrm{L}} \smile \theta_{2_{\mathrm{L}}}$ and $\theta_{1\mathrm{R}} \smile \theta_{2\mathrm{R}}$, respectively. By induction hypothesis, we obtain two traces

$$X_{\mathrm{L}} \xrightarrow{\theta_{2_{\mathrm{L}}}} X_{2_{\mathrm{L}}} \xrightarrow{\theta_{1\mathrm{L}}} Y_{\mathrm{L}} \qquad \text{and} \qquad X_{\mathrm{R}} \xrightarrow{\theta_{2\mathrm{R}}} X_{2_{\mathrm{R}}} \xrightarrow{\theta_{1\mathrm{R}}} Y_{\mathrm{R}}$$

that we can then re-combine using syn. twice to obtain, as desired,

$$X_{\mathrm{L}} \mid X_{\mathrm{R}} \xrightarrow{\langle |_{\mathrm{L}}\theta_{2_{\mathrm{L}}}, |_{\mathrm{R}}\theta_{2\mathrm{R}}\rangle} X_{2_{\mathrm{L}}} \mid X_{2_{\mathrm{R}}} \xrightarrow{\langle |_{\mathrm{L}}\theta_{1\mathrm{L}}, |_{\mathrm{R}}\theta_{1\mathrm{R}}\rangle} Y_{\mathrm{L}} \mid Y_{\mathrm{R}}.$$

$+_{\mathrm{L}}$ There exists X_{L}, X_{R}, θ'_1, θ'_2, $X_{1\mathrm{L}}$, and Y_{L} s.t. $X \xrightarrow{\theta_1} X_1 \xrightarrow{\theta_2} Y$ is

$$X_{\mathrm{L}} + X_{\mathrm{R}} \xrightarrow{+_{\mathrm{L}}\theta'_1} X_{1\mathrm{L}} + X_{\mathrm{R}} \xrightarrow{+_{\mathrm{L}}\theta'_2} Y_{\mathrm{L}} + X_{\mathrm{R}}.$$

All transitions happen on "X_{L}'s side" and X_{R} remains unchanged as otherwise we could not sum two non-standard terms, so that θ_2 must be of the form $+_{\mathrm{L}}\theta'_2$. Then, we can use Lemma 6 to obtain

$$X_{\mathrm{L}} \xrightarrow{\theta'_1} X_{1\mathrm{L}} \xrightarrow{\theta'_2} Y_{\mathrm{L}}$$

and $\theta'_1 \smile \theta'_2$. Hence we can use the induction hypothesis to obtain X_2 s.t. $X_{\mathrm{L}} \xrightarrow{\theta'_2} X_2 \xrightarrow{\theta'_1} Y_{\mathrm{L}}$. From this, it is easy to obtain

$$X_{\mathrm{L}} + X_{\mathrm{R}} \xrightarrow{+_{\mathrm{L}}\theta'_2} X_2 + X_{\mathrm{R}} \xrightarrow{+_{\mathrm{L}}\theta'_1} Y_{\mathrm{L}} + X_{\mathrm{R}} = Y_{\mathrm{L}} + Y_{\mathrm{R}}$$

and this concludes this case.

$+_{\mathrm{R}}$ This is symmetric to $+_{\mathrm{L}}$. □

Example 2. Re-using Example 1, since $|_{\mathrm{L}}\overline{b}[n] \smile |_{\mathrm{R}}+_{\mathrm{R}}c[n']$, Lemma 7 allows to re-arrange the trace

$$a[m].\overline{b} \mid b + c \xrightarrow{|_{\mathrm{L}}\overline{b}[n]} a[m].\overline{b}[n] \mid b + c \xrightarrow{|_{\mathrm{R}}+_{\mathrm{R}}c[n']} a[m].\overline{b}[n] \mid b + c[n']$$

as

$$a[m].\overline{b} \mid b + c \xrightarrow{|_{\mathrm{R}}+_{\mathrm{R}}c[n']} a[m].\overline{b} \mid b + c[n'] \xrightarrow{|_{\mathrm{L}}\overline{b}[n]} a[m].\overline{b}[n] \mid b + c[n'].$$

We state, discuss and then prove the following two lemmas:

Lemma 8 (Reverse Diamond)**.** *For all $X \xrightarrow{\theta_1} X_1 \overset{\theta_2}{\rightsquigarrow} Y$ with $\theta_1 \smile \theta_2$, there exists X_2 s.t. $X \overset{\theta_2}{\rightsquigarrow} X_2 \xrightarrow{\theta_1} Y$.*

Lemma 9 (Square Property)**.** *For all $X \overset{\theta_1}{\rightsquigarrow} X_1 \xrightarrow{\theta_2} Y$ with $\theta_1 \smile \theta_2$, there exists X_2 s.t. $X \xrightarrow{\theta_2} X_2 \overset{\theta_1}{\rightsquigarrow} Y$.*

In both cases, in the particular cases of $t; t^\bullet : X \xrightarrow{\theta_1} X_1 \overset{\theta_1}{\rightsquigarrow} X$, or of $t^\bullet; t$, note that $\theta_1 \smile \theta_1$ never holds since $\theta_1 \lessdot \theta_1$ always holds by reflexivity of $\lessdot$ (Claim 1) and hence Lemmas 8 and 9 cannot apply. The proofs re-use the proof of Lemma 7 almost as it is, since Lemmas 3, 5 and 6 hold for both directions.

Proof of Lemma 8. The only case that diverges with the proof of Lemma 7 is if the deduction for $X \xrightarrow{\theta_1} X_1$ have for last rule pre. In this case, $\alpha[k].X' \xrightarrow{\theta_1} \alpha[k].X'_1 \overset{\theta_2}{\rightsquigarrow} Y$, but we cannot deduce that $\not\kappa(\theta_2) \neq k$ immediately. However, if $\not\kappa(\theta_2) = k$, then we would have $\alpha[k].X'_1 \overset{\alpha[k]}{\rightsquigarrow} \alpha.Y' = Y$, but this application of act$^\bullet_\cdot$ is not valid, as $\mathrm{std}(X'_1)$ does not hold, since X'_1 was obtained from X' after it made a *forward* transition. Hence, we obtain that $\not\kappa(\theta_2) \neq k$, that k occurs in X, X_1 and Y attached to α, so that $k \notin \mathrm{key}(\mathrm{rm}^\alpha_k(X)) \cup \mathrm{key}(\mathrm{rm}^\alpha_k(X_1) \cup \mathrm{key}(\mathrm{rm}^\alpha_k(Y))$, and we can carry out the rest of the proof, using Lemma 3, as before. □

Example 3. Re-using Example 1, since $|_{\mathrm{R}} +_{\mathrm{R}} c[n'] \smile |_{\mathrm{L}} \overline{b}[n]$, the trace

$$a[m].\overline{b}[n] \mid b + c \xrightarrow{|_{\mathrm{R}} +_{\mathrm{R}} c[n']} a[m].\overline{b}[n] \mid b + c[n'] \overset{|_{\mathrm{L}} \overline{b}[n]}{\rightsquigarrow} a[m].\overline{b} \mid b + c[n'],$$

can be rearranged using Lemma 8 as

$$a[m].\overline{b}[n] \mid b + c \overset{|_{\mathrm{L}} \overline{b}[n]}{\rightsquigarrow} a[m].\overline{b} \mid b + c \xrightarrow{|_{\mathrm{R}} +_{\mathrm{R}} c[n']} a[m].\overline{b} \mid b + c[n'].$$

For Lemma 9, the main difference lies in leveraging the dependency of sum prefixes between e.g., $+_{\mathrm{R}}\theta_1$ and $+_{\mathrm{L}}\theta_2$ in $X + O_Y \overset{+_{\mathrm{R}}\theta_1}{\rightsquigarrow} O_X + O_Y \xrightarrow{+_{\mathrm{L}}\theta_2} O_X + Y$.

Proof of Lemma 9. The proof is very similar to the proof of Lemma 7, but we detail it nevertheless for completeness, and also because the sum case diverges and exposes the design choices made in Definition 9 for the sum group.

It proceeds by induction on the length of the deduction for the derivation for $X \overset{\theta_1}{\rightsquigarrow} X_1$:

Length 1 In this case, the derivation is a single application of act$^\bullet_\cdot$, and θ_1 is of the form $\alpha[k]$. But $\alpha[k] \smile \theta_2$ cannot hold, as $\alpha[k] \lessdot \theta_2$ always holds, and this case is vacuously true.

Length > 1 We proceed by case on the last rule.

pre$^\bullet_\cdot$ There exists α, k, X' and X'_1 s.t. $X = \alpha[k].X' \overset{\theta_1}{\rightsquigarrow} \alpha[k].X'_1 = X_1$ and that $\not\kappa(\theta_1) \neq k$. As $\alpha[k].X'_1 \xrightarrow{\theta_2} Y$ we know that $\not\kappa(\theta_2) \neq k$ [7, Lemma 3.4]. Furthermore, since k occurs attached to α in X_1 and since X_1 makes

a *forward* transition to reach Y, we know that $k \notin \mathrm{key}(\mathrm{rm}_k^\alpha(X_1)) \cup \mathrm{key}(Y)$. Hence, we can apply Lemma 3 twice to obtain

$$\mathrm{rm}_k^\alpha(\alpha[k].X') = X' \overset{\theta_1}{\rightsquigarrow} \mathrm{rm}_k^\alpha(\alpha[k].X_1') = X_1' \xrightarrow{\theta_2} \mathrm{rm}_k^\alpha(Y)$$

As $\theta_1 \smile \theta_2$ by hypothesis, we can use the induction hypothesis to obtain that there exists X_2 s.t. $X' \xrightarrow{\theta_2} X_2 \overset{\theta_1}{\rightsquigarrow} \mathrm{rm}_k^\alpha(Y)$. Since $\not{k}(\theta_2) \neq k$, we can append pre. to the derivation of $X' \xrightarrow{\theta_2} X_2$ to obtain $\alpha[k].X' = X \xrightarrow{\theta_2} \alpha[k].X_2$. Using Lemma 3 one last time, we obtain that $\mathrm{rm}_k^\alpha(\alpha[k].X_2) = X_2 \overset{\theta_1}{\rightsquigarrow} \mathrm{rm}_k^\alpha(Y)$ implies $\alpha[k].X_2 \overset{\theta_1}{\rightsquigarrow} Y$, which concludes this case.

res$^\bullet$ This is immediate by induction hypothesis.

$|^\bullet_{\mathrm{L}}$ There exists X_{L}, X_{R}, θ_1', $X_{1_{\mathrm{L}}}$, and Y_{L}, Y_{R} s.t. $X \overset{\theta_1}{\rightsquigarrow} X_1 \xrightarrow{\theta_2} Y$ is

$$X_{\mathrm{L}} \mid X_{\mathrm{R}} \overset{|_{\mathrm{L}}\theta_1'}{\rightsquigarrow} X_{1_{\mathrm{L}}} \mid X_{\mathrm{R}} \xrightarrow{\theta_2} Y_{\mathrm{L}} \mid Y_{\mathrm{R}}.$$

Then, $X_{\mathrm{L}} \overset{\theta_1'}{\rightsquigarrow} X_{1_{\mathrm{L}}}$ and the proof proceeds by case on θ_2:

θ_2 **is** $|_{\mathrm{R}}\theta_2'$ Then $X_{\mathrm{R}} \xrightarrow{\theta_2'} Y_{\mathrm{R}}$, $X_{1_{\mathrm{L}}} = Y_{\mathrm{L}}$ and the occurrences of the rules $|^\bullet_{\mathrm{L}}$ and $|_{\mathrm{R}}$ can be swapped to obtain

$$X_{\mathrm{L}} \mid X_{\mathrm{R}} \xrightarrow{|_{\mathrm{R}}\theta_2'} X_{\mathrm{L}} \mid Y_{\mathrm{R}} \overset{|_{\mathrm{L}}\theta_1'}{\rightsquigarrow} Y_{\mathrm{L}} \mid Y_{\mathrm{R}}.$$

θ_2 **is** $|_{\mathrm{L}}\theta_2'$ Then, $X_{\mathrm{L}} \overset{\theta_1'}{\rightsquigarrow} X_{1_{\mathrm{L}}} \xrightarrow{\theta_2'} Y_{\mathrm{L}}$ and $X_{\mathrm{R}} = Y_{\mathrm{R}}$. As $|_{\mathrm{L}}\theta_1' = \theta_1 \smile \theta_2 = |_{\mathrm{L}}\theta_2'$, it is the case that $X_{\mathrm{L}} \overset{\theta_1'}{\rightsquigarrow} X_{1_{\mathrm{L}}} \xrightarrow{\theta_2'} Y_{\mathrm{L}}$ and $\theta_1' \smile \theta_2'$ by Lemma 5, and we can use induction to obtain X_2 s.t. $X_{\mathrm{L}} \xrightarrow{\theta_2'} X_2 \overset{\theta_1'}{\rightsquigarrow} Y_{\mathrm{L}}$, from which it is immediate to obtain $X_{\mathrm{L}} \mid X_{\mathrm{R}} \xrightarrow{|_{\mathrm{L}}\theta_2'} X_2 \mid X_{\mathrm{R}} \xrightarrow{|_{\mathrm{L}}\theta} Y_{\mathrm{L}} \mid X_{\mathrm{R}} = Y_{\mathrm{L}} \mid Y_{\mathrm{R}}$.

θ_2 **is** $\langle |_{\mathrm{L}}\theta_{2_{\mathrm{L}}}, |_{\mathrm{R}}\theta_{2\mathrm{R}} \rangle$ Since $|_{\mathrm{L}}\theta_1' = \theta_1 \smile \theta_2 = \langle |_{\mathrm{L}}\theta_{2_{\mathrm{L}}}, |_{\mathrm{R}}\theta_{2\mathrm{R}} \rangle$, we have $X_{\mathrm{L}} \overset{\theta_1'}{\rightsquigarrow} X_{1_{\mathrm{L}}} \xrightarrow{\theta_{2_{\mathrm{L}}}} Y_{\mathrm{L}}$ and $\theta_1' \smile \theta_{2_{\mathrm{L}}}$ by Lemma 5. Hence, we can use induction to obtain $X_{\mathrm{L}} \xrightarrow{\theta_{2_{\mathrm{L}}}} X_2 \overset{\theta_1'}{\rightsquigarrow} Y_{\mathrm{L}}$. Since we also have that $X_{\mathrm{R}} \xrightarrow{\theta_{2\mathrm{R}}} Y_{\mathrm{R}}$, we can compose both traces using first syn., then $|^\bullet_{\mathrm{L}}$ to obtain

$$X_{\mathrm{L}} \mid X_{\mathrm{R}} \xrightarrow{\langle |_{\mathrm{L}}\theta_{2_{\mathrm{L}}}, |_{\mathrm{R}}\theta_{2\mathrm{R}} \rangle} X_2 \mid Y_{\mathrm{R}} \overset{|_{\mathrm{L}}\theta_1'}{\rightsquigarrow} Y_{\mathrm{L}} \mid Y_{\mathrm{R}}.$$

$|^\bullet_{\mathrm{R}}$ This is symmetric to $|^\bullet_{\mathrm{L}}$.

syn$^\bullet$ There exists X_L, X_R, $\theta_{1\mathrm{L}}$, $\theta_{1\mathrm{R}}$, X_{1_L}, X_{1_R}, Y_L and Y_R s.t. $X \overset{\theta_1}{\rightsquigarrow} X_1 \xrightarrow{\theta_2} Y$ is

$$X_\mathrm{L} \mid X_\mathrm{R} \overset{\langle |_\mathrm{L}\theta_{1\mathrm{L}}, |_\mathrm{R}\theta_{1\mathrm{R}}\rangle}{\rightsquigarrow} X_{1_\mathrm{L}} \mid X_{1_\mathrm{R}} \xrightarrow{\theta_2} Y_\mathrm{L} \mid Y_\mathrm{R}.$$

Then, $X_\mathrm{L} \overset{\theta_{1\mathrm{L}}}{\rightsquigarrow} X_{1_\mathrm{L}}$, $X_\mathrm{R} \overset{\theta_{1\mathrm{R}}}{\rightsquigarrow} X_{1_\mathrm{R}}$ and the proof proceeds by case on θ_2:

θ_2 **is** $|_\mathrm{R}\theta_{2\mathrm{R}}$ Then $X_{1_\mathrm{R}} \xrightarrow{\theta_{2\mathrm{R}}} Y_\mathrm{R}$, $X_{1_\mathrm{L}} = Y_\mathrm{L}$ and $\langle |_\mathrm{L}\theta_{1\mathrm{L}}, |_\mathrm{R}\theta_{1\mathrm{R}}\rangle \smile |_\mathrm{R}\theta_{2\mathrm{R}}$ implies $X_\mathrm{R} \overset{\theta_{1\mathrm{R}}}{\rightsquigarrow} X_{1_\mathrm{R}} \xrightarrow{\theta_{2\mathrm{R}}} Y_\mathrm{R}$ and $\theta_{1\mathrm{R}} \smile \theta_{2\mathrm{R}}$ by Lemma 5. We can then use the induction hypothesis to obtain $X_\mathrm{R} \xrightarrow{\theta_{2\mathrm{R}}} X_{2_\mathrm{R}} \overset{\theta_{1\mathrm{R}}}{\rightsquigarrow} Y_\mathrm{R}$ from which it is immediate to obtain

$$X_\mathrm{L} \mid X_\mathrm{R} \xrightarrow{|_\mathrm{R}\theta_{2\mathrm{R}}} X_\mathrm{L} \mid X_{2_\mathrm{R}} \overset{\langle |_\mathrm{L}\theta_{1\mathrm{L}}, |_\mathrm{R}\theta_{1\mathrm{R}}\rangle}{\rightsquigarrow} X_{1_\mathrm{L}} \mid Y_\mathrm{R} = Y_\mathrm{L} \mid Y_\mathrm{R}.$$

θ_2 **is** $|_\mathrm{L}\theta_{2_\mathrm{L}}$ This is symmetric to $|_\mathrm{R}\theta_{2\mathrm{R}}$.

θ_2 **is** $\langle |_\mathrm{L}\theta_{2_\mathrm{L}}, |_\mathrm{R}\theta_{2\mathrm{R}}\rangle$ This case is essentially a combination of the two previous cases. Since $\langle |_\mathrm{L}\theta_{1\mathrm{L}}, |_\mathrm{R}\theta_{1\mathrm{R}}\rangle = \theta_1 \smile \theta_2 = \langle |_\mathrm{L}\theta_{2_\mathrm{L}}, |_\mathrm{R}\theta_{2\mathrm{R}}\rangle$, Lemma 5 gives two traces

$$X_\mathrm{L} \overset{\theta_{1\mathrm{L}}}{\rightsquigarrow} X_{1_\mathrm{L}} \xrightarrow{\theta_{2_\mathrm{L}}} Y_\mathrm{L} \qquad \text{and} \qquad X_\mathrm{R} \overset{\theta_{1\mathrm{R}}}{\rightsquigarrow} X_{1_\mathrm{R}} \xrightarrow{\theta_{2\mathrm{R}}} Y_\mathrm{R}$$

and $\theta_{1\mathrm{L}} \smile \theta_{2_\mathrm{L}}$ and $\theta_{1\mathrm{R}} \smile \theta_{2\mathrm{R}}$, respectively. By induction hypothesis, we obtain two traces

$$X_\mathrm{L} \xrightarrow{\theta_{2_\mathrm{L}}} X_{2_\mathrm{L}} \overset{\theta_{1\mathrm{L}}}{\rightsquigarrow} Y_\mathrm{L} \qquad \text{and} \qquad X_\mathrm{R} \xrightarrow{\theta_{2\mathrm{R}}} X_{2_\mathrm{R}} \overset{\theta_{1\mathrm{R}}}{\rightsquigarrow} Y_\mathrm{R}$$

that we can then re-combine using syn. and syn$^\bullet$ to obtain, as desired,

$$X_\mathrm{L} \mid X_\mathrm{R} \xrightarrow{\langle |_\mathrm{L}\theta_{2_\mathrm{L}}, |_\mathrm{R}\theta_{2\mathrm{R}}\rangle} X_{2_\mathrm{L}} \mid X_{2_\mathrm{R}} \overset{\langle |_\mathrm{L}\theta_{1\mathrm{L}}, |_\mathrm{R}\theta_{1\mathrm{R}}\rangle}{\rightsquigarrow} Y_\mathrm{L} \mid Y_\mathrm{R}.$$

$+^\bullet_\mathrm{L}$ There exists X_L, X_R, $X_{1\mathrm{L}}$, and Y_L s.t. $X \overset{\theta_1}{\rightsquigarrow} X_1 \xrightarrow{\theta_2} Y$ is

$$X_\mathrm{L} + X_\mathrm{R} \overset{+_\mathrm{L}\theta'_1}{\rightsquigarrow} X_{1\mathrm{L}} + X_\mathrm{R} \xrightarrow{\theta_2} Y_\mathrm{L} + Y_\mathrm{R}.$$

Then, $X_\mathrm{L} \overset{\theta'_1}{\rightsquigarrow} X_{1\mathrm{L}}$ and we proceed by case on θ_2:

θ_2 **is** $+_\mathrm{L}\theta'_2$ Then, $X_{1\mathrm{L}} \xrightarrow{\theta'_2} Y_\mathrm{L}$ and $X_\mathrm{R} = Y_\mathrm{R}$. Since $+_\mathrm{L}\theta'_1 \smile +_\mathrm{L}\theta'_2$, we can use Lemma 6 to obtain

$$X_\mathrm{L} \overset{\theta'_1}{\rightsquigarrow} X_{1\mathrm{L}} \xrightarrow{\theta'_2} Y_\mathrm{L}$$

and $\theta'_1 \smile \theta'_2$, and by induction hypothesis there exists X_2 such that

$$X_{\mathrm{L}} \xrightarrow{\theta'_2} X_2 \overset{\theta'_1}{\rightsquigarrow} Y_{\mathrm{L}}$$

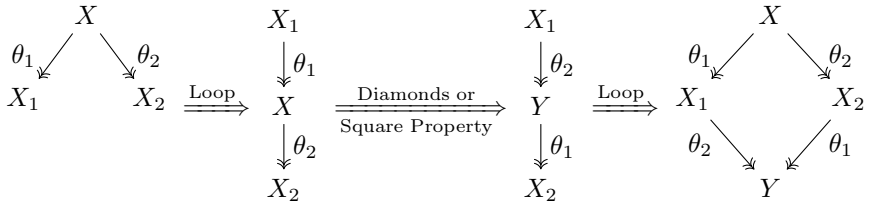
from which it is easy to obtain

$$X_{\mathrm{L}} + X_{\mathrm{R}} \xrightarrow{+_{\mathrm{L}}\theta'_2} X_2 + X_{\mathrm{R}} \overset{+_{\mathrm{L}}\theta'_1}{\rightsquigarrow} Y_{\mathrm{L}} + X_{\mathrm{R}} = Y_{\mathrm{L}} + Y_{\mathrm{R}}.$$

θ_2 **is** $+_{\mathrm{R}}\theta'_2$ Since $+_{\mathrm{L}}\theta'_1 \lessdot +_{\mathrm{R}}\theta'_2$, it cannot be the case that $\theta_1 \smile \theta_2$, so this case is vacuously true. □

Theorem 1 (Main Theorem)**.** *For all* $t_1 : X \overset{\theta_1}{\twoheadrightarrow} X_1$ *and* $t_2 : X \overset{\theta_2}{\twoheadrightarrow} X_2$ *with* $\theta_1 \smile \theta_2$*, there exist* $t'_1 : X_1 \overset{\theta_2}{\twoheadrightarrow} Y$ *and* $t'_2 : X_2 \overset{\theta_1}{\twoheadrightarrow} Y$*.*

The proof is by case on the directions of the arrows, but always follow the same pattern: use the Loop lemma to orient the arrows to be able to use either Lemma 7, 8 or 9, then use the appropriate Lemma to obtain a trace, and use again the Loop lemma to orient it as desired.



Proof. The proof proceeds by case on the directions of t_1 and t_2.

If $t_1 : X \overset{\theta_1}{\rightsquigarrow} X_1$ **and** $t_2 : X \xrightarrow{\theta_2} X_2$ This corresponds to this case:  .

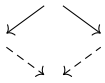
The Loop lemma gives $t_1^{\bullet}; t_2 : X_1 \xrightarrow{\theta_1} X \xrightarrow{\theta_2} X_2$, and since we know that $\theta_1 \smile \theta_2$, we can use the sideways diamond (Lemma 7) to obtain $t''_1; t''_2 : X_1 \xrightarrow{\theta_2} Y \xrightarrow{\theta_1} X_2$, and letting $t'_1 = t''_1$ and $t'_2 = t''^{\bullet}_2$, we obtain $t'_1 : X_1 \xrightarrow{\theta_2} Y$ and $t'_2 : X_2 \overset{\theta_1}{\rightsquigarrow} Y$ as desired.

If $t_1 : X \xrightarrow{\theta_1} X_1$ **and** $t_2 : X \overset{\theta_2}{\rightsquigarrow} X_2$ This corresponds to this case:  .

By symmetry it is identical to the previous one.

If $t_1 : X \overset{\theta_1}{\rightsquigarrow} X_1$ **and** $t_2 : X \overset{\theta_2}{\rightsquigarrow} X_2$ This corresponds to this case:  .

The Loop lemma gives $t_1^\bullet; t_2 : X_1 \xrightarrow{\theta_1} X \overset{\theta_2}{\rightsquigarrow} X_2$, and since we know that $\theta_1 \smile \theta_2$, we can use the reverse diamond Lemma 8 to obtain $t_1''; t_2'' : X_1 \overset{\theta_2}{\rightsquigarrow} Y \xrightarrow{\theta_1} X_2$, and letting $t_1' = t_1''$ and $t_2' = t''^\bullet_2$, we obtain $t_1' : X_1 \overset{\theta_2}{\rightsquigarrow} Y$ and $t_2' : X_2 \overset{\theta_1}{\rightsquigarrow} Y$ as desired.

If $t_1 : X \xrightarrow{\theta_1} X_1$ **and** $t_2 : X \xrightarrow{\theta_2} X_2$ This corresponds to this case:  .

The Loop lemma gives $t_1^\bullet; t_2 : X_1 \overset{\theta_1}{\rightsquigarrow} X \xrightarrow{\theta_2} X_2$, and since we know that $\theta_1 \smile \theta_2$, we can use the square property (Lemma 9) to obtain $t_1''; t_2'' : X_1 \xrightarrow{\theta_2} Y \overset{\theta_1}{\rightsquigarrow} X_2$, and letting $t_1' = t_1''$ and $t_2' = t''^\bullet_2$, we obtain $t_1' : X_1 \xrightarrow{\theta_2} Y$ and $t_2' : X_2 \xrightarrow{\theta_1} Y$ as desired. □

Example 4. Following Example 1, we can obtain e.g., from the coinitial transitions

$$a[m].\overline{b}[n] \mid b + c \xrightarrow{|_{\mathrm{R}} + {}_{\mathrm{L}} b[n']} a[m].\overline{b}[n] \mid b[n'] + c$$

and

$$a[m].\overline{b}[n] \mid b + c \overset{|_{\mathrm{L}} \overline{b}[n]}{\rightsquigarrow} a[m].\overline{b} \mid b + c$$

the transitions converging to $a[m].\overline{b} \mid b[n'] + c$,

$$a[m].\overline{b}[n] \mid b[n'] + c \overset{|_{\mathrm{L}} \overline{b}[n]}{\rightsquigarrow} a[m].\overline{b} \mid b[n'] + c$$

and

$$a[m].\overline{b} \mid b + c \xrightarrow{|_{\mathrm{R}} + {}_{\mathrm{L}} b[n']} a[m].\overline{b} \mid b[n'] + c.$$

4.3. Causal Consistency & Other Properties

Formally, causal consistency (Theorem 2) states that any two coinitial and cofinal traces are causally equivalent. The empty trace being denoted by ϵ (Definition 7), causally equivalence is defined as follows:

Definition 13 (Causally equivalent)**.** Two traces T_1, T_2 are *causally equivalent*, if they are in the least equivalence relation closed by composition satisfying $t; t^\bullet \sim \epsilon$ and $t_1; t_2' \sim t_2; t_1'$ for any $t_1; t_2' : X \xrightarrow{\theta_1} \!\!\!\!\rightarrow \xrightarrow{\theta_2} \!\!\!\!\rightarrow Y$, $t_2; t_1' : X \xrightarrow{\theta_2} \!\!\!\!\rightarrow \xrightarrow{\theta_1} \!\!\!\!\rightarrow Y$.

Theorem 2. *All coinitial and cofinal traces are causally equivalent.*

The axiomatic approach to reversible computation [11] allows to obtain causal consistency from other properties that are generally easier to prove. We state and prove them so that the proof of Theorem 2 becomes a simple corollary.

Lemma 10 (Backward transitions are concurrent)**.** *Any two different coinitial backward transitions $t_1 : X \overset{\theta_1}{\rightsquigarrow} X_1$ and $t_2 : X \overset{\theta_2}{\rightsquigarrow} X_2$ are concurrent.*

The proof is by induction on the length of the deduction for the derivation for $X \overset{\theta_1}{\rightsquigarrow} X_1$ and leverages that $\not{k}(\theta_1) \neq \not{k}(\theta_2)$ for both transitions to be different.

Proof. The first important fact to note is that $\not{k}(\theta_1) \neq \not{k}(\theta_2)$: by a simple inspection of the backward rules in Fig. 3, it is easy to observe that if a reachable process X can perform two different backward transitions, then they must have different keys.

We then proceed by induction on the length of the deduction for the derivation for $X \overset{\theta_1}{\rightsquigarrow} X_1$:

Length 1 In this case, the derivation is a single application of act$^\bullet$, and θ_1 is of the form $\alpha[k]$, with $X = \alpha[k].X'$ and $\mathrm{std}(X')$. Hence, X cannot perform two different transitions, and this case is vacuously true.

Length > 1 We proceed by case on the last rule.

pre$^\bullet$ There exists α, k, X' and X_1' s.t. $X = \alpha[k].X' \overset{\theta_1}{\rightsquigarrow} \alpha[k].X_1' = X_1$. Then, it must be the case that $X' \overset{\theta_1}{\rightsquigarrow} X_1'$ and X' is not standard. Since X' is not standard, the last rule for the derivation of $X \overset{\theta_2}{\rightsquigarrow} X_2$ cannot be act$^\bullet$, and since $X = \alpha[k].X'$, it must be pre$^\bullet$, hence it must be the case that $X = \alpha[k].X' \overset{\theta_2}{\rightsquigarrow} \alpha[k].X_2' = X_2$, and we know that $X' \overset{\theta_2}{\rightsquigarrow} X_2'$. We conclude by using the induction hypothesis on the two backward transitions of X' and the observation that pre$^\bullet$ preserves the label and hence concurrency.

res$^\bullet$ This is immediate by induction hypothesis.

$|^\bullet_{\mathrm{L}}$ There exists X_{L}, X_{R}, θ'_1 and $X_{1_{\mathrm{L}}}$ s.t. $X \overset{\theta_1}{\rightsquigarrow} X_1$ is

$$X_{\mathrm{L}} \mid X_{\mathrm{R}} \overset{|_{\mathrm{L}}\theta'_1}{\rightsquigarrow} X_{1_{\mathrm{L}}} \mid X_{\mathrm{R}}.$$

Then, $X_{\mathrm{L}} \overset{\theta'_1}{\rightsquigarrow} X_{1_{\mathrm{L}}}$ and the proof proceeds by case on θ_2, using Lemma 5 to decompose the traces:

θ_2 **is** $|_{\mathrm{R}}\theta'_2$ Then this is immediate, as $|_{\mathrm{L}}\theta'_1 \nleqslant |_{\mathrm{R}}\theta'_2$ and $|_{\mathrm{R}}\theta'_2 \nleqslant |_{\mathrm{L}}\theta'_1$ never hold.

θ_2 **is** $|_{\mathrm{L}}\theta'_2$ Then there exists $X_{2_{\mathrm{L}}}$ such that $X_{\mathrm{L}} \overset{\theta'_2}{\rightsquigarrow} X_{2_{\mathrm{L}}}$, and we conclude by induction on X_{L}'s backward transitions.

θ_2 **is** $\langle |_{\mathrm{L}}\theta_{2_{\mathrm{L}}}, |_{\mathrm{R}}\theta_{2\mathrm{R}} \rangle$ Then we know that

$$X_L \mid X_{\mathrm{R}} \overset{\langle |_{\mathrm{L}}\theta_{2_{\mathrm{L}}}, |_{\mathrm{R}}\theta_{2\mathrm{R}} \rangle}{\rightsquigarrow} X_{2_{\mathrm{L}}} \mid X_{2_{\mathrm{R}}}.$$

For $|_{\mathrm{L}}\theta'_1$ and $\langle |_{\mathrm{L}}\theta_{2_{\mathrm{L}}}, |_{\mathrm{R}}\theta_{2\mathrm{R}} \rangle$ to be concurrent, we must have $\theta'_1 \smile \theta_{2_{\mathrm{L}}}$. By induction hypothesis on $X_{\mathrm{L}} \overset{\theta'_1}{\rightsquigarrow} X_{1_{\mathrm{L}}}$ and $X_{\mathrm{L}} \overset{\theta_{2_{\mathrm{L}}}}{\rightsquigarrow} X_{2_{\mathrm{L}}}$, we know that those two transitions are concurrent, which concludes this case.

$|^\bullet_{\mathrm{R}}$ This is symmetric to $|^\bullet_{\mathrm{L}}$.

syn$^\bullet$ This case is similar to the two previous ones and does not offer any insight nor resistance.

$+^\bullet_{\mathrm{L}}$ There exists X_{L}, X_{R}, and $X_{1\mathrm{L}}$ s.t. $X \overset{\theta_1}{\rightsquigarrow} X_1$ is

$$X_{\mathrm{L}} + X_{\mathrm{R}} \overset{+_{\mathrm{L}}\theta'_1}{\rightsquigarrow} X_{1\mathrm{L}} + X_{\mathrm{R}}.$$

Then, note that θ_2 must also be of the form $+_{\mathrm{L}}\theta'_2$, as X_{R} must be standard. Hence, this follows by induction hypothesis on the transitions $X_{\mathrm{L}} \overset{\theta'_1}{\rightsquigarrow} X_{1\mathrm{L}}$ and $X_{\mathrm{L}} \overset{\theta'_2}{\rightsquigarrow} X_{2\mathrm{L}}$, using Lemma 6 to decompose the trace. □

Lemma 11 (Well-foundedness)**.** *For all X there exists $n \in \mathbb{N}$, $X_0, \ldots, X_n$ s.t. $X \rightsquigarrow X_n \rightsquigarrow \cdots \rightsquigarrow X_1 \rightsquigarrow X_0$, with* $\mathrm{std}(X_0)$.

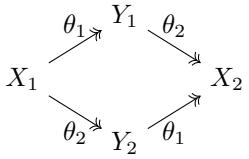
This lemma forbids infinite reverse computation, and is obvious in CCSK as any backward transition strictly decreases the number of occurrences of keys.

Proof of Theorem 2. We can use the results of the axiomatic approach [11] since our forward LTS is the symmetric of our backward LTS, and as our concurrency relation (that the authors call the independence relation, following a common usage [36, Definition 3.7]) is irreflexive and symmetric (Claim 2). Then, by Theorem 1 and Lemma 10, the parabolic lemma holds [11, Proposition 3.4], and since the parabolic lemma and well-foundedness hold (Lemma 11), causal consistency holds as well [11, Proposition 3.5]. □

We use here the axiomatic approach [11] in a narrow sense, to obtain causal consistency—which was our main goal—, but we can use it to obtain many other desirable properties for this system "for free". For instance, since our system enjoys Coinitial Propagation of Independence (Corollary 1) and—as we just proved—a principle the authors call "BTI" (Lemma 10), we obtain "independence of diamonds" automatically:

Corollary 2 (Independence of Diamonds [11, Definition 4.6])**.** *For all* $t_1 : X_1 \xrightarrow{\theta_1} Y_1$, $t_2 : X_1 \xrightarrow{\theta_2} Y_2$, $t_3 : Y_1 \xrightarrow{\theta_2} X_2$ *and* $t_4 : Y_2 \xrightarrow{\theta_1} X_2$ *with* $Y_1 \neq Y_2$ *if all transitions are forward or if all transitions are backward,* $X_1 \neq X_2$ *otherwise, then* $t_1 \smile_i t_2$.

In picture, we have



Proof. This is a direct consequence of [11, Proposition 4.7], as our system enjoys Corollary 1 and Lemma 10. □

Example 5. Re-using the full trace presented in Example 1, we can re-organize the transitions using the diamonds so that every undone transition is undone immediately, and we obtain up to causal equivalence the trace

$$a.\overline{b} \mid b + c \xrightarrow{|_{\mathrm{L}} a[m]} a[m].\overline{b} \mid b + c \xrightarrow{\langle |_{\mathrm{L}} \overline{b}[n], |_{\mathrm{R}} +_{\mathrm{L}} b[n] \rangle} a[m].\overline{b}[n] \mid b[n] + c$$

5. Comparing Concurrencies Accross Calculi

We detail in this section how the concurrency we defined is universal, in the following sense:

- It is equivalent to the restriction to CCSK of the definition of concurrency on composable transitions for a reversible π-calculus extending CCSK [18] (Sect. 5.1),
- Our definition, when adapted to RCCS (Sect. 5.3), yields a concurrency that extends (Sect. 5.4) existing definitions for RCCS (Sect. 5.2),
- Our definition can similarly be adapted to an "identified" declension of RCCS and proven equal to its definition of concurrency (Sect. 5.5).

It should be noted, with respect to this second point, that existing definitions for RCCS do not define concurrency on transitions of opposite directions, whereas ours does (Sect. 5.2.3): in this sense, recognizing more transitions as concurrent is an interesting improvement testtest

We also briefly illustrate, p. 43, that the concurrency stemming from the first item does not satisfy the "denotationality" [8, Section 6] criteria, i.e., that it is not preserved by CCSK's structural congruence.

Comparing accross calculi requires to introduce two other reversible systems and four other definitions of concurrency. This a lot of technical content, but we tried to make it as compact and as intuitive as we could, and we would like to stress that the results stated below are fairly routine to prove.

5.1. Comparing With Concurrency Stemming From Reversible π-Calculus

A definition of concurrency was introduced for a reversible π-calculus extending CCSK [18], but without sum. We offer to restrict it to CCSK (without sum), to compare the resulting relation with our definition using proved labels, and to assess how it fares with respect to structural equivalence for CCSK.

5.1.1. Causalities: Definitions and Adequacy

The following definitions can easily be extended to CCSK with sum, so we preserves the "full" system for this study of the adequacy of causality.

Definition 14 (Context)**.** A *context* is a CCSK process with a slot $\cdot$:

$$C[\cdot] \coloneqq \cdot \parallel C[\cdot] + X \parallel X + C[\cdot] \parallel C[\cdot] | X \parallel X | C[\cdot] \parallel \alpha[k].C[\cdot] \parallel C[\cdot] \backslash \alpha$$

Note that the context $\alpha.C[\cdot]$ (i.e., without the key) is missing as it does not play any role in the following definition.

Definition 15 (Structural cause [18, Definition 21])**.** For all X, $m_1, m_2 \in \text{key}(X)$, the prefix with key m_1 is *a structural cause* of the prefix with key m_2, denoted $m_1 \sqsubset_X m_2$, if $\exists C[\cdot]$ s.t. $X = C[\alpha[m_1].Y]$ with $m_2 \in \text{key}(Y)$.

Definition 16 (Structural causality [18, Definition 22])**.** In

$$t_1; t_2 : X \xrightarrow{\alpha_1[m_1]} \!\!\!\!\rightarrow X_1 \xrightarrow{\alpha_2[m_2]} \!\!\!\!\rightarrow X_2,$$

t_1 is *a structural cause* of t_2, denoted $t_1 \sqsubset t_2$, if

- $i_1 \sqsubset_{X_2} i_2$, if t_1 and t_2 are both forward,
- $i_2 \sqsubset_X i_1$, if t_1 and t_2 are both backward.

We now prove that the structural causality we just defined agrees with the dependency relation (Definition 9), letting f be the function that maps keyed labels to proved labels obtained from Lemma 2.

Lemma 12 (Adequacy of the structural causality with the dependency relation)**.** *In* $t_1; t_2 : X \xrightarrow{\alpha_1[m_1]} \!\!\!\!\rightarrow X_1 \xrightarrow{\alpha_2[m_2]} \!\!\!\!\rightarrow X_2$*, if* t_1 *and* t_2 *have the same directions, then* $t_1 \sqsubset t_2$ *iff* $f(\alpha_1[m_1]) \lessdot f(\alpha_2[m_2])$*.*

Proof. First, observe that $t_1 \sqsubset t_2$ iff $t_2^\bullet \sqsubset t_1^\bullet$, and since similarly $\theta_1 \lessdot \theta_2$ in $t_1; t_2 : X \xrightarrow{\theta_1} X_1 \xrightarrow{\theta_2} X_2$ iff $\theta_2 \lessdot \theta_1$ in $t_2^\bullet; t_1^\bullet : X_2 \xrightarrow{\theta_2} X_1 \xrightarrow{\theta_2} X$, it suffices to prove the statement for both t_1 and t_2 forward.

We prove the statement from right to left first, proceeding by induction on the length of the deduction for the derivation for $X \xrightarrow{\alpha_1[m_1]} X_1$.

Length 1 In this case, the derivation is a single application of act., and it is easy to see that $f(\alpha_1[m_1])$ is $\alpha_1[m_1]$, and since $\alpha_1[m_1] \lessdot f(\alpha_2[m_2])$ and $X_2 = \alpha_1[m_1].Y$ with $m_2 \in \not\in(Y)$, both causality relations coincide.

Length > 1 We proceed by case on the last rule.

pre., res., $+_{\mathrm{L}}$**,** $+_{\mathrm{R}}$ This is immediate by induction hypothesis, once noted that the derivation for $X_1 \xrightarrow{\alpha_2[m_2]} X_2$ must also end with the same rule.

$|_{\mathrm{L}}$ Then we know that $X \xrightarrow{\alpha_1[m_1]} X_1$ is of the form

$$X_{\mathrm{L}} \mid X_{\mathrm{R}} \xrightarrow{\alpha_1[m_1]} C_{\mathrm{L}}[\alpha_1[m_1].Y_{\mathrm{L}}] \mid X_{\mathrm{R}}$$

and there are three cases, depending on the last rule in the deduction for the derivation for $X_1 \xrightarrow{\alpha_2[m_2]} X_2$:

$|_{\mathrm{L}}$ Then we proceed by induction hypothesis, observing that, for $i \in \{1, 2\}$, $f(\alpha_i[k_i])$ is of the form $|_{\mathrm{L}}\theta_i$, and that $|_{\mathrm{L}}\theta_1 \lessdot |_{\mathrm{L}}\theta_2$ if $\theta_1 \lessdot \theta_2$.

$|_{\mathrm{R}}$ Then it cannot be the case that $f(\alpha_1[m_1]) \lessdot f(\alpha_2[m_2])$ by definition, and and it cannot be the case that $t_1 \sqsubset t_2$, since $X_2 = C_{\mathrm{L}}[\alpha_1[m_1].Y_{\mathrm{L}}] \mid C_{\mathrm{R}}[\alpha_2[m_2].Y_{\mathrm{R}}]$.

syn. Then $X_2 = Y'_{\mathrm{L}} \mid C_{\mathrm{R}}[\alpha_2[m_2].Y_{\mathrm{R}}]$, with $m_2 \in \not\in(X_2)$, and it suffices to reason by induction on the derivations of $C_{\mathrm{L}}[\alpha_1[m_1].Y_{\mathrm{L}}] \mid X_{\mathrm{R}}$ and Y'_{L}.

$|_{\mathrm{R}}$ **and syn.** Those cases are similar to $|_{\mathrm{L}}$.

We now prove the statement from left to right, by induction on the length of $f(\alpha_1[m_1])$ and $f(\alpha_2[m_2])$, and by case analysis on the rules of the dependency relation given in Fig. 7:

Action If $f(\alpha_1[m_1]) = \alpha_1[m_1] \lessdot f(\alpha_2[m_2])$, then $t_1 \sqsubset t_2$ is immediate.

Sum First, note that since both t_1 and t_2 are forward, it cannot be the case that $f(\alpha_1[m_1])$ and $f(\alpha_2[m_2])$ are prefixed with different $+_d$ symbols, since a forward trace cannot execute the right operand of a sum then its left operand (or reciprocally). Hence, $f(\alpha_1[m_1]) = +_d\theta_1 \lessdot f(\alpha_2[m_2]) = +_d\theta_2$ holds iff $\theta_1 \lessdot \theta_2$, which is necessary and sufficient for $t_1 \sqsubset t_2$ to hold by induction hypothesis.

Parallel Each of those four rules state that $f(\alpha_1[m_1]) \lessdot f(\alpha_2[m_2])$ holds if and only if a dependency exists in "the same thread" of the process, which is exactly the notion captured by the requirement on the existence of a context of the form $C[\alpha_1[i_1].Y]$, hence both notions coincide. □

5.1.2. Conflict and Concurrencies

For reversible π-calculus, the causality relation requires to account for names previously shared, using an object causality [18, Definition 23], that is not meaningful nor required in CCSK. However, transitions of opposite direction need to be accounted for with a conflict relation that we restate below:

Definition 17 (Conflict relation [18, Definition 25])**.** In

$$t_1; t_2 : X \xrightarrow{\alpha_1[m_1]} X_1 \xrightarrow{\alpha_2[m_2]} X_2,$$

t_1 and t_2 are *in conflict* if

- t_1 is a forward transition, and $t_2 = t_1^\bullet$,
- t_1 is a backward transition, t_2 is a forward one, and t_2 consumes a prefix freed by t_1.

Note that the conflict relation falls short on detecting conflict in the presence of sum: indeed, taking e.g., $t_1; t_2 : a[m] + b \overset{a[m]}{\rightsquigarrow} a + b \xrightarrow{b[k]} a + b[k]$, t_1 and t_2 would not be in conflict according to Definition 17, as t_2 does not "consume" a prefix freed by t_1. However, it would not be correct to declare them concurrent (as would this work [18, Definition 26]), since they cannot be swapped and are, indeed, dependent. This is fine in the sum-free reversible π-calculus, but also illustrates how concurrency cannot be defined by "simply" restricting the π's calculus definition to CCSK, in the presence of sum.

Lemma 13 (Adequacy of conflict and causality on transitions of opposite directions)**.** In a sum-free CCSK*, in* $t_1; t_2 : X \xrightarrow{\alpha_1[m_1]} X_1 \xrightarrow{\alpha_2[m_2]} X_2$*, if* t_1 *and* t_2 *have opposite directions, then* t_1 *and* t_2 *are in conflict iff* $f(\alpha_1[m_1]) \lessdot f(\alpha_2[m_2])$.

Proof. If $t_2 = t_1^\bullet$, then note that t_2 consumes a prefix freed by t_1 if t_1 was backward, so t_2 and t_1 are in confilct no matter their directions. In this case, it is immediate that $f(\alpha_1[m_1]) \lessdot f(\alpha_2[m_2]) = f(\alpha_1[m_1])$, as $\lessdot$ is reflexive (Claim 1), so both relations coincide.

If $t_2 \neq t_1^\bullet$, then we need to proceed by case on the direction of t_1:

If t_1 is forward Then observe that t_1 and t_2 are never in conflict. We need to prove that $f(\alpha_1[m_1]) \lessdot f(\alpha_2[m_2])$ never holds, but it follows easily from Lemma 10: since $t_2 \neq t_1^\bullet$, we know that the coinitial backward transitions $t_1^\bullet$ and t_2 are different, and hence by Lemma 10 that they are concurrent, proving that $f(\alpha_1[m_1]) \lessdot f(\alpha_2[m_2])$ does not hold.

If t_1 is backward Then we have to prove that $f(\alpha_1[m_1]) \lessdot f(\alpha_2[m_2])$ iff t_2 consumes a prefix freed by t_1. Proving this statement from left to right is easy: it is immediate that if t_2 consumes a prefix freed by t_1, then $f(\alpha_1[m_1]) \lessdot f(\alpha_2[m_2])$ will hold. For the reverse direction, inspecting the Action and Parallel rules of Fig. 7 suffices to prove that $f(\alpha_1[m_1]) \lessdot f(\alpha_2[m_2])$ implies that t_2 have consumed a prefix freed by t_1. □

Hence, in the absence of sum, both notions coincide. It should be noted that our definition of concurrency based on proved labels offers a couple of benefits:

1. It requires only one relation to define concurrency, while the concurrency stemming from reversible π-calculus requires two relations (structural causality *and* conflict).

2. By our definition, it is obvious that t_1 and t_2 are concurrent iff $t_2^\bullet$ and $t_1^\bullet$ are, whereas this result is not obvious for the concurrency stemming from reversible π-calculus.

3. There is no need to inspect the keys or to build appropriate contexts to decide if transitions are concurrent: it suffices to read their (enhanced keyed) labels.

5.1.3. Interplay Between Concurrency and Structural Congruence

Last, but not least, we prove that this concurrency stemming from reversible π-calculus does not fare well with CCSK's structural congruence.

Definition 18 (Free and bound keys [7, Definition 2.1])**.** A key k is *bound in* X iff it occurs either twice, attached to complementary prefixes, or once, attached to a τ prefix, in X. A key k is *free in* X if it occurs once in X, attached to a non-τ prefix.

Definition 19 (Structural equivalence [7, p. 133])**.** The structural equivalence of CCSK is the smallest equivalence relation (that is, reflexive, symmetric, and transitive relation) closed under the following rule:

$$X \equiv X[n/m] \qquad m \text{ bound in } X,\ n \notin \text{key}(X)$$

where $[n/m]$ denotes the substitution of all the occurrences of key m with key n.

The labeled transition system of CCSK is then endowed with the following rules:

$$\frac{Y \equiv X \qquad X \xrightarrow{\alpha[k]} X' \qquad X' \equiv Y'}{Y \xrightarrow{\alpha[k]} Y'}\text{equiv.}$$

$$\frac{Y' \equiv X' \qquad X' \overset{\alpha[k]}{\rightsquigarrow} X \qquad X \equiv Y}{Y' \overset{\alpha[k]}{\rightsquigarrow} Y}\text{equiv}^\bullet.$$

For technical reasons beyond the scope of this exposition, those rules can only be used last when proving a derivation. However, taken as defined, this relation does not play well with the concurrency relation inherited from the reversible π-calculus:

Theorem 3. *The conflict relation inherited from the reversible π-calculus is not adequate for CSSK endowed with structural congruence.*

Proof. Consider the following two equations (the first one is just reflexivity of $\equiv$) and derivation:

$$a[m].c|\overline{a}[m] \equiv a[m].c|\overline{a}[m] \tag{7}$$

$$a[m].c[m']|\overline{a}[m] \equiv a[h].c[m']|\overline{a}[h] \tag{8}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{c \xrightarrow{c[m']} c[m']}\text{act.}}{a[m].c \xrightarrow{c[m']} a[m].c[m']}\text{pre.} \qquad (8)}{a[m].c|\overline{a}[m] \xrightarrow{c[m']} a[m].c[m']|\overline{a}[m]}|_{\text{L}}.}{a[m].c|\overline{a}[m] \xrightarrow{c[m']} a[h].c[m']|\overline{a}[h]}\text{equiv.}$$

(7)

Then, it is clear that

$$t_1; t_2 : a.c \mid \overline{a} \xrightarrow{\tau[m]} a[m].c|\overline{a}[m] \xrightarrow{c[m']} a[h].c[m']|\overline{a}[h]$$

and yet since $m \notin \text{key}(a[h].c[m']|\overline{a}[h])$, t_1 is not seen as a structural cause of t_2 according to Definition 15, even if it should based on intuitive understanding of concurrency. □

We conjecture that the structural causality could be adapted to account for the substitution of bound keys, but that it will make the definitions quite tedious, since the structural cause relation is purely local.

5.2. Recalling RCCS's Concurrencies

It is relatively easy to adapt our proved labeled to RCCS, no matter which declension of the calculus you consider [9, 16, 22, 23, 20]. Below, we look at the "early" version of RCCS [16, 22] because, to our knowledge, it is the only version that received a syntactical definition of concurrency, relying on memory inclusion [22, Definition 3.11] or disjointness [16, Definition 7]. This version has the heaviest notation, since transitions are labeled with the memory of the thread executing, in addition to the label, but it is immediate to add prefixes to those labels. We briefly remind this system below, and refer to the original presentations [16, 22] for more details. We do not consider recursive definitions, briefly discussed in some versions of RCCS.

5.2.1. Syntax and Semantics of RCCS

The CCS processes used to build RCCS processes follow a slightly different presentation from Sect. 2.1, since the prefix operator can appear only below a n-ary sum. This allows to combine three operators and two rules into one:

- letting $n = 0$ allows to represent 0, letting $n = 1$ allows to recover the usual prefix, and any $n > 1$ represents the sum,
- the rule (also called act.) subsumes the rules for the prefix and the sum.

For simplicity, we will however generally use (guarded) binary sum, written $+$, write $\alpha.P$ for $\alpha.P + 0$ [9, Sect. 2.2], and define the structural equivalence using this binary sum (Definition 21).

Definition 20 (RCCS Processes)**.** The set of *reversible processes* R is built on top of the set of CCS processes by adding memories to the threads:

$$\begin{aligned}
P, Q &\coloneqq P \mid Q \mid \textstyle\sum_{i\geqslant 0}\lambda_i.P_i \mid P\backslash a && \text{(CCS Processes)}\\
m &\coloneqq \langle\rangle \mid \langle 1\rangle \cdot m \mid \langle 2\rangle \cdot m \mid \langle m', a, P\rangle \cdot m \mid \langle \star, \alpha, P\rangle \cdot m && \text{(Memory)}\\
T &\coloneqq m \rhd P && \text{(Reversible Threads)}\\
R, S &\coloneqq T \mid R \mid S \mid R\backslash a && \text{(RCCS Processes)}
\end{aligned}$$

We let $\mathrm{nm}(m) = \{\alpha \mid \alpha \in \mathsf{N} \text{ or } \overline{\alpha} \in \overline{\mathsf{N}} \text{ occurs in } m\}$ be the set of (co-)names occurring in m.

Definition 21 (Structural equivalence)**.** We write $\equiv_{+,\backslash,\alpha}$ the congruence on CCS terms obtained by the symmetric and transitive closure of the following equations, letting $=_\alpha$ being the usual α-equivalence on labels:

$$P + 0 \equiv P \qquad P + Q \equiv Q + P$$
$$(P_1 + P_2) + P_3 = P_1 + (P_2 + P_3) \qquad P \equiv Q \quad \text{if } P =_\alpha Q$$

Structural equivalence on R is the smallest equivalence relation generated by

the following rules:

$$R|S \equiv S|R \qquad \text{(Composition Symmetry)}$$

$$(R_1|R_2)|R_3 \equiv R_1|(R_2|R_3) \qquad \text{(Composition Associativity)}$$

$$\frac{P \equiv_{+,\backslash,\alpha} Q}{m \rhd P \equiv m \rhd Q} \qquad \text{(CCS congruence)}$$

$$m \rhd (P \mid Q) \equiv (\langle 1\rangle.m \rhd P) \mid (\langle 2\rangle.m \rhd Q) \qquad \text{(Distribution of Memory)}$$

$$m \rhd P\backslash a \equiv (m \rhd P)\backslash a \text{ with } a \notin \mathrm{nm}(m) \qquad \text{(Scope of Restriction)}$$

The (Distribution of Memory) rule is the reason why this formalism has often been dubbed "dynamic" [20], since the memory can "move" during execution.

Notation 2. We let $\zeta = \alpha \mid \alpha^-$ be a *directed action* and μ ranges over memories and memory pairs. We write $m \in \mu$ if $\mu = m$ or if $\mu = \{m, m'\}$, and, accordingly, $m_1 \cap m_2 = m$ if $m \in m_1$ and $m \in m_2$. Finally, given two memories m_1, m_2, we write $m_1 \sqsubset m_2$ if $\exists m$ such that $m \cdot m_1 = m_2$.

Definition 22 (Replacement operator)**.** The operation @ is defined as follows:

$$\begin{aligned}
(R|S)_{m_2@m_1} &= R_{m_2@m_1}|S_{m_2@m_1} \\
(R\backslash a)_{m_2@m_1} &= (R_{m_2@m_1})\backslash a && \text{(If } a \notin m_2\text{)} \\
(\langle \star, \alpha, Q\rangle \cdot m_1 \rhd P)_{m_2@m_1} &= \langle m_2, \alpha, Q\rangle \cdot m_1 \rhd P \\
R_{m_2@m_1} &= R && \text{(In all the remaining cases)}
\end{aligned}$$

The forward and backward LTS for RCCS, that we denote $\xrightarrow{\mu:\zeta}\!\!\!\!\!\!\rightarrow = \xrightarrow{\mu:\zeta} \cup \overset{\mu:\zeta}{\rightsquigarrow}$, is given in Fig. 5. In RCCS, the loop lemma [22, Lemme 2.2.1] also holds, and we write t^- the reverse of t.

5.2.2. Definitions of Concurrencies

Concurrency on coinitial Transitions. We first remind of the original definition of concurrency on coinitial transitions.

Definition 23 (Concurrency on coinitial transitions in RCCS [16, Definition 7])**.** Let $t_1 = R \xrightarrow{\mu_1:\zeta_1} S_1$ and $t_2 = R \xrightarrow{\mu_2:\zeta_2} S_2$ be two coinitial transitions, t_1 and t_2 are said to be *concurrent* if $\mu_1 \cap \mu_2 = \emptyset$, and we write $t_1 \smile_i^{\mathrm{o}} t_2$.

$$\frac{}{(m \rhd \lambda.P + Q) \xrightarrow{m:\lambda} \langle \star, \lambda, Q \rangle \cdot m \rhd P}\text{act.}$$

$$\frac{}{\langle \star, \lambda, Q \rangle \cdot m \rhd P \overset{m\,:\,\lambda^-}{\rightsquigarrow} m \rhd (\lambda \cdot P + Q)}\text{act.}^-$$

$$\frac{R \xrightarrow{m_1:\lambda} R' \qquad S \xrightarrow{m_2:\overline{\lambda}} S'}{R \mid S \xrightarrow{m_1,m_2:\tau} R'_{m_2@m_1} \mid S'_{m_1@m_2}}\text{syn.}$$

$$\frac{R \overset{m_1\,:\,\lambda^-}{\rightsquigarrow} R' \qquad S \overset{m_2\,:\,\overline{\lambda}^-}{\rightsquigarrow} S'}{R_{m_2@m_1} \mid S_{m_1@m_2} \overset{m_1,\,m_2\,:\,\tau^-}{\rightsquigarrow} R' \mid S'}\text{syn.}^-$$

$$\frac{R \xrightarrow{\mu:\zeta}\!\!\!\!\rightarrow R'}{R \mid S \xrightarrow{\mu:\zeta}\!\!\!\!\rightarrow R' \mid S}\text{par.} \qquad \frac{R \xrightarrow{\mu:\zeta}\!\!\!\!\rightarrow R' \qquad \zeta \notin \{a, \overline{a}, a^-, \overline{a}^-\}}{R\backslash a \xrightarrow{\mu:\zeta}\!\!\!\!\rightarrow R'\backslash a}\text{res.}$$

$$\frac{R_1 \equiv R \qquad R \xrightarrow{\mu:\zeta}\!\!\!\!\rightarrow R' \qquad R' \equiv R'_1}{R_1 \xrightarrow{\mu:\zeta}\!\!\!\!\rightarrow R'_1}\equiv$$

Figure 5: Rules of the labeled transition system (LTS) for RCCS

Concurrency on Composable Transitions. We now remind of the original definition of concurrency on composable transitions.

Definition 24 (Precedence [22, Definition 3.1.1])**.** Given $t = R \xrightarrow{\mu:\zeta}\!\!\!\!\!\rightarrow R'$ and $t' = R' \xrightarrow{\mu':\zeta'}\!\!\!\!\!\rightarrow R''$ two composable transitions, we say that t *precedes* t' if

- t and t' are forward, and $\exists m \in \mu, \exists m' \in \mu'$, and $m \sqsubset m'$,
- t and t' are backward, and $\exists m \in \mu, \exists m' \in \mu'$, and $m' \sqsubset m$.

Definition 25 (Concurrency on composable transitions in RCCS)**.** Two composable transitions t, t' with the same direction are *concurrent* if t does not precedes t', and we write $t \smile^{\circ}_{c} t'$.

5.2.3. On Transitions of Opposite Directions

Neither $\smile^{\circ}_{i}$ nor $\smile^{\circ}_{c}$ account for transitions of opposite directions. For $\smile^{\circ}_{c}$ it is obvious: composable transitions of opposite directions are neither concurrent nor not concurrent, since precedence is not defined on those transitions.

For $\smile^{\circ}_{i}$, even if the original definition does not make any explicit requirement about the direction of the transitions, and could be read as valid if t_1 and t_2 had opposite directions, it actually requires t_1 and t_2 to be both forward or backward. Indeed, for the two transitions

$$t_1 : \langle \star, a, Q' \rangle \cdot \langle\rangle \rhd (b.P + Q) \xrightarrow{\langle \star, a, Q' \rangle \cdot \langle\rangle : b} \langle \star, b, Q \rangle \cdot \langle \star, a, Q' \rangle \cdot \langle\rangle \rhd P$$
$$t_2 : \langle \star, a, Q' \rangle \cdot \langle\rangle \rhd (b.P + Q) \overset{\langle\rangle \,:\, a^{-}}{\rightsquigarrow} \langle\rangle \rhd a.(b.P + Q) + Q'$$

we have $t_1 \smile^{\circ}_{i} t_2$, since $\langle \star, a, Q' \rangle \cdot \langle\rangle \cap \langle\rangle = \emptyset$. However, the intuitive understanding of concurrency (as well as the sideways diamonds) shows that those two transitions should actually *not* be concurrent.

On top of appearing incomplete, those definitions further prevents checking the validity of (Correctness of Concurrencies). Given two composable transitions t_1 and t_2, it makes no sense to wonder whether

$$t_1 \smile^{\circ}_{c} t_2 \iff t_1^{-} \smile^{\circ}_{i} t_2.$$

Indeed, since there will be transitions of opposite directions on one side of the implication, and since both $\smile_c^{\circ}$ and $\smile_i^{\circ}$ requires both transitions to have the same direction, one cannot compare the two relations.

5.3. Defining Proved RCCS

We define a proved declension of RCCS exactly like we did for CCSK in Sect. 3.1, by enriching the labels and letting the proved LTS propagate them. Many optimizations could be done (ignoring direction, replacing memories with identifiers as frequently done in subsequent versions of RCCS, etc.), but we focus on providing a simple definition, to ease the proof burden in Sect. 5.4, when we prove that enriched labels give a notion of concurrency equivalent to the previous ones (when both are applicable).

We begin by defining the enhanced labels and the proved LTS first. Note that since action and prefixes are mixed, and since sum are not "preserved" as main operator after a reduction, as opposed to CCSK, there is no need for the $+_{\mathrm{L}}$ and $+_{\mathrm{R}}$ annotations anymore, hence we can simplify Definition 3 as follows:

Definition 26 (Enhanced labels (bis))**.** Let υ, υ_{L} and υ_{R} range over strings in $\{|_{\mathrm{L}}, |_{\mathrm{R}}\}^*$, *enhanced labels* are defined as

$$\theta \coloneqq \upsilon\zeta \parallel \upsilon\overline{\zeta} \parallel \upsilon\langle |_{\mathrm{L}}\upsilon_{\mathrm{L}}\zeta, |_{\mathrm{R}}\upsilon_{\mathrm{R}}\overline{\zeta}\rangle$$

And we let

$$\ell(\upsilon\zeta) = \zeta \qquad \ell(\langle |_{\mathrm{L}}\upsilon_{\mathrm{L}}\alpha, |_{\mathrm{R}}\upsilon_{\mathrm{R}}\overline{\alpha}\rangle) = \tau$$
$$\ell(\upsilon\overline{\zeta}) = \overline{\zeta} \qquad \ell(\langle |_{\mathrm{L}}\upsilon_{\mathrm{L}}\alpha^-, |_{\mathrm{R}}\upsilon_{\mathrm{R}}\overline{\alpha^-}\rangle) = \tau^-$$

In this particular case, since the congruence relation is needed because of the (Distribution of Memory) rule, we keep it, but remove the (Composition Symmetry) and (Composition Associativity) rules, as they do not fare well with proved labels (Sect. 6). As a consequence, we also need to replace the par. rule with two rules, $\mathrm{par.}_{\mathrm{L}}$ and $\mathrm{par.}_{\mathrm{R}}$, as presented in Fig. 6. And, from now on, we will assume that the structural congruence used by both systems does not contain (Composition Symmetry) nor (Composition Associativity).

$$\frac{}{(m \rhd \lambda.P + Q) \xrightarrow{m:\lambda} \langle \star, \lambda, Q \rangle \cdot m \rhd P}\text{act.} \qquad \frac{R \xrightarrow{\mu:\theta}\!\!\!\!\rightarrow R'}{R \mid S \xrightarrow{\mu:|_{\mathrm{L}}\theta}\!\!\!\!\rightarrow R' \mid S}\text{par.}_{\mathrm{L}}$$

$$\frac{}{\langle \star, \lambda, Q \rangle \cdot m \rhd P \overset{m\,:\,\lambda^-}{\rightsquigarrow} m \rhd (\lambda \cdot P + Q)}\text{act.}^- \qquad \frac{S \xrightarrow{\mu:\theta}\!\!\!\!\rightarrow S'}{R \mid S \xrightarrow{\mu:|_{\mathrm{R}}\theta}\!\!\!\!\rightarrow R \mid S'}\text{par.}_{\mathrm{R}}$$

$$\frac{R \xrightarrow{\mu:\theta}\!\!\!\!\rightarrow R' \qquad \ell(\theta) \notin \{a, \overline{a}, a^-, \overline{a}^-\}}{R\backslash a \xrightarrow{\mu:\theta}\!\!\!\!\rightarrow R'\backslash a}\text{res.}$$

$$\frac{R \xrightarrow{m_1:\theta_{\mathrm{L}}\lambda} R' \qquad S \xrightarrow{m_2:\theta_{\mathrm{R}}\overline{\lambda}} S'}{R \mid S \xrightarrow{m_1,m_2:\langle|_{\mathrm{L}}\theta_{\mathrm{L}}\lambda,|_{\mathrm{R}}\theta_{\mathrm{R}}\overline{\lambda}\rangle} R'_{m_2@m_1} \mid S'_{m_1@m_2}}\text{syn.}$$

$$\frac{R \overset{m_1\,:\,\lambda^-}{\rightsquigarrow} R' \qquad S \overset{m_2\,:\,\overline{\lambda}^-}{\rightsquigarrow} S'}{R_{m_2@m_1} \mid S_{m_1@m_2} \overset{m_1,\,m_2\,:\,\langle|_{\mathrm{L}}\theta_{\mathrm{L}}\lambda^-,|_{\mathrm{R}}\theta_{\mathrm{R}}\overline{\lambda^-}\rangle}{\rightsquigarrow} R' \mid S'}\text{syn.}^-$$

$$\frac{R_1 \equiv R \qquad R \xrightarrow{\mu:\theta}\!\!\!\!\rightarrow R' \qquad R' \equiv R'_1}{R_1 \xrightarrow{\mu:\theta}\!\!\!\!\rightarrow R'_1}\equiv$$

Figure 6: Rules of the proved labeled transition system (LTS) for RCCS

Action
$\zeta \lessdot \theta$
$\overline{\zeta} \lessdot \theta$

For $d \in \{\mathrm{L}, \mathrm{R}\}$

Palallel Group	
$\vert_d\theta \lessdot \vert_d\theta'$	if $\theta \lessdot \theta'$
$\langle \theta_\mathrm{L}, \theta_\mathrm{R} \rangle \lessdot \theta$	if $\exists d$ s.t.$\theta_d \lessdot \theta$
$\theta \lessdot \langle \theta_\mathrm{L}, \theta_\mathrm{R} \rangle$	if $\exists d$ s.t.$\theta \lessdot \theta_d$
$\langle \theta_\mathrm{L}, \theta_\mathrm{R} \rangle \lessdot \langle \theta'_\mathrm{L}, \theta'_\mathrm{R} \rangle$	if $\exists d$ s.t.$\theta_d \lessdot \theta'_d$

Figure 7: Dependency Relation on Enhanced Keyed Labels

Definition 27 (Dependency relation)**.** The *dependency relation* on enhanced keyed labels is induced by the axioms of Fig. 7.

It should be noted that this relation is the same as in the forward-only CCS, further illustrating how resilient the proved label technique is. Transitions, traces and concurrencies are defined as in Definitions 7, 11 and 12.

Exactly like for CCSK with Lemma 2, it is easy to prove the adequacy of the proved system w.r.t. the original one:

Lemma 14 (Adequacy of the proved labeled transition system)**.** *The transition* $R \xrightarrow{\mu:\zeta}\!\!\!\!\!\rightarrow S$ *can be derived using Fig. 5 iff* $R \xrightarrow{\mu:\theta}\!\!\!\!\!\rightarrow S$ *with* $\ell(\theta) = \zeta$ *can be derived using Fig. 6.*

Proof. This is obvious, and we write f the mapping from ζ to θ. □

5.4. Adequacies of RCCS's Concurrencies

We now prove that the original two definitions of concurrency coincide with the one resulting from adopting proved labels for RCCS.

5.4.1. On coinitial Traces

Theorem 4. *For all co-intial transitions with the same direction* $t_1 = R \xrightarrow{\mu_1:\zeta_1} S_1$ *and* $t_2 = R \xrightarrow{\mu_2:\zeta_2} S_2$*,* $t_1 \smile_i^o t_2$ *iff* $\neg(f(\zeta_1) \lessdot f(\zeta_2))$*.*

Proof. We start by proving the left-to-right direction first, by case on the structure of R:

$m \rhd P$ Then we proceed by induction on the size of P, and by case on the structure of P:

0 This is vacuously true, since 0 cannot reduce.

$\sum \alpha_i.P_i$ Then all the transitions from $m \rhd P$ are of the form

$$m \rhd \sum \alpha_i.P_i \xrightarrow{m:\alpha_i} m' \cdot m \rhd P_i$$

But since $m \cap m \neq \emptyset$, any two such transitions are not pairwise concurrent. Since we always have that $\alpha_i \lessdot \theta$, we have that $f(\zeta_1) \lessdot f(\zeta_2)$.

$P|Q$ Then $m \rhd P|Q$ cannot reduce, without using (Distribution of Memory) to become of the form $R_1|R_2$, that we study next.

$R_1|R_2$ Then, every transition that $R_1|R_2$ can perform has for memory either a pair, or a memory prefixed by $\langle 1 \rangle$ or by $\langle 2 \rangle$. We proceed by case on μ_1 and μ_2:

If μ_1 and μ_2 are prefixed by the same number Then since $|_d\theta \lessdot |_d\theta'$ if $\theta \lessdot \theta'$, we can proceed by induction.

If μ_1 and μ_2 are prefixed by different numbers Then the transitions are concurrent, and since neither $|_{\mathrm{L}}\theta \lessdot |_{\mathrm{R}}\theta'$ nor $|_{\mathrm{R}}\theta \lessdot |_{\mathrm{L}}\theta'$ hold, we are done with this case.

If μ_1 or μ_2 is a pair Then we simply reason on its elements, exactly like the rules of Fig. 7 concerned with tuples $\langle \theta_{\mathrm{L}}, \theta_{\mathrm{R}} \rangle$ decompose them to assess whenever they are dependent of other labels.

$(a)R$ Then this is immediate by induction.

For the converse direction, it suffices to observe the rules of Fig. 7 and to note that all the rules imply that the memories of the process initiating the two transitions must have a non-empty intersection, hence providing the desired result. □

5.4.2. On Composable Transitions

Using the following lemma, it is enough to prove the adequacy of our notion for one direction only:

Lemma 15. *For all composable forward transitions t_1, t_2,*

$$(t_1 \textit{ precedes } t_2 \iff f(\zeta_1) \lessdot f(\zeta_2)) \iff (t_2^- \textit{ precedes } t_1^- \iff f(\zeta_2) \lessdot f(\zeta_1))$$

Proof. This is immediate by symmetry of Definitions 24 and 27. □

Lemma 16. *For all composable forward transitions, $t_1 = R \xrightarrow{\mu_1:\zeta_1} S_1$ and $t_2 = S_1 \xrightarrow{\mu_2:\zeta_2} S_2$, $t_1 \smile_i^o t_2$ iff $\neg(f(\zeta_1) \lessdot f(\zeta_2))$.*

Proof. We need to prove that t_1 precedes t_2 iff $f(\zeta_1) \lessdot f(\zeta_2)$. We reason by case on the last rule of the derivation for t_1:

act. Then, letting $\mu = m$, $\mu_2 = \langle \star, \lambda, Q \rangle \cdot m$ for some λ and Q, and hence $m_1 \sqsubset m_2$ and t_1 precedes t_2. That $f(\zeta_1) \lessdot f(\zeta_2)$ is also immediate.

par.$_{\mathrm{L}}$ Then $R = R_1|R_2$, $S_1 = T_1|T_2$, $S_2 = T_3|T_4$ and we proceed by case on the last rule in the derivation of t_2:

par.$_{\mathrm{L}}$ Then we proceeds by induction on the trace $R_1 \xrightarrow{\mu_1:\zeta_1} T_1 \xrightarrow{\mu_2:\zeta_2} T_3$.

par.$_{\mathrm{R}}$ Then t_1 cannot precede t_2, and $f(\zeta_1) \lessdot f(\zeta_2)$.

syn. Then t_1 precedes t_2 (resp. $f(\zeta_1) \lessdot f(\zeta_2)$) iff t_1' precedes t_2' (resp. $f(\zeta_1) \lessdot f(\zeta_2')$) in $t_1'; t_2' : R_1 \xrightarrow{\mu_1:\zeta_1} T_1 \xrightarrow{\mu_2':\zeta_2'} T_3$, and we proceed by induction

syn. and par.$_{\mathrm{L}}$ Those two cases are similar to the previous one.

res. and $\equiv$ are immediate by induction hypothesis. □

Theorem 5. *For all different composable transitions with the same direction $t_1 = R \xrightarrow{\mu_1:\zeta_1} S_1$ and $t_2 = S_1 \xrightarrow{\mu_2:\zeta_2} S_2$, $t_1 \smile_i^o t_2$ iff $\neg(f(\zeta_1) \lessdot f(\zeta_2))$ if t_1 and t_2 are forward, or if $\neg(f(\zeta_2) \lessdot f(\zeta_1))$ if t_1 and t_2 are backward.*

Proof. This is an immediate consequence of Lemmas15 and 15. □

5.5. Reversible and Identified CCS

We refer to the original paper [21] for the precise definition of (this declension of) RCCS, and only recall the strict minimum below. In a nutshell, this calculus endows RCCS processes with a *seed* [21, Definition 4], which is an *identifier patterns* [21, Definition 1] that dynamically generates the identifiers for each transition, and that can get split [21, Definition 3] between threads if needed. Being able to know ahead of time the identifer generated for each transition was leveraged to offer an original definition of concurrency, where identifiers need to be compatible [21, Definition 12]—written $i_1 \perp i_2$—or not downstream, both conditions essentially stating that the transition involved different threads. We keep the development rather informal not to burden the reader, but the proofs could be worked out in details based on the sketches we provide below.

This calculus also explored different types of sums, but we restrict ourselves to the "classical one", denoted + as usual.

Definition 28 (Concurrency)**.** Two different coinitial transitions

$$t_1 : \mathsf{s} \circ m \rhd P \xrightarrow{\alpha_1[i_1]} \mathsf{s}_1 \circ m_1 \rhd P_1 \text{ and } t_2 : \mathsf{s} \circ m \rhd P \xrightarrow{\alpha_2[i_2]} \mathsf{s}_2 \circ m_2 \rhd P_2$$

are *concurrent* iff

- t_1 and t_2 are forward transitions and $i_1 \perp i_2$;
- t_1 is a forward and t_2 is a backward transition and i_1 (or i_1^1 and i_1^2 if $i_1 = i_1^1 \oplus i_1^2$) is not downstream of ip_{t_2} (or $\mathsf{ip}_{t_2}^1$ nor $\mathsf{ip}_{t_2}^2$);
- t_1 and t_2 are backward transitions.

It is easy to similarly adjust the system to use proved labels, and then to prove its adequacy in the sense of Lemma 14—we will also write f the mapping from labels to proved labels. Note that the dependency relation is defined as with RCCS here: since the sum operator is not preserved, it is not needed to account for it in the proved label.

Theorem 6. *For all* $\mathsf{s} \circ P \xrightarrow{\alpha_1[i_1]} \mathsf{s}_1 \circ P_1$ *and* $\mathsf{s} \circ P \xrightarrow{\alpha_2[i_2]} \mathsf{s}_2 \circ P_2$, $i_1 \perp i_2$ *are concurrent iff* $f(\alpha_1) \lessdot f(\alpha_2)$ *does not hold.*

Proof. For forward transition, it is not difficult to observe that, given two different coinitial transitions $\mathsf{s} \circ P \xrightarrow{\alpha_1[i_1]} \mathsf{s}_1 \circ P_1$ and $\mathsf{s} \circ P \xrightarrow{\alpha_2[i_2]} \mathsf{s}_2 \circ P_2$, $i_1 \perp i_2$ iff $\neg(f(i_1 : \alpha_1) \lessdot f(i_2 : \alpha_2))$:

- both transitions cannot come from reducing the very same action, which means that P must have a different operator at top level,
- if they result from the execution of the left- and right-hand-side of the same sum operator, then they get assigned the same identifier, and since they will both be labeled with actions, they will not be concurrent according to both definitions,
- if they result from the execution of a multi-threaded process, then it is easy to observe that the condition on the incompatibility of the identifiers match the definition of dependencies, as transitions resulting from synchronizations are concurrent iff their components are in both cases.

For transitions with opposite directions, the "downstream" condition essentially ensures that the identifiers originate from different seeds, e.g., from different threads. That this condition is equivalent to the inexistence of a dependency between proved labels on transitions of opposite direction is a direct, though tedious, result of the unfolding of both definitions.

For backward transitions, it is immediate: any two backward transitions are concurrent according to Definition 28, and we have this result as well for proved labels, by adapting the proof for proved CCSK (Lemma 10) to this proved identified RCCS. □

6. Structural Congruence and Other Criteria

Causality for a semantics of concurrent computation should satisfy a variety of critera, the diamonds being the starting point, and causal consistency being arguably the most important for reversible systems. This section aims at briefly presenting additional criteria.

Concurrency-Preserving Structural Congruences. "Denotationality" [8, Section 6] is a criteria stating that structural congruence should be preserved by the causal semantics. Unfortunately, our system only vacuously meets this criteria—since it does not possess a structural congruence. The "usual" structural congruence is missing from all the proved transition systems [26, 32, 33, 37], or missing the associativity and commutativity of the parallel composition [34, p. 242]. While adding such a congruence would benefits the expressiveness, making it interact nicely with the derived proof system *and* the reversible features [7, Section 4][38] is a challenge we prefer to postpone.

Comparing with concurrency inspired by reversible π-calculus. It is possible to restrict the definition of concurrency for reversible π-calculi extending CCSK back to CCSK. We did it in Sect. 5.1 for a particular line of work [18], but it is not the only one that can be the source of comparison. Indeed, a similar work could have been done by restricting concurrency for e.g., reversible higher-order π-calculus [39, Definition 9], reversible π-calculus [40, Definition 4.1] or croll-π [41, Definition 1]. However, it seems more constructive to extend our definition to a reversible π-calculus rather than proceeding the other way around.

Optimality, Parabolic Lemma, and RPI. The optimality criteria is the adequacy of the concurrency definitions for the LTS and for the reduction semantics [40, Theorem 5.6]. While this criteria requires a reduction semantics and a notion of reduction context to be formally proven, we believe it is easy to convince oneself that the gist of this property—the fact that non-τ-transitions are concurrent iff there exists a "closing" context in which the resulting τ-transitions are still concurrent—holds in our system: as concurrency on τ-transitions is defined in terms of concurrency of its elements (e.g., $\langle \theta^1_{\mathrm{R}}, \theta^1_{\mathrm{L}} \rangle \smile \langle \theta^2_{\mathrm{R}}, \theta^2_{\mathrm{L}} \rangle$ iff $\theta^1_d \smile \theta^2_d$ for at least one $d \in \{\mathrm{L}, \mathrm{R}\}$), this criteria is obtained "for free".

Properties such as the parabolic lemma [16, Lemma 10]—"any trace is equivalent to a backward trace followed by a forward trace"—or "RPI" [11, Definition 3.3]—"reversing preserves independence", i.e., $t \smile t'$ iff $t^\bullet \smile t'$–follow immediately, by our definition of concurrencies for this latter. We furthermore be-

lieve that "baking in" the RPI principle in definitions of reversible concurrencies should become the norm, as it facilitates proofs and forces to have $t_1 \smile t_2$ iff $t_1^\bullet \smile t_2^\bullet$, which seems a very sound principle. How those properties logically relates to our (Correctness of Concurrencies) is another venue that could be fruitlful to explore.

7. Conclusion and Perspectives

We believe our proposal to be not only elegant, but also extremely resilient and easy to work with. It should be stressed that it *does not* require to observe the directions, but also ignore keys or identifiers, that should in our opinion only be technical annotations disallowing processes that have been synchronized to backtrack independently. We had previously defended that identifier should be considered only up to isomorphisms [9, p. 13], or explicitly generated by a built-in mechanism [21, p. 152], and re-inforce this point of view here. From there, much can be done. A first interesting line of work would be to compare our syntactical definition with the semantical definition of concurrency in models of RCCS [9, 12, 23] and CCSK [6, 13, 24]. Of course, as we already mentioned, extending this definition to reversible π-calculi, taking inspiration from e.g., the latest development in forward-only π [37], would allow to re-inforce the interest and solidity of this technique.

Another interesting track would be to consider infinite extensions of CCSK, since infinite behaviors in the presence of reversibility is not well-understood nor studied: an attempt to extend algebras of communicating processes [42], including recursion, seems to have been unsuccessful [43]. A possible approach would be to define recursion and iteration in CCSK, to extend our definition of concurrency to those infinite behaviors[8] and to attempt to reconstruct the separation results from the forward-only paradigm [45]. Whether finer, "reversible",

[8] This is, by the way, the reason why we piced this particular formalism and presentation over concurrent formalisms for CCS. Older presentations [30, 31] are indeed better equipped to possibly accommodate replication for reversible calculi [44], in our opinion.

equivalences can preserve this distinction despite the greater flexibility provided by backward transitions is an open problem. Another interesting point is the study of infinite behaviors that duplicate past events, including their keys or memories: whether this formalism could preserve causal consistency, or what benefits there would be in tinkering this property, is also an open question.

Last but not least, these last investigations would require to define and understand relevant properties, or metrics, for reversible systems. In the forward-only world, termination or convergence were used to compare infinite behaviors [45], and additional criteria were introduced to study causal semantics [8]. Those properties may or may not be suited for reversible systems, but it is difficult to decide as they sometimes even lack a definition. This could help in solving the more general question of deciding *what* it is that we want to observe and assess when evaluating reversible, concurrent systems [46, 47].

Acknowledgments

I would like to thank Doriana Medić for suggesting that I adapt the definition of concurrency for a reversible π-calculus extending CCSK [18] and compare it to the concurrency developed in this paper, as done in Sect. 5.1. Peter Browning—who is working on implementing CCSK [48]—and Neea Rusch gave me an early feedback on the Preamble that helped me in making it more accessible, and I would like to thank them for their time. I really appreciated presenting a preliminary version of this work [27] at the 14th Conference on Reversible Computation, and would like to thank Claudio Antares Mezzina for organizing it, and him, Ivan Lanese and Irek Ulidowski for the very interesting discussions that followed. I am also extremely thankful to the reviewers for their careful reading of the first version of this technical paper, and for their enlightening suggestions.

References

[1] L. Kristiansen, Reversible computing and implicit computational complexity, Science of Computer Programming 213 (2022) 102723. doi:10.1016/j.scico.2021.102723.

[2] C. E. McDowell, D. P. Helmbold, Debugging concurrent programs, ACM Computing Surveys 21 (4) (1989) 593–622. doi:10.1145/76894.76897.

[3] H. Wayne, Why don't people use formal methods? (01 2019). URL https://www.hillelwayne.com/post/why-dont-people-use-formal-methods/

[4] C. Aubert, R. Horne, C. Johansen, Diamonds for security: A non-interleaving operational semantics for the applied pi-calculus, in: B. Klin, S. Lasota, A. Muscholl (Eds.), 33rd International Conference on Concurrency Theory, Vol. 243 of Leibniz International Proceedings in Informatics, Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2022, pp. 30:1–30:26. doi:10.4230/LIPIcs.CONCUR.2022.30.

[5] I. Phillips, I. Ulidowski, Reversing algebraic process calculi, The Journal of Logic and Algebraic Programming 73 (1-2) (2007) 70–96. doi:10.1016/j.jlap.2006.11.002.

[6] I. Phillips, I. Ulidowski, Reversibility and models for concurrency, Electronic Notes in Theoretical Computer Science 192 (1) (2007) 93–108. doi:10.1016/j.entcs.2007.08.018.

[7] I. Lanese, I. Phillips, Forward-reverse observational equivalences in CCSK, in: S. Yamashita, T. Yokoyama (Eds.), Reversible Computation - 13th International Conference, RC 2021, Virtual Event, July 7-8, 2021, Proceedings, Vol. 12805 of Lecture Notes in Computer Science, Springer, 2021, pp. 126–143. doi:10.1007/978-3-030-79837-6_8.

[8] I. Cristescu, J. Krivine, D. Varacca, Rigid families for CCS and the π-calculus, in: M. Leucker, C. Rueda, F. D. Valencia (Eds.), Theoretical Aspects of Computing - ICTAC 2015 - 12th International Colloquium Cali, Colombia, October 29-31, 2015, Proceedings, Vol. 9399 of Lecture Notes in Computer Science, Springer, 2015, pp. 223–240. doi:10.1007/978-3-319-25150-9_14.

[9] C. Aubert, I. Cristescu, How reversibility can solve traditional questions: The example of hereditary history-preserving bisimulation, in: I. Konnov, L. Kovács (Eds.), 31st International Conference on Concurrency Theory, CONCUR 2020, September 1–4, 2020, Vienna, Austria, Vol. 2017 of Leibniz International Proceedings in Informatics, Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2020, pp. 13:1–13:24. doi:10.4230/LIPIcs.CONCUR.2020.13.

[10] I. Lanese, From reversible semantics to reversible debugging, in: J. Kari, I. Ulidowski (Eds.), Reversible Computation - 10th International Conference, RC 2018, Leicester, UK, September 12-14, 2018, Proceedings, Vol. 11106 of Lecture Notes in Computer Science, Springer, 2018, pp. 34–46. doi:10.1007/978-3-319-99498-7_2.

[11] I. Lanese, I. C. C. Phillips, I. Ulidowski, An axiomatic approach to reversible computation, in: J. Goubault-Larrecq, B. König (Eds.), Foundations of Software Science and Computation Structures - 23rd International Conference, FOSSACS 2020, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2020, Dublin, Ireland, April 25-30, 2020, Proceedings, Vol. 12077 of Lecture Notes in Computer Science, Springer, 2020, pp. 442–461. doi:10.1007/978-3-030-45231-5_23.

[12] C. Aubert, I. Cristescu, Reversible barbed congruence on configuration structures, in: S. Knight, A. Lluch Lafuente, I. Lanese, H. T. Vieira (Eds.), ICE 2015, Vol. 189 of Electronic Proceedings in Theoretical Computer Science, 2015, pp. 68–95. doi:10.4204/EPTCS.189.7.

[13] E. Graversen, I. C. C. Phillips, N. Yoshida, Event structure semantics of (controlled) reversible CCS, Journal of Logical and Algebraic Methods in Programming 121 (2021) 100686. doi:10.1016/j.jlamp.2021.100686.

[14] R. Milner, A Calculus of Communicating Systems, Lecture Notes in Computer Science, Springer-Verlag, 1980. doi:10.1007/3-540-10235-3.

[15] D. Sangiorgi, D. Walker, The Pi-calculus, Cambridge University Press, 2001.

[16] V. Danos, J. Krivine, Reversible communicating systems, in: P. Gardner, N. Yoshida (Eds.), CONCUR 2004 - Concurrency Theory, 15th International Conference, London, UK, August 31 - September 3, 2004, Proceedings, Vol. 3170 of Lecture Notes in Computer Science, Springer, 2004, pp. 292–307. doi:10.1007/978-3-540-28644-8_19.

[17] Arpit, D. Kumar, Calculus of concurrent probabilistic reversible processes, in: ICCCT-2017: Proceedings of the 7th International Conference on Computer and Communication Technology, ICCCT-2017, Association for Computing Machinery, New York, NY, USA, 2017, pp. 34–40. doi:10.1145/3154979.3155004.

[18] D. Medić, C. A. Mezzina, I. Phillips, N. Yoshida, A parametric framework for reversible π-calculi, Information and Computation 275 (2020) 104644. doi:10.1016/j.ic.2020.104644.

[19] C. A. Mezzina, V. Koutavas, A safety and liveness theory for total reversibility, in: F. Mallet, M. Zhang, E. Madelaine (Eds.), 11th International Symposium on Theoretical Aspects of Software Engineering, TASE 2017, Sophia Antipolis, France, September 13-15, IEEE, 2017, pp. 1–8. doi:10.1109/TASE.2017.8285635.

[20] I. Lanese, D. Medić, C. A. Mezzina, Static versus dynamic reversibility in CCS, Acta Informatica (Nov. 2019). doi:10.1007/s00236-019-00346-6.

[21] C. Aubert, D. Medić, Explicit identifiers and contexts in reversible concurrent calculus, in: S. Yamashita, T. Yokoyama (Eds.), Reversible Computation - 13th International Conference, RC 2021, Virtual Event, July 7-8, 2021, Proceedings, Vol. 12805 of Lecture Notes in Computer Science, Springer, 2021, pp. 144–162. doi:10.1007/978-3-030-79837-6_9.

[22] J. Krivine, Algèbres de processus réversible - programmation concurrente déclarative, Ph.D. thesis, Université Paris 6 & INRIA Rocquencourt (2006).
URL https://tel.archives-ouvertes.fr/tel-00519528

[23] C. Aubert, I. Cristescu, Contextual equivalences in configuration structures and reversibility, Journal of Logical and Algebraic Methods in Programming 86 (1) (2017) 77–106. doi:10.1016/j.jlamp.2016.08.004.

[24] I. Ulidowski, I. Phillips, S. Yuen, Concurrency and reversibility, in: S. Yamashita, S. Minato (Eds.), Reversible Computation - 6th International Conference, RC 2014, Kyoto, Japan, July 10-11, 2014. Proceedings, Vol. 8507 of Lecture Notes in Computer Science, Springer, 2014, pp. 1–14. doi:10.1007/978-3-319-08494-7_1.

[25] P. Degano, F. Gadducci, C. Priami, Causality and replication in concurrent processes, in: M. Broy, A. V. Zamulin (Eds.), Perspectives of Systems Informatics, 5th International Andrei Ershov Memorial Conference, PSI 2003, Akademgorodok, Novosibirsk, Russia, July 9-12, 2003, Revised Papers, Vol. 2890 of Lecture Notes in Computer Science, Springer, 2003, pp. 307–318. doi:10.1007/978-3-540-39866-0_30.

[26] P. Degano, C. Priami, Enhanced operational semantics, ACM Computing Surveys 33 (2) (2001) 135–176. doi:10.1145/384192.384194.

[27] C. Aubert, Concurrencies in Reversible Concurrent Calculi, in: C. A. Mezzina, K. Podlaski (Eds.), Reversible Computation - 14th International Conference, RC 2022, Urbino, Italy, July 5-6, 2022, Proceedings, Vol. 13354 of Lecture Notes in Computer Science, Springer, 2022, pp. 146–163. doi:10.1007/978-3-031-09005-9_10.

[28] C. Aubert, Concurrencies in Reversible Concurrent Calculi, technical Report (Mar. 2022).
URL https://hal.archives-ouvertes.fr/hal-03605003

[29] N. Busi, M. Gabbrielli, G. Zavattaro, On the expressive power of recursion, replication and iteration in process calculi, Mathematical Structures in Computer Science 19 (6) (2009) 1191–1222. doi:10.1017/S096012950999017X.

[30] G. Boudol, I. Castellani, A non-interleaving semantics for CCS based on proved transitions, Fundamenta Informaticae 11 (1988) 433–452.

[31] G. Boudol, I. Castellani, Three equivalent semantics for CCS, in: I. Guessarian (Ed.), Semantics of Systems of Concurrent Processes, LITP Spring School on Theoretical Computer Science, La Roche Posay, France, April 23-27, 1990, Proceedings, Vol. 469 of Lecture Notes in Computer Science, Springer, 1990, pp. 96–141. doi:10.1007/3-540-53479-2_5.

[32] G. Carabetta, P. Degano, F. Gadducci, CCS semantics via proved transition systems and rewriting logic, in: C. Kirchner, H. Kirchner (Eds.), 1998 International Workshop on Rewriting Logic and its Applications, WRLA 1998, Abbaye des Prémontrés at Pont-à-Mousson, France, September 1998, Vol. 15 of Electronic Notes in Theoretical Computer Science, Elsevier, 1998, pp. 369–387. doi:10.1016/S1571-0661(05)80023-4.

[33] P. Degano, C. Priami, Proved trees, in: W. Kuich (Ed.), Automata, Languages and Programming, 19th International Colloquium, ICALP92, Vienna, Austria, July 13-17, 1992, Proceedings, Vol. 623 of Lecture Notes in Computer Science, Springer, 1992, pp. 629–640. doi:10.1007/3-540-55719-9_110.

[34] P. Degano, C. Priami, Non-interleaving semantics for mobile processes, Theoretical Computer Science 216 (1-2) (1999) 237–270. doi:10.1016/S0304-3975(99)80003-6.

[35] R. Milner, A Calculus of Communicating Systems, Vol. 92 of Lecture Notes in Computer Science, Springer-Verlag, New York, NY, 1980.

[36] V. Sassone, M. Nielsen, G. Winskel, Models for concurrency: Towards a classification, Theoretical Computer Science 170 (1-2) (1996) 297–348. doi:10.1016/S0304-3975(96)80710-9.

[37] R. Demangeon, N. Yoshida, Causal computational complexity of distributed processes, in: A. Dawar, E. Grädel (Eds.), LICS, Association for Computing Machinery, 2018, pp. 344–353. doi:10.1145/3209108.3209122.

[38] C. Aubert, I. Cristescu, Structural equivalences for reversible calculi of communicating systems (oral comm Research report, Augusta University, communication at ICE 2020 (2020). URL https://hal.archives-ouvertes.fr/hal-02571597

[39] I. Lanese, C. A. Mezzina, J. Stefani, Reversibility in the higher-order π-calculus, Theoretical Computer Science 625 (2016) 25–84. doi:10.1016/j.tcs.2016.02.019.

[40] I. Cristescu, J. Krivine, D. Varacca, A compositional semantics for the reversible p-calculus, in: LICS, IEEE Computer Society, 2013, pp. 388–397. doi:10.1109/LICS.2013.45.

[41] I. Lanese, M. Lienhardt, C. A. Mezzina, A. Schmitt, J. Stefani, Concurrent flexible reversibility, in: M. Felleisen, P. Gardner (Eds.), Programming Languages and Systems - 22nd European Symposium on Programming, ESOP 2013, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2013, Rome, Italy, March 16-24, 2013. Proceedings, Vol. 7792 of Lecture Notes in Computer Science, Springer, 2013, pp. 370–390. doi:10.1007/978-3-642-37036-6_21.

[42] J. C. M. Baeten, A brief history of process algebra, Theoretical Computer Science 335 (2-3) (2005) 131–146. doi:10.1016/j.tcs.2004.07.036.

[43] Y. Wang, Retracted article: An algebra of reversible computation, SpringerPlus 5 (1) (2016) 1659. doi:10.1186/s40064-016-3229-7.

[44] C. Aubert, Causal consistent replication in reversible concurrent calculi, under revision (Oct. 2021).
URL https://hal.archives-ouvertes.fr/hal-03384482

[45] C. Palamidessi, F. D. Valencia, Recursion vs replication in process calculi: Expressiveness, Bulletin of the EATCS 87 (2005) 105–125.
URL http://eatcs.org/images/bulletin/beatcs87.pdf

[46] C. Aubert, D. Varacca, Processes, systems & tests: Defining contextual equivalences, in: J. Lange, A. Mavridou, L. Safina, A. Scalas (Eds.), Proceedings 14th Interaction and Concurrency Experience, Online, 18th June 2021, Vol. 347 of Electronic Proceedings in Theoretical Computer Science, Open Publishing Association, 2021, pp. 1–21. doi:10.4204/EPTCS.347.1.

[47] C. Aubert, D. Varacca, Processes against tests: On defining contextual equivalences, Journal of Logical and Algebraic Methods in Programming (2022) 100799doi:10.1016/j.jlamp.2022.100799.

[48] P. Browning, C. Aubert, IRDC-CCSK (8 2022).
URL https://github.com/CinRC/IRDC-CCSK