Concurrencies in Reversible Concurrent Calculi

Clément Aubert(✉)

School of Computer and Cyber Sciences, Augusta University, Augusta, USA
caubert@augusta.edu
https://spots.augusta.edu/caubert/

Abstract. The algebraic specification and representation of networks of agents have been greatly impacted by the study of reversible phenomena: reversible declensions of the calculus of communicating systems (CCSK and RCCS) offer new semantic models, finer congruence relations, original properties, and revisits existing theories and results in a finer light. But much remains to be done: concurrency, a central notion in establishing *causal consistency*–a crucial property for reversible systems–, was never given a syntactical definition in CCSK. We remedy this gap by leveraging a definition of concurrency developed for forward-only calculi using proved transition systems, and prove that CCSK still enjoys causal consistency for this elegant and syntactical notion of reversible concurrency. We also compare it to a definition of concurrency inspired by reversible π-calculus, discuss its relation with structural congruence, and prove that it can be adapted to any CCS-inspired reversible system and is equivalent—or refines—existing definitions of concurrency for those systems.

Keywords: Formal semantics · Process algebras · Concurrency

1 Introduction: Reversibility, Concurrency–Interplays

Concurrency Theory is being reshaped by reversibility: fine distinctions between causality and causation [37] contradicted Milner's expansion laws [30, Example 4.11], and the study of causal models for reversible computation led to novel correction criteria for causal semantics—both reversible and irreversible [17]. "Traditional" equivalence relations have been captured syntactically [6], while original observational equivalences were developed [30]: reversibility triggered a global reconsideration of established theories and tools, with the clear intent of providing actionable methods for reversible systems [26], novel axiomatic foundations [31] and original non-interleaving models [4,17,24].

Two Formalisms extend the Calculus of Communicating Systems (CCS) [34]—the godfather of π-calculus [38], among others—with reversible features. Reversible CCS (RCCS) [18] and CCS with keys (CCSK) [37] are similarly the source of most [1,17,32,33]—if not all—of later formalism developed to

C. A. Mezzina and K. Podlaski (Eds.): RC 2022, LNCS 13354, pp. 146–163, 2022.
https://doi.org/10.1007/978-3-031-09005-9_10

enhance reversible systems with some respect (rollback operator, name-passing abilities, probabilistic features, ...). Even if those two systems share a lot of similarities [28], 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 [30, Proposition 4.9] or the impossibility thereof for RCCS [8, 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, easy to manipulate, that satisfies the usual sanity checks.

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 [37]. 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 [37, Propositions 5.10 and 5.19], but to our knowledge no "definitive" definition of concurrency was proposed. Ad-hoc definitions relying on memory inclusion [25, Definition 3.1.1] or disjointness [18, Definition 7] for RCCS, and semantical notions for both RCCS [4–6] and CCSK [24,36,40] have been proposed, but, to our knowledge, none of those have ever been 1. compared to each other, 2. compared to pre-existing forward-only definitions of concurrency.

Our Contribution introduces the first syntactical definition of concurrency for CCSK (Sect. 3.1), by extending the "universal" concurrency developed for forward-only CCS [19], that leveraged *proved* transition systems [22]. We make crucial use of the loop lemma (Lemma 5) to define concurrency between coinitial traces in terms of concurrency between composable traces—a mechanism that considerably reduces the definition and proof burdens: typically, the square property is derived from the sideways and reverse diamonds. We furthermore establish the correctness of this definition by proving the expected reversible properties—causal consistency (Sect. 3.3), among others—and by discussing how our definition relates to definitions of concurrency in similar systems—obtained by porting our technique to RCCS [18,25] and its "identified" declensions [8], or by restricting a notion of concurrency for π-calculus—and to structural congruence (Sect. 4). With respect to this last point, 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.

Additional details are contained in our preliminary technical report [3], i.e. all proofs [3, Sect. B], and the technical justification of the claims made in Sect. 4 about the "universality" of our approach [3, Sect. C].

2 Finite and Reversible Process Calculi

2.1 Finite, Forward-Only CCS

Finite Core CCS. We briefly recall the (forward-only) "finite fragment" of the core of CCS (simply called CCS) following a standard presentation [14].

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.*

Definition 2 (Operators). *We let* P, Q *range over* CCS *processes, defined as usual, using restriction* $(P\backslash\alpha)$, *sum* $(P + Q)$, *prefix* $(\alpha.P)$ *and parallel composition* $(P \mid Q)$. *The inactive process* 0 *is omitted when preceded by a prefix, and the binding power of the operators [34, 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*).*

The labeled transition system for CCS, denoted $\xrightarrow{\alpha}$, is reminded in Fig. 1.

Action and Restriction

$$\frac{}{\alpha.P \xrightarrow{\alpha} P}\ \text{act.} \qquad \alpha \notin \{a, \overline{a}\}\ \frac{P \xrightarrow{\alpha} P'}{P\backslash a \xrightarrow{\alpha} P'\backslash a}\ \text{res.}$$

Parallel Group

$$\frac{P \xrightarrow{\alpha} P'}{P \mid Q \xrightarrow{\alpha} P' \mid Q}\ {\mid_{\text{L}}} \qquad \frac{P \xrightarrow{\lambda} P' \quad Q \xrightarrow{\overline{\lambda}} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'}\ \text{syn.} \qquad \frac{Q \xrightarrow{\alpha} Q'}{P \mid Q \xrightarrow{\alpha} P \mid Q'}\ {\mid_{\text{R}}}$$

Sum Group

$$\frac{P \xrightarrow{\alpha} P'}{Q + P \xrightarrow{\alpha} P'}\ {+_{\text{L}}} \qquad \frac{Q \xrightarrow{\alpha} Q'}{Q + P \xrightarrow{\alpha} Q'}\ {+_{\text{R}}}$$

Fig. 1. Rules of the labeled transition system (LTS) for CCS

2.2 CCSK: A "Keyed" Reversible Concurrent Calculus

CCSK captures uncontrolled reversibility using two symmetric LTS—one for forward computation, one for backward computation—that manipulates *keys* marking executed prefixes, to guarantee that reverting synchronizations cannot be done without both parties agreeing. We use the syntax of the latest paper on the topic [30], that slightly differs [30, Remark 4.2] with the classical definition [37]. However, those changes have no impact since we refrain from using CCSK's newly introduced structural congruence, but discuss it in Sect. 4.

Definition 3 (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, that we denote $\xrightarrow{\alpha[k]}$, is given in Fig. 2—with key and std defined below. The reverse LTS $\overset{\alpha[k]}{\rightsquigarrow}$ is the exact symmetric of $\xrightarrow{\alpha[k]}$ [30, Figure 2] (it can also be read from Fig. 3), and we write $X \overset{\alpha[k]}{\twoheadrightarrow} Y$ 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 4 (Standard and reachable processes). *The set of keys occuring in* X *is written* $\mathrm{key}(X)$*, and* X *is* standard—$\mathrm{std}(X)$—*iff* $\mathrm{key}(X) = \emptyset$*. If there exists a process* O_X *s.t.* $\mathrm{std}(O_X)$ *and* $O_X \twoheadrightarrow^* X$*, then* X *is* reachable*.*

The reader eager to see this system in action can fast-forward to Example 1, but should be aware that this example uses proved labels, introduced next.

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

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

Fig. 2. Rules of the forward labeled transition system (LTS) for CCSK

3 A New Causal Semantics for CCSK

The only causal semantics for CCS with replication we are aware of [19][1] remained unnoticed, despite some interesting qualities: 1. it enables the definition of causality for replication while agreeing with pre-existing causal semantics of CCS and CCS with recursion [19, Theorem 1] 2. it leverages the technique of *proved* transition systems that encodes information about the derivation in the labels [22], 3. it was instrumental in one of the first results connecting implicit computational complexity and distributed processes [23], 4. last but not least, as we will see below, it allows to define an elegant notion of causality for CCSK with "built-in" reversibility, as *the exact same definition will be used for forward and backward transitions*, without making explicit mentions of the keys or directions. We believe our choice is additionally compact, elegant and suited for reversible computation: defining concurrency on composable transitions first allows *not* to consider keys in our definition, as the LTS guarantees that the same key will not be re-used. *Then*, the loop lemma allows to "reverse" transitions so that concurrency on coinitial transitions can be defined from concurrency on composable transitions. This allows to carry little information in the labels—the direction is not needed—and to have a definition insensitive to keys and identifiers for the very modest cost of prefixing labels with some annotation tracking the thread(s) or branch(es) performing the transition.

3.1 Proved Labeled Transition System for CCSK

We adapt the proved transition system [15,19,20] to CCSK: this technique enriches the transitions label with prefixes that describe parts of their derivation, to keep track of their dependencies or lack thereof. We adapt an earlier formalism [21]—including information about sums [19, footnote 2]—but extend the concurrency relation to internal (i.e. τ-) transitions, omitted from recent work [19, Definition 3] but present in older articles [15, Definition 2.3].

Definition 5 (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 $\not{k} : \mathsf{E} \to \mathsf{K}$:

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

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:

[1] We preferred to refer to this work over older presentations [12,13] to be better equipped to later on accommodate replication for reversible calculi [2].

Lemma 1 (Adequacy of the proved labeled transition system). *The transition* $X \xrightarrow{\alpha[m]}\!\!\!\!\!\rightarrow X'$ *can be derived using Fig. 2 iff* $X \xrightarrow{\theta}\!\!\!\!\!\rightarrow X'$ *with* $\not\kappa(\theta) = m$ *and* $\ell(\theta) = \alpha$ *can be derived using Fig. 3.*

Definition 6 (Dependency relation). *The* dependency relation *on enhanced keyed labels is induced by the axioms of Fig. 4, for* $d \in \{\mathrm{L}, \mathrm{R}\}$.

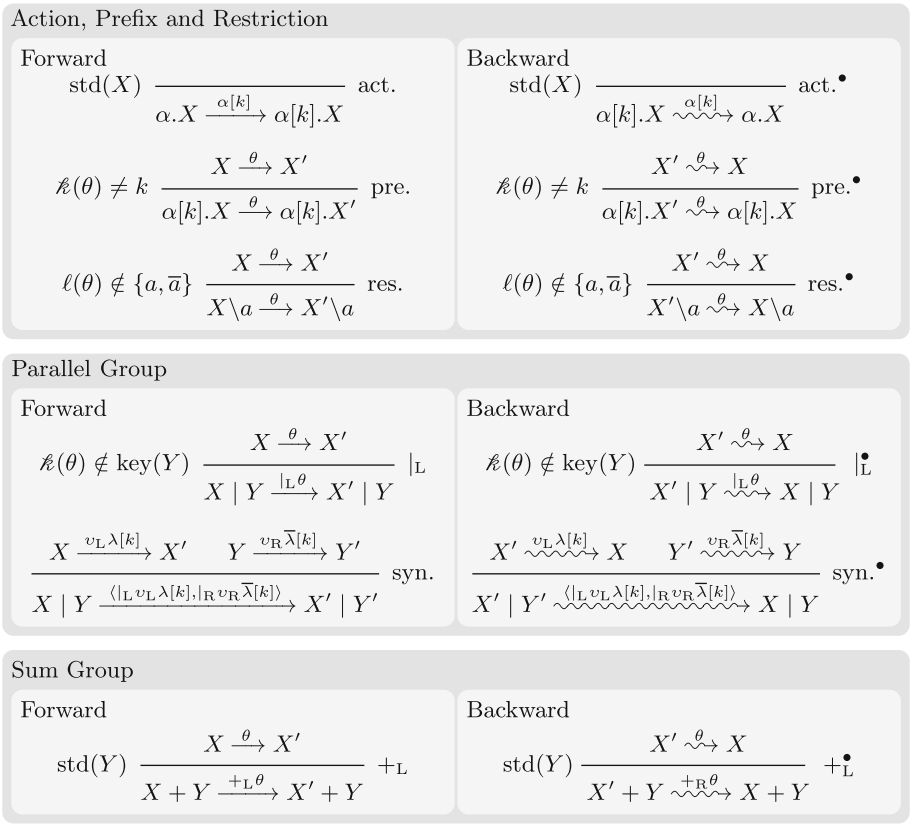


Fig. 3. Rules of the *proved* LTS for CCSK

A dependency $\theta_0 \lessdot \theta_1$ means "whenever there is a trace in which θ_0 occurs before θ_1, then the two associated transitions are causally related". The following definitions will enable more formal examples, but we can stress that 1. the "action" rule enforces that executing or reversing a prefix at top level, e.g.$\alpha.X \xrightarrow{\alpha[k]} \alpha[k].X$ or $\alpha[k].X \overset{\alpha[k]}{\rightsquigarrow} \alpha.X$, makes the prefix ($\alpha[k]$) a dependency of all further transitions; 2. as the forward and backward versions of the same rule share the same enhanced keyed labels, a trace where a transition and its

Action

$$\alpha[k] \ \lessdot \ \theta$$

Sum Group

$$+_d\theta \lessdot +_d\theta' \quad \text{if } \theta \lessdot \theta'$$
$$+_{\mathrm{L}}\theta \lessdot +_{\mathrm{R}}\theta'$$
$$+_{\mathrm{R}}\theta \lessdot +_{\mathrm{L}}\theta'$$

Palallel Group

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

Fig. 4. Dependency Relation on Enhanced Keyed Labels

reverse both occur will have the first occurring be a dependency of the second, as $\lessdot$ is reflexive; 3. no additional relation (such as a conflict or causality relation) is needed to define concurrency; 4. this dependency relation matches the forward-only definition for action and parallel composition, but not for sum: while the original system [19, 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.

Definition 7 (Transitions and traces). *In a* transition $t : X \xrightarrow{\theta}\!\!\!\!\rightarrow X'$, *$X$ 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*, $t_1; t_2$, if the target of t_1 is the source of t_2. The* reverse *of $t : X' \overset{\theta}{\rightsquigarrow} X$ is $t^\bullet : X \xrightarrow{\theta} X'$, and similarly if t is forward, letting $(t^\bullet)^\bullet = t$*[2]*.*

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

Definition 8 (Causality relation). *Let T be a trace $X_1 \xrightarrow{\theta_1}\!\!\!\!\rightarrow \cdots \xrightarrow{\theta_n}\!\!\!\!\rightarrow X_n$ and $i, j \in \{1, \cdots, n\}$ with $i < j$, θ_i* causes *θ_j in T ($\theta_i \lessdot_T \theta_j$) iff $\theta_i \lessdot \theta_j$.*

Definition 9 ((Composable) Concurrency). *Let T be a trace $X_1 \xrightarrow{\theta_1}\!\!\!\!\rightarrow \cdots \xrightarrow{\theta_n}\!\!\!\!\rightarrow X_n$ and $i, j \in \{1, \cdots, n\}$, θ_i* is concurrent with *θ_j ($\theta_i \smile_T \theta_j$, or simply $\theta_i \smile \theta_j$) iff neither $\theta_i \lessdot_T \theta_j$ nor $\theta_j \lessdot_T \theta_i$.*

Coinitial concurrency (Definition 11) will later on be defined using composable concurrency and the loop lemma (Lemma 5).

Example 1. Consider the following trace, dependencies, and concurrent transitions, where the subscripts to $\lessdot$ and $\smile$ have been omitted:

[2] The existence and uniqueness of the reverse transition is immediate in CCSK. This property, known as the loop lemma (Lemma 5) is sometimes harder to obtain.

$(a.\overline{b}) \mid (b + c)$

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

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

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

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

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

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

And we have, e.g.

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

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

and also

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

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

but

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

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

To prove the results in the next section, we need an intuitive and straightforward lemma (Lemma 2) that decomposes a concurrent trace involving two threads into one trace involving one thread while maintaining concurrency, i.e. proving that a trace e.g. of the form $T : X \mid Y \xrightarrow{|_{\mathrm{L}}\theta} X' \mid Y \xrightarrow{|_{\mathrm{L}}\theta'} X'' \mid Y$ with $|_{\mathrm{L}}\ \theta \smile_T |_{\mathrm{L}}\ \theta'$ can be decomposed into a trace $T' : X \xrightarrow{\theta} X' \xrightarrow{\theta'} X''$ with $\theta \smile_{T'} \theta'$. A similar lemma is also needed to decompose sums (Lemma 3), and their proofs proceed by simple case analysis and offer no resistance.

Lemma 2 (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 $\pi_{\mathrm{L}}(X_{\mathrm{L}} \mid X_{\mathrm{R}}) = X_{\mathrm{L}}$, $\pi_{\mathrm{L}}(|_{\mathrm{L}}\ \theta) = \theta$, $\pi_{\mathrm{L}}(\langle |_{\mathrm{L}}\ \theta_{\mathrm{L}}, |_{\mathrm{R}}\ \theta_{\mathrm{R}} \rangle) = \theta_{\mathrm{L}}$, $\pi_{\mathrm{L}}(|_{\mathrm{R}}\ \theta) =$ undefined, and define similarly π_{R}.*

Whenever $T : X_{\mathrm{L}} \mid X_{\mathrm{R}} \xrightarrow{\theta_1} Y_{\mathrm{L}} \mid Y_{\mathrm{R}} \xrightarrow{\theta_2} Z_{\mathrm{L}} \mid Z_{\mathrm{R}}$ with $\theta_1 \smile_T \theta_2$, then for $d \in \{\mathrm{L}, \mathrm{R}\}$, if $\pi_d(\theta_1)$ and $\pi_d(\theta_2)$ are both defined, then, $\pi_d(\theta_1) \smile_{\pi_d(T)} \pi_d(\theta_2)$ with $\pi_d(T) : \pi_d(X_{\mathrm{L}} \mid X_{\mathrm{R}}) \xrightarrow{\pi_d(\theta_1)} \pi_d(Y_{\mathrm{L}} \mid Y_{\mathrm{R}}) \xrightarrow{\pi_d(\theta_2)} \pi_d(Z_{\mathrm{L}} \mid Z_{\mathrm{R}})$.

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

θ_1	θ_2	θ_1	θ_2	θ_1	θ_2	θ_1	θ_2	θ_1	θ_2	θ_1	θ_2	θ_1	θ_2
θ_1	θ_2	$\|_{\mathrm{L}}\ \theta_1'$	$\|_{\mathrm{L}}\ \theta_2'$	$\|_{\mathrm{L}}\ \theta_1'$	$\langle \|_{\mathrm{L}}\ \theta_2', \|_{\mathrm{R}}\ \theta_2'' \rangle$	$\|_{\mathrm{R}}\ \theta_1'$	$\|_{\mathrm{R}}\ \theta_2'$	$\|_{\mathrm{R}}\ \theta_1'$	$\langle \|_{\mathrm{L}}\ \theta_2', \|_{\mathrm{R}}\ \theta_2'' \rangle$	$\langle \|_{\mathrm{L}}\ \theta_1', \|_{\mathrm{R}}\ \theta_1'' \rangle$	$\|_{\mathrm{L}}\ \theta_2'$	$\langle \|_{\mathrm{L}}\ \theta_1', \|_{\mathrm{R}}\ \theta_1'' \rangle$	$\|_{\mathrm{R}}\ \theta_2'$	$\langle \|_{\mathrm{L}}\ \theta_1', \|_{\mathrm{R}}\ \theta_1'' \rangle$	$\langle \|_{\mathrm{L}}\ \theta_2', \|_{\mathrm{R}}\ \theta_2'' \rangle$

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

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

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

a) If $\theta'_1 \lessdot_{\pi_{\mathrm{L}}(T)} \theta'_2$, then $\theta'_1 \lessdot \theta'_2$, and it is immediate that $\theta_1 = |_{\mathrm{L}}\, \theta'_1 \lessdot_T |_{\mathrm{L}}\, \theta'_2 = \theta_2$, contradicting $\theta_1 \smile_T \theta_2$.
b) If $\theta'_1 \lessdot_{\pi_{\mathrm{L}}(T)} \theta'_2$, then $\theta'_1 \lessdot \theta'_2$, $|_{\mathrm{L}}\, \theta'_1 \lessdot |_{\mathrm{L}}\, \theta'_2$ and $\langle |_{\mathrm{L}}\, \theta'_1, |_{\mathrm{R}}\, \theta''_1 \rangle \lessdot |_{\mathrm{L}}\, \theta'_2$, from which we can deduce $\theta_1 \lessdot_T \theta_2$, contradicting $\theta_1 \smile_T \theta_2$.
c) If $\theta'_1 \lessdot_{\pi_{\mathrm{L}}(T)} \theta'_2$, then $\theta'_1 \lessdot \theta'_2$, $|_{\mathrm{L}}\, \theta'_1 \lessdot |_{\mathrm{L}}\, \theta'_2$ 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 $\theta_1 \lessdot_T \theta_2$, contradicting $\theta_1 \smile_T \theta_2$.

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

Lemma 3 (Decomposing concurrent sum transitions). *Let $i \in \{1, 2\}$ and $\theta_i \in \{+_{\mathrm{L}}\theta'_i, +_{\mathrm{R}}\theta''_i\}$, define $\rho_{\mathrm{L}}(X_{\mathrm{L}} + X_{\mathrm{R}}) = X_{\mathrm{L}}$, $\rho_{\mathrm{L}}(+_{\mathrm{L}}\theta) = \theta$, $\rho_{\mathrm{L}}(+_{\mathrm{R}}\theta) =$ undefined, and define similarly ρ_{R}.*

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 $\theta_1 \smile_T \theta_2$, then for $d \in \{\mathrm{L}, \mathrm{R}\}$, if $\rho_d(\theta_1)$ and $\rho_d(\theta_2)$ are both defined, then, $\rho_d(\theta_1) \smile_{\pi_d(T)} \rho_d(\theta_2)$ with $\rho_d(T) : \rho_d(X_{\mathrm{L}} + X_{\mathrm{R}}) \xrightarrow{\rho(\theta_1)}\!\!\!\!\!\rightarrow \rho_d(Y_{\mathrm{L}} + Y_{\mathrm{R}}) \xrightarrow{(\theta_2)}\!\!\!\!\!\rightarrow \rho_d(Z_{\mathrm{L}} + Z_{\mathrm{R}})$.

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

$$\begin{array}{c|cc} \theta_1 & +_{\mathrm{L}}\theta'_1 & +_{\mathrm{R}}\theta'_1 \\ \hline \theta_2 & +_{\mathrm{L}}\theta'_2 & +_{\mathrm{R}}\theta'_2 \end{array}$$

In each case, assume $\rho_{\mathrm{L}}(\theta_1) = \theta'_1 \smile_{\rho_{\mathrm{L}}(T)} \theta'_2 = \rho_{\mathrm{L}}(\theta_2)$ does not hold, then it is immediate to note that $\theta_1 \smile_T \theta_2$ cannot hold either, a contradiction. □

3.2 Diamonds and Squares: Concurrency in Action

Square properties and concurrency diamonds express that concurrent transitions are *actually* independent, in the sense that they can be swapped if they are composable, or "later on" agree if they are co-initial. That our definition of concurrency enables those, *and* to allows inter-prove them, is a good indication that it is resilient and convenient.

Theorem 1 (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. $X \xrightarrow{\theta_2} X_2 \xrightarrow{\theta_1} Y$.*

The proof, sketched, requires a particular care when X is not standard. Using pre. is transparent from the perspective of enhanced keyed labels, as no "memory" of its usage is stored in the label of the transition. This is essentially because—exactly like for act.—all the dependency information is already present in the term or its enhanced keyed label. To make this more formal, we introduce a function that "removes" a keyed label, and prove that it does not affect derivability.

Definition 10. *Given α and k, we define* $\mathrm{rm}_{\alpha[k]}$ *by* $\mathrm{rm}_{\alpha[k]}(0) = 0$ *and*

$$\mathrm{rm}_{\alpha[k]}(\beta.X) = \beta.X \qquad \mathrm{rm}_{\alpha[k]}(X \mid Y) = \mathrm{rm}_{\alpha[k]}(X) \mid \mathrm{rm}_{\alpha[k]}(Y)$$
$$\mathrm{rm}_{\alpha[k]}(X\backslash a) = (\mathrm{rm}_{\alpha[k]}X)\backslash a \qquad \mathrm{rm}_{\alpha[k]}(X + Y) = \mathrm{rm}_{\alpha[k]}(X) + \mathrm{rm}_{\alpha[k]}(Y)$$
$$\mathrm{rm}_{\alpha[k]}(\beta[m].X) = \begin{cases} X & \text{if } \alpha = \beta \text{ and } k = m \\ \beta[m].\mathrm{rm}_{\alpha[k]}(X) & \text{otherwise} \end{cases}$$

We let $\mathrm{rm}_k^\lambda = \mathrm{rm}_{\lambda[k]} \circ \mathrm{rm}_{\overline{\lambda}[k]}$ *if* $\lambda \in \mathsf{L}\backslash\{\tau\}$, $\mathrm{rm}_k^\tau = \mathrm{rm}_{\tau[k]}$ *otherwise.*

The function $\mathrm{rm}_{\alpha[k]}$ simply looks for an occurrence of $\alpha[k]$ and removes it: as there is at most one, 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 4. *For all X, α and k, $X \xrightarrow{\theta}\!\!\!\!\rightarrow Y$ with $\not\kappa(\theta) \neq k$ iff* $\mathrm{rm}_k^\alpha(X) \xrightarrow{\theta}\!\!\!\!\rightarrow \mathrm{rm}_k^\alpha(Y)$.

Proof. Assume $\alpha[k]$ or $\overline{\alpha}[k]$ (if $\alpha \neq \tau$) occur in X (otherwise the result is straightforward), as $\not\kappa(\theta) \neq k$, the same holds for Y. As keys occur at most twice, attached to complementary names, in reachable processes [30, Lemma 3.4], $k \notin \mathrm{key}(\mathrm{rm}_k^\alpha(X)) \cup \mathrm{key}(\mathrm{rm}_k^\alpha(Y))$. Then the proof follows by induction on the length of the derivation for $X \xrightarrow{\theta}\!\!\!\!\rightarrow Y$: as neither pre. nor pre.$^\bullet$ change the enhanced keyed label, we can simply "take out" the occurrences of those rules when they concern $\alpha[k]$ or $\overline{\alpha}[k]$ and still obtain a valid derivation, with the same enhanced keyed label, hence yielding $\mathrm{rm}_k^\alpha(X) \xrightarrow{\theta}\!\!\!\!\rightarrow \mathrm{rm}_k^\alpha(Y)$. For the converse direction, pre. or pre.$^\bullet$ can be reintroduced to the derivation tree and in the appropriate location, as k is fresh in $\mathrm{rm}_k^\alpha(X)$ and $\mathrm{rm}_k^\alpha(Y)$. □

Proof (of Theorem 1 (sketch)). The proof proceeds by induction on the length of the deduction for the derivation for $X \xrightarrow{\theta_1} X_1$, using Lemmas 2 and 3 to enable the induction hypothesis if θ_1 is not a prefix. The only delicate case is if the last rule is pre.: in this case, 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 $\alpha[k].X_1' \xrightarrow{\theta_2} Y$, $\not\kappa(\theta_2) \neq k$ [30, Lemma 3.4], and since $\theta_1 \smile \theta_2$, we apply Lemma 4 twice to obtain the trace T:

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

with $\theta_1 \smile_T \theta_2$, and we can use the induction hypothesis to obtain 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 4 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. □

Example 2. Re-using Example 1, since $|_\mathrm{L}\, \overline{b}[n] \smile |_\mathrm{R} +_\mathrm{R} c[n']$ in

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

Theorem 1 allows to re-arrange this trace as

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

Theorem 2 (Reverse diamonds).

1. *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.* $X \overset{\theta_2}{\rightsquigarrow} X_2 \xrightarrow{\theta_1} Y$.
2. *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.* $X \xrightarrow{\theta_2} X_2 \overset{\theta_1}{\rightsquigarrow} Y$.

It should be noted that in the particular case of $t; t^\bullet : X \xrightarrow{\theta_1} X_1 \overset{\theta_1}{\rightsquigarrow} X$, or $t^\bullet; t$, $\theta_1 \lessdot \theta_1$ by reflexivity of $\lessdot$ and hence the reverse diamonds cannot apply. The name "reverse diamond" was sometimes used for different properties [37, Proposition 5.10], [36, Definition 2.3] that, in the presence of the loop lemma (Lemma 5), are equivalent to ours, once the condition on keys is replaced by our condition on concurrency. It is, however, to our knowledge the first time this property, stated in this particular way, is isolated and studied on its own.

Proof (Sketch). We can re-use the proof of Theorem 1 almost as it is, since Lemmas 4, 2 and 3 hold for both directions.

For 1., the only case that diverges 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 $\not{k}(\theta_2) \neq k$ immediately. However, if $\not{k}(\theta_2) = k$, then we would have $\alpha[k].X_1' \overset{\alpha[k]}{\rightsquigarrow} \alpha.Y' = Y$, but this application of act.$^\bullet$ is not valid, as std(X_1') does not hold, since X_1' was obtained from X' after it made a *forward* transition. Hence, we obtain that key$(\theta_2) \neq k$ and we can carry out the rest of the proof as before.

For 2., the main difference lies in leveraging the dependency of sum prefixes between e.g. $+_{\mathrm{R}}\theta_1$ and $+_{\mathrm{L}}\theta_2$ in $X + O_Y \overset{+_{\mathrm{R}}\theta_1}{\rightsquigarrow} O_X + O_Y \xrightarrow{+_{\mathrm{L}}\theta_2} O_X + Y$. □

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

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

Theorem 2 allows to re-arrange this trace as

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

Concurrency on coinitial traces is defined using concurrency on composable traces and the loop lemma, immediate in CCSK.

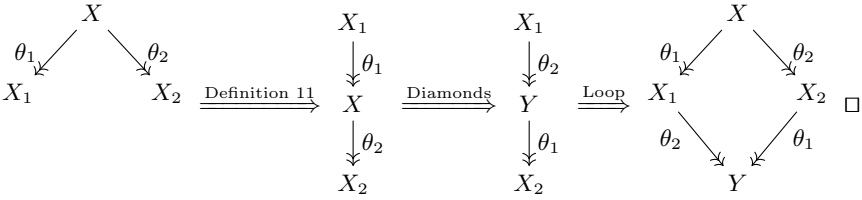
Lemma 5 (Loop lemma [37, Prop. 5.1]). *For all* $t : X \xrightarrow{\theta} X'$, *there exists a unique* $t^\bullet : X' \overset{\theta}{\rightsquigarrow} X$, *and conversely. We let* $(t^\bullet)^\bullet = t$.

Definition 11 (Coinitial concurrency). *Let* $t_1 : X \xrightarrow{\theta_1} Y_1$ *and* $t_2 : X \xrightarrow{\theta_2} Y_2$ *be two coinitial transitions,* θ_1 *is concurrent with* θ_2 *(*$\theta_1 \smile \theta_2$*) iff* $\theta_1 \smile \theta_2$ *in the trace* $t_1^\bullet; t_2 : Y_1 \xrightarrow{\theta_1} X \xrightarrow{\theta_2} Y_2$.

To our knowledge, this is the first time co-initial concurrency is defined from composable concurrency: while the axiomatic approach discussed coinitial concurrency [31, Section 5], it primarily studied independence relations that could be defined in any way, and did not connect these two notions of concurrency.

Theorem 3 (Square property). *For all* $t_1 : X \xrightarrow{\theta_1\!\!\rightarrow} X_1$ *and* $t_2 : X \xrightarrow{\theta_2\!\!\rightarrow} X_2$ *with* $\theta_1 \smile \theta_2$*, there exist* $t_1' : X_1 \xrightarrow{\theta_2\!\!\rightarrow} Y$ *and* $t_2' : X_2 \xrightarrow{\theta_1\!\!\rightarrow} Y$*.*

Proof (sketch). By Definition 11 we have that $\theta_1 \smile \theta_2$ in $t_1^\bullet; t_2 : X_1 \xrightarrow{\theta_1\!\!\rightarrow} X \xrightarrow{\theta_2\!\!\rightarrow} X_2$. Hence, depending on the direction of the arrows, and possibly using the loop lemma to convert two backward transitions into two forward ones, we obtain by Theorems 1 or 2 $t_1''; t_2'' : X_1 \xrightarrow{\theta_2\!\!\rightarrow} Y \xrightarrow{\theta_1\!\!\rightarrow} X_2$, and we let $t_1' = t_1''$ and $t_2' = t''^\bullet_2$:



Example 4. Following Example 1, we can get e.g. from $a[m].\bar{b}[n] \mid b + c \xrightarrow{|_R + _L b[n']}$ $a[m].\bar{b}[n] \mid b[n'] + c$ and $a[m].\bar{b}[n] \mid b + c \overset{|_L \bar{b}[n]}{\rightsquigarrow} a[m].\bar{b} \mid b + c$ the transitions converging to $a[m].\bar{b} \mid b[n'] + c$.

3.3 Causal Consistency

Formally, causal consistency (Theorem 4) states that any two coinitial and cofinal traces are causally equivalent:

Definition 12 (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 4. *All coinitial and cofinal traces are causally equivalent.*

The "axiomatic approach" to reversible computation [31] allows to obtain causal consistency from other properties that are generally easier to prove.

Lemma 6 (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.*

Proof (Sketch). The proof is by induction on the size of θ_1 and leverages that $\not\kappa(\theta_1) \neq \not\kappa(\theta_2)$ for both transitions to be different. □

Lemma 7 (Well-foundedness). *For all* X *there exists* $n \in \mathbb{N}$*,* $X_0, \cdots, X_n$ *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 4). We can re-use the results of the axiomatic approach [31] 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 [39, Definition 3.7]) is indeed an irreflexive symmetric relation: symmetry is immediate by definition, irreflexivity follows from the fact that $\lessdot$ is reflexive. Then, by Theorem 3 and Lemma 6, the parabolic lemma holds [31, Proposition 3.4], and since the parabolic lemma and well-foundedness hold (Lemma 7), causal consistency holds as well [31, Proposition 3.5]. □

We use here the axiomatic approach [31] in a very narrow sense, to obtain only causal consistency—which was our main goal—, but we could have used those lemmas to obtain many other desirable properties for this system "for free". An interesting problem, as suggested by a reviewer, would also be to establish whenever our system enjoys Coinitial Propagation of Independence [31, Definition 4.2], which in turns would allow it to fulfil Independence of Diamonds [31, Definition 4.6].

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

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

4 Structural Congruence, Universality and Other Criteria

Causality for a semantics of concurrent computation should satisfy a variety of criteria, the squares and diamonds being the starting point, and causal consistency being arguably the most important. This section aims at briefly presenting additional criteria and at defending the "universality" of our approach. Since this last point requires to introduce two other reversible systems and four other definitions of concurrency, the technical content is only in our research report [3, Sect. C], but we would like to stress that the results stated below are fairly routine to prove—introducing all the material to enable the comparisons is the only lengthy part.

Concurrency-Preserving Structural Congruences. "Denotationality" [17, 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 [15,20,22,23], or missing the associativity and commutativity of the parallel composition [21, p. 242]. While adding such a congruence would benefits the expressiveness, making it interact

nicely with the derived proof system *and* the reversible features [30, Section 4], [7] 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 a reversible π-calculus extending CCSK [32], back to a sum-free version of CCSK. The structural causality [32, Definition 22]—for transitions of the same direction—and conflict relation [32, Definition 25]—for transitions of opposite directions—can then both be proven to match our dependency relation in a rather straightforward way, hence proving the adequacy of notions. However, this inherited concurrency relation cannot be straightforwardly extended to the sum operator, and requires two relations to be defined: for those reasons, we argue that our solution is more convenient to use. It should also be noted that this concurrency does not meet the denotationality criteria either, when the congruence includes renaming of bound keys [30].

A similar work could have been done by restricting concurrency for e.g. reversible higher-order π-calculus [29, Definition 9], reversible π-calculus [16, Definition 4.1] or croll-π [27, Definition 1], but we reserve it for future work, and would prefer to extend our definition to a reversible π-calculus rather than proceeding the other way around.

Comparing with RCCS*-Inspired Systems.* In RCCS, the definition of concurrency fluctuated between a condition on memory inclusion for composable transitions [25, Definition 3.1.1] and a condition on disjointness of memories on coinitial transitions [18, Definition 7], both requiring the entire memory of the thread to label the transitions, and neither been defined on transitions of opposite directions. It is possible to adapt our proved system to RCCS, and to prove that the resulting concurrency relation is equivalent to those two definitions, when restricted to transitions of equal direction. A similar adaptation is possible for reversible and identified CCS [8], that came with *yet* another definition of concurrency leveraging its built-in mechanism to generate identifiers.

Optimality, Parabolic Lemma, and RPI. The optimality criteria is the adequacy of the concurrency definitions for the LTS and for the reduction semantics [16, 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 [18, Lemma 10]—"any trace is equivalent to a backward trace followed by a forward trace"—or "RPI" [31, 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 believe 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.

5 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 [6, p. 13], or explicitly generated by a built-in mechanism [8, 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 [4–6] and CCSK [24,36,40]. 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 [23], 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 [11], including recursion, seems to have been unsuccessful [41]. A possible approach would be to define recursion and iteration in CCSK, to extend our definition of concurrency to those infinite behaviors, and to attempt to reconstruct the separation results from the forward-only paradigm [35]. Whether finer, "reversible", 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 [35], and additional criteria were introduced to study causal semantics [17]. 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 [9,10].

Acknowledgments. I would like to thank Doriana Medić for suggesting that I adapt the definition of concurrency for a reversible π-calculus extending CCSK [32] and compare it to the concurrency developed in this paper, as done in [3, Sect. C]. I am also extremely thankful to the reviewers for their careful reading of this technical paper, and for their enlightening suggestions.

References

1. Arpit, Kumar, D.: Calculus of concurrent probabilistic reversible processes. In: ICCCT-2017: Proceedings of the 7th International Conference on Computer and Communication Technology, pp. 34–40. ICCCT-2017. ACM, New York (2017). https://doi.org/10.1145/3154979.3155004
2. Aubert, C.: Causal consistent replication in reversible concurrent calculi, October 2021. https://hal.archives-ouvertes.fr/hal-03384482. Under revision
3. Aubert, C.: Concurrencies in reversible concurrent calculi. Technical report, March 2022. https://hal.archives-ouvertes.fr/hal-03605003
4. Aubert, C., Cristescu, I.: Reversible barbed congruence on configuration structures. In: Knight, S., Lluch Lafuente, A., Lanese, I., Vieira, H.T. (eds.) ICE 2015. EPTCS, vol. 189, pp. 68–95 (2015). https://doi.org/10.4204/EPTCS.189.7
5. Aubert, C., Cristescu, I.: Contextual equivalences in configuration structures and reversibility. J. Log. Algebr. Methods Program. **86**(1), 77–106 (2017). https://doi.org/10.1016/j.jlamp.2016.08.004
6. Aubert, C., Cristescu, I.: How reversibility can solve traditional questions: the example of hereditary history-preserving bisimulation. In: Konnov, I., Kovács, L. (eds.) 31st International Conference on Concurrency Theory, CONCUR 2020, 1–4 September 2020, Vienna, Austria. LIPIcs, vol. 2017, pp. 13:1–13:24. Schloss Dagstuhl (2020). https://doi.org/10.4230/LIPIcs.CONCUR.2020.13
7. Aubert, C., Cristescu, I.: Structural equivalences for reversible calculi of communicating systems (oral communication). Research report, Augusta University (2020). https://hal.archives-ouvertes.fr/hal-02571597. Communication at ICE 2020
8. Aubert, C., Medić, D.: Explicit identifiers and contexts in reversible concurrent calculus. In: Yamashita, S., Yokoyama, T. (eds.) RC 2021. LNCS, vol. 12805, pp. 144–162. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-79837-6_9
9. Aubert, C., Varacca, D.: Processes, systems & tests: defining contextual equivalences. In: Lange, J., Mavridou, A., Safina, L., Scalas, A. (eds.) Proceedings 14th Interaction and Concurrency Experience, Online, 18th June 2021. EPTCS, vol. 347, pp. 1–21. Open Publishing Association (2021). https://doi.org/10.4204/EPTCS.347.1
10. Aubert, C., Varacca, D.: Processes against tests: Defining contextual equivalences. Invited submission to the Journal of Logical and Algebraic Methods in Programming (2022). https://hal.archives-ouvertes.fr/hal-03535565
11. Baeten, J.C.M.: A brief history of process algebra. Theor. Comput. Sci. **335**(2–3), 131–146 (2005). https://doi.org/10.1016/j.tcs.2004.07.036
12. Boudol, G., Castellani, I.: A non-interleaving semantics for CCS based on proved transitions. Fund. Inform. **11**, 433–452 (1988)
13. Boudol, G., Castellani, I.: Three equivalent semantics for CCS. In: Guessarian, I. (ed.) LITP 1990. LNCS, vol. 469, pp. 96–141. Springer, Heidelberg (1990). https://doi.org/10.1007/3-540-53479-2_5
14. Busi, N., Gabbrielli, M., Zavattaro, G.: On the expressive power of recursion, replication and iteration in process calculi. MSCS **19**(6), 1191–1222 (2009). https://doi.org/10.1017/S096012950999017X

15. Carabetta, G., Degano, P., Gadducci, F.: CCS semantics via proved transition systems and rewriting logic. In: Kirchner, C., Kirchner, H. (eds.) 1998 International Workshop on Rewriting Logic and its Applications, WRLA 1998, Abbaye des Prémontrés at Pont-à-Mousson, France, September 1998. Electron. Notes Theor. Comput. Sci. **15**, 369–387 (1998). https://doi.org/10.1016/S1571-0661(05)80023-4. https://www.sciencedirect.com/journal/electronic-notes-in-theoretical-computer-science/vol/15/suppl/C
16. Cristescu, I., Krivine, J., Varacca, D.: A compositional semantics for the reversible p-calculus. In: LICS, pp. 388–397. IEEE Computer Society (2013). https://doi.org/10.1109/LICS.2013.45
17. Cristescu, I.D., Krivine, J., Varacca, D.: Rigid families for CCS and the π-calculus. In: Leucker, M., Rueda, C., Valencia, F.D. (eds.) ICTAC 2015. LNCS, vol. 9399, pp. 223–240. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-25150-9_14
18. Danos, V., Krivine, J.: Reversible communicating systems. In: Gardner, P., Yoshida, N. (eds.) CONCUR 2004. LNCS, vol. 3170, pp. 292–307. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-28644-8_19
19. Degano, P., Gadducci, F., Priami, C.: Causality and replication in concurrent processes. In: Broy, M., Zamulin, A.V. (eds.) PSI 2003. LNCS, vol. 2890, pp. 307–318. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-39866-0_30
20. Degano, P., Priami, C.: Proved trees. In: Kuich, W. (ed.) ICALP 1992. LNCS, vol. 623, pp. 629–640. Springer, Heidelberg (1992). https://doi.org/10.1007/3-540-55719-9_110
21. Degano, P., Priami, C.: Non-interleaving semantics for mobile processes. Theor. Comput. Sci. **216**(1–2), 237–270 (1999). https://doi.org/10.1016/S0304-3975(99)80003-6
22. Degano, P., Priami, C.: Enhanced operational semantics. ACM Comput. Surv. **33**(2), 135–176 (2001). https://doi.org/10.1145/384192.384194
23. Demangeon, R., Yoshida, N.: Causal computational complexity of distributed processes. In: Dawar, A., Grädel, E. (eds.) LICS, pp. 344–353. ACM (2018). https://doi.org/10.1145/3209108.3209122
24. Graversen, E., Phillips, I.C.C., Yoshida, N.: Event structure semantics of (controlled) reversible CCS. J. Log. Algebr. Methods Program. **121**, 100686 (2021). https://doi.org/10.1016/j.jlamp.2021.100686
25. Krivine, J.: Algèbres de Processus Réversible - Programmation Concurrente Déclarative. Ph.D. thesis, Université Paris 6 & INRIA Rocquencourt (2006). https://tel.archives-ouvertes.fr/tel-00519528
26. Lanese, I.: From reversible semantics to reversible debugging. In: Kari, J., Ulidowski, I. (eds.) RC 2018. LNCS, vol. 11106, pp. 34–46. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-99498-7_2
27. Lanese, I., Lienhardt, M., Mezzina, C.A., Schmitt, A., Stefani, J.-B.: Concurrent flexible reversibility. In: Felleisen, M., Gardner, P. (eds.) ESOP 2013. LNCS, vol. 7792, pp. 370–390. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-37036-6_21
28. Lanese, I., Medić, D., Mezzina, C.A.: Static versus dynamic reversibility in CCS. Acta Inform. (2019). https://doi.org/10.1007/s00236-019-00346-6
29. Lanese, I., Mezzina, C.A., Stefani, J.: Reversibility in the higher-order π-calculus. Theor. Comput. Sci. **625**, 25–84 (2016). https://doi.org/10.1016/j.tcs.2016.02.019
30. Lanese, I., Phillips, I.: Forward-reverse observational equivalences in CCSK. In: Yamashita, S., Yokoyama, T. (eds.) RC 2021. LNCS, vol. 12805, pp. 126–143. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-79837-6_8

31. Lanese, I., Phillips, I., Ulidowski, I.: An axiomatic approach to reversible computation. In: FoSSaCS 2020. LNCS, vol. 12077, pp. 442–461. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45231-5_23
32. Medić, D., Mezzina, C.A., Phillips, I., Yoshida, N.: A parametric framework for reversible π-calculi. Inf. Comput. **275**, 104644 (2020). https://doi.org/10.1016/j.ic.2020.104644
33. Mezzina, C.A., Koutavas, V.: A safety and liveness theory for total reversibility. In: Mallet, F., Zhang, M., Madelaine, E. (eds.) 11th International Symposium on Theoretical Aspects of Software Engineering, TASE 2017, Sophia Antipolis, France, 13–15 September, pp. 1–8. IEEE (2017). https://doi.org/10.1109/TASE.2017.8285635. https://ieeexplore.ieee.org/xpl/conhome/8277122/proceeding
34. Milner, R. (ed.): A Calculus of Communicating Systems. LNCS, vol. 92. Springer, Heidelberg (1980). https://doi.org/10.1007/3-540-10235-3
35. Palamidessi, C., Valencia, F.D.: Recursion vs replication in process calculi: expressiveness. Bull. EATCS **87**, 105–125 (2005). http://eatcs.org/images/bulletin/beatcs87.pdf
36. Phillips, I., Ulidowski, I.: Reversibility and models for concurrency. Electron. Notes Theor. Comput. Sci. **192**(1), 93–108 (2007). https://doi.org/10.1016/j.entcs.2007.08.018
37. Phillips, I., Ulidowski, I.: Reversing algebraic process calculi. J. Log. Algebr. Program. **73**(1–2), 70–96 (2007). https://doi.org/10.1016/j.jlap.2006.11.002
38. Sangiorgi, D., Walker, D.: The Pi-calculus. CUP (2001)
39. Sassone, V., Nielsen, M., Winskel, G.: Models for concurrency: towards a classification. Theor. Comput. Sci. **170**(1–2), 297–348 (1996). https://doi.org/10.1016/S0304-3975(96)80710-9
40. Ulidowski, I., Phillips, I., Yuen, S.: Concurrency and reversibility. In: Yamashita, S., Minato, S. (eds.) RC 2014. LNCS, vol. 8507, pp. 1–14. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-08494-7_1
41. Wang, Y.: RETRACTED ARTICLE: an algebra of reversible computation. SpringerPlus **5**(1), 1–35 (2016). https://doi.org/10.1186/s40064-016-3229-7