Concurrencies in Reversible Concurrent Calculi

Clément Aubert

Concurrencies in Reversible Concurrent Calculi

Clément Aubert[0000−0001−6346−3043]

School of Computer & 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 [33] contradicted Milner's expansion laws [26, Example 4.11], and the study of causal models for reversible computation led to novel correction criteria for causal semantics—both reversible and irreversible [13]. "Traditional" equivalence relations have been captured syntactically [4], while original observational equivalences were developed [26]: reversibility triggered a global reconsideration of established theories and tools, with the clear intent of providing actionable methods for reversible systems [22], novel axiomatic foundations [27] and original non-interleaving models [2, 13, 20].

Two Systems extend the Calculus of Communicating Systems (CCS) [30]—the godfather of π-calculus [34], among others—with reversible features. Reversible CCS (RCCS) [14] and CCS with keys (CCSK) [33] are similarly the source of most [1, 13, 28, 29]—if not all—of later systems 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 [24], 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 [26, Proposition 4.9] or the impossibility thereof for RCCS [6, Theorem 2]. However, they also share some shortcomings, and we offer to tackle one of them for CCSK: a syntactical definition of concurrency, preferably easy to manipulate and satisfying 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 [33]. As such, it have 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 [33, Propositions 5.10 and 5.19], but to our knowledge no "definitive" definition of concurrency was proposed. Ad-hoc definitions relying on memory inclusion [21, Definition 3.1.1] or disjointness [14, Definition 7] for RCCS, and semantical notions for both RCCS [2–4] and CCSK [20,32,36] 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 [15], that leveraged *proved* transition systems [18]. We make crucial use of the loop lemma (Lemma 3) 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 [14,21] and its "identified" declensions [6], 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.

Small technical lemmas, explained in the paper, are in Sect. A, and all proofs are in Sect. B, with their main arguments sometimes in the paper. Sect. C justifies the claims made in Sect. 4 about the "universality" of our approach.

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 [10].

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). CCS *processes range over P, Q and are 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, from highest to lowest, is $\backslash\alpha$, $\alpha.$, $+$ and $\mid$, 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 [26], that slightly differs [26, Remark 4.2] with the classical definition [33]. However, those changes have no impact since we refrain from using CCSK's newly introduced structural congruence, but discussed 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

$\xrightarrow{\alpha[k]}$ [26, 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 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, p. 7, 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 [15] 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 [15, Theorem 1] 2. it leverages the technique of *proved* transition systems that encodes information about the derivation in the labels [18], 3. it was instrumental in one of the first result connecting implicit

computational complexity and distributed processes [19], 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 insensible 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 [11, 15, 16] 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 [17]—including information about sums [15, footnote 2]—but extend the concurency relation to internal (i.e. τ-) transitions, omitted from recent work [15, Definition 3] but present in older articles [11, 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 have its derivation in bijection with CCSK's original LTS:

Lemma 1 (Adequation 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 $\not{k}(\theta) = m$ 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 $d \in \{\mathrm{L}, \mathrm{R}\}$.*

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

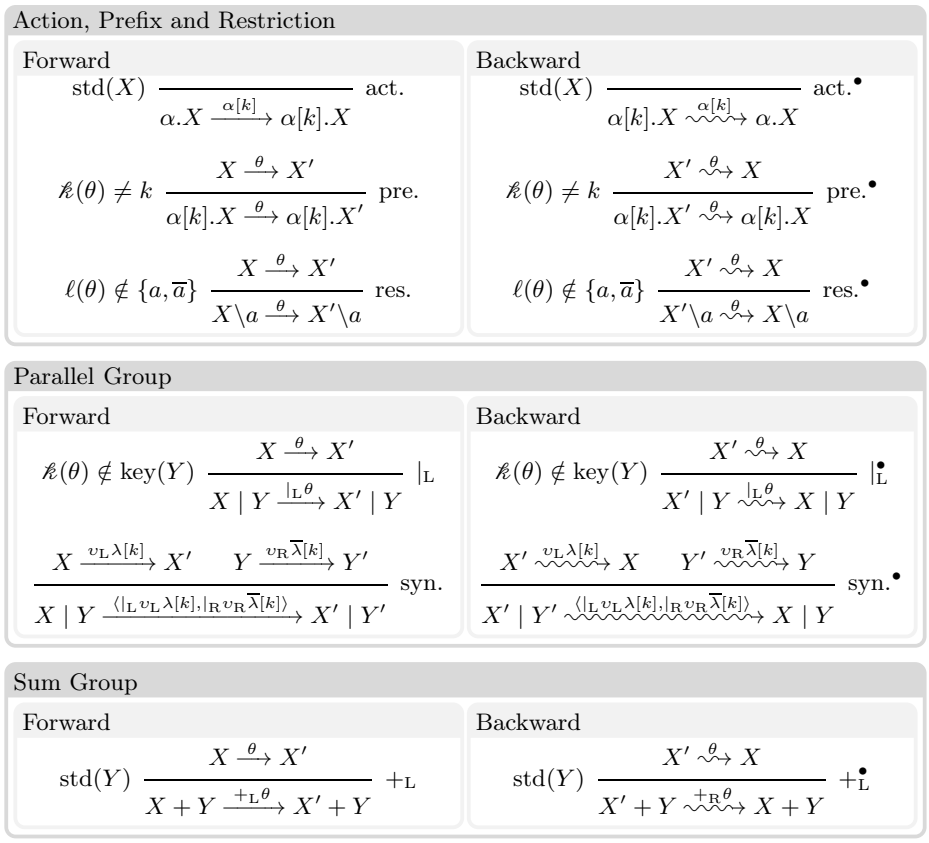


Fig. 3. Rules of the *proved* LTS for CCSK

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 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 [15, 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*,*

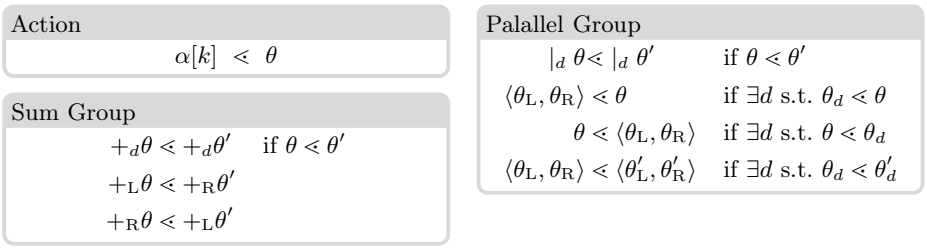


Fig. 4. Dependency Relation on Enhanced Keyed Labels

$t_1;t_2$, *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* $(t^\bullet)^\bullet = t$.[1]

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* $X_1 \xrightarrow{\theta_1} \cdots \xrightarrow{\theta_n} X_n$ *and* $i,j \in \{1,\ldots,n\}$ *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* $X_1 \xrightarrow{\theta_1} \cdots \xrightarrow{\theta_n} X_n$ *and* $i,j \in \{1,\ldots,n\}$, 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 3).

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

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

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.

[1] The existence and uniqueness of the reverse transition is immediate in CCSK. This property, known as the loop lemma (Lemma 3) is sometimes harder to obtain.

To prove the results in the next section, we need an intuitive and straightforward lemma (Lemma 6) 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 7), and their statements and proofs are in Sect. A: they both proceed by simple case analysis and offer no resistance.

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 p. 9, 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 $\mathrm{rm}^{\lambda}_{k} = \mathrm{rm}_{\lambda[k]} \circ \mathrm{rm}_{\overline{\lambda}[k]}$ *if* $\lambda \in \mathsf{L}\backslash\{\tau\}$, $\mathrm{rm}^{\tau}_{k} = \mathrm{rm}_{\tau[k]}$ *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 2. *For all* X, α *and* k, $X \xrightarrow{\theta} Y$ *with* $\not\kappa(\theta) \neq k$ *iff* $\mathrm{rm}^{\alpha}_{k}(X) \xrightarrow{\theta} \mathrm{rm}^{\alpha}_{k}(Y)$.

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 [26, 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 6 and 7 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 $\alpha[k].X'_1 \xrightarrow{\theta_2} Y$, $\not\kappa(\theta_2) \neq k$ [26, Lemma 3.4], and since $\theta_1 \smile \theta_2$, we apply Lemma 2 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 2 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

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

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 [33, Proposition 5.10; 32, Definition 2.3] that, in the presence of the loop lemma (Lemma 3), 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 2, 6 and 7 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 $\not\kappa(\theta_2) \neq k$ 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.$^\bullet$ 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 $\mathrm{key}(\theta_2) \neq k$ 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 $|_{\mathrm{R}} +_{\mathrm{R}} c[n'] \smile |_{\mathrm{L}} \overline{b}[n]$ in

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

Theorem 2 allows to re-arrange this trace as

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

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

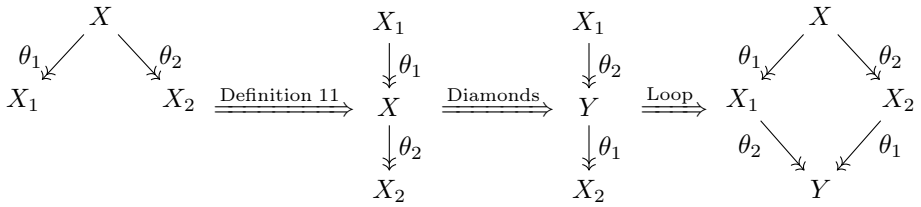
Lemma 3 (Loop lemma [33, Prop. 5.1]). *For all $t : X \xrightarrow{\theta} X'$, there exists a unique $t^\bullet : X' \overset{\theta}{\rightsquigarrow} X$, and conversely. We let $(t^\bullet)^\bullet = t$.*

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 on coinitial concurrency [27, Section 5], it primarily studied independence relations that could be defined in any way, and did not connect those two notions of concurrencies.

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

Proof (sketch). By Definition 11 we have that $\theta_1 \smile \theta_2$ in $t_1^\bullet; t_2 : X_1 \xrightarrow{\theta_1} X \xrightarrow{\theta_2} X_2$. 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 $t_1''; t_2'' : X_1 \xrightarrow{\theta_2} Y \xrightarrow{\theta_1} X_2$, and we let $t_1' = t_1''$ and $t_2' = t''^\bullet_2$:



□

Example 4. Following Example 1, we can get e.g. from $a[m].\overline{b}[n] \mid b+c \xrightarrow{|_{\mathrm{R}}+_{\mathrm{L}}b[n']} a[m].\overline{b}[n] \mid b[n'] + c$ 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 $a[m].\overline{b} \mid b[n'] + c$.

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 [27] allows to obtain causal consistency from other properties that are generally easier to prove.

Lemma 4 (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 $\not{k}(\theta_1) \neq \not{k}(\theta_2)$ for both transitions to be different. □

Lemma 5 (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 4). We can re-use the results of the axiomatic approach [27] 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 [35, 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 4, the parabolic lemma holds [27, Proposition 3.4], and since the parabolic lemma and well-foundedness hold (Lemma 5), causal consistency holds as well [27, Proposition 3.5]. □

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 critera, 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 placed in 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" [13, 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 [11, 16, 18, 19], or missing the associativity and commutativity of the parallel composition [17, p. 242]. While adding such a congruence would benefits the expressiveness, making it interact nicely with the derived proof system *and* the reversible features [26, Section 4; 5] 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 [28], back to a sum-free version of CCSK. The structural causality [28, Definition 22]—for transitions of the same direction—and conflict relation [28, 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 adequation 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 [26].

A similar work could have been done by restricting concurrency for e.g. reversible higher-order π-calculus [25, Definition 9], reversible π-calculus [12, Definition 4.1] or croll-π [23, 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 [21, Definition 3.1.1] and a condition on disjointness of memories on coinitial transitions [14, 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 [6], 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 adequation of the concurrency definitions for the LTS and for the reduction semantics [12, 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 [14, Lemma 10]—"any trace is equivalent to a backward trace followed by a forward trace"—or "RPI" [27, 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 [4, p. 13], or explicitly generated by a built-in mechanism [6, 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 [2–4] and CCSK [20, 20, 32, 36]. 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 [19], 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: some attempts to extend algebras of communicating processes [9], including recursion, seems to have been unsuccessful [37]. 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 [31]. 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, this 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 [31], and additional criteria were introduced to study causal semantics [13]. 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 [7, 8].

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. p. 34–40. ICCCT-2017, ACM, New York, NY, USA (2017). https://doi.org/10.1145/3154979.3155004
2. 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
3. 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
4. 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, September 1–4, 2020, Vienna, Austria. LIPIcs, vol. 2017, pp. 13:1–13:24. Schloss Dagstuhl (2020). https://doi.org/10.4230/LIPIcs.CONCUR.2020.13
5. 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
6. Aubert, C., Medić, D.: Explicit identifiers and contexts in reversible concurrent calculus. In: Yamashita, S., Yokoyama, T. (eds.) Reversible Computation - 13th International Conference, RC 2021, Virtual Event, July 7-8, 2021, Proceedings. LNCS, vol. 12805, pp. 144–162. Springer (2021). https://doi.org/10.1007/978-3-030-79837-6_9
7. 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
8. Aubert, C., Varacca, D.: Processes against tests: Defining contextual equivalences (2022), https://hal.archives-ouvertes.fr/hal-03535565, submitted to Journal of Logical and Algebraic Methods in Programming
9. 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
10. 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
11. 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., vol. 15, pp. 369–387. Elsevier (1998). https://doi.org/10.1016/S1571-0661(05)80023-4, https://www.sciencedirect.com/journal/electronic-notes-in-theoretical-computer-science/vol/15/supp
12. 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
13. Cristescu, I., Krivine, J., Varacca, D.: Rigid families for CCS and the π-calculus. In: Leucker, M., Rueda, C., Valencia, F.D. (eds.) Theoretical Aspects of Com-

puting - ICTAC 2015 - 12th International Colloquium Cali, Colombia, October 29-31, 2015, Proceedings. LNCS, vol. 9399, pp. 223–240. Springer (2015). https://doi.org/10.1007/978-3-319-25150-9_14

14. Danos, V., Krivine, J.: Reversible communicating systems. In: Gardner, P., Yoshida, N. (eds.) CONCUR 2004 - Concurrency Theory, 15th International Conference, London, UK, August 31 - September 3, 2004, Proceedings. LNCS, vol. 3170, pp. 292–307. Springer (2004). https://doi.org/10.1007/978-3-540-28644-8_19
15. Degano, P., Gadducci, F., Priami, C.: Causality and replication in concurrent processes. In: Broy, M., Zamulin, A.V. (eds.) Perspectives of Systems Informatics, 5th International Andrei Ershov Memorial Conference, PSI 2003, Akademgorodok, Novosibirsk, Russia, July 9-12, 2003, Revised Papers. LNCS, vol. 2890, pp. 307–318. Springer (2003). https://doi.org/10.1007/978-3-540-39866-0_30
16. Degano, P., Priami, C.: Proved trees. In: Kuich, W. (ed.) Automata, Languages and Programming, 19th International Colloquium, ICALP92, Vienna, Austria, July 13-17, 1992, Proceedings. LNCS, vol. 623, pp. 629–640. Springer (1992). https://doi.org/10.1007/3-540-55719-9_110
17. 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
18. Degano, P., Priami, C.: Enhanced operational semantics. ACM Comput. Surv. **33**(2), 135–176 (2001). https://doi.org/10.1145/384192.384194
19. 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
20. 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
21. 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
22. Lanese, I.: From reversible semantics to reversible debugging. In: Kari, J., Ulidowski, I. (eds.) Reversible Computation - 10th International Conference, RC 2018, Leicester, UK, September 12-14, 2018, Proceedings. LNCS, vol. 11106, pp. 34–46. Springer (2018). https://doi.org/10.1007/978-3-319-99498-7_2
23. Lanese, I., Lienhardt, M., Mezzina, C.A., Schmitt, A., Stefani, J.: Concurrent flexible reversibility. In: Felleisen, M., Gardner, P. (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. LNCS, vol. 7792, pp. 370–390. Springer (2013). https://doi.org/10.1007/978-3-642-37036-6_21
24. Lanese, I., Medić, D., Mezzina, C.A.: Static versus dynamic reversibility in CCS. Acta Inform. (Nov 2019). https://doi.org/10.1007/s00236-019-00346-6
25. 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
26. Lanese, I., Phillips, I.: Forward-reverse observational equivalences in CCSK. In: Yamashita, S., Yokoyama, T. (eds.) Reversible Computation - 13th International Conference, RC 2021, Virtual Event, July 7-8, 2021, Proceedings. LNCS, vol. 12805, pp. 126–143. Springer (2021). https://doi.org/10.1007/978-3-030-79837-6_8

27. Lanese, I., Phillips, I.C.C., Ulidowski, I.: An axiomatic approach to reversible computation. In: Goubault-Larrecq, J., König, B. (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. LNCS, vol. 12077, pp. 442–461. Springer (2020). https://doi.org/10.1007/978-3-030-45231-5_23
28. 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
29. 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, September 13-15. pp. 1–8. IEEE (2017). https://doi.org/10.1109/TASE.2017.8285635, https://ieeexplore.ieee.org/xpl/conhome/8277122/proceeding
30. Milner, R.: A Calculus of Communicating Systems. LNCS, Springer-Verlag (1980). https://doi.org/10.1007/3-540-10235-3
31. 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
32. 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
33. 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
34. Sangiorgi, D., Walker, D.: The Pi-calculus. CUP (2001)
35. 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
36. Ulidowski, I., Phillips, I., Yuen, S.: Concurrency and reversibility. In: Yamashita, S., Minato, S. (eds.) Reversible Computation - 6th International Conference, RC 2014, Kyoto, Japan, July 10-11, 2014. Proceedings. LNCS, vol. 8507, pp. 1–14. Springer (2014). https://doi.org/10.1007/978-3-319-08494-7_1
37. Wang, Y.: Retracted article: An algebra of reversible computation. SpringerPlus **5**(1), 1659 (Sep 2016). https://doi.org/10.1186/s40064-016-3229-7

A Decomposing Processes

Lemma 6 (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	$\|_{\mathrm{L}}\ \theta_1'$	$\|_{\mathrm{L}}\ \theta_1'$	$\|_{\mathrm{R}}\ \theta_1'$	$\|_{\mathrm{R}}\ \theta_1'$	$\langle \|_{\mathrm{L}}\ \theta_1', \|_{\mathrm{R}}\ \theta_1''\rangle$	$\langle \|_{\mathrm{L}}\ \theta_1', \|_{\mathrm{R}}\ \theta_1''\rangle$	$\langle \|_{\mathrm{L}}\ \theta_1', \|_{\mathrm{R}}\ \theta_1''\rangle$
θ_2	$\|_{\mathrm{L}}\ \theta_2'$	$\langle \|_{\mathrm{L}}\ \theta_2', \|_{\mathrm{R}}\ \theta_2''\rangle$	$\|_{\mathrm{R}}\ \theta_2'$	$\langle \|_{\mathrm{L}}\ \theta_2', \|_{\mathrm{R}}\ \theta_2''\rangle$	$\|_{\mathrm{L}}\ \theta_2'$	$\|_{\mathrm{R}}\ \theta_2'$	$\langle \|_{\mathrm{L}}\ \theta_2', \|_{\mathrm{R}}\ \theta_2''\rangle$

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 $\pi_{\mathrm{L}}(\theta_1) = \theta_1' \smile_{\pi_{\mathrm{L}}(T)} \theta_2' = \pi_{\mathrm{L}}(\theta_2)$ 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 7 (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 $T : X_{\mathrm{L}} + X_{\mathrm{R}} \xrightarrow{\theta_1} Y_{\mathrm{L}} + Y_{\mathrm{R}} \xrightarrow{\theta_2} Z_{\mathrm{L}} + Z_{\mathrm{R}}$ 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 $\rho_d(T) : \rho_d(X_{\mathrm{L}} + X_{\mathrm{R}}) \xrightarrow{\rho(\theta_1)} \rho_d(Y_{\mathrm{L}} + Y_{\mathrm{R}}) \xrightarrow{(\theta_2)} \rho_d(Z_{\mathrm{L}} + Z_{\mathrm{R}})$.

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:

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

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. □

B Sketched and Omitted Proofs

Lemma 1 (Adequation 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.*

Proof. The proof is by induction on the length of the derivation: since the only axiom rule (act.) is identical, it easily follow by an inspection of Fig 2 and 3. □

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

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 that $\not\kappa(\theta_1) \neq k$. As $\alpha[k].X_1' \xrightarrow{\theta_2} Y$ we know that $\not\kappa(\theta_2) \neq k$ [26, Lemma 3.4], and we can apply Lemma 2 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 2 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', $X_{1_{\mathrm{L}}}$, 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 6, 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 6. 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_{L}, X_{R}, $\theta_{1\mathrm{L}}$, $\theta_{1\mathrm{L}}$, $X_{1_{\mathrm{L}}}$, $X_{1_{\mathrm{R}}}$, Y_{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 $\langle |_{\mathrm{L}}\,\theta_{1\mathrm{L}}, |_{\mathrm{R}}\,\theta_{1\mathrm{R}}\rangle \smile |_{\mathrm{R}}\,\theta_{2\mathrm{R}}$ implies $\theta_{1\mathrm{R}} \smile \theta_{2\mathrm{R}}$ in $X_{\mathrm{R}} \xrightarrow{\theta_{1\mathrm{R}}} X_{1_{\mathrm{R}}} \xrightarrow{\theta_{2\mathrm{R}}} Y_{\mathrm{R}}$ by Lemma 6. 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 the previous one.

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 6 gives two traces

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

where $\theta_{1\mathrm{L}} \smile \theta_{2\mathrm{L}}$ 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

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

Note that we know 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 7 to obtain

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

and as $\theta'_1 \smile \theta'_2$ in this transition as well, 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 $X_{\mathrm{L}} + X_{\mathrm{R}} \xrightarrow{+_{\mathrm{L}}\theta'_2} X_2 + X_{\mathrm{R}} \xrightarrow{+_{\mathrm{L}}\theta'_1} Y_{\mathrm{L}} + X_{\mathrm{R}}$ and this concludes this case.

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

□

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

Proof. The proof is very similar to the proof of Theorem 1 in both cases. The proof of the first part is sketched in the body of the paper, and we detail below the complete proof of the second part, for completeness, and also because the sum case diverges and exposes the design choices made in Definition 6 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 $act.^\bullet$, 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.

$pre.^\bullet$ 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 $\not\kappa(\theta_2) \neq k$ [26, Lemma 3.4], and we can apply Lemma 2 twice to obtain

$$\mathrm{rm}^\alpha_k(\alpha[k].X') = X' \overset{\theta_1}{\rightsquigarrow} \mathrm{rm}^\alpha_k(\alpha[k].X'_1) = X'_1 \xrightarrow{\theta_2} \mathrm{rm}^\alpha_k(Y)$$

As $\theta_1 \smile \theta_2$ by hypothesis, we can use the induction hypothesis to obtain that there exists X_2 s.t. $X' \xrightarrow{\theta_2} X_2 \overset{\theta_1}{\rightsquigarrow} \mathrm{rm}^\alpha_k(Y)$. 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 2 one last time, we obtain that $\mathrm{rm}^\alpha_k(\alpha[k].X_2) = X_2 \overset{\theta_1}{\rightsquigarrow} \mathrm{rm}^\alpha_k(Y)$ 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 $|_{\mathrm{L}}$ 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 $\theta'_1 \smile \theta'_2$ in $X_{\mathrm{L}} \overset{\theta'_1}{\rightsquigarrow} X_{1_{\mathrm{L}}} \xrightarrow{\theta'_2} Y_{\mathrm{L}}$ by Lemma 6, 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 that $\theta'_1 \smile \theta_{2\mathrm{L}}$ in $X_{\mathrm{L}} \overset{\theta'_1}{\rightsquigarrow} X_{1_{\mathrm{L}}} \xrightarrow{\theta_{2\mathrm{L}}} Y_{\mathrm{L}}$ by Lemma 6. 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}, $\theta_{1\mathrm{L}}$, $\theta_{1\mathrm{R}}$, $X_{1_{\mathrm{L}}}$, $X_{1_{\mathrm{R}}}$, 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 $\theta_{1\mathrm{R}} \smile \theta_{2\mathrm{R}}$ in $X_{\mathrm{R}} \overset{\theta_{1\mathrm{R}}}{\rightsquigarrow} X_{1_{\mathrm{R}}} \xrightarrow{\theta_{2\mathrm{R}}} Y_{\mathrm{R}}$ by Lemma 6. 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 the previous one.

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 6 gives two traces

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

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

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

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

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

$+^{\bullet}_{\mathrm{L}}$ There exists X_{L}, X_{R}, $X_{1\mathrm{L}}$, and Y_{L} s.t. $X \overset{\theta_1}{\rightsquigarrow} X_1 \overset{\theta_2}{\longrightarrow} Y$ is

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

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, $X_{1\mathrm{L}} \overset{\theta'_2}{\longrightarrow} Y_{\mathrm{L}}$ and $X_{\mathrm{R}} = Y_{\mathrm{R}}$. Since $+_{\mathrm{L}}\theta'_1 \smile +_{\mathrm{L}}\theta'_2$, we can use Lemma 7 to obtain

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

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

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

from which it is easy to obtain

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

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 3 (Square property). *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* $t'_2 : X_2 \overset{\theta_1}{\twoheadrightarrow} Y$*.*

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

If $t_1 : X \overset{\theta_1}{\rightsquigarrow} X_1$ **and** $t_2 : X \overset{\theta_2}{\longrightarrow} X_2$ Since t_1 and t_2 are concurrent, by Definition 11 we have that $\theta_1 \smile \theta_2$ in $t_1^{\bullet}; t_2 : X_1 \overset{\theta_1}{\longrightarrow} X \overset{\theta_2}{\longrightarrow} X_2$. Hence, by the sideways diamond (Theorem 1) we obtain $t''_1; t''_2 : X_1 \overset{\theta_2}{\longrightarrow} Y \overset{\theta_1}{\longrightarrow} X_2$, and letting $t'_1 = t''_1$ and $t'_2 = t''^{\bullet}_2$, we obtain $t'_1 : X_1 \overset{\theta_2}{\longrightarrow} Y$ and $t'_2 : X_2 \overset{\theta_1}{\rightsquigarrow} Y$ as desired.

If $t_1 : X \overset{\theta_1}{\longrightarrow} X_1$ **and** $t_2 : X \overset{\theta_2}{\rightsquigarrow} X_2$ Since t_1 and t_2 are concurrent, t_2 and t_1 also are, and by Definition 11 we have that $\theta_2 \smile \theta_1$ in $t_2^{\bullet}; t_1 : X_2 \overset{\theta_2}{\longrightarrow} X \overset{\theta_1}{\longrightarrow} X_1$. Hence, by the sideways diamond (Theorem 1) we obtain $t''_2; t''_1 : X_2 \overset{\theta_1}{\longrightarrow} Y \overset{\theta_2}{\longrightarrow} X_1$, and letting $t'_2 = t''_2$ and $t'_1 = t''^{\bullet}_1$, we obtain $t'_1 : X_1 \overset{\theta_2}{\rightsquigarrow} Y$ and $t'_2 : X_2 \overset{\theta_1}{\longrightarrow} Y$ as desired.

If $t_1 : X \overset{\theta_1}{\rightsquigarrow} X_1$ **and** $t_2 : X \overset{\theta_2}{\rightsquigarrow} X_2$ Since t_1 and t_2 are concurrent, by Definition 11 we have that $\theta_1 \smile \theta_2$ in $t_1^{\bullet}; t_2 : X_1 \overset{\theta_1}{\longrightarrow} X \overset{\theta_2}{\rightsquigarrow} X_2$. Hence, by the first part of the reverse diamonds (Theorem 2), we obtain $t''_1; t''_2 : X_1 \overset{\theta_2}{\rightsquigarrow} Y \overset{\theta_1}{\longrightarrow} X_2$, 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 $t_1 : X \overset{\theta_1}{\longrightarrow} X_1$ **and** $t_2 : X \overset{\theta_2}{\longrightarrow} X_2$ Since t_1 and t_2 are concurrent, by Definition 11 we have that $\theta_1 \smile \theta_2$ in $t_1^{\bullet}; t_2 : X_1 \overset{\theta_1}{\rightsquigarrow} X \overset{\theta_2}{\longrightarrow} X_2$. Hence, by the second part of the reverse diamonds (Theorem 2) we obtain $t''_1; t''_2 : X_1 \overset{\theta_2}{\longrightarrow} Y \overset{\theta_1}{\rightsquigarrow} X_2$, and letting $t'_1 = t''_1$ and $t'_2 = t''^{\bullet}_2$, we obtain $t'_1 : X_1 \overset{\theta_2}{\longrightarrow} Y$ and $t'_2 : X_2 \overset{\theta_1}{\longrightarrow} Y$ as desired.

□

Lemma 4 (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. The first important fact to note is that $\not\kappa(\theta_1) \neq \not\kappa(\theta_2)$: 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 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. $X = \alpha[k].X' \overset{\theta_1}{\rightsquigarrow} \alpha[k].X'_1 = X_1$, hence 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 $X = \alpha[k].X' \overset{\theta_2}{\rightsquigarrow} \alpha[k].X'_2 = X_2$, $X' \overset{\theta_2}{\rightsquigarrow} X'_2$, and we conclude by using the induction hypothesis on the two backward transitions of X' and the observation that $pre.^\bullet$ preserves 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 6 to decompose the traces:

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

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 $X_L \mid X_{\mathrm{R}} \overset{\langle |_{\mathrm{L}}\theta_{2\mathrm{L}}, |_{\mathrm{R}}\theta_{2\mathrm{R}} \rangle}{\rightsquigarrow} X_{2_{\mathrm{L}}} \mid X_{2_{\mathrm{R}}}$, and we know that $|_L\ \theta'_1$ and $\langle |_{\mathrm{L}}\ \theta_{2\mathrm{L}}, |_{\mathrm{R}}\ \theta_{2\mathrm{R}} \rangle$ are concurrent if θ'_1 and $\theta_{2\mathrm{L}}$ are. 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 $|_{\mathrm{L}}$.

$syn.^\bullet$ This case is very 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 a simple 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 7 to decompose the trace.

□

C Comparing Concurrencies Accross Calculi

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

- It is equivalent to the restriction to CCSK of the definition of concurrency for a reversible π-calculus extending CCSK [28] (Sect. C.1),
- Our definition, when adapted to RCCS (Sect. C.3), yields a concurrency that extends (Sect. C.4) existing definitions for RCCS (Sect. C.2),
- Our definition can similarly be adapted to an "identified" declension of RCCS and proven equal to its definition of concurrency (Sect. C.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: in this sense, recognizing more transitions as concurrent is an interesting improvement. We also briefly illustrate, p. 27, that the concurrency stemming from the first item does not satisfy the "denotationality" [13, Section 6] criteria, i.e. that it is not preserved by CCSK's structural congruence.

C.1 Comparing With Concurrency Stemming From Reversible π-Calculus

A definition of concurrency was introduced for a reversible π-calculus extending CCSK [28], 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.

Causalities: Definitions and Adequations The following definitions can easily be extended to CCSK with sum, so we preserves the "full" system for this study of the adequation of causality.

Definition 13 (Context). *A* context *is a* CCSK *process with a slot* $\cdot$*:*

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

Definition 14 (Structural cause [28, Definition 21]). *For all* X, $m_1, m_2 \in \mathrm{key}(X)$*, 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* $m_2 \in \mathrm{key}(Y)$*.*

Definition 15 (Structural causality [28, Definition 22]). *In* $t_1; t_2 : X \xrightarrow{\alpha_1[m_1]} X_1 \xrightarrow{\alpha_2[m_2]} X_2$*,* 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.*

Below, we let f be the function that maps keyed labels to proved labels obtained from Lemma 1.

Lemma 8 (Adequation of the causalities). *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 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 $m_2 \in \not\Subset(Y)$, 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 $m_2 \in \not\Subset(X_2)$, 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.** 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 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 + 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.

□

Conflict and Concurrencies For reversible π-calculus, the causality relation requires to account for names previously shared, using an object causality [28, Definition 23], that is not meaningful nor required in CCSK. However, transitions of opposite direction need to be accounted for with a conflict relation [28, Definition 25] that we restate below:

Definition 16 (Conflict relation [28, 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. $t_1; t_2 : a[k] + b \overset{a[k]}{\rightsquigarrow} a + b \xrightarrow{b[k]} a + b[k]$, t_1 and t_2 would not be in conflict according to Definition 16, as t_2 does not "consume" a prefix freed by t_1. However, it would not be correct to declare them concurrent (as would [28, Definition 26] do), 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 9 (Adequation 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* $f(\alpha_1[m_1]) \preccurlyeq f(\alpha_2[m_2])$.

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 $f(\alpha_1[m_1]) \preccurlyeq f(\alpha_2[m_2])$, 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 $f(\alpha_1[m_1]) \preccurlyeq f(\alpha_2[m_2])$ never holds, but it follows easily from Lemma 4: since $t_2 \neq t_1^\bullet$, we know that the co-initial backward transitions $t_1^\bullet$ and t_2 are different, and hence by Lemma 4 that they are concurrent, proving that $f(\alpha_1[m_1]) \preccurlyeq f(\alpha_2[m_2])$ does not hold.

If t_1 is backward Then we have to prove that $f(\alpha_1[m_1]) \preccurlyeq f(\alpha_2[m_2])$ 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 $f(\alpha_1[m_1]) \preccurlyeq f(\alpha_2[m_2])$ will hold. For the reverse direction, inspecting the Action and Parallel rules of Fig. 7 suffices to prove that $f(\alpha_1[m_1]) \preccurlyeq f(\alpha_2[m_2])$ 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 by definition for this latter definition of concurrency.

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 17 (Free and bound keys [26, 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 18 (Structural equivalence [26, p. 133]). *The structural equivalence of* CCSK *is the smallest equivalence relation (that is, reflexive, symmetric, and transitive relation) closed under the following rule:*

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

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 rule and its reverse:

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

For technical reasons beyond the scope of this exposition, this rule 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 5. *The conflict relation inherited from the reversible π-calculus is not adequate for* CCSK *endowed with structural congruence.*

Proof. Consider the following two equations and derivation:

$$a[k].c|\overline{a}[k] \equiv a[k].c|\overline{a}[k] \tag{1}$$
$$a[k].c[k']|\overline{a}[k] \equiv a[h].c[k']|\overline{a}[h] \tag{2}$$

$$\frac{\dfrac{\dfrac{\dfrac{}{c \xrightarrow{c[k']} c[k']}\ \text{act.}}{a[k].c \xrightarrow{c[k']} a[k].c[k']}\ \text{pre.} \qquad (2)}{a[k].c|\overline{a}[k] \xrightarrow{c[k']} a[k].c[k']|\overline{a}[k]}\ |_{\mathrm{L}}. \qquad (1)}{a[k].c[k']|\overline{a}[k] \xrightarrow{c[k']} a[h].c[k']|\overline{a}[h]}\ \text{equiv.}$$

Then, it is clear that $t_1; t_2 : a.c \mid \overline{a} \xrightarrow{\tau[k]} a[k].c|\overline{a}[k] \xrightarrow{c[k']} a[h].c[k']|\overline{a}[h]$ and yet since $k \notin \mathrm{key}(a[h].c[k']|\overline{a}[h])$, t_1 is not seen as a structural cause of t_2 according to Definition 14, 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.

C.2 Recalling RCCS's Concurrencies

It is relatively easy to adapt our proved labeled to RCCS, no matter which declension of the calculus you look at [3,4,14,21,24]. Below, we look at the "early" version of RCCS [14, 21] because, to our knowledge, it is the only version that received a syntactical definition of concurrency, relying on memory inclusion [21, Definition 3.11] or disjointness [14, Definition 7]. This version is fairly "heavy", since transitions are labeled with the memory of the thread executing, but it is immediate to add prefixes to those labels. We briefly remind this system below, and refer to their original presentations [14, 21] for more details. We do not consider recursive definitions, briefly discussed in this versions of RCCS.

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 two rules into one, to recover the classical prefix by letting $n = 1$, but also to represent 0 by letting $n = 0$. But we generally use binary sum, written $+$, write $\alpha.P$ for $\alpha.P + 0$ [4, Sect. 2.2], and define the structural equivalence using this binary sum (Definition 20).

Definition 19 (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 $\mathrm{nm}(m) = \{\alpha \mid \alpha \in \mathsf{N}$ *or* $\overline{\alpha} \in \overline{\mathsf{N}}$ *occurs in* $m\}$ *be the set of (co-)names occurring in* m.

Definition 20 (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:*

$$
\begin{aligned}
P + 0 &\equiv P & P + Q &\equiv Q + P\\
(P_1 + P_2) + P_3 &= P_1 + (P_2 + P_3) & P &\equiv Q \quad \textit{if } P =_\alpha Q
\end{aligned}
$$

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" [24], since the memory can "move" during execution.

Notation 1. *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 21 (Replacement operator). *The operation* @ *is defined as follows:*

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

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.

Concurrency on Co-initial Transitions

Definition 22 (Concurrency on co-initial transitions in RCCS [14, 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* $t_1 \smile_o t_2$*.*

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, taking

$$t_1 : \langle \star, a, Q'\rangle \cdot \langle\rangle \rhd (b.P + Q) \xrightarrow{\langle \star, b, 0\rangle \cdot \langle\rangle : b} \langle \star, b, Q\rangle \cdot \langle \star, a, Q'\rangle \cdot \langle\rangle \rhd P$$
$$t_2 : \langle \star, a, Q'\rangle \cdot \langle\rangle \rhd (b.P + Q) \overset{\langle\rangle:a}{\rightsquigarrow} \langle\rangle \rhd a.(b.P + Q) + Q'$$

would give $t_1 \smile_o t_2$, since $\langle \star, a, 0\rangle \cdot \langle\rangle \cap \langle\rangle = \emptyset$, but the intuitive understanding of concurrency shows that those two transitions should actually *not* be concurrent. The definition also does not require that t_1 and t_2 should be different, but

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

$$\frac{}{\langle \star, \lambda, Q\rangle \cdot m \rhd P \overset{m:\lambda^-}{\rightsquigarrow} m \rhd (\lambda \cdot P + Q)}\ \text{act.}^- \qquad \frac{R \xrightarrow{\mu:\zeta}\!\!\!\!\rightarrow R' \quad \zeta \notin \{a, \overline{a}, a^-, \overline{a}^-\}}{R\backslash a \xrightarrow{\mu:\zeta}\!\!\!\!\rightarrow R'\backslash a}\ \text{res.}$$

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

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

Fig. 5. Rules of the labeled transition system (LTS) for RCCS

we assume that they must be, since otherwise transitions would not be concurrent with themselves. We will make those requirements explicit when proving the adequacy result with our definition of concurrency relying on proved labels (Theorem 6).

Concurrency on Composable Transitions

Definition 23 (Precedence [21, 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 24 (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* $t \smile_o t'$.

Note that we use the same symbol $\smile_o$ in Definitions 22 and 24, but that there is no ambiguity, since the transitions needs to be either composable or co-initial for the relations to be defined for them.

Composable transitions of opposite directions are neither concurrent nor not concurrent: precedence is not defined on those transitions, and so neither is concurrency. In RCCS, the loop lemma [21, Lemme 2.2.1] also holds, and we write t^- the reverse of t. Note that, given $t_1; t_2$ two composable transition, it is not possible to ask whenever t_1 and t_2 are concurrent w.r.t. composable concurrency iff t_1^- and t_2 are concurrent w.r.t. to the co-initial concurrency: since both notions requires both transitions to have the same direction, one cannot compare the two relations.

C.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 proving how enriched labels give a notion of concurrency equivalent to the previous ones.

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 primary connector after a reduction, as opposed to CCSK, there is no need for the $+_{\mathrm{L}}$ and $+_{\mathrm{R}}$ annotations anymore.

Definition 25 (Enhanced labels). *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 $\ell(\upsilon\zeta) = \zeta$, $\ell(\upsilon\overline{\zeta}) = \overline{\zeta}$, and $\ell(\langle |_{\mathrm{L}}\, \upsilon_{\mathrm{L}}\zeta, |_{\mathrm{R}}\, \upsilon_{\mathrm{R}}\overline{\zeta}\rangle) = \tau$.

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. 4). 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) and (Composition Associativity).

Definition 26 (Dependency relation). *The* dependency relation *on enhanced keyed labels is induced by the axioms of Fig. 7, for $d \in \{\mathrm{L}, \mathrm{R}\}$.*

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, causality relation and concurrency are defined as in Definitions 7–9 and 11.

Exactly like for CCSK with Lemma 1, it is easy to prove the adequation of the proved system w.r.t. the original one:

Lemma 10 (Adequation 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. □

C.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.

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

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

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

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

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

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

Fig. 6. Rules of the proved labeled transition system (LTS) for RCCS

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

Palallel Group	
$\|_d\ \theta \lessdot \|_d\ \theta'$	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. $\theta_d \lessdot \theta'_d$

Fig. 7. Dependency Relation on Enhanced Keyed Labels

On Co-initial Traces

Theorem 6. *For all different co-intial transitions with the same direction* $t_1 = R \xrightarrow{\mu_1:\zeta_1} S_1$ *and* $t_2 = R \xrightarrow{\mu_2:\zeta_2} S_2$, $t_1 \smile_o t_2$ *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 for all i, $m \rhd \sum \alpha_i.P_i \xrightarrow{m:\alpha_i} P_i$, and since $m \cap m = m$, those transitions are not pairwise concurrent. Since $\alpha_i \lessdot \theta$, 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 its memory component either be a pair, or it is prefixed by $\langle 1\rangle$ or by $\langle 2\rangle$. If at least one memory 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. Hence, we reason below on 2, 3 or 4 memories, depending on the number of synchronizations between the two transitions concerned.
If memories in both transitions are prefixed by the same $\langle i\rangle$, then since $|_d\theta \lessdot |_d\theta'$ if $\theta \lessdot \theta'$, we can proceed by induction. If memories in both transitions are prefixed by different $\langle i\rangle$, then the transitions will be concurrent, and since $|_{\mathrm{L}}\theta \lessdot |_{\mathrm{R}}\theta'$ (and reciprocally) never holds, we are done with this case.

$(a)R$ 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. □

On Composable Transitions

Theorem 7. *For all different composable transitions with the same direction* $t_1 = R \xrightarrow{\mu_1:\zeta_1} S_1$ *and* $t_2 = S_1 \xrightarrow{\mu_2:\zeta_2} S_2$, $t_1 \smile_o t_2$ *iff* $\neq (f(\zeta_1) \lessdot f(\zeta_2))$

Proof. We need to prove that t_1 precedes t_2 iff $f(\zeta_1) \lessdot f(\zeta_2)$. We can prove only the forward case, since if both transitions are backward, t_1 precedes t_2 iff t_2^- preceeds t_1^-. 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 $f(\zeta_1) \ll f(\zeta_2)$.
syn. Then t_1 precedes t_2 (resp. $f(\zeta_1) \ll f(\zeta_2)$) iff t_1' precedes t_2' (resp. $f(\zeta_1) \ll f(\zeta_2')$) 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.

□

C.5 Reversible and Identified CCS

We refer to the original paper [6] 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* [6, Definition 4], which is an *identifier patterns* [6, Definition 1] that dynamically generates the identifiers for each transition, and that can get split [6, 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 [6, Definition 12]—written $i_1 \perp i_2$—or not downstream, both conditions essentially stating that the transition involved different threads.

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

Definition 27 (Concurrency). *Two different coinitial transitions* $t_1 : \mathsf{s} \circ m \rhd P \xrightarrow{\alpha_1[i_1]} \mathsf{s}_1 \circ m_1 \rhd P_1$ *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 adequation in the sense of Lemma 10—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 8. *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* $f(\alpha_1) \ll f(\alpha_2)$ *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 $\neg(f(i_1 : \alpha_1) \ll f(i_2 : \alpha_2))$:

- 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 27, and we have this result as well for proved labels, by adapting the proof for proved CCSK (Lemma 4) to this proved identified RCCS. □