Replications in Reversible Concurrent Calculi

Clément Aubert

Replications in Reversible Concurrent Calculi (Short paper)

Clément Aubert[0000−0001−6346−3043]

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

Abstract. Process calculi such as CCS or the π-calculus provide specification languages for the study and correctness of communication protocols. They also served in detailing subtle differences between formalisms to represent infinite behaviors, notably in expressiveness [4,14]. To our knowledge, such results were never investigated from a reversible perspective. One question we would like to answer is how recursion, replication and iteration compare in the reversible setting. Of course, comparing them requires to define them first, and this short note draws possible paths toward a definition of replication for reversible concurrent calculi.

Keywords: Formal semantics · Process algebras and calculi · Reversible Computation · Concurrency · Replication

1 What Makes a Good Reversible Calculi?

A specification language for protocols should itself has strong, desirable properties, and presenting it as a labeled transition system (LTS) often makes it amenable to mathematical reasoning. Thanks to the axiomatic approach to reversible computation [12], it is known that most of the desirable properties for reversible LTSes can be derived from three axioms: the square property (SP), the fact that backward transitions are independent (BTI), and well-foundedness (WF). We recently offered to extend the square property with what we called "the diamonds" [2], but this does not change the fact that there is a general consensus in the community that a LTS not respecting those properties will not characterize faithfully (causal consistent) reversible communications.

As a consequence, endowing reversible LTSes with infinite behaviors requires to check that those axioms are preserved. Representing infinite behaviors is of interest for multiple reasons: in addition to mathematical curiosity (what *is* infinity when you can go back at any time?), it is a staple element of any good specification language, since it allows to represent the spawning of new threads.

This short note offers to look at the reference process calculus for reversible systems—CCSK [11,15]—endowed with an original definition of concurrency [1,2], and to highlight some of the difficulties arising from the addition of a replication operator to the LTS.

2 Reminders on Reversible Concurrent Calculi

We very briefly restate the syntax and semantics of CCSK, using the definition of concurrency inspired by proved transition systems [5,6,7]–a variation on LTSes that eases the syntactical definition of concurrency.

Definition 1 ((Co-)names, labels and keys). *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 written* $\overline{\cdot}$*, gives the* complement *of a name, and we let* $\overline{\tau} = \tau$*. Finally, we let* $\mathsf{K} = \{m, n, \dots\}$ *be a set of* keys*, and let* k *range over them.*

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

$$
\begin{array}{rrrr}
X, Y ::= 0 & \text{(Inactive process)} & X\backslash\alpha & \text{(Restriction)} \\
\alpha.X & \text{(Prefix)} & X + Y & \text{(Sum)} \\
\alpha[k].X & \text{(Keyed prefix)} & X \mid Y & \text{(Parallel composition)}
\end{array}
$$

The inactive process 0 *is omitted when preceded by a (keyed) prefix, we write* $\mathrm{key}(X)$ *for the set of keys in* X*, and* $\mathrm{std}(X)$ *iff* $\mathrm{key}(X) = \emptyset$*, i.e., if* X *is* standard.

Definition 3 (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*

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

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

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

We present in Fig. 1 the rules for the *proved* forward and backward LTS for CCSK, let $\twoheadrightarrow = \rightarrow \cup \rightsquigarrow$, and $\cdot^*$ be the transitive closure of any of those three relations. The rules $|_{\mathrm{R}}$, $|^{\bullet}_{\mathrm{R}}$, $+_{\mathrm{R}}$ and $+^{\bullet}_{\mathrm{R}}$ are omitted but can easily be inferred.

Definition 4 (Dependency and concurrency). *The* dependency relation $\lessdot$ *on enhanced keyed labels is induced by the axioms below:*

Action

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

Sum Group

$$
\begin{array}{ll}
+_{\mathrm{L}}\theta \lessdot +_{\mathrm{R}}\theta' & \\
+_{\mathrm{R}}\theta \lessdot +_{\mathrm{L}}\theta' & \\
+_{d}\theta \lessdot +_{d}\theta' & \text{if } \theta \lessdot \theta'
\end{array}
$$

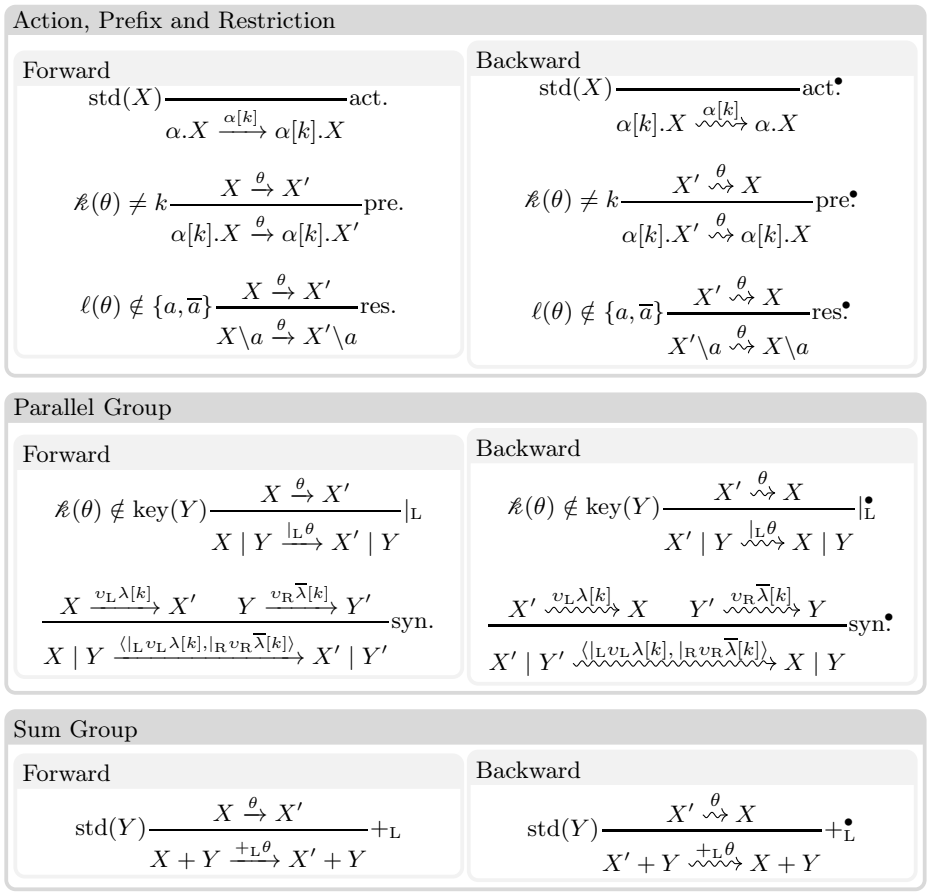

Fig. 1. Rules of the *proved* LTS for CCSK

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

Parallel Group	
$\lvert_d\theta \lessdot \lvert_d\theta'$	*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.*$\theta_d \lessdot \theta'_d$

Two enhanced keyed labels θ_1 *and* θ_2 *are* concurrent, $\theta_1 \smile \theta_2$, *iff neither* $\theta_1 \lessdot \theta_2$ *nor* $\theta_2 \lessdot \theta_1$.

Definition 5 (Concurrencies). *A transition* $t_1 : X \xrightarrow{\theta_1} Y_1$ *is* concurrent *with* $t_2 : Y_1 \xrightarrow{\theta_2} Y_2$ *or with* $t_2 : X \xrightarrow{\theta_2} Y_3$, $t_1 \smile t_2$, *iff* $\theta_1 \smile \theta_2$.

We refer the reader to our pre-print [2] for an in-depth discussion of the benefits and correctness of this definition.

3 Replication and Reversibility

Two different sets of rules for replications have been considered for CCS and π-calculus. Their benefits and drawbacks have been discussed in detail [16, pp. 42–43] for forward-only systems, but, to our knowledge, never in a reversible setting. We briefly discuss the elements needed to define them, and then how they behave.

3.1 Preamble – How to Make My LTS Infinite?

Adding replication to CCSK requires to

1. Add $!X$ to the set of operators (Definition 2),
2. Add $!$ to the set of strings over which υ, υ_{L} and υ_{R} can range in Definition 3,
3. Add rules for the $!$ operator in the proved LTS (Fig. 1),
4. Fine-tune the definition of the dependency relation (Definition 4).

Then, proving that it is "correct" requires to prove SP,[1] BTI and WF:

$$\forall t_1 : X \xrightarrow{\theta_1}\!\!\!\!\rightarrow X_1, t_2 : X \xrightarrow{\theta_2}\!\!\!\!\rightarrow X_2 \text{ with } t_1 \smile t_2, \exists t_1' : X_1 \xrightarrow{\theta_2}\!\!\!\!\rightarrow Y, t_2' : X_2 \xrightarrow{\theta_1}\!\!\!\!\rightarrow Y \quad \text{(SP)}$$

$$\forall t_1 : X \overset{\theta_1}{\rightsquigarrow} X_1, t_2 : X \overset{\theta_2}{\rightsquigarrow} X_2, t_1 \neq t_2 \implies t_1 \smile t_2 \quad \text{(BTI)}$$

$$\forall X, \exists X_0, \dots, X_n \text{ with } \mathrm{std}(X_0) \text{ s.t. } X \rightsquigarrow X_n \rightsquigarrow \cdots \rightsquigarrow X_1 \rightsquigarrow X_0 \quad \text{(WF)}$$

Below, we will assume that the first and second steps described above have been completed (they are straightforward), and compare two different sets of rules for the semantics of the $!$ operator, and how they impact the possibility of preserving those properties.

3.2 Replication – First Version

The first set of rules we consider is made of the following two and their reverses:

Infinite Group

$$\frac{X \xrightarrow{\theta} X'}{!X \xrightarrow{!\theta} !X \mid X'}\text{repl.}_1 \qquad \frac{X \xrightarrow{\theta_{\mathrm{L}}\lambda[k]} X' \qquad X \xrightarrow{\theta_{\mathrm{R}}\overline{\lambda}[k]} X''}{!X \xrightarrow{!\langle|_{\mathrm{L}}\theta_{\mathrm{L}}\lambda[k], |_{\mathrm{R}}\theta_{\mathrm{R}}\overline{\lambda}[k]\rangle} !X \mid (X' \mid X'')}\text{repl.}_2$$

And the dependency relation (Definition 4) now includes

$$!\theta \lessdot !\theta' \quad \text{if } \theta \lessdot \theta' \qquad !\theta \lessdot |_{\mathrm{R}}\theta' \text{ if } \begin{cases} \theta = \langle \theta_{\mathrm{L}}, \theta_{\mathrm{R}}\rangle \text{ and } \theta' =|_{\mathrm{L}} \theta'' & \text{or} \\ \theta = \langle \theta_{\mathrm{L}}, \theta_{\mathrm{R}}\rangle \text{ and } \theta' = |_{\mathrm{R}}\theta'' & \text{or} \\ \theta \lessdot \theta' & \text{otherwise.} \end{cases}$$

$$!\theta \lessdot |_{\mathrm{L}}\, \theta'$$

[1] This version of SP is a slight generalization, as motivated in our related work [2].

There are three issues with this version: 1. The dependency relation becomes cumbersome to manipulate, 2. We are forced to treat as dependent transitions that *should* be concurrent, 3. We cannot obtain (BTI). The details supporting those claims are shared in Sect. A.

3.3 Replication – Second Version

The second set we consider is actually older [13], and made of the following rules:

Infinite Group

Forward	Backward
$\dfrac{!X \mid X \xrightarrow{\theta} X'}{!X \xrightarrow{!\theta} X'}\text{rep.}$	$\dfrac{X' \overset{\theta}{\rightsquigarrow} !X \mid X}{X' \overset{!\theta}{\rightsquigarrow} !X}\text{rep}^\bullet$

We extend the dependency relation (Definition 4) with $\begin{array}{l} !\theta \lessdot |_{\mathrm{L}}\theta' \\ !\theta \lessdot |_{\mathrm{R}}\theta' \end{array}$ if $\theta \lessdot \theta'$.

This version seems to behave more nicely w.r.t. (BTI): e.g., we have

$$!a.X \mid a[m].P \overset{|_{\mathrm{R}}a[m]}{\rightsquigarrow} !a.X \mid a.X \qquad \text{and} \qquad !a.X \mid a[m].P \overset{!|_{\mathrm{R}}a[m]}{\rightsquigarrow} !a.X$$

but as $|_{\mathrm{R}}a[m] \smile !|_{\mathrm{R}}a[m]$, both transitions *are* independent. However, obtaining (SP) would require to find backward transitions from $!a.X \mid a.X$ and $!a.X$ converging to the same process, which is impossible since they are both standard.

In addition, this formalism "make[s] it difficult to obtain any result concerning causality in our approach, which is based on enhanced labels, hence it relies on proofs by structural induction" [6, p. 310]. Those types of proofs are really difficult to carry on with the possibility offered by those rules to (des)activate a copy of X at any depth, and all our attempts at proving (SP) have failed so far (even setting aside the case we discussed earlier). The impossibility to carry on proofs by induction with this rule has been known for a long time [16, p. 43], and it does not seem that adding reversibility allows to sidestep any of it.

4 Concluding Notes

Two interesting features of those two versions are first that we do not need to worry about memory duplication, since $!X$ is reachable iff X is standard. Second, we can note that every process reachable from $!X$ has infinitely many origins: if $!X \rightarrow^* Y$, then $Y \rightsquigarrow^* !X$, $Y \rightsquigarrow^* !X \mid X$, $Y \rightsquigarrow^* (!X \mid X) \mid X$, etc. Obtaining a well-behaved system with replication could require adopting a powerful structural equivalence, but this is a difficult task, both for proved transition systems and for reversible systems. Indeed, the "usual" structural congruence is missing from all the proved transition systems [5,7,9,10], or missing the associativity and commutativity of the parallel composition [8, p. 242]. While adding such a congruence would benefit the expressiveness, making it interact nicely with the

derived proof system *and* the reversible features [11, Section 4][3] is a challenge that may not even grant the required properties to define a replication satisfying the desired properties we discussed in this short note.

Acknowledgments This material is based upon work supported by the National Science Foundation under Grant No. 2242786 (SHF:Small:Concurrency In Reversible Computations).

References

1. Aubert, C.: Concurrencies in Reversible Concurrent Calculi. In: Mezzina, C.A., Podlaski, K. (eds.) RC 2022. LNCS, vol. 13354, pp. 146–163. Springer (2022). https://doi.org/10.1007/978-3-031-09005-9_10
2. Aubert, C.: The Correctness of Concurrencies in (Reversible) Concurrent Calculi (Jan 2023), https://hal.science/hal-03950347, submitted to JLAMP
3. Aubert, C., Cristescu, I.: Structural equivalences for reversible calculi of communicating systems (oral communication). Research report, Augusta University (2020), https://hal.science/hal-02571597, communication at ICE 2020
4. 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
5. Carabetta, G., Degano, P., Gadducci, F.: CCS semantics via proved transition systems and rewriting logic. In: Kirchner, C., Kirchner, H. (eds.) WRLA 1998. Electron. Notes Theor. Comput. Sci., vol. 15, pp. 369–387. Elsevier (1998). https://doi.org/10.1016/S1571-0661(05)80023-4
6. Degano, P., Gadducci, F., Priami, C.: Causality and replication in concurrent processes. In: Broy, M., Zamulin, A.V. (eds.) PSI 2003. LNCS, vol. 2890, pp. 307–318. Springer (2003). https://doi.org/10.1007/978-3-540-39866-0_30
7. Degano, P., Priami, C.: Proved trees. In: Kuich, W. (ed.) ICALP92. LNCS, vol. 623, pp. 629–640. Springer (1992). https://doi.org/10.1007/3-540-55719-9_110
8. 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
9. Degano, P., Priami, C.: Enhanced operational semantics. ACM Comput. Surv. **33**(2), 135–176 (2001). https://doi.org/10.1145/384192.384194
10. 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
11. Lanese, I., Phillips, I.: Forward-reverse observational equivalences in CCSK. In: Yamashita, S., Yokoyama, T. (eds.) RC 2021. LNCS, vol. 12805, pp. 126–143. Springer (2021). https://doi.org/10.1007/978-3-030-79837-6_8
12. Lanese, I., Phillips, I.C.C., Ulidowski, I.: An axiomatic approach to reversible computation. In: Goubault-Larrecq, J., König, B. (eds.) FOSSACS 2020. LNCS, vol. 12077, pp. 442–461. Springer (2020). https://doi.org/10.1007/978-3-030-45231-5_23
13. Milner, R.: Functions as processes. In: Paterson, M. (ed.) ICALP90. LNCS, vol. 443, pp. 167–180. Springer (1990). https://doi.org/10.1007/BFb0032030
14. 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

15. 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
16. Sangiorgi, D., Walker, D.: The Pi-calculus. CUP (2001)

A Some of the Issues With Version 1

Adding *five* axioms to the dependency relation to handle *one* operator is clearly not ideal. Some of that comes from the fact that we use the same relation for co-initial and composable transitions, and we believe that all of them are necessary. If we take the first one as an example ($!\theta \lessdot !\theta'$ if $\theta \lessdot \theta'$), it is justified by the fact that we want to see the following two co-initial transitions as dependent:

$$!(a+b) \xrightarrow{!+_{\mathrm{L}} a[m]} !(a+b) \mid (a[m]+b) \qquad !(a+b) \xrightarrow{!+_{\mathrm{R}} b[m]} !(a+b) \mid (a+b[m]).$$

While this first condition seems to be fulfilling a useful purpose, the conditions for $!\theta \lessdot |_{\mathrm{R}}\theta'$ are equally necessary, but force some transitions to be dependent while the common sense would see them as concurrent. For instance, letting $X = a \mid (\overline{a}+b)$, we have

$$X \xrightarrow{|_{\mathrm{L}} a[m]} a[m] \mid (\overline{a}+b) \xrightarrow{|_{\mathrm{R}}+_{\mathrm{R}} b[n]} a[m] \mid (\overline{a}+b[n])$$
$$X \xrightarrow{|_{\mathrm{R}}+_{\mathrm{L}} \overline{a}[m]} a \mid (\overline{a}[m]+b)$$

and hence

$$!X \xrightarrow{!\langle |_{\mathrm{L}}|_{\mathrm{L}} a[m], |_{\mathrm{R}}|_{\mathrm{R}}+_{\mathrm{L}} \overline{a}[m]\rangle} !X \mid \big((a[m] \mid (\overline{a}+b)) \mid (a \mid (\overline{a}[m]+b))\big)$$
$$\xrightarrow{|_{\mathrm{R}}|_{\mathrm{L}}|_{\mathrm{R}}+_{\mathrm{R}} b[n]} !X \mid \big((a[m] \mid (\overline{a}+b[n])) \mid (a \mid (\overline{a}[m]+b))\big)$$

It would seem intuitive to let those two transitions be concurrent, since $!X$ can decide to act on b first, and then synchronize a and $\overline{a}$. Indeed, since we have

$$X \xrightarrow{|_{\mathrm{R}}+_{\mathrm{R}} b[n]} a \mid (\overline{a}+b[n]) \xrightarrow{|_{\mathrm{L}} a[m]} a[m] \mid (\overline{a}+b[n])$$

The resulting transitions would be:

$$!X \xrightarrow{!|_{\mathrm{R}}+_{\mathrm{R}} b[n]} !X \mid (a \mid (\overline{a}+b[n]))$$
$$\xrightarrow{\langle |_{\mathrm{L}}!|_{\mathrm{R}}+_{\mathrm{L}} \overline{a}[m], |_{\mathrm{R}}|_{\mathrm{L}} a[m]\rangle} \big(!X \mid (a \mid (\overline{a}[m]+b))\big) \mid (a[m] \mid (\overline{a}+b[n]))$$

Even if we could let the proved labels be equivalent in some ways, since our system do not have a structural congruence, we cannot consider the two terms obtained after the transitions as equal or equivalent. So, to have a chance of obtaining (SP), we have to make more transitions dependent than we would like. We can observe, however, that the original system for forward-only transition had the same flaw [6, Definition 2].

Last but not least, this system does not enjoy (BTI). Indeed, consider e.g., the reachable process $X = !a.P \mid a[m].P$, we have

$$t_1 : X \overset{!a[m]}{\rightsquigarrow} !a.P \qquad \text{and} \qquad t_2 : X \overset{|_{\mathrm{R}} a[m]}{\rightsquigarrow} !a.P \mid a.P,$$

and since $!a[m] \lessdot |_{\mathrm{R}} a[m]$, t_1 and t_2 are *not* concurrent–this dependency is actually useful in proving (SP).

There are at least three options that could be explored to restore (BTI):

1. One could argue that t_1 and t_2 are the same up to the structural congruence $!P \mid P \equiv !P$ [4, Definition 12] (which is missing in our system and introduces many additional complications),
2. One could "mark" the key m so that it cannot backtrack independently, thus forbidding the $|_{\mathrm{R}} a[m]$-transition,
3. One could decide that rep.$_1$ and rep.$_2$ are irreversible rules.

Before exploring those options in too much details, we thought it would be preferable to explore the alternative formalism for replication, as we discuss in Sect. 3.3.