Reversibility and Predictions

Martin Vassor(✉)

Univ. Grenoble-Alpes, Inria, CNRS, Grenoble Inp, LIG Grenoble, Grenoble, France
martin.vassor@inria.fr

Abstract. This paper analyses the introduction of a non-reversible mechanism in a reversible calculus (called $\Omega\rho\pi$), intended to be used as an oracle which contains persistent memories of previously reversed computation. As a second step, we introduce the notion of weak causal consistency which relaxes the classical causal consistency by allowing the backward semantics not to revert to a previous state, but to a state related to a previous state and we show that $\Omega\rho\pi$ is weakly causally consistent. We finally present a practical application of this calculus.

1 Introduction

Motivations. Reversibility is the ability, for a system, to undo some actions that were previously taken. We can approach this field from various perspectives such as circuit design, quantum algorithms, automaton, etc. In this paper, we are interested in the application of reversibility to concurrent systems. There already exists multiple works in this context, for debugging [4], for fault tolerance [14,18], for biological or chemical modelling [1,2,7,16], or even for reliable concurrent programming abstraction [12].

In concurrent systems, the very notion of reversibility is not trivial. Indeed, since reductions are not deterministic, defining the notion of predecessor is less intuitive. For instance, consider the following CCS process [13]: $a.0 \mid b.0$. There are two possible forward reductions: either $a.0 \mid b.0 \xrightarrow{a} b.0$, or $a.0 \mid b.0 \xrightarrow{b} a.0$, and both reduce to 0: both $a.0$ and $b.0$ are predecessors of 0. Intuitively, reverting to any of those states is acceptable, regardless of the forward sequence.

The standard property of a reversible system is causal consistent reversibility, which captures this intuition. *Causal consistent reversibility* states that a backward reduction can undo an action, provided that its consequences (if any) are undone beforehand.

There are works which intentionally break causal consistent reversibility. Typical applications include reversible calculi to model chemical reactions with catalyst: an example is a reaction between three molecules A, B, and C, where the objective is to bind A and B together. For the reaction to happen, B first have to bind with the catalyst C, to create molecule BC, which can in turn bind with A, to create ABC. Finally, the first binding is reverted, which results in AB and C apart. One can clearly see that such reduction is not causally consistent: the first reaction is reverted while the second holds. Such reversibility is called

© Springer Nature Switzerland AG 2021
S. Yamashita and T. Yokoyama (Eds.): RC 2021, LNCS 12805, pp. 163–181, 2021.
https://doi.org/10.1007/978-3-030-79837-6_10

out-of-causal-order reversibility. Calculi which model such chemical reactions include [16] (which first explicitly mentions *out-of-causal-order reversibility*) and Kuhn and Ulidowski's Calculus of Covalent Bonding [6,7] (in which they distinguish CCB, which is not causally consistent, and CCBs a core calculus which is). [5] compares in detail three out-of-causal-order reversible process calculi, while [15] studies three forms of reversibility (backtracking, causal reversibility, and out-of-causal-order reversibility) in Petri nets.

Breaking causal consistency can also be motivated by practical applications. For instance, in [8], Lanese *et al.* introduce the **croll**-π calculus, which is a reversible π-calculus in which, when a memory is reinstalled in a revert sequence, the original process is replaced by a continuation. Such approach is not causally-consistent, *stricto sensu*[1].

In this paper, we study the relation between algorithms based on *trial and error* and reversibility. An example of such algorithm is a naive consensus algorithm, in which each process tries its own value, and when it receives a smaller value from a peer, it rolls back and tries again with this smaller value. Such algorithm converges toward a state in which all processes agree on the same value.

Informally, we focus on a class of algorithms that behave in four steps: (i) read the current state; (ii) improve it; (iii) store it; and (iv) start over. Among those four steps, only the second depends on the algorithm. Standard reversible process algebra, such as $\rho\pi$, implement the fourth one (or $\texttt{roll}-\pi$ [9] for explicit rollback control).

In this paper, our goal is to define a calculus which also covers steps (i) and (iii). In particular, the stored state should *not* be reverted during backward execution. Another way to view the mechanism is that processes can *predict* information from the future state of the system. This is the reason why we call the context an *oracle*.

Our second objective is to characterise the reversible nature of such calculus. Intuitively, such calculus is not causally-consistent, since the state of the store is not reverted. Therefore, we relax the notion of causal-consistency by introducing *weak causal consistency*. In a nutshell, weak causal consistency takes as parameter a relation $\mathcal{R}$, and allows backward reduction to reach new states (thus breaking regular causal consistency), under the condition that there exists a related state which is reachable using forward only reduction. In particular, for our application, we are interested in a simulation relation. Taking the identity as the parameter relation, our definition falls back on regular causal consistency. We expect this property to be an intermediate point between strong causal consistency and out-of-causal-order reversibility.

Another interesting aspect of this calculus is that the backward semantics is *partial*, in the sense that its effects are confined to some parts of the term. This notion of partial reversibility is barely discussed in this paper, but we think it is important to explicit it for its potential practical applications.

[1] Note that, actually, it still has some sort of causal consistency, in that backward semantics undo the latter messages first. Therefore it is not possible to have an effect without its cause, but the resulting state is not reachable without backward sequence.

Approach. At first, we introduce a calculus (called $\Omega\rho\pi$) based on $\rho\pi$ [10], to which we add two primitives: $\texttt{inform}\langle\cdot\rangle$ and $\texttt{forecast}(\cdot) \triangleright \cdot$. These two primitives are used to interact with a context (which we call the oracle): sending on $\texttt{inform}\langle\cdot\rangle$ stores a process in the context, and receiving from $\texttt{forecast}(\cdot) \triangleright \cdot$ reads the context. The important aspect of the context is that it is preserved through backward reductions.

Introducing these primitives prevents our calculus to be causally consistent: it is possible to revert to the original configuration, but with a different context. Nonetheless, we still have a notion of consistency: a configuration with an uninitialised context can simulate any context. The second part of this work is to characterise this weaker notion of causal consistency.

We finally conclude with a practical application: we implement a distributed Sieve of Eratosthenes.

Contributions. The main contributions of this papers are: (i) a partially reversible calculus $\Omega\rho\pi$, which adds two primitives to save a process across backward reductions; (ii) the definition of weak causal consistency, which relaxes the usual causal consistency; (iii) a proof that $\Omega\rho\pi$ is weakly causally consistent; and (iv) an application of $\Omega\rho\pi$, which illustrates the capabilities and limitations of the calculus.

Outline. The paper is organised as follow: Sect. 2 introduces informally the $\rho\pi$ calculus, on which $\Omega\rho\pi$ is based, and the notion of simulation which is latter used. Section 3 defines the $\Omega\rho\pi$ calculus. We explain how the calculus relates to $\rho\pi$ and we argue that the oracle behaves as expected. Section 4 is a small section devoted to the introduction of weak causal consistency. Section 5 shows our main result, which is that $\Omega\rho\pi$ is weakly causally consistent. Section 6 contains the application example. Finally, Sect. 7 concludes the paper.

2 Preliminary Considerations

In this first section, we present existing work on which the core of this paper is based. In the first subsection, we informally present $\rho\pi$, a reversible variant of the higher-order π-calculus. In the second subsection, we present the notion of simulation, used in multiple works.

2.1 Informal Introduction to $\rho\pi$

The $\rho\pi$ calculus is a reversible higher-order process calculus, first introduced in [10]. In this section, we informally introduce the $\rho\pi$ calculus, and we refer the interested reader to Lanese's paper for a more detailed presentation.

Terms of the $\rho\pi$ calculus (whose syntax is shown in Fig. 1) are composed of a *configuration*, built up from *threads* and *memories*. Threads are themself composed of a *process* (which is similar to processes of the higher-order π-calculus)) and a *tag*, used as an identifier for the process.

$$
\begin{array}{lll}
\mathcal{P}, \mathcal{Q} & ::= 0 \mid X \mid \nu a.P \mid P \mid Q & \textit{Process} \\
& \quad \mid a\langle P \rangle \mid a(X) \triangleright P & \\
\mathcal{M}, \mathcal{N} & ::= 0 \mid \nu u.M \mid M \mid N \mid \kappa : P \mid [\mu; k] & \textit{Configuration} \\
\kappa & ::= k \mid \langle h, \tilde{h} \rangle \cdot k & \textit{Tag} \\
\mu & ::= \kappa_1 : a\langle P \rangle \mid \kappa_2 : a(X) \triangleright Q & \textit{Memory content}
\end{array}
$$

Fig. 1. Syntax of $\rho\pi$

The basic building blocks of processes are the emission of a message P on a name a (written $a\langle P \rangle$) and the reception of a message on a name a (written $a(X) \triangleright P$). When a message (for instance Q) is received, the free occurrences of X in P are replaced by Q, and the resulting process is assigned a new fresh tag. In addition, upon message exchange, a new memory is created, which contains the state of the two processes prior to the message exchange. Informally, the forward reduction rule is the following:

$$k_1 : a\langle P \rangle \mid k_2 : a(X) \triangleright Q \twoheadrightarrow \nu k.k : Q\{^P/_X\} \mid [k_1 : a\langle P \rangle \mid k_2 : a(X) \triangleright Q; k]$$

In this forward rule, notice that the memory contains the tag of the resulting process. This allows the backward execution of the configuration, by replacing a process by the relevant memory:

$$k : P \mid [M; k] \rightsquigarrow M$$

2.2 State Simulation

Given a set of states $\mathbb{S}$ and a reduction relation $\rightarrow$ over states of $\mathbb{S}$, the notion of simulation, originally defined in [17], formalises the statement "any reduction of state S_1 can be done by state S_2".

Definition 1 (Weak simulation). *Given two states $S_1, S_2 \in \mathbb{S}$, a state S_2* simulates *another state S_1 (noted $S_1 \precsim S_2$) if and only if:*

$$\forall S_1' \cdot S_1 \rightarrow S_1' \Rightarrow \exists S_2' \cdot S_2 \rightarrow^\star S_2' \wedge S_1' \precsim S_2'$$

where $\rightarrow^\star$ is the reflexive and transitive closure of $\rightarrow$.

Notice that state simulation is reflexive and transitive.

Remark 1. In Sects. 5 and following of this paper, we use a stronger form of simulation in which $S_2 \rightarrow S_2'$ and $S_1 \rightarrow S_1'$ using the same reduction rule.

3 The $\Omega\rho\pi$ Calculus

In this first section, we present the $\Omega\rho\pi$ calculus, which is built on top of the $\rho\pi$ calculus, itself built on top of the higher-order π calculus (HOπ).

Processes of HOπ are composed of a multiple sequences of message sending and receiving, with possible name restrictions. The semantics of HOπ is that when a process sends a process P and, simultaneously, a process expects to receive a process (variable X) on the same channel, then the second process replaces free occurances of X by P. The $\rho\pi$ calculus decorates HOπ processes in order to allow reverse executions of message communications. The first subsection of this section informally introduces $\rho\pi$.

In $\Omega\rho\pi$, we decorate $\rho\pi$ configurations with a process. We say a decorated $\rho\pi$ configuration is a *context*. We obtain the semantics of contexts by lifting the $\rho\pi$ semantics to contexts. To interact with the context, we add two primitives `inform` and `forecast` (which act as special channels) to write and read the context.

3.1 Syntax

The syntax of the $\Omega\rho\pi$ calculus is given in Fig. 2. Processes are similar to the regular HOπ calculus, with the addition of the `inform` and `forecast` primitives. Configurations, tags and memories are similar to those of $\rho\pi$ (up to the addition of the primitives).

Contrary to $\rho\pi$, configurations are not directly executed: there are embedded in a *context*, which annotates the configuration with a process.

$\mathcal{P}, \mathcal{Q}$	$::= 0 \mid X \mid \nu a.P \mid P \mid Q$		*Process*
	$\mid a\langle P\rangle \mid a(X) \triangleright P$		
	$\mid \texttt{inform}\langle P\rangle \mid \texttt{forecast}(X) \triangleright P$		
$\mathcal{M}, \mathcal{N}$	$::= 0 \mid \nu u.M \mid M \mid N \mid \kappa : P \mid [\mu; k]$		*Configuration*
$\mathcal{C}$	$::= \mathcal{M}\vert_P$		*Context*
κ	$::= k \mid \langle h, \tilde{h}\rangle \cdot k$		*Tag*
μ	$::= \kappa_1 : a\langle P\rangle \mid \kappa_2 : a(X) \triangleright Q$		*Memory content*
	$\mid \kappa : \texttt{forecast}(X) \triangleright P$		

Fig. 2. Syntax of $\Omega\rho\pi$. The differences with $\rho\pi$ are highlighted.

Let $\mathbb{C}$ be the set of contexts, $\mathbb{M}$ the set of configurations and $\mathbb{P}$ the set of processes. We let P, Q and their decorated variants range over $\mathbb{P}$; M, N and their decorated variants range over $\mathbb{M}$ and C and its decorated variants range over $\mathbb{C}$. We say that a context C is *initial* when it does not contain memory.

Names $a, b, \ldots$ take their values from $\mathbb{N}$, which does not contains `forecast` and `inform`.

As in $\rho\pi$, $\tilde{h}$ denotes a vector of keys.

3.2 Semantics

The semantics of $\Omega\rho\pi$ is defined in two successive parts: first we define the semantics of configurations, as a labelled transition system, then we define the semantics of contexts, using the semantics of configurations. Intuitively, the semantics

of configurations acts as the semantics of $\rho\pi$ (up to the required modifications), and the labels of transitions expose the interactions with the oracle. The semantics of contexts simply interprets the labels, updates the oracle accordingly, or constraint the transitions that can be taken by the configurations.

Configuration Semantics. The configuration semantics is defined using two reduction relations (a forward reduction relation $\twoheadrightarrow_c$ and a backward reduction relation $\rightsquigarrow_c$). As usual, we use a structural congruence relation (see Fig. 3), which allows to reorder the processes.

$$M \mid N \equiv N \mid M \qquad (M_1 \mid M_2) \mid M_3 \equiv M_1 \mid (M_2 \mid M_3) \qquad M \mid 0 \equiv M \qquad \nu u.0 \equiv 0$$

$$\nu u.\nu v.M \equiv \nu v.\nu u.M \qquad (\nu u.M) \mid N \equiv \nu u.(M \mid N) \quad u \text{ does not appear free in } N$$

$$M =_\alpha N \Rightarrow M \equiv N \qquad\qquad k : \nu a.P \equiv \nu a.k : P$$

$$k : (\prod_{1 \le i \le n} P_i) \equiv \nu \tilde{h}.(\prod_{1 \le i \le n} \langle h_i, \tilde{h} \rangle \cdot k : P_i) \quad \tilde{h} = \{h_1, \ldots, h_n\}$$

Fig. 3. Structural congruence for $\Omega\rho\pi$ configurations.

We also use an evaluation context (see Fig. 4). Intuitively, an evaluation context is a process with a hole.

$$\mathcal{E} ::= \cdot \quad | \quad \nu u.\mathcal{E} \quad | \quad M \mid \mathcal{E}$$

Fig. 4. Evaluation context

A relation $\mathcal{R}$ over configurations is *evaluation-closed* if it satisfies the two inference rules in Fig. 5.

$$\frac{M \ \mathcal{R} \ N}{\mathcal{E}[M] \ \mathcal{R} \ \mathcal{E}[N]} \qquad\qquad \frac{M \equiv M' \qquad M \ \mathcal{R} \ N \qquad N \equiv N'}{M' \ \mathcal{R} \ N'}$$

Fig. 5. Inference rules for evaluation-closed relations.

The configuration semantics is defined as the least evaluation-closed relation that satisfies the rules in Fig. 6.

Reduction rules are heavily based to $\rho\pi$ rules, with the following differences:

$$\text{(C.Fw)}\ \kappa_1 : a\langle P\rangle \mid \kappa_2 : a(X) \triangleright Q \overset{\tau}{\twoheadrightarrow}_c \nu k.k : Q\{^P/_X\} \mid [\kappa_1 : a\langle P\rangle \mid \kappa_2 : a(X) \triangleright Q; k]$$

$$\text{(C.Bw)}\ \nu k.k : P \mid [Q; k] \overset{\tau}{\rightsquigarrow}_c Q$$

$$\text{(C.Inf)}\ \kappa : \texttt{inform}\langle P\rangle \overset{\overline{P}}{\twoheadrightarrow}_c \nu k.k : 0 \mid [\kappa : \texttt{inform}\langle P\rangle; k]$$

$$\text{(C.For)}\ \kappa : \texttt{forecast}(X) \triangleright Q \overset{P}{\twoheadrightarrow}_c \nu k.k : Q\{^P/_X\} \mid [\kappa : \texttt{forecast}(X) \triangleright Q; k]$$

Fig. 6. Reduction rules of $\Omega\rho\pi$ configuration semantics.

- transitions are labeled: reading a process P from the oracle labels the transition with $\overline{P}$, setting the oracle to P labels it with P, and all other transitions are labeled with the special symbol τ; and
- memories created by the modification of the oracle do not contain a receiver process.

Notice that the primitives seemingly act like regular channels.

The two rules (C.Fw) and (C.Bw) correspond to the forward and backward rules of the regular $\rho\pi$ calculus: the former perform the exchange of a message, and creates an associated memory, and the backward rule replace a process by the corresponding memory. Notice that, since those two rules do not interact with the oracle, their label is τ.

Rule (C.Inf) allows a process to update the oracle. Since we are at the configuration level, and therefore the oracle is not visible here, this rule simply reduces to an empty process, and emits a label $\overline{P}$, where P is the new process stored in the oracle.

On the other hand, with rule (C.For), a process $\texttt{forecast}(X) \triangleright Q$ can read a process P from the oracle, and substitute free occurrences of X by P in Q. Since we are at the configuration level, the oracle is not visible, and the process P read is not constrained at this point. Instead, a label P is emitted, which is then used below to constraint the reduction.

Global Semantics. The global semantics is defined using two reduction relations: a forward reduction relation (noted $\twoheadrightarrow$) and a backward reduction relation (noted $\rightsquigarrow$), defined according to the reduction rules given in Fig. 7.

Silent configuration transitions are simply lifted to the global semantics (rules G.Fw and G.Bw). If the configuration transition exposes a $\overline{Q}$ label, then the context is updated accordingly (rule G.Inform). Notice that we require the newly stored process to simulates the previous one, which captures the intuition of refinement of the stored value[2]. On the other hand, for *forecast* labels (P),

[2] We could generalize this rule by relaxing the constraint that $Q \precsim P$, by introducing a binary relation of processes $\mathcal{R}$ as parameter and requiring that $\langle P, Q\rangle \in \mathcal{R}$, and then instantiating our semantics with $\precsim$ as $\mathcal{R}$ in this paper. However, the implications

$$\text{(G.Fw)}\ \frac{M \overset{\tau}{\twoheadrightarrow}_c N}{M|_P \twoheadrightarrow N|_P} \qquad \text{(G.Bw)}\ \frac{M \overset{\tau}{\rightsquigarrow}_c N}{M|_P \rightsquigarrow N|_P}$$

$$\text{(G.Inform)}\ \frac{M \overset{\overline{Q}}{\twoheadrightarrow}_c N \qquad Q \precsim P}{M|_P \twoheadrightarrow N|_Q}$$

$$\text{(G.Forecast)}\ \frac{M \overset{P}{\twoheadrightarrow}_c N}{M|_P \twoheadrightarrow N|_P}$$

Fig. 7. Reduction rules of $\Omega\rho\pi$ global semantics.

the corresponding configuration transition is allowed only if the label matches the context (rule G.Forecast).

We note $\rightarrow$ the semantics of $\Omega\rho\pi$, defined as $\rightarrow = \twoheadrightarrow \cup \rightsquigarrow$. Also, $\rightarrow^\star$, $\twoheadrightarrow^\star$ and $\rightsquigarrow^\star$ are the transitive and reflective closure of $\rightarrow$, $\twoheadrightarrow$ and $\rightsquigarrow$.

A trace is a sequence of transitions $\sigma = t_1; \ldots; t_n$ with $t_i = C_{i-1} \rightarrow C_i$. When C_0 is initial, we call the trace *initial*. When $t_1; \ldots; t_i$ are the only G.Inform reduction of σ (i.e. if none of $t_{i+1}; \ldots; t_n$ is a G.Inform reduction), we call σ a *forecast sequence* (t_i is called the *last inform transition* of the sequence). A trace that contains only forward (resp. backward) transitions is called *forward trace* (resp. *backward trace*). We note ϵ for an empty trace. Two traces $t_1; \ldots; t_n$ and $t'_1; \ldots; t'_m$ are said coinitial if $t_1 = C_1 \rightarrow C$ and $t'_1 = C_1 \rightarrow C'$ and cofinal if $t_n = C \rightarrow C_f$ and $t'_m = C' \rightarrow C_f$.

Example. The example in Fig. 8 shows an execution of a $\Omega\rho\pi$ context. The initial context is $c\langle P_1\rangle \mid c\langle P_2\rangle$. This context contains two processes to be read on c. The configuration is composed of two threads. The first one (initially with tag k_1) reads the context, and then receives one of the two process on c (due to the non-deterministic semantics, the choice is random), and runs it. Intuitively, it launches one of the process at random. Notice, in particular, that if it rolls back to the initial configuration, an *other* process can be selected during a second attempt. The second process (initially with tag k_2), performs a *definitive* choice. Similarly to the first thread, it selects one of the possible process at random, but contrary to the first thread, it modifies the context to store that choice.

In the example, first k_1 reduces, and first chooses process P_1, which is run (the two first reductions). At this point, if the process rolls back and restarts, it still has the choice (not shown). After, k_2 reads the context, then selects P_2, and finally modifies the oracle (transitions 3, 4 and 5). At this point, the selection

of such generalization are not trivial, in particular with respect to the weak causal consistency result presented latter in this paper. Therefore, for the sake of simplicity, we restrict ourself to the restricted definition.

is definitive[3]. A sequence of backward reduction revert the configuration in its initial state, but with the context modified. Now, when the first two reduction are replayed, k_1 has no choice and selects P_2.

3.3 Oracle Soundness

In this subsection, we argue that the oracle behaves as expected: we expect that, looking at a trace, any reduction G.Forecast that occurs should forecast P, when the previous reduction G.Inform that occurs in the trace sets the oracle with process P, regardless of any backward reduction in between (or with the initial P if no G.Inform transition is taken).

More formally, given a trace σ, for any subtrace $\sigma_{ij} = t_i; \ldots; t_j$ of σ with $t_k = M|_P \rightarrow N|_Q$ the last inform transition of σ_{ij}, for any $t \in t_{k+1}; \ldots; t_j$, t is either G.Fw, G.Bw or G.Forecast with context Q.

To begin with, we show that the context does not change, except during G.Inform transition.

Lemma 1. *Given a trace σ such that the final context is $M|_P$.*
The last G.Inform *reduction (if any) in σ is $N|_Q \twoheadrightarrow N'|_P$.*

Proof. By induction on the length of σ. In the case σ is ϵ, then trivially, $Q = P$ and no reduction occurs.

In the case $\sigma = \sigma'; t$. Let $N|_R$ be the final context of the σ' trace. Let t be $N|_R \rightarrow M|_P$. We proceed by case analysis of the transition t, the cases G.Fw, G.Forecast and G.Bw are trivial. If t is G.Inform, then $t = N|_R \twoheadrightarrow M|_P$, and it is the last G.Inform transition.

Using this fact, we show that G.Forecast reductions read the context set by the previous G.Inform reduction[4].

Lemma 2. *Given a trace $\sigma = t_1; \ldots; t_n$ with $t_i = M|_P \twoheadrightarrow M'|_Q$ being the last* G.Inform *reduction of σ, then for any* G.Forecast *reduction t in the subtrace $t_{i+1}; \ldots; t_n$, $t = N|_Q \twoheadrightarrow N'|_Q$.*

Corollary 1 (Oracle soundness). *Given a trace σ, for any* G.Forecast *reduction $M|_P \twoheadrightarrow M'|_P$, the preceding* G.Inform *transition is $N|_Q \twoheadrightarrow N'|_P$.*

4 Weak Causal Consistency

The $\Omega\rho\pi$ calculus is not causally consistent: it is possible to inform the oracle and then go back to the initial configuration (up to context), which is, obviously, not reachable using only forward reductions (see Fig. 9: the modification of the oracle *happens after* —in Lamport terms— the message exchange).

[3] Notice that, due to the pending $\langle k_5^1, \tilde{k_5} \rangle : c\langle P_1 \rangle$ that remains after the choice, if k_1 reduces at this point, when reading c it could actually receive from this pending process. For the sake of simplicity, we ignore this, since that garbage process is cleaned up when k_2 returns in its initial state.

[4] The proof is trivial. Due to length constraints, we omit it.

$k_1 : \texttt{forecast}(X) \triangleright (X \mid c(Y) \triangleright Y) \mid k_2 : \texttt{forecast}(X) \triangleright (X \mid c(Y) \triangleright \texttt{inform}\langle c\langle Y\rangle\rangle)|_{c\langle P_1\rangle \mid c\langle P_2\rangle}$

$\twoheadrightarrow$ G.Forecast

$\nu k_3.k_3 : c\langle P_1\rangle \mid c\langle P_2\rangle \mid c(Y) \triangleright Y \mid k_2 : \texttt{forecast}(X) \triangleright (X \mid c(Y) \triangleright \texttt{inform}\langle c\langle Y\rangle\rangle)$
$\mid [k_1 : \texttt{forecast}(X) \triangleright (X \mid c(Y) \triangleright Y); k_3]|_{c\langle P_1\rangle \mid c\langle P_2\rangle}$

$\twoheadrightarrow$ G.Fw

$\nu k_3.\nu\tilde{k_3}.\langle k_3^1, \tilde{k_3}\rangle : c\langle P_1\rangle \mid \nu k_4.k_4 : P_2 \mid k_2 : \texttt{forecast}(X) \triangleright (X \mid c(Y) \triangleright \texttt{inform}\langle c\langle Y\rangle\rangle)$
$\mid [k_1 : \texttt{forecast}(X) \triangleright (X \mid c(Y) \triangleright Y); k_3] \mid [\langle k_3^2, \tilde{k_3}\rangle : c\langle P_2\rangle \mid \langle k_3^3, \tilde{k_3}\rangle : c(Y) \triangleright Y; k_4]|_{c\langle P_1\rangle \mid c\langle P_2\rangle}$

$\twoheadrightarrow$ G.Forecast

$\nu k_3.\nu\tilde{k_3}.\langle k_3^1, \tilde{k_3}\rangle : c\langle P_1\rangle \mid \nu k_4.k_4 : P_2 \mid \nu k_5.k_5 : c\langle P_1\rangle \mid c\langle P_2\rangle \mid c(Y) \triangleright \texttt{inform}\langle c\langle Y\rangle\rangle$
$\mid [k_1 : \texttt{forecast}(X) \triangleright (X \mid c(Y) \triangleright Y); k_3] \mid [\langle k_3^2, \tilde{k_3}\rangle : c\langle P_2\rangle \mid \langle k_3^3, \tilde{k_3}\rangle : c(Y) \triangleright Y; k_4]$
$\mid [k_2 : \texttt{forecast}(X) \triangleright (X \mid c(Y) \triangleright \texttt{inform}\langle c\langle Y\rangle\rangle); k_5]|_{c\langle P_1\rangle \mid c\langle P_2\rangle}$

$\twoheadrightarrow$ G.Fw

$\nu k_3.\nu\tilde{k_3}.\langle k_3^1, \tilde{k_3}\rangle : c\langle P_1\rangle \mid \nu k_4.k_4 : P_2 \mid \nu k_5.\nu\tilde{k_5}.\langle k_5^1, \tilde{k_5}\rangle : c\langle P_1\rangle \mid \nu k_6.k_6 : \texttt{inform}\langle c\langle P_2\rangle\rangle$
$\mid [k_1 : \texttt{forecast}(X) \triangleright (X \mid c(Y) \triangleright Y); k_3] \mid [\langle k_3^2, \tilde{k_3}\rangle : c\langle P_2\rangle \mid \langle k_3^3, \tilde{k_3}\rangle : c(Y) \triangleright Y; k_4]$
$\mid [k_2 : \texttt{forecast}(X) \triangleright (X \mid c(Y) \triangleright \texttt{inform}\langle c\langle Y\rangle\rangle); k_5]$
$\mid [\langle k_5^2, k_5\rangle : c\langle P_2\rangle \mid \langle k_5^3, k_5\rangle : c(Y) \triangleright \texttt{inform}\langle c\langle Y\rangle\rangle; k_6]|_{c\langle P_1\rangle \mid c\langle P_2\rangle}$

$\twoheadrightarrow$ G.Inform

$\nu k_3.\nu\tilde{k_3}.\langle k_3^1, \tilde{k_3}\rangle : c\langle P_1\rangle \mid \nu k_4.k_4 : P_2 \mid \nu k_5.\nu\tilde{k_5}.\langle k_5^1, \tilde{k_5}\rangle : c\langle P_1\rangle \mid \nu k_6.\nu k_7.k_7 : 0$
$\mid [k_1 : \texttt{forecast}(X) \triangleright (X \mid c(Y) \triangleright Y); k_3] \mid [\langle k_3^2, \tilde{k_3}\rangle : c\langle P_2\rangle \mid \langle k_3^3, \tilde{k_3}\rangle : c(Y) \triangleright Y; k_4]$
$\mid [k_2 : \texttt{forecast}(X) \triangleright (X \mid c(Y) \triangleright \texttt{inform}\langle c\langle Y\rangle\rangle); k_5]$
$\mid [\langle k_5^2, k_5\rangle : c\langle P_2\rangle \mid \langle k_5^3, k_5\rangle : c(Y) \triangleright \texttt{inform}\langle c\langle Y\rangle\rangle; k_6] \mid [k_6 : \texttt{inform}\langle c\langle P_2\rangle\rangle; k_7]|_{c\langle P_2\rangle}$

$\rightsquigarrow$ G.Bw

$\nu k_3.\nu\tilde{k_3}.\langle k_3^1, \tilde{k_3}\rangle : c\langle P_1\rangle \mid \nu k_4.k_4 : P_2 \mid \nu k_5.\nu\tilde{k_5}.\langle k_5^1, \tilde{k_5}\rangle : c\langle P_1\rangle \mid \nu k_6.k_6 : \texttt{inform}\langle c\langle P_2\rangle\rangle$
$\mid [k_1 : \texttt{forecast}(X) \triangleright (X \mid c(Y) \triangleright Y); k_3] \mid [\langle k_3^2, \tilde{k_3}\rangle : c\langle P_2\rangle \mid \langle k_3^3, \tilde{k_3}\rangle : c(Y) \triangleright Y; k_4]$
$\mid [k_2 : \texttt{forecast}(X) \triangleright (X \mid c(Y) \triangleright \texttt{inform}\langle c\langle Y\rangle\rangle); k_5]$
$\mid [\langle k_5^2, k_5\rangle : c\langle P_2\rangle \mid \langle k_5^3, k_5\rangle : c(Y) \triangleright \texttt{inform}\langle c\langle Y\rangle\rangle; k_6]|_{c\langle P_2\rangle}$

$\rightsquigarrow$ G.Bw

$\nu k_3.\nu\tilde{k_3}.\langle k_3^1, \tilde{k_3}\rangle : c\langle P_1\rangle \mid \nu k_4.k_4 : P_2 \mid \nu k_5.k_5 : c\langle P_1\rangle \mid c\langle P_2\rangle \mid c(Y) \triangleright \texttt{inform}\langle c\langle Y\rangle\rangle$
$\mid [k_1 : \texttt{forecast}(X) \triangleright (X \mid c(Y) \triangleright Y); k_3] \mid [\langle k_3^2, \tilde{k_3}\rangle : c\langle P_2\rangle \mid \langle k_3^3, \tilde{k_3}\rangle : c(Y) \triangleright Y; k_4]$
$\mid [k_2 : \texttt{forecast}(X) \triangleright (X \mid c(Y) \triangleright \texttt{inform}\langle c\langle Y\rangle\rangle); k_5]|_{c\langle P_2\rangle}$

$\rightsquigarrow$ G.Bw

$\nu k_3.\nu\tilde{k_3}.\langle k_3^1, \tilde{k_3}\rangle : c\langle P_1\rangle \mid \nu k_4.k_4 : P_2 \mid k_2 : \texttt{forecast}(X) \triangleright (X \mid c(Y) \triangleright \texttt{inform}\langle c\langle Y\rangle\rangle)$
$\mid [k_1 : \texttt{forecast}(X) \triangleright (X \mid c(Y) \triangleright Y); k_3] \mid [\langle k_3^2, \tilde{k_3}\rangle : c\langle P_2\rangle \mid \langle k_3^3, \tilde{k_3}\rangle : c(Y) \triangleright Y; k_4]|_{c\langle P_2\rangle}$

$\rightsquigarrow$ G.Bw

$\nu k_3.k_3 : c\langle P_1\rangle \mid c\langle P_2\rangle \mid c(Y) \triangleright Y \mid k_2 : \texttt{forecast}(X) \triangleright (X \mid c(Y) \triangleright \texttt{inform}\langle c\langle Y\rangle\rangle)$
$\mid [k_1 : \texttt{forecast}(X) \triangleright (X \mid c(Y) \triangleright Y); k_3]|_{c\langle P_2\rangle}$

$\rightsquigarrow$ G.Bw

$k_1 : \texttt{forecast}(X) \triangleright (X \mid c(Y) \triangleright Y) \mid k_2 : \texttt{forecast}(X) \triangleright (X \mid c(Y) \triangleright \texttt{inform}\langle c\langle Y\rangle\rangle)|_{c\langle P_2\rangle}$

$\twoheadrightarrow$ G.Forecast

$\nu k_3.k_3 : c\langle P_2\rangle \mid c(Y) \triangleright Y \mid k_2 : \texttt{forecast}(X) \triangleright (X \mid c(Y) \triangleright \texttt{inform}\langle c\langle Y\rangle\rangle)$
$\mid [k_1 : \texttt{forecast}(X) \triangleright (X \mid c(Y) \triangleright Y); k_3]|_{c\langle P_2\rangle}$

Fig. 8. Example of a $\Omega\rho\pi$ forward and backward execution. On each line, the part of the term that takes the next transition is coloured in red. The result of the previous transition is coloured in blue on the next line. When the result of the previous transition also takes the next transition, it is coloured in green. (Color figure online)

$$
\begin{aligned}
& k_1 : a\langle P\rangle \mid k_2 : a(X) \triangleright \texttt{inform}\langle X\rangle|_{P \mid Q} \\
\twoheadrightarrow\ & \text{G.Fw} \\
& \nu k_3.k_3 : \texttt{inform}\langle P\rangle \mid [k_1 : a\langle P\rangle \mid k_2 : a(X) \triangleright \texttt{inform}\langle X\rangle; k_3]|_{P \mid Q} \\
\twoheadrightarrow\ & \text{G.Inform} \\
& \nu k_3.\nu k_4.k_4 : 0 \mid [k_3 : \texttt{inform}\langle P\rangle; k_4] \mid [k_1 : a\langle P\rangle \mid k_2 : a(X) \triangleright \texttt{inform}\langle X\rangle; k_3]|_P \\
\rightsquigarrow\ & \text{G.Bw (twice)} \\
& k_1 : a\langle P\rangle \mid k_2 : a(X) \triangleright \texttt{inform}\langle X\rangle|_P
\end{aligned}
$$

Fig. 9. Example of a sequence of reductions which leads to a configuration that can not be reached using only forward reductions.

However, our calculus still exhibits an almost causally consistent behaviour: the embedded configuration is the same and the initial $P \mid Q$ context is *more general* than a specific context P, in the sense that any reduction with a context P can be done by a context $P \mid Q$.

This section formalises this intuition, which we call *weak causal consistency*. In this section, we consider a generic transition system, with a set of states $\mathbb{S}$, equipped with a forward transition relation $\twoheadrightarrow$ and a backward transition relation $\rightsquigarrow$ and a (general) transition relation $\rightarrow = \twoheadrightarrow \cup \rightsquigarrow$.

$\mathcal{R}$-weak Causal Consistency. Given a relation $\mathcal{R} \subseteq \mathbb{S} \times \mathbb{S}$, a reversible system is $\mathcal{R}$-weakly causally consistent is for each state C_f reachable from an initial state C_i, there exists a related state C'_f reachable using only forward transitions. We first define the notion of *initial state* (a state that can only take forward reductions), and we then formalise our notion of weak causal consistency.

Definition 2 (Initial state). *A state C_i is initial (noted $C_i \not\rightsquigarrow$) if and only if there exists no C such that $C_i \rightsquigarrow C$.*

Definition 3 (Weak causal consistency). *A reversible transition system $\Sigma = \langle \mathbb{S}, \twoheadrightarrow, \rightsquigarrow \rangle$ is* weakly causally consistent *(with respect to $\mathcal{R}$) if and only if:*

$$\forall C_i, C_f \in \mathbb{S} \cdot C_i \not\rightsquigarrow \wedge C_i \rightarrow^\star C_f \Rightarrow \exists C'_f \in \mathbb{S} \cdot C_i \twoheadrightarrow^\star C'_f \wedge \langle C'_f, C_f \rangle \in \mathcal{R}$$

This definition is intentionally very broad, depending on the chosen $\mathcal{R}$. In the rest of this paper, we will only consider some particular cases. As we will see in the rest of this paper, we think interesting cases include preorder relations, e.g. simulation relation, or other evaluation-closed relations.

Notice that this definition is close to the definition of reversibility developed by Caires *et al.* in [3] (Definition 3.4).

Remark 2. Notice that if the relation $\mathcal{R}$ we consider is the identity, we fall back on the definition of (strong) causal consistency. Therefore, weak causal consistency is a conservative extension of causal consistency.

5 Weak Causal Consistency of $\Omega\rho\pi$

In this section, we show that $\Omega\rho\pi$ is $\precsim$-weakly causally consistent. We can not show weak causal consistency as causal consistency is usually shown (see for instance the proof of causal consistency of the $\rho\pi$ calculus [10], the details of the proof are shown in Mezzina's thesis [11]), since the loop lemma does not hold in our calculus. Instead, we show the causal consistency in two steps: (i) we show that, if an initial trace σ does not contain G.Inform reductions, then there exists a coinitial and cofinal forward trace $\sigma_{\twoheadrightarrow}$, this is shown by relating this particular trace to an equivalent one in $\rho\pi$ and using the causal consistency of $\rho\pi$; (ii) we show that, for every trace σ, there is a coinitial and cofinal trace σ_s free of G.Inform. A summary of the proof is shown in Fig. 10.

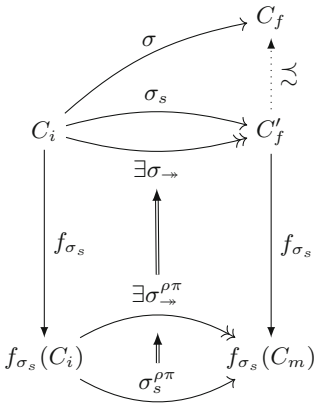

Fig. 10. Summary of the causal consistence proof: for any trace σ (top) from C_i to C_f, it is possible to find a coinitial and cofinal trace σ_s. Aside, we show that any G.Inform free trace (in particular σ_s) of the $\Omega\rho\pi$ calculus can be played in $\rho\pi$ calculus ($\sigma^{\rho\pi}_s$, bottom left). We introduce a function f_{σ_s} to do the conversion. Since $\rho\pi$ is causally consistent, there necessarily exists an equivalent forward sequence $\sigma^{\rho\pi}_{\twoheadrightarrow}$ (middle left), which can finally be played instead of σ_g ($\sigma_{\twoheadrightarrow}$, top left).

5.1 States Simulation

As we have seen in the previous section, weak causal consistency relies on states simulating each other. Hence, we first exhibit some similar states we will rely on in the subsequent sections.

First, a term with P in the context simulates any term composed of the same configuration and a process Q such that $P \precsim Q$ in the context. This is trivial, since $\precsim$ is evaluation closed.

Lemma 3. $\forall M \in \mathbb{M}, P, Q \in \mathbb{P} \cdot P \precsim Q \Rightarrow M|_P \precsim M|_Q$

Also, any $\kappa : \texttt{inform}\langle P\rangle$ simulates $\kappa : 0$. Surprisingly the rule is easy, but not trivial, due to backward reductions.

Lemma 4. $\forall M \in \mathbb{M}, P, S \in \mathbb{P} \cdot M \mid \kappa : \texttt{inform}\langle P\rangle|_S \precsim M \mid \kappa : 0|_S$

Proof. We have to show that, for any C such that $M \mid \kappa : 0|_S \rightarrow C$, there exists C' such that $M \mid \kappa : \texttt{inform}\langle P\rangle|_S \rightarrow C'$.

We proceed by case analysis on the reduction rule used. Only G.Bw and G.Inform are not trivial.

G.Bw: From the premisses of G.Bw and C.Bw, $M \mid \kappa : 0 \equiv \nu k.k : P \mid [Q;k]$. If κ is independent of k, then the result trivially holds.
If $\kappa = \langle h_i, \tilde{h}\rangle \cdot k$, then $M \mid \kappa : 0 \equiv M' \mid (\nu k.\langle h_j, \tilde{h}\rangle \cdot k : R \mid \langle h_i, \tilde{h}\rangle \cdot k : 0) \mid [Q;k] \equiv M' \mid (\nu k.k : R \mid 0) \mid [Q;k]$ (notice that it is possible that $\kappa = k$, in which case $R = 0$). Then the backward reduction that occurs is $M' \mid (\nu k.k : R \mid 0) \mid [Q;k]|_S \rightsquigarrow M' \mid Q|_S$.
Finally, with the same reasoning, we have $M \mid \kappa : \texttt{inform}\langle P\rangle \equiv M' \mid (\nu k.k : R \mid \texttt{inform}\langle P\rangle) \mid [Q;k]$, which can reduce using G.Bw: $M' \mid (\nu k.k : R \mid \texttt{inform}\langle P\rangle) \mid [Q;k]|_S \rightsquigarrow M' \mid Q|_S$. The result holds by reflexivity of $\precsim$.

G.Inform: We suppose the $\texttt{inform}\langle\cdot\rangle$ that reduces is $k : \texttt{inform}\langle P\rangle$. In the case it is an other $\texttt{inform}\langle\cdot\rangle$ in M, the result trivially holds.
From the premisses of G.Inform, $M \mid k : \texttt{inform}\langle P\rangle|_S$ reduces to $M \mid \nu k'.k' : 0 \mid [k : \texttt{inform}\langle P\rangle; k']|_P$ and $P \precsim S$. We have to show that $M \mid \nu k'.k' : 0 \mid [k : \texttt{inform}\langle P\rangle; k']|_P \precsim M \mid k : 0|_S$. Only the case G.Forecast is relevant.
In that case, according to the premisses of G.Forecast and C.For, $M \equiv M' \mid \kappa : \texttt{forecast}(X) \triangleright Q$. Therefore, $M \mid \nu k'.k' : 0 \mid [k : \texttt{inform}\langle P\rangle; k']|_P$ reduces to $M' \mid \nu k''.k'' : Q\{^P/_X\} \mid [\mu;\kappa] \mid \nu k'.k' : 0 \mid [k : \texttt{inform}\langle P\rangle; k']|_P$ and, similarly, $M \mid k : 0|_S$ reduces to $M' \mid \nu k.k : Q\{^S/_X\} \mid [\mu;\kappa] \mid k : 0|_S$. Since $P \precsim S$, $M' \mid \nu k.k : Q\{^P/_X\} \mid [\mu;\kappa] \mid \nu k'.k' : 0 \mid [k : \texttt{inform}\langle P\rangle; k']|_P \precsim M' \mid \nu k.k : Q\{^S/_X\} \mid [\mu;\kappa] \mid k : 0|_S$.

Corollary 2. $\forall M \in \mathbb{M}, P, R, S \in \mathbb{P} \cdot R \precsim S \Rightarrow M \mid \kappa : \texttt{inform}\langle P\rangle|_R \precsim M \mid \kappa : 0|_S$

Proof. From Lemmas 3 and 4, $M \mid \kappa : \texttt{inform}\langle P\rangle|_R \precsim M \mid \kappa : 0|_R \precsim M \mid \kappa : 0|_S$. The result holds by transitivity of $\precsim$.

5.2 Causal Consistency of the Traces Without G.Inform Reductions

When a $\Omega\rho\pi$ trace σ does not contain G.Inform reduction, there is a one-to-one mapping between the global semantics of $\Omega\rho\pi$ contexts, and the configuration semantics of $\Omega\rho\pi$ configurations. To clarify this paragraph, we will only work with the configuration fragment of $\Omega\rho\pi$.

The configuration semantics is analogous to the regular $\rho\pi$ semantics, except for $\texttt{inform}\langle P\rangle$ and $\texttt{forecast}(X) \triangleright Q$ primitives. Encoding the $\texttt{inform}\langle P\rangle$ primitive in $\rho\pi$ is easy: it acts like an oblivious channel and one just need to add

a repeating $\texttt{inform}(X) \triangleright 0$ process (and, anyway, since σ does not contain G.Inform reduction, we could even ignore them).

Encoding the $\texttt{forecast}(X) \triangleright Q$ primitive is almost as simple. Since the trace does not contain G.Inform reduction, the context is constant. Let P be the process it contains. We only need to add enough $\texttt{forecast}\langle P \rangle$. To avoid any problem with the key, we can simply add them under the same key than the forecast. That is, we replace each $\kappa : \texttt{forecast}(X) \triangleright Q$, by $\kappa : \texttt{forecast}(X) \triangleright Q \mid \texttt{forecast}\langle P \rangle$. Note that this replacement also includes occurrences in memories.

Definition 4. *The function* $[\![M|_R]\!]_= [\![M]\!]_R$ *encodes an* $\Omega\rho\pi$ *context into a* $\rho\pi$ *configuration, where* $[\![M]\!]_R$ *is defined in Figs. 11 and 12.*

$$[\![0]\!]_R = 0 \qquad [\![\nu u.M]\!]_R = \nu u.[\![M]\!]_R \qquad [\![M \mid N]\!]_R = [\![M]\!]_R \mid [\![N]\!]_R$$

$$[\![k : P]\!]_R = k : [\![P]\!]_R \qquad [\![[\mu; k]]\!]_R = [[\![\mu]\!]_R; k]$$

Fig. 11. Rules to encode an $\Omega\rho\pi$ configuration into a $\rho\pi$ configuration.

$$[\![0]\!]_R = 0 \qquad [\![X]\!]_R = X \qquad [\![\nu a.P]\!]_R = \nu a.[\![P]\!]_R \qquad [\![P \mid Q]\!]_R = [\![P]\!]_R \mid [\![Q]\!]_R$$

$$[\![a\langle P \rangle]\!]_R = a\langle [\![P]\!]_R \rangle \qquad [\![a(X) \triangleright P]\!]_R = a(X) \triangleright [\![P]\!]_R$$

$$[\![\texttt{inform}\langle P \rangle]\!]_R = \texttt{inform}\langle [\![P]\!]_R \rangle \mid \texttt{inform}(X) \triangleright 0$$

$$[\![\texttt{forecast}(X) \triangleright P]\!]_R = \texttt{forecast}(X) \triangleright [\![P]\!]_R \mid \texttt{forecast}\langle R \rangle$$

Fig. 12. Rules to encode an $\Omega\rho\pi$ process into a $\rho\pi$ process.

Trivially, ignoring G.Inform transitions, an encoded $\Omega\rho\pi$ configuration simulates the original configuration, and an $\Omega\rho\pi$ configuration simulates the forward reductions of its encoded counterparts, using only forward rules:

Lemma 5. *For any* $\Omega\rho\pi$ *contexts* C_1 *and* C_2, $[\![C_1]\!] \twoheadrightarrow [\![C_2]\!] \Rightarrow C_1 \twoheadrightarrow C_2$. *If* $C_1 \twoheadrightarrow C_2$ *without a* G.Inform *reduction, then* $[\![C_1]\!] \twoheadrightarrow [\![C_2]\!]$. *If* $C_1 \rightsquigarrow C_2$, *then* $[\![C_1]\!] \rightsquigarrow [\![C_2]\!]$.

Corollary 3. *$\Omega\rho\pi$, without* G.Inform *reductions, is causally consistent.*

Proof. Suppose $C_i \to^\star C_f$, for $C_i \not\to$. Then there exists a $\rho\pi$ reduction $[\![C_i]\!] \to^\star [\![C_f]\!]$.

Since $\rho\pi$ is causally consistent, there exists a forward reduction $[\![C_i]\!] \twoheadrightarrow^\star [\![C_f]\!]$. Therefore there exists a forward reduction $C_i \twoheadrightarrow^\star C_f$.

5.3 Existence of a Trace Free of G.Inform Reductions

We are given an initial trace $\sigma = t_1; \ldots; t_n$ with C_i the initial configuration. Our goal is to show that there exists a coinitial and cofinal trace $\sigma_f; \sigma_i$ such that σ_f is free of G.Inform reductions. We proceed in two steps: (i) we consider a forecast sequence σ' and we show that we can move the (initial) G.Inform reductions at the end of the trace (see Fig. 13); (ii) we consider an initial trace and we show that we can then move all G.Inform reductions at the end of the trace, by successively pushing the first G.Inform reductions toward the end of the trace.

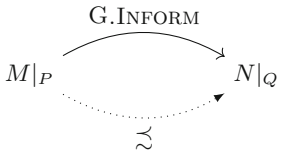

Fig. 13. Illustration of Lemma 6. If a configuration $M|_P$ reduces to $N|_Q$ using a G.Inform reduction, then the initial configuration simulates the final configuration.

Lemma 6 (Inform removal). $\forall M \in \mathbb{M} \cdot M|_P \to N|_Q \Rightarrow M|_P \precsim N|_Q$

Proof. Since the transition changes the context, it is a G.Inform transition. Thus, $M = M' \mid \kappa : \texttt{inform}\langle P \rangle$ and $N = M' \mid \kappa : 0$. From Lemma 4, we have that $M|_P \precsim N|_P$. From the premisses of rule G.Inform, $P \precsim Q$. From Lemma 3, we have that $M|_P \precsim M|_Q$. Finally, from the transitivity of the simulation relation, $M|_P \precsim N|_Q$.

A corollary of this lemma is that for any sequence of reductions from C_i to C_f, it is possible to remove all G.Inform reductions and reach a new C'_f which can simulate C_f.

Corollary 4. *For all initial configuration C_i, for all configuration C_f such that $C_i \to^\star C_f$, there exists a configuration C'_f such that $C_i \to^\star C'_f$ without* G.Inform *reduction and $C'_f \precsim C_f$.*

Proof. Let $\sigma = t_1; \ldots; t_n$ be the trace of the sequence of reduction $C_i \to^\star C_f$.

By induction on the number of G.Inform reductions in σ. The base case (0 G.Inform reduction in σ) follows from the previous section.

For the inductive case, consider there are n G.Inform reductions in σ, let $t_j = C_{j-1} \twoheadrightarrow C_j$ be the first one and $t_k = C_{k-1} \twoheadrightarrow C_k$ the second one. That is, $C_j \to^\star C_{k-1}$ without G.Inform reduction.

From Lemma 6, $C_{j-1} \precsim C_j$, there exists a C'_{k-1} such that $C_{j-1} \to^\star C'_{k-1}$ and $C'_{k-1} \precsim C_{k-1}$. Let σ_1 be the trace of that sequence of reductions. From Remark 1, σ_s does not contain G.Inform reduction.

Also, since $C'_{k-1} \precsim C_{k-1}$, there exists a C'_f such that $C_{k-1} \to^\star C'_f$ and $C'_f \precsim C_f$.

Finally, we have $C_i \to^\star C'_{k-1} \to^\star C'_f$. This sequence contains one less G.Inform reduction. Thus, from the induction hypothesis, there exists C''_f such that $C_i \to^\star C''_f$ without G.Inform reduction and such that $C''_f \precsim C'_f$.

Finally, from the transitivity of simulation, $C''_f \precsim C_f$.

6 Application: Sieve of Eratosthenes

Presentation. In this section, we informally discuss an example application of the oracle introduced in $\Omega\rho\pi$. This example consists in a distributed implementation of the Sieve of Eratosthenes algorithm to find prime numbers. Despite being quite simple, it shows that partial reversibility of $\Omega\rho\pi$ allows a notion of forward progress, which is not the case in pure causally consistent reversible calculi.

For this example, for the sake of simplicity, we add integers, sets and tuples as primitive values of our calculus, and we assume we have usual primitive functions to manipulate those values.

The Sieve of Eratosthenes is a simple algorithm to find prime numbers under a limit l. This algorithm begins with a set of integers from 2 to l, and iterate over each integers i in the set, removing multiples of i. When a fixpoint is reached, the set contains all prime numbers below l.

In our example, we adapt this algorithm for a distributed setting: instead of iterating over i, we take a second set of integer between 2 and $\lceil\sqrt{l}\rceil$, from which concurrent processes select a local i.

For our example, the oracle contains a tuple of two sets: the first is the set of prime candidates, which we call the *sieve* and the second is the set of possible values for i (which we call *values*). Each thread reads this oracle, selects a possible i and removes its multiples from the sieve. Figure 14 shows an implementation of this distributed variant of the Sieve of Eratosthenes. For the sake of conciseness, we only show a single process, but one could choose an arbitrary number of similar processes. Initially, the oracle contains the two sets $\{2,\dots,l\}$ and $\{2,\dots\lceil\sqrt{l}\rceil\}$. Notice that, once a possible i is tested, it is removed from the set of possible values.

$$
\begin{aligned}
k_1 : \ & \texttt{forecast}(\langle \mathit{sieve}, \mathit{values}\rangle)\triangleright \\
& \texttt{let } i \in \mathit{values} \texttt{ in} \\
& \texttt{inform}\langle\langle \mathit{sieve} \setminus \{j | j = k \times i,\ k > 1\}, \mathit{values} \setminus i\rangle\rangle|_{\langle\{2,\dots,l\},\{2,\dots\lceil\sqrt{l}\rceil\}\rangle}
\end{aligned}
$$

Fig. 14. A distributed implementation of the Sieve of Eratosthenes. This implementation has only one process, with tag k_1, but it could contain an arbitrary number of similar processes.

Discussion. The term we show in Fig. 14 is safe, in the sense that when we reach a configuration in which the oracle is $\langle sieve, \emptyset \rangle$, we know that *sieve* contains only prime numbers. However, there is no guarantee that this state is eventually reached. First, as with regular reversible calculi, we can loop in a forward/backward reduction for ever (if the oracle is not updated in between, there is no progress). We ignore this problem since it is common in reversible calculi.

However, the primitives we introduced with the oracle introduce a new problem, which is that forecasts and informs are not atomic: if two receptions are done concurrently, there is a possibility to have a *read–modify–write* inconsistency. The second issue is deeper. To solve it, we would have to introduce standard atomic primitives, such as *compare-and-swap*, to interact with the oracle.

Nonetheless, even with this drawback, this example is interesting. It shows that we have a notion of progress, which can be used to implement standard algorithms for concurrent programming.

7 Conclusion and Future Work

We presented $\Omega\rho\pi$, a reversible calculus in which process can be stored in an *oracle*, which is preserved during backward reductions. This *oracle* is controlled by two primitives which act as two channels `inform` and `forecast`. Until a process is set, any process can be received from the `forecast` primitive (it acts as a random process). Once a process P is sent to the `inform` channel, any message received from the `forecast` channel is that process P (until a new process Q is sent to `inform`) even if the configuration does backward reductions.

Our second main contribution is the definition of a notion of *weak causal consistency*. Weak causal consistency states that for any reachable state, there must exists a similar state reachable using only forward reductions. We think that, in addition to the calculus presented here, this notion of weak causal consistency may be suitable to study other reversible process calculi, for instance those in which backward reductions introduce some garbage which should be ignored.

Future work could improve this paper on two directions.

First, our work can be extended by allowing the process stored in the context to reduce as any other process, following standard $\texttt{HO}\pi$ semantics. Thus, terms would have two parts: a reversible part (in the configuration) and a non-reversible side (in the context). Our `forecast` and `inform` primitives would allow processes to cross the boundary between the two sides. On a longer term, we could imagine allowing reversible and non-reversible processes to communicate via standard channels (and removing `forecast` and `inform` channel, which would become useless). Such approach would result in a reversibility confined to some processes, in a globally non-reversible process (or vice-versa).

On the other hand, we could try to relax the simulation constraint in the premisses of rule G.Inform, which is an important practical limitation. Instead of having a simulation constraint, we could allow a relation $\mathcal{R}$, given as a parameter

of the semantics. With sufficient constraints on this relation (typically reflexivity, transitivity and evaluation closure), we could try to prove the weak causal consistency property of $\Omega\rho\pi$ with respect to such $\mathcal{R}$.

Finally, an underlying aspect of this work is to introduce some notion of *progress* in the context of reversible computation. Usually, reversible computation loses this notion by the very nature of the computation: there is an initial configuration, but no final one, as it is always possible to take backward and forward steps; nor any notion of progress, as anything that is done can be undone. Using oracles and contexts as presented in this paper can be used to reintroduce a notion of progress, for instance by having a convergence criterion on the context.

References

1. Berry, G., Boudol, G.: The chemical abstract machine. In: Proceedings of the 17th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 1990, pp. 81–94 (1989). https://doi.org/10.1145/96709.96717
2. Berry, G., Boudol, G.: The chemical abstract machine. Theoret. Comput. Sci. **96**(1), 217–248 (1992). https://doi.org/10.1016/0304-3975(92)90185-I
3. Caires, L., Ferreira, C., Vieira, H.: A process calculus analysis of compensations. In: Kaklamanis, C., Nielson, F. (eds.) TGC 2008. LNCS, vol. 5474, pp. 87–103. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-00945-7_6
4. Giachino, E., Lanese, I., Mezzina, C.A.: Causal-consistent reversible debugging. In: Gnesi, S., Rensink, A. (eds.) FASE 2014. LNCS, vol. 8411, pp. 370–384. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54804-8_26
5. Kuhn, S., Aman, B., Ciobanu, G., Philippou, A., Psara, K., Ulidowski, I.: Reversibility in chemical reactions. In: Ulidowski, I., Lanese, I., Schultz, U.P., Ferreira, C. (eds.) RC 2020. LNCS, vol. 12070, pp. 151–176. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-47361-7_7
6. Kuhn, S., Ulidowski, I.: A Calculus for Local Reversibility (2016). https://core.ac.uk/display/191241654
7. Kuhn, S., Ulidowski, I.: Local reversibility in a Calculus of Covalent Bonding (2017). https://core.ac.uk/display/328692337?source=3. Publisher: 'Elsevier BV'
8. Lanese, I., Lienhardt, M., Mezzina, C.A., Schmitt, A., Stefani, J.-B.: Concurrent flexible reversibility. In: Felleisen, M., Gardner, P. (eds.) ESOP 2013. LNCS, vol. 7792, pp. 370–390. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-37036-6_21
9. Lanese, I., Mezzina, C.A., Schmitt, A., Stefani, J.-B.: Controlling reversibility in higher-order Pi. In: Katoen, J.-P., König, B. (eds.) CONCUR 2011. LNCS, vol. 6901, pp. 297–311. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-23217-6_20
10. Lanese, I., Mezzina, C.A., Stefani, J.-B.: Reversing higher-order Pi. In: Gastin, P., Laroussinie, F. (eds.) CONCUR 2010. LNCS, vol. 6269, pp. 478–493. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15375-4_33
11. Mezzina, C.A.: Reversing execution in Higher-Order Pi. Theses, Université de Grenoble, February 2012. https://tel.archives-ouvertes.fr/tel-00683964
12. Mezzina, C.A.: On reversibility and broadcast. In: Kari, J., Ulidowski, I. (eds.) RC 2018. LNCS, vol. 11106, pp. 67–83. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-99498-7_5

13. Milner, R. (ed.): A Calculus of Communicating Systems. LNCS, vol. 92. Springer, Heidelberg (1980). https://doi.org/10.1007/3-540-10235-3
14. Perumalla, K.S., Park, A.J.: Reverse computation for rollback-based fault tolerance in large parallel systems. Clust. Comput. **17**(2), 303–313 (2013). https://doi.org/10.1007/s10586-013-0277-4
15. Philippou, A., Psara, K.: Reversible computation in petri nets. In: Kari, J., Ulidowski, I. (eds.) RC 2018. LNCS, vol. 11106, pp. 84–101. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-99498-7_6
16. Phillips, I., Ulidowski, I., Yuen, S.: A reversible process calculus and the modelling of the ERK signalling pathway. In: Glück, R., Yokoyama, T. (eds.) RC 2012. LNCS, vol. 7581, pp. 218–232. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36315-3_18
17. Sangiorgi, D.: Introduction to Bisimulation and Coinduction. University Press, Cambridge (2011). https://doi.org/10.1017/CBO9780511777110
18. Vassor, M., Stefani, J.-B.: Checkpoint/rollback vs causally-consistent reversibility. In: Kari, J., Ulidowski, I. (eds.) RC 2018. LNCS, vol. 11106, pp. 286–303. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-99498-7_20