Towards parametric causal semantics in π-calculus [⋆]

Doriana Medić and Claudio Antares Mezzina

IMT School for Advanced Studies Lucca, Italy

Abstract. In a concurrent setting, causally-consistent reversibility relates causality and reversibility. In this note we overview three causal semantics for π-calculus: two classical and a recent one used for a reversible variant of π-calculus. We show the differences between them via examples, and discuss how to revise the classical one in order to be used as the underlying machinery for a reversible calculus. We propose a reinterpretation of such notions in particular when it comes to silent actions and names extrusion. Our ultimate goal is to devise a general reversible framework parametric into the underlying notion of causality.

1 Introduction

A reversible system is capable of executing both in the forward (normal) direction and in the backward one. In a sequential setting, executing backwards is quite straightforward since there exists only one execution order. In a concurrent system, things are more complex as there exist no clear notion of *last* action: indeed, several independent processes may execute concurrently. Causally-consistent reversibility [7] relates causality and reversibility of a concurrent system in the following way: an action can be reverted provided all its consequences have been reverted.

In Milner's CCS, there exists just one notion of causality: the so-called *structural* which is imposed by the prefixing '.' operator and by synchronizations. An evidence of this is that the two reversible variants of CCS, RCCS [4] and CCSK [9] are shown to be equivalent [8]. When moving to more expressive calculi with name creation and passing like π-calculus things are more complex, since there exist different notions of causality as witnessed by the plethora of causal semantics for π-calculus [2, 5, 1] (just to cite a few of them). In π-calculus, structural causality is determined by the nesting of the prefixes; for example, in process $\overline{b}a.\overline{c}e$ the output on channel c structurally depends on the output on b. Extruding (or opening) a name generates an *object* dependency; for example, in process $\nu a\ (\overline{b}a \mid a(z))$ the input action on a depends on the output on b. Sending a bound name to the context will make all the successive actions using that name dependent on the extruder. There exist different interpretations on how the first extrusion of a name causes the processes using it. In this short note we

[⋆] Partially supported by EU COST IC1405 (Reversible Computation - Extending Horizons of Computing).

will consider the causal semantics introduced by Boreale and Sangiorgi [1], by Degano and Priami [5] and by Cristescu, Krivine and Varacca [2]; and we will discuss on how these notions generate different interpretations of backward moves especially when considering silent actions. Our goal is to devise a reversible calculus parametric with respect the underlying causal semantics, in order to better compare them.

2 Causal semantics by example

Processes in π-calculus are derived from the following syntax:

$$P,\ Q \ \ ::= \mathbf{0} \mid \pi.P \mid \ P \mid Q \ \mid \nu a(P) \qquad\qquad \pi \ \ ::= \overline{b}c \mid b(a) \mid \tau$$

A process can be inactive $\mathbf{0}$, a prefixed process $\pi.P$, the parallel composition $P \mid Q$ and the restriction of a name $\nu a(P)$ meaning that the channel name a is only known in process P. A prefix π denotes the output, input and the silent action, respectively. The semantics of π-calculus is expressed via an LTS and the generated actions/labels are $\mu ::= \overline{b}c \mid b(a) \mid \overline{b}\langle\nu a\rangle \mid \tau$, where $\overline{b}\langle\nu a\rangle$ represents sending of a bound name a. Let us consider the π process $P = \nu a\ (\overline{b}a \mid \overline{c}a \mid a(z))$, where the actions are executed in following order: $P \xrightarrow{\overline{b}a} \xrightarrow{\overline{c}a} \xrightarrow{a(*)} P'$ and $*$ stands for either y or z depending on whether the considered semantics is a late ($* = z$) or an early ($* = y$) one. We now show how the different causal semantics behave.

Boreale et al. Causal information is added to the syntax and semantics of π-calculus in order to track down subject dependencies. Causal processes are of the form $K :: P$, where the cause-set K contains all the causes of the process P. Every visible action is associated with a unique cause k. The τ actions are not observable and do not exhibit causes. The set of the previous causes is noted by K and in our example it is $\emptyset$. Object dependencies can be observed in the labels of the transitions, by looking at the process trace/run. In our example:

$$P \xrightarrow[\emptyset;k_1]{\overline{b}\langle\nu a\rangle} k_1 :: \mathbf{0} \mid \overline{c}a \mid a(z) \xrightarrow[\emptyset;k_2]{\overline{c}a} k_1 :: \mathbf{0} \mid k_2 :: \mathbf{0} \mid a(z) \xrightarrow[\emptyset;k_3]{a(y)} k_1 :: \mathbf{0} \mid k_2 :: \mathbf{0} \mid k_3 :: \mathbf{0}$$

According to [1], the extrusion (the first action) causes every following action in which a is a free name. This implies that both of the actions $\overline{c}a$ and $a(y)$ are caused by $\overline{b}\langle\nu a\rangle$. Hence $\overline{b}\langle\nu a\rangle$ cannot be undone unless $\overline{c}a$ and $a(y)$ are undone. To be able to capture this behaviour we need to be sure that action $\overline{b}\langle\nu a\rangle$ is the last one to be undone, the other two can be undone in any order.

Degano et al. The authors keep track of the structural dependence by specifying which component of the process is performing a move. To uniquely identify the actions, they use labels of the form $\vartheta\mu$ and $\vartheta\langle\|_0\ \vartheta_0\mu_0, \|_1\ \vartheta_1\mu_1\rangle$ where $\vartheta \in \{\|_0, \|_1\}^*$ represents the position of the (sub-)process making the action, and $\mu_0 = b(x)$ iff μ_1 is either $\overline{b}a$ or $\overline{b}\langle\nu a\rangle$, or vice versa. The tag $\|_0$ ($\|_1$) is used to record that the left (right) component in the process is moving. The object (link) dependency can be observed by looking at the process run. In our example:

$$P \xrightarrow{\|_0\|_0\overline{b}\langle\nu a\rangle} \mathbf{0} \mid \overline{c}a \mid a(z) \xrightarrow{\|_0\|_1\overline{c}a} \mathbf{0} \mid \mathbf{0} \mid a(z) \xrightarrow{\|_1 a(z)} \mathbf{0} \mid \mathbf{0} \mid \mathbf{0}$$

The extrusion $\bar{b}\langle\nu a\rangle$ causes every following action that has the name a in the subject position. In this way, the input $a(z)$ is caused by the first action. The output $\bar{c}a$ is neither object dependent on the extrusion, nor concurrent with it. To order these kind of actions, the authors introduced the notion of *temporal precedence* (structural and object). In this way, action $\bar{b}\langle\nu a\rangle$ has object precedence over action $\bar{c}a$ even if they are not causally dependent. From the reversible point of view this notion of causality is similar to [1].

Cristescu et al. A compositional semantics for the reversible π-calculus is introduced in [2]. The information about the past actions are kept into a memory added to every process. A term of the form $m \triangleright P$ represents the reversible process, where memory m is a stack of events and P is process. A memory[1] event $\langle i, k, \alpha\rangle$ contains the identifier i, the contextual cause k and executed action α, respectively. If the action does not have a cause, it is noted with $*$. The indexed restriction νa_Γ behaves as the classical restriction when $\Gamma = \emptyset$, otherwise it is used to keep track of the past scope of the variable. The process that we get from P after the execution of the first two actions is:

$$\nu a_{i,h}(\langle i, *, \bar{b}a\rangle \triangleright \mathbf{0} \mid \langle h, *, \bar{c}a\rangle \triangleright \mathbf{0} \mid a(z))$$

The outputs $\bar{b}a$ and $\bar{c}a$ are associated with the identifiers i and h, respectively. Since these two actions were sending the bound name a to the context, their identifiers are recorded into the process $\nu a_{i,h}$. According to the semantics, these two actions are meant to be executed concurrently and both of them can be seen as the extrusion of the name a. Hence, the input action $a(z)$ can choose its cause between i and h, according to the context. For example, by choosing h, we will get the process:

$$\nu a_{i,h}(\langle i, *, \bar{b}a\rangle \triangleright \mathbf{0} \mid \langle h, *, \bar{c}a\rangle \triangleright \mathbf{0} \mid \langle l, h, a[*/z]\rangle \triangleright \mathbf{0})$$

Since in the memory $\langle l, h, a[*/z]\rangle$, h is saved as a cause of the action l, that force us to undo the action $a(z)$ before the extrusion $\bar{c}a$. The other extrusion ($\bar{b}a$) can be reversed at any time.

2.1 Causalities and silent actions

The semantics introduced in [1, 5] lose information about object dependence when it comes to consider silent actions. Let us consider the process $P = \nu a\ (\bar{b}a \mid \bar{c}a \mid a(z))$ with a context $b(x).\bar{x}y \mid c(w)$ and see what happens.

Boreale et al. By adding a context to the example, all the executed actions will become silent. Since after a synchronization the causes of the two synchronizing processes are merged, and no new causes are created, in order to keep track of them we need to give to processes *initial* and *unique* causes. We have:

$$\nu a\ (k_1 :: \bar{b}a \mid k_2 :: \bar{c}a \mid k_3 :: a(z)) \mid k_4 :: b(x).\bar{x}y \mid k_5 :: c(w) \xrightarrow{\tau}$$
$$\nu a\ (\{k_1, k_4\} :: \mathbf{0} \mid k_2 :: \bar{c}a \mid k_3 :: a(z) \mid \{k_1, k_4\} :: \bar{a}y) \mid k_5 :: c(w) \xrightarrow{\tau}$$
$$\nu a\ (\{k_1, k_4\} :: \mathbf{0} \mid \{k_2, k_5\} :: \mathbf{0} \mid k_3 :: a(z) \mid \{k_1, k_4\} :: \bar{a}y \mid \{k_2, k_5\} :: \mathbf{0}) \xrightarrow{\tau}$$
$$\nu a\ (\{k_1, k_4\} :: \mathbf{0} \mid \{k_2, k_5\} :: \mathbf{0} \mid \{k_3, k_1, k_4\} :: \mathbf{0} \mid \{k_1, k_4, k_3\} :: \mathbf{0} \mid \{k_2, k_5\} :: \mathbf{0}))$$

[1] For the sake of simplicity we are simplifying the information contained into the memory m. We refer to [2] for more details on memories.

We can notice that the (silent) actions are no longer object-dependent (as the example of the previous section). The third τ action is structurally dependent on the first one and we can detect it in the set of the causes $\{k_1, k_4, k_3\}$. If instead of imposing unique causes k_i to the initial processes, we were to use $\emptyset$ (as prescribed by the silent actions) we were unable to observe this fact. In [6], authors proved that the causal information used to support reversibility in $\rho\pi$ is consistent with this notion of causality, if a reduction semantics is considered.

Degano et al. The object dependence after the communication is not needed because through the rule Close the object is localised to the rest of the communicating processes via νa. Hence these three silent actions are not object dependent. The computation with a context will look like:

$$P \mid b(x).\overline{x}y \mid c(w) \xrightarrow{\langle \|_0\|_0\|_0\overline{b}\langle \nu a\rangle, \|_1\|_0 b(a)\rangle} \nu a\ (\mathbf{0} \mid \overline{c}a \mid a(z) \mid \overline{a}y) \mid c(w) \xrightarrow{\langle \|_0\|_0\|_1\overline{c}\langle \nu a\rangle, \|_1\|_1 c(a)\rangle}$$

$$\nu a\ (\mathbf{0} \mid \mathbf{0} \mid a(z) \mid \overline{a}y \mid \mathbf{0}) \xrightarrow{\langle \|_0\|_1 a(z), \|_1\|_0\overline{a}y\rangle} \nu a\ (\mathbf{0} \mid \mathbf{0} \mid \mathbf{0} \mid \mathbf{0} \mid \mathbf{0})$$

From the structural point of view the third action is depending on the first one and we can notice it in the labels. Output $\|_1\|_0\ \overline{a}y$ has a prefix in the first label.

Cristescu et al. After the two synchronizations on the channel b and c, we will get the process:

$$\nu a_\emptyset(\nu a_h(\nu a_{i,h}(\langle i, *, \overline{b}a\rangle \triangleright \mathbf{0} \mid \langle h, *, \overline{c}a\rangle \triangleright \mathbf{0} \mid a(z)) \mid \langle i, *, b[a/x]\rangle \triangleright \overline{x}y) \mid \langle h, *, c[a/w]\rangle \triangleright \mathbf{0})$$

According to the authors, processes $a(z)$ and $\langle i, *, b[a/x]\rangle \triangleright \overline{x}y$ can communicate[2] only if the instantiator of the action $\overline{x}y$ is equal to the cause of the action $a(z)$. Since action i instantiates the name x with the name a, the cause of the input action on the channel a is $k = i$. Considering the silent actions, the causal order does not change, as one property of their causal semantics is that object causality correspond to the subject one.

2.2 Silently regaining information

From these examples we can notice the change in the dependences between the visible actions and the silent ones. When considering just τ actions in [1, 5], there is no object dependency among actions; they are considered as concurrent events. Moreover, information on extruders is lost. To be able to use these causal semantics for a reversible calculus we need to keep track of which process did the extrusion and to record where the binder was before. Moreover, also silent actions have to bring the same causal information as their visible counterpart. But when considering these *modified* semantics we will have that in the structurally equivalent processes, the same actions have different causal order. If we consider our example we will have that $Q = \nu a\ (\overline{b}a \mid \overline{c}a \mid a(z)) \mid b(x).\overline{x}y \mid c(w)$, $Q' = \nu a(\ (\overline{b}a \mid \overline{c}a \mid a(z)) \mid b(x).\overline{x}y \mid c(w))$ with $Q \equiv Q'$, we will have that if $Q \xrightarrow{\tau_b}\xrightarrow{\tau_c}$ executes the synchronization on b and then on c these two silent actions will be object dependent, while in $Q' \xrightarrow{\tau_b}\xrightarrow{\tau_c}$ the two actions are concurrent. This

[2] Memory m is used as a variables store, hence variable x is evaluated into a.

is not the case of [2] since this semantics enjoys some correctness criteria [3] (one of which is that causality is preserved via structural congruence) that the other two semantics do not.

3 Conclusions and future work

We reviewed three notions of causality for π-calculus, two non reversible [1, 5] and one reversible [2], and we showed the differences among them. From the reversible perspective we pointed out the difficulties while using semantics [1, 5] and proposed a reinterpretation of them in particular when it comes to consider silent actions. More in details, information about the extruder have to be maintained, in order to bring back the binder while reverting a bound action. It seems that different data structures can be used to maintain such information; indeed, [2] uses a *set* while in [1, 5] since the first extruder causes the other an *indexed set* should be the ideal data structure. Different data structures will induce a different notion of causality. Currently we are working on developing a general framework for reversible π-calculi parametric with respect to the *data structure* used to save extruders information. Then, depending on the underlying data structure used to instantiate the framework, we could obtain different causal semantics. In this way, once we are able to capture the two known reversible variants of π-calculus [6, 2], it will be simpler to compare them.

Acknowledgments

We are grateful to Daniele Varacca for his useful remarks and suggestions which led to substantial improvements. Furthermore, we thanks Jorge A. Pérez for his comments on a preliminary version of this document.

References

1. M. Boreale and D. Sangiorgi. A fully abstract semantics for causality in the π-calculus. *Acta Inf.*, 35(5):353–400, 1998.
2. I. Cristescu, J. Krivine, and D. Varacca. A compositional semantics for the reversible π-calculus. In *LICS 2013*, pages 388–397, 2013.
3. I. D. Cristescu, J. Krivine, and D. Varacca. Rigid families for CCS and the π-calculus. In *ICTAC 2015*, volume 9399 of *LNCS*, pages 223–240. Springer, 2015.
4. V. Danos and J. Krivine. Reversible communicating systems. In *CONCUR 2004*, volume 3170 of *LNCS*, pages 292–307. Springer, 2004.
5. P. Degano and C. Priami. Non-interleaving semantics for mobile processes. *Theor. Comput. Sci.*, 216(1-2):237–270, 1999.
6. I. Lanese, C. A. Mezzina, and J. Stefani. Reversibility in the higher-order π-calculus. *Theor. Comput. Sci.*, 625:25–84, 2016.
7. I. Lanese, C. A. Mezzina, and F. Tiezzi. Causal-consistent reversibility. *Bulletin of the EATCS*, 114, 2014.
8. D. Medic and C. A. Mezzina. Static VS dynamic reversibility in CCS. In *RC 2016*, pages 36–51, 2016.
9. I. C. C. Phillips and I. Ulidowski. Reversing algebraic process calculi. *J. Log. Algebr. Program.*, 73(1-2):70–96, 2007.