Operational Semantics of Reversibility in Process Algebra

Iain Phillips [1]

Department of Computing
Imperial College London
England

Irek Ulidowski [2]

Department of Computer Science
University of Leicester
England

Abstract

Reversible computation has a growing number of promising application areas such as the modelling of biochemical systems, program debugging and testing, and even programming languages for quantum computing. We discuss reversibility in major process algebras from the point of view of operational semantics. The main difficulty seems to be with the definitions of forward and reverse computation for the dynamic operators, and we confine ourselves to these, leaving the static operators for further work. We consider a solution where predicates in SOS rules play a vital role.

Key words: Reversible computation, process calculi, SOS rules

1 Introduction

Reversible computation has a growing number of promising application areas such as the modelling of biochemical systems, program debugging and testing, and even programming languages for quantum computing. We have been inspired to look at this area by the work of Danos and Krivine on reversible CCS [3,4,5], and the structural approach of Abramsky [1].

We wish to investigate reversibility for algebraic process calculi in the style of CCS [7], with Structural Operational Semantics (SOS) [8] rules. Given a forward labelled transition relation (ltr) $\rightarrow$ we are interested in obtaining a

[1] Email: iccp@doc.ic.ac.uk
[2] Email: iu3@mcs.le.ac.uk

This paper is electronically published in
Electronic Notes in Theoretical Computer Science
URL: www.elsevier.nl/locate/entcs

reverse ltr $\rightsquigarrow$ which is the inverse of $\rightarrow$. This can always be done, but if we just reverse a standard process language we end up with too many possibilities, since processes do not "remember" their past states. Danos and Krivine solve this problem by storing "memories" of past behaviour which are carried along with processes. We would like to see whether we can achieve a similar effect in a more algebraic fashion, by altering the standard rules for operators and possibly introducing auxiliary operators.

The operators of a language like CCS can be divided into the *static* operators, where the operator remains present after a transition, and the *dynamic* operators, where the operator is destroyed by the transition. Dynamic operators are more "forgetful" than static operators. In this note we shall concentrate on reversing dynamic operators, such as CCS prefixing and summation (choice).

We are interested in conditions under which we can make processes constructed with such dynamic operators *unambiguously* reversible, i.e. if P, Q, R are processes and $P \stackrel{a}{\rightarrow} Q \stackrel{b}{\rightsquigarrow} R$, then $b = a$ and $R = P$. We shall see that this goal is attainable for CCS prefixing and summation, among other operators.

In the case of static operators such as parallel composition, unambiguous reversibility is probably too strong a condition; we should aim for some confluence property instead.

We shall proceed rather informally, partly because of space constraints, and partly because this work is still at an early stage.

2 Processes and Predicates

Given a signature Σ we let $T(\Sigma)$ denote the set of closed terms over Σ. We shall assume that we have a "standard" process language consisting of terms over the signature Σ_{S}. We let f range over Σ_{S}. If f is n-ary, its set of arguments is $N_f = \{1, \ldots, n\}$. We say that $T(\Sigma_{\mathrm{S}})$ is the set of *standard* terms. With $T(\Sigma_{\mathrm{S}})$ is associated an ltr $\rightarrow_{\mathrm{S}}$ with labels drawn from a set of actions Act, ranged over by $a, b, \ldots$. This ltr is defined by means of SOS rules.

We shall need to introduce a further set Σ_{A} of *auxiliary* operators. We let $\Sigma_{\mathrm{SA}} = \Sigma_{\mathrm{S}} \cup \Sigma_{\mathrm{A}}$. For terms P in $T(\Sigma_{\mathrm{SA}})$ we define a predicate $\mathsf{std}(P)$, which holds iff $P \in T(\Sigma_{\mathrm{S}})$ (i.e. P is a standard term).

Our aim is to give a procedure for defining a new forward ltr $\rightarrow$, together with a reverse ltr $\rightsquigarrow$ which is the inverse of $\rightarrow$. Standard terms will evolve into nonstandard terms under $\rightarrow$. Moreover, standard terms will have no reverse transitions.

3 Reversing Choice Operators

We now sketch a method for making dynamic operators reversible. We shall confine our attention to a very simple class of dynamic operators, which we

call *choice operators*. These are defined by SOS rules which allow for the selection of arguments, either through the action of one of the arguments (as in CCS summation), or without such action (as in CCS action prefixing or CSP internal choice [6]).

Definition 3.1 An n-ary operator f is a *choice operator* if it is either the inactive constant $\mathbf{0}$ (with no transition rules), or $n \geq 1$ and the SOS rules defining f satisfy the following:

(i) There is a set $D_f \subseteq N_f$ of *permissive* arguments, and for each $d \in D_f$ and each $a \in \mathsf{Act}$ there is a rule

$$\mathsf{SP}_{f,d,a} \quad \frac{X_d \xrightarrow{a}_{\mathrm{S}} X'_d}{f(\overrightarrow{X}) \xrightarrow{a}_{\mathrm{S}} X'_d}$$

(ii) All other rules of f are axioms of the form

$$r \quad \frac{}{f(\overrightarrow{X}) \overset{\mathsf{act}(r)}{\rightarrow}_{\mathrm{S}} X_{\mathsf{ta}(r)}}$$

where $\mathsf{act}(r) \in \mathsf{Act}$ is the *action* of r and $\mathsf{ta}(r) \in N_f$ is the *target argument* of r. We call these rules *choice axioms*.

Let us suppose that all operators in Σ_{S} are choice operators.

We now describe the new reversible rules for a choice operator f. Our aim is that the rules for the reverse transition relation $\rightsquigarrow$ will simply be symmetric versions of the rules for the new forward transition relation $\rightarrow$. The basic idea is to transform the existing rules into static rules.

First we deal with the permissive arguments. Each rule $\mathsf{SP}_{f,d,a}$ is transformed into a new static forward rule

$$\mathsf{FP}_{f,d,a} \quad \frac{X_d \xrightarrow{a} X'_d \quad \{\mathsf{std}(X_i)\}_{i \in N_f \setminus \{d\}}}{f(\overrightarrow{X}) \xrightarrow{a} f(\overrightarrow{X'})}$$

together with a companion reverse rule

$$\mathsf{RP}_{f,d,a} \quad \frac{X_d \overset{a}{\rightsquigarrow} X'_d \quad \{\mathsf{std}(X_i)\}_{i \in N_f \setminus \{d\}}}{f(\overrightarrow{X}) \overset{a}{\rightsquigarrow} f(\overrightarrow{X'})} \ .$$

Here $X'_i = X_i$ for $i \in N_f \setminus \{d\}$. Clearly, the reverse rule is exactly the inverse of the forward rule.

Now we turn to the choice axioms. If we redefined a choice axiom r of f as

$$\frac{\{\mathsf{std}(X_i)\}_{i \in N_f}}{f(\overrightarrow{X}) \overset{\mathsf{act}(r)}{\rightarrow} f(\overrightarrow{X})}$$

then we would allow extra forward transitions—in fact we would create an infinite loop. So instead we employ a new auxiliary operator $f_r \in \Sigma_{\mathrm{A}}$. We

give to f the new forward rule

$$\mathsf{FA}_r \quad \frac{\{\mathsf{std}(X_i)\}_{i\in N_f}}{f(\overrightarrow{X}) \overset{\mathsf{act}(r)}{\rightarrow} f_r(\overrightarrow{X})} \ .$$

We need to ensure that f_r propagates the actions of $X_{\mathsf{ta}(r)}$, leaving other arguments unchanged, and so we give f_r the following forward rule schema:

$$\mathsf{FAX}_{r,a} \quad \frac{X_{\mathsf{ta}(r)} \overset{a}{\rightarrow} X'_{\mathsf{ta}(r)} \quad \{\mathsf{std}(X_i)\}_{i\in N_f\setminus\{\mathsf{ta}(r)\}}}{f_r(\overrightarrow{X}) \overset{a}{\rightarrow} f_r(\overrightarrow{X'})} \qquad (\text{all } a \in \mathsf{Act})$$

The corresponding reverse rules are

$$\mathsf{RA}_r \quad \frac{\{\mathsf{std}(X_i)\}_{i\in N_f}}{f_r(\overrightarrow{X}) \overset{\mathsf{act}(r)}{\rightsquigarrow} f(\overrightarrow{X})}$$

and

$$\mathsf{RAX}_{r,a} \quad \frac{X_{\mathsf{ta}(r)} \overset{a}{\rightsquigarrow} X'_{\mathsf{ta}(r)} \quad \{\mathsf{std}(X_i)\}_{i\in N_f\setminus\{\mathsf{ta}(r)\}}}{f_r(\overrightarrow{X}) \overset{a}{\rightsquigarrow} f_r(\overrightarrow{X'})} \qquad (\text{all } a \in \mathsf{Act})$$

Again, the reverse rules are exactly the inverses of the forward rules.

Note that it is only through the new forward rules FA_r for choice axioms r that nonstandard terms are introduced. We suppose that the new auxiliary operators are all distinct, and that the only operators in Σ_{A} are those already specified.

As computation proceeds in the new ltr, terms keep essentially the same structure, except that during each forward transition exactly one operator f in a term changes to f_r, for some choice axiom r of f. This idea of keeping the structure is present in [2], though that work relates to true concurrency rather than reversible computation.

If a standard term P performs a $\rightarrow$-computation to get to Q, we can retrieve the term arrived at in the corresponding $\rightarrow_{\mathrm{S}}$-computation by "pruning" the parts of Q that would normally be discarded (cf. the forgetful map of [4]). The pruning map $\pi : T(\Sigma_{\mathrm{SA}}) \rightarrow T(\Sigma_{\mathrm{S}})$ is defined as follows:

$$\pi(\mathbf{0}) \overset{\mathrm{df}}{=} \mathbf{0}$$

$$\pi(f(\overrightarrow{P})) \overset{\mathrm{df}}{=} \begin{cases} \pi(P_d) & \text{if } d \in D_f \wedge \forall i \in N_f.\, (\mathsf{std}(P_i) \text{ iff } i \neq d) \\ f(\overrightarrow{P}) & \text{if } \forall i \in N_f.\, \mathsf{std}(P_i) \\ \mathbf{0} & \text{otherwise} \end{cases}$$

$$\pi(f_r(\overrightarrow{P})) \overset{\mathrm{df}}{=} \begin{cases} \pi(P_{\mathsf{ta}(r)}) & \text{if } \forall i \in N_f \setminus \{\mathsf{ta}(r)\}.\, \mathsf{std}(P_i) \\ \mathbf{0} & \text{otherwise} \end{cases}$$

for any choice axiom r with operator f. Clearly, if $\mathsf{std}(P)$ then $\pi(P) = P$.

We state without proof some properties of the new forward and reverse ltrs:

- the forward and reverse ltrs are mutually inverse: for any $P, Q \in T(\Sigma_{\mathrm{SA}})$ and $a \in \mathsf{Act}$, $P \xrightarrow{a} Q$ iff $Q \overset{a}{\rightsquigarrow} P$;
- the new forward ltr is conservative over the standard ltr: for any $P, Q \in T(\Sigma_{\mathrm{SA}})$ and $a \in \mathsf{Act}$, if $P \xrightarrow{a} Q$ then $\pi(P) \xrightarrow{a}_{\mathrm{S}} \pi(Q)$, and if $\pi(P) \xrightarrow{a}_{\mathrm{S}} Q$ then there is $R \in T(\Sigma_{\mathrm{SA}})$ such that $P \xrightarrow{a} R$ and $\pi(R) = Q$;
- the new forward ltr is unambiguously reversible: for any $P, Q, R \in T(\Sigma_{\mathrm{SA}})$ and $a, b \in \mathsf{Act}$, if $P \overset{a}{\rightsquigarrow} Q$ and $P \overset{b}{\rightsquigarrow} R$ then $a = b$ and $Q = R$.

4 Examples

We give some examples of choice operators and their new reversible rules.

CCS summation has the following rule schemas:

$$\frac{X \xrightarrow{a}_{\mathrm{S}} X'}{X + Y \xrightarrow{a}_{\mathrm{S}} X'} \qquad \frac{Y \xrightarrow{a}_{\mathrm{S}} Y'}{X + Y \xrightarrow{a}_{\mathrm{S}} Y'} \quad (\text{all } a \in \mathsf{Act})$$

Both arguments are permissive and there are no choice axioms. We can apply our procedure to turn the standard rules into reversible ones. This gives the following forward and reverse schemas:

$$\frac{X \xrightarrow{a} X' \quad \mathsf{std}(Y)}{X + Y \xrightarrow{a} X' + Y} \qquad \frac{Y \xrightarrow{a} Y' \quad \mathsf{std}(X)}{X + Y \xrightarrow{a} X + Y'} \quad (\text{all } a \in \mathsf{Act})$$

$$\frac{X \overset{a}{\rightsquigarrow} X' \quad \mathsf{std}(Y)}{X + Y \overset{a}{\rightsquigarrow} X' + Y} \qquad \frac{Y \overset{a}{\rightsquigarrow} Y' \quad \mathsf{std}(X)}{X + Y \overset{a}{\rightsquigarrow} X + Y'} \quad (\text{all } a \in \mathsf{Act})$$

We can also handle CCS prefixing:

$$\frac{}{a.X \xrightarrow{a}_{\mathrm{S}} X}$$

Here each operator $a.X$ is equipped with a single choice axiom. For each $a \in \mathsf{Act}$ we introduce the auxiliary operator $\underline{a}$, giving the following new forward and reverse rules:

$$\frac{\mathsf{std}(X)}{a.X \xrightarrow{a} \underline{a}.X} \quad \frac{X \xrightarrow{b} X'}{\underline{a}.X \xrightarrow{b} \underline{a}.X'} \qquad \frac{\mathsf{std}(X)}{\underline{a}.X \overset{a}{\rightsquigarrow} a.X} \quad \frac{X \overset{b}{\rightsquigarrow} X'}{\underline{a}.X \overset{b}{\rightsquigarrow} \underline{a}.X'} \quad (\text{all } b \in \mathsf{Act})$$

A CCS computation such as $a.b.\mathbf{0} + c.d.\mathbf{0} \xrightarrow{a}_{\mathrm{S}} b.\mathbf{0} \xrightarrow{b}_{\mathrm{S}} \mathbf{0}$ becomes $a.b.\mathbf{0} + c.d.\mathbf{0} \xrightarrow{a} \underline{a}.b.\mathbf{0} + c.d.\mathbf{0} \xrightarrow{b} \underline{a}.\underline{b}.\mathbf{0} + c.d.\mathbf{0}$. The new computation can be reversed unambiguously to get $\underline{a}.\underline{b}.\mathbf{0} + c.d.\mathbf{0} \overset{b}{\rightsquigarrow} \overset{a}{\rightsquigarrow} a.b.\mathbf{0} + c.d.\mathbf{0}$.

The internal choice operator of CSP may be defined by two choice axioms using $\tau \in \mathsf{Act}$:

$$\frac{}{X \sqcap Y \xrightarrow{\tau}_{\mathrm{S}} X} \qquad \frac{}{X \sqcap Y \xrightarrow{\tau}_{\mathrm{S}} Y}$$

Neither argument is permissive. We require two auxiliary operators $\sqcap_1$ and $\sqcap_2$. We give the converted rules and the reverse rules for only the first argument X:

$$\frac{\mathsf{std}(X) \quad \mathsf{std}(Y)}{X \sqcap Y \xrightarrow{\tau} X \sqcap_1 Y} \qquad \frac{X \xrightarrow{a} X' \quad \mathsf{std}(Y)}{X \sqcap_1 Y \xrightarrow{a} X' \sqcap_1 Y} \quad (\text{all } a \in \mathsf{Act})$$

$$\frac{\mathsf{std}(X) \quad \mathsf{std}(Y)}{X \sqcap_1 Y \xrightarrow{\tau} X \sqcap Y} \qquad \frac{X \xrightarrow{a} X' \quad \mathsf{std}(Y)}{X \sqcap_1 Y \xrightarrow{a} X' \sqcap_1 Y} \quad (\text{all } a \in \mathsf{Act})$$

5 Conclusions

We have sketched a procedure by which certain dynamic process operators can be made reversible. In further work, we shall extend this to integrate static operators into the picture.

References

[1] Abramsky, S., *A structural approach to reversible computation*, Theoretical Computer Science **347** (2005), pp. 441–464.

[2] Boudol, G. and I. Castellani, *Flow models of distributed computations: three equivalent semantics for CCS*, Information and Computation **114** (1994), pp. 247–314.

[3] Danos, V. and J. Krivine, *Formal molecular biology done in CCS-R*, in: *Proceedings of Bioconcur 2003, Marseille*, 2003.

[4] Danos, V. and J. Krivine, *Reversible communicating systems*, in: *Proceedings of the 15th International Conference on Concurrency Theory (Concur 2004)*, Lecture Notes in Computer Science **3170** (2004), pp. 292–307.

[5] Danos, V. and J. Krivine, *Transactions in RCCS*, in: *Proceedings of the 16th International Conference on Concurrency Theory (Concur 2005)*, Lecture Notes in Computer Science **3653** (2005), pp. 398–412.

[6] Hoare, C., "Communicating Sequential Processes," Prentice-Hall, 1985.

[7] Milner, R., "Communication and Concurrency," Prentice-Hall, 1989.

[8] Plotkin, G., *A structural approach to operational semantics*, Journal of Logic and Algebraic Programming **60-61** (2004), pp. 17–139.