A Simple Categorical Calculus of Interacting Processes

Chad Nester
University of Tartu *
Tartu, Estonia
nester@ut.ee

Niels Voorneveld
Cybernetica AS †
Tallinn, Estonia
niels.voorneveld@cyber.ee

We give a calculus of interacting processes and investigate its categorical structure. Our calculus is parameterised by a multicategory $\mathscr{M}$, the morphisms of which we think of as non-interactive processes. Our calculus augments these non-interactive processes with the ability to interact via a simple message-passing mechanism. More concretely, our calculus consists of a collection $T(\mathscr{M})$ of terms (Figure 1) together with a rewrite relation $\to$ on those terms generated by the rewrites of Figure 2. This rewrite relation is confluent and terminating. Normal forms contain no let-bindings, establishing a sort of cut-elimination result for the calculus. Moreover, terms of $T(\mathscr{M})$ form a virtual double category $[\mathscr{M}]$ when considered modulo the induced convertibility relation $\overset{*}{\leftrightarrow}$. This virtual double category admits a functor $[\![-]\!] : [\mathscr{M}] \to U\left(\ulcorner F(\mathscr{M})\urcorner\right)$ into the underlying virtual double category [1] of the free cornering [2, 3] of the free monoidal category [1] of $\mathscr{M}$. A preprint is available [4].

The process-theoretic interpretation of the calculus is as follows: terms of the form $[v]$ perform no interaction, and represent the same process that v does in $\mathscr{M}$. The term $\mathsf{putR}(v,t)$ represents the process in which we perform v on some inputs, send the result out along the right boundary, and proceed as in t. The term $\mathsf{getR}(x.t)$ represents the process in which we wait for some input along the right boundary, and proceed as in t once we have it. The left-facing terms are interpreted similarly. Finally, the term $\mathsf{let}\ x_1,..,x_n \downarrow (t_1 \mid \cdots \mid t_n)\ \mathsf{in}\ t'$ is the process in which we perform the processes indicated by the t_i and, once they are all finished, use the resulting values as the input to the process t'. Crucially, the processes indicated by the t_i may interact as they are performed by exchanging resources along their left and right boundaries via the get and put constructors.

We give a simple example demonstrating the way in which this calculus models process interaction. Say $\mathscr{M}$ has objects including **Pants**, **Shirt**, **Clothes**, **Pattern** and **Thread**, and has morphisms including:

$$\mathbf{cut} \in \mathscr{M}(\mathbf{Fabric};\mathbf{Pattern}) \qquad \mathbf{sew} \in \mathscr{M}(\mathbf{Pattern},\mathbf{Thread};\mathbf{Shirt}) \qquad \mathbf{pack} \in \mathscr{M}(\mathbf{Pants},\mathbf{Shirt};\mathbf{Clothes})$$

Then we obtain the following sequence of rewrites:

$$\begin{aligned}
&\mathsf{let}\ x,y \downarrow (\mathsf{putR}(\mathbf{cut}(f),[p]) \mid \mathsf{getL}(a.[\mathbf{sew}(a,t)]))\ \mathsf{in}\ [\mathbf{pack}(x,y)]\\
&\to \mathsf{let}\ x,y \downarrow ([p] \mid [\mathbf{sew}(a,t)][\mathbf{cut}(f)/a])\ \mathsf{in}\ [\mathbf{pack}(x,y)]\\
&= \mathsf{let}\ x,y \downarrow ([p] \mid [\mathbf{sew}(\mathbf{cut}(f),t)])\ \mathsf{in}\ [\mathbf{pack}(x,y)]\\
&\to [\mathbf{pack}(x,y)][p,\mathbf{sew}(\mathbf{cut}(f),t)/x,y]\\
&= [\mathbf{pack}(p,\mathbf{sew}(\mathbf{cut}(f),t))]
\end{aligned}$$

In this way, subterms of a message passing term may exchange resources during evaluation.

*Chad Nester was supported by Estonian Research Council Grant PRG2764.
†Niels Voorneveld was supported by Estonian Research Council Grant PRG1780.

Submitted to:
ACT 2026

We indicate morphisms of $\mathscr{M}$ using the sequent calculus notation (see e.g., [5]), so that a derivation of $\Gamma \vdash v : A$ indicates a morphism of $\mathscr{M}(\Gamma;A)$. In all of our derivations any context Γ is assumed not to contain any repeated variables. Terms of $T(\mathscr{M})$ are constructed as follows:

$$\frac{\Gamma \vdash v : A}{\Gamma \Vdash [v] : (I,A,I)}\ \text{SEQ} \qquad \frac{(\Gamma_i \Vdash t_i : (U_{i-1},A_i,U_i))_{i=1}^n \qquad x_1{:}A_1,..,x_n{:}A_n \Vdash t : (V,B,W)}{\Gamma_1,..,\Gamma_n \Vdash \mathsf{let}\ x_1,..,x_n \downarrow (t_1 \mid \cdots \mid t_n)\ \mathsf{in}\ t : (U_0V,B,U_nW)}\ \text{LET}$$

$$\frac{\Delta \Vdash t : (U,B,W) \qquad \Gamma \vdash v : A}{\Delta,\Gamma \Vdash \mathsf{putR}(v,t) : (U,B,A^\circ W)}\ \text{PUTR} \qquad \frac{x : A,\Gamma \Vdash t : (U,B,W)}{\Gamma \Vdash \mathsf{getL}(x.t) : (A^\circ U,B,W)}\ \text{GETL}$$

$$\frac{\Gamma,x : A \Vdash t : (U,B,W)}{\Gamma \Vdash \mathsf{getR}(x.t) : (U,B,A^\bullet W)}\ \text{GETR} \qquad \frac{\Gamma \vdash v : A \qquad \Delta \Vdash t : (U,B,W)}{\Gamma,\Delta \Vdash \mathsf{putL}(v,t) : (A^\bullet U,B,W)}\ \text{PUTL}$$

Figure 1: Term formation rules for $T(\mathscr{M})$.

[R0] $\mathsf{let}\ x_1,..,x_n \downarrow ([v_1] \mid \cdots \mid [v_n])\ \mathsf{in}\ t \to t[v_1,..,v_n/x_1,..,x_n]$

[R1] $\mathsf{let}\ x_1,..,x_n \downarrow (\overline{a} \mid \mathsf{putR}(v,t) \mid \mathsf{getL}(y.s) \mid \overline{b})\ \mathsf{in}\ r \to \mathsf{let}\ x_1,..,x_n \downarrow (\overline{a} \mid t \mid s[v/y] \mid \overline{b})\ \mathsf{in}\ r$

[R2] $\mathsf{let}\ x_1,..,x_n \downarrow (\overline{a} \mid \mathsf{getR}(y.t) \mid \mathsf{putL}(v,s) \mid \overline{b})\ \mathsf{in}\ r \to \mathsf{let}\ x_1,..,x_n \downarrow (\overline{a} \mid t[v/y] \mid s \mid \overline{b})\ \mathsf{in}\ r$

[R3] $\mathsf{let}\ x_1,..,x_n \downarrow (\mathsf{putL}(v,t) \mid \overline{a})\ \mathsf{in}\ r \to \mathsf{putL}(v,\mathsf{let}\ x_1,..,x_n \downarrow (t \mid \overline{a})\ \mathsf{in}\ r)$

[R4] $\mathsf{let}\ x_1,..,x_n \downarrow (\mathsf{getL}(y.t) \mid \overline{a})\ \mathsf{in}\ r \to \mathsf{getL}(y.\mathsf{let}\ x_1,..,x_n \downarrow (t \mid \overline{a})\ \mathsf{in}\ r)$

[R5] $\mathsf{let}\ x_1,..,x_n \downarrow (\overline{a} \mid \mathsf{putR}(v,t))\ \mathsf{in}\ r \to \mathsf{putR}(v,\mathsf{let}\ x_1,..,x_n \downarrow (\overline{a} \mid t)\ \mathsf{in}\ r)$

[R6] $\mathsf{let}\ x_1,..,x_n \downarrow (\overline{a} \mid \mathsf{getR}(y.t))\ \mathsf{in}\ r \to \mathsf{getR}(y.\mathsf{let}\ x_1,..,x_n \downarrow (\overline{a} \mid t)\ \mathsf{in}\ r)$

[R7] $\mathsf{putR}(v,\mathsf{putL}(w,t)) \to \mathsf{putL}(w,\mathsf{putR}(v,t))$

[R8] $\mathsf{putR}(v,\mathsf{getL}(x.t)) \to \mathsf{getL}(x.\mathsf{putR}(v,t))$ when x does not occur in v

[R9] $\mathsf{getR}(x.\mathsf{putL}(v,t)) \to \mathsf{putL}(v,\mathsf{getR}(x.t))$ when x does not occur in v

[R10] $\mathsf{getR}(x.\mathsf{getL}(y.t)) \to \mathsf{getL}(y.\mathsf{getL}(x.t))$

Figure 2: Generating rewrites for $\to$.

References

[1] Tom Leinster (2004): *Higher Operads, Higher Categories*. Cambridge University Press. Preprint version available at arXiv:math.CT/0305049.

[2] Chad Nester (2021): *The Structure of Concurrent Process Histories*. In: *International Conference on Coordination Languages and Models*, Springer, pp. 209–224, doi:10.1007/978-3-030-78142-2_13.

[3] Chad Nester (2023): *Concurrent Process Histories and Resource Transducers*. *Logical Methods in Computer Science* Volume 19, Issue 1, doi:10.46298/lmcs-19(1:7)2023. Available at `https://lmcs.episciences.org/10825`.

[4] Chad Nester & Niels Voorneveld (2026): *A Simple Categorical Calculus of Interacting Processes*. arXiv:2603.18321.

[5] Michael Shulman (2016): *Categorical Logic from a Categorical Point of View*. Technical Report, AARMS Summer School. Available at `https://mikeshulman.github.io/catlog/catlog.pdf`.