What's TyCO, After All?

Uwe Nestmann (BRICS Aalborg, DK)
António Ravara (IST Lisbon, PT)

May 2000

Abstract

$$\begin{array}{rcl}
\text{uniform TyCO} & \overset{!}{\equiv} & \pi_{\mathrm{a}} \text{ with flat variants} \\
\text{uniform TyCO} & \overset{\text{“!”}}{\approx} & \pi_{\mathrm{a}} \text{ with nested variants} \\
\text{non-uniform TyCO} & \overset{!?}{>} & \pi_{\mathrm{a}} \text{ with nested variants}
\end{array}$$

The setting

TyCO is an name-passing process calculus featuring asynchronous communication between concurrent objects via labeled messages carrying names, proposed by Vasconcelos [Vas94]. In comparison to a standard π-calculus [MPW92], TyCO-processes are indeed more like objects in the sense that *every* communication exchanges a structured value, reminiscent of method invocation, consisting of a *name* tagged with a *label.* More recently, Sangiorgi conceived a π-calculus equipped with *labeled values*, called *variants*, which has been brought up as a suitable vehicle for the description, by means of encodings, of the semantics of typed higher-level object calculi [San98].

The two calculi have several commonly defined operators like the inactive process $\mathbf{0}$, parallel composition $P_1|P_2$, and restriction $(\boldsymbol{\nu}a)\,P$. However, their constructs for communication differ. While output in TyCO is always of the form $a!\mathrm{l}\langle b\rangle$, requesting "method" l of "object" a with "argument" b, output in π_{a}^{V} is more liberal in that $a!v$ may send any value v, generated from names b and arbitrary levels of (method) labels, e.g., $v = \mathrm{l}_2\langle \mathrm{l}_1\langle b\rangle\rangle$. Correspondingly, on the receiver's side, TyCO "objects" $a?\{\mathrm{l}_j(x_j){=}P_j\}_{j\in J}$ can only accept requests for "methods", while π_{a}^{V} receivers $a?(x).P$ may accept any value, even unlabeled ones. (For recursive behaviors, both calculi offer replicated receivers.) In order to unpack received values, π_{a}^{V} is equipped with a *variant destructor* $\mathsf{case}\,v\,\mathsf{of}\,\{\mathrm{l}_j(x_j){=}P_j\}_{j\in J}$, which is not known—and actually not necessary—in TyCO.

In both calculi, the exchange of values is disciplined by the use of static type systems that know recursive types $\mu X.T$, channel types $[T]$ and variant types $\{\mathrm{l}_j{:}T_j\}_{j\in J}$. The type systems guarantee that communication on ("object") names is *uniform* in that at any time all the values possibly exchanged over a name have the same type. In other words, in *uniform* calculi, the "interface" of "objects" does not vary over time. The discipline of TyCO is more restrictive, as exhibited by the fact that names always have a type of the shape $[\{\mathrm{l}_j{:}T_j\}_{j\in J}]$, where channel and variant type constructors alternate.

Two Questions of Expressive Power Apparently, TyCO seems to be just an *asynchronous* version of the π-calculus with variants, which we refer to as π_{a}^{V}, but imposing a more restrictive type discipline. Indeed, as we may show formally, TyCO represents a proper subcalculus of π_{a}^{V}.

The obvious question arises, whether this subcalculus is less expressive.

Ultimately, however, we are interested in *non-uniform* TyCO [RL99, RV00, Rav], proposed by Ravara, which admits that "objects" may change their interface over time. It is instructive to try to understand the idea of non-uniformity in TyCO by means of its standard π-calculus-counterpart. The interesting point is that this way of explanation does not work. Non-uniformity in TyCO is based on the fact a communication performs two checks atomically: (i) the check of availability of a message on a channel name; (ii) the check of *current* availability of a method for the requested label mentioned in the message.

This atomicity cannot be directly mimicked in the π-calculus, so it is not clear how to define non-uniform π_{a}^{V}. There, the message needs to be accepted independent of its label, and only afterwards can the current availability of a method be checked. The reader may be reminded of the known discrepancy of π-calculus equivalences due to the early-late distinction of input and communication. In the early case, both the agreement on a communication channel and the actual transfer of the value happen at the same time; in the late case the two aspects of communication are distinguished. However, the atomicity of communication (of a value) and selection (depending on the label of the value) may prevent the first action, the communication from happening when the second would not succeed. In contrast, the early-late distinction just provides more observation points, but the second action (the transfer of a value) is always possible when the first action (the agreement on a channel) has succeeded.

The obvious question arises, whether non-uniform TyCO is more expressive than π_{a}^{V}, and how this expressiveness gap could be captured formally.

Our Contribution

We want to formalize the similarity—and difference—between the two calculi. To this aim, we supply mutual encodings and study their properties.

One Observation We present a simple encoding from TyCO into π_{a}^{V} that makes explicit that (i) the former is a subcalculus of the latter, in that TyCO only knows values with precisely one level of label nesting; (ii) the difference in the atomicity of communication bears no problems (at least in this uniform setting). As a side-effect, the encoding imports to TyCO the theory of π_{a}^{V}.

One "Answer" We give encodings (one local, one non-local, see below) of π_{a}^{V} into TyCO, which shows that nested variants can be encoded using only flat variants, i.e., where there is precisely one level of labeling. We are on the verge of proving that this encoding is fully abstract with respect to *barbed congruence* in the source and *receptive barbed congruence* in the target, which is defined on a non-trivial adaptation to our asynchronous setting of Sangiorgi's theory of receptiveness [San99]. So (assuming that we succeed in finishing our proofs), although TyCO is a proper subcalculus of π_{a}^{V}, no expressive power is lost in the above formal sense.

Our local encoding of π_{a}^{V} into TyCO turns out to be a generalization of an encoding of Lπ (Local π), the asynchronous π-calculus with output-only mobility [Mer00], into πI, the π-calculus with internal-only mobility [San96], that has first been studied by Boreale [Bor98], then by Merro and Sangiorgi [MS98]. In our case, the source language is Local π_{a}^{V}, i.e., Lπ extended with nested labeled values, while the target language is TyCO, as we know by now, is just π_{a}^{V} with only flat variants. In fact, our encoding uses only the subset of TyCO that is both local and internal, which is a promising setting, because it has been shown that only when using this restricted target language the above encoding is known to be fully abstract with respect to barbed congruence [MS98].

While generalizing the encoding Lπ $\to$ πI to labeled values, we recall the idea of encoding the higher-order π-calculus into first-order π-calculus [San93], where values are not transmitted themselves, but only private references to them. We apply the same idea here in that complex values—variants—are not transmitted themselves, but only private references to them. (In fact, already the encoding Lπ $\to$ πI resembles that idea.) Our encoding generalizes the previous encoding of Lπ by the distinction of the transmitted value, using labels n and v: if the value is a name, then mere *forwarders* are created, if the value is a variant, then a local (receptive) resource is created instead that implements a protocol of stepwise variant decomposition through communication, using another label d. According to the protocol, a client may get access to the next inner layer of a nested variant by sending a private d-labeled (linearly receptive) return channel to the resource, in turn receiving on this return channel an access name for a (receptive) resource that represents the inner layer. An accompanying type translation witnesses the fact that our encoding indeed translates the free nesting of π_{a}^{V} into the alternating nesting of TyCO.

In order to also encompass non-local terms, we must enhance our encoding. The reason is that the substitution lemma (a standard statement relating corresponding substitutions in the source and target languages) would not carry over. Also for this encoding, we expect that full abstraction can be proved using receptive barbed congruence.

One Conjecture With respect to the second question of expressive power, we have succeeded in providing two different encodings of non-uniform TyCO, but they do not satisfy standard criteria that are often used to separate two calculi apart regarding their expressiveness: an encoding should be *distributed*, i.e., map parallel composition into the mere parallel composition of its translated components, and it should be *deadlock-* and *divergence-free*; then, we may call it a *good* encoding. One encoding uses a variant destructor extended with an `else` branch; by it it is possibly to simply resend a received value in the case that the method label is not provided by the `case`. However, this encoding immediately introduces divergence to the behavior of translated terms that is not present in the source term. Another encoding proceeds along the lines of an implementation of non-uniform TyCO [RL99], but that encoding cannot be made distributed.

We strongly conjecture that there is no good encoding of non-uniform TyCO into π_{a}^{V}, but we currently do not see how to prove this.

References

[Bor98] M. Boreale. On the Expressiveness of Internal Mobility in Name-Passing Calculi. *Theoretical Computer Science*, 195(2):205–226, 1998. An extended abstract appeared in *Proceedings of CONCUR '96*, LNCS 1119: 163–178.

[Mer00] M. Merro. *Local π: A Model for Concurrent and Distributed Programming Languages*. PhD thesis, Ecole des Mines, France, 2000.

[MPW92] R. Milner, J. Parrow and D. Walker. A Calculus of Mobile Processes, Part I/II. *Information and Computation*, 100:1–77, Sept. 1992.

[MS98] M. Merro and D. Sangiorgi. On Asynchrony in Name-Passing Calculi. In K. G. Larsen, S. Skyum and G. Winskel, eds, *Proceedings of ICALP '98*, volume 1443 of *LNCS*, pages 856–867. Springer, July 1998.

[Rav] A. Ravara. Non-Uniform Concurrent Objects in Process Calculus. PhD thesis, to be submitted in 2000, supervised by Vasco T. Vasconcelos and Amílcar Sernadas, IST, Universidade Técnica de Lisboa, Portugal.

[RL99] A. Ravara and L. Lopes. Programming and Implementation Issues in Non-Uniform TyCO. Technical Report DCC-99-1, Universidade do Porto, 1999.

[RV00] A. Ravara and V. T. Vasconcelos. Typing Non-Uniform Concurrent Objects. In *Proceedings of CONCUR 2000*, LNCS. Springer, 2000. To appear.

[San93] D. Sangiorgi. From π-calculus to Higher-Order π-calculus — and back. In M.-C. Gaudel and J.-P. Jouannaud, eds, *Proceedings of TAPSOFT '93*, volume 668 of *LNCS*, pages 151–166. Springer, 1993.

[San96] D. Sangiorgi. π-Calculus, Internal Mobility and Agent-Passing Calculi. *Theoretical Computer Science*, 167(1,2):235–274, 1996. Also as Rapport de Recherche RR-2539, INRIA Sophia-Antipolis, 1995. Extracts of parts of the material contained in this paper can be found in *Proceedings of TAPSOFT '95* and *ICALP '95*.

[San98] D. Sangiorgi. An Interpretation of Typed Objects into Typed π-Calculus. *Information and Computation*, 143(1):34–73, 1998. Earlier version published as Rapport de Recherche RR-3000, INRIA Sophia-Antipolis, August 1996.

[San99] D. Sangiorgi. The Name Discipline of Uniform Receptiveness. *Theoretical Computer Science*, 221(1–2):457–493, 1999. An abstract appeared in the *Proceedings of ICALP '97*, LNCS 1256, pages 303–313.

[Vas94] V. T. Vasconcelos. *A process-calculus approach to typed concurrent objects*. PhD thesis, Keio University, 1994.