Expressiveness of Recursion, Replication and Scope Mechanisms in Process Calculi

Catuscia Palamidessi
INRIA Futurs and LIX, École Polytechnique
catuscia@lix.polytechnique.fr

Frank D. Valencia
CNRS and LIX, École Polytechnique
fvalenci@lix.polytechnnique.fr

Process calculi such as CCS [Mil89], the π-calculus [MPW92] and Ambients [CG00] are among the most influential formal methods for modelling and analyzing the behaviour of concurrent systems; i.e. systems consisting of multiple computing agents, usually called *processes*, that interact with each other. A common feature of these calculi is that they treat processes much like the λ-calculus treats computable functions. They provide a language in which the structure of *terms* represents the structure of processes together with an *operational semantics* to represent computational steps.

For example, a typical process term is the *parallel composition* $P \mid Q$, which is built from the terms P and Q with the constructor $\mid$ and it represents the process that results from the parallel execution of the processes P and Q. Another typical term is the *restriction* $(\nu x)P$ which represents a process P with a private resource x—e.g., a location, a link, or a name. An operational semantics may dictate that if P can reduce to (or evolve into) P', written $P \longrightarrow P'$, then we can also have the reductions $P \mid Q \longrightarrow P' \mid Q$ and $(\nu x)P \longrightarrow (\nu x)P'$.

Infinite behaviour is ubiquitous in concurrent systems (e.g., browsers, search engines, reservation systems). Hence, it ought to be represented by process terms. Two standard term representations of them are *recursive process expressions* and *replication*.

Recursive process expressions are reminiscent of the recursive expressions used in other areas of computer science, such as for example Functional Programming. They may come in the form $\mu X.P$ where P may have occurrences of X. The process $\mu X.P$ behaves as P with the (free) occurrences of X replaced by $\mu X.P$. Another presentation of recursion is by using *parametric processes* of the form $A(y_1, \ldots, y_n)$ each assumed to have a unique, possibly recursive, *definition* $A(x_1, \ldots, x_n) \stackrel{\texttt{def}}{=} P$ where the x_i's are pairwise distinct, and the intuition is that $A(y_1, \ldots, y_n)$ behaves as its P with each y_i replacing x_i.

Replication, syntactically simpler than recursion, takes the form $!P$ and it is reminiscent of Girard's bang operator; an operator used to express unlimited number of copies of a given resource in linear-logic. Intuitively, $!P$ means $P \mid P \mid \cdots$; an unbounded number of copies of the process P.

Now, it is not uncommon that a given process calculus, originally presented with one form of defining infinite behavior, is later presented with the other. For example, the π-calculus was originally presented with recursive expressions and later with replication. The Ambient calculus was originally presented with replication and later with recursion. This is reasonable as a variant may simplify the presentation of the calculus or be tailored to specific applications.

From the above intuitive description it should be easy to see that $\mu X.(P \mid X)$ expresses the unbounded parallel behaviour of $!P$. It is less clear, however, whether replication can be used to express the unbounded behaviour of $\mu X.P$. In particular, processes that allow for unboundedly many *nested* scopes for a name, as, for example, in $\mu X.(\nu x)(P \mid X)$ which behaves as $(\nu x)(P \mid (\nu x)(P \mid (\nu x)(P \mid \cdots)))$. In fact, the ability of expressing recursive behaviours via replication depends on the particular process calculus under consideration.

In this paper, we shall focus on the case of CCS, the π-calculus, the Ambient calculus, timed concurrent constraint programming (tccp, [NPV02a]) and some calculi for Cryptographic Protocols.

One of the most interesting results that we will recall in our survey is the gap in expressive power between recursion and replication in calculi like CCS [BGZ03] and tccp [NPV02b]. This may look surprising to those acquainted with the π-calculus where this gap does not arise (indeed, in the π-calculus recursion is a derived operation). Our interpretation of this difference is that the link mobility of the π-calculus is a powerful mechanism which makes up for the weakness of replication.

Another surprising result is that if we redefine the recursion rule in CCS so to handle the scope statically, then the above nesting of scope cannot be produced, and recursion loses its additional expressive power [GSV04].

The bottomline message of our survey is that the ability of expressing recursive behaviours via replication in a given process calculus depends on the mechanisms of the calculus to compensate for the restriction of replication as well as on how the scope is managed.

References

[BGZ03] N. Busi, M. Gabbrielli, and G. Zavattaro. Replication vs. recursive definition in Channel Based Calculi. In *Proc. of ICALP 03*, LNCS. Springer-Verlag, 2003.

[CG00] Luca Cardelli and Andrew D. Gordon. Mobile ambients. *Theoretical Computer Science (TCS)*, 240(1):177–213, 2000.

[GSV04] Pablo Giambiagi, Gerardo Schneider, and Frank D. Valencia. On the expressiveness of infinite behavior and name scoping in process cal-

culi. In Igor Walukiewicz, editor, *Proceedings of the 7th International Conference on the Foundations of Software Science and Computation Structures (FOSSACS 2004)*, volume 2987 of *Lecture Notes in Computer Science*, pages 226–240. Springer, 2004.

[Mil89] R. Milner. *Communication and Concurrency*. International Series in Computer Science. Prentice Hall, 1989.

[MPW92] Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes, I and II. *Information and Computation*, 100(1):1–40 & 41–77, 1992. A preliminary version appeared as Technical Reports ECF-LFCS-89-85 and -86, University of Edinburgh, 1989.

[NPV02a] M. Nielsen, C. Palamidessi, and F. Valencia. Temporal concurrent constraint programming: Denotation, logic and applications. *Nordic Journal of Computing*, 9:145–188, 2002. http://www.lix.polytechnique.fr/ catuscia/papers/Ntcc/njc02.ps.

[NPV02b] Mogens Nielsen, Catuscia Palamidessi, and Frank D. Valencia. On the expressive power of temporal concurrent constraint programming languages. In *Proceedings of the Fourth ACM SIGPLAN Conference on Principles and Practice of Declarative Programming*, pages 156–167, New York, October 6–8 2002. ACM Press. http://www.lix.polytechnique.fr/ catuscia/papers/ppdp02.ps.