Relatório Técnico

Núcleo de Computação Eletrônica

Communication Problems in the π-Calculus

M. R. F. Benevides
F. Protti

NCE - 02/2000

Universidade Federal do Rio de Janeiro

Communication Problems in the π-Calculus

M. R. F. Benevides*   F. Protti†

Abstract

In this paper we deal with the notions of deadlock, starvation, and communication errors in the asynchronous polyadic π-calculus. We show that detecting deadlock or starvation in a given specification in π-calculus is an undecidable problem. We also extend the proof of undecidability of the notion of communication errors in the polyadic π-calculus presented in [14].
Keywords: Communicating Systems; Correctness of Concurrent Programs; Process Calculus

1 Introduction

When specifying distributed systems and concurrent programs, one crucial question is to ensure the absence of deadlock, starvation, and communication errors. However, standard mechanisms for detecting *a priori* such situations in a given specification can hardly be found for general cases, since those notions are usually difficult to deal with. In fact, this is the case for the polyadic π-calculus: in this work, we show that detecting deadlock or starvation in a given specification in π-calculus is an undecidable problem. We

*Universidade Federal do Rio de Janeiro, Instituto de Matemática and COPPE, Caixa Postal 68511, 21945-970, Rio de Janeiro, RJ, Brasil. Partially supported by the Conselho Nacional de Desenvolvimento Científico e Tecnológico, CNPq, Brasil. E-mail: `mario@cos.ufrj.br`

†Universidade Federal do Rio de Janeiro, NCE, Caixa Postal 2324, 20001-970, Rio de Janeiro, RJ, Brasil. E-mail: `fabiop@nce.ufrj.br`

also extend the proof of undecidability of the notion of communication errors in the polyadic π-calculus presented in Vasconcelos and Ravara's work [14].

The proof of the undecidability of deadlock detection follows the same strategy employed in [14]: the problem of deciding whether a lambda term has a normal form [6] is reduced to the problem of deciding whether a process is capable of reaching a deadlock situation, by defining a computable function f from λ-terms into processes of the π-calculus, and showing that the decidability of the predicate '$f(M) \in DEAD$' implies the decidability of '$M \downarrow$'. The reduction imposes no restrictions on λ-terms, which may be either open or closed. The definition of f embodies the encoding of the lazy λ-calculus into the the π-calculus described in [10, 12].

The proof of undecidability for communication errors also employs the approach described above. It is in fact an slight extension of the proof in [14], which is based on a reduction that considers closed λ-terms only. Here, open terms are also allowed.

The undecidability of starvation detection follows as a corollary of the undecidability for deadlock.

2 Notions of communication error, deadlock, and starvation in the π-calculus

The concepts and definitions of the asynchronous polyadic π-calculus are used here as usual [4, 7, 9]. Below, we briefly present some definitions.

Definition 1 *The set Π of processes of the polyadic π-calculus is given by the following grammar:*

$$P ::= \bar{a}[\tilde{v}].P \mid a(\tilde{x}).P \mid P|Q \mid \nu x P \mid !a(\tilde{x}).P \mid \mathbf{0}$$

Definition 2 *The set of* action labels *is given by the following grammar, where* $\{\tilde{x}\} \subseteq \{\tilde{v}\} \setminus \{a\}$*:*

$$\alpha ::= \tau \mid a[\tilde{v}] \mid \nu\tilde{x}\bar{a}[\tilde{v}]$$

An internal communication within a process is denoted by τ (*silent action*). The *input action* $a[\tilde{v}]$ denotes the reception on a of the sequence of names $\tilde{v}$. The *output action* $\nu\tilde{x}\bar{a}[\tilde{v}]$ denotes the emission to a of the sequence of names $\tilde{v}$, where some of them are bound. The symbol $\Longrightarrow$ denotes the reflexive and transitive closure of $\xrightarrow{\tau}$, and $\xRightarrow{\alpha}$ denotes $\Longrightarrow\xrightarrow{\alpha}\Longrightarrow$.

In what follows, we present the notions of communication errors, deadlock, and starvation. A process with a communication error, after some silent transitions, is capable of reaching a situation in which there is a discordancy on the number of parameters involved in a communication. A process is capable of deadlock if it may reach a situation in which the computation cannot evolve. Finally, a process is capable of starvation if some part of the system may become precluded from computations.

Definition 3 [13] *The set ERR of π-processes with a communication error is the following set:*

$$ERR = \{P \mid P \Rightarrow \nu\tilde{u}(\bar{a}[v_1, \ldots, v_n].Q \mid a(x_1, \ldots, x_m).R \mid S), a \in \tilde{u}, \text{ and } n \neq m\}.$$

Definition 4 *The set DEAD of π-processes which are capable of deadlock is the following set:*

$$DEAD = \{P \mid P \Rightarrow Q \text{ and } Q \overset{\alpha}{\not\rightarrow}, \text{ for all } \alpha.\}$$

Definition 5 *The set STARV of π-processes which are capable of starvation is the following set:*

$$STARV = \{P \mid P \Rightarrow \nu\tilde{u}(\alpha.A \mid B), \text{ where } \alpha \text{ is } a(\tilde{x}) \text{ or } \bar{a}[\tilde{v}], a \in \tilde{u}, \text{ and } B \overset{\bar{\alpha}}{\not\Rightarrow}\}.$$

3 Encoding the lazy λ-calculus into the π-calculus

The transference of results from the λ-calculus to the π-calculus is achieved by using the encoding of the lazy λ-calculus described below.

Definition 6 [3, 6, 12] *The set Λ^0 of λ-terms is defined by the grammar below, where x and y range over the set of λ-calculus variables:*

$$M ::= x \mid \lambda x.M \mid MN$$

Free variables, closed terms, substitution, alpha-conversion etc. are defined as usual. The reduction relation is $\longrightarrow$, and the reflexive and transitive closure of $\longrightarrow$ is $\Longrightarrow$. We write $M\downarrow$ if M is convergent, and $M\uparrow$ otherwise. In the lazy λ-calculus [1], the redex is always at the left extreme of a term. Milner's encoding of the lazy λ-calculus into the π-calculus is given in the next definition:

Definition 7 [10, 12] *The encoding of the lazy λ-calculus into the π-calculus is given by the following rules:*

$$[\![\lambda x.M]\!]_p \stackrel{\text{def}}{=} p(x,q).[\![M]\!]_q$$

$$[\![x]\!]_p \stackrel{\text{def}}{=} \bar{x}[p]$$

$$[\![MN]\!]_p \stackrel{\text{def}}{=} \nu uv([\![M]\!]_u \mid \bar{u}[v,p] \mid !v(q).[\![N]\!]_q)$$

In what follows, some properties of the encoding are presented.

Lemma 8 [12, 14]

a. *If* $[\![M]\!]_p \Longrightarrow\stackrel{\alpha}{\rightarrow}$ *then* $\alpha = p(x,q)$ *and* $M\downarrow$.

b. *If* $M \uparrow$ *then* $[\![M]\!]_p \stackrel{\alpha}{\not\to}$, *for all* α.

c. *If* $M \Longrightarrow \lambda x.N$ *then* $[\![M]\!]_p \stackrel{p(x,q)}{\Longrightarrow} \gtrsim [\![N]\!]_q$.

d. *If* $M \Longrightarrow x$ *then* $[\![M]\!]_p \stackrel{\bar{x}[q]}{\Longrightarrow} \gtrsim \mathbf{0}$.

e. $[\![M]\!]_p \notin ERR$.

Proof. Clauses a, b, and c are clauses 1, 3, and 4 of Lemma 2 in [14]. Clause d is Lemma 3 in [14]. Clause e is Proposition 5.4, 3 in [12]. □

4 Undecidability proofs

Now we are ready to present the undecidability results. Theorem 9 is a generalization of Theorem 4 in [14]. The remaining results, although not surprising, up to the authors' knowledge are new.

Theorem 9 *The problem '$P \in ERR$' is undecidable.*

Proof. Let us show that if the problem '$P \in ERR$' is decidable, then the problem '$M \downarrow$' is decidable. Let $f : \Lambda^0 \to \Pi$ be such that

$$f(M) \stackrel{\text{def}}{=} \nu p x_1 x_2 \ldots x_n([\![M]\!]_p \mid \overline{p}[] \mid x_1() \mid x_2() \mid \ldots \mid x_n()),$$

where $x_1, x_2, \ldots, x_n$ are the free variables ocurring in M. The function f is clearly computable by the encoding of the lazy λ-calculus into the π-calculus. Now we will prove that $f(M) \notin ERR$ if and only if $M \uparrow$. First, assume that $M \uparrow$. Then, by Lemma 8b, $[\![M]\!]_p \stackrel{\alpha}{\not\to}$, for all α. This means that there is no interaction between $[\![M]\!]_p$ and $\overline{p}[], x_1(), x_2(), \ldots, x_n()$. That is, there is no communication error due to an interaction involving $[\![M]\!]_p$ and the ports above. Since $[\![M]\!]_p \notin ERR$ by Lemma 8e, it follows that $f(M) \notin ERR$. Now, assume that $f(M) \notin ERR$. We claim that:

(a) $M \not\Rightarrow \lambda x.N$. Otherwise, if $M \Rightarrow \lambda x.N$, then $[\![M]\!]_p \stackrel{p(x,q)}{\Longrightarrow}$ by Lemma 8c. That is, there is an interaction between $[\![M]\!]_p$ and $\overline{p}[]$ that causes a communication error.

(b) $M \not\Rightarrow x$. Otherwise, if $M \Rightarrow x$, then $[\![M]\!]_p \stackrel{\overline{x}[q]}{\Longrightarrow}$, by Lemma 8d. This means that there is an interaction between $[\![M]\!]_p$ and $x()$ leading to a communication error.

By a) and b), we conclude that $M \Rightarrow MN$. Therefore, $M \uparrow$. □

Theorem 10 *The problem '$P \in DEAD$' is undecidable.*

Proof. Let us show that if the problem '$P \in DEAD$' is decidable, then the problem '$M \downarrow$' is decidable. Let $f : \Lambda^0 \to \Pi$ be such that

$$f(M) \stackrel{\text{def}}{=} \nu p x_1 x_2 \ldots x_n([\![M]\!]_p \mid \overline{p}[y,w].P_0 \mid x_1(p_1).P_1 \mid x_2(p_2).P_2 \mid \ldots \mid x_n(p_n).P_n),$$

where: $x_1, x_2, \ldots, x_n$ are the free variables ocurring in M; $y, w, p_1, \ldots, p_n$ are new names; and $P_i = \nu a_i(A_i \mid \bar{A}_i)$, where $A_i = a_i().A_i$ and $\bar{A}_i = \bar{a}_i[].\bar{A}_i$. The function f is clearly computable by the encoding of the lazy λ-calculus into the π-calculus. Now we will prove that $f(M) \notin DEAD$ if and only if $M \downarrow$. Assume that $f(M) \notin DEAD$. Then there is α such that $[\![M]\!]_p \stackrel{\alpha}{\Longrightarrow}$. Thus, by Lemma 8b, $M \downarrow$. On the other hand, assume that $M \downarrow$ and $f(M) \in DEAD$. Under these assumptions, we have that:

(a) $M \not\Rightarrow \lambda x.N$. Otherwise, if $M \Rightarrow \lambda x.N$, then $[\![M]\!]_p \stackrel{p(x,q)}{\Longrightarrow}$ by Lemma 8c. That is, there is a possible action for $f(M)$, and after this action the computation goes on, contradicting the assumption.

(b) $M \not\Rightarrow x$. Otherwise, if $M \Rightarrow x$, then $[\![M]\!]_p \stackrel{\bar{x}[p]}{\Rightarrow}$, by Lemma 8d. Again, this means that there is a possible action for $f(M)$, and after this action the computation goes on, contradicting the assumption.

By a) and b), we conclude that $M \Rightarrow MN$. Therefore, $M \uparrow$, a contradiction. □

Theorem 11 *The problem '$P \in STARV$' is undecidable.*

Proof. Let us show that if the problem '$P \in STARV$' is decidable, then the problem '$P \in DEAD$' is decidable. Let $f : \Pi \rightarrow \Pi$ be such that

$$f(P) = \nu a_1 \ldots a_n (P \mid A_1 \mid \bar{A}_1 \mid \ldots \mid A_n \mid \bar{A}_n)$$

where $a_1 \ldots a_n$ are the free names in P, $A_i = a_i().A_i$, and $\bar{A}_i = \bar{a}_i[].\bar{A}_i$, $1 \leq i \leq n$. The function f is clearly computable, since $f(P)$ can be constructed in finite time by examining the definition of P. It remains to prove that $f(P) \in STARV$ if and only if $P \in DEAD$. If $P \in DEAD$, then process P is clearly precluded from executing in $f(P)$, that is $f(P) \in STARV$. On the other hand, if $P \notin DEAD$, then $f(P)$ will never reach a state where some of its parts cannot execute, that is, $f(P) \notin STARV$. □

References

[1] S. Abramsky. The lazy lambda calculus. *Research Topics in Functional Programming*, pp. 65–116. Addison-Wesley, 1989.

[2] S. Arun-Kumar and M. Hennessy. An efficiency preorder of processes. *Acta Informatica* 29(8) (1992) 737–760.

[3] H. Barendregt. *The Lambda Calculus: Its Syntax and Semantics. Studies in Logic*, vol. 103. North Holland, 1984. Revised edition.

[4] G. Boudol. Asynchrony and the *pi*-calculus. Technical Report RR-1702, INRIA Sophia-Antipolis, 1992.

[5] N. Cutland. *Computability: An Introduction to Recursive Function Theory.* Cambridge University Press, 1980.

[6] J. R. Hindley and J. P. Seldin. *Introduction to Combinators and λ-Calculus.* Cambridge University Press, 1986, p. 63.

[7] K. Honda and M. Tokoro. An object calculus for asynchronous communication. *5th European Conference on Object-Oriented Programming,* LNCS 512 (1991) 141–162. Springer-Verlag.

[8] R. Milner. *Communication and Concurrency.* Prentice-Hall, London, 1989.

[9] R. Milner. The polyadic π-calculus: a tutorial. In F. L. Bauer, W. Brauer, and H. Schwichtenberg, eds. *Logic and Algebra of Specification,* Springer-Verlag, 1993.

[10] R. Milner. Functions as processes. *Journal of Mathematical Structures in Computer Science* 2(2) (1992) 119–141.

[11] D. Sangiorgi and R. Milner. The problem of "Weak bisimulation up to". *3rd International Conference on Concurrency Theory,* LNCS 630 7 (1994) 331–336.

[12] D. Sangiorgi. Lazy functions and mobile processes. Technical Report RR-2515, INRIA, Sophia-Antipolis, 1995.

[13] V. T. Vasconcelos and K. Honda. Principal typing schemes in a polyadic π-calculus. *4th International Conference on Concurrency Theory,* LNCS 715 (1993) 524–538. Springer-Verlag.

[14] V. T. Vasconcelos and A. Ravara. Communication errors in the π-calculus are undecidable. *Information Processing Letters* 71(5-6) (1999), 229–233.