ENSURING TERMINATION BY TYPABILITY

Yuxin Deng[1] and Davide Sangiorgi[2]
[1]*INRIA and University Paris 7,* [2]*University of Bologna*

Abstract A term terminates if all its reduction sequences are of nite length. We show four type systems that ensure termination of well-typed π-calculus processes. The systems are obtained by successive re nements of the types of the simply typed π-calculus. For all (but one of) the type systems we also present upper bounds to the number of steps well-typed processes take to terminate. The termination proofs use techniques from term rewriting systems.

We show the usefulness of the type systems on some non-trivial examples: the encodings of primitive recursive functions, the protocol for encoding separate choice in terms of parallel composition, a symbol table implemented as a dynamic chain of cells.

Keywords: Concurrency, the π-calculus, type system, termination

1. Introduction

A term terminates if all its reduction sequences are of nite length. As far as programming languages are concerned, termination means that computation in programs will eventually stop. In computer science termination has been extensively investigated in term rewriting systems [5, 3] and -calculi [7, 2] (where strong normalization is a synonym more commonly used). Termination has also been discussed in process calculi, notably the -calculus [12, 17], a formalism widely used to address issues related to concurrency.

Indeed, termination is interesting in concurrency. For instance, if we interrogate a process, we may want to know that an answer is eventually produced (termination alone does not guarantee this, but termination would be the main ingredient in a proof). Similarly, when we load an applet we would like to know that the applet will not run for ever on our machine, possibly absorbing all the computing resources (a 'denial of service' attack). In general, if the lifetime of a process can be in nite, we may want to know that the process does not remain alive simply because of non-terminating internal activity, and that, therefore, the process will eventually accept interactions with the environment.

Languages of terminating processes are proposed in [19] and [16]. In both cases, the proofs of termination make use of logical relations, a well-known