Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 242 (2009) 3–17
www.elsevier.com/locate/entcs

On Convergence-sensitive Bisimulation and the Embedding of CCS in Timed CCS

Roberto M. Amadio[1,2]

Université Paris Diderot

Abstract

We propose a notion of convergence-sensitive bisimulation that is built just over the notions of (internal) reduction and of (static) context. In the framework of timed CCS, we characterise this notion of 'contextual' bisimulation via the usual labelled transition system. We also remark that it provides a suitable semantic framework for a fully abstract embedding of untimed processes into timed ones. Finally, we show that the notion can be refined to include sensitivity to divergence.

Keywords: Bisimulation, convergence, timed CCS.

1 Introduction

The main motivation for this work is to build a notion of convergence-sensitive bisimulation from first principles, namely from the notions of *internal reduction* and of (static) *context*. A secondary motivation is to understand how asynchronous/untimed behaviours can be embedded fully abstractly into synchronous/timed ones. Because the notion of convergence is very much connected to the notion of time, it seems that a convergence-sensitive bisimulation should find a natural application in a synchronous/timed context. Thus, in a nutshell, we are looking for an 'intuitive' semantic framework that spans both untimed/asynchronous and timed/synchronous models.

For the sake of simplicity we will place our discussion in the well-known framework of (timed) CCS. We assume the reader is familiar with CCS [10]. Timed CCS (TCCS) is a 'timed' version of CCS whose basic principle is that *time passes exactly when no internal computation is possible*. This notion of 'time' is inspired by

[1] PPS, UMR-7126. Work partially supported by ANR-06-SETI-010-02.
[2] Email: amadio@pps.jussieu.fr

doi:10.1016/j.entcs.2009.06.010

early work on the ESTEREL synchronous language [3], and it has been formalised in various dialects of CCS [14,12,6]. Here we shall follow the formalisation in [6].

As in CCS, one models the internal computation with an action τ while the passage of (discrete) time is represented by an action tick that implicitly synchronizes all the processes and moves the computation to the next instant. [3]

In this framework, the basic principle we mentioned is formalised as follows:

$$P \xrightarrow{\mathsf{tick}} \cdot \text{ iff } P \not\xrightarrow{\tau} \cdot$$

where we write $P \xrightarrow{\mu} \cdot$ if P can perform an action μ. TCCS is designed so that if P is a process built with the usual CCS operators and P cannot perform τ actions then $P \xrightarrow{\mathsf{tick}} P$. In other terms, CCS processes are *time insensitive*. To compensate for this property, one introduces a new binary operator $P \triangleright Q$, called *else_next*, that tries to run P in the current instant and, if it fails, runs Q in the following instant.

We assume countably many names $a, b, \ldots$ For each name a there is a communication action a and a co-action $\overline{a}$. We denote with $\alpha, \beta, \ldots$ the usual CCS actions which are composed of either an internal action τ or of a communication action $a, \overline{a}, \ldots$. We denote with $\mu, \mu', \ldots$ either an action α or the distinct action tick.

The TCCS processes $P, Q, \ldots$ are specified by the following grammar

$$P ::= 0 \mid a.P \mid P + P \mid P \mid P \mid \nu a\ P \mid A(\mathbf{a}) \mid P \triangleright P\ .$$

We denote with $fn(P)$ the names free in P. We adopt the usual convention that for each thread identifier A there is a unique defining equation $A(\mathbf{b}) = P$ where the parameters $\mathbf{b}$ include the names in $fn(P)$. The related labelled transition system is specified in table 1.

Say that a process is a CCS process if it does not contain the *else_next* operator. The reader can easily verify that:

(1) $P \xrightarrow{\mathsf{tick}} \cdot$ if and only if $P \not\xrightarrow{\tau} \cdot$.

(2) If $P \xrightarrow{\mathsf{tick}} Q_i$ for $i = 1, 2$ then $Q_1 = Q_2$. One says that the passage of time is *deterministic*.

(3) If P is a CCS process and $P \xrightarrow{\mathsf{tick}} Q$ then $P = Q$. Hence CCS processes are closed under labelled transitions.

It will be convenient to write $\tau.P$ for $\nu a\ (a.P \mid \overline{a}.0)$ where $a \notin fn(P)$, $\mathsf{tick}.P$ for $0 \triangleright P$, and Ω for the diverging process $\tau.\tau.\ldots$.

Remark 1.1 (1) In the labelled transition system in table 1, the definition of the tick action relies on the τ action and the latter relies on the communication actions $a, a', \ldots$. There is a well known method to give a direct definition of the τ action that does not refer to the communication actions. Namely, one defines (internal) reduction rules such as $(a.P + Q \mid \overline{a}.P' + Q') \to (P \mid P')$ which are applied modulo a suitable structural equivalence.

[3] There seems to be no standard terminology for this action. It is called ϵ in [14], χ in [12], σ in [6], and sometimes 'next' in 'synchronous' languages à la ESTEREL [2].