Testing Equivalence for Mobile Processes
(Extended Abstract)

Michele Boreale and Rocco De Nicola

Università di Roma "La Sapienza"
Dipartimento di Scienze dell'Informazione
Via Salaria 113, I-00198 Roma (Italy)

Abstract. The impact of applying the testing approach to a calculus of processes with a dynamically changing structure is investigated. A proof system for the finite fragment of the calculus is introduced which consists of two groups of laws: those for strong observational equivalence and those needed to deal with \( \tau \) actions. Soundness and completeness w.r.t. a testing preorder are shown. A fully abstract denotational model for the language is presented which relies on the existence of normal forms for processes.

1. Introduction

Process Algebras ([Mil89], [Hoa85], [BK89]) are generally recognized as a good formalism for describing and studying properties of distributed concurrent systems. A process algebra is often defined by specifying its syntax and the transitional semantics of its terms by means of Structured Operational Semantics [Plo81]. By now, this approach has become a standard tool for specifying basic semantics of process algebras, but it was early recognized that it does not yield extensional accounts of processes. Thus, techniques have been developed to abstract from unwanted details in systems descriptions. Many of these techniques are based on behavioural equivalences; two terms are identified if and only if no observer can notice any difference between their external behaviours.

Process description languages, such as CCS, have been (and are) thoroughly studied using equivalence notions based on bisimulations (see e.g. [Mil89]) or on testing (see e.g. [Hen88]). Complete axiomatizations have been put forward which are of fundamental importance for manipulating process expressions by means of simple axioms and inference rules and constitute the theoretical basis for a class of verification tools (see e.g. [DIN90, Hui92]).

Almost all process algebras which have been considered permit describing only systems whose subparts can interact by performing pure synchronizations. Lately, a language with value-passing has been investigated and a complete axiomatization of a testing-based equivalence has been performed by provided for it [HI91]. A further step toward improving the descriptive power of process algebras has been performed by adding

Work partially supported by “Progetto Finalizzato Sistemi Informatici e Calcolo Parallelo” of CNR, contract n. 91.00894.69 and by Istituto di Elaborazione dell’Informazione of CNR, Pisa.
primitives for expressing exchange of more complex objects such as channel names or processes themselves, see e.g. ECCS of [EN86] and CHOCS, [Tho90]. Recently an extension of CCS called \( \pi \)-calculus has been put forward [MPW89] that permits describing systems whose linking configuration may dynamically change by exchanging channel names (in passing, we note that this permits simulating process passing). For these higher order languages only (strong-)bisimulation-based theories have been investigated; and no weak (i.e. abstracting from internal moves) behavioural equivalence has been considered and e.g. equipped with a complete axiomatization.

The aim of this paper is therefore twofold: to investigate the applicability of the testing approach to higher order process description languages and to provide an axiomatization of a weak equivalence for \( \pi \)-calculus. The new equivalence is defined by following the general approach of [DH84, DeN87, Hen88]. It requires formally defining a set of observers, a way of observing and a general criterion for interpreting the result of observations. If we call \( P \) the set of systems we want to experiment upon, to apply the general setting, we need to define a set of observers, say \( O \), and explain the evolution of pairs such as \( <p,o> \in P \times O \) which will represent the interactions between \( p \) and \( o \). Interactions may be failing or successful depending on whether particular states (which report success) are reached. For specific process \( p \) and observer \( o \), one might be interested in knowing whether a successful interaction does exists, i.e. whether \( p \) may satisfy \( o \), or whether all interactions are successful, i.e. whether \( p \) must satisfy \( o \). Two testing preorders over \( P \), naturally arise, associated with each of the above points of view:

- \( p \preceq_m q \) if and only if \( \forall o \in O : p \) may satisfy \( o \) implies \( q \) may satisfy \( o \);
- \( p \preceq_M q \) if and only if \( \forall o \in O : p \) must satisfy \( o \) implies \( q \) must satisfy \( o \).

These preorders can be naturally combined to get a third one:

- \( p \preceq q \) if and only if \( \forall o \in O : p \preceq_m q \) and \( p \preceq_M q \).

In [DH84] this machinery is applied to CCS and gives rise to three testing preorders which have simple axiomatic and denotational characterization communication scheme (bound names of input actions are instantiated only when the COM rule is actually used, see rules INPUT, OUTPUT, COM and CLOSE of the transitional semantics). Indeed, whereas the so-called early instantiation scheme of CCS with value passing ([Mi88]) permits easily determining the pairs of complementary sequences of visible actions which make an interaction possible, the same is not true for late instantiation.

Additional complications are induced by the fact that, in \( \pi \)-calculus, beside distinguishing between input and output actions, two distinct kinds of output actions are considered; in particular, a process can export public names, free outputs, or private ones, bound outputs.

The presence of input actions implies that, when performing a synchronization, the next state of a process might depend on the received name. The distinction between free and bound outputs gives a process different options when prompted for an output. To fully discriminate the different behaviours, observers are needed which can test whether two (received) names are the same name. This has called for an extension of the experimenter language with a sort of if-then-else construct; the original \( \pi \)-calculus has been extended with a mismatch operator which together with the original match operator gives us the needed observational power.

In this paper we concentrate on the finite fragment of the calculus. Extensions to the whole calculus are not difficult, but would require additional technicalities which would