Multi-Action Process Algebra

Wojciech Frączak
Université de Paris-Sud, LRI, Bât. 490,
F-91405 Orsay Cedex, France

Abstract. In this paper we propose a new process algebra based upon only three combinators: prefixing, composition, and restriction, but whose events (visible aspects of an evolution step) are structured as finite bags of actions. These structured events, called multi-actions, represent simultaneous execution of their actions and allow to handle the “simultaneity dependence” on events. This approach gives rise to a non trivial notion of communication channels, which parameterize composition and restriction operations. Multi-actions allow to avoid the “choice” as a primitive operation without loss of expressiveness of the algebra, which in turn ensures that all the defined equivalences are congruences.

Introduction

Process algebras, e.g. CCS [Mil80, Mil89], MEIJE [AB84], SCCS [Mil83], TCSP [BHR84, Old86], or PBC [BDH92], can be seen as specification languages for describing communicating system behaviors, called processes, which consist of discrete actions. Actually, a process defines relations between its events (occurrences of actions), usually causal dependence, concurrency, or conflict. In this work we consider a new kind of dependence between events, called simultaneity. Two events are in the “dependence of simultaneity” when neither of them can occur without the other. This relation leads us to enlarge the notion of event and to consider it as a set of actions occurring simultaneously, and no more as an occurrence of a single action. Such an event will be called multi-action.

We propose a new process algebra, allowing to handle the simultaneity dependence, which we call Multi-Action Process Algebra (MAPA). The syntax of MAPA includes only three operators: prefixing, composition, and restriction. The main features of MAPA are the following. The first one is that a process is prefixed by a finite bag of actions instead of a single action. The second one is that the composition of two processes is accompanied by an explicit specification of communication restrictions, formalized by the notion of channels.

A channel is characterized by the type and the number of actions able to be carried by it simultaneously. Two processes connected by a channel can only communicate via multi-actions allowed by the channel. We define operations on channels which, for example, allow to reverse the direction of a channel or to group two channels into a single one.

An operational semantics of MAPA is proposed in standard manner via “derivation rules” (à la Plotkin). In the case of MAPA, where “choice” as a
primitive operation does not exist, it is possible to consider different (but somehow equivalent) sets of derivation rules, each such set defining a slightly different operational semantics. We analyze a few such semantics and we choose the "most operational" one.

Apart from standard equivalences, like process graph isomorphism (≡), strong equivalence (≃), and weak equivalence (≈), which in case of MAPA can be seen as a new characterization of failure equivalence [BHR84]. The absence of choice as a primitive operator ensures that all the defined equivalences are congruences. We prove the following strict inclusions of the equivalences: \(\equiv \subseteq \simeq \subseteq \approx\), and we demonstrate that, w.r.t. \(\approx\) (and thus also w.r.t. \(\approx\)), all analyzed operational semantics are equivalent.

At the end we show that MAPA has the "completely general expressive power" w.r.t. at least two of the three criteria proposed by Vaandrager in [Vaa92].

The technical proofs are not included in this version of the paper because of the lack of space.

1 Communication primitives

In this section we describe the "communication primitives", i.e. the primitives which will be used by agents defined by our process algebra in order to interact. We start from a very intuitive and simple idea proposed by Milner for CCS [Mil80]. In his approach, agents can communicate by means of "links". An agent disposes of two primitive complementary actions for a link: sending (writing) on the link and receiving (reading) on the link. All the communications by links are supposed to be synchronous (i.e. neither a single sending nor a single receiving can occur without its inverse) and binary (only one receiving of one sending is allowed). In what follows, we will assume a nonempty set of atomic actions, \(L\), together with bijection \(-: L \rightarrow L\), called conjugation, which verifies \(a = a\), for each \(a \in L\). By \(\overline{a}\) we mean double application of the bijection on \(a\), i.e. \(\overline{a}\) actually denotes \(-(-a))\).

Multi-actions. Atomic actions are allowed to be grouped into 'bags' called multi-actions. More precisely, a multi-action is a finite bag\(^1\) (multi-set) of atomic actions. The set of all multi-actions over \(L\) will be denoted by \(\mu_f(L)\). The first intuition behind a multi-action is that it represents a simultaneous execution of several actions (its components), like in [BDH92] or [BB93], but it slightly differs from the intuitive meaning of actions from MEIJE [AB84] or SCCS [Mil83], where a simultaneous execution of two actions always leads to their synchronization.

Let \(\{c, b, a\}\) and \(\{\overline{a}, \overline{b}\}\) be two multi-actions which represent a simultaneous execution of atomic actions \(c, b, a\), and a simultaneous execution of \(\overline{a}, \overline{b}\), respectively. Intuitively, a simultaneous execution of these two multi-actions can be seen from outside as:

\(^1\) See Appendix for definitions and notations relative to bags (multi-sets).