Permutation of Transitions:
An Event Structure Semantics for CCS and SCCS

Gérard Boudol & Ilaria Castellani
INRIA Sophia-Antipolis
06560-VALBONNE FRANCE

Abstract. We apply Berry & Lévy's notion of equivalence by permutations to CCS and MEIJE/SCCS, thus obtaining a pomset transition semantics for these calculi. We show that this provides an operational counterpart for an event structure semantics for CCS and SCCS similar to the one given by Winskel.

Keywords: process algebras, pomset-labelled transition systems, event structures.

Contents

1. Introduction
2. Pure CCS: terms and transitions
3. Permutation of transitions
4. Event structures
5. Synchrony and asynchrony

1. Introduction.

A computational system evolves by elementary computations from one state to the other, in notation \( s \rightarrow s' \). Examples of state changes are transitions of a machine, \( \beta \)-reductions of \( \lambda \)-terms and rewritings in a term rewriting system. When states are abstract programs one may extract from their syntactical structure some indication of what has been performed and where it has happened. In other words, one may decorate transitions with a label \( w \), thus obtaining \( s \xrightarrow{w} s' \), where \( w \) is an occurrence of action. Now assume that \( s \xrightarrow{u} s_0 \) and \( s \xrightarrow{v} s_1 \): in many cases we may have the intuition that these two moves are compatible, or independent. This means that we are able to define what remains of one move after the other, in notation \( v/u \) and \( u/v \), in such a way that \( v/u \) can happen in state \( s_0 \), that is \( s_0 \xrightarrow{v/u} s'_0 \), and similarly \( s_1 \xrightarrow{u/v} s''_1 \). If \( u \) and \( v \) are really compatible, we should be able to perform them in any order, without affecting the result, that is: \( s'_0 = s''_1 \). This is known as the diamond property, or the parallel moves property. Moreover, two sequences of transitions should be regarded as equivalent, if they are equal up to commutation of compatible
moves, typically:
\[ s \xrightarrow{u} s_0 \xrightarrow{v/u} s' \sim s \xrightarrow{v} s_1 \xrightarrow{u/v} s' \]

This is the essence of Berry and Lévy's equivalence by permutations for sequences of (elementary) computations.

This equivalence was first elaborated by Lévy in his thesis (cf. [15]) upon Church notion of residual for the \( \lambda \)-calculus, and then used for recursive program schemes in [1]. It was further extended to deterministic term rewriting systems by Huet and Lévy in [14], and to non-deterministic ones by Boudol in [3]. In any case, this equivalence allows one to associate with each “state” a complete partial order of computations. These computations are equivalence classes of sequences of elementary moves, ordered by the prefix ordering, up to commutations. A similar notion is used for Petri nets by Nielsen, Plotkin and Winskel, who define in [20] an equivalence that “abstracts away from the ordering of concurrent firings of transitions” (this is also used by van Glabbeek and Vaandrager in [13], and by Best and Devillers in [2]; a similar idea is that of trace of Mazurkiewicz [16]). Moreover they show that for nets the ordered space of computations has a nice characterization: it is the space of configurations of an event structure. As a matter of fact, the three basic connectives of event structures - causality, concurrency and conflict - are already present in computations. Roughly speaking, two occurrences of actions (events) \( u \) and \( v \) are consistent (non-conflicting), with respect to a state \( s \) if they can appear in the same computation of \( s \): 
\[ s \cdots \xrightarrow{u} \cdots \xrightarrow{v} \cdots \]

In this case they are concurrent if they may be permuted:
\[ s \cdots \xrightarrow{u} \cdots \xrightarrow{v} \cdots \sim s \cdots \xrightarrow{v} \cdots \xrightarrow{u} \cdots \]

Otherwise they are causally related: one of them must precede the other.

In this note we propose an equivalence by permutations for Milner’s calculi CCS and SCCS [17,18], and show that the ordered space of computations of a term is the poset of configurations of an event structure. The events are simply occurrences of actions, and, roughly speaking, they are compatible if they lie on different sides of a parallel system, though some complications arise from communication. We show that each equivalence class of computations (up to permutations) may be represented as a one step transition, where the action is a labelled poset of events. With the exception of communication, this corresponds exactly to our semantics for “true concurrency” in [6,7]. Our operational semantics for CCS is similar to the one given by Degano, De Nicola and Montanari in [11], who obtain a poset transition from a sequence of “atomic transitions” that they call atomic concurrent histories. The poset transition semantics provides us with an operational counterpart to the interpretation of CCS terms as event structures. However it remains to be checked that our constructions coincide, at least in interpreting CCS, with those given by Winskel in [22] (see also [23]).

2. Pure CCS: terms and transitions.

As in [17], we assume a fixed set \( \Delta \) of names. We use \( \alpha, \beta, \ldots \) to stand for names. We assume a set \( \overline{\Delta} \) of co-names (complementary names), disjoint from \( \Delta \) and in bijection with it: the co-name of \( \alpha \) is \( \overline{\alpha} \), and its name is \( \text{nm}(\alpha) = \alpha = \text{nm}(\overline{\alpha}) \). Then \( \Delta = \Delta \cup \overline{\Delta} \) is the set of labels. We shall use