Partial Ordering Derivations for CCS

by Pierpaolo Degano*, Rocco De Nicola** and Ugo Montanari*

*Dipartimento di Informatica, Università di Pisa

**Istituto di Elaborazione dell'Informazione del CNR, Pisa

Abstract

In this paper we extend CCS transitions, labelled by strings, to concurrent histories, i.e. to transitions labelled by partial orderings. The two notions are linked by a theorem which shows that the strings can be obtained by taking all interleavings compatible with the partial orderings.

1. Introduction

In the last few years many mathematical models of concurrent and communicating systems have been proposed. If we take a particular standpoint, that of considering the way they describe the temporal ordering of events, we can classify them in two broad groups: models based on interleaving and models based on "true" concurrency.

The models based on interleaving describe the fact that a set of events may occur concurrently (independently from each other) by saying that they may occur in any order. In this way a total ordering among the possibly spatially separated and causally independent events is imposed. The proposers of models in this group stress the simplicity of the underlying mathematics, which offers the possibility of reasoning about systems, as a sufficient reason to stick to these models /11,12,14,1/.

The models based on true concurrency use, instead, partial orderings to explicitly describe the fact that events may take place concurrently. According to their proposers /10,8,9,13,17,15,16,18,3,7/, these models offer a more faithful picture of "reality" for their sharp distinction between nondeterminism and nonsequentiality. Moreover, some liveness properties of concurrent systems can be better understood and studied in this framework /4/.

In this paper we try to make a contribution to this debate by defining the notion of partially ordered computation for CCS. We chose CCS since it is a well-studied model based on interleaving and plays the rôle of test language for many proposed models of concurrency.

In Section 4, we first define recursively a function 'dec' on CCS terms which decomposes a term into a set of grapes. A grape represents a sequential process, and consists of a CCS (sub)term together with an access path, which identifies uniquely the subterm within the given term.

Then we generalize Milner's relation

$E_1 \xrightarrow{\mu} E_2$

introducing the notion of move

$I_1 \xrightarrow[I_3]{L} I_2$.

Here I_1, I_2 and I_3 are sets of grapes such that $I_1 \cup I_3 = dec(E_1)$ and $I_2 \cup I_3 = dec(E_2)$. Set I_1 represents the grapes of E_1 rewritten into the grapes I_2 of E_2. Grapes in I_3 stay idle. The label L is a synchronization term expressing the structure of the synchronization. By evaluating L within CCS synchronization algebra we obtain μ.

Moves can be seen as atomic concurrent histories. A concurrent history is a partial ordering relating head processes, events and tail processes. This is the basic element of the model for concurrent systems proposed in /4/. An atomic history contains a single event. A notion of concatenation, called replacement, is defined between histories. A computation in this model is a sequence of concurrent histories, the first being atomic and the others obtained by concatenating to the previous one a new atomic history. We briefly introduce concurrent histories in Section 3, a more detailed presentation being in /5/.

The results for concurrent histories can now be carried over to CCS, including the definition and characterization of the limits of infinite computations with respect to different liveness properties /4/. In particular, we apply in Section 5 a basic theorem on histories which allows us to show the correspondence between histories derivable by computations and Milner's many-step derivations $E_1 \xrightarrow{t} E_2$, t being a non-empty string of CCS atomic actions.

In summary, we start from the syntax of a model conceived to be based on interleaving and provide it with a notion of computation based on partial orderings. The standard notion of derivation can be easily recovered from this new notion. We hope that this work will help develop a deeper and more concrete insight into the relationships between the two approaches to modelling concurrency.

2. CCS - A Calculus for Communicating Systems

In this section we review the definition of CCS and its operational semantics. We use "pure" CCS, i.e. CCS without value passing /11/, and our version will be close to that presented in /6/. We refer to /11/ for the intuition behind the various operators.

Let X be a set of variables, ranged over by x. Let Σ_k be a set of operators of arity k. We use Σ to denote $\bigcup\{\Sigma_k \mid k \geq 0\}$. The set of recursive terms over Σ, ranged