Models for Data-Flow Sequential Processes

Mark B. Josephs

Centre for Concurrent Systems and Very-Large-Scale Integration,
Faculty of BCIM, London South Bank University,
103 Borough Road, London SE1 0AA, UK
josephmb@lsbu.ac.uk

Abstract. A family of mathematical models of nondeterministic data flow is introduced. These models are constructed out of sets of traces, successes, failures and divergences, cf. Hoare's traces model, Roscoe's stable-failures model and Brookes and Roscoe's failures/divergences model of Communicating Sequential Processes. As in CSP, operators are defined that are convenient for constructing processes in the various models.

1 Introduction

Consider sequential processes that communicate via input streams and output streams (FIFO buffers of unlimited storage capacity), as in Kahn-MacQueen data-flow networks [17, 18]. They are capable of the following actions:

- selectively reading data from their input streams,
- unreading (pushing back) data to their input streams,
- writing data to their output streams, and
- termination.

Processes can be composed in parallel. In particular, an output stream of one process may be connected to an input stream of a second process. Any datum written to the output stream by the first process should be transferred (eventually and automatically) to the input stream, where it becomes available for reading by the second process.

Processes can also be composed in sequence. When one process terminates its successor starts to execute. An important point here (implicit in [10]) is that *termination does not destroy the contents of input streams and output streams.*

Some years ago, the author, Hoare and He [16, 9] devised a process algebra for (nondeterministic) data flow, as a variant of Communicating Sequential Processes (CSP) [12]. (Part of this work was reproduced in [13].) We showed how to simplify the failures/divergences model [4] of CSP so that refusal sets were no longer required; failures could instead be identified with (finite) 'quiescent traces' [7, 14] or 'traces of completed computation sequences' [23].

At the time we did not consider a binary angelic choice operator, nor sequential composition. One purpose of this article is to rectify those omissions. Note that termination is modelled in CSP by a special symbol $\surd$ (success) [11],

A.E. Abdallah, C.B. Jones, and J.W. Sanders (Eds.): CSP25, LNCS 3525, pp. 85–97, 2005.

but that would not work for what we shall call Data-Flow Sequential Processes (DFSP). The solution is to create a 'stub' (a sequence of unread inputs) when termination occurs and there are no pending outputs, cf. [10, 6].

Another purpose of this article is to show how the more recent stable-failures model [27] of CSP can be adapted for DFSP. Indeed, a series of increasingly sophisticated models for DFSP will be introduced in a step-by-step manner, cf. [22]. Note that fairness issues, the focus of [23, 2, 3], are not addressed in these models.

The rest of this article is organised as follows. In Section 2, we recall the reordering relation [16, 9] between traces of directed events, a relation that captures the essence of data-flow communication. Subsequently, we define partial-correctness models (in Sections 3–5) and a total-correctness model (in Section 6) for DFSP, guided by what Roscoe [27] has done for CSP. In each case we consider the semantics of operators appropriate to the model. Conclusions are drawn in Section 7.

2 Directed Events, Traces and Reordering

A process is associated with an alphabet A, a (possibly infinite) set of symbols[1], partitioned into an input alphabet I and an output alphabet O. A symbol in I designates the transfer of a particular datum to a particular input stream; a symbol in O designates the transfer of a particular datum from a particular output stream. Such *directed events* are considered to be atomic, i.e., instantaneous.

Following Hoare [11], we define a *trace* to be a finite sequence (string) of symbols in A that expresses the occurrence of events over time as a linear order. In respect of a process that communicates through streams of unbounded capacity, however, two facts are noteworthy:

1. Events are independent if they are in the same direction but act upon different streams.
2. The occurrence of an input event does not depend upon the prior occurrence of an output event.

The first fact would justify taking a more abstract approach, namely, to follow Mazurkiewicz [21] by defining a trace to be an equivalence class on A^*. The two facts taken together would justify being more abstract still, namely, to follow Pratt [24] by defining a trace to be a partially-ordered multiset (pomset) on A. For example, if a and b are independent input events and c and d are independent output events, then the strings $cabd$ and $cbad$ are equivalent, but the only ordering between events is given by $a < d$ and $b < d$.

[1] To be more concrete, we have in mind compound symbols with $s.d$ referring to stream s and datum d. We would then require that $s_0.d_0 \in I$ and $s_1.d_1 \in O$ implies that $s_0 \neq s_1$. Moreover, if D is a data type associated with stream s, then $s.d \in A$ for all $d \in D$.