PARTIAL CORRECTNESS OF EXITS FROM CONCURRENT STRUCTURES

SIGURD MELDAL

Institute of Informatics, University of Oslo, P.O. Box 1080, Blindern, 0316 Oslo 3, Norway

Abstract.

A rudimentary exit-mechanism from the parallel command of the language fragment CSP is introduced. A method for embedding invariants in a standard partial correctness system with pre- and postconditions is presented. Proof rules for exits from concurrent systems are introduced, and a simple data flow system is verified.

CR categories: D.2.4, F.3.1.

1. Introduction.

Hoare's Communicating Sequential Processes [1] has proved an interesting vehicle for exploring various issues about the programming of concurrent systems, and has been influential on the design of modern programming languages, e.g. Ada 1 [2] and Occam [3]. In [4] we showed how we may give a compositional, fully abstract partial correctness semantics for nested concurrency in CSP. In this paper we introduce a simple exit-mechanism into CSP, allowing jumps out of processes and parallel commands. We also show how invariants may be embedded in the usual partial correctness context, and use this to give proof rules for exits from concurrent systems.

2. Restrictions and notation.

We assume the following simplifications about CSP programs:

- We shall deal only with flat concurrency (no nesting of parallel commands), though the methods are applicable to more complex structures as well – even spawning. (See Meldal [5, 6].)

- We do not employ the distributed termination convention. (We could handle

---

it at the cost of some complexity in a similar way to that of Soundararajan and Dahl [7].)

- Concurrent processes have disjoint variable spaces.
- Processes are uniquely named.
- Only integers are communicated between processes.

We extend the grammar of CSP with a new

\[
\langle \text{simple command} \rangle : \langle \text{exit command} \rangle,
\]

where \( \langle \text{exit command} \rangle \) is given as

\[
\langle \text{exit command} \rangle ::= \uparrow
\]

\( \uparrow \) is understood as an exit from the concurrent system, in effect stopping the execution. The difference between this command and an \textbf{abort} command is that the termination of a system by execution of \( \uparrow \) is considered a normal termination in the partial correctness context. (The terse syntax is chosen as being in the spirit of CSP.)

This is a rudimentary exit-mechanism, but it suffices for presenting the principles of our system, which is extendable to more complex mechanisms. A generalization of this proof system to handling a richer language with multi-level concurrency and exits to any given level is presented in [5].

We assume that a first order theory with the necessary arithmetic, definitions of sequences etc. is given. On top of this we define a theory of partial correctness statements.

Let \( \text{ident} \) denote the type of identifiers, i.e. character sequences.

Let \( \text{messages} \) denote the type \( \text{integer} + (\text{ident} \times \text{integer}^*) \). A message is either just an integer or a constructor with a (possibly empty) sequence of integers.

Let \( \text{proc} \) denote the type of process identities.

Let \( \text{history} \) denote the type \( \text{proc} \times \text{proc} \times \text{messages}^* \). A triple \((p, p', m)\) in a history denotes a communication of the message \( m \) from process \( p \) to process \( p' \).

Each process \( p \) has a mythical variable \( h \) of type \( \text{history} \) whose value is the history of communications that \( p \) has been involved in in the past. At the initiation of a process, \( h \) equals the empty sequence.

We introduce, for each process, a local constant \( t \) denoting that process.

Whenever a communication command is executed the history of a process

\[^2\) \( D^* \) for some set \( D \) is the set of finite sequences on \( D \). \( D + D' \) is the disjoint union of \( D \) and \( D' \).\]