What Process Algebra Proofs Use Instead of Invariance

Leslie Lamport

Sun 22 Jan 1995 [15:07]

Here's how I think process-algebraic proofs work in general—at least, in the "Amsterdam style". It all sounds pretty abstract, but I've looked at a couple of simple examples and it seems to actually work.

I'll consider the problem of proving equivalence of two processes. Proof of implementation should be analogous, for some suitable notion of implementation.

First, consider the finite-state case, with no $\sum$'s or data-bearing actions, where the process essentially specifies a finite-state machine. In this case, we can apply the process-algebra laws to symbolically execute the state machine, rewriting the process P in the form

$$P \;=\; \ldots + \alpha_1 \,.\, \alpha_2 \,.\, \ldots \,.\, \alpha_k \,.\, P \;+\; \ldots$$

where the sum is finite. The terms in the sum essentially describe the reachable states of the state machine—that is, the invariant. I'm sure there are algorithms to determine if two such representations are equivalent. (I would guess there's a smallest such representation, but I don't know much about finite state machines. If there is, then it can be defined to be the canonical form of P, and two processes are equivalent iff they have the same canonical form.)

Next, consider the case with data. Actions are of the form $\alpha(v)$ for some constant expression v, processes can have parameters, and the processes can use the operator $\sum_{v \in S}$. (For simplicity, I'll exclude the corresponding parallel composition operator $\prod_{v \in S}$; I don't think there's any inherent problem adding it.) I believe that, without loss of generality, by taking suitable Cartesian product spaces and applying the laws of process algebra, any process P can be rewritten in the form

$$\begin{aligned} P(v) \;=\; & \\ \ldots + & \sum_{w_1 \in S_1} \ldots \sum_{w_n \in S_n} \alpha_1(f_1(w_1, \ldots, w_n)) \,.\, \alpha_2(f_2(w_1, \ldots, w_n)) \,. \\ & \qquad\qquad \ldots \,.\, \alpha_k(f_k(w_1, \ldots, w_n)) \,.\, (P(w_1) \| \ldots \| P(w_n)) \\ & + \ldots \end{aligned}$$

By taking more complicated data spaces, using trees instead of just Cartesian products, one can define a new machine—basically, replacing $P(w_1)\| \ldots \|P(w_n)$ by $P(\langle w_1, \ldots, w_n \rangle)$—I think a process can be rewritten in the form

$$P(v) \;=\; \ldots + \sum_{w \in S} \alpha_1(f_1(w)) \,.\, \ldots \,.\, \alpha_k(f_k(w)) \,.\, P(w) \;+\; \ldots$$

In either case, these summands represent the reachable states in an unbounded-state machine—in other words, the invariant. Proving the equivalence of two such representations is, in general, undecidable. Once one has such a representation, one can keep expanding it by substituting for $P(w)$ in one term and applying algebraic identities to get back to the standard form. In practice, to prove equivalence of two processes, one has to expand their representations until they become isomorphic. I suspect that there's a relative completeness theorem asserting that if you can express all the sets S and functions f_i you wanted and could prove all you valid facts about them, then you can prove the equivalence of any two equivalent processes. Rob, do you know of such a result?