A CLP Approach to Modelling Systems

Joxan Jaffar*

School of Computing, National University of Singapore,
Republic of Singapore 117543
joxan@comp.nus.edu.sg

We present a formal method for modelling the operational behavior of various kinds of systems of concurrent processes. A first objective is that the method be broadly applicable. A system can be described in terms of its processes written in a traditional syntax-based manner, or in some non-traditional form such as a timed automaton. The number of processes may be fixed, or parameterized, or, because of dynamic process creation, unbounded. The communication and synchronization between processes may be synchronous or not, and via shared variables or some form of channels. We may have a traditional interleaving of processes, or use a specific scheduling strategy. The observables modelled should not be restricted to just the values of the program variables, but possibly other attributes of the system such as its registers and cache, its clock and battery values, etc. An example application area which touches upon these characteristics is that of determining worst-case execution time.

We choose to model a generic system S in the form of a CLP program P. The model-theoretic semantics of P shall characterize the "collecting" semantics of S, that is, those states that are observable. The proof-theoretic semantics of P, on the other hand, further characterize the "trace" semantics of S. An advantage of this CLP approach is that intricate details of the system can be captured in a familiar logical framework.

We then present a specification language for an extensive class of system behaviors. In addition to the traditional safety and liveness properties which specify the universality or eventuality of certain predicates on states, we introduce the notions of *relative safety* and *relative progress*. The former extends traditional safety assertions to accommodate non-behavioral properties such as symmetry, serializability and commutativity between processes. The latter provides for specifying progress properties. Our specification method is not just for stating the property of interest, but also for the *assertion* of properties held at various program points.

Finally, we present an inference method, based upon a notion of *inductive tabling*, for proving an assertion A. This method can use assertions that have already been proven, use the assertion A itself, in a manner prescribed by induction principles, and dynamically generate new assertions. All these properties are shown to be useful in preventing redundant computations, which then can lead to efficient proofs. Our proof method thus combines the search characteristic of model-checking and abstract interpretation, and methods of inductive assertions.

We demonstrate a prototype implementation on some benchmark examples.

* Joint work with Andrew Santosa and Răzvan Voicu.

W.-N. Chin (Ed.): APLAS 2004, LNCS 3302, p. 1, 2004.