Tutorial on FDR and its Applications

Philippa Broadfoot and Bill Roscoe

Oxford University Computing Laboratory
Wolfson Building, Parks Road
Oxford OX1 3QD, UK
{pb,awr}@comlab.ox.ac.uk

FDR[1] is a refinement checker for the process algebra CSP [2,4], based on that language's well-established semantic models. FDR stands for Failures-Divergences Refinement, after the premier model. In common with many other model checkers, it works by "determinising" (or normalising) a specification and enumerating states in the cartesian product of this and the implementation. Unlike most, the specification and implementation are written in the same language. Under development by its creators, Formal Systems (a spin-off of the Computing Laboratory) since 1991, it now offers a range of state compression methods. On current workstations it can work at up to 20M states/hour with only a small degradation on moving to disc-based storage.

Adaptations of FDR have been, or are being made, to accommodate other input notations such as UML, but in this tutorial we will concentrate on CSP. We will give a brief introduction to the CSP input language, and demonstrate FDR's use in modelling

– Timed systems
– Fault tolerance
– Cryptographic protocols: FDR was, we believe, the first general-purpose model checker to be used for these, and we will demonstrate the Casper protocol-to-CSP compiler [3].
– Information flow analysis

as well as discussing the techniques it uses for addressing the state explosion problem.

FDR has been much used in industrial work in areas such as computer security, safety-critical systems, communications networks and telecommunications.

References

1. Formal Systems. FDR web site: `http://www.formal.demon.co.uk/FDR2.html`
2. C. A. R. Hoare. "Communicating Sequential Processes", Prentice Hall (1985).
3. Gavin Lowe. Casper web site: `http://www.mcs.le.ac.uk/~gl7/Security/Casper/`
4. A. W. Roscoe. "The Theory and Practice of Concurrency", Prentice Hall (1998).