Part III
Using CSP in Practice

In Chaps. 14–17 we mirror the structure of TPC by including in-depth material on the practical use of CSP. In TPC there were three chapters in Part III: deadlock, time, and case studies. The case studies in the printed version of TPC were peg solitaire to show how CSP can be used to model combinatorial systems, a distributed memory to introduce the topic of data independence, and a cryptographic protocol to model a system with complex data.

Chapter 13 of TPC is an extensive study of design rules by which deadlock-free systems can be developed. Some of the more basic material from it was introduced in Sect. 4.2 of the present book. The reader is advised to look at TPC for further details.

Timed systems had already become one of the main practical application areas of FDR when TPC was published, but no real theoretical understanding existed of the relationship between the continuous Timed CSP language of Reed (Oxford University D.Phil. Thesis, 1988), Schneider (Concurrent and Real-Time Systems: The CSP Approach, Wiley, New York, 2000) and the discrete-time dialect introduced in Chap. 14 of TPC. Since that time there have been considerable advances both in the theoretical understanding of what can be done with discrete time in CSP, and in the support for time provided by FDR. Our Chaps. 14 and 15 bring this topic up to date.

In using FDR one usually concentrates on checks of the form $Spec \ [X= Imp$ where $Spec$ is a finite-state process representing a specification, $X$ is one of the CSP models, and $Imp$ is another finite-state process representing the implementation. Much has been discovered in recent years about what can be achieved by looking at the more general model $LHS(Imp) \ [X= RHS(Imp)$, namely applying two different contexts to the same process and comparing them. Chapter 16 looks at this topic, gives details of FDR’s compression algorithms, and discusses other advanced techniques in the use of FDR.

Anyone who has experimented with FDR to a significant extent will realise that what one can do with it is limited by the state explosion problem, by which the number of states a system has tends to grow exponentially with the numbers of parallel components and parameters involved in its definition. There is no real doubt that in some cases this increase in complexity is genuine in the sense that the verification
problem truly is a lot harder as the system grows. But equally there are many cases
where one has the strong feeling that no significant new issues arise as the system
grows, so checks really should not become so much harder. Chapter 17 looks at
some techniques that can be used to counter this problem in the context of CSP.
These include data independence, induction and a study of buffered systems.