

Exploiting formal methods in the real world: a case study of an academic spin-off company

G. M. Reed

Oxford University and Formal Systems

There have many academic spin-off companies set up over the past decade to exploit fundamental research in formal methods. Very few of these companies have survived, and even fewer have made significant profits. In this paper, as co-founder and director of one such company, I will give a brief history of the company, and discuss the challenges and opportunities involved.

1 Formal Systems

Formal Systems (Europe) Ltd was founded in Oxford in 1989 to exploit the research on CSP and Z done at Oxford University in the 80's. This research had high international recognition, and very strong links to UK industry. There is a very prestigious UK award (the Queen's Award for Technological Achievement) which is essentially an institutional "knighthood". In 1990, Oxford University and Inmos Ltd won this award for the development of formal methods in the specification and design of microcomputers; this work involved CSP and the occam programming language. The award cited that the use of formal methods had reduced the development time of the IMS T800 Transputer by 12 months. In 1992, Oxford University and IBM won the award for the use of formal methods - notations, theories, and processes, and specifically the use of the Z notation - in the production of the CICS transaction processing software. The award cited that formal methods had reduced the development cost by an estimated half million dollars. For each of these two awards, the Oxford team leaders (Bill Roscoe and Jim Woodcock) became directors of Formal Systems, and several of the team members became full-time employees (in particular, Michael Goldsmith became managing director).

During the early 90's, Formal Systems developed the FDR model-checker for concurrent finite state machines. FDR was developed as a result of collaborative work between Formal Systems and Inmos Ltd as a tool for verifying VLSI designs. It was used extensively in the design of the T9000 transputer and the C104 routing chip. Subsequently, it was used in the design and verification of a fault-tolerant processing system for a high-reliability embedded control system in conjunction with Draper Laboratories in the US. FDR found previously undetected faults in the prototype developed by Draper, and was used in a redesign of the software. This redesign was proved correct, and the final code was reduced 75 percent from that of the original.

Given the above start, it was assumed that Formal Systems would quickly grow into a large profitable company. Although successful, this growth has not yet happened. I will indicate below some of the lessons we have learned, and indicate some new commercial strategies.

2 Problems

2.1 Marketing problems with customers

- reluctance to use new notation
- confusion over multitude of formal methods
- level of ability and experience needed to understand and design models simply too high
- correctness of software not crucial; post-production errors can be patched over internet
- formal methods seen as causing unacceptable delay in the development process
- contact with customer is via their research department which comes to view you as a competitor for resources

2.2 Technical problems

- methods often do not scale
- state-explosion in model-checking
- difficult to make tool support usable for practitioners

2.3 Internal problems

- good researchers are often not good at marketing
- resistance of academics to a change of culture
- temptation to become subdepartment of university

3 Some solutions

3.1 Simplify notations and combine methods

Most of the current work at Formal Systems involves the use of FDR, which explores the behaviour of a pair of processes, using the operational semantics of CSP to determine whether one process (the implementation) refines another (the specification) or, if not, to find a counterexample by producing a behaviour of the implementation not found in the specification. Using the same language for describing both systems and properties greatly simplifies stepwise refinement and compositional development (see [R]). It also presents the user with only one notation to master.

It is clear that some system properties are best described in terms of state changes and others in terms of communications. It is also clear that automated verification, if possible, is sometimes more efficient via theorem-provers and sometimes more efficient via model-checkers ([RSR],[RSG]). Using action systems, it is possible to give a state-based approach within the semantics of CSP ([B],[M]), and thus use FDR to check state-based properties. Recent work in [RS] gives a technique for coupling specification and verification of components within a system using state-based specification and theorem-proving on some components and event-based specification and model-checking on others. This work is based on the theory of refinement in the semantic models of CSP.

Recent work [OR] has also shown how the semantics of (real-time) Timed CSP [RR] can be translated into a discrete timed model for CSP in which properties can be model-checked on FDR. Hence, it is now possible to specify and verify temporal behaviour within the same notation.

3.2 Attack state-explosion and scaling problem

New compression techniques developed by Formal Systems over the past few years together with increased processing power has significantly raised the number of states which can be explored. It is now practical to explore $O(10^8)$ states at 10^7 per hour on standard hardware. However, this is still not sufficient for many real-world problems.

Real advances to the problem are coming from the work in ([Rl],[SRe],[SRo]) on data independence and induction. The work in [RL] gives methods to calculate automatically thresholds for model parameters such as size of data types, number of nodes, etc. If a model is instantiated with parameters of at least the size of the thresholds and a property is proved correct, then the property is established correct for all values of the parameter. New induction techniques in [SRe] and combined with data-independence results in [SRo] are also proving powerful techniques in the reduction of the state space. These techniques allow FDR to formally establish properties of arbitrary branching networks.

3.3 Identify the most advantageous problem domain

As mentioned above, the original application domains for Formal Systems and FDR were in the design and verification of VLSI and embedded control systems. It was only when noticing that a large percentage of the audience at an industrial course offered by Formal Systems in Boston were from “security” agencies, that it became evident it might be beneficial to produce applications in the domain of computer security. The resulting applications to security protocols and information flow developed by Roscoe and Woodcock have since achieved an almost benchmark status for applying model-checkers.

Currently, eighty percent of Formal System’s contracts are in the domain of computer security and are funded by UK and US agencies. We are now moving into the commercial security market.

3.4 Hide the formal methods from the customers

As noted, there is considerable resistance to the use of formal methods by potential customers. One strategy is to hide the pain and only show the benefits.

In collaboration with GrammaTech Inc., and with funding from the US Office of Naval Research, Formal Systems is engaged in a two-year project aimed at property-checking in the UML context. The goal is to exploit state-machines which are created by the Rational/ObjecTime Rose RealTime tool [RSC]. Such state-machines are currently relegated to a documentary role. By applying FDR behind the scenes, it is possible to provide valuable information to the user without requiring their explicit knowledge of FDR [WRG].

Formal Systems is also currently negotiating with Motorola and Telelogic in the joint development of a verification tool with SDL as the specification language with a translation compatible to FDR. Such a tool would be of considerable use in telecommunications.

Gavin Lowe has developed the tool Casper [L] which allows security protocols to be specified in the usual ASCII-fied script found in the security literature, and then be translated into a script for verification by FDR. Hence, security protocols can be verified by people with no previous experience of model-checking.

A new commercial strategy in the use of formal methods by Formal Systems is to establish a “bank” for verified software. Software in the bank has been specified in conjunction with its owner and (to the state-of-the-possible) verified by Formal Systems. It is held for reusability, updating, and legal proceedings. It is not necessary for the owner to know formal methods. Once software is in the bank, it becomes available as a “component” to be used with only the necessity of verifying the connections between other components. The use of formal methods is completely hidden from the customer, but provides a clear benefit.

3.5 Bring in the suits

Finally, it is probably necessary to take on venture capital, buy in management and marketing, and become greedy capitalists. Once over the culture shock, we can talk about the good old innocent days while drinking margaritas in the Caribbean.

4 Acknowledgements

Much of the technical background for this paper was taken from [G] and [Z]. The reader should consult these papers for more detail.

5 References

- [B] M.J. Butler, *A CSP approach to action systems*, DPhil thesis, Oxford University, 1992.
- [CRe] S. Creese and J.N. Reed, *Verifying end-to-end protocols using CSP/FDR*, Proceedings of IPPS/SPDP Workshop on Parallel and Distributed Processing, LNCS 1586, Springer, 1999.
- [CRo] S. Creese and A.W. Roscoe, *Formal verification of arbitrary network topologies*, Proceedings of PDPTA'99, CSERA Press, 1033-1039.
- [G] M. Goldsmith, *Challenges to process-algebraic property-checking*, Proceedings of PDPTA'99, CSREA Press, 273-278.
- [L] G. Lowe, *Casper: a compiler for the analysis of security protocols*, Proceedings of the 10th IEEE Computer Security Foundations Workshop, 1997.
- [LR] R. Lazic and A.W. Roscoe, *Data independence with generalised predicate symbols*, Proceedings of PDPTA'99, CSREA Press, 319-325.
- [M] C.C. Morgan, *Of wp and CSP*, Beauty is our business: a birthday salute to Edsger W. Dijkstra, editors: D. Gries, W.H.J. Feijen, A.G.M. van Gasteren, and J. Misra, Springer-Verlag, 1990.
- [OR] J. Ouakine and G.M. Reed, *Model-checking temporal behaviour in CSP*, Proceedings of PDPTA'99, CSREA Press, 295-304.
- [R] A.W. Roscoe, *The Theory and Practice of Concurrency*, Prentice Hall, 1998.
- [RR] G.M. Reed and A.W. Roscoe, *The timed failures-stability model for CSP*, Theoretical Computer Science 211 (1999), 85-127.
- [RS] J.N. Reed and J.E. Sinclair, *Bicompositional refinements of loosely coupled specifications*, submitted for publication.
- [RSG] J.N. Reed, J.E. Sinclair, and F. Guigand, *Deductive reasoning versus model checking: two formal approaches for system development*, Proceedings of Integrated Formal Methods 99, 1999.
- [RSR] J.N. Reed, J.E. Sinclair, and G.M. Reed, *Routing: a challenge to formal methods*, Proceedings of PDPTA'99, CSERA Press, 305-311.
- [WRG] P. Whittaker, G.M. Reed, and M. Goldsmith, *Formal methods adding value behind the scenes*, Proceedings of PDPTA'2000, CSERA Press.
- [Z] I. Zakiuddin, *Current limits for exploiting automated verification*, Proceedings of PDPTA'99, CSERA Press, 319-326.