Abstract Partial Deduction Challenged
(Summary)

Stefan Gruner

Declarative Systems and Software Engineering Group
Department of Electronics and Computer Science
University of Southampton - SO17 1BJ (GB)
sg@ecs.soton.ac.uk

"Every day try to falsify your favourite theory" (Karl Popper)

This short paper (poster) summarises my presentation [6] at this workshop [1] in September 2002[1]. It also reflects some of the discussions at (and some new insights since) the workshop, before this summary went into print.

After several years of Abstract Partial Deduction (APD) —which is, for reasons of simplicity, here just regarded as a joint technique combining *Partial Deduction* and *Abstract Interpretation*— at least two APD tool prototypes are available: These are SP [4] and ECCE [8] which both operate in the abstract domain of Regular Unary Logic (RUL) [13]. Both are implemented in PROLOG, and both transform PROLOG input to PROLOG output during an abstract partial deduction run. These two are the tools under consideration here. (The system CPE [12] is not considered because it doesn't support the language PROLOG.) Due to cooperation between the authors of [8][4], essential parts of the RUL processing source code of SP are reused and integrated into the source code of the ECCE software system. Despite of certain problems which these tools may still run into on particular input programs (especially nontermination? — see discussion in [10]), it had been conjectured in [9] that RUL-APD could produce interesting results in *Infinite State Systems Verification* (ISSV), especially when applied to systems described in terms of a process algebra (e.g. CSP). My experiments referred to in this summary were mainly motivated by that conjecture [9] (but also by the insights of [11]). Further I was interested into a direct performance comparison between the above mentioned RUL-APD tools in their currently available versions [8][4]. Finally, by choosing the well-known Bakery protocol [7] as an example which has already been successfully used as a test case for another logic-based approach to ISSV [2][3], the RUL-APD approach explored here can be roughly compared to that other (and partially successful) approach[2].

[1] I have safely kept my program source codes for those who wish to repeat or extend my experiments. Please contact me to obtain the according files which, due to lack of space, cannot be printed out into an appendix to this summary.

[2] Note though that [2] did not specify Lamport's original protocol. They analysed an over-simplified variant of it which is flawed by circularity: mutual exclusion is there achieved by means of mutual exclusion because their specification does not allow more than one involved process at the same time to read the registers of the other processes.

In the context sketched above, my experiments have shown that in two variations of the Bakery system with parameters (variables) for the number of processes and the number of system steps, neither ECCE nor SP were able to detect the implicit system safety property by means of APD. In a parameterless CSP-wrapped variant with two "hard-wired" processes, however, the safety property could indeed be APD-detected[3]. Sufficient explanations of the reasons of those phenomena are still missing, but I regard this epistemical gap as a reasonable motivation for further research[4]. Latest work on APD with an abstract domain different from RUL already seems to indicate a promising direction [5].

References

1. F. Bueno & M. Leuschel (Eds.): *LOPSTR'02 PreProceedings of the International Workshop on Logic Based Program Development and Transformation* (This Workshop). Technical Report, Facultad de Informática, Universidad Politécnica de Madrid (E), September 2002.
2. F. Fioravanti & A. Pettorossi & M. Proietti: *Verification of Sets of Infinite State Processes using Program Transformation.* Proc. of LOPSTR'2001, LNCS 2372, Springer, 2002.
3. F. Fioravanti & A. Pettorossi & M. Proietti: *Combining Logic Programs and Monadic Second Order Logic by Program Transformation.* This Workshop [1], pp.166-181.
4. J. Gallagher: *SP System.* `http://www.cs.bris.ac.uk/~john/software.html`
5. J. Gallagher & J. Peralta: *Convex Hull Abstractions in Specialisation of CLP Programs.* This Workshop [1], pp.104-114.
6. S. Gruner: *Abstract Partial Deduction Challenged — Extended Abstract of Ongoing Work.* This Workshop [1], pp.251-258.
7. L. Lamport: *A New Solution to Dijkstra's Concurrent Programming Problem.* Communications of the ACM 17/8, pp.453-455, 1974.
8. M. Leuschel: *ECCE.* `http://www.ecs.soton.ac.uk/~mal/systems/ecce.html`
9. M. Leuschel & S. Gruner: *Abstract Conjunctive Partial Deduction using Regular Types and its Application to Model Checking.* Proc. of LOPSTR'2001, LNCS 2372, Springer, 2002.
10. P. Mildner: *Type Domains for Abstract Interpretation — A Critical Study.* Uppsala Theses in Comp. Sc. 31, ISBN 91-506-1345-6, Univ. Uppsala (S), 1999.
11. G. Snelting: *Paul Feyerabend und die Softwaretechnologie.* Informatik Spektrum 21/5, pp.273-276, Springer 1998.
12. G. Vidal: *Curry Partial Evaluator.* `http://www.dsic.upv.es/users/elp/peval/peval.html`
13. E. Yardeni & E. Shapiro: *A Type System for Logic Programs.* Journal of Logic Programming 10/2, pp.125-153, Elsevier, 1991.

[3] A significant difference between SP and ECCE could not be found in these experiments.

[4] In a later experiment (after this workshop) I found out that in the context of the test specification the tools could not detect the uniqueness of the minimum operation, thus the validity of $((S = S') \longrightarrow (min(S) = min(S'))$, which is a crucial preliminary of Lamport's original safety property proof.