Towards the Pervasive Verification of Automotive Systems

Thomas In der Rieden*, Dirk Leinenbach*, and Wolfgang Paul

Saarland University, Computer Science Dept., 66123 Saarbrücken, Germany
{idr, dirkl, wjp}@cs.uni-sb.de

Abstract. The tutorial reviews recent results from the Verisoft project [1]. We present a uniform mathematical theory, in which we can formulate pervasive correctness proofs for very large portions of automotive computer systems.
The basic ingredients of this theory are (i) correctness of processors with memory mamagement units and external interrupts [2], (ii) correctness of a compiler for (a subset of) C [3], (iii) correctness of the generic multitasking operating system kernel CVM [4], (iv) formal modeling of I/O devices and correctness of drivers [5], (v) correctness of serial interfaces [6], (vi) clock synchronization [7, 8], (vii) worst case execution time analysis using abstract interpretation [9].
Using ingredients (i), (iv), (v), and (vi) one can construct electronic control units (ECU) consisting of processors and interfaces to a FlexRay like bus [10]; timers on the ECUs are kept synchronized. An OSEKTime like real time operating system is derived from CVM [11].
The programming model for applications under this operating system is very simple: several (compiled) C programs run on each ECU in so called *rounds* under a fixed schedule. With the help of system calls the applications can update and poll a set of shared variables. The times for updating each shared variable are fixed by the schedule, too. An update to a shared variable in round k is visible to all application programs that poll this variable in round $k + 2$. This programming model is very close to the model used in [12], where formal correctness proofs for a distributed emergency call application in cars are reported.
Worst case timing analysis permits to guarantee, that applications and drivers satisfy the requirements of the schedule. If the requirements of the schedule are satisfied and the interfaces are programmed as prescribed by the schedule, then one can show that the user model is implememented by compiler, operating system and hardware [6].
An effort for the formal verification of all parts of the theory presented here is under way [13]. We report also on the status of this effort.

References

1. The Verisoft Consortium: The Verisoft Project. http://www.verisoft.de/
2. Dalinger, I., Hillebrand, M., Paul, W.: On the verification of memory management mechanisms. In Borrione, D., Paul, W., eds.: Proceedings of the 13th Advanced Research Working Conference on Correct Hardware Design and Verification Methods (CHARME 2005). Springer (2005).

* Work funded by the German Federal Ministry of Education and Research (BMBF) in the Verisoft project under grant 01 IS C38.

3. Leinenbach, D., Paul, W., Petrova, E.: Towards the formal verification of a C0 compiler: Code generation and implementation correctness. In: Proceedings of the 3rd International Conference on Software Engineering and Formal Methods (SEFM 2005). IEEE Computer Society (2005).
4. Gargano, M., Hillebrand, M., Leinenbach, D., Paul, W.: On the correctness of operating system kernels. In Hurd, J., Melham, T.F., eds.: Proceedings of the 18th International Conference on Theorem Proving in Higher Order Logics (TPHOLs 2005), Springer (2005) 1–16
5. Hillebrand, M., In der Rieden, T., Paul, W.: Dealing with I/O devices in the context of pervasive system verification. In: Proceedings of the 23rd International Conference on Computer Design (ICCD 2005), IEEE Computer Society (2005)
6. Beyer, S., Böhm, P., Gerke, M., Hillebrand, In der Rieden, T., Knapp, S., Leinenbach, D., Paul, W.J.: Towards the formal verification of lower system layers in automotive systems. In: Proceedings of the 23rd International Conference on Computer Design (ICCD 2005). IEEE Computer Society (2005).
7. Schneider, F.B.: Understanding protocols for byzantine clock synchronization. Technical report, Ithaca, NY, USA (1987)
8. Rushby, J.: A formally verified algorithm for clock synchronization under a hybrid fault model. In: Proceedings of the 13th annual ACM Symposium on Principles of Distributed Computing (PODC 1994), New York, NY, USA, ACM Press (1994) 304–313
9. Ferdinand, C., Heckmann, R.: Verifying timing behavior by abstract interpretation of executable code. In Borrione, D., Paul, W., eds.: Proceedings of the 13th Advanced Research Working Conference on Correct Hardware Design and Verification Methods (CHARME 2005). Springer (2005).
10. FlexRay Consortium: FlexRay Communications System Specifications Version 2.1. (2005)
11. OSEK group: OSEK/VDX time-triggered operating system. (2001) `http://www.osek-vdx.org/mirror/ttos10.pdf`.
12. Botaschanjan, J., Kof, L., Kühnel, C., Spichkova, M.: Towards verified automotive software. In Press, A., ed.: Proceedings of the 2nd International ICSE Workshop on Software Engineering for Automotive Systems (SEAS 2005), ACM Press (2005).
13. In der Rieden, T., Knapp, S.: An approach to the pervasive formal specification and verification of an automotive system. In: Proceedings of the 10th International Workshop on Formal Methods for Industrial Critical Systems (FMICS 2005), IEEE Computer Society (2005).