INDUSTRIAL EXPERIENCE OF ABSTRACT INTERPRETATION-BASED STATIC ANALYZERS

Jean Souyris
Airbus France
jean.souyris@airbus.com

Abstract This paper presents two Abstract Interpretation-based static analysers used by Airbus on safety-critical avionics programs: aiT [Thesing et al., 2003], a Worst case Execution Time analyzer developed by AbsInt, and ASTRÉE [Blanchet et al., 2003], aiming at the proof of absence of Run Time Errors and developed by the École normale supérieure.

Keywords: Abstract Interpretation, Worst Case Execution Time, Run Time errors, Product Based Assurance.

1. Introduction

The current verification process of an avionics program is almost exclusively based on tests. Testing has the great advantage of producing results by real executions of the program being verified but, unfortunately, this kind of verification has the major disadvantage of covering a tiny subset of the huge (very often considered as infinite) set of all possible executions. Moreover, if one takes into account the rate at which avionics program size is increasing from one aircraft development to the next one, maintaining the coverage of tests at the same level than it is today is increasingly expensive.

In this context, complementary tool-aided verification techniques must be used, which must compensate the disadvantages just mentioned. That is why Airbus makes a significant R&D effort, in collaboration with fundamental research laboratories and toolmakers, on static analyzers based on Abstract Interpretation [Cousot, 2000]. Such tools must satisfy the following constraints: they must analyze real code (source, assembly or binary) automatically, be usable by "normal" engineers and lead to results that can be used without heavy human effort.

The experimentations presented in this WCC 2004 Topical Session on Abstract Interpretation are an essential part of the process, which will allow using Abstract Interpretation-based static analyzers massively.

Two Abstract Interpretation-based static analyzers are presented in this paper: aiT from AbsInt, which computes Worst Case Execution Times by analysis of the binary of a program, and ASTRÉE, which proves the absence of Run Time Errors (division by zero, numerical overflows, array overflow, etc), both analyzing synchronous programs.

The paper is organized as follows: section 2 presents Airbus' R&D effort around Abstract Interpretation-based static analyzers, section 3 gives an idea of what the introduction of these kind of tools and techniques in the verification Workbench is, section 4 is about aiT, section 5 is about ASTRÉE , section 6 presents the Product Based Assurance, beyond the use of aiT or ASTRÉE, section 7 concludes.

2. An R&D Effort for Getting Industrial Static Analyzers

Abstract Interpretation-based static analyzers are specialized. The first specialization is due to the class of properties they prove, e.g., Run Time Errors. As Airbus' motivation is the proof, not only debugging, and because Abstract Interpretation means approximation (most of the time over-approximation), consequently "false alarms", a second specialization is required: the "per family of program" specialization. This is mandatory for limiting false alarms to the minimum (the "zero false alarm" objective), in the case of an analyzer of RTEs, or for obtaining a tight Worst Case Execution Time, in the case of aiT.

These specializations must be done by the tool makers with a close interaction with the End-User (here: Airbus) this is the R&D effort presented here.

This effort aims at obtaining static analyzers for solving industrial verification problems and introducing them into the development process of avionics programs. This cannot be achieved without fulfilling the following acceptance criteria:

- a static analyzer must constitute a *sound application of theoretical principles* (Abstract Interpretation framework),
- it must analyze *non modified programs ("the ones which fly"),*
- *"normal" engineers* can use it, on *normal machines,*
- it must be possible to *integrate it into the existing DO178B conforming process and verification workbench,*
- it must be fully *accepted by the developers,*
- and, from an industrial point of view, it must *reduce the cost of the verification significantly.*

The R&D maturation process for achieving the fulfillment of the acceptance criteria, for a particular static analyzer, consists in assessing the tool on real

programs, while defining the associated method of use, reporting problems to the tool maker, assessing again, and so on till the analyzer and the associated method are ready for an industrial use.

This process requires the following enabling conditions: close interaction with development teams and tool makers; and trainings, even on the theoretical framework, i.e., Abstract Interpretation.

The tools on which this process is currently applied, or has been applied on are: Caveat [Randimbivololona et al., 1999] (Program proof – Commissariat à l'énergie atomique – Caveat is used in A380 program development); AbsInt's Stackanalyzer; AbsInt's aiT [Thesing et al., 2003] (Worst Case Execution Time Analyzer); ASTRÉE [Blanchet et al., 2003] (École normale supérieure); Fluctuat [Goubault et al., 2002] (CEA - Floating-point calculus analysis).

3. Introduction to the Verification Workbench

DO178B conforming aspects

In its section 6.3.4. "Reviews and Analyses" paragraph f: "accuracy and consistency", DO178B [Randimbivololona et al., 1999] mentions a subset of the RTEs treated by ASTRÉE and the necessity of computing the WCET, but this standard does not impose a way to perform the relevant demonstrations.

More generally, Abstract Interpretation-based static analyzers are of great interest with respect to the fundamental objective of DO178B: *dependability.*

Industrial aspects

The major industrial characteristic of the avionic program verification workbench is to be as automated as possible. In fact, all phases of the development process are strongly integrated under the control of a configuration and process management tool. Static analyzers must be automatically controllable from this tool.

4. aiT (ABSINT – Worst Wase Execution Time Analyzer)

R&D effort

The collaboration between AbsInt and Airbus on WCET analysis started during the EU project DAEDALUS (on Abstract Interpretation, research and applications). The first aiT was developed for Coldfire 5307 micro-processor, which was used in a fly-by-wire avionics computer.

Why is computing WCET a challenge?

There are two main reasons: firstly, finding the inputs leading to the worst case is most of the time impossible, particularly by executing the program, or

by intellectual analyses; Secondly, capturing the subtle effects on the accelerating mechanisms of modern processors is also a challenge.

Legacy method

On safety-critical avionics programs like fly-by-wire programs, a measurement-based method exists, which deals with the above mentioned difficulties. The structure of the program allows reducing the problem of the identification of the inputs leading to the worse case of the complete program to such an identification at the level of very simple elementary operators the whole program is built with. In this context, the WCET of each basic operator is measured safely. The WCET of the whole program is computed by applying a formula deduced from the structure of the program and whose inputs are the WCET of the basic operators.

Limitation of the legacy method

The limitation of this method is mainly due to the conditions in which the measurements of the basic WCETs are preformed. Indeed, since the measurements are performed on basic operators individually, the initial state of the execution must be the worst possible: something like empty caches and pipeline. For some processors, this is sound but might lead to a very pessimistic WCET (waste of available CPU power), for others like the PowerPC 755, it is almost impossible to find the worst initial state (e.g., Pseudo LRU cache line replacement strategy).

Ait

Ait is an application of the theory of Abstract Interpretation. This basically allows to safely compute the WCET of a superset of all possible executions of the analyzed program. What is analyzed is the binary of the program. AiT embeds a model of the micro-processor and peripheral components in which characteristics having no impact on the timings have been abstracted away. Thanks to this model, the effects of the computation history based accelerating mechanisms are precisely captured. One can notice that even when a quite deterministic processor is used, aiT is worthwhile because fully automatic.

Targets

Safety-critical avionics program embedded into the A330/340 and A380 aircrafts, and using the following processors: PowerPC 755, Texas TMS320C33 and Coldfire 5307.

Status on the fulfillment of the acceptance criteria (sect. 2)

All criteria are fulfilled; the next step is the qualification of aiT and the associated method in the operational contexts in which they will be used, as required by DO178B.

Results

Two aiT have already been used in real industrial context: aiT for Coldfire 5307 and aiT for PowerPC 755.

AiT Coldfire 5307. When the aiT was available, the WCET of the fly-by-wire program running on a Coldfire 5307 board was already computed by the legacy method. Indeed the traditional approach can be applied to this processor. In fact, the pre-existence of timing figures allowed us to compare them with the ones produced by aiT. These comparisons were made for each basic operator, each module and each task. The conclusion was a completely deterministic behavior of aiT, and WCETs of tasks (which are relevant for the schedulability analysis) less pessimistic than the ones obtained by the legacy method.

Ait PowerPC 755. The legacy method is not reliable with this processor any more. So aiT is the only solution. Like for aiT Coldfire 5307, current status is aiT's capability to compute a tight WCET, and a deterministic behavior, even if less comparisons than for the other application could be made. Current Airbus' effort is on the user-validation of the tool on two safety-critical avionics programs, in order to be used for the temporal aspects of these program's certification process.

5. ASTRÉE (ENS – Proof of Absence of RTE)

R&D effort

This Abstract Interpretation-based static analyzer is developed by the École normale supérieure with the support of the French RNTL (Réseau national des technologies du logiciel) via the ASTRÉE project. Airbus also participates in this project as an End-User. The first class of program ASTRÉE is specialized for is safety-critical avionics synchronous programs (Fly-by-Wire programs).

Proving the absence of RTE "by hand"

Absence of RTE is impossible to prove by hand for real life programs and testing cannot be considered as a proof. The question is "how to be confident"

(and the 20 year service history of RTE-free operations is a strong indication) in the behaviors of a safety-critical avionics program with respect to RTE?

Legacy method

Precise answer to the above question must be made "per RTE type" and three kinds of requirements must be considered. First requirement: safety; if a runtime mechanism, like some exceptions of the processor (ex: floating-point overflow), is available then it is used. Since such an exception forces the computer to a no-return failure mode, the second requirement is for the availability of the avionics function. It states that RTEs must be detected during program development, as soon as possible. Last requirement: design and coding rules must be defined and well applied in order to avoid situations in which RTEs might occur.

ASTRÉE

This analyzer attempts to prove the absence of a Run Time Error in a synchronous program. Its maximum precision (zero false alarm) is obtained on code controlling servo-loops. In this case, if an alarm is raised by ASTRÉE, it is either a real problem or it is due to an imprecision of the description of the execution environment of the analyzed program (assertions on the inputs). With respect to the requirements mentioned just above, ASTRÉE is suitable for being sure of the absence of RTE as soon as the code is available, which is a lot better than getting a partial confidence after the heavy campaign of tests of the program.

Targets

Safety-critical avionics C programs embedded into A330/340 and A380 aircrafts.

Status on the fulfillment of the acceptance criteria (sect. 2)

OK for the A330/340 application. ASTRÉE is currently improved by ENS for being able to accept a wider class of programs, including A380 programs.

Results

Zero false alarm is a reality on the first family of programs ASTRÉE has been tuned for. The main computational characteristics of these programs are: about 100,000 lines of code, synchronous behavior, intensive numerical computations using the floating-point representation of real numbers, control-flow often "encoded" into boolean variables, type of calculus: digital filtering, servo-loop control.

The experimentations of ASTRÉE particularly revealed its ability to cope with the subtle effects of floating-point operations, specially when these computations are performed during hours, 10 hour typical flight duration for an A340, at a 10ms rate.

6. Towards the Product Based Assurance

AiT, ASTRÉE and the other Abstract Interpretation-based static analyzers are a first step towards the effective application Product Based Assurance concept. It should be noticed that this first set of static analyzers, e.g., WCET analyzers (aiT), RTE analyzers (ASTRÉE), and stack analyzers, share the following characteristic: the properties they prove are defined in their specification, i.e., they are not user-defined. The next step in the application of the Product Based Assurance will be supported by static analyzers that will allow to prove user-specified properties.

7. Conclusion

The main goal of this paper was to show how promising Abstract Interpretation is, for the verification of safety-critical avionics programs. Two successful applications of this theory were presented for illustrating this point of view.

With respect to Airbus' needs, the development of industrial applications of the Abstract Interpretation theory really started in 2000. To conclude, a lot of work is still needed for concretizing all the potentialities of the theory, and, for that, Abstract Interpretation must leave the state of confidentiality in which it currently is.

References

Randimbivololona, F., Souyris, J., Baudin, P., Pacalet, A., Raguideau, J., and Schoen, D. (1999). Applying formal proof techniques to avionics software: A pragmatic approach. In Wing, J.M., Woodcock, J., and Davies, J., editors, *Proceedings of the World Congress on Formal Methods in the Development of Computing Systems, FM '99,* volume II of *Toulouse, France, Lecture Notes in Computer Science 1709,* pages 1798–1815. Springer.

Thesing, S., Souyris, J., Heckmann, R., Randimbivololona, F., Langenbach, M., Wilhelm, R., and Ferdinand, C. (2003). Abstract interpretation-based timing validation of hard real-time avionics software. In *Proceedings of the International Conference on Dependable Systems and Networks (DSN 2003),* San Francisco, California, United States, pages 625–634. IEEE Computer Society Press, Los Alamitos, California.

Cousot, P. (2000). Interprétation abstraite. *Technique et science informatique,* 19(1-2-3): 155–164.

Blanchet, B., Cousot, P., Cousot, R., Feret, J., Mauborgne, L., Miné, A., Monniaux, D., and Rival, X. (2003). A static analyzer for large safety-critical software. In *Proceedings of the ACM SIGPLAN '2003 Conference on Programming Language Design and Implementation (PLDI),* pages 196–207, San Diego, California, United States. ACM Press.

Goubault, É., Martel, M., and Putot, S. (2002). Asserting the precision of floating-point computations: A simple abstract interpreter. In Le Métayer, D., editor, *Proceedings of the Eleventh European Symposium on Programming Languages and Systems, ESOP '2002,* Grenoble, France, Lecture Notes in Computer Science 2305, pages 209–212. Springer.