Recent advances of proof assistance in Ada95

C James
Ersatz Systems Machine Cognition, LLC, Colorado Springs, CO 80905; Tel: +1 719 21 9534; email: info@ersatz-systems.com

Abstract

This ongoing project maps a strongly typed theorem prover from source code in True BASIC into target code in Ada. The motivation is for portable embedding onto hardware applications. As a medium sized project, FORTH was not considered. Recent Ada standards were rejected due to refutation of the Eiffel constraints known as pre- and post-conditions. The chosen version of Ada95 has three types of vendors: large commercial; free products based on public domain C compilers; and small commercial. We chose one of the last with deep Ada historicity. By-product is refutation in mathematical logic of SPARK which claimed no formal proof. The perfect Ada95 code failed to auto translate to VHDL or C for Xilinx (AMD) and Altera (Intel) modelling engines. The subsequent solution was a small footprint SoC computer compliant with ITAR.

Keywords: Ada95, bivalent logic, classical logic, mathematical logic, M8VL4, Meth8/VŁ4, modal logic, universal logic system, proof assistant, SPARK, theorem prover, True BASIC

1 Introduction

In 2016, the first bivalent, universal, modal logic proof assistant appeared as the model checker Meth8/VŁ4 [1]. Its acronym is *Me*chanical *th*eorem prover in *8*-bits / *V*ariant Łukasiewicz *4*-valued logic. Using the software development method (SDM), it is in 4,000 lines of code (loc) in ANSI True BASIC, the educators' language, and compliant with tailored DoD-STD-2167A/2168. A recent advance was that the user manual became the requirements document. The speed for parsing theory of models was improved by introduction of the novel sliding windows technology. Versions of the prover are available as based on the number of propositional variables processed. The free student demo uses 2-variables. The pay-to-play is for 3, 4, and 11-variables (p-z), and up to 22-variables are by external look up tables on a 4.70 GB DVD.

Since 2016 Meth8/VŁ4 was used to evaluate 1,800 artifacts in 17,000 assertions for a refutation rate of 93% [1]. The artifacts span the gamut of pure mathematics, physics, and philosophy. It was of immediate use in analytic*al* theology where writers can now supply reproducible logic scripts for conjectures, heretofore avoided, to bring that field into the realm of exact, bivalent scientific research. Metaphysical assertions are reliably evaluated by using bivalent mathematical logic [2, 3]. A recent advance is the inadvertent discovery that George Boole invented modal logic 100 years before the claimant Saul Kripke [7]. Denied also is analogical and causal reasoning [4] with paraconsistent logics in hybrid systems used in artificial intelligence (AI) for look up engines, such as iAsk.ai [5] of Google.

2 The move to Ada

The motivation to recast Meth8/VŁ4 to Ada was based on portability, scalability, usability, and maintainability (pSUM). Because of the medium size of the product to be embedded, unthreaded FORTH was not considered. That left only Ada, and with which version to consider. Recent Ada standards introduced constraints as pre- and post-conditions. About 60-years ago from the first DoD DARPA computer lab of Dartmouth BASIC, Adm. Grace Hopper's COBOL included the technique to improve reliability of business software. About 30-years later, the idea was taken up again by Bertrand Meyer under the name of programming by contract for Eiffel as was codified in 2015 by set theory. However in 2023 Meth8/VŁ4 refuted that [6] and also AdaCore's SPARK touting the same model.

3 The jump backward to Ada95

Moving backward from Ada standards embellished with constraint conditions leaves the earlier Ada95 standard. The question was which vendor to choose for an Ada95 offering. We went to the three most visible vendors with pricing about $25,000 for minimal seats. Due to our not being a federal subcontractor they denied interest to

suggest Gnat-based vendors of free Ada95 compilers. Those are based on gnu/gcc which uses public domain C compilers. Our experience with C as *not* strongly typed was that the same program run on different C compilers produced different results in floating point arithmetic, due to no standard run-time implementation. Hence we found a smaller vendor with Ada historicity.

4 Designing from True BASIC to Ada95

Meth8/VŁ4 was designed as a recursive descent parser. There are 18 tokens: initially 11-variables (p-z), three operators (Not, possibly or existential, and necessarily or universal), and four connectives (And, Or, Equivalent, Imply). It turns out that the respective modal operators and quantifiers are proved as logically interchangeable.

The Xref utility in True BASIC (TB) allowed for instant cross reference from the source code for the labels of 54-subroutines and 2-user-functions. These translated in 56 packages in Ada95.

A strength of TB is in string handling, so the string-utility package uses original names for 2-procedures of Divide and UnpackB and for 28-functions as: CharInt, Chars, Control, Cpos, Cposr, DelChar, DelStr, GetQuotient, GetRemainder, KeepChar, LJust, Lower, MapChar, Max, MaxNum, Min, Ncpos, Ncposr, NoSpace, NPlugChar, PackB, Pos, Posr, RepChar, RepStr, Round, Uniq, and Upper. Fortunately most source code is included in the TB product. The certified test harness in Ada95 validated the string utility conversion for 76-cases with 55-cases from TB documents. The TB utilities contain programming nuances unknown to Ada reference manuals. For example in string processing, to avoid many large reads and writes in memory a string is divided into chunks less than 500 characters by recursion.

The conversion discovered the difference between one steeped in TB and one with C as the first language (CFL). This was borne out in the Reference Manual of Ada95 (RM95) as compiled by contractors and not by professional educators.

A persistent danger was the temptation to recast TB utilities into Ada as the CFL deemed Ada should be taught, implemented, and properly used. Hence, content was missed, not duplicated from the original code. For example, explicit border test conditions as accommodated in TB string utilities were not deemed as necessary to replicate, to engender a skewed functionality. Therefore the original approach to translate literally the origin code into Ada95 was thwarted by CFL ignorance of the lifecycle approach, namely, that the DoD definition of quality is “meeting requirements”.

We found that CFL with Ada as a second language (ASL) could not demonstrate claimed knowledge of ANSI BASIC. This previously metastasized in an 800-page book on computer programming languages by RW Sebesta (2012). It rewrote the genesis of ANSI TB pejoratively to raise Quick Basic or *street basic* to non-standard Visual Basic.

Some Ada replacements were fortuitous. For example the Ada attribute Float'ROUNDING was more compact than many conversions of integer to float, because TB has only one numeric type of IEEE-32 float. The attribute 'LENGTH translated over as did the string functions Delete and Replace_slice. The endless cyclical transforms of To_unbounded_string from To_string served as a nuisance source for typo mistakes.

For the TB procedure Packb(string$, bit_start, bit_length, number) and function Unpackb(string$, bit_start, bit_length) without source code, we programmed without need for a look up table of 256 bit codes to manipulate a bit stream string. A word boundary for Unpackb can now be ignored.

To minimize raising exceptions in scope and visibility, the layout flow of the mainline program component is one package of nested endless loops with equivalent of verbose "WHEN ERROR IN" wrappers around each of the units. That package is then called by the input and output package. Results of the prover can output to the user monitor and persistent files named with unique time-stamps. The product can be stopped and reset.

5 Some lessons learned

An Ada flaw is limiting numeric loop parameters as valid only for type Integer. For example, Ada has no native support for iterators of type Float as in FOR i = 9.0 TO 1.0 STEP -0.5 ... NEXT i. A work around is the much slower LET i = 9.0, WHILE i <= 9.0 AND i >= 1.0 DO, ... LET i = i - 0.5, LOOP. A shortcoming is not allowing an EXCEPTION handler inside the SELECT CASE.

Another flaw in Ada95 is the daunting complexity of file i/o. For example, the text book of Feldman et al 1999 avoids teaching that subject at all and with no index listings. In fact in RM95 the Annex A or elsewhere in the Ada literature, the file_type is not defined as taking a string argument, but which became common knowledge via Adalore.

The file i/o seems a side effect of the enormity of Richard Stallman's dictum at NYU to produce the

Gnat/gcc compilers as file-based and not library-based. It turned out that Government funded pork barrel spawned millionaires and the NYU spin off named AdaCore to sell public domain source code.

The Ada95 compiler of 2002 named gnat-3.15p was universally accepted as the stable last release of Ada95 from AdaCore. We found a compiler error in the conditional format of IF a_str = b_str THEN flag_unb_str := true_unb_str ;. Unless this was subsequently fixed without corrigendum, one may assume the anomaly propagated into Gnat Ada2024. AdaCore sells SPARK software to verify and validate other software products. We find no academic documents to use SPARK on gnat-3.15p or subsequent compilers and products of AdaCore.

We found not to trust three famous Ada95 texts having non informative indexes: RM95; Feldman et al 3rd ed 1999; and Barnes 2005. From 15 years prior, Cohen 2nd ed 1986 remains the default standard as written by a professional educator.

The resource support option of internet AdaForum turned out to render inexact responses seeped in Adalore. Volunteers misread subject line topics to morph topics into ill-formed essays. The cabal running the site allows for newbie questions to be critically labelled as homework problems and hence *unethical* to answer. The cabal also censors content as an organ of AdaCore. For example, that vendor supports a big_numbers-type using vectors. However for checking prime numbers, it is much slower than the TB approach to manipulate huge number as strings with an ensemble of functions. When this was pointed out, with the comment to test assertions of CFL workers, the posting account was summarily terminated for insulting volunteers.

Finally, the Ada vendor chosen with a perfect compiler was ultimately quite disappointing. While delayed answers were adequate, the result of the Windows 10 22H2 update caused the project manager tool to crash. The only remedy was a nuisance project: rebuild the laptop from an earlier version; program the hive to quash updating; and reinstall the compiler, a quirky task. When that failed, an offer to buy closer support within one-year as already provided went unanswered since its website crashed for two months, losing our emails.

Consequently on that laptop we installed Gnat studio manager which is freeware with no support. A compile switch for Ada95 was said not to work. AdaForum support turned out as guessing at best with no AdaCore public direction given for links, ignoring the forum as its tacit marketing tool.

6 Output from system on a chip (SoC)

M8VL4 in Ada95 has 4,100 loc with 500-global variables. It is compliant with the International Traffic in Arms Regulations (ITAR). Stand alone high-level synthesis (HLS) tools do not exist for converting Ada to portable VHDL or C for model input of FPGA at Xilinx (AMD) and Altera(Intel). From an AI subscription service, sold under false colors, we found it could not translate Ada95 to VHDL or to C. That forced our evaluation of the sacrilege of Ada-to-C translators. GnatPro CCG for Ada to C was not salable due to high cost and a 39-page user guide “supplement” by foreign non educators. We opted for a system on a chip (SoC) on a tiny foot print system board computer (SBC).

The immediate benefit from M8VL4 as hardware is the ironic correction of defective non bivalent logic systems as basis of the probabilistic vector space of AI to guess answers with percentages of veracity.

Acknowledgments

Thanks are due to referees for helpful comments.

References

[1] C. James ([current date]-2016), *Recent advances in the modal model checker Meth8 and VŁ4 universal logic.* Ersatz Systems Machine Cognition, LLC. ersatz-systems.com/RA.Meth8.abstract.pdf

[2] C. James (2024), *Mathematical logic of metaphysical perfection: the comprehensive mapping of analytical theology.* PhD dissertation. 87 p. 1/2024.

[3] C. James (2023), *Mathematical foundations in bivalent logic: 8-three hour graduate courses.* MS thesis. 367 p. 10/2023.

[4] C. James (2023), *Refutation of analogical and causal reasoning to invalidate artificial intelligence (AI) engines.* ersatz-systems.com/Digest%202023.10.07.01.pdf

[5] C. James (2023), *Refutation of AI as hybrid logic system of the iask.ai propaganda engine.* ersatz-systems.com/Digest%202023.06.16.01.pdf

[6] C. James (2022), *Refutation of theory of programs from Bertrand Meyer.* ersatz-systems.com/Digest%202022.09.30.03.pdf

[7] C. James (2020), *Refutation of Boole’s development theorem (and false attribution of modal logic to Kripke).* ersatz-systems.com/Digest%202022.01.05.01.pdf