Formalizing Java's Two's-Complement Integral Type in Isabelle/HOL

Nicole Rauch[*]
Technische Universität Kaiserslautern

Burkhart Wolff
Albert-Ludwigs-Universität Freiburg

We present a formal model of the Java two's-complement integral arithmetics. The formalization is based on a direct analysis of the Java Language Specification (JLS) [GJSB00] and led to the discovery of several underspecifications and ambiguities. Underspecifications are highly undesirable since even compliant Java compilers may interpret the same program differently, which leads to unportable code.

The Java integral types are finite datatypes that possess a surprisingly rich theory (they form a ring, for example) which comprises a number of highly non-standard and tricky laws with non-intuitive and subtle preconditions. With respect to their formalization, we followed the so-called *wrap-around approach*: integers are defined on $[-2^{n-1} .. 2^{n-1} - 1]$, where in case of overflow the results of the arithmetic operations are mapped back into this interval through modulo calculations. These numbers can be equivalently realized by bitstrings of length n in the widely-used two's-complement representation system [Gol02].

The advantage of this approach is that it closely follows the definition of the Java type `int` in the JLS, for which certain surprising properties like "Maxint + 1 = Minint" or crucial laws like the associativity law "a + (b + c) = (a + b) + c" hold. The formal model should reflect these properties. A "partial approach" (i.e. a modelling that does not allow overflows) is not able to prove them. The wrap-around approach therefore gives better support for automated reasoning.

The theory is formally analyzed in Isabelle/HOL [Pau94], that is, machine-checked proofs for the ring properties, divisor/remainder theorems etc. are provided. This work is a necessary prerequisite for machine-supported reasoning over arithmetic formulae in the context of Java source-code verification, especially of efficient arithmetic Java programs such as encryption algorithms, in particular in tools like Jive [MPH00] that generate verification conditions over arithmetic formulae from such programs.

In the future, we strongly suggest to supplement informal language definitions by machine-checked specifications like the one referred to in this abstract as a part of the normative basis of a programming language.

References

[GJSB00] James Gosling, Bill Joy, Guy Steele, and Gilad Bracha. *The Java^{TM} Language Specification – Second Edition.* Addison-Wesley, June 2000.

[Gol02] D. Goldberg. Computer arithmetic. In J.L. Hennessy and D.A. Patterson, editors, *Computer Architecture: A Quantitative Approach.* Morgan Kaufmann Publishers, third edition, 2002.

[MPH00] Jörg Meyer and Arnd Poetzsch-Heffter. An architecture for interactive program provers. In S. Graf and M. Schwartzbach, editors, *TACAS00, Tools and Algorithms for the Construction and Analysis of Systems*, volume 276 of *Lecture Notes in Computer Science*, pages 63–77. Springer Verlag, 2000.

[Pau94] Lawrence C. Paulson. *Isabelle: A Generic Theorem Prover*, volume 828 of *Lecture Notes in Computer Science.* Springer Verlag, New York, NY, USA, 1994.

[*]Partially funded by IST VerifiCard (IST-2000-26328)