The Interval-Enhanced GNU Fortran Compiler

MICHAEL J. SCHULTE, VITALY ZELOV, and AHMET AKKAS
EECS Dept., Lehigh University, Bethlehem, PA 18015, USA, e-mail: mschulte@eeecs.lehigh.edu

and

JAMES CRAIG BURLEY
Free Software Foundation, 59 Temple Place, Suite 330, Boston, MA 02111, USA,
e-mail: burley@gnu.org

(Received: 2 November 1998; accepted: 3 February 1999)

Abstract. Compiler support for intervals as intrinsic data types is essential for promoting the development and wide-spread use of interval software. It also plays an important role in encouraging the development of hardware support for interval arithmetic. This paper describes modifications made to the GNU Fortran Compiler to provide support for interval arithmetic. These modifications are based on a recently proposed Fortran 77 Interval Arithmetic Specification, which provides a standard for supporting interval arithmetic in Fortran. This paper also describes the design of the compiler's interval runtime libraries and the methodology used to test the compiler. The compiler and runtime libraries are designed to be portable to platforms that support the IEEE 754 floating point standard.

1. Introduction

Interval arithmetic provides an efficient method for performing operations on intervals of real numbers [19]. With interval arithmetic, each interval is represented by its lower and upper endpoints. More formally, an interval \(X = [a, b]\) is defined as

\[X = [a, b] = \{x \in \mathbb{R} \mid a \leq x \leq b\}.\]

For example, the interval \(X = [1.23, 1.24]\) is a closed interval that includes all values greater than or equal to 1.23 and less than or equal to 1.24.

Although the concept of interval arithmetic is relatively straightforward, it provides a powerful mechanism for bounding the results of floating point computations. As demonstrated in [19] and [1], interval arithmetic provides a practical method for bounding errors in numerical computations including roundoff errors, approximation errors, and errors due to inexact inputs. In addition, interval arithmetic has been used to develop validated algorithms for solving problems in several areas. For example, interval arithmetic has been used to provide validated solutions in global optimization, finding roots of functions, solving systems of linear and non-linear equations, performing numerical differentiation and integration, and solving systems of ordinary and partial differential equations. These and other applications for interval arithmetic are presented in [1], [5], [9], [10], [20].
Because of its ability to provide validated solutions, several software tools have been developed to support interval arithmetic. These include interval arithmetic libraries, [12], [16], [18], language extensions with support for interval arithmetic [7], [17], [24], and interval application software [9], [13], [14]. Although these tools give programmers access to interval arithmetic, they do not conform to a specified standard.

As noted in [15], the lack of a specified standard for interval arithmetic has the following disadvantages:

- Resources are unnecessarily expended to redevelop tools for interval arithmetic.
- Diverse semantics in programming interval computations inhibit multi-person development of interval application software.
- Interval arithmetic packages written in high-level languages seldom take advantage of machine hardware, which may lead to poor runtime performance.
- Interval arithmetic packages designed on one platform may not be portable to other platforms.

Furthermore, current interval arithmetic software packages do not always guarantee containment. As noted in [22], several packages for interval arithmetic produce incorrect results due to incorrect handling of arithmetic exceptions. Some of these packages also depend on the underlying platform’s built-in libraries when performing interval mathematical functions and interval input and output. Consequently, the correctness and tightness of the interval routines are dependent on the accuracy of the built-in libraries. This accuracy is often difficult to determine and can vary on different platforms [21]. Furthermore, previous software packages for interval arithmetic sometimes cause unexpected behavior. For example, with certain interval arithmetic libraries, the machine floating point rounding mode can be changed as a side-effect of performing interval arithmetic operations.

To overcome these difficulties, a standard for supporting interval arithmetic in Fortran was recently proposed [15]. This proposal specifies support for interval data types, interval arithmetic operations, interval relations, interval versions of mathematical functions, and interval I/O. The Fortran 77 Interval Arithmetic Specification builds upon [15] to provide a complete standard for supporting intervals in Fortran [4]. It adds several new operators and intrinsic functions, which are defined in [26]. It also defines a set of extended real intervals and their internal representation for IEEE 754 compliant processors. The set of extended real intervals is closed with respect to interval arithmetic operations and interval enclosures of real functions [25]. The Fortran 77 Interval Arithmetic Specification also gives algorithms for correct nonstop (i.e., without interrupts for arithmetic exceptions) handling of several interval intrinsics and operations, discusses interval expression optimization and mixed-mode evaluation, and gives several examples to illustrate the correct implementation of the specification.