Reducing Error on Modulo Interval Arithmetic

Noriko Soyama * Tsuneo Nakanishi † Fujio Kako‡ Akira Fukuda †

* Graduate School of Human Culture, Nara Women's University
† Graduate School of Information Science, Nara Institute of Science and Technology
‡ Faculty of Science, Nara Women's University

{soyama,kako}@ics.nara-wu.ac.jp {tun,fukuda}@is.aist-nara.ac.jp

Abstract

Modulo interval arithmetic is an arithmetic system on sets of integers. Since modulo interval is useful for representing a set of integers with discreteness and cyclicity such as loop indices or array subscripts, it is expected to use for various program analysis applied by compilers. However, naive application of modulo interval arithmetic possibly produces a bigger set of integers than the exact set. In this paper we discuss a technique to reduce error on modulo interval arithmetic especially for polynomials with modulo intervals.

1 Introduction

Interval arithmetic is arithmetic for intervals of real numbers. Originally interval arithmetic was developed to estimate rounding errors[3]. Interval arithmetic was employed for program analysis later[1].

In typical programs integers are used for loop indices or array subscripts and real numbers are almost used only for physical data to be computed. Thus we have proposed *modulo interval arithmetic* as useful arithmetic for handling data with these properties[5]. The modulo interval are denoted by $[a,b]_{m(r)}$: a is a lower bound value, b is a upper bound value, m is a modulus, r is a residue. A modulo interval $[a,b]_{m(r)}$ represents the set of the integers $\{x \in Z | a \leq x \leq b, x = nm + r, m \in Z\}$. We define modulo interval arithmetic likewise originally interval arithmetic. We introduce one application of the modulo interval on program analysis.

Consider the following loop.

```
Loop:  DO  30 I=4, 1000, 4
S:        A(3I) = ...
T:        ...   = A(2I-1)
    30 CONTINUE
```

Suppose that we want to analyze whether iterations of this loop can run in parallel or not. If there exist i and i' included in $[4,1000]_{4(0)}$ such that $3i = 2i' - 1$, there exists a dependence since the same element of the array A is referenced by the statement S at $I = i$ and by the statement T at $I = i'$. To check if there exist such i and i', it is enough to check whether the set of integers taken by $3i - 2i' + 1$ includes zero or not. The set of integers taken by $3i - (2i' + 1)$ where i and i' are included in $[4,1000]_{4(0)}$ is calculated as follows.

$$\begin{aligned} 3 * [4,1000]_{4(0)} \ominus_1 2 * [4,1000]_{4(0)} \oplus_1 1 \\ &= [12,3000]_{12(0)} \ominus_1 [8,2000]_{8(0)} \oplus_1 1 \\ &= [-1987,2993]_{4(1)} \end{aligned}$$

where the symbol $\oplus_1, \ominus_1$ and $\otimes_1$ are addition, subtraction and multiplication operations on modulo intervals respectively.

Obviously, $[-1987,2993]_{4(1)}$ does not include zero. This implies iterations in this loop can run in parallel.

Although the modulo interval has only the modulus and the residue for additional information to the interval numbers, it can represent values taken by indices of loops or subscripts of array variables more precisely than

interval numbers, as can be seen in before example. However, most properties of modulo interval arithmetic which are defined at present can produces a bigger set of integers than the exact set, since most arithmetic properties on modulo intervals are provided as inclusion relations. If we do not take notice of arithmetic algorithms, errors of operation can increase. Therefore, in case of program analysis which needs polynomial calculations on modulo interval, errors can degrade accuracy of program analyses.

In this paper, we propose arithmetic on modulo interval and methods of error reduction on modulo interval arithmetic especially for a polynomial calculation.

2 Modulo Interval Arithmetic

In this section, the definition and mathematical properties of the modulo interval are presented. See Reference[5] for proofs of mathematical properties.

2.1 The Modulo Interval

A modulo interval is defined for an arbitrary pair of real numbers a and b with $a \leq b$, a non-zero integer m referred to as *modulus*, and an integer r referred to as *residue*, and denoted by $[a, b]_{m(r)}$. A modulo interval $[a, b]_{m(r)}$ means the set of integers $\{x \in Z | a \leq x \leq b, x = mn + r, n \in Z\}$.

The modulo interval $[a, b]_{m(r)}$ is *normalized* if and only if both a and b are in $[a, b]_{m(r)}$ and $0 \leq r < m$.

2.2 Exact Arithmetic

Addition, subtraction and multiplication on modulo intervals are defined as follows.

Definition 1

$$\begin{aligned} A + B &\equiv \{a + b \mid a \in A, b \in B\}, \\ A - B &\equiv \{a - b \mid a \in A, b \in B\}, \\ A \times B &\equiv \{a \times b \mid a \in A, b \in B\}, \\ A + z &\equiv \{a + z \mid a \in A\}, \\ z + B &\equiv \{z + b \mid b \in B\}, \\ A - z &\equiv \{a - z \mid a \in A\}, \\ z - B &\equiv \{z - b \mid b \in B\}, \\ A \times z &\equiv \{a \times z \mid a \in A\}, \\ z \times B &\equiv \{z \times b \mid b \in B\}, \end{aligned}$$

where A and B are arbitrary modulo intervals, and z is an arbitrary integer.

2.3 Simplified Arithmetic

It is not practical to perform arithmetic operations as Definition 1 for computers in terms of computation time and memory consumption. Therefore we define a simplified arithmetic. Simplified addition ($\oplus_1$), subtraction ($\ominus_1$), and multiplication ($\otimes_1$) of modulo intervals $[a, b]_{m(r)}$ and $[c, d]_{n(s)}$ are defined as follows.

Definition 2

$$\begin{aligned} &[a, b]_{m(r)} \oplus_1 [c, d]_{n(s)} \\ &\quad \equiv [a + c, b + d]_{\gcd(m,n)(r+s)}, \\ &[a, b]_{m(r)} \ominus_1 [c, d]_{n(s)} \\ &\quad \equiv [a - d, b - c]_{\gcd(m,n)(r-s)}, \\ &[a, b]_{m(r)} \otimes_1 [c, d]_{n(s)} \\ &\quad \equiv [\min\{ac, ad, bc, bd\}, \\ &\qquad \max\{ac, ad, bc, bd\}]_{\gcd(m,n)(rs)} \end{aligned}$$

where the greatest common divisor(GCD) of integers m and n, denoted by $\gcd(m, n)$, is defined as the greatest positive integer which divides both m and n.

The following relations hold where A and B are modulo intervals.

$$\begin{aligned} A \oplus_1 B &\supseteq A + B, \\ A \ominus_1 B &\supseteq A - B, \\ A \otimes_1 B &\supseteq A \times B. \end{aligned}$$

When moduli of the each operand are same, the results of addition and subtraction become the same results of exact arithmetic.

$$\begin{aligned} A \oplus_1 B &= A + B, \\ A \ominus_1 B &= A - B. \end{aligned}$$

Similarly, simplified addition, subtraction and multiplication of a modulo interval A and an integer z are defined as follows.

Definition 3

$$
\begin{aligned}
[a,b]_{m(r)} \oplus_1 z &\equiv [a+z, b+z]_{m(r+z)}\,, \\
z \oplus_1 [a,b]_{m(r)} &\equiv [z+a, z+b]_{m(z+r)}\,, \\
[a,b]_{m(r)} \ominus_1 z &\equiv [a-z, b-z]_{m(r-z)}\,, \\
z \ominus_1 [a,b]_{m(r)} &\equiv [z-b, z-a]_{m(z-r)}\,, \\
[a,b]_{m(r)} \otimes_1 z &\equiv [a\times z, b\times z]_{mz(rz)}\,, \\
z \otimes_1 [a,b]_{m(r)} &\equiv [z\times a, z\times b]_{mz(zr)}\,.
\end{aligned}
$$

Similarly, the following relations hold where A is a modulo interval and z is integer.

$$
\begin{aligned}
A \oplus_1 z &= A+z, \\
z \oplus_1 A &= z+A, \\
A \ominus_1 z &= A-z, \\
z \ominus_1 A &= z-A, \\
A \otimes_1 z &= A\times z, \\
z \otimes_1 A &= z\times A.
\end{aligned}
$$

Division of a modulo interval by an integer are defined as follows.

Definition 4

$$
\begin{aligned}
&[a,b]_{m(r)} \oslash_1 z \\
&\quad\equiv [\min\{a \operatorname{div} z, b \operatorname{div} z\}, \\
&\qquad \max\{a \operatorname{div} z, b \operatorname{div} z\}]_{m \operatorname{div} z(r \operatorname{div} z)}
\end{aligned}
$$

where z is an arbitrary non zero integer and m is divisible by z.

The following relation holds for modulo intervals and a positive integer z.

$$
[a,b]_{m(r)} = \bigcup_{k=0}^{z-1} [a,b]_{mz(km+r)}\,.
$$

Using this relation, when m is not divisible by z, above property enables division by multiplying the modulus of a modulo interval by z as follows.

$$
\begin{aligned}
&[2,30]_{7(2)} \oslash_1 3 \\
&= ([2,23]_{21(2)} \cup [9,30]_{21(9)} \cup [16,16]_{21(16)}) \oslash_1 3 \\
&= [0,7]_{7(0)} \cup [3,10]_{7(3)} \cup [5,5]_{7(5)}\,.
\end{aligned}
$$

2.4 Error on Modulo Interval Arithmetic

The results of simplified arithmetic of modulo intervals can be estimated as bigger sets than the results of exact arithmetic of the modulo intervals.

Consider addition of modulo intervals $[2,10]_{4(2)}$ and $[8,14]_{3(2)}$ for an example. The results of exact addition and simplified addition are given as follows.

$$
\begin{aligned}
&[2,10]_{4(2)} + [8,14]_{3(2)} \\
&\qquad = \{10,13,14,16,17,18,20,21,24\}, \\
&[2,10]_{4(2)} \oplus_1 [8,14]_{3(2)} = [10,24]_{1(0)} \\
&\qquad = \{10,11,12,13,14,\cdots,24\}.
\end{aligned}
$$

The result of simplified addition is estimated too bigger than the result of exact addition. As this example, the result of simplified arithmetic of modulo intervals whose moduli are different is estimated bigger than the result of exact arithmetic of the modulo intervals. This is because the modulus of the result of simplified arithmetic is given as the GCD of the moduli of operand modulo intervals. Moreover, the result of simplified multiplication of modulo intervals is estimated bigger, even if the moduli of operand modulo intervals are same.

3 Arithmetical Error Reduction

In this section, we propose the redefined arithmetic for reducing error on modulo interval arithmetic.

3.1 Error Reduction for Addition and Subtraction

We propose improved addition and subtraction algorithms that reduce error by using the lowest common multiple (LCM) of both moduli of modulo intervals. First, expand one modulo interval by the LCM of both moduli of modulo interval. Second, calculate the expanded modulo interval and another modulo interval. However, error remains when the LCM of both moduli is bigger than interval width of the no-expanded modulo interval.

Definition 5

Improved simplified addition ($\oplus_2$) and subtraction ($\ominus_2$) of modulo intervals $[a,b]_{m(r)}$ and $[c,d]_{n(s)}$ are defined as follows.

$$
\begin{aligned}
&[a,b]_{m(r)} \oplus_2 [c,d]_{n(s)} \\
&\quad \equiv \ ([a_1,b_1]_{l(u_1)} \oplus_1 [c,d]_{n(s)}) \\
&\quad \cup \ ([a_2,b_2]_{l(u_2)} \oplus_1 [c,d]_{n(s)}) \\
&\qquad \cdots \cup ([a_k,b_k]_{l(u_k)} \oplus_1 [c,d]_{n(s)}),
\end{aligned}
$$

$$
\begin{cases}
u_1 &= a \bmod l, \\
a_1 &= a, \\
b_1 &= \lfloor (b-u_1)/l \rfloor \times l + u_1,
\end{cases}
$$

$$
\begin{cases}
u_k &= u_{k-1} + m, \\
a_k &= a_{k-1} + m, \\
b_k &= b_{k-1} + m \ (k = 2, \cdots, l/m),
\end{cases}
$$

where $l = \mathrm{lcm}(m,n)$.

Addition and subtraction of modulo intervals by Definition 5 can reduce error than addition and subtraction by Definition 2 as proved in the following theorem.

Theorem 1

The following relation holds for modulo intervals $A = [a,b]_{m(r)}$ and $B = [c,d]_{n(s)}$.

$$A \oplus_1 B \supseteq A \oplus_2 B \supseteq A + B,$$

$$A \ominus_1 B \supseteq A \ominus_2 B \supseteq A - B.$$

If $l \leq d - c$, then

$$A \oplus_2 B = A + B,$$

$$A \ominus_2 B = A - B.$$

(Proof.)

First, $A \oplus_1 B \supseteq A \oplus_2 B \supseteq A + B$ is proved as follows where $A = [a,b]_{m(r)}$, $B = [c,d]_{n(s)}$, $k = l/m$.

$$
\begin{aligned}
[a,b]_{m(r)} = [a_1,b_1]_{l(r)} \cup [a_2,b_2]_{l(r+m)} \cup \\
\cdots \ \cup [a_k,b_k]_{l(l-m+r)}\,.
\end{aligned}
$$

$A + B$ is rewrote as follows.

$$
\begin{aligned}
[a,b]_{m(r)} + [c,d]_{n(s)} &= ([a_1,b_1]_{l(r)} + [c,d]_{n(s)}) \\
&\cup \ ([a_2,b_2]_{l(r+m)} + [c,d]_{n(s)}) \\
&\cup \ \cdots \ \cup ([a_k,b_k]_{l(l-m+r)} + [c,d]_{n(s)}).
\end{aligned}
$$

Likewise $A \oplus_2 B$ is rewrote as follows.

$$
\begin{aligned}
[a,b]_{m(r)} \oplus_2 [c,d]_{n(s)} &= ([a_1,b_1]_{l(r)} \oplus_1 [c,d]_{n(s)}) \\
&\cup \ ([a_2,b_2]_{l(r+m)} \oplus_1 [c,d]_{n(s)}) \\
&\cup \ \cdots \ \cup ([a_k,b_k]_{l(l-m+r)} \oplus_1 [c,d]_{n(s)}),
\end{aligned}
$$

$$[a_i,b_i]_{l(r)} \oplus_1 [c,d]_{n(s)} \supseteq [a_i,b_i]_{l(r)} + [c,d]_{n(s)}\,.$$

Hence, $A \oplus_2 B \supseteq A + B$ is held.

On the other side, $A \oplus_1 B \supseteq A \oplus_2 B$ is held from the following relation.

$$
\begin{aligned}
&[a_i,b_i]_{m(r)} \oplus_1 [c,d]_{n(s)} \\
&\qquad \supseteq \ [a_i,b_i]_{l((i-1)\cdot m+r)} \oplus_1 [c,d]_{n(s)}\,.
\end{aligned}
$$

Therefore, $A \oplus_1 B \supseteq A \oplus_2 B \supseteq A + B$ is satisfied. Likewise for subtraction. □

Second, If $l \leq d - c$, then $A \oplus_2 B = A + B$ is satisfied from the relation of Lemma 1: $[a_i,b_i]_{l(r)} \oplus_1 [c,d]_{n(s)} \subseteq [a_i,b_i]_{l(r)} + [c,d]_{n(s)}$ and the property of Definition 2: $[a_i,b_i]_{l(r)} \oplus_1 [c,d]_{n(s)} \supseteq [a_i,b_i]_{l(r)} + [c,d]_{n(s)}$.

Likewise for subtraction. □

Lemma 1

If $l \leq d-c$, then the following relation holds for modulo intervals $A = [a,b]_{m(r)}$ and $B = [c,d]_{n(s)}$.

$$[a_i,b_i]_{l(r)} \oplus_1 [c,d]_{n(s)} \subseteq [a_i,b_i]_{l(r)} + [c,d]_{n(s)}.$$

(Proof.)

Let $\lambda = \frac{l}{m}$.

$$[a,b]_{\lambda m(r)} \oplus_1 [c,d]_{n(s)} \subseteq [a+c,b+d]_{n(r+s)}.$$

When z denote an arbitrary element on the right side of the above expression, $\{z = a + c + n\gamma | z \in [a+c,b+d]_{n(r+s)}\}$,where $0 \leq \gamma \leq \frac{b+d-a-c}{n}$. Each values of α, β are defined by a value of γ as follows.

(Case 1) When $0 \leq \gamma < \frac{b-a}{n}$,

$$
\begin{cases}
\alpha &= \left[\frac{\gamma}{\lambda}\right] \\
\beta &= \gamma - \alpha\lambda = \gamma - \left[\frac{\gamma}{\lambda}\right]\lambda
\end{cases}
$$

where α, β are in the following range respectively.

$$\begin{cases} 0 \leq \alpha < \frac{b-a}{\lambda n} \\ 0 \leq \beta \leq \lambda \leq \frac{d-c}{n}. \end{cases}$$

(Case 2) When $\frac{b-a}{n} \leq \gamma \leq \frac{b+d-a-c}{n}$,

$$\begin{cases} \alpha = \frac{b-a}{\lambda n} \\ \beta = \gamma - \alpha\lambda = \gamma - \frac{b-a}{n} \end{cases}$$

where α, β are as follows respectively.

$$\begin{cases} \alpha = \frac{b-a}{\lambda n} \\ 0 \leq \beta \leq \frac{d-c}{n}. \end{cases}$$

Let x, y denote as follows.

$$\begin{cases} x = \alpha\lambda n + a \\ y = \beta n + c. \end{cases}$$

Then, x, y are included in a multiplicand and a multiplier as follows respectively.

$$\begin{aligned} x &\in [a, b]_{l(r)}, \\ y &\in [c, d]_{n(r)}. \end{aligned}$$

Therefore, Lemma 1 is satisfied. □

The following example computes $[2, 402]_{8(2)} \oplus_1 [22, 604]_{6(4)}$ and $[2, 402]_{8(2)} \oplus_2 [22, 604]_{6(4)}$. Since the LCM of the moduli of $[2, 402]_{8(2)}$ and $[22, 604]_{6(4)}$, namely 2, is less than the interval width of $[22, 604]_{6(4)}$, $[2, 402]_{8(2)} \oplus_2 [22, 604]_{6(4)}$ is equal to $[2, 402]_{8(2)} + [22, 604]_{6(4)}$. Therefore, addition by Definition 3 produces no error for this example.

$$[2, 402]_{8(2)} \oplus_1 [22, 604]_{6(4)} = [24, 1006]_{2(0)},$$

$$\begin{aligned} &[2, 402]_{8(2)} \oplus_2 [22, 604]_{6(4)} \\ &\quad = ([2, 386]_{24(2)} \oplus_1 [22, 604]_{6(4)}) \\ &\quad \cup ([10, 394]_{24(10)} \oplus_1 [22, 604]_{6(4)}) \\ &\quad \cup ([18, 402]_{24(18)}) \oplus_1 [22, 604]_{6(4)}) \\ &\quad = [24, 990]_{6(0)} \cup [32, 998]_{6(2)} \\ &\qquad \cup [40, 1006]_{6(4)}. \end{aligned}$$

3.2 Error Reduction for Multiplication

Another simplified multiplication of modulo intervals is defined as follows.

Definition 6

$$\begin{aligned} &[a, b]_{m(r)} \otimes_2 [c, d]_{n(s)} \\ &\quad \equiv [\min\{ac, ad, bc, bd\}, \\ &\qquad \max\{ac, ad, bc, bd\}]_{\gcd(mn,ms,nr)(rs)}. \end{aligned}$$

Multiplication of modulo intervals defined by Definition 6 can reduce error than multiplication defined by Definition 2 as proved in the following theorem.

Theorem 2

The following relation holds for modulo intervals $A = [a, b]_{m(r)}$ and $B = [c, d]_{n(s)}$.

$$A \otimes_1 B \supseteq A \otimes_2 B \supseteq A \times B.$$

(Proof.)

$A \otimes_1 B$ and $A \otimes_2 B$ are defined as follows by Definition 2 and Definition 6, respectively.

$$\begin{aligned} &A \otimes_1 B \\ &\quad = [\min\{ac, ad, bc, bd\}, \\ &\qquad \max\{ac, ad, bc, bd\}]_{\gcd(m,n)(rs)}, \\ &A \otimes_2 B \\ &\quad = [\min\{ac, ad, bc, bd\}, \\ &\qquad \max\{ac, ad, bc, bd\}]_{\gcd(mn,ms,nr)(rs)}. \end{aligned}$$

Since $\gcd(mn, ms, nr)$ is divisible by $\gcd(m, n)$, $\gcd(mn, ms, nr) \geq \gcd(m, n)$, and both $A \otimes_1 B$ and $A \otimes_2 B$ have the identical residue rs. Hence, $A \otimes_1 B \supseteq A \otimes_2 B$ is proved.

Let x denote an integer in A and y denote an integer in B. x and y can be expressed as $x = mu + r$ and $y = nv + s$ with appropriate integers u and v, respectively. The product of x and y is represented with $g = \gcd(mn, ms, nr)$ as follows.

$$\begin{aligned} x \times y &= (mu + r)(nv + s) \\ &= mnuv + msu + nrv + rs \\ &= (mnuv/g + msu/g + nrv/g)g + rs. \end{aligned}$$

Since mn, ms, nr are divisible by g, $mnuv/g + msu/g + nrv/g$ is an integer. On the other hand, since $a \leq x \leq b$ and $c \leq y \leq d$, the following relation holds.

$$\min\{ac, ad, bc, bd\} \leq x \times y \leq \max\{ac, ad, bc, bd\}$$

These prove $A \otimes_2 B \supseteq A \times B$.

Note that the proof of Theorem 2 proves $A \otimes_1 B \supset A \otimes_2 B$ in case $\gcd(mn, ms, nr) > \gcd(m, n)$ with a little modification. In this case multiplication defined by Definition 6 certainly reduces error than multiplication defined by Definition 2. Moreover, multiplication defined by Definition 6 reduces error drastically in case $r = s = 0$ and mn is greater, since $\gcd(mn, 0, 0) = mn \gg \gcd(m, n)$. An example is shown below.

$$\begin{aligned} [0, 32]_{8(0)} \otimes_1 [16, 48]_{16(0)} &= [0, 1536]_{8(0)}\,, \\ [0, 32]_{8(0)} \otimes_2 [16, 48]_{16(0)} &= [0, 1536]_{128(0)}\,. \end{aligned}$$

4 Algebraic Error Reduction

On program analysis, if an index value is polynomial, it is difficult to analyze data dependence in general.

When data in an array lays out a triangular matrix, a layout of array is often converted from a two-dimensional array to an one-dimensional array in program transformation for saving memory area. Through this process, *induction variable substitution* is performed. Induction variable substitution erases scalar variables which increase or decrease by a constant value through the iterations of loop. After induction variable substitution, indices of array reference are converted to a quadratic polynomial.

The following programs are examples which are an original program and a program converted by induction variable substitution.

Original

```
   k=1
   DO 50 j = 2, 20,3
     DO 60 i = 1, j
        A(3 * k) = ...
         ....    =  A(2 * k - 1)
                 k = k + 3
60   CONTINUE
50 CONTINUE
```

After induction variable substitution

```
   DO 50 j=2,20,3
    DO 60 i=1, j
     A((18*i-9*j+3*j*j-6)/2) = ...
     ... = A(6*i-3*j+j*j-3) ...
60  CONTINUE
50 CONTINUE
```

Although there are a lot of effective algorithms for calculating polynomials on computer, their algorithms do not work effectively on modulo interval operations, since if we do not take notice of operation order, errors of operation can increase. We propose a method of formula transformation for error reduction.

4.1 Polynomials with modulo interval

In this section, we propose the way of error reduction which transform a polynomial expression before performing modulo interval operations. Techniques of transforming a formula can be broken down into two parts: modulus part and interval part.

First, for modulus part, a property of Definition 6 is used to get a modulus approximated to exact sets, that is, a new modulus become a product of both moduli by shifting residues of modulo interval to zero. Second, calculation of interval part can be used methods of estimating amount of rounding errors on the original interval arithmetic. When the polynomial expression $p(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$ is calculated, we can compute just only n times multiplication and n times addition using Horner method. We can also get interval values approximated to exact sets by Horner method on modulo interval. In this section, we will show formula transformation ways on a quadratic polynomial with one variable: $ax^2 + bx + c$.

4.1.1 Modulus part

We propose two methods which transform modulo intervals such as a residue of modulo interval is shifted to zero before performing modulo interval operations. There are the following two methods to shift residue r to

zero on $[L, U]_{m(r)}$, we call them rshift(0) and rshift($-$) respectively.

$$\begin{array}{ll} \text{rshift}(+) & [L-m+r, U-m+r]_{m(r+m-r)}, \\ \text{rshift}(-) & [L-r, U-r]_{m(r-r)}. \end{array}$$

Suppose to calculate modulus and residue on a quadratic polynomial ax^2+bx+c. A modulus of calculation result without transformation and each modulus calculated after transforming a formula by rshift($+$), rshift($-$) and Horner method are as follows.

rshift($+$)

On ax^2+bx+c, let modulo interval transformed by rshift($+$) denote x'. x''s residue is zero. Where $x = x' - (m-r)$, ax^2+bx+c can be transformed as follows.

$$\begin{aligned} & a(x'-m+r)^2 + b(x'-m+r) + c \\ = \; & ax'^2 + (2ar-2am+b)x' \\ + \; & (am-b-2ar)m + (ar+b)r + c. \end{aligned}$$

A modulus of calculation result of this expression is $m \cdot gcd(am, 2ar-2am+b)$.

rshift($-$)

On ax^2+bx+c, let modulo interval transformed by rshift($-$) denote x'. x''s residue is zero. Where $x = x' + r$, ax^2+bx+c can be transformed as follows.

$$\begin{aligned} & a(x'+r)^2 + b(x'+r) + c \\ = \; & ax'^2 + (2ar+b)x' + (ar+b)r + c. \end{aligned}$$

A modulus of calculation result of this expression is $m \cdot gcd(am, 2ar+b)$.

Horner method

ax^2+bx+c can be transformed to $x(ax+b)+c$ by Horner method. A modulus of calculation result of this expression is $m \cdot gcd(am, ar, ar+b)$.

no transformation

When ax^2+bx+c is calculated without transformation, a modulus of calculation result is $m \cdot gcd(am, ar, b)$.

We prove that methods of transformation by rshift($+$) and rshift($-$) are the best way as follows.

Theorem 3

On ax^2+bx+c, if $x = [L, U]_{m(r)}$, then the following relation is satisfied.

$$\begin{array}{lcl} m \cdot gcd(am, & & m \cdot gcd(am, \\ \quad 2ar-2am+b) & \subseteq & \quad ar, ar+b) \\ m \cdot gcd(am, 2ar+b) & & m \cdot gcd(am, ar, b). \end{array}$$

Proof.

Let am, ar, b denote the following expressions, where $g = gcd(am, ar, b)$.

$$\begin{cases} am & = g \cdot k_1 \\ ar & = g \cdot k_2 \\ b & = g \cdot k_3. \end{cases}$$

Then, the following relation is held.

$$\begin{aligned} & gcd(am, 2ar+b) - gcd(am, ar, b) \\ = \; & gcd(g \cdot k_1, g \cdot (2k_2+k_3)) - g \geq 0. \end{aligned}$$

Likewise for the following formulas.
$gcd(am, 2ar+b) - gcd(am, 2ar, b) \geq 0$,
$gcd(am, 2ar-2am+b) - gcd(am, ar, b) \geq 0$,
$gcd(am, 2ar-2am+b) - gcd(am, 2ar, b) \geq 0$.
Hence, Theorem 3 is satisfied. □

4.1.2 Interval part

We propose a way of calculating modulo interval after transforming a formula by Horner method for getting sharper interval width. We show here that a way of transformation by Horner method is the best way by the following simulation.

Simulation

On a quadratic polynomial ax^2+bx+c, let $x = [L, U]_{m(r)}$ denote modulo interval. First, generate original values of a, b, c, d, L, U such as all values are positive integer. Second, generate different expressions by combining each value. Third, we calculate all expressions by 4 ways, operation without transformation, operation after transformation with Horner method, operation after transformation with rshift($+$) and operation after transformation with rshift($-$). Finally, we count the number of getting the sharpest interval among results of 4 ways.

1. Generate 5 random number with one figure, two figures, three figures and four figures each for coefficients a, b, c.

2. Generate 5 random number which are included in $0 \leq L \leq 20$ for L.

3. Generate 5 random number with two figures, three figures and four figures each for U.

4. Generate 300 different expressions by combining each values which are generated by the above 1-3.

5. Generate expressions which have each coefficient as follows. $a, b, c \geq 0$, $a, b \geq 0$ & $c < 0$, $a, c \geq 0$ & $b < 0$, $b, c \geq 0$ & $a < 0$, $a \geq 0$ & $b, c < 0$, $b \geq 0$ & $a, c < 0$, $c \geq 0$ & $a, b < 0$, $a, b, c < 0$. Each negative integer is generated by multiplying each value by -1.

6. Generate expressions which have L and U such as $L \geq 0$ & $U \geq 0$, $L < 0 \& U \geq 0$, $L < 0$ & $U < 0$. Each negative value is generated by normalizing after multiplying each value by -1.

7. Calculate all expressions by 4 ways: operation without transformation, operation after transformation with Horner method, operation after transformation with rshift(+) and operation after transformation with rshift(−).

8. Count the number of getting the sharpest interval among results of 4 ways. When the sharpest interval exist two or more as well, count the number.

Table.1 shows us that a method of transforming by Horner method is the best way.

4.2 Error Reduction for Polynomials

Let us summarize how to reduce error of modulo interval operation for a polynomial expression.

1. Get upper bound and lower bound of modulo interval more exactly.
   (a) Before operation, normalize upper bound value and lower bound value of modulo interval.
   (b) Before operation, transform a polynomial expression by *Horner method*.

2. Find a modulus approximated to exact sets.
   (a) Before operation, transform a polynomial expression by *rshift*.

That is, the most exactly modulo interval is a modulo interval combined the sharpest interval gotten by above 1 with a modulus found by above 2. We show an example of application by above method.

See the calculation results of $5x^2 + 10x + 10$, where $x = [-47, 37]_{6(1)}$. We show two results of calculation with a method of error reduction for polynomials and without that.

Calculation without a method of error reduction

$$\begin{aligned} & 5 * x * x + 10 * x + 10 \\ = \; & 5 \otimes_1 [-47, 37]_{6(1)} \otimes_2 [-47, 37]_{6(1)} \\ & \oplus_2 10 \otimes_1 [-47, 37]_{6(1)} \oplus_1 10 \\ = \; & [-9155, 11425]_{30(25)}. \end{aligned}$$

Calculation using Horner method

Calculating $5x(x + 2) + 10$ which is transformed by Horner method.

$$\begin{aligned} & 5 * x * (x + 2) + 10 \\ = \; & 5 \otimes_1 [-47, 37]_{6(1)} \\ & \otimes_2 ([-47, 37]_{6(1)} \oplus_1 2) \oplus_1 10 \\ = \; & [-9155, 10585]_{30(25)}. \end{aligned}$$

Calculation using rshift(+)

Calculating $5x'^2 - 40x' + 85$ which is transformed, where $x' = [-42, 42]_{6(0)}$ $(x = x' - 5)$.

$$\begin{aligned} & 5 * x' * x' - 40 * x' + 85 \\ = \; & 5 \otimes_1 [-42, 42]_{6(0)} \otimes_2 [-42, 42]_{6(0)} \\ & \ominus_2 40 \otimes_1 [-42, 42]_{6(0)} \oplus_1 85 \\ = \; & [-10415, 10585]_{60(25)}. \end{aligned}$$

$[-9155, 10585]_{60(25)}$ is obtained by combining an interval which is calculated with transformation by Horner method and an modulus and a residue which is calculated with transformation by rshift(+), that is, $[-9155, 10585]_{60(25)}$ is result with the least error.

5 Related Works

Linear Memory Access Descriptor(LMAD) proposed by Peak et al. is a descriptor to

Table 1: The numbers of getting the shortest interval width

case	original	Horner	rshift(−)	rshift(+)
$L \geq 0$ & $U \geq 0$ & $b \geq 0$	271	300	297	13
$L \geq 0$ & $U \geq 0$ & $b < 0$	29	296	279	2
$L < 0$ & $U \geq 0$ & $b \geq 0$	59	275	28	26
$L < 0$ & $U \geq 0$ & $b < 0$	220	276	59	2
$L < 0$ & $U < 0$ & $b \geq 0$	29	300	1	8
$L < 0$ & $U < 0$ & $b < 0$	271	300	22	298

represent access regions. LMAD can operate multi-dimensional array. Modulo interval can describe it likewise LMAD by representing a part of modulo with modulo interval recursively, so both are equivalence on descriptive ability or mutual transformability. LMAD has only set operators: a set union operation, a set subtraction operation and a set intersection operation, while modulo interval has arithmetic operators. It has mathematical properties based on the elementary theory of numbers which can be implemented easily. Arithmetic operators are efficient about scalar analysis or symbolic analysis, while set operators are efficient about array reference analysis or dependence analysis. Modulo interval can be handled as either arithmetic operator or set operator depending on the circumstances, while LMAD cannot handle likewise.

6 Conclusion

The modulo interval has mathematical properties based on the elementary theory of numbers besides similar to properties of the originally interval. Both arithmetic and set operations can be applied to the modulo interval depending on the context. Although the modulo interval requires only the modulus and the residue for additional information to the interval numbers, it can represent values taken by indices of loops or subscripts of array variables more precisely than the original interval numbers. These features of the modulo interval are advantageous to employ the modulo interval for program analysis. However, Most properties of the modulo interval arithmetic are provided as inclusion, that is, an arithmetic operation of modulo intervals of different moduli provides a subset of a modulo interval whose modulus is the GCD of both moduli of operand modulo interval as result. This degrades accuracy of program analysis.

In this paper, the modulo interval, its mathematical properties and some examples applying the modulo interval to program analysis have been described. Moreover we proposed the redefined arithmetic of modulo interval and discussed how to reduce errors on polynomial operations on modulo intervals.

It is a prior future work to establish an algorithm which applies arithmetic and set operations to the modulo intervals without losing accuracy of the calculation results. From the view point of arithmetic of modulo intervals, actually current arithmetic are defined only in terms of application to the data dependence analysis, so various arithmetic of modulo intervals should be examined for another application on hardware or software.

Acknowledgements The authors would like to thank Ms. Kazuko Kambe for valuable comments and discussion. This work is partially supported by "Research for the Future" Program of Japan Society for the Promotion of Science.

References

[1] W. H. Harrison, *Compiler Analysis of the Value Ranges for Variables,* IEEE Trans. on Software Engineering, Vol.SE-3, No.3, May 1977.

[2] A.Inoue, H.Tomiyama, T.Okuma, H.Kanbara, and H.Yasuura, *Language and*

compiler for optimizing datapath widths of embedded systems," IEICE Trans. Fundamentals, vol.E81-A, no.12, pp.2595-2604, Dec 1998.

[3] R. E. Moore, *Interval Analysis.* Prentice-Hall, Englewood Cliffs, New Jersey, 1966.

[4] Ramon E. Moore, Fritz Bibliography on Interval-mathematics Bierbaum , *Methods and Applications of Interval Analysis (SIAM Studies in Applied and Numerical Methematics)*, Society for Industrial & Applied Mathematics, Dec 1979.

[5] Tsuneo Nakanishi, et al., "The Modulo Interval: A Simple and Practical Representation for Program Analysis," Proc. of the 1999 Int. Conf. on Parallel Architectures and Compilation Techniques (PACT '99), pp.91-96, Oct. 1999.

[6] Osamu Ogawa, Kazuyoshi Takagi, Yasufumi Itoh, Shinji Kimura, Katsumasa Watanabe, *Hardware Synthesis from C Programs with Estimation of Bit Length of Variables,*" IEICE Trans. Fundamentals, vol.E82-A, no.11, pp.2338-2346, Nov 1999.

[7] Y. Peak, J. Hoeflinger and D. Padua, *Simplification of Array Access Patterns for Compiler Optimizations,* Proc. of the 1998 ACM SIGPLAN Conf. on Programing Language Design and Implimentation, pp.60-71, May 1998.