計算機アーキテクチャ 139－8
（2000．8．3）

剰余区間演算規則とその応用例

曽山典子 * 中西恒夫 † 加古富志雄 ‡ 福田晃 †

* 奈良女子大学人間文化研究科 ‡ 奈良女子大学理学部
† 奈良先端科学技術大学院大学情報科学研究科

概要

剰余区間演算は実数上の区間に含まれる剰余群を対象とする集合演算と算術演算からなる演算体系として提案された．剰余区間は，計算機で取り扱う離散的かつ循環的な整数の集合を表すのに適した表現であり，その特長はプログラム解析等に用いる上で適している．本研究では，剰余区間演算を使ったプログラム解析の応用例として，自動ベクトル化／並列化コンパイラにおいて行われるデータ依存解析等のプログラム解析，高位論理合成における値域解析への応用例を紹介する。また，基本演算による誤差を削減する演算規則を提案し、その有用性を示す．

An Algorithm of Modulo Interval Arithmetic and Its Applications

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

Abstract

Modulo interval arithmetic is an arithmetic system on sets of integers included in real intervals. 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 its application to data flow analysis for vectorizing/parallelizing compilers or range analysis for hardwaresoftware codesign. Moreover, we introduce a effective method to reduce error of modulo interval arithmetic for polynomials with modulo intervals.

1 はじめに

実数上の区間を対象とする演算体系として「区間演算 (interval arithmetic)」があり [4]，区間演算はプログラム解析にも利用されている [1]．しかし，一般のプログラムにおいては実数は演算のデータとして扱われており，ループの指標や配列変数の添字等には専ら整数が扱われている．このような特徴のあるデータを取り扱うのに適した演算体系として本研究では「剰余区間演算」を提案している [6]．剰余区間は整数の集合を従来の実数上の区間より詳細に表現することができ，従来の実数上の区間に追加された情報は，法と剰余の２つの整数のみであるが，計算機上で取り扱うデータの離散的かつ循環的な整数の集合を表すのに適した表現である．またその状況に応じて算術演算的あるいは，集合演算的に取り扱うことができる．この特長はプログラム解析に用いる上で好ましいものである [7]．本研究では剰余区間演算の応用例として，自動ベクトル化／並列化コンパイラにおいて行われるデータ依存解析等のプログラム解析，高位論理合成における値域解析への応用例を紹介する．また，現在定義されている剰余区間演算規則の多くは，真の演算結果より大きい集合を表すという性質を持ち，解析への応用においては，

解析精度の低下につながることになる．本研究では，現在定義されている剰余区間の基本演算による誤差を削減する演算方法を提案し，データ依存解析への応用例でその有用性を示す．

本論文の構成は以下の通りである．第2章では，現在までに定義された剰余区間演算の基本演算規則とその性質について述べる．第3章で現在までに定義された基本演算規則の問題点について解説し，剰余区間演算の精度を上げる新たな基本演算規則を提案する．第4章では，データ依存解析，高位論理合成における値域解析への応用例を紹介する．第5章で関連研究について紹介し，最後にまとめを述べる．

2 剰余区間演算規則とその基本性質

本章では，これまでに定義されている剰余区間演算の基本演算規則とその性質を示す．なお，本章の定義に関する証明については文献[6]を参照されたい．

2.1 剰余区間演算の基本演算の定義

a, b を実数，m, r を整数とするとき，整数の集合:

$$\{x \in \boldsymbol{Z} | a \leq x \leq b, x = m\alpha + r, \alpha \in \boldsymbol{Z}\}$$

を a 以上 b 以下の法 m，余り r の剰余区間と呼び，$[a,b]_{m(r)}$ と表記する．剰余区間 $[a,b]_{m(r)}$ について，$a \in [a,b]_{m(r)}$ かつ $b \in [a,b]_{m(r)}$ かつ $0 \leq r < m$ であるならば，その剰余区間は正規形であるという．

(剰余区間の加法，減法，乗法に関する定義1)

整数上に定義される加減乗算は，剰余区間に対する演算として以下のように定義される．但し，A, B は任意の剰余区間で，z は任意の整数である．

[定義 1]

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

定義1の演算結果は，剰余区間に含まれる全要素の組み合わせの演算結果の集合である．

(剰余区間の加法，減法，乗法に関する定義2)

定義1による演算を簡単化した剰余区間の加法・減法・乗法を以下の通り定義する．

[定義 2]

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

定義1と定義2の演算について次の性質が成り立つ．但し，A, B は任意の剰余区間である．

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

但し，法の等しい剰余区間どうしの加法，減法については次の性質が成り立つ．

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

(剰余区間と整数との演算に関する定義)

剰余区間と整数との加法，減法，乗法について，以下の通り定義する．但し，z は任意の整数である．

[定義 3]

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

剰余区間と整数との演算については，次の性質が成り立つ．但し，Aは任意の剰余区間，zは任意の整数である．

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

(剰余区間と整数との除算に関する定義)

剰余区間の整数による除法について，以下の通り定義する．但し，zは任意の整数で，$z \neq 0$かつnがzで割り切れるとする．

[定義 4]

$$[a,b]_{m(r)} \oslash_1 z \equiv [a \text{ div } z, b \text{ div } z]_{m \text{ div } z(r \text{ div } z)}$$

尚，$z \neq 0$かつnがzで割り切れない場合は，剰余区間をz倍展開することで，上記の除法の定義で演算が可能となる．z倍展開とは，任意の剰余区間と整数$z(z \geq 0)$について成り立つ以下の性質を利用したものである．

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

(剰余区間と整数乗に関する性質)

剰余区間の整数乗について，mが素数のとき次の性質が成り立つ．

$$[a,b]^m_{m(r)} \subseteq \begin{cases} [a^m, b^m]_{m(r)} & \text{if } a \geq 0 \text{ or } m \text{ is odd} \\ [b^m, a^m]_{m(r)} & \text{if } b \geq 0 \text{ or } m \text{ is even} \\ [0, \max\{a^m, b^m\}]_{m(r)} & \text{otherwise} \end{cases}$$

3 剰余区間演算の誤差問題とその基本的解決法

3.1 剰余区間演算における誤差の問題

現在定義されている剰余区間演算どうしの演算では，法の異なる剰余区間どうしを加減乗した場合，その結果となる剰余区間の法はオペランドの剰余区間の法の最大公約数 (GCD) となる．これは，演算結果の整数集合が，真の結果よりも大きな整数集合となる原因となり，プログラム解析への応用においては解析精度の低下につながる．また，解析に利用する値は剰余区間の上限，下限，法，剰余の４つの値である．つまりこの４つの値をできる限り真の結果を表す値となるように演算を行わなければならない．本研究で言う「誤差を削減する」とは，剰余区間演算の結果が，真の結果の集合を表すことができるように，剰余区間の上限，下限，法，剰余の４つの値を求めることである．

例として定義２の剰余区間演算の加法による誤差を以下に示す．定義１と定義２の演算規則による加算の結果は以下の通りである．

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

定義２による演算結果は，真の結果よりも大きい整数集合になる．この問題は減算も同様である．乗算は法が等しい場合においても誤差が大きくなる．

次節では，定義２の加減法の演算規則で生じる誤差を消去する演算方法を提案する．また，乗算においては演算誤差を完全になくすことは不可能であるが，条件によって誤差を削減させる演算規則を提案する．なお，本章の定義に関する証明については文献 [10] を参照されたい．

3.2 加法・減法における誤差の削減

法が異なる剰余区間の加減における誤差の消去方法として，双方の法の最小公倍数 (LCM) を求め，これを法として一方の剰余区間を展開し，展開後の各剰余区間と他方の剰余区間の間で演算を行う方法を提案する．但し，双方の法の LCM が，展開しない方の剰余区間幅より大きい時は，誤差を完全に消去することはできない．

(誤差を削除する剰余区間演算の加法の演算規則)

新しい剰余区間演算の加法を以下の通り定義する．

[定義 5]

$l = \text{LCM}(m,n)$ とおく

$[a,b]_{m(r)} \oplus_2 [c,d]_{n(s)}$

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

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

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

3.3 乗法における誤差の削減方法

(誤差を削減する剰余区間演算の乗法の演算規則)
新しい剰余区間演算の乗法を以下の通り定義する.
[定義 6]

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

定義 6 の演算規則を使用することによって，法をより真の結果を表す値に近似することができ，剰余区間の乗算を行う時に生じる誤差を削減することができる．特に剰余がともに 0 である時は，法は両方の法の積となり，最も真の結果を表す値に近似できる．しかし，$\gcd(m,n)$ と $\gcd(mn,ms,nr)$ が同じ値である場合も存在し，この場合にはこの規則でも誤差を削減することはできない．

4 剰余区間演算の応用例

剰余区間演算はループの指標や配列変数の添字等に使われている整数を扱うのに適した演算体系である．ここでは，自動ベクトル化／並列化コンパイラにおいて行われるデータ依存解析，さらに値域解析への剰余区間演算の応用例を紹介する．

4.1 データ依存解析への応用

剰余区間演算の応用例として，剰余区間演算を自動ベクトル化／並列化コンパイラにおいて用いるデータフロー解析への適用を紹介する．

制御変数が多項式で表されている場合，依存解析を行うことは一般的に困難である．ここでは，剰余区間演算を使って解析する例を紹介する．配列のデータが三角行列で存在している場合において，メモリを節約するために 2 次元配列を 1 次元配列として参照するプログラムに変形することがある [3]．この変形において誘導変数消去 (ループの中で iteration ごとに一定の値，増減するスカラ変数を消去する) を行い，その結果，配列参照の制御変数が 2 次式で表される．以下は誘導変数消去前と後のプログラムの例である．(誘導変数消去前)

```
        k=1
Loop:   DO j=1, 20
          DO i=1, j
S:         A(3k) = ...
T:         ...   = A(2k-1) ...
           k = k + 1
          ENDDO
        ENDDO
```

(誘導変数消去後)

```
Loop:   DO j=1, 20
          DO i=1, j
S:         A(3*(-j+j*j+2*i)/2)) = ...
T:         ... = A(2*(-j+j*j+2*i)/2)-1) ...
          ENDDO
        ENDDO
```

この誘導変数消去後のプログラム断片のループをイタレーション単位で並列実行できるかどうか解析する．もし $3*(-j+j*j+2*i)/2 = 2*(-j'+j'*j'+2*i')/2) - 1$ を満足するような i,j ならびに i',j' が $[1,20]_{1(0)}$ に含まれれば，このループをイタレーション単位で並列実行することはできない．$3*(-j+j*j+2*i)/2 = 2*(-j'+j'*j'+2*i')/2)-1$ を満足するような i,j ならびに i',j' が $[1,20]_{1(0)}$ に含まれているか否か調べるには，$(-j+j^2+2i)/2+1$ が取り得る値に 0 が含まれるかどうかを検査すればよい．誘導変数消去後のプログラムにおいて，配列 A の制御変数は i と j で表された多項式である．i が取り得る範囲はイタレーション毎に上限が変わるため，解析が困難である．剰余区間演算を使って解析する上で，i の取り得る範囲を j の取り得る範囲と同じとして演算を行う．

剰余区間演算において多項式の値を求める場合問題になるのが，演算誤差である．第 2 章でも示した通り，通常の実数などの演算規則と異なり，不用意に

演算を行うと誤差が累積するおそれがあるからである．多項式をそのまま単純に各項を独立に計算する場合に生じる誤差を，多項式を変形することによって削減することが可能である（文献 [10] 参照）．

多項式を変形せずに演算を行った場合とホーナ法を使って変形後，演算を行った場合との結果は以下の通りである．

・誘導変数消去後の左辺を変形せずに計算した結果

$$\begin{aligned}(-j+j*j+2*i)/2+1 &= \Big(-[1,20]_{1(0)} \oplus_2 [1,20]_{1(0)} \otimes_2 [1,20]_{1(0)} \\ &\quad \oplus_2 2 \otimes_1 [1,20]_{1(0)}\Big) \oslash_1 2 + 1 \\ &= [-7,220]_{1(0)}\end{aligned}$$

この結果から，$0 \in [-7,220]_{1(0)}$ ということがわかり，このループはイタレーション単位で並列実行できないことが結論される．

・誘導変数消去後の左辺をホーナー法を使って変形後計算した結果

$$\begin{aligned}(-j+j*j+2*i)/2+1 = (j(j-1)+2i)/2+1 &= \Big([1,20]_{1(0)} \otimes_2 \big([1,20]_{1(0)} \ominus_1 1\big) \\ &\quad \oplus_2 2 \otimes_1 [1,20]_{1(0)}\Big) \oslash_1 2 + 1 \\ &= [2,211]_{1(0)}\end{aligned}$$

この結果より，$0 \notin [2,211]_{1(0)}$ ということがわかり，このループはイタレーション単位で並列実行できることが結論される．このようにプログラム解析への応用において，剰余区間演算の誤差は重要な問題であり，解析精度の低下につながる．上記の例では，ホーナー法を使って式を変形した後，計算することにより誤差がなくなり，S と T に依存がないことがわかった．

4.2 値域解析への応用

高位論理合成においては，高級言語で記述された変数の有効ビット幅を求め，演算器のビット精度をその有効ビット幅に抑えることにより，使用ゲート数の削減が行われる [2, 8]．ここでは，剰余区間演算を使って，プログラム中に宣言された変数の有効ビット幅を解析する方法を示す．以下は素数を見つけるプログラムである．各変数の有効ビット幅を剰余区間演算を利用して計算する．

```
prime(){
  int p_num, is_p, divsr, N ;
  p_num = 2;
  printf("%d\n", p_num);
  for(N=0, is_p=3; is_p<=999; is_p+=2){
    for(divsr=3; divsr<=31 && divsr<(is_p>>1)
        && N!=is_p; divsr+=2){
      for(N=divsr; N <is_p; N+=divsr);
    }
    if(N!=is_p){
      p_num=is_p;
      printf("%d\n", p_num);
    }
  }
}
```

剰余区間演算を利用して計算した結果，各変数の取り得る範囲，及びそれから得られた有効ビット幅は以下のようになる．

			有効 bit 幅
p_num	=	$[2,997]_{1(0)}$	10bit
is_p	=	$[3,1001]_{2(1)}$	9bit
N	=	$[0,1023]_{1(0)}$	10bit
divsr	=	$[3,31]_{2(1)}$	5bit

法が 2^n の場合，各変数の取る得る値の $n$bit 分は同じ値を取ることになり，有効ビット幅を $n$bit 削減して，設計することが可能になる．is_p と divsr の法は 2 であるので，それぞれ，1bit 減らすことができる．

5 関連研究

Paek らが文献 [9] において提案している Linear Memory Access Descriptor(LMAD) は，配列の参照領域を表現するための記述子である．LMAD では多次元配列を考慮に入れているが，剰余区間においても剰余部をさらに剰余区間で再帰的に表現することで LMAD と同様の記述ができ，両者は相互変換可能かつ記述力は等価である．定義されている演算を比較すると，LMAD は配列参照領域の表現に特化されていることもあり，集合演算のみ（共通部分，交わり，差，簡単化）が定義されているのに対し，剰余区間は整数論で裏付けられるわかりやすく実装の容易な算術演算を整備している．算術演算はスカラ解析やシンボリック解析において，集合演算は配列

参照解析や依存解析において，望まれる演算である．剰余区間演算は状況に応じて算術演算的にあるいは集合演算的に取り扱え，これは LMAD にはない特徴である．

6 まとめ

本稿では，剰余区間演算の基本演算規則とその性質について解説し．基本演算の誤差を削減する演算規則を提案した．さらに自動ベクトル化／並列化コンパイラにおいて行われるデータフロー解析，高位論理合成における値域解析への剰余区間演算の応用例を紹介した．またデータ依存解析の応用において，多項式における剰余区間演算の演算誤差を削減するための方法を適用し，その有用性を示した．

剰余区間は整数の集合を従来の実数上の区間より詳細に表現することができ，その表現方法及び演算方法はシンプルである．それゆえに解析手法をコンパイラ等に実装する上でも簡易になり得ると考える．今後の課題として，コンパイラへの応用の観点からはスカラ解析やシンボリック解析への応用，高位論理合成においては，値域解析への応用を検討したい．また，現時点での剰余区間演算規則は，もともとデータ依存解析のために開発されたものであるため，定義されている演算がデータ依存解析で有用なものに限られている．剰余区間演算をその他のアプリケーションに応用できるようにするべく，剰余区間演算の演算法則の整理を行わなければならない．また，演算結果を正確に保つよう剰余区間演算を系統的に適用するアルゴリズムを開発することも今後の課題として挙げられる．

謝辞　本研究を遂行するにあたり，貴重な助言をいただいた奈良女子大学加古研究室の神戸氏に感謝致します．本研究の一部（中西，福田担当分）は日本学術振興会未来開拓学術研究推進事業（知能情報・高度情報処理分野）JSPS-RFTF96P00505 の援助による．

参考文献

[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] 神戸和子, 加古富志雄,「データ依存解析に関する二次不定方程式の分解」, 情処研報, 2000-ARC-, (submitting)

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

[5] 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.

[6] 中西恒夫, 福田晃,「剰余区間演算の定義とプログラム解析への応用」, JSPP2000 論文集, pp.293–300, Jun 2000.

[7] 中西恒夫, 福田晃,「学振 MIRAI コンパイラへの剰余区間伝播の実装」, 情処研報, 2000-HPC-82, (submitting)

[8] 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.

[9] Y. Paek, 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.

[10] 曽山典子, 中西恒夫, 加古富士雄, 福田晃,「多項式における剰余区間演算誤差削減のための演算規則」, 情報処理学会シンポジウムシリーズ IPSJ Symposium Series Vol.2000, No.5, pp.223-230, Mar 2000