数理モデル化と問題解決 23－7
（1999. 2. 19）

剰余区間演算: データフロー解析のための数学的ツール

中西 恒夫 * 城 和貴 † Constantine D. Polychronopoulos ‡ 福田 晃 *

narafrase@is.aist-nara.ac.jp

* 奈良先端科学技術大学院大学情報科学研究科
† 和歌山大学システム工学部
‡ イリノイ大学アーバナ・シャンペイン校

概要

実数上の区間に対する演算体系として区間演算がある．本来，区間演算は浮動小数点計算の丸め誤差解析を行うために考案されたものであるが，プログラム解析にも応用されている．しかしながら，プログラムのループの指標あるいは配列変数の添字等に用いられるのは専ら整数であり，従来の実数上の区間ではプログラム解析に必要な情報を十分に表現できない．本稿では，区間演算を拡張し，実数上の区間に含まれる整数の集合に対する演算体系，剰余区間演算を定義し，その数学的性質を明らかにする．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への剰余区間演算の応用を検討する．

Modulo Interval Arithmetic: A Mathematical Tool for Data Flow Analysis

Tsuneo Nakanishi * Kazuki Joe † Constantine D. Polychronopoulos ‡ Akira Fukuda *

* Nara Institute of Science and Technology
† Wakayama University
‡ University of Illinois at Urbana-Champaign

Abstract

Interval arithmetic is an arithmetic system on intervals on real numbers. Although interval arithmetic was established to estimate rounding errors caused by floating point calculation originally, it is also applied to program analysis. However, interval arithmetic on real numbers cannot represent essential information for program analysis enough since integer numbers are mostly used for loop indices, array subscripts, *etc.* In this paper we define *modulo interval arithmetic*, an arithmetic system on sets of integers included in real intervals, and describe its mathematical properties. Moreover, we discuss its application to data flow analysis for vectorizing/parallelizing compilers.

1 はじめに

実数上の区間 $[a,b]$，すなわち任意の2実数 $a,b \in \boldsymbol{R}$ $(a \leq b)$ によって定義される実数の集合 $\{x \in \boldsymbol{R} | a \leq x \leq b\}$ に対して定義される四則演算，累乗等からなる演算体系を**区間演算（interval arithmetic）**という．そもそも区間演算は，浮動小数点表現による実数計算を行う代わりに，計算機表現可能な2実数で定義される真の実数値を含む区間に対して演算を行うことによって，実数計算の結果に含まれる丸め誤差の累積分を明らかにするべく考案されたものである [3]．その後，区間演算はプログラム解析にも応

用されるようになる [1].

区間演算をプログラム解析に使うことを考える. 一般のプログラムにおいては, 実数は演算の対象となるデータとして扱われるのが普通であり, ループの指標や配列変数の添字等には専ら整数が用いられる. そのため区間が表現し得る情報はプログラムの解析に用いるには極めて不十分である. 例えば, イタレーション毎の指標の増分が 1 でないループの指標がとり得る値を区間で表現する場合, i) 指標値の下界と上界からなる区間で表現する, あるいは ii) ループの各指標値のみを含む区間の集合で表現することが考えられる. i) は簡潔であるが有用な情報がほとんど得られない. ii) はループが大規模になるにつれて表現に要する記憶量が膨大となり現実的でない. 仮にイタレーション毎の指標の増分が 1 であったとしても, 指標を引数として算出されるループ内の配列変数の添字のとり得る値を算出する場合や, コンパイラによるコード最適化によってループに対して様々の変形を施す場合には同様の問題が生じる.

以上の考察に基づいて本稿では, 区間演算を拡張し, 実数上の区間に含まれる整数の集合に対する演算体系, **剰余区間演算**を提案および定義する. 剰余区間は記号 $[a,b]_{n(p)}$ で表され, 実数上の区間 $[a,b]$ 内の整数のうち, 整数 n で整除した場合の余りが p になるような整数の集合を意味する. 実数上の区間に対して追加された情報は法 n ならびに余り p のみであるが, ループのイタレーションの集合を表現する上で有利な表現であり, また整数論で裏付けられる興味深い数学的性質を多く有する. 本稿ではそれらの数学的性質を明らかにする. さらに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用いるデータフロー解析への適用を検討する.

2 区間演算

任意の 2 実数 $a, b \in \boldsymbol{R}$ $(a < b)$ について定義される実数の集合:

$$\{x \in \boldsymbol{R} | a \leq x \leq b\}$$

を a 以上 b 以下の実数上の (閉) **区間** (**interval**) と呼び, $[a,b]$ と表記する.

四則演算, 累乗など実数上に定義される任意の演算 $\odot$ は, 実数上の区間に対する演算として以下のように再定義される. 但し, A, B は任意の実数上の区間である.

$$A \odot B \quad = \quad \{x = a \odot b | a \in A, b \in B\}$$

この定義に基づき, 実数上の区間の四則演算が以下のように導出される. 但し, 除算は $0 \in [c,d]$ のときに限り定義される.

$$[a,b] + [c,d] = [a+c, b+d]$$

$$[a,b] - [c,d] = [a-d, b-c]$$

$$[a,b] * [c,d] = [\min\{ac, ad, bc, bd\}, \max\{ac, ad, bc, bd\}]$$

$$[a,b] \,/\, [c,d] = [a,b] * [1/d, 1/c]$$

さらに区間の整数乗が以下のように導出される.

$$[a,b]^n = \begin{cases} [1,1] & \text{if } n = 0 \\ [a^n, b^n] & \text{if } a \geq 0 \text{ or } n \text{ is odd} \\ [b^n, a^n] & \text{if } b < 0 \text{ and } n \text{ is even} \\ [0, max\{a^n, b^n\}] & \text{otherwise} \end{cases}$$

加法, 乗法について, 交換則ならびに結合則は成立するが, 分配則は一般には成立しない. 代わりに以下に示す部分分配則が一般に成立する. 但し, A, B, C は任意の実数上の区間である.

$$A * (B + C) \subseteq A * B + A * C$$

3 剰余区間演算

本節では, 剰余区間演算を定義し, その数学的性質を明らかにする.

3.1 定義

a, b を実数, n, p を整数とするとき, 整数の集合:

$$\{q \in \boldsymbol{Z} | a \leq q \leq b, q = nm + p, m \in \boldsymbol{Z}\}$$

を a 以上 b 以下の**法** n, **余り** p **の剰余区間**と呼び, $[a,b]_{n(p)}$ と表記する. 例を以下に示す.

$$\begin{aligned} [7,21]_{5(4)} &= \{9, 14, 19\} \\ [-6,10]_{-4(2)} &= \{-6, -2, 2, 6, 10\} \\ [-7,9]_{4(-3)} &= \{-7, -3, 1, 5, 9\} \\ [3,7]_{2(3)} &= \{3, 5, 7\} \end{aligned}$$

剰余区間 $[a,b]_{n(p)}$ について，$a \in [a,b]_{n(p)}$ かつ $b \in [a,b]_{n(p)}$ かつ $0 \leq p < n$ であるならば，その剰余区間は**正規形**であるという．

3.2 基本演算

四則演算，累乗など整数上に定義される任意の演算 $\odot$ は，剰余区間に対する演算として以下のように再定義される．但し，A，B は任意の剰余区間，z は任意の整数である．

$$\begin{aligned} A \odot B &= \{x = a \odot b | a \in A, b \in B\} \\ A \odot z &= \{x = a \odot z | a \in A\} \\ z \odot B &= \{x = z \odot b | b \in B\} \end{aligned}$$

剰余区間の加法，減法，乗法，ならびに整数乗について以下の性質が成り立つ．

性質 1 （加法・減法・乗法に関する性質） 剰余区間どうしの加法，減法，乗法について次の性質が成り立つ．

$$\begin{aligned} &[a,b]_{m(p)} + [c,d]_{n(q)} \\ &\quad \subseteq [a+c, b+d]_{\gcd(m,n)(p+q)} \\ &[a,b]_{m(p)} - [c,d]_{n(q)} \\ &\quad \subseteq [a-d, b-c]_{\gcd(m,n)(p-q)} \\ &[a,b]_{m(p)} * [c,d]_{n(q)} \\ &\quad \subseteq [\min\{ac, ad, bc, bd\}, \\ &\qquad \max\{ac, ad, bc, bd\}]_{\gcd(m,n)(pq)} \end{aligned}$$

（証明）$[a,b]_{m(p)}$，$[c,d]_{n(q)}$ の任意の 1 要素をそれぞれ s，t とする．定義より 2 つの整数 u，v を用いて $s = mu + p$，$t = nv + q$ と表される．

s と t の和は $g = \gcd(m,n)$ とすると以下のように表される．

$$\begin{aligned} s + t &= (mu + p) + (nv + q) \\ &= (mu/g + nv/g)g + (p + q) \end{aligned}$$

g は m と n の最大公約数であるから $m/g \in \boldsymbol{Z}$，$n/g \in \boldsymbol{Z}$ である．

一方，$a \leq s \leq b$，$c \leq t \leq d$ であるから，

$$a + c \leq s + t \leq b + d$$

である．

ゆえに本性質の成立が証明される．減法，乗法についても同様である． □

性質 1 において一般的には等号は成立しないが，次の特別な場合においては成立する．

性質 2 （法の等しい剰余区間の加法・減法） 法の等しい正規形の剰余区間どうしの加法，減法について次の性質が成り立つ．

$$\begin{aligned} [a,b]_{n(p)} + [c,d]_{n(q)} &= [a+c, b+d]_{n(p+q)} \\ [a,b]_{n(p)} - [c,d]_{n(q)} &= [a-d, b-c]_{n(p-q)} \end{aligned}$$

（証明）正規形の剰余区間であるから，適当な整数 $u_a, u_b, u_c, u_d \in \boldsymbol{Z}$ を用いて $a = nu_a + p$，$b = nu_b + p$，$c = nu_c + q$，$d = nu_d + q$ と書き表すことができる．ここで，$a \leq b$，$c \leq d$ であるから $u_a \leq u_b$，$u_c \leq u_d$ である．

$[a+c, b+d]_{n(p+q)}$ の任意の 1 要素を $s = nu + p + q$（但し，$u \in \boldsymbol{Z}$）とすると，$u_a + u_c \leq u \leq u_b + u_d$ である．$u = u_a + u_c + \Delta$ と書き表した場合，$0 \leq \Delta \leq (u_b + u_d) - (u_a + u_c)$ である．$0 \leq \Delta < u_b - u_a$ のとき，$s = \{n(u_a + \Delta) + p\} + (nu_c + q)$ と変形すると，明らかに $n(u_a + \Delta) + p \in [a,b]_{n(p)}$，$nu_c + q \in [c,d]_{n(q)}$ である．$u_b - u_a \leq \Delta \leq (u_b + u_d) - (u_a + u_c)$ のとき，$s = (nu_b + p) + \{n(u_a + u_c - u_b + \Delta) + q\}$ と変形すると，明らかに $nu_b + p \in [a,b]_{n(p)}$，$n(u_a + u_c - u_b + \Delta) + q \in [c,d]_{n(q)}$ である．ここで，

$$[a+c, b+d]_{n(p+q)} \quad \subseteq \quad [a,b]_{n(p)} + [c,d]_{n(q)}$$

が証明される．

性質 1 より，$[a,b]_{n(p)} + [c,d]_{n(q)} \subseteq [a+c, b+d]_{n(p+q)}$ であるから，

$$[a,b]_{n(p)} + [c,d]_{n(q)} \quad = \quad [a+c, b+d]_{n(p+q)}$$

が証明される．

減法に関しても同様に証明される． □

剰余区間の整数乗については次の性質が成り立つ．

性質 3 （整数乗に関する性質） 剰余区間の整数乗について，n が素数のとき次の性質が成り立つ．

$$[a,b]^n_{n(p)} \subseteq \begin{cases} [1,1]_{n(p)} & \text{if } n = 0 \\ [a^n, b^n]_{n(p)} & \text{if } a \geq 0 \text{ or } n \text{ is odd} \\ [b^n, a^n]_{n(p)} & \text{if } b < 0 \text{ and } n \text{ is even} \\ [0, \max\{a^n, b^n\}]_{n(p)} & \text{otherwise} \end{cases}$$

（証明）第2章で述べた区間の整数乗に関する性質，ならびにフェルマの定理（Fermat's Theorem）から証明される．フェルマの定理は，pが任意の素数のとき任意の整数aについて$a^p \equiv a \pmod{p}$が成り立つことを示す定理で，公開鍵暗号の基本原理として知られている．フェルマの定理の詳細については適当な整数論の文献を参照されたい． □

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．性質1の場合とは異なり，これらの性質は等式として表される．

性質 4（**剰余区間と整数との演算に関する性質**）剰余区間と整数との加法，減法，乗法について次の性質が成り立つ．

$$\begin{aligned}
[a,b]_{n(p)} + z &= [a+z, b+z]_{n(p+z)} \\
z + [a,b]_{n(p)} &= [z+a, z+b]_{n(z+p)} \\
[a,b]_{n(p)} - z &= [a-z, b-z]_{n(p-z)} \\
z - [a,b]_{n(p)} &= [z-b, z-a]_{n(z-p)} \\
[a,b]_{n(p)} * z &= [a*z, b*z]_{nz(pz)} \\
z * [a,b]_{n(p)} &= [a*z, b*z]_{nz(pz)}
\end{aligned}$$

（証明）定義によれば$[a,b]_{n(p)} + z$の結果は次の集合である．

$$\{x = m + z | m \in [a,b]_{n(p)}\}$$

この集合は以下のように変形される．

$$\begin{aligned}
&\{x = m + z | m \in [a,b]_{n(p)}\} \\
&= \{x = m + z | m = nu + p, u \in \boldsymbol{Z}, a \le m \le b\} \\
&= \{x | x = nu + p + z, u \in \boldsymbol{Z}, a + z \le x \le b + z\} \\
&= [a+z, b+z]_{n(p+z)}
\end{aligned}$$

これにより剰余区間と整数との加法に関する性質が証明される．減法，乗法についても同様である． □

3.3 集合演算

剰余区間を単に整数の集合とみなせば，剰余区間に対して集合演算を定義することができる．

性質 5（**剰余区間の共通部分**）任意の2つの剰余区間$[a,b]_{m(p)}$，$[c,d]_{n(q)}$の共通部分は

$$\begin{aligned}
&[a,b]_{m(p)} \cap [c,d]_{n(q)} \\
&= \{x \in \boldsymbol{Z} \,|\, \max\{a,c\} \le x \le \min\{b,d\}, \\
&\quad x \equiv p \pmod{m}, x \equiv q \pmod{n}\}
\end{aligned}$$

で与えられる．

（証明）省略．剰余区間の定義より自明． □

性質5で与えられる剰余区間の共通部分を求めるには，連立合同方程式：

$$\begin{cases} x \equiv p \pmod{m} \\ x \equiv q \pmod{n} \end{cases}$$

を解く必要がある．このような連立合同方程式の解は中華剰余定理（Chinese Remainder Theorem）により求められる．法の等しい剰余区間の共通部分については次の性質6により簡単に求めることができる．

性質 6（**法の等しい剰余区間の共通部分**）任意の2つの法の等しい剰余区間$[a,b]_{n(p)}$，$[c,d]_{n(q)}$の共通部分は次式で与えられる．

$$[a,b]_{n(p)} \cap [c,d]_{n(q)} = \begin{cases} [\max\{a,c\}, \min\{b,d\}]_{n(p)} & \text{if } p = q \\ \emptyset & \text{if } p \neq q \end{cases}$$

（証明）自明につき省略． □

4 データフロー解析への応用

本節では前節で述べた剰余区間演算の応用例として，自動ベクトル化／並列化コンパイラのデータフロー解析における剰余区間演算の利用を検討する．

4.1 配列参照解析

リモートメモリアクセスあるいはプロセッサ間通信のオーバーヘッドの大きい，分散共有メモリ型あるいはメッセージパッシング型の並列計算機では，配列変数を分散されたメモリモジュールにいかに分散配置するかによってプログラムの実行時間が大きく変わる．これらの並列計算機では，各プロセッサが頻繁に参照する配列要素をそのローカルメモリに配置することが，プログラムの実行性能保証上極めて重要である．そのためプログラム中の個々のループにおいて，配列変数のどの要素が参照されるか解析する必要がある．

図1のループL1において，指標のとり得る値の集合は$[1,1000]_{1(0)}$である．よってL1で参照される

```
L1: DO I=1, 1000
      ... = A(3I-1) + A(3I) + A(3I+1)
    EndDO
L2: DO I=1, 999, 2
      ... = A(3I-1) + A(3I) + A(3I+1)
    EndDO
```

図 1: 配列参照解析

配列変数 A の要素の添字の集合は以下のように求められる.

$$
\begin{aligned}
&(3*[1,1000]_{1(0)}-1)\cup(3*[1,1000]_{1(0)})\\
&\cup(3*[1,1000]_{1(0)}+1)\\
&= ([3,3000]_{3(0)}-1)\cup([3,3000]_{3(0)})\\
&\quad\cup([3,3000]_{3(0)}+1)\\
&= [2,2999]_{3(2)}\cup[3,3000]_{3(0)}\cup[4,3001]_{3(1)}\\
&= [2,3001]_{1(0)}
\end{aligned}
$$

すなわち, A(2:3001) が L1 において参照される.

同じくループ L2 において, 指標のとり得る値の集合は $[1,999]_{2(1)}$ である. ゆえに L2 で参照される配列変数 A の要素の添字の集合は以下のように求められる.

$$
\begin{aligned}
&(3*[1,999]_{2(1)}-1)\cup(3*[1,999]_{2(1)})\\
&\cup(3*[1,999]_{2(1)}+1)\\
&= ([3,2997]_{6(3)}-1)\cup([3,2997]_{6(3)})\\
&\quad\cup([3,2997]_{6(3)}+1)\\
&= [2,2996]_{6(2)}\cup[3,2997]_{6(3)}\cup[4,2998]_{6(4)}
\end{aligned}
$$

すなわち, A(2:4),A(8:10),A(14:16),...,A(2996:2998) が参照される.

4.2 データ依存解析

プログラム中のタスク T_1, T_2 について, T_1 がある変数に書き込んだ値を T_2 が後から参照する場合, T_1 と T_2 は同時に実行することはできない. このようなデータ参照に伴うプログラム中のタスクの実行順序に関する制約条件をデータ依存とよぶ. T_1 において書き込みが行われる変数の集合を $O(T_1)$, T_2 によって読み込まれる変数の集合を $I(T_2)$ とする. このとき, $O(T_1)\cap I(T_2)=\emptyset$ ならば T_1 と T_2 の間には依存が存在せず, 両タスクを並列実行することができる (Bernstein の条件) [2].

図 2 のプログラム断片のループ L3 全体とループ L4 全体を並列実行できるかどうか解析する. L3 において参照される配列変数 A の要素の添字の集合を剰余区間で表現すると次のようになる (全小節 L2 の場合と同じ).

$$[4,1000]_{4(0)}$$

同様に L4 において参照される配列変数 A の要素の添字の集合を剰余区間で表現すると次のようになる.

$$[2,2996]_{6(2)}\cup[3,2997]_{6(3)}\cup[4,2998]_{6(4)}$$

これらの剰余区間の共通部分は以下のように空集合でないことがわかる.

$$
\begin{aligned}
&[4,1000]_{4(0)}\\
&\quad\cap\ ([2,2996]_{6(2)}\cup[3,2997]_{6(3)}\cup[4,2998]_{6(4)})\\
&\quad= [8,992]_{12(8)}\cup[4,1000]_{12(4)}\\
&\quad\neq \emptyset
\end{aligned}
$$

すなわち, L3 と L4 を並列に正しく実行できる保証はない.

次に図 3 のプログラム断片のループ L5 全体とループ L6 全体が並列に実行できるかどうかを解析する. それぞれのループで参照される A の要素の添字を剰余区間で表し, その共通部分を求めると以下のようになる.

$$[4,1000]_{4(0)}\cap[3,2997]_{6(3)} \quad= \quad\emptyset$$

すなわち, L5 と L6 の間にはデータ依存がなく, 両ループを正しく並列実行できることがわかる. 一方, 剰余区間の代わりに実数上の区間を用いてそれぞれのループで参照される A の要素を表現してデータ依存解析を行った場合, 以下のように L5 と L6 の間にはデータ依存があるものと誤判定されてしまう.

$$[4,1000]\cap[3,2997]=[4,1000]\neq\emptyset$$

最後に図 4 のプログラム断片のループ L7 をイタレーション単位で並列実行できるかどうかを解析する. もし, $3i=2i'-1$ を満足するような i ならびに i' が $[4,1000]_{4(0)}$ に含まれれば, 文 S が $I=i$ の時に書き込んだ配列変数 A の要素を文 T が $I=i'$ の時に読み込む, あるいは T が $I=i'$ の時に読み込んだ A の要素に S が $I=i$ の時に書き込むことを意味す

る．このような場合，L7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．$3i = 2i' - 1$を満足するようなiならびにi'が$[4,1000]_{4(0)}$に含まれるかどうか検査するには，$3i - 2i' + 1$がとり得る値に0が含まれるかどうかを検査すればよい．iならびにi'のとり得る値の集合は$[4,1000]_{4(0)}$であるから，$3i - 2i' + 1$のとり得る値の集合は以下の通りとなる．

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

明らかに$0 \notin [-1987,2993]_{4(1)}$である．すなわち，L7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．

```
L3: DO I=4, 1000, 4
      A(I) = ...
    EndDO
L4: DO I=1, 999, 2
      ... = A(3I-1) + A(3I) + A(3I+1)
    EndDO
```

図 2: ループ例

```
L5: DO I=4, 1000, 4
      A(I) = ...
    EndDO
L6: DO I=1, 999, 2
      ... = A(3I)
    EndDO
```

図 3: ループ例

```
L7: DO I=4, 1000, 4
S:    A(3I) = ...
T:    ... = A(2I-1)
    EndDO
```

図 4: ループ例

5 まとめ

本稿では，実数上の区間演算を拡張し，実数上の区間に含まれる整数の集合に対する演算体系，剰余区間演算を定義し，その数学的性質について述べた．さらに本稿では，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フロー解析への剰余区間演算のいくつかの応用例を示した．

剰余区間は区間の部分集合であるため，その演算は区間に対する演算とよく似た数学的性質を有する．また剰余区間は整数の集合を従来の実数上の区間よりも詳細に表現することができる．剰余区間を定義するにあたって実数上の区間に追加された情報は法と余りのみであるが，これだけの拡張によってループの指標や配列変数の添字のとり得る値が効率よく表現される．また，剰余区間演算は整数論で裏付けられる興味深い数学的性質を多く有し，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うことができる．以上に述べた特長はいずれもプログラム解析に用いる上で好ましいものである．

剰余区間演算の基本演算に関する性質は，等式としてではなく，左辺が右辺の部分集合となるような形で与えられている．また，法の異なる剰余区間に対して演算を行った場合，その結果はそれらの法の最大公約数を法とする剰余区間の部分集合として与えられる．これらはいずれも剰余区間演算の結果となる整数集合を，真の結果よりも大きなものとする原因となる．剰余区間演算をプログラム解析に用いた場合，これは解析結果が不正確なものとなることを意味する．剰余区間演算の結果ができるだけ真の結果に近くなるように保つには，演算の手順を最適化する必要がある．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の開発が，今後の課題として挙げられる．

参考文献

[1] M. R. Haghighat, *Symbolic Analysis for Parallelizing Compilers*, Kluwer Academic Publishers, 1995.

[2] K. Hwang, *Advanced Computer Architecture: Parallelism, Scalability, Programmability*, McGraw-Hill, 1993.

[3] D. E. Knuth, *The Art of Computer Programming*, Vol.II, Addison-Wesley, 2nd edition, 1981.