数式処理 *J.JSSAC (2003)*
Vol. 9, No. 3, pp. 36 - 49

論文

剰余区間の拡張とデータ依存解析への応用

曽山 典子* 加古 富志雄†

天理大学人間学部 奈良女子大学理学部

(Received 2002/4/2 Revised 2002/10/8)

Abstract

データ依存解析で扱われる数は、ループ変数や配列の添字変数等が取り得る値であり、規則的な性質を持った整数の集合である。剰余区間はこのような整数の集合を表すのに適した表現であり、その演算規則はシンプルでコンパイラへの実装も容易である。しかし、剰余区間演算は集合の要素間で演算した結果の集合として定義されており、剰余区間同士の演算結果は一般的には剰余区間では表すことは不可能で、近似的にしか表せない。近似した演算規則による結果は、厳密な演算結果より大きい集合を表す性質を持つため、演算を繰り返すと演算誤差が累積することになる。本論文では、剰余区間演算による誤差を削減するため、剰余区間を一般化した拡張剰余区間とその演算規則を考える。また拡張剰余区間をデータ依存解析に応用し、その有効性を示す。

1 はじめに

コンピュータ上で行う数値計算は実数値を対象とすることが多いが、規則的な整数の集合を計算の対象とする場合もある。例えば、ループの指標や配列変数の添え字等には専ら整数が扱われており、データ依存解析などにおいては、これらの整数は規則的な性質を持った整数の集合として取り扱われている。このような性質を持ったデータを取り扱うのに適した数として、「剰余区間」が考えられた [4]。

剰余区間は $[a,b]_{m(r)}$ と表現し、a 以上 b 以下に含まれる整数のうち、整数 m（法）で整除した場合の剰余が r となる整数の集合を表す。剰余区間演算は、剰余区間で表した集合の要素間で演算した結果の集合として算術演算を定義しており、剰余区間同士の演算結果は一般的には剰余区間では表すことは不可能で、近似的にしか表せない。近似した結果を求める演算規則では、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となる。それゆえ文献 [4] で定義されている剰余区

*soyama@sta.tenri-u.ac.jp
†kako@ics.nara-wu.ac.jp

間演算規則では、厳密な演算結果より大きい集合を表すという性質を持つことになり、演算回数が増えれば演算誤差が累積することになる。

本論文では、剰余区間演算による演算誤差を削減するため、剰余区間を一般化した「拡張剰余区間」とその演算規則を考える。そして、これをデータ依存解析に応用した例を示し、拡張剰余区間とその演算規則が有効であることを示す。

本論文の構成は次のとおりである。第 2 章では拡張剰余区間を定義し、その基本演算規則について述べる。また拡張剰余区間の次数の簡約について述べる。そして第 3 章で拡張剰余区間をデータ依存解析に応用した例を示し、最後にまとめる。

2 拡張剰余区間と基本演算規則

ここでは、中西と福田 [4] によって考えられた剰余区間を一般化し、規則性を持った整数の集合をより詳細に記述することができるものとして「拡張剰余区間」を定義する。また、拡張剰余区間に対する加減乗算等の基本的な演算について調べる。

2.1 拡張剰余区間

拡張剰余区間は、次の式で表されるデータの集合を表現する。

$$\{x \mid x = l + \sum_{i=1}^{n} \alpha_i \rho_i\}.$$

ただし、$l, \alpha_i, \beta_i, \rho_i \in \boldsymbol{Z}$, $0 \leq \rho_i \leq \beta_i$ である。α_i, β_i をそれぞれ i 番目のステップ、およびカウントと呼ぶ。α_i がすべて正の整数である場合、l, u は区間の下限値および上限値であり、$u = l + \sum_{i=1}^{n} \alpha_i \beta_i$ となる。このような整数の集合を

$$[l, u]^{\alpha_1, \alpha_2, \cdots, \alpha_n}_{\beta_1, \beta_2, \cdots, \beta_n}$$

と表し、(n 次の) 拡張剰余区間と呼ぶことにする。ここで、ステップとカウントの組の個数 n を拡張剰余区間の次数と呼ぶ。

1 次の拡張剰余区間は、剰余区間に等しい。実際、正規形である剰余区間 $[l, u]_{m(r)}$ (ただし、$l, u \in [l, u]_{m(r)}$) は 1 次の拡張剰余区間として、

$$[l, u]^{m}_{(u-l)/m}$$

と表される。

1. i と j の入れ換えに関して不変である。

$$[l, u]^{\alpha_1, \cdots, \alpha_i, \cdots, \alpha_j, \cdots, \alpha_n}_{\beta_1, \cdots, \beta_i, \cdots, \beta_j, \cdots, \beta_n} = [l, u]^{\alpha_1, \cdots, \alpha_j, \cdots, \alpha_i, \cdots, \alpha_n}_{\beta_1, \cdots, \beta_j, \cdots, \beta_i, \cdots, \beta_n}.$$

2. $\alpha_i = \alpha_j$ のとき、一つにまとめる事ができる。

$$[l,u]^{\alpha_1,\cdots,\alpha_i,\cdots,\alpha_j,\cdots,\alpha_n}_{\beta_1,\cdots,\beta_i,\cdots,\beta_j,\cdots,\beta_n} = [l,u]^{\alpha_1,\cdots,\alpha_i,\cdots,\alpha_n}_{\beta_1,\cdots,\beta_i+\beta_j,\cdots,\beta_n}.$$

3. 負のステップをもつ拡張剰余区間に対して、これを正のステップの拡張剰余区間で表す事が出来る。

$$[l,u]^{\alpha_1,\cdots,-\alpha_i,\cdots,\alpha_n}_{\beta_1,\cdots,\beta_i,\cdots,\beta_n} = [l-\alpha_i\beta_i\ ,\ u-\alpha_i\beta_i]^{\alpha_1,\cdots,\alpha_i,\cdots,\alpha_n}_{\beta_1,\cdots,\beta_i,\cdots,\beta_n}.$$

4. ステップが 0 であるものを取り除いても不変である。

$$[l,u]^{\alpha_1,\cdots,0,\cdots,\alpha_n}_{\beta_1,\cdots,\beta_i,\cdots,\beta_n} = [l,u]^{\alpha_1,\cdots,\alpha_{i-1},\alpha_{i+1},\cdots,\alpha_n}_{\beta_1,\cdots,\beta_{i-1},\beta_{i+1},\cdots,\beta_n}.$$

5. カウントが 0 であるものを取り除いても不変である。

$$[l,u]^{\alpha_1,\cdots,\alpha_i,\cdots,\alpha_n}_{\beta_1,\cdots,0,\cdots,\beta_n} = [l,u]^{\alpha_1,\cdots,\alpha_{i-1},\alpha_{i+1},\cdots,\alpha_n}_{\beta_1,\cdots,\beta_{i-1},\beta_{i+1},\cdots,\beta_n}.$$

拡張剰余区間 $[l,u]^{\alpha_1,\alpha_2,\cdots,\alpha_n}_{\beta_1,\beta_2,\cdots,\beta_n}$ において、$0<\alpha_1<\alpha_2<\cdots<\alpha_n$, $\beta_i>0\ (1\le i\le n)$ を満たすとき、その拡張剰余区間は正規形であるという。

2.2 拡張剰余区間の包含関係

拡張剰余区間に対して、次の関係が成り立つ。なお、本論文では、$A\subseteq B$ は A は B の部分集合であることを表し、$A\subset B$ は A は B の真の部分集合であることを表すとする。

1. 任意の拡張剰余区間 $A=[l,u]^{\alpha}_{\beta}$ と、α の約数 $g>1$ に対して、

$$[l,u]^{\alpha}_{\beta}\subseteq[l,u]^{\alpha/g}_{g\beta}.$$

2. 任意の拡張剰余区間 $A=[l,u]^{\alpha}_{\beta}$ と任意の整数 $k>1$ に対して

$$[l,u']^{k\alpha}_{\beta'}\subseteq[l,u]^{\alpha}_{\beta}.$$

ここで、$\beta'=\lfloor\beta/k\rfloor$、$u'=u+\alpha(k\beta'-\beta)$ とおいた。

3. 任意の拡張剰余区間 $A=[l,u]^{\alpha_1,\alpha_2}_{\beta_1,\beta_2}$ に対して $\alpha_2>\alpha_1$ のとき、

$$[l,u]^{\alpha_1,\alpha_2}_{\beta_1,\beta_2}\subseteq[l,u]^{\alpha_1,\alpha_2-\alpha_1}_{\beta_1+\beta_2,\beta_2}.$$

4. 任意の拡張剰余区間 $A=[l,u]^{\alpha_1,\alpha_2}_{\beta_1,\beta_2}$ に対して、$\beta_1>\alpha_2$ の時、$k+1>\beta_1/\alpha_2\ge k$ を満たす整数を k と置くと、

$$[l,u']^{\alpha_1,\alpha_2}_{\alpha_2,\beta_2+(k-1)\alpha_1}\subseteq[l,u]^{\alpha_1,\alpha_2}_{\beta_1,\beta_2}.$$

ただし、$u'=u-\alpha_1(\beta_1-\alpha_2k)$ とおいた。

特に、$\beta_1=k\alpha_2$ の時、

$$[l,u]^{\alpha_1,\alpha_2}_{\alpha_2,\beta_2+(k-1)\alpha_1}=[l,u]^{\alpha_1,\alpha_2}_{\beta_1,\beta_2}.$$

5. 任意の拡張剰余区間 $A = [l, u]^{\alpha_1,\alpha_2}_{\beta_1,\beta_2}$ に対して、$\alpha_1 \mid \alpha_2$、すなわちある整数 k が存在して $\alpha_2 = k\alpha_1$、を満たす場合

$$[l, u]^{\alpha_1,\alpha_2}_{\beta_1,\beta_2} \subseteq [l, u]^{\alpha_1}_{\beta_1+k\beta_2}.$$

ここで、等号が成り立つのは $\beta_1 \geq k - 1$ の時である。

6. 任意の拡張剰余区間 $A = [l, u]^{\alpha_1,\alpha_2}_{\beta_1,\beta_2}$ に対して、γ を α_1 と α_2 の公約数とすると、

$$[l, u]^{\alpha_1,\alpha_2}_{\beta_1,\beta_2} \subseteq [l, u]^{\gamma}_{(\alpha_1\beta_1+\alpha_2\beta_2)/\gamma}.$$

等号が成り立つのは $\gamma = \alpha_1$ かつ $\beta_1 \geq \alpha_2/\gamma - 1$ 、あるいは $\gamma = \alpha_2$ かつ $\beta_2 \geq \alpha_1/\gamma - 1$ が成り立つ場合であり、かつその時に限る。

7. 任意の拡張剰余区間 $A = [l, u]^{\alpha_1,\alpha_2}_{\beta_1,\beta_2}$ に対して、γ を α_1 と α_2 の公倍数とすると、

$$[l, u']^{\gamma}_{\beta'} \subseteq [l, u]^{\alpha_1,\alpha_2}_{\beta_1,\beta_2}.$$

ここで、$\beta' = \lfloor(\alpha_1\beta_1 + \alpha_2\beta_2)/\gamma\rfloor$、$u' = l + \gamma\beta' = u + \gamma\beta' - \alpha_1\beta_1 - \alpha_2\beta_2$ とおいた。等号が成り立つのは $\gamma = \alpha_1 = \alpha_2$ の場合であり、かつその時に限る。

これらの関係式の証明はほとんど自明である。ここでは、6 についてのみ証明を与える。

証明

拡張剰余区間 A を $[a, b]^{\alpha_1,\alpha_2}_{\beta_1,\beta_2}$ とし、γ を α_1 と α_2 の公約数とする。A の任意の要素 x について

$$x = a + \alpha_1 i + \alpha_2 j, \quad 0 \leq i \leq \beta_1, 0 \leq j \leq \beta_2$$

と表される。これを変形して、

$$x = a + \gamma(i\frac{\alpha_1}{\gamma} + j\frac{\alpha_2}{\gamma}).$$

今、

$$0 \leq (i\frac{\alpha_1}{\gamma} + j\frac{\alpha_2}{\gamma}) \leq \frac{\alpha_1\beta_1 + \alpha_2\beta_2}{\gamma}$$

したがって、

$$x \in [a, b]^{\gamma}_{\beta}.$$

ただし、$\beta = \frac{\alpha_1\beta_1+\alpha_2\beta_2}{\gamma}$.

逆に、任意の y:

$$y \in [a, b]^{\gamma}_{\beta}$$

ただし $\beta = \frac{\alpha_1\beta_1+\alpha_2\beta_2}{\gamma}$、に対して、$y \in [a, b]^{\alpha_1,\alpha_2}_{\beta_1,\beta_2}$ となるのは、 $y = a + k\gamma, 0 \leq k \leq \beta$ と置いて、

$$k = i\frac{\alpha_1}{\gamma} + j\frac{\alpha_2}{\gamma}$$

を満たす i, j が $0 \leq i \leq \beta_1, 0 \leq j \leq \beta_2$ の範囲に存在する時である。

しかしながら、$k=1$, $\frac{\alpha_1}{\gamma} \neq 1$, $\frac{\alpha_2}{\gamma} \neq 1$ の場合を考えると、この式を満足する i, j は存在しない。したがって、この定理の逆は一般には成り立たない。

逆が成り立つのは、$\gamma = \alpha_1$ かつ $\beta_1 \geq \alpha_2/\gamma - 1$ 、あるいは $\gamma = \alpha_2$ かつ $\beta_2 \geq \alpha_1/\gamma - 1$ が成立する場合である。

∎

なお、これらの包含関係について、一般の拡張剰余区間 $[a,b]^{\alpha_1,\alpha_2,\cdots,\alpha_n}_{\beta_1,\beta_2,\cdots,\beta_n}$ に対しても同様な関係式が成り立つ。

2.3 拡張剰余区間の演算

ここでは、拡張剰余区間に対する算術演算について調べる。まず、拡張剰余区間に対して、負の拡張剰余区間を次のように定義する。

定義 1

拡張剰余区間 $A = [l,u]^{\alpha_1,\alpha_2,\cdots,\alpha_m}_{\beta_1,\beta_2,\cdots,\beta_m}$ に対して、

$$-A = [-l,-u]^{-\alpha_1,-\alpha_2,\cdots,-\alpha_m}_{\beta_1,\beta_2,\cdots,\beta_m} = [-u,-l]^{\alpha_1,\alpha_2,\cdots,\alpha_m}_{\beta_1,\beta_2,\cdots,\beta_m}.$$

拡張剰余区間に対して、集合の要素毎に演算した結果の集合として加減算を次のように定義する。

定義 2

拡張剰余区間 $A = [l_1,u_1]^{\alpha_1,\alpha_2,\cdots,\alpha_m}_{\beta_1,\beta_2,\cdots,\beta_m}$ と $B = [l_2,u_2]^{\gamma_1,\gamma_2,\cdots,\gamma_n}_{\delta_1,\delta_2,\cdots,\delta_n}$ に対して、

$$\begin{aligned} A+B &\equiv [l_1+l_2, u_1+u_2]^{\alpha_1,\alpha_2,\cdots,\alpha_m,\gamma_1,\gamma_2,\cdots,\gamma_n}_{\beta_1,\beta_2,\cdots,\beta_m,\delta_1,\delta_2,\cdots,\delta_n}. \\ A-B &= A+(-B) = [l_1-u_2, u_1-l_2]^{\alpha_1,\alpha_2,\cdots,\alpha_m,\gamma_1,\gamma_2,\cdots,\gamma_n}_{\beta_1,\beta_2,\cdots,\beta_m,\delta_1,\delta_2,\cdots,\delta_n}. \end{aligned}$$

例えば、$[14,24]^5_2$ と $[6,14]^4_2$ を定義 2 を使って計算すると、以下のようになる。図 1 はその演算結果 $[20,38]^{4,5}_{2,2}$ を表したものである。

$$[14,24]^5_2 + [6,14]^4_2 = [20,38]^{4,5}_{2,2}.$$

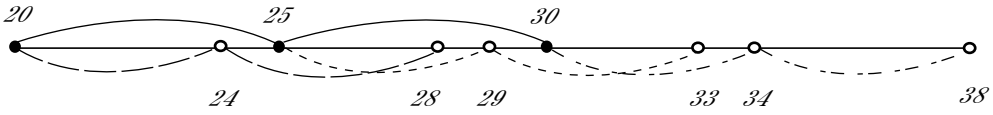

図 1: 拡張剰余区間 $[20,38]^{4,5}_{2,2}$

上に示したように、任意の二つの拡張剰余区間の和および差は拡張剰余区間で表される。しかしながら、二つの拡張剰余区間の積をとったものは、一般に拡張剰余区間では表せない。例えば、

$$A = [0,10]^2_5 = \{0,2,4,6,8,10\}$$

に対して、集合 $A \times A$ は

$$A \times A = \{0, 4, 8, 12, 16, 20, 24, 32, 36, 40, 48, 60, 64, 80, 100\}$$

という集合になるが、これを拡張剰余区間として表す事は不可能である。なぜなら、この集合が拡張剰余区間で表されるとするとステップが 4 となるものがある。このような拡張剰余区間に属する整数 x に対し、$x+4$ または $x-4$ はこれに含まれる。しかし、80 に対しては 84 も 76 もこの集合に含まれていない。したがって、この集合を拡張剰余区間として表現することは出来ない。

ここでは、拡張剰余区間同士の積を、要素毎に積をとった結果の集合を含む拡張剰余区間として、次のように定義する。

定義 3
拡張剰余区間 $A = [l_1, u_1]^{\alpha_1,\alpha_2,\cdots,\alpha_m}_{\beta_1,\beta_2,\cdots,\beta_m}$ と $B = [l_2, u_2]^{\gamma_1,\gamma_2,\cdots,\gamma_n}_{\delta_1,\delta_2,\cdots,\delta_n}$ に対して、

$$A \times B = [l_1 l_2, u_1 u_2]^{l_2\alpha_1,\cdots,l_2\alpha_m,l_1\gamma_1,\cdots,l_1\gamma_n,\alpha_1\gamma_1,\cdots,\alpha_1\gamma_n,\cdots,\alpha_m\gamma_1,\cdots,\alpha_m\gamma_n}_{\beta_1,\cdots,\beta_m,\delta_1,\cdots,\delta_n,\beta_1\delta_1,\cdots,\beta_1\delta_n,\cdots,\beta_m\delta_1,\cdots,\beta_m\delta_n}.$$

2.4 拡張剰余区間の簡約

前節で定義した拡張剰余区間どうしの演算では、演算した結果の次数は、演算前の次数より一般に大きくなる。このことは、計算を続けていくためには好ましくない。

場合によっては低い次数の拡張剰余区間として表すことができることがある。また、例えばデータ依存解析への応用等のように必ずしも正確な演算結果でなくとも、近似的な結果でも十分である場合がある。そのような場合においては、厳密な計算結果に近似した値を保持するように次数を簡約し、計算時間を短縮させた方が解析に有効である場合がある。

この章では、誤差なく次数を簡約する方法と近似による簡約の方法を示す。また多項式を計算する場合に、拡張剰余区間の計算を行う前に多項式を変形することによって、次数が簡約される可能性がある。その式の変形方法を述べる。

2.4.1 誤差なしでの簡約アルゴリズム

前述の「拡張剰余区間の性質」と「拡張剰余区間の包含関係」を使って、拡張剰余区間 $A = [l, u]^{\alpha_1,\alpha_2,\cdots,\alpha_n}_{\beta_1,\beta_2,\cdots,\beta_n}$ に対して誤差を生じることなく簡約するアルゴリズムを示す。

1. $\alpha_i = \alpha_j$ となる α_i, α_j が存在する場合、これをまとめて

   $$A = [l, u]^{\alpha_1,\cdots,\alpha_i,\cdots,\alpha_n}_{\beta_1,\cdots,\beta_i+\beta_j,\cdots,\beta_n}$$

   と簡約する。

2. $\alpha_i | \alpha_j$ かつ $\beta_i \geq \alpha_j/\alpha_i - 1$ を満たす α_i, α_j が存在する場合、

   $$A = [l, u]^{\alpha_1,\cdots,\alpha_i,\cdots,\alpha_n}_{\beta_1,\cdots,\beta'_i,\cdots,\beta_n}$$

   と簡約する。ここで、$\beta'_i = \beta_i + \frac{\alpha_j\beta_j}{\alpha_i}$.

このステップをそれぞれ、条件を満たす組合せ $\{\alpha_i, \alpha_j\}(0 < i < j \leq n)$ が存在しなくなるまで繰り返す。

2.4.2 近似による簡約アルゴリズム

次に誤差は生じるが、拡張剰余区間の次数を n から任意の次数 $m < n$ まで簡約するアルゴリズムを示す。

拡張剰余区間 $A = [l, u]_{\beta_1, \beta_2, \cdots, \beta_n}^{\alpha_1, \alpha_2, \cdots, \alpha_n}$ に対して、以下の操作を結果として得られる拡張剰余区間の次数が m 以下になるまで繰り返す。ここで A は正規形である、つまり $0 < \alpha_1 < \cdots < \alpha_n$、と仮定する。

1. 最大公約数
$$\gamma_{i,j} = \gcd(\alpha_i, \alpha_j), \quad 0 < i < j \leq n$$
を求める。
2. $\dfrac{1}{\gamma_{i,j}}\left(\dfrac{\alpha_i}{\gamma_{i,j}} - 1\right)$ が最小の値を取る i, j および $\gamma_{i,j}$ の値をそれぞれ、i^*, j^* および γ^* と置く。$\dfrac{1}{\gamma_{i,j}}\left(\dfrac{\alpha_i}{\gamma_{i,j}} - 1\right)$ が最小値を取る i, j の組合せが複数ある場合には、$\alpha_j/\gamma_{i,j}$ の値が小さい方を選ぶ。
3. A を
$$[l, u]_{\beta_1, \cdots, \beta'_{i^*}, \cdots, \beta_n}^{\alpha_1, \cdots, \gamma^*, \cdots, \alpha_n}$$
と簡約する。ここで、$\beta'_{i^*} = (\alpha_{i^*}\beta_{i^*} + \alpha_{j^*}\beta_{j^*})/\gamma^*$.

なお、この方法を繰り返し行い、最終的に次数を 1 にまで簡約した結果は、文献 [4] で定義された剰余区間演算の演算結果と一致する。

2.4.3 多項式の計算における簡約

拡張剰余区間の演算では、分配則が一般に成り立たない。そのため多項式の値を計算する場合など、計算の方法によって結果が異なる。

拡張剰余区間演算は「誤差なしでの簡約アルゴリズム」が使えない場合、演算を行うごとに次数が高くなる。特に乗法では、演算項の各下限とステップの積が演算結果のステップとして加わり、より次数が高くなることになる。多項式を計算する前に、各演算項の下限値が 0 になるように式を変形後、計算を行うことによって次数を削減することができる。

例えば、1 変数の二次式 $aX^2 + bX + c$ について、X が拡張剰余区間 $[L, U]_{\beta_1, \beta_2, \cdots, \beta_n}^{\alpha_1, \alpha_2, \cdots, \alpha_n}$ である場合にそのまま計算した結果は次式で与えられる。

$$aX^2 + bX + c \quad = \quad [L', U']_{\beta_1^2, \beta_1\beta_2, \cdots, \beta_n^2, \beta_1, \cdots, \beta_n, \beta_1, \cdots, \beta_n}^{a\alpha_1^2, a\alpha_1\alpha_2, \cdots, a\alpha_n^2, aL\alpha_1, \cdots, aL\alpha_n, b\alpha_1, \cdots, b\alpha_n}$$

ここで、L', U' は演算後の下限値、上限値である。

この計算を $X = L + X'$ とし、式を次のように変形し、

$$\begin{aligned} a(L+X')^2 + b(L+X') + c &= aX'^2 + (2aL+b)X' + aL^2 + bL + c \\ &\equiv aX'^2 + b'X' + c' \end{aligned}$$

として計算すると、

$$X'^2 = \left[0, (U-L)^2\right]^{\alpha_1^2, \alpha_1\alpha_2, \cdots, \alpha_n\alpha_{n-1}, \alpha_n^2}_{\beta_1^2, \beta_1\beta_2, \beta_n\beta_{n-1}, \beta_n^2}$$

より、結果は次式で与えられる。

$$aX'^2 + b'X' + c' = [L', U']^{a\alpha_1^2, a\alpha_1\alpha_2, \cdots, a\alpha_n^2, b'\alpha_1, \cdots, b'\alpha_n}_{\beta_1^2, \beta_1\beta_2, \cdots, \beta_n^2, \beta_1, \cdots, \beta_n}$$

したがって、一般にこのように式を変形後演算する方が次数を削減することができることがわかる。

3 データ依存解析への応用

自動並列化コンパイラは、逐次言語あるいはその拡張で記述された原始プログラムを、それと等価な並列プログラムに自動的に変換するコンパイラである。このためには、データの流れや制御を解析し、並列実行可能な部分を検出する必要がある。この並列性抽出のために行なわれる最も基本的なプログラム解析の 1 つがデータ依存解析である。

```
          do i_1 = l_1, u_1
            ...
              do i_d = l_d, u_d
S1:             A(f_1(i_1, ..., i_d), ..., f_n(i_1, ..., i_d)) = ...
S2:             ... = A(g_1(i_1, ..., i_d), ..., g_n(i_1, ..., i_d))
              continue
            ...
          continue
```

図 2: 多重ループを持つプログラム

上の図 2 に示したプログラムの断片において、S1 での配列 A への書き込みと S2 での配列 A の読みだしの間に依存関係が存在する場合にはループを並列化することは出来ない。この依存関係の存在は次のディオファントス方程式と不等式を満たす整数解を求めることによって判定できる。

$$\begin{cases} f_j(i_1, \cdots, i_d) = g_j(i'_1, \cdots, i'_d)\ , & 1 \le j \le n\ , \\ l_k \le i_k \le u_k\ ,\ l_k \le i'_k \le u_k\ , & 1 \le k \le d \end{cases}$$

データ依存解析の問題は、この方程式が解を持つかどうかを判定すること、および解を持つ場合に実際に解を求めるという二つの問題に分けられる。

データ依存解析においては、必ずしも方程式の解を求める必要はなく、解を持つか否かを判定することで解析は行なえる。もちろん、解を実際に求めることによって更に詳細な解析が行なえるが、これは一般に非常に困難な問題（特に 3 変数以上の多変数方程式あるいは高次の方程式である場合）である。

次節以降では、ディオファントス方程式の解の存在あるいは非存在を判定する問題を拡張剰余区間を用いて解く方法を議論し、データ依存解析への応用について述べる。

また、制御変数の初期値や終値、ストライド等が定数ではなく変数を含む式で与えられており、コンパイル時にその値が不明である場合には、データ解析が困難になる。コンパイル時にデータ依存解析が不可能なコードに対し、実行時に簡単なデータ依存解析を行い、並列実行可能ならば並列ループとして実行し、そうでなければ逐次で実行するという方法が新しい依存問題の解決方法として研究されている [3, 6]。実行時に行なうデータ依存解析はシンプルでなければならないが、多くのデータ依存解析テストは複雑で、実行時のデータ依存解析には向かない。実行時に行うテストとしては、１次元の GCD テストのようなものが考えられるが、このテストでは self-output 依存の場合や、２つ以上の Loop Level のあるループに対しては適用できない。

拡張剰余区間を使ったデータ依存解析は、実行時に行うテストとしてはシンプルであり、self-output 依存や多次元配列、複数の Loop Level を持つループに対しても適用することが可能である。一般的な依存解析テスト（GCD テスト [8] や Banerjee テスト [1]）では解析が不可能であるような添字式への拡張剰余区間演算の応用例を示す。

3.1 拡張剰余区間を使ったデータ依存解析の方法

ディオファントス方程式の解の存在あるいは非存在を判定する問題への応用にあたって、次のように考える。方程式の左辺式（配列で write アクセスされる要素）で表される整数の集合と右辺式（配列で read アクセスされる要素）で表される整数の集合をそれぞれ拡張剰余区間で計算を行ない。２つの集合の共通集合（積集合）の有無を調べる方法である。積集合が空であれば、解を持たないと判断できる。

これから、拡張剰余区間とその演算規則を使って、データ依存の有無を決定するアルゴリズムは次のようになる。

1. 依存関係の有無を決定する必要がある配列の添字式を調べる。多次元配列の場合は、線形化して１次元配列の添字式として表す。
2. 添字式の各制御変数が取り得る値を拡張剰余区間に置換し、拡張剰余区間演算を使って計算する。2.4.1 節で述べた、誤差なしでの簡約アルゴリズムを使って簡約する。
3. 簡約後の２つの拡張剰余区間の積集合の有無を調べる。以下に挙げる拡張剰余区間の積集合が空集合になる条件いずれかが成立する時、２つの配列要素にオーバーラップするアクセス領域が存在しないことになり、データ依存を起こしていないと決定する。

(条件 1) 2 つの拡張剰余区間 $A = [l_1, u_1]^{\alpha_1,\cdots,\alpha_m}_{\beta_1,\cdots,\beta_m}, B = [l_2, u_2]^{\alpha'_1,\cdots,\alpha'_n}_{\beta'_1,\cdots,\beta'_n}$ において、$l_1 > u_2$ 、あるいは $u_1 < l_2$ ならば、$A \cap B = \emptyset$ である。

(条件 2) 2 つの拡張剰余区間 $A = [l_1, u_1]^{\alpha_1,\cdots,\alpha_m}_{\beta_1,\cdots,\beta_m}, B = [l_2, u_2]^{\alpha'_1,\cdots,\alpha'_n}_{\beta'_1,\cdots,\beta'_n}$ において、$\gamma = \gcd(\alpha_1, \cdots, \alpha_m, \alpha'_1, \cdots, \alpha'_n)$ としたとき、$\gamma \nmid (l_2 - l_1)$ ならば、$A \cap B = \emptyset$ である。

(条件 3) 正規化された 2 つの拡張剰余区間 $A = [l_1, u_1]^{\alpha_1,\cdots,\alpha_m}_{\beta_1,\cdots,\beta_m}$, $B = [l_2, u_2]^{\alpha_1,\cdots,\alpha_m}_{\beta_1,\cdots,\beta_m}$ において、次の両条件を満たすならば、$A \cap B = \emptyset$ である。

(1) $\alpha_1 < |l_1 - l_2|$.

(2) $\displaystyle\sum_{j=1}^{i-1} \alpha_j \beta_j + |l_1 - l_2| < \alpha_i \ \ (i = 2, 3, \cdots, m)$.

(条件 4) 正規化された 2 つの拡張剰余区間 $A = [l_1, u_1]^{\alpha_1,\cdots,\alpha_m}_{\beta_1,\cdots,\beta_m}$, $B = [l_2, u_2]^{\gamma_1,\cdots,\gamma_n}_{\delta_1,\cdots,\delta_n}$ において、$l_1 \leq l_2$ の時、次の全ての条件を満たすならば、$A \cap B = \emptyset$ である。

(1) $\alpha_m = \gamma_n$.

(2) $\displaystyle\sum_{i=1}^{m-1} \alpha_i \beta_i < |l_1 - l_2|$.

(3) $\displaystyle\sum_{i=1}^{n-1} \gamma_j \delta_i + |l_1 - l_2| < \alpha_m$.

例えば、次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可能か否かを解析する。

```
Loop:  do i = 4, 1000, 4
S:        A(3i) = ...
T:        ...   = A(2i-1)
       continue
```

図 3: 1 次元配列のデータ依存解析

文 S で定義された配列変数 A の要素の集合 $\mathtt{A}_s$ と、文 T で使用される配列要素 A の要素の集合 $\mathtt{A}_t$ をそれぞれ拡張剰余区間で表すと、$\mathtt{A}_s = [12, 3000]^{12}_{249}, \mathtt{A}_t = [7, 1999]^{8}_{249}$ となる。この例においては、条件 2 を利用することができる。両拡張剰余区間の全ステップの最大公約数は 4 となるが、これは双方の下限値の差 5 を割り切れない。よって $\mathtt{A}_s \cap \mathtt{A}_t = \emptyset$ であり、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わかる。

3.2 上限が変数であるループへの応用

図 4 のプログラム断片は、ベンチマークプログラム The Perfect Benchmarks の OCEAN にある、サブルーチン FTRVMT の一部分を簡単化したループである。ループ L2 において、jj が取り得る値の上限が配列の要素になっている。コンパイル時におけるデータ依存解析では、このようなプログラムの解析は非常に困難である。実行時にデータ依存解析を行う上でも、ループ繰り返し毎に変わる上限値を使ってデータ依存解析を行うのは負荷がかかり過ぎる。拡張剰余区間を使うことによって、上限値が不明であっても依存関係の有無を決定できる。

ここでは、最外側ループ (L1) において、配列 data がループ繰返し依存 (loop carried dependence) を起こしているか調べる。

例えば、文 S で定義される data(js) の内容を異なるループ繰返しで data(js2) によって参照されるならば、配列 data がループ繰返し依存を起こしていると言う。変数 js, js2 の取りうる値を拡張剰余区間を使って求め、その積集合が空集合であれば、ループ繰返し依存がないと決定することができる。

```
L1:     do 109 ij=0, i2k-1
          exj = ....
L2:       do 108 jj = 0,x(jl)
L3:          do 108 mm = 0,128
               js = 258 * i2k * jj + 129 * jl + mm + 1
               js2 = js + 129 * i2k
               h = data(js) - data(js2)
S              data(js) = data(js) + data(js2)
               data(js2) = k * exj
108       continue
109     continue
```

図 4: サブルーチン FTRVMT コードの一部

i2k を 100 とした時、変数 js と js2 の値域を拡張剰余区間で求めると次のような結果を得る。x(jl) は変数 X として表す。

$$\begin{aligned}
\texttt{js} &= 258 \times 100 \times [0, X]^1_{100} + 129 \times [0, 99]^1_{99} + [0, 128]^1_{128} + 1 \\
&= [1, 12900 + 25800X]^{1,129,25800}_{128,99,X} \\
&= [1, 12900 + 25800X]^{1,25800}_{12899,X} \\
\texttt{js2} &= \texttt{js} + 129 \times 100 \\
&= [12901, 25800 + 25800X]^{1,25800}_{12899,X}
\end{aligned}$$

条件 4 から、$[1, 12900 + 25800X]^{1,25800}_{12899,X} \cap [12901, 25800 + 25800X]^{1,25800}_{12899,X} = \emptyset$ となる。この結果より、最外側ループ (L1) において配列 data がループ繰返し依存を起こしていないこ

とを決定できる。

3.3 二次式への応用

配列のデータが三角行列や対称行列で存在している場合にメモリを節約するため、二次元配列を一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。変形後のプログラムにおいて誘導変数消去[1] を行った場合に、配列参照の添字式に二次式が現れる。コンパイル時のデータ依存解析において、添字式が非線形の場合は一般に依存解析が困難である [2, 7]。

実行時のデータ依存解析においても、添字式が二次式である時に GCD テストや Banerjee テストで解析することは不可能である。ここでは、添字式が二次式である時の依存テストに拡張剰余区間を応用した例を示す。

図 5 は、ベンチマークプログラム The Perfect Benchmarks の TRFD にある、サブルーチン OLDA の一部分を簡単化したプログラムである。

```
      nrs = (num*(num+1))/2
      mrsij0 = 0
      do 100  mrs = 1,nrs
        mrsij = mrsij0
        do 90  mi = 1,num
          do 80  mj = 1,mi
            mrsij =  mrsij + 1
            xrsij(mrsij) = xij(mj)
 80       continue
 90      continue
         mrsij0 = mrsij0 + nrs
100     continue
```

図 5: サブルーチン OLDA コードの一部

ここで、制御変数 mj は外側ループの制御変数 mi に依存している ($1 \leq$ mj $\leq$ mi)。このようなループにおいて誘導変数消去を行うと、配列参照の添字式が二次式で表される。誘導変数 mrsij0 と mrsij を消去した後に得られたコードが図 6 である。

ここでは、拡張剰余区間を使って、最外側ループ (L1) における配列 xrsij の self-output 依存の有無を決定する。

図 6 の文 S において、実行時の num が 10 であるとした時、最外側ループ (L1) の i 回目の実行でアクセスされる要素の集合 X_i と、i' 回目の実行でアクセスされる要素の集合 $X_{i'}$ は

[1] 誘導変数とは、ループの中での繰り返しで、一定値増減する変数のことで、配列の添字を参照するために使用されている変数である。制御変数以外の変数が添字式に現れると、コンパイラによる解析が非常に難しくなる。このため、データ解析を行なう前の処理として誘導変数の消去（誘導変数の制御変数での書き換え）や定数伝搬、デッドコードの消去等が行なわれる。

```
L1:       do 100 mrs = 1,(num + num * num) / 2
L2:         do 90 mi = 1,num
L3:           do 80 mj = 1,mi
S:              xrsij(mj + (-mi+mi*mi+mrs*(num+num*num))/2)
                    = xij(mj)
 80           continue
 90         continue
100       continue
```

図 6: 誘導変数消去後のプログラム

次の式で表される。ただし、$1 \leq i < i' \leq 55$。

$$X_i = \mathtt{mj} + (-\mathtt{mi} + \mathtt{mi} \times \mathtt{mi} + i \times 110)/2$$
$$X_{i'} = \mathtt{mj} + (-\mathtt{mi} + \mathtt{mi} \times \mathtt{mi} + (i+1) \times 110)/2$$

計算の簡単化のため、計算する前に添字式を 2 倍する。また、演算結果の誤差を削減するために、式を次のように変形して計算する。制御変数 mj の拡張剰余区間は、$[1,10]^1_9$ として計算する。

$$\begin{aligned} X_i &= 2 \times \mathtt{mj} + \mathtt{mi} \times (\mathtt{mi} - 1) + i \times 110 \\ &= 2 \times [1,10]^1_9 + [1,10]^1_9 \times ([1,10]^1_9 - 1) + 110 \times [1,55]^1_{54} \\ &= [112, 6160]^{1,2,110}_{90,9,54} \\ &= [112, 6160]^{1,110}_{108,54} \end{aligned}$$

$$\begin{aligned} X_{i'} &= 2 \times \mathtt{mj} + \mathtt{mi} \times (\mathtt{mi} - 1) + (i+1) \times 110 \\ &= 2 \times [1,10]^1_9 + [1,10]^1_9 \times ([1,10]^1_9 - 1) + 110 \times ([1,54]^1_{53} + 1) \\ &= [222, 6160]^{1,2,110}_{90,9,53} \\ &= [222, 6160]^{1,110}_{108,53} \end{aligned}$$

条件 4 より、集合 $X_i \cap X_{i'} = \emptyset$ がいえる。この結果より、最外側ループ (L1) の繰り返しにおいて、配列 `xrsij` は self-output 依存を起こしていないことを決定できる。

4 まとめ

本論文では、剰余区間を一般化した拡張剰余区間とその演算規則を考えた。拡張剰余区間は n 組のステップとカウントを情報として保持し、その演算規則による演算結果は、剰余区

間による演算結果より正確になる。また、データ依存解析に応用する上で、添字式や変数を拡張剰余区間で表すことによって、データ依存解析に利用できる重要な情報を保持することができる。拡張剰余区間とその演算規則はシンプルで実装も容易であり、実行時に行うデータ依存解析に応用する上で好ましい。拡張剰余区間を実行時におけるデータ依存解析に応用した例を紹介し、その有効性を示した。

参 考 文 献

[1] U. Banerjee. : *Dependence Analysis for Supercomputing*, Kluwer Academic Publishers, Boston, MA, 1988.

[2] W. Blume and R. Eigenmann. : The Range Test: A Dependence Test for Symbolic, Non-linear Expressions, In *Proceedings of the Conference on Supercomputing*, 1994, pp.528–537.

[3] J. Hoeflinger : Run-Time Dependence Testing by Integer Sequence Analysis, *Technical Report 1194, University of Illinois at Urbana-Champaign, Center for Supercomputing Research and Development*, 1992.

[4] T. Nakanishi , A.Fukuda. : The Modulo Interval : A Simple and Practical Representation for Program Analysis 情報処理学会論文誌, 42 (4) 2001 , pp.829–837 .

[5] Y. Paek, J. Hoeflinger, D. Padua. : Simplification of Array Access Patterns for Compiler Optimizations, In *Proceedings of the 1998 ACM SIGPLAN Conf. on Programming Language Design and Implementation (PLDI)*, pp.60–71, 1998.

[6] C. Polychronopoulos : Advanced loop optimizations for parallel computers, *Lecture Notes in Computer Science No.297: Proc. of First Int'l. Conf. on Supercomputing, Athens, Greece*, pp.255-277, New York, 1987, Springer-Verlag.

[7] W. Pugh : The Omega test: A fast and practical integer programing algorithm for dependence analysis, In *Proceedings of Supercomputing 1991*, 1991, pp.4–13

[8] H.P. Zima, B.M. Chapman.: *Supercompilers for Parallel and Vector Computers*, ACM Press Frontier Series, Addison-Wesley, 1990. 邦訳: 村岡 洋一, スーパーコンパイラ, オーム社, 1995.