6G－4

精密な依存解析のための変数値域の静的予測

酒井 淳嗣　津田 孝夫
京都大学情報工学教室

1 はじめに

言語 Pascal は型チェックの厳しい言語であり、プログラム実行時にも配列参照のたびにその添字式が宣言時の添字範囲内に収まっているか否かを調べている。Pascal のこの性質を利用すると、配列添字式中に現れる整変数のとりうる値の範囲すなわち値域を推定することができる。

本稿では Pascal プログラムにおける変数の値域推定アルゴリズムについて論述し、得られた値域情報によって依存解析の能力向上に寄与できる例をあげる。

2 値域推定の対象

文献 [1] では制御フローに着目した値域推定が論じられているが、ここで提案する値域推定の考え方はそれに加え、Pascal の厳密な型チェックを利用したものである。ソースとして与えられるプログラムが実行時エラーを引き起こさない正しいプログラムであることを仮定すれば、配列要素参照の時点での添字式 (以下単に添字式) の値は配列宣言時の添字範囲内に収まっているはずである。このことから添字式中に用いられている整数型変数の値の範囲が求められる。

この値域推定手法はコンパイラの最適化処理部で利用するためのものであるため、ソースプログラムに対して上のような仮定をおくことは問題にはならないと考えられる。

また添字式を元に値域を推定するため、得られる情報は整数範囲であり、解析対象とする変数も今回は整数型の単純変数に限定している。

3 値域情報の抽出

A は添字範囲 $[A_{lb}..A_{ub}]$ で宣言された配列で、Vは整数型の変数、C_0, C_1は整定数とする。$A[C_0V + C_1]$ という配列要素参照があった場合、$A_{lb} \leq C_0V + C_1 \leq A_{ub}$ が成り立つから、変数 Vに関して

$$\left\lceil \frac{A_{lb} - C_1}{C_0} \right\rceil \leq V \leq \left\lfloor \frac{A_{ub} - C_1}{C_0} \right\rfloor \quad (C_0 > 0 \text{ の場合})$$

$$\left\lceil \frac{A_{ub} - C_1}{C_0} \right\rceil \leq V \leq \left\lfloor \frac{A_{lb} - C_1}{C_0} \right\rfloor \quad (C_0 < 0 \text{ の場合})$$

という値域情報が得られる。

配列要素参照が中途脱出のない **for** ループ中にあり、その添字式がループ制御変数 Iも含んだ線形式 $C_0V + C_1I + C_2$ (C_0, C_1, C_2は整定数) である場合を考える。制御変数 Iは初期値 I_{ini}から終値 $I_{fin}(I_{ini} \leq I_{fin})$ までの全ての値をとるとする。Iの初期値と終値に注目して、$C_0 > 0, C_1 > 0$ の場合には $A_{lb} \leq C_0V+C_1I_{ini}+C_2 \leq C_0V+C_1I_{fin}+C_2 \leq A_{ub}$ が満たされることから

$$\left\lceil \frac{A_{lb} - C_1I_{ini} - C_2}{C_0} \right\rceil \leq V \leq \left\lfloor \frac{A_{ub} - C_1I_{fin} - C_2}{C_0} \right\rfloor$$

という値域情報が得られる。

4 フロー処理

配列要素を参照する文から得られたある変数 Vに関する値域情報は、Vがその参照直前で最後に定義された文からそれ以降最初に Vが再定義される文まで有効である。プログラム中の任意の点で任意の変数に関する値域情報が得られるようにするために、従来から用いられてきた定数伝搬の解析に用いられている手法と類似したフロー処理を行う。

まず値域情報を得た文 (これを配列要素参照点と呼ぶ) と対応するデータフロー集合から配列要素参照点に到達する変数定義 (その点を変数定義点と呼ぶ) を探す。

次に変数定義点と配列要素参照点の実行条件を調べる。ここで実行条件とは、その点が実行される

Range Estimation of Integer Variables for Precise Dependence Analysis
Junji SAKAI, Takao TSUDA
Kyoto University

ためには制御フローパス中の各条件分岐が真側/偽側いずれに選択されなければならないかを示すものであり、**if**文による条件分岐ネストの状態を表現している。

変数定義点と配列要素参照点の間に条件分岐がない場合など、両者の実行条件が同じ場合は、配列要素参照点から得られた値域情報がそのまま変数定義点で発生したとして扱う。配列要素参照点の方が変数定義点よりも深い条件分岐ネストの中にある場合は、両者の実行条件の差分と得られた値域情報を組み合わせたものが変数定義点で発生したとして扱う。例えば、**if** $V < 100$ という条件分岐の真節の中で $1 \leq V \leq 10$ という値域情報が発生した場合、条件節の外側にある変数 Vの定義点には $1 \leq V \leq 10$ or $V \geq 100$ という値域情報が発生したものと見なす。

このようにして変数定義点に集められた値域情報は、プログラム実行の流れに沿って下流に流されていく。条件分岐があれば、そこに到達した値域情報と条件式の論理積が下流側に流される。制御フローが合流するところでは、到達した値域情報の論理和がとられる。

5 ループの扱い

前節で述べたフロー処理を**for**ループに対してそのまま適用すると、解析時に実行時のループ回転数と同じ回数だけループを回って解析しなければならず、非効率的である。そのため初期値/終値が既知の**for**ループについては、ループ本体内では制御変数が初期値から終値までの全ての値をとるとして特別に処理する。

forループで初期値/終値の一方または両方が式である場合や**while**ループの場合は、ループ内で定義される各変数の単調性を調査する。ループ開始直前における変数 Vの値域が $V_{0l} \leq V \leq V_{0u}$で、$V$がループ内で広義単調増加であることが分かれば、ループ内およびループ終了後の Vの値域は $V \geq V_{0l}$とすることができる。

また初期値/終値が式である**for**ループの場合、ループ制御変数を含む添字式がループ内にあれば、これから制御変数の値域を計算し、初期値/終値の式に含まれる変数の値域を推定可能な場合がある。

6 適用例

値域推定の手法の適用により自動ベクトル化コンパイラにおける依存解析の能力向上に寄与できる例を図1にあげる(左端の数字は説明の都合上設けた行番号)。4行目において変数 vの値域は $1 \leq v \leq 49$とわかる。この値域情報は6,7行目でも有効であり、依存解析部がこの情報を用いることにより jループがベクトル化可能と判断される。

```
1: var A: array[1..100,1..100] of real;
2:       v, i, j: integer;
     ...
3: for i:=1 to 50 do begin
4:    A[v+i+1,v] := ...
5:    for j:=1 to 50 do begin
6:       A[100-v,i+j] := ...
7:       ... := A[v,j]...
8:    end
9: end;
```

図1. 値域情報の適用例

またプログラムの自動並列化を試みる際、ループ制御変数の初期値/終値のとりうる値の範囲を計算することでループの粒度をより正しく認識することも可能になる。

7 おわりに

本稿では言語Pascalの型の厳密性を利用してプログラム中の変数のとりうる値の範囲を推定する手法を提案した。我々は研究室で研究開発中の自動ベクトル化コンパイラV-Pascalにこの値域推定手法を組み込んでいる。

この手法を実現する上で、値域情報間の論理和/論理積などの演算が必要になるが、演算が複雑すぎると値域情報を扱いきれなくなる。これは複雑な条件式を持つ条件分岐が多用される場合に起こりやすい。また、手続き/関数間にまたがる解析、ループ構造に対する更に詳細な値域推定解析をどう実現するかが今後の課題である。

参考文献

[1] Harrison, W. H.: *Compiler Analysis of the Value Range for Variables*, IEEE Trans. Softw. Eng., 3-3, 1977