ループ不変条件を利用したコード作成支援手法の検討

2000MT066 中里 誠義
指導教員　真野 芳久

1　はじめに

プログラム作成時における繰返し部分の作成は困難であり、動作を理解するには時間とともに変化する様々な状況を想定する必要があり、その把握は人にとって困難である。

繰返し部分の変化する様々な状況を抽象化し、一つの条件式で表したものにループ不変条件がある。ループ不変条件は繰返し内のある地点で常に成立しており、繰返しを表現するのに適している [1]。また、ループ不変条件を保持するようにコードを作成することによって、正しい繰返し部分の作成ができ、系統的にコード作成ができる。

本研究では、ループ不変条件を利用してのコード作成の利点をふまえ、プログラム作成者の負担を軽減することを目的とし、ループ不変条件と本体コードから護衛を求めること、および、機械 (コンピュータ) での自動化に向けて、知識を持たない機械にどのような知識を与えれば、護衛を求めることができるかを検討する。

2　護衛の導き方

ある条件 B が成り立つときだけ、文 S を実行する命令は if 文や while 文などを使って書けるが、この条件 B は目標である事後条件を満たすように制限している。このような意味から条件 B を護衛と呼ぶ。

ここでは繰返しについて考え、ループ不変条件を利用することとする。ループ不変条件と本体コードが既にわかっていることを前提として、それらから護衛を導く方法を述べる。

2.1　求める方法

ループ不変条件を I、繰返しの本体コードを S とする。また、S が実行される直前に成り立っている条件を P とし、I と P は述語論理式で表され、和積標準形になっているとする。

$$I = i_1 \wedge i_2 \wedge \cdots \wedge i_k$$
$$P = p_1 \wedge p_2 \wedge \cdots \wedge p_l$$

また、P は S と I から wp[*1]を用いて求めることができる。特に、S が代入文の場合、代入文の公理を用いて P を求めることができる。

ここで、P は S と I から求められているので、似た形をとる。また、$I \wedge B \rightarrow P$ であるので、P には I と B を含んでいる可能性がある。そこで、$i_m \rightarrow p_n (1 \leq m \leq k, 1 \leq n \leq l)$ と導ける積項を P から取り除き、残った積項が護衛の候補になるのではないかと考えた。実際にいくつかの例題では、きれいに護衛が求められたので、次節で述べる。

2.2　例題

例題として要素の和を求める問題を例にあげる。整数 $n > 0$ と配列 $b[0:n-1]$ を入力として、b の要素の和を s に入れるプログラムを作る問題。繰返しを用いて解き、0 から $i-1$ までの和が s に入っている不変条件 I を、

$$I = 0 \leq i \leq n \wedge (s = \sum_{0 \leq j < i} b[j])$$

とする。さらに、繰返しの本体コードは `s=s+b[i]; i=i+1;` がわかっているとする。ここで、代入文の公理を用いて、I と本体コードから条件 P を求めると、

$$P = 0 \leq i+1 \leq n \wedge (s + b[i] = \sum_{0 \leq j < i+1} b[j])$$

となる。これから、I の積項から P の積項が導けるかどうか判断する。まず、$0 \leq i \rightarrow 0 \leq i+1$ は導けるので、P から取り除く、次に、$i \leq n \rightarrow i+1 \leq n$ は導くことができないので、$i+1 \leq n$ が護衛の候補として残る。最後に、$(s = \sum_{0 \leq j < i} b[j])$ と $(s + b[i] = \sum_{0 \leq j < i+1} b[j])$ は変形すれば同じ形になるので、導くことができ取り除ける。結局、残った $i+1 \leq n$ が護衛の候補となり、この問題の場合はそのまま護衛となる。このように、導ける積項を取り除くことで護衛の候補を求めることができる。

この例題の他にも、ある値をこえない最大の 2 のべき乗を求める問題、配列中にある値を探索する順探索の問題、フィボナッチ数を求める問題、配列中の要素をある値を基準に分割する問題などでも、求められることが確認できた。

3　護衛を求めるために必要な知識

前節の護衛の導き方を機械で自動的にできないかを考える。機械は人間が指示した通りにしか動かないので、導くための知識を機械に与える必要がある。ここでは、どのような知識を与えてやればよいか述べる。

3.1　必要な知識

3.1.1　増加. 減少の知識

不等式の左辺、右辺の増加や減少によって真であるかどうかが決まる。$A < B \rightarrow A' < B'$ が真であるかどうかは以下の表の通りである。

[*1] wp とは、コード S を実行した直後の状態が I となるための最弱事前条件を表している

A’	B’	真といえるか
A ＜ A’	B	真とはいえない
A’ ≦ A	B	真
A	B ≦ B’	真
A	B’ ＜ B	真とはいえない

不等号 (＜) において、右辺はそのままで左辺が増加している場合 (1 行目) は真とはいえないので、取り除けずに残る。また、右辺はそのまま、左辺が減少もしくはそのままの場合 (2 行目) は真となり、取り除ける。このように不等号に対する値の増加や減少に対して取り除けるかどうかが決まる。

3.1.2 数学の式変形の知識

初等数学の知識全般が必要となる。等式の性質、指数法則、シグマ (Σ) の記号の意味など、数を扱う問題で必要になる。

3.1.3 定義の利用

例えば、フィボナッチ数を求める問題では、フィボナッチ数の定義が必要になるし、オランダ国旗問題や配列の逆転のような問題では、swap という変数の値を入れ換える操作が必要で、swap も定義する必要がある (swap は代入文として扱える)。このような定義を扱える能力が必要になる。

3.1.4 論理記号 (∀、∃) の意味

∀と∃の記号を使った項は表現の仕方を変えることができる。例えば、$n \geq 0, m \geq 0$ を前提として

$$\begin{aligned} & (\forall i : 0 \leq i \leq n : f[i] = x) \\ \equiv & (\forall i : 0 \leq i < n : f[i] = x) \bigwedge f[n] = x \\ & (\exists k : 0 \leq k \leq m : g[k] = y) \\ \equiv & (\exists k : 0 \leq k < m : g[k] = y) \bigvee g[m] = y \end{aligned}$$

のように書き換えることができる。このように書き換え、項を取り除ける形にすればよい。

3.2 例題

オランダ国旗問題を例としてとりあげる。$0 < n$ を入力として、配列 $f[0 : n-1]$ の要素は、*red,white,blue* のいずれかであるとする。配列 f の要素を置換して、オランダの国旗のように左は *red*、真中は *white*、右に *blue* ばかりが並ぶようにするプログラムを作る。

ループ不変条件 I を、

$$\begin{aligned} I\text{: } & 0 \leq r \leq w \leq b+1 \leq n \\ & \wedge(\forall i : 0 \leq i < r : f[i] = red) \\ & \wedge(\forall j : r \leq j < w : f[j] = white) \\ & \wedge(\forall k : b < k < n : f[k] = blue) \end{aligned}$$

とする。*white* と *blue* の間に未処理の部分がある。ここで、本体コードは `swap(f[r],f[w]); w=w+1; r=r+1;` とする。関数 swap は値の互換を行うものとする (定義は省略する)。ループ不変条件 I と本体コードから条件 P を求めると、

$$\begin{aligned} P\text{: } & 0 \leq r+1 \leq w+1 \leq b+1 \leq n \\ & \wedge(\forall i : 0 \leq i < r : f[i] = red) \\ & \wedge(\forall j : r \leq j < w : f[j] = white) \wedge f[w] = red \\ & \wedge(\forall k : b < k < n : f[k] = blue) \end{aligned}$$

となる。P を求める際に、代入文の公理と 3.1.3 の swap の定義、3.1.4 の ∀ の変形を利用して求めている。I と P の ∀ を使っている積項は、全く同じであるので取り除くことができる。また、1 行目は $r+1 \leq w+1$ を数学の式変形により $w \leq b$ とみなすことによって、増加. 減少の知識を適用すると、導けずに残る不等式は $w \leq b$ となる。他の不等式は増加. 減少の知識により、導けるか、全く同じ不等式になっているため、取り除くことができる。よって、$w \leq b$ と $f[w] = red$ が残るので、`swap(f[r],f[w]); w=w+1; r=r+1;` に対する護衛は

$$w \leq b \wedge f[w] = red$$

となる。

3.3 より複雑な例題

配列 $b[0 : n-1]$ の最小の要素を x に入れるという問題で、$x \leq b[0 : i-1] \rightarrow b[i] \leq b[0 : i-1]$ から護衛を導く状況を考える。この場合、比較する対象が x と $b[i]$ となっており導くことができないため、$b[i] \leq b[0 : i-1]$ が護衛の候補となる。しかし、このまま護衛として使うには護衛のためだけに繰返しを作らなくてはならず、護衛として適していない。ここでは十分条件でよいことから発見的に求めることはできる。その方法についてはさらに検討が必要である。

4 おわりに

これまでに述べてきたことだけでは護衛が求められない場合もあり、さらに検討が必要である。また、式変形をどの方向 (目標) に向かって変形するのか、機械による判断は困難である場合も少なくない。そのため、全て機械にやらせるのではなく、人との協調による対話型支援システムであれば実現できるのではないだろうかと考える。

参考文献

[1] D. グリース.; “プログラミングの科学”、 培風館 1991.

[2] 真野芳久; “プログラミング科学” 講義資料、 南山大学経営学部情報管理学科 2001.

[3] 古田壮宏, 真野芳久; ループ不変図式に基づく文芸的プログラミング —繰返しコードの系統的作成方法について—、 南山大学経営研究 2003.