3M-2

クリーネの閉包の繰返し表現法をもつ記号実行法

小池　雅樹† 水穂　良平† 會澤　邦夫‡ 佐藤　匡正*

島根大学大学院総合理工学研究科† 島根大学総合理工学部‡

１．序論

ソフトウェア開発において，仕様をもたないプログラムから静的な分析によって仕様を抽出する場合がある．その抽出の目的はプログラムの振舞いを知ることや，ソースコードと仕様の対応関係を把握することが挙げられる．仕様抽出の一助となるプログラムの内部動作を抽象化し，入出力の特性や機能を得ることにより実現できる．これを実現する手法として，記号実行法がある[1][2]．

記号実行法は，プログラムの入力を記号(変数)と考え，経路を選択し，各ステップの変数の値(以後，変数値)の変化を分析する手法である．経路は，プログラムの開始から終了までに実行される要素処理を表し，繰返し構造の繰返し継続条件や選択構造の選択条件に基づいて一意に定めることができる．これにより，入力とプログラムの内部動作を切り離して変数値の変化を分析できる．プログラムの入力特性は経路の差異，機能は入力に対する内部動作，出力特性はプログラムの出力に着目した経路の差異と入力に対する内部動作として現れる．したがって，プログラムの出力に着目し，記号実行結果を抜粋することで，プログラムの内部動作を抽象化し，入出力特性や機能を仕様として抽出できる．

記号実行法は，プログラムに繰返し構造が含まれる場合，適用することができない．これは，繰返し継続条件や繰返し構造内の選択条件に記号が含まれ場合，経路が確定せず，経路数が増大するためである．これを解決するために，プログラムに含まれる繰返し構造を関数化し，プログラムと繰返し構造を個別に記号実行する方法が提案された[3]．この手法は，繰返し構造をプログラムから切り離し，一つの部分的な経路とみなすことにより，プログラム全体の分析経路数を低減することができる．しかし，関数化した繰返し構造内部を記号実行する場合，繰返し継続条件や選択条件に基づいて経路を定める必要があるため，適用できない．

本稿では，記号実行法によって仕様抽出を行なう新たな試みとして，プログラム構造の考えに基づいて，記号実行法を拡張し，繰返し構造中に選択構造を含む適用例を用いて，その有効性を示す．

２．プログラム構造と記号実行

２．１　プログラム構造の考え

プログラムの静的分析法として，プログラム構造形式化手法がある[4]．この手法は，プログラムの経路を一意に定める条件を機構とし，プログラムが機能を実現するための構成要素を構造とする．機構を除去することにより，構造を非決定性の要素処理列として捉えることができ，プログラムの全経路は正規表現で表すことができる．

２．２　構造に基づく経路選択の考え

プログラム構造に基づき，繰返し構造の経路を考えると次のようになる．繰返し構造に選択構造が複数含まれる場合，一度の繰返しで実行される部分的な経路は複数存在する．これを部分経路の集合A=$\{a_1,..,a_n\}$とする．これらの部分経路は，繰返し構造がもつ機能と考えることができ，仕様との追従性を分析できる[5]．この繰返し構造の経路を正規表現で表すと，次のようになる．

$$(a_1+a_2...+a_n)^*$$

構造より，繰返し構造の経路は，次の経路と等価であるために，変換を行なうことができる．

$$(a_1+a_2...+a_n)^* = ((a_1)^*(a_2)^*...(a_n)^*)^*$$

繰返し構造の経路を，集合 A の各部分経路の連接と繰返しを用いて表現することで，繰返し構造全体を一つの経路と考えることができ，経路数の低減を図ることができる．この経路を鎖状反復経路(CIP:Concatenating Iteration Path)とする．CIP は，繰返し構造がもつ機能を連接して表現することができるため，記号実行により各機能間の関連や振舞いを分析できる．

変換した経路上では，繰返し回数の抽象化に伴い変数値は一意に定まらず，記号実行を行なうのは難しい．そこで，記号実行の際に，繰返しによる変数の振舞いを定めずに，$(a_1)^*$から$(a_n)^*$のそれぞれを一つの要素処理と見なし，この要素処理内で行なわれる変数値の変化を一般化する．この繰返しの振舞いをもつ要素処理を鎖状反復要素(CIE:Concatenating Iteration Element)とする．

２．３　強制制御の扱い

繰返し構造は，前判定繰返し，後判定繰返し，継続繰返しの３つに分類でき，これらの部分経路に打ち切りや戻りなどの強制制御が含まれることがある．繰返し構造の種類や部分経路内の強制制御に対して，強制制御を含まない前判定繰返し構造に変換する方法が提案されている[6]．

この方法に基づくと，繰返し構造の部分経路の集合Aに対して，A=$\{a_1,a_2,\#\}$とした場合，強制制御の変換を以下に示す．ここでは，#を強制制御を含む部分経路としている．また，後判定，継続繰返し構造の前判定繰返し構造への変換も示す．

表１ 強制制御の変換

繰返しの種類	強制制御の変換
前判定	$(a_1+a_2+\#)^* = ((a_1)^*(a_2)^*)^*(\varepsilon+\#)$
	$(a_1+a_2)^* = ((a_1)^*(a_2)^*)^*$
後判定	$(a_1+a_2+\#)^+ = (a_1+a_2)((a_1)^*(a_2)^*)^*(\varepsilon+\#)+\#$
	$(a_1+a_2)^+ = (a_1+a_2)((a_1)^*(a_2)^*)^*$
継続	$(a_1+a_2+\#)^\infty = ((a_1)^*(a_2)^*)^*\#$

２．４　CIE 導出

CIP は要素処理の繰返しと連接により経路を表現できるため，繰返し構造に含まれる変数に設定される値は，

Symbolic Execution with Iteration Representation of Kleene Closure

†Koike Masaki, Mizuho Ryouhei・Interdisciplinary Graduate School of Science and Engineering, Shimane University

‡Aizawa Kunio・Interdisciplinary Faculty of Science and Engineering, Shimane University

* Satou Tadamasa

繰返し前に設定される値と繰返し構造内で実行される要素処理で演算される値，そして CIP の構成により定まる．このために，繰返しにより演算される値を抜き出し，CIP に基づいて一般化を行い，直前の記号実行結果に反映させる．

２．５　制約

本稿において，動的なデータ構造，配列，ポインタを含むプログラムは対象外とする．

３．適用手順

プログラム構造に基づく記号実行法の適用手順を次に示す．

手順 1:入力に対して記号を付与する．

手順 2:プログラム構造に基づき経路を選択する．

プログラム構造を正規表現に形式化する．要素処理に対して，式変数を割り当て，連接(・),選択(+)，繰返し(*,$^+$,$^\infty$)として形式化し，経路を列挙する．

手順 3:繰返し構造内の各変数値を一般化する．

選択した経路から CIE を導出する方法と，一般化する式の記法を示す．

- 繰返し構造内の中間処理を除去する．
- 繰返し構造前に，変数に設定される値を抜き出す．この値を初期値とする．
- 繰返し構造内において，変数に値を設定する処理を抜き出す．この処理から，変数に対する演算を抜き出す．この演算を傾きとする．初期値と傾きを以下に示す記法に基づいて一般化する．

<鎖状反復要素> ::= <初期値><傾き>* | '(' <鎖状反復要素> ')' *
<傾き> ::= <演算子><記号値> | (<傾き>) *
<初期値> ::= '[' <記号値> ']' | <関数>
<記号値> ::= <数> | <記号> | <関数> |
<記号値><演算子><記号値>
<関数> ::= <関数名> '(' <引数>* ')'
<引数> ::= <記号値> | <初期値>
<数> ::= [0-9]$^+$
<記号> ::= [a-z]$^+$

<関数名>はプログラミング言語で定義される関数名を表し，<演算子>は演算子を表す．

手順 4:経路内の命題から経路が選択される条件を抜粋する．

経路条件はプログラム構造に基づく経路が選択される場合に実行される選択構造，繰返し構造の命題を抜き出し，論理式の∧，∨，¬，→により表現する．命題中の変数値を記号実行により得た値に置換する．

４．適用例

記号実行の適用例として素数判定を例に挙げる．図 1 に素数判定の HCP 図を示す．

(1)素数判定への適用

手順 1：

変数 a に対して，入力記号αを与える．

手順 2：

プログラム構造を正規表現により形式化し，強制制御を含まない繰返し構造に変換する．

$$in \cdot s(cr(pn\#+\varepsilon))^* pt = in \cdot s(cr(pn\#+\varepsilon))^*(pn\#+ pt)$$

手順 3：

経路① $in \cdot s \cdot (cr)^* pn\#$
経路② $in \cdot s(cr)^* pt$

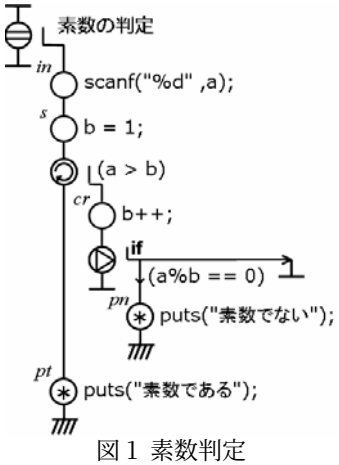

図 1 素数判定

要素処理 cr を一般化すると次のようになる．経路①，②ともに結果は等しい．

$$b=[1](+1)^*$$

手順 4：

経路① $in \cdot s \cdot (cr)^* pn\#$

経路と命題より次の条件が考えられる．

$$a>b \rightarrow a\%b =0$$

記号実行結果より，次の経路条件が得られる．

$$\alpha>[1](+1)^* \rightarrow \alpha\% [1](+1)^*=0$$

経路② $in \cdot s(cr)^* pt$

経路と命題より次の条件が考えられる．

$$(a>b \rightarrow \neg(a\%b =0)) \wedge \neg(a > b)$$

記号実行結果より，次の経路条件が得られる．

$$(\alpha>[1](+1)^* \rightarrow \neg(\alpha\%[1](+1)^*=0)) \wedge \alpha <= [1](+1)^*$$

(2)まとめ

プログラムの出力に着目し，実行結果を整理すると次のようになる．

経路①条件 $\alpha>[1](+1)^* \rightarrow \alpha\% [1](+1)^*=0$
出力 puts(“素数でない”)

経路②条件 $(\alpha>[1](+1)^* \rightarrow \neg(\alpha\%[1](+1)^*=0)) \wedge \alpha <=[1](+1)^*$
出力 puts(“素数である”)

５．結論

本稿では，プログラム構造に基づいて記号実行法を拡張し，繰り返し構造中に選択構造を含む例を用いて，有効性を示した．構造に基づく経路選択方法では，繰返しによる経路増大を低減させる方法を示した．CIE の導出方法では，繰返しの振舞いをもつ要素処理を定義し，一般化する方法を示した．これらより，繰返し構造内への記号実行が可能となり，プログラム全体の動作を抽象化し，仕様抽出を行なう上での一助を得ることができた．

参考文献

[1]J.C,King「Symbolic Execution and Program Testing,」ACM,Vol19,No7,July1976
[2]森本,佐藤「記号実行法による仕様要素の抽出」,SS2001-46(2002-2)電子情報通信学会
[3]森本,佐藤「記号実行における繰返しの関数化」,平成 13 年度電気・情報関連学会中国支部連合大会
[4]佐藤「HCP 図法で記述されたプログラム解法の S 代数による定式化」,情報処理学会誌,Vol27,No6,June1986
[5]佐藤「プログラム構造の仕様逆追従性」,SS2002-3(2002-5)電子情報通信学会
[6]佐藤「打ち切りのある繰返しをもつプログラム構造の形式化」, SS2000-8(2000-5)電子情報通信学会