The Institute of Electronics, Information and Communication Engineers

社団法人 電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report
MSS2011-61 SS2011-46 (2012-1)

関数呼び出しを持つプログラムの非線形ループ不変式の自動生成

鈴木 英一† 坂部 俊樹† 酒井 正彦† 草刈圭一朗† 西田 直樹†

† 名古屋大学大学院情報科学研究科
〒 464–8601 名古屋市千種区不老町
E-mail: †esuzuki@sakabe.i.is.nagoya-u.ac.jp, ††{sakabe,sakai,kusakari,nishida}@is.nagoya-u.ac.jp

あらまし プログラム検証において，ループ実行中に常に成り立つ論理式である不変式が重要な役割を持っている．しかし，検証に有効なループ不変式を自動的に発見することは一般には困難である．本稿では，プログラム変数と関数呼び出し項に関する非線形の不等式で表されるループ不変式を，線形計画法などで利用される Farkas の補題を拡張した定理に基づいて自動生成する手法を示す．

キーワード 静的解析，プログラム検証，不変式生成，最弱事前条件

Automatic Generation of Non-linear Loop Invariants for Programs with Function Calls

Eiichi SUZUKI†, Toshiki SAKABE†, Masahiko SAKAI†,
Keiichirou KUSAKARI†, and Naoki NISHIDA†

† Graduate School of Information Science, Nagoya University
Furoucho, Chikusaku, Nagoyashi, 464–8601 Japan
E-mail: †esuzuki@sakabe.i.is.nagoya-u.ac.jp, ††{sakabe,sakai,kusakari,nishida}@is.nagoya-u.ac.jp

Abstract Finding loop invariants is one of the most important tasks in program verification. It is, however, difficult to automatically find meaningful loop invariants. In this report, we present a method for automatically generating loop invariants in the form of extended polynomial inequality, in which function instances may be included, over program variables. The method is based on the extended polynomial lemma which is improved to Farkas' Lemma.

Key words static analysis, program verification, invariant generation, weakest preconditon

1. はじめに

ループ不変式の自動的な発見は，プログラムの検証やバグの発見の双方に重要な役割を持っている．伝統的に，命令型プログラムの仕様に対する正当性を検証する手法として，モデル検査やホーア論理 [7], [9] に基づく検証手法などが研究されている．モデル検査は，ハードウェアの設計分野で広く利用されているものであるが，抽象化という技法を用いることでソフトウェアの検証にも有効であることが知られている．特に，プログラムの命令文からプログラム変数に関する述語への抽象化 [6] による検証手法があり，変更が加えられる事が前提となるソフトウェア設計の観点から，プログラムの述語の自動変換に基づくモデル検査ツールの開発が盛んである．しかし，これらの手法を用いた検証を完全自動化させるとき障害となるのが事前条件や事後条件といった実行時の条件の設定や，検証に利用される実用的なループ不変式を自動的に発見することが一般的には困難なことである．

ループ不変式の自動的な発見手法として，これまでワイディングと呼ばれる近似ヒューリスティックを用いた手法 [10]〜[12] やテンプレートに基づく手法 [4], [5], [13], [14] など，様々な方法が提案されている．さらにこれらの不変式生成法に基づくツール [8] や SAT/SMT ソルバを利用して，証明に対して十分強力で帰納的な不変式を生成する手法 [2], [3] がある．

本稿では文献 [4] で紹介される制約ベースの解析方法の基礎となる Farkas の補題を拡張し，その拡張定理に基づいて関数呼び出しを持つ命令型プログラムに対するループ不変式を，関数呼び出し項を含む非線形不等式で表されるテンプレートと関数の性質を表す等式の集合を用いて生成する手法を提案する．

本稿は次のように構成される．2. 節では，本手法で対象とするプログラムなどの諸定義を示す．3. 節では，Farkas の補題の

This article is a technical report without peer review, and its polished and/or extended version may be published elsewhere.
Copyright ©2012 by IEICE
NII-Electronic Library Service

The Institute of Electronics, Information and Communication Engineers

拡張定理である拡張多項式の補題の紹介とその証明を示す．4. 節では，拡張多項式の補題に基づいた不変式の自動生成法について示す．5. 節では，4. 節で紹介した手法による例題を示す．6. 節では，まとめと今後の課題を述べる．

2. 準　　備

本節では，本稿で必要な諸定義を与える．

2.1 拡張不等式

X を変数の集合とする．Σ を関数記号の集合とし，各関数記号には引数の個数が指定されているとする．k 引数関数記号 $f \in \Sigma$ と変数 $x_i \in X$ $(i = 1, \ldots, k)$ に対して，$f(x_1, \ldots, x_k)$ を関数呼び出し項という．$xyf(x, y)z$ などのような変数と関数呼び出し項の積を X に関する Σ 上の拡張積項という．実数定数 c_i $(i = 0, \ldots, n)$ と拡張積項 t_i $(i = 1, \ldots, n)$ に対して，$c_0 + c_1 t_i + \cdots + c_n t_n$ を X に関する Σ 上の拡張多項式という．また，$c_0 + c_1 t_i + \cdots + c_n t_n \geqq 0$ を X に関する Σ 上の拡張不等式という．

本稿で扱う論理式は，拡張不等式を原子論理式とする一階論理式とする．

2.2 ループプログラム

本稿で扱うプログラムは，次の形の重なりのない単一のループからなるループプログラムとする．

while true do c **od**

ループの本体 c は命令文であり，

- スキップ文： **skip**,
- 代入文： $x := t$,
- 逐次実行文： $c_1; c_2$，または，
- if 文： **if** q **then** c_1 **else** c_2 **fi**

である．ここに，x は変数，t は拡張多項式，q は拡張不等式を原子論理式とする論理式 (限量子を含まないとする)，c_1，c_2 は命令文である．なお，while 文の条件を true にしているのは議論を簡単にするためである．

本稿では，ループプログラムに現れる関数記号は実数上の関数として解釈され，その関数は別に存在する関数定義プログラムによって定まっていると仮定する．その上で，ループプログラムの意味は，通常のように変数への値割り当てを状態とする状態遷移系として定められるとする．

2.3 帰納的不変式

ループプログラムの帰納的不変式は，初期条件を満たす状態 (変数への値割当て) から実行を開始したとき，ループ本体の先頭で常に成り立つ論理式であり，次のように定められる．

ループ本体が c であるループプログラムと初期条件と呼ぶ論理式 θ に対して，論理式 φ は，初期制約，遷移制約と呼ぶ次の論理式が，その論理式に現れるすべての変数の集合 $\{x_1 \ldots x_k\}$ に関して実数上で成り立つとき帰納的不変式であるという．

- 初期制約：$\forall x_1 \ldots \forall x_k(\theta \Rightarrow \varphi)$
- 遷移制約：$\forall x_1 \ldots \forall x_k(\varphi \Rightarrow wp(c, \varphi))$

また，$\mathrm{wp}(c, \varphi)$ は命令文 c と論理式 φ に対して以下のように定められる論理式である．

$$\mathrm{wp}(\mathbf{skip}, q) = q$$
$$\mathrm{wp}(x := t, q) = q[t/x]$$
$$\mathrm{wp}(c_0; c_1, q) = \mathrm{wp}(c_0, \mathrm{wp}(c_1, q))$$
$$\mathrm{wp}(\mathbf{if}\ b\ \mathbf{then}\ c_0\ \mathbf{else}\ c_1\ \mathbf{fi}, q) = (b \Rightarrow \mathrm{wp}(c_0, q)) \wedge (\neg b \Rightarrow \mathrm{wp}(c_1, q))$$

ここで，$q[t/x]$ は，論理式 q に現れる変数 x を拡張多項式 t で同時に置き換えて得られる論式である．

2.4 Farkas の補題に基づく不変式の生成

文献 [4], [13] において，次の Farkas の補題を利用して，線形不等式の形の不変式を求める手法が提案されている．

定理 1 (Farkas の補題) $\mathcal{S}$ および φ を，それぞれ，次のような変数の集合 $\{x_1, \ldots, x_m\}$ に関する線形不等式の連言，および，線形不等式とする．

$$\mathcal{S}: \begin{bmatrix} a_{1,0} + a_{1,1}x_1 + \cdots + a_{1,m}x_m & \geqq 0 & \wedge \\ a_{2,0} + a_{2,1}x_1 + \cdots + a_{2,m}x_m & \geqq 0 & \wedge \\ \cdots & & \\ a_{n,0} + a_{n,1}x_1 + \cdots + a_{n,m}x_m & \geqq 0 & \end{bmatrix}$$

$$\varphi:\ p_0 + p_1 x_1 + \cdots + p_m x_m \geqq 0$$

また，$\mathcal{T}$ を，次のような変数集合 $\{\lambda_1, \ldots, \lambda_n\}$ に関する線形不等式の連言とする．

$$\mathcal{T}: \begin{bmatrix} p_0 \geqq & a_{1,0}\lambda_1 + a_{2,0}\lambda_2 + \cdots + a_{n,0}\lambda_n & \wedge \\ p_1 = & a_{1,1}\lambda_1 + a_{2,1}\lambda_2 + \cdots + a_{n,1}\lambda_n & \wedge \\ \cdots & & \\ p_m = & a_{1,m}\lambda_1 + a_{2,m}\lambda_1 + \cdots + a_{n,m}\lambda_n & \end{bmatrix}$$

実数上で $\mathcal{S}$ が成り立つような各変数 x_i への値の割り当てが存在するとする．このとき，

$$\forall x_1 \cdots \forall x_m(\mathcal{S} \Rightarrow \varphi) \tag{1}$$

が実数上で成り立つことと，

$$\exists \lambda_1 \geqq 0 \cdots \exists \lambda_n \geqq 0(\mathcal{T}) \tag{2}$$

が実数上で成り立つことは互いに必要十分条件である．

文献 [4], [13] では概略として次のような方法で線形不等式の形の不変式を求める．

- φ は p_i を未知係数とする不変式のテンプレートとする．
- ループ本体を 1 回実行した後の変数の値を表す新しい変数を導入して，実行前から実行後への状態の遷移を線形等式で表す．
- φ が帰納的不変式であるための条件を Farkas の補題の (1) の形の論理式で表現する．
- Farkas の補題の (2) における λ_i を適宜定めて，未知係数を求める．

さらに，文献 [2], [3] では，Farkas の補題の系である多項式の補題を用いて非線形不等式の形の不変式の自動生成法が提案されている．

NII-Electronic Library Service

The Institute of Electronics, Information and Communication Engineers

3. 拡張多項式の補題

本節では，多項式の補題を拡張積項に対応させた拡張多項式の補題を与える．

定理 2 (拡張多項式の補題) Σ を関数記号の集合とし，それぞれの関数記号の解釈として実数上の関数が定められているとする．変数集合 $\{x_1,\ldots,x_k\}$ に関する Σ 上の拡張積項 t_i $(i=0,\ldots,m)$ に対して，$\mathcal{S}$ および φ は，それぞれ，以下のようなの拡張不等式の連言および拡張不等式とする．

$$\mathcal{S}:\left[\begin{array}{lll} a_{1,0}t_0+a_{1,1}t_1+\cdots+a_{1,m}t_m & \geqq 0 & \wedge \\ \cdots & & \\ a_{n,0}t_0+a_{n,1}t_1+\cdots+a_{n,m}t_m & \geqq 0 & \end{array}\right]$$

$$\varphi:\ p_0t_0+p_1t_1+\cdots+p_mt_m\geqq 0$$

また，$\mathcal{S}$ と φ に対して $\mathcal{T}$ を次のような変数集合 $\{\lambda_1,\ldots,\lambda_n\}$ に関する線形不等式の連言とする．

$$\mathcal{T}:\left[\begin{array}{ll} \bigwedge_{\forall x_1\cdots\forall x_k(t_i\geqq 0)} & p_i\geqq\lambda_1a_{1,i}+\cdots+\lambda_na_{n,i} \\ \wedge \bigwedge_{\neg\forall x_1\cdots\forall x_k(t_i\geqq 0)} & p_i=\lambda_1a_{1,i}+\cdots+\lambda_na_{n,i} \end{array}\right]$$

このとき，実数上で

$$\exists\lambda_1\geqq 0\cdots\exists\lambda_n\geqq 0\ (\mathcal{T}) \tag{3}$$

が成り立つならば，

$$\forall x_1\cdots\forall x_k(\mathcal{S}\Rightarrow\varphi) \tag{4}$$

が実数上で成り立つ．

証明　$\mathcal{T}$ が成り立つ $\lambda_i\geqq 0$ $(i=1,\ldots,n)$ が存在するとする．また，x_j $(j=1,\ldots,m)$ を任意の値とし，$\mathcal{S}$ が成り立つとする．各 $i=1,\ldots,n$ に対して，
($t_i\geqq 0$ の場合)　$\mathcal{T}$ より $p_i\geqq\lambda_1a_{1,i}+\cdots+\lambda_na_{n,i}$ である．両辺に t_i を乗じると $p_it_i\geqq(\lambda_1a_{1,i}+\cdots+\lambda_na_{n,i})t_i$ を得る．
(それ以外の場合)　$\mathcal{T}$ より両辺に t_i を乗じると $p_it_i=(\lambda_1a_{1,i}+\cdots+\lambda_na_{n,i})t_i$ を得る．
各 i について得られた拡張不等式を辺々加えると

$$p_0t_0+\cdots+p_mt_m\geqq(\sum_{j=1}^{n}\lambda_ja_{j,0})t_0+\cdots+(\sum_{j=1}^{n}\lambda_ja_{j,m})t_m$$

となる．右辺は次のように整理することができる．

$$\begin{array}{l} \lambda_1(a_{1,0}t_0+a_{1,1}t_1+\cdots+a_{1,m}t_m)\quad+ \\ \cdots \\ \lambda_n(a_{n,0}t_0+a_{1,1}t_1+\cdots+a_{n,m}t_m) \end{array}$$

$\mathcal{S}$ と $\lambda_i\geqq 0$ より，$(\sum_{j=1}^{n}\lambda_ja_{j,0})t_0+\cdots+(\sum_{j=1}^{n}\lambda_ja_{j,m})t_m\geqq 0$ である．よって $p_0t_0+\cdots+p_mt_m\geqq 0$ が成り立つ．　□

4. 拡張不等式の形の不変式の生成

本節では，拡張不等式の形の不変式を拡張多項式の補題に基づいて生成する手法を提案する．その手順は概略次のとおりである．

1. 入力として，ループプログラム，初期条件，テンプレートと呼ぶ拡張不等式，および，関数の性質と呼ぶ等式集合が与えられるとする．テンプレートの係数は未知係数として扱う．
2. テンプレートに対して初期制約と遷移制約の論理式を構成する．
3. 初期制約と等価な拡張多項式の補題の式 (4) の形の論理式を構成し，それに対して，拡張多項式の補題の式 (3) の形の論理式を構成し，λ_j の取りうる値の範囲を考慮して，未知係数に関する制約を求める．
4. 遷移制約についても同様にして，未知係数に関する制約を求める．
5. 両者の制約を満たす未知係数を求める．

以下に，生成手法の詳細を説明する．まず，関数の性質を用いない場合の例を示す．

例 3　次のようなプログラム，初期条件，テンプレートを考える．

- 初期条件 θ　　$x\geqq 0\wedge y=0$
- テンプレート φ　$p_0+p_1x+p_2y+p_3f(x)\geqq 0$
  ただし $f(x)\geqq 0$
- プログラム c　　`while true`
  `do` $y:=f(x)+x$ `od`

初期制約 $\theta\Rightarrow\varphi$ は次のような論理式である．

$$\forall(x\geqq 0\wedge y\geqq 0\wedge -y\geqq 0\Rightarrow p_0+p_1x+p_2y+p_3f(x)\geqq 0)$$

これは (4) の形の論理式であり，$\mathcal{S}$ と φ を以下のような表で表す．各列は $\mathcal{S}$ と φ に現れる拡張積項に対応する．最下行は φ を表す．下から 2 行目は $\forall x_1\cdots\forall x_k(t_i\geqq 0)$ が成り立つときは $\geqq$，そうでないときは $=$ であり，$\mathcal{T}$ の構成に必要な情報である．それ以外の行は，$\mathcal{S}$ の拡張不等式に対応する．最左列は，各行に対応する $\mathcal{T}$ の変数 λ_i である．

λ_1	$0\cdot 1$	$1\cdot x$	$0\cdot y$	$0\cdot f(x)$	$\geqq$	0
λ_2	$0\cdot 1$	$0\cdot x$	$1\cdot y$	$0\cdot f(x)$	$\geqq$	0
λ_3	$0\cdot 1$	$0\cdot x$	$-1\cdot y$	$0\cdot f(x)$	$\geqq$	0
t_i	$\geqq$	$=$	$=$	$\geqq$		
φ	p_0	p_1x	p_2y	$p_3f(x)$	$\geqq$	0

これから (3) の論理式を構成すると以下のようになる．

$$\begin{array}{l} \exists\lambda_1\geqq 0\exists\lambda_2\geqq 0\exists\lambda_3\geqq 0 \\ \quad(p_0\geqq 0\wedge p_1=\lambda_1\wedge p_2=\lambda_2-\lambda_3\wedge p_3\geqq 0) \end{array}$$

$\lambda_j\geqq 0$ の取りうる値を考慮して p_i に関する制約を適宜求めた結果が次の制約である．

$$p_0\geqq 0\wedge p_1\geqq 0\wedge p_3\geqq 0$$

NII-Electronic Library Service

The Institute of Electronics, Information and Communication Engineers

次に，遷移制約 $\varphi \Rightarrow \mathrm{wp}(c,\varphi)$ を求めて表にすると次のようになる．

λ	p_0	p_1x	p_2y	$p_3f(x)$	$\geqq$	0
t_i	$\geqq$	$=$	$=$	$\geqq$		
$\mathrm{wp}(c,\varphi)$	p_0	$(p_1+p_2)x$	$0\cdot y$	$(p_2+p_3)f(x)$	$\geqq$	0

これより (3) の論理式を構成する．

$$\exists\lambda \geqq 0$$
$$(p_0 \geqq \lambda p_0 \wedge p_1+p_2 = \lambda p_1 \wedge 0 = \lambda p_2 \wedge p_2+p_3 \geqq \lambda p_3)$$

$\lambda_j \geqq 0$ の取りうる値を考慮して p_i に関する制約を適宜求めると $p_2 = 0$ が得られる．

以上の p_i に関する制約の解の一つは $p_0 = 1, p_1 = 1, p_2 = 0, p_3 = 1$ である．これをテンプレートの未知係数に代入すると $1+x+f(x) \geqq 0$ という拡張不等式が不変式として得られる．

一般には，初期制約と遷移制約はそのままでは (3) の形にはならない．それらを

$$\begin{aligned}&(\mathcal{S}_{1,1} \Rightarrow \varphi_{1,1}) \vee \cdots \vee (\mathcal{S}_{1,n_1} \Rightarrow \varphi_{1,n_1})\\ &\cdots\\ \wedge\ &(\mathcal{S}_{m,1} \Rightarrow \varphi_{m,1}) \vee \cdots \vee (\mathcal{S}_{m,m_1} \Rightarrow \varphi_{m,m_1})\end{aligned}$$

の形に整理した後，各 i について $(\mathcal{S}_{i,1} \Rightarrow \varphi_{i,1}) \vee \cdots \vee (\mathcal{S}_{i,i_1} \Rightarrow \varphi_{i,i_1})$ から未知係数を求める．

次に，与えられた関数の性質を利用する場合を説明する．

関数の性質は Σ 上の等式の集合とする．以下の議論では，与えられたループプログラムに付随する関数記号の解釈である実数上の関数はこの等式集合のすべての等式を満たすものとする．また，関数の性質は，実際には，停止性と合流性のある項書換え系で与えられるとする．項書換え系に関しては，例えば，文献 [1] を参照されたい．

関数の性質は，初期制約と遷移制約の論理式中に関数呼び出し項以外の項 $f(t_1,\ldots,t_k)$ が含まれる場合，その項を簡約するのに利用する．そのことによって，未知係数に関するより多くの制約を得ることができる．

例 4 関数の性質による最弱事前条件の書換え例を示す．

- テンプレート φ　$p_0+p_1x+p_2y+p_3g(x) \geqq 0$
- 関数の性質 R　$g(v+1) = g(v)+2$
- プログラム c　**while true do if** $(x \geqq 2)$
  then $y := y+g(x); x := x+1$
  else $y := y+1; x := x+2$

このプログラムの $\mathrm{wp}(c,\varphi)$ を計算すると

$$(x \geqq 2 \Rightarrow p_0+p_1(x+1)+p_2(y+g(x))+p_3g(x+1) \geqq 0) \wedge$$
$$(\neg(x \geqq 2) \Rightarrow p_0+p_1(x+2)+p_2(y+a)+p_3g(x+2) \geqq 0)$$

となる．R から $g(x+1) \to g(x)+2$，$g(x+2) \to g(x)+4$ となり，これによって書換えた論理式 q を連言標準形に変換する．

$$(\neg(x \geqq 2) \vee p_0+p_1(x+1)+p_2(y+g(x))+p_3(g(x)+2) \geqq 0)$$
$$\wedge(x \geqq 2 \vee p_0+p_1(x+2)+p_2(y+1)+p_3(g(x)+4) \geqq 0)$$

このように R によって書換えられた q は，任意の変数に対して $\mathrm{wp}(c,\varphi)$ と等価であり，ここから遷移制約を生成すると，$\varphi \wedge (x \geqq 2) \Rightarrow p_0+p_1(x+1)+p_2(y+g(x))+p_3(g(x)+2) \geqq 0$ かつ，$(\varphi \Rightarrow x \geqq 2) \vee (\varphi \Rightarrow p_0+p_1(x+1)+p_2(y+g(x))+p_3(g(x)+2) \geqq 0)$ となる．

このようにして得られた論理式を行列表現へ変換して定理 2 を使って解くことにより，未知係数 p に関する論理式が得られる．同様の手順により初期制約についても未知係数 p に関する論理式を計算する．未知係数に関する論理式を解くヒューリスティックについては文献 [13] を参考にする．そして決定した未知係数を φ に代入した論理式を出力する．この φ が本手法により得られる不変式である．次にこの手法によって得られた論理式がプログラムに関する帰納的不変式となることを示す．

定理 5 (不変式の生成) θ は初期条件を表す論理式，c はプログラム，R は関数の性質とする．φ が $p_0+p_1t_1+\cdots+p_mt_m \geqq 0$ となるテンプレートとするならば，本手法実行後の φ はプログラムに関する帰納的不変式である．

証明　拡張多項式の補題を用いることで証明可能．　□

5. 例　　題

4. 節で示した手法をいくつかの例で紹介する．ただし，プログラムで扱う変数や関数の引数，返し値は全て整数型とする．

例 6 ループプログラムに関する簡単な例題を示す．

- 初期条件 θ　$\{\ x = 0 \wedge y = 0\ \}$
- テンプレート φ　$p_0+p_1x+p_2y+p_3f(x) \geqq 0$
- 関数の性質 R　$f(0) = 0$
  $f(v+1) = 1-f(v)$
- プログラム c　**while true**
  do $y := y+f(x); x := x+1$ **od**

まず初期制約，遷移制約それぞれから (3) の形の論理式を構成して計算する．

- 初期制約の場合　初期条件 θ から最弱事前条件を計算する．計算は $x = 0 \wedge y = 0$ を直接代入することで簡略化している．

$$p_0+p_10+p_20+p_3f(0) \geqq 0$$

R から $f(0) \to 0$ という書換え規則が得られ，これにより書換えを行うと次の未知係数 p に関する論理式が得られる．

$$p_0 \geqq 0$$

- 遷移制約の場合　プログラム c から最弱事前条件を計算する．

$$p_0+p_1(x+1)+p_2x(y+f(x))+p_3f(x+1) \geqq 0$$

R から $f(x+1) \to 1-f(x)$ という書換え規則が得られ，これにより最弱事前条件を書換えすると次の未知係数 p に関する論理式が得られる．

$$p_0+p_1+p_3+p_1x+p_2y+(p_2-p_3)f(x) \geqq 0$$

NII-Electronic Library Service

Λ を $\{\lambda\}(\geqq 0)$ となるベクトルとして，計算した p に関する論理式を用いて以下のような行列を作る．

$$\begin{bmatrix} p_0+p_1+p_3 \\ p_1 \\ p_2 \\ p_2-p_3 \end{bmatrix}^t \begin{bmatrix} \geqq \\ = \\ = \\ = \end{bmatrix}^t \lambda \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix}^t$$

次に λ に関する双対制約を作り，$\lambda \geqq 0$ を探して消去すると $p_1+p_3 \geqq 0 \wedge p_2 = 2p_3$ という論理式が得られる．

以上の結果から $p_0 \geqq 0 \wedge p_1+p_3 \geqq 0 \wedge p_2 = 2p_3$ を満たす解として $p_0=0, p_1=-1, p_2=2, p_3=1$ がある．これを φ に代入すると $-x+2y+f(x) \geqq 0$ という不変式が発見できる．さらに $p_0=0, p_1=1, p_2=-2, p_3=-1$ も解となることから $-(-x+2y+f(x)) \geqq 0$ も不変式として発見できる．プログラムをトレースした結果は以下のとおり．

表 1　例 6 のトレース結果

x	0	1	2	3	4	5	6	7	8	$\cdots$
y	0	0	1	1	2	2	3	3	4	$\cdots$
$f(x)$	0	1	0	1	0	1	0	1	0	$\cdots$
x-2y-$f(x)$	0	0	0	0	0	0	0	0	0	$\cdots$

表 1 から確かに $x-2y-f(x)=0$ が例 6 のプログラムの不変式となっていることが確認できる．

例 7　プログラムに複数の関数が現れる例を次に示す．

- 初期条件 θ　　$\{\ x=0 \wedge y=0\ \}$
- テンプレート φ　　$p_0+p_1x+p_2y+$ $p_3f(x)+p_4g(x)+p_5h(x) \geqq 0$
- 関数の性質 R　　$f(0)=0$
  $f(v+1)=1-f(v)$
  (ただし $f(v) \geqq 0$ とする)
  $g(0)=1$
  $g(v+1)=h(v)+1$
  (ただし $g(v) \geqq 0$ とする)
  $h(0)=0$
  $h(v+1)=g(v)+1$
  (ただし $h(v) \geqq 0$ とする)
- プログラム c　　**while true do**
  if $0=f(x)$ **then** $y:=y+g(x)$
  else $y:=y+h(x)$ **fi**
  $x:=x+1$ **od**

- 初期制約の場合　初期条件 θ から最弱事前条件を計算する．

$$p_0+p_1 0+p_2 0+p_3 f(0)+p_4 g(0)+p_5 h(0) \geqq 0$$

R から $f(0) \to 0, g(0) \to 1, h(0) \to 0$ という書換え規則が得られ，これにより書換えを行うと次の未知係数 p に関する論理式が得られる．

$$p_0+p_4 \geqq 0$$

- 遷移制約の場合　$b \equiv 0=f(x)$ とすると，プログラム c の最弱事前条件は，$(b \Rightarrow p_0+p_1(x+1)+p_2(y+g(x))+p_3f(x+1)+p_4g(x+1)+p_5h(x+1) \geqq 0) \wedge (\neg b \Rightarrow p_0+p_1(x+1)+p_2(y+h(x))+p_3f(x+1)+p_4g(x+1)+p_5h(x+1) \geqq 0)$ となる．$\neg b$ は変数が整数域かつ $f(x) \geqq 0$ であることから $\neg b \equiv f(x) \geqq 1$ となり，R から $f(x+1) \to 1-f(x), g(x+1) \to h(x)+1, h(x+1) \to g(x)+1$ という書換え規則が得られ，これにより最弱事前条件を書換えすると次の未知係数 p に関する論理式が得られる．

$$(f(x)=0 \Rightarrow p_0+p_1+p_3+p_4+p_5+p_1x+p_2y-f(x)p_3+(p_2+p_5)g(x)+p_4h(x) \geqq 0) \wedge (f(x) \geqq 1 \Rightarrow p_0+p_1+p_3+p_4+p_5+p_1x+p_2y-f(x)p_3+p_5g(x)+(p_2+p_4)h(x) \geqq 0)$$

それぞれの節を q_1, q_2 とすると，遷移制約は $(\varphi \Rightarrow q_1) \wedge (\varphi \Rightarrow q_2)$ となり，これらを行列化する．

- $(\varphi \Rightarrow q_1)$ の場合　$f(x)=0$ より，$f(x)$ に関する行が省略され，4,5 行目は R に $g(v), h(v) \geqq 0$ という条件があることから $\geqq$ が使われる．

$$\begin{bmatrix} p_0+p_1+p_3+p_4+p_5 \\ p_1 \\ p_2 \\ p_2+p_5 \\ p_4 \end{bmatrix}^t \begin{bmatrix} \geqq \\ = \\ = \\ \geqq \\ \geqq \end{bmatrix}^t \lambda \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_4 \\ p_5 \end{bmatrix}^t$$

λ に関する双対制約を作り，$\lambda \geqq 0$ を探して消去すると $p_1+p_3+p_4+p_4+p_5 \geqq 0 \wedge p_2 \geqq 0 \wedge p_4=p_5$ という論理式が得られる．

- $(\varphi \Rightarrow q_2)$ の場合　4,5,6 行目は R に $f(v), g(v), h(v) \geqq 0$ という条件があることから $\geqq$ が使われる．

$$\begin{bmatrix} p_0+p_1+p_3+p_4+p_5 \\ p_1 \\ p_2 \\ -p_3 \\ p_5 \\ p_2+p_4 \end{bmatrix}^t \begin{bmatrix} \geqq \\ = \\ = \\ \geqq \\ \geqq \\ \geqq \end{bmatrix}^t \begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix}^t \begin{bmatrix} p_0 & -1 \\ p_1 & 0 \\ p_2 & 0 \\ p_3 & 1 \\ p_4 & 0 \\ p_5 & 0 \end{bmatrix}^t$$

λ_1, λ_2 に関する双対制約を作り，$\lambda_1, \lambda_2 \geqq 0$ を探して消去すると $0 \geqq p_3$ という論理式が得られる．

以上の結果から $p_0+p_4 \geqq 0 \wedge p_1+p_3+p_4+p_4+p_5 \geqq 0 \wedge p_2 \geqq 0 \wedge p_4=p_5 \wedge 0 \geqq p_3$ を満たす解として $p_0=-1, p_1=-2, p_2=0, p_3=0, p_4=1, p_5=1$ がある．これを φ に代入すると $-1-2x+g(x)+h(x) \geqq 0$ という不変式が発見できる．プログラムをトレースした結果は以下のとおり．

表 2　例 7 のトレース結果

x	0	1	2	3	4	5	6	7	8	$\cdots$
$g(x)$	1	1	3	3	5	5	7	7	9	$\cdots$
$h(x)$	0	2	2	4	4	6	6	8	8	$\cdots$
-1-2x+$g(x)$+$h(x)$	0	0	0	0	0	0	0	0	0	$\cdots$

NII-Electronic Library Service

The Institute of Electronics, Information and Communication Engineers

例 8　$m(u,v)$ という二引数関数についての不変式を生成する．

- 初期条件 θ　　$\{\, x=3 \wedge y=0 \wedge z=0 \,\}$
- テンプレート φ　$p_0+p_1x+p_2xy+p_3m(x,y) \geqq 0$
- 関数の性質 R　　$m(u,0)=0$
  $m(u,v+1)=u+m(u,v)$
  ただし $m(u,v) \geqq 0$ とする．
- プログラム c　　**while true**
  do $z := z+m(x,y); y := y+1$ **od**

- 初期制約の場合　初期条件 θ から最弱事前条件を計算する．

$$p_0+p_1 3+p_2 3*0+p_3 m(3,0) \geqq 0$$

R から $m(u,0) \to 0$ という書換え規則が得られ，これにより書換えを行うと次の未知係数 p に関する論理式が得られる．

$$p_0+3p_1 \geqq 0$$

- 遷移制約の場合　プログラム c から最弱事前条件を計算する．

$$p_0+p_1x+p_2x(y+1)+p_3m(x,y+1) \geqq 0$$

R から $m(x,y+1) \to x+m(x,y)$ という書換え規則が得られ，これにより最弱事前条件を書換えすると次の未知係数 p に関する論理式が得られる．

$$p_0+(p_1+p_2+p_3)x+p_2xy+p_3m(x,y) \geqq 0$$

Λ を $\{\lambda\}(\geqq 0)$ となるベクトルとして，計算した p に関する論理式を用いて以下のような行列を作る．4 行目は R に $m(u,v) \geqq 0$ という条件があることから $\geqq$ が使われる．

$$\begin{bmatrix} p_0 \\ p_1+p_2+p_3 \\ p_2 \\ p_3 \end{bmatrix}^t \begin{bmatrix} \geqq \\ = \\ = \\ \geqq \end{bmatrix}^t \lambda \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix}^t$$

次に λ に関する双対制約を作り，$\lambda \geqq 0$ となる λ を探して消去すると $p_2+p_3=0$ という論理式が得られる．

以上の結果から $p_0+3p_1 \geqq 0 \wedge p_2+p_3=0$ を満たす解として $p_0=0, p_1=0, p_2=1, p_3=-1$ がある．これを φ に代入すると $xy-m(x,y) \geqq 0$ という不変式が発見できる．プログラムをトレースした結果は以下のとおり．

表 3　例 8 のトレース結果

x	3	3	3	3	3	3	3	3	3	$\cdots$
y	0	1	2	3	4	5	6	7	8	$\cdots$
$m(x,y)$	0	3	6	9	12	15	18	21	24	$\cdots$
xy-$m(x,y)$	0	0	0	0	0	0	0	0	0	$\cdots$

6. 終わりに

本稿では，命令型プログラムの検証において重要なループ不変式を，Farkas の補題の拡張定理となる拡張多項式の補題を用いて，関数呼び出し項を含む非線形不等式で表される未知係数を持つテンプレートと関数の性質を表す等式の集合を入力することで自動生成する手法を示した．

今後の課題として，本手法によって得られたループ不変式を用いたプログラム検証の結果が，現在のプログラムや関数呼び出しされる別のプログラムの関数の性質として追加した際に，異なる不変式を導く場合があることから，検証結果と関数の性質との詳細な関係について研究を進める必要がある．また，本稿で対象としたプログラムにはリストや配列などデータ構造が含まれておらず，これらを扱えるよう対応する必要がある．さらには本手法に基づいた実装による実験を行い，確かに計算機による自動生成によって目的となるループ不変式が得られることや，得られたループ不変式の有効性を検討する必要がある．

謝辞　本研究は，一部，文部科学省科学研究費#20300010, #20500008, #21700011 の助成を受けたものである．

文　献

[1] F. Baader and T. Nipkow. *Term rewriting and all that.* Cambridge University Press, 1998.
[2] A. R. Bradley and Z. Manna. Verification constraint problems with strengthening. In K. Barkaoui, A. Cavalcanti, and A. Cerone, editors, *ICTAC*, volume 4281 of *Lecture Notes in Computer Science*, pages 35–49. Springer, 2006.
[3] A. R. Bradley and Z. Manna. Property-directed incremental invariant generation. *Formal Asp. Comput.*, 20(4-5):379–405, 2008.
[4] M. Colón, S. Sankaranarayanan, and H. Sipma. Linear invariant generation using non-linear constraint solving. In *CAV*, pages 420–432, 2003.
[5] P. Cousot. Proving program invariance and termination by parametric abstraction, lagrangian relaxation and semidefinite programming. In R. Cousot, editor, *VMCAI*, volume 3385 of *Lecture Notes in Computer Science*, pages 1–24. Springer, 2005.
[6] P. Cousot and R. Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *POPL*, pages 238–252, 1977.
[7] R. W. Floyd. Assigning meanings to program. In *Proc. Symposia in Applied Mathematics*, volume 19, pages 19–32, 1967.
[8] A. Gupta and A. Rybalchenko. Invgen: An efficient invariant generator. In *CAV*, pages 634–640, 2009.
[9] C. A. R. Hoare. An axiomatic basis for computer programming. *Commun. ACM*, 12(10):576–580, 1969.
[10] E. Rodríguez-Carbonell and D. Kapur. An abstract interpretation approach for automatic generation of polynomial invariants. In *Static Analysis, 11th International Symposium, SAS*, volume 3148, pages 280–295. Springer, 2004.
[11] E. Rodríguez-Carbonell and D. Kapur. Automatic generation of polynomial invariants of bounded degree using abstract interpretation. *Sci. Comput. Program.*, 64(1):54–75, 2007.
[12] S. Sankaranarayanan, H. Sipma, and Z. Manna. Non-linear loop invariant generation using gröbner bases. In *POPL*, pages 318–329, 2004.
[13] S. Sankaranarayanan, H. B. Sipma, and Z. Manna. Constraint-based linear-relations analysis. In R. Giacobazzi, editor, *SAS*, volume 3148 of *Lecture Notes in Computer Science*, pages 53–68. Springer, 2004.
[14] S. Srivastava and S. Gulwani. Program verification using templates over predicate abstraction. In *PLDI*, pages 223–234, 2009.

NII-Electronic Library Service