北海道大学工学部研究報告
第81号（昭和51年）

Bulletin of the Faculty of Engineering,
Hokkaido University, No. 81 (1976)

プログラム停止性に関するある種の検証システム

浦崎道教* 柳 繁* 新保 勝* 河口至商*

（昭和51年3月31日受理）

A Verification Condition Generator System for Program Termination

Michinori URASAKI, Shigeru YANAGI, Masaru SHIMBO
and Michiaki KAWAGUCHI

(Received March 31, 1976)

Abstract

An implementation of a system we have described in this paper is concerned with an automatic verification condition generator for the termination of computer programs. This is performed by the inductive assertion method, and the system is a deductive one which defines the semantics of an ALGOL-like language. The problem whether any program terminates or not is reduced by this implementation to verify a set of verification conditions thereof.

1. まえがき

プログラムを数学的対象として把えた場合，プログラムの性質として，正当性，同値性，停止性などが考えられる。これらの性質を研究するために，従来，いろいろなアプローチが考えられてきた。正当性，停止性を考える場合に有力なアプローチとして，述語論理を用いてプログラムを形式的に扱う方法がある。すなわち，プログラムの性質を証明するためにこの問題を述語論理でかかれた論理的条件に帰着させるのである。この論理的条件を検証条件（Verification Condition）と呼ぶ。この方法は，1967年に R. W. Floyd[1] によって考えられた。

Floyd の方法は更に発展させられて，正当性に関しては，1969年に C. A. R. Hoare[2] によって，また停止性に関しては，1973年に Z. Manna and A. Pnueli[4] によって理論的基礎が確立された。しかし，実際に実用的なプログラムの正当性，停止性を自動的に証明しようとする立場から考えると，Hoare や Manna et al. の理論は適していない。そこでプログラムの正当性のための検証条件を自動的に生成するシステム（Verification Condition Generator: VCG）が，1973年に S. Igarashi, R. L. London and D. C. Luckham[3] によって考えられた。彼らは正当性に関して相当強力なシステムを試作した。

しかし，停止性に関してはまだ理論的考察しかなされていない。停止性のための自動立証システムは正当性の場合と同様，重要である。本論文はプログラムの停止性のための立証条件を自動的に生成するシステム—停止条件生成システム（Termination Condition Generator: TCG)—を試作することを目的としている。本論文で扱うプログラムのクラスは while プログラムのクラ

* 工学部情報数理工学第一講座

スである。while プログラムは ALGOL 型言語で書かれ，割当て文，条件文，そして while 文が許されている。

2. 公理と推論規則

与えられたプログラムが確かにその仕様 (Specification) を満たしていることを言うためには，プログラミング言語のセマンティクスを正確に定義しなければならない。そして，プログラミング言語を定義するためにはある演繹体系が必要となってくる。そうすると，プログラムの性質は原理的には演繹体系を用いてプログラムそれ自体から証明されることになる。演繹体系は公理と推論規則から成り立っている。次にこの公理的アプローチの方法について述べる。

正当性を証明するための関係式を，Hoare の記法を用いると，次のような定理の形で表現することができる。

$$\{P(x) \mid B \mid Q(x)\}$$

ただし，P, Q は述語であり，B はプログラムの一部分である。また x はプログラム変数である。この記法による意味は，もし P が B の実行前に真であるとすると，B の実行後，Q も真であるということである。そうすると，プログラム A のステートメントに関して公理から出発し，推論規則を使って

$$\{\phi(x) \mid A \mid \psi(x)\}$$

を演繹することができたとすると，その時，ϕ と ψ に関してプログラム A の正当性が示されたと言う。ただし，ϕ, ψ を各々，入力述語，出力述語と呼ぶ。

一般に検証条件を自動的に求めるためには，プログラム A と入出力述語 ϕ, ψ だけでは不十分であることが知られている。そのため，プログラムの停止性に関しては，while 文に対して帰納的アサーション (Inductive Assertion) と停止関数を与えることで停止条件を生成し，証明することができる。帰納的アサーションとは，while 文に対してプログラム変数間の関係を表わす述語である。

ところで停止性の証明をしようとする場合，理論的には Hoare の演繹体系と同様な方法で考えることができる。しかし，停止条件を自動的に求める立場から考えると，

(i) プログラム変数の書き換えの必要性

(ii) 推論規則の一意性

が問題となってくる。本論文では以上の 2 点を解決するために，停止性のための公理と推論規則に次のような新しい記法を導入する。

$$\langle F(x) \mid B \mid F(x) \rangle$$

ただし，F は停止関数，B はプログラムの一部分，x はプログラム変数である。この記法による意味は，B を実行したときに

$$F(x) \succ F'(x)$$

が成り立つということである。

ここで，F' は B の実行前の停止関数 F が B の実行によってプログラム変数 x を変化させた結果であることを意味する。$\succ$ は半順序関係を表わし，空でない集合を W とすると，$(W, \succ)$ は半整列集合を表わす。半整列集合とは，無限減少列を含まない半順序集合であり，$F(x) \in W$ である。この記法はプログラムにループがある場合，そのループが無限ループか否かを判定するための重要な役割りを果たす。

プログラムの停止性のための公理と推論規則を停止規則と呼び，停止関数を記述するための公

理と推論規則を停止関数規則と呼ぶ。

推論規則は次のような形で表わされる。

$$\frac{P}{R}, \qquad \frac{P, Q}{R}$$

T 1. 割当て規則

$$\frac{\{P \mid B \mid Q(e)\}}{\{P \mid B\,;\ x := e \mid Q(x)\}}$$

T 2. 結果の規則

(i)
$$\frac{P \supset Q}{\{P \mid \mathrm{Null} \mid Q\}}$$

(ii)
$$\frac{\{P \mid B \mid S \supset Q\}}{\{P \mid B\,;\ S\text{-if} \mid Q\}}$$

T 3. 条件規則

$$\frac{\{P \mid B\,;\ S\text{-if}\,;\ C \mid Q\} \quad \{P \mid B\,;\ \neg S\text{-if}\,;\ D \mid Q\}}{\{P \mid B\,;\ \text{if } S \text{ then } C \text{ else } D \mid Q\}}$$

T 4. while 規則

$$\frac{\{P \mid B \mid I\} \quad I \wedge \neg S \supset Q \quad I \supset F \in W \quad \{I \wedge S \mid C \mid I\} \quad I \wedge S \supset \langle F \mid C \mid F \rangle}{\{P \mid B\,;\ \text{while } S \text{ do } C \mid Q\}}$$

表 1 停止規則の集合：T

F 1. 割当て規則

(i)
$$\frac{\langle F \mid B \mid F'(e) \rangle}{\langle F \mid B\,;\ x := e \mid F'(x) \rangle}$$

(ii)
$$\frac{\langle F \mid B \mid S(e) \supset F'(e) \rangle}{\langle F \mid B\,;\ x := e \mid S(x) \supset F'(x) \rangle}$$

F 2. 結果の規則

(i)
$$\frac{F \succ F'}{\langle F \mid \mathrm{Null} \mid F' \rangle}$$

(ii)
$$\frac{S \supset F \succ F'}{\langle F \mid \mathrm{Null} \mid S \supset F' \rangle}$$

(iii)
$$\frac{\langle F \mid B \mid S \supset F' \rangle}{\langle F \mid B\,;\ S\text{-if} \mid F' \rangle}$$

F 3. 条件規則

$$\frac{\langle F \mid B\,;\ S\text{-if}\,;\ C \mid F' \rangle \quad \langle F \mid B\,;\ \neg S\text{-if}\,;\ D \mid F' \rangle}{\langle F \mid B\,;\ \text{if } S \text{ then } C \text{ else } D \mid F' \rangle}$$

F 4. while 規則

$$\frac{\langle F \mid B \mid F' \rangle}{\langle F \mid B\,;\ \text{while } S \text{ do } C \mid F' \rangle}$$

表 2 停止関数規則の集合：F

注) (1) P, Q, S は述語である。
(2) Null は空プログラムを表わす。
(3) 各規則において B は Null でもよい。
(4) C, D はプログラムの一部分である。
(5) S-if, $\neg S$-if はマークされた述語である。
(6) I は帰納的アサーションである。
(7) F, F' は停止関数である。
(8) F 4 には次のような仮定がある。ネストされているプログラムに対して外側の while 文の停止関数は内側にネストしている while 文によって影響されないものとする。

P, Q は前提（この規則が適用可能であるための条件）であり，R は結果（推論される述語）である。前提はすでに成り立っている述語か，別に前で証明されている述語である。

停止規則は表1に示す。停止関数規則は表2に示す。

3. TCG と例

TCG とは与えられたプログラムに対して停止規則と停止関数規則を実行するシステムであると言える。TCG への入力は while プログラム，入出力述語，帰納的アサーション，および停止関数である。TCG の出力は停止条件である。プログラムの停止性の問題は，TCG によって述語論理で書かれた数学的補題を証明する問題に還元されるのである。その方法は本質的にはプログラムステートメントの各タイプに対して公理または推論規則が正確にそのタイプに適用することができるようにすることである。

この TCG によって使われた例について述べる。図1は最大公約数を計算するプログラム (GCD) である。

```
begin
y1 : =x1;
y2 : =x2;
while  y1 ¬= y2 do
       if  y1 > y2  then  y1 : =y1−y2
                    else  y2 : =y2−y1;
z : =y1
end;
```

図1 z=GCD ($x1, x2$) を計算するプログラム

このプログラムに対して

入力述語 $\phi(x1, x2)$: $x1 > 0 \wedge x2 > 0$

出力述語 $\psi(y1, y2)$: $y1 = y2$

帰納的アサーション I : $y1 > 0 \wedge y2 > 0$

停止関数 $F(y1, y2)$: Max($y1, y2$)

を与える。ここで半整列集合として $(N, <)$ をとる。N は自然数の集合であり，$<$ は普通の意味の不等号を表わす。

次にこのプログラムに対して，停止規則，停止関数規則を適用すると，TCG は図2のような停止条件を生成する。

```
TC 1.  Y1>0 & Y2>0 ⊃ MAX(Y1, Y2)>0
TC 2.  Y1>0 & Y2>0 & ¬(Y1¬=Y2) ⊃ Y1=Y2
TC 3.  Y1>0 & Y2>0 & Y1¬=Y2 & ¬(Y1>Y2) ⊃ MAX(Y1, Y2)>MAX(Y1, (Y2−Y1))
TC 4.  Y1>0 & Y2>0 & Y1¬=Y2 & Y1>Y2 ⊃ MAX(Y1, Y2)>MAX((Y1−Y2), Y2)
TC 5.  Y1>0 & Y2>0 & ¬(Y1>Y2) & Y1¬=Y2 ⊃ Y1>0 & (Y2−Y1)>0
TC 6.  Y1>0 & Y2>0 & Y1>Y2 ⊃ (Y1−Y2)>0 & Y2>0
TC 7.  X1>0 & X2>0 ⊃ (X1)>0 & (X2)>0
```

図2 停止条件のリスト

TC 1～TC 7 はすべて真であることは容易にわかるので，GCD のプログラムは停止すると言える。他の例として，巾乗，割り算，平方根などを計算する整数上の算術プログラムが TCG によって調べられている。TCG のプログラムは *PL/I* で書かれている。プログラム例の停止条件を出力するために，北海道大学大型計算センター FACOM 230/75 を使用した。

4. あ と が き

プログラムの停止性について，停止条件を自動的に生成するための演繹体系および TCG について考察した。その結果，簡単な while プログラムに対しては，TCG の試作は可能であることがわかった。これは従来の記法による停止規則に加えて，新しい記法による停止関数規則をつくることによって，プログラム変数の書き換えをする必要がなくなり，また推論規則も一意的に定めることができたことによるものと考えられる。

取り扱うプログラムのクラスに，更に，配列割当て文，go to 文，procedure call なども含むようにすると，TCG の機能が強化されるので，一層複雑なプログラムも扱うことができるものと思われる。

参 考 文 献

1) Floyd, R. W.: Proc. Symp. Appl. Math., Amer. Math. Soc., 19, pp. 19–32 (1967).
2) Hoare, C. A. R.: C. ACM, 12 (10), pp. 576–580, 583 (1969).
3) Igarashi, S., London, R. L. and Luckham, D. C.: Stanford Univ. AI-MEMO 200 (1973).
4) Manna, Z. and Pnueli, A.: Acta Informatica 3, pp. 243–263 (1973).