人工知能

第8回講義課題 課題番号23

提出締切: 2014.02.06

提出日:　2014.02.05

工学部電子情報工学科

03-123006 岩成達哉

1　概要

Chaitin の Ω は，無作為に選ばれたプログラムが停止する確率を表した定数であり，定義はできるが計算は出来ない．しかし，ある論文では，その最初の 64 ビットが求まったと書かれている．本課題では，この証明について簡単に説明し，その意義を考察した．

2　証明の説明

参考文献 [1] の証明を簡単に説明する．

2.1　対象

対象とするのは，文献 [1] の著者らが定義した，命令セットと文法に従うプログラムであり，このプログラムに任意の入力を与えたときにそのプログラムが終了 (停止) するかどうかを判定する．

プログラムは，アルファベットが a から z までの 26 種，特殊記号が 12 種，空白 2（space と tab）種，数字が 0 から 9 までの 10 種の計 50 種類の記号と，定数（0 から 127）によって記述できる．

記述されたプログラムは，空白 2 種が除かれ，プログラムの引数の関係 (例えば，空白を削除した際に”1 1”と”11”を区別できるように) がわかるように適宜”,”が追加された形に圧縮される．つまり，圧縮されたプログラムで現れる記号は $50 - 2 + 1 = 49$ 種類である．

圧縮されたプログラムは，さらに 1 文字 (1 記号) が 7bit で表される．そして，このプログラムを 7bit ずつ長くして，あらゆるプログラムを作成し，任意の入力を与えてその停止確率を見るのである．この文献では，84bit までで表せる (プログラム+データ) について検証している．

2.2　必要な条件

仮定する条件を以下に示した．

1. 宣言のない変数を用いた場合や文法を満たさない場合などは実行エラーとなり，停止しないと見なす
2. あまりに多くの命令を実行する場合は，停止しないプログラムであるとする

1 の仮定は，実行エラーを表すから終了するとしても良いが，2.3.4 節で説明する理由から，停止しないと見なすことが望ましい．

2 の仮定は,「実は停止するがあまりに多くの命令を実行するため停止しないと判定される」プログラムに謝った判定をする可能性があるが，停止しないと見なす回数の閾値をプログラムの長さに対して適切に設定すれば，表現できるプログラムの種類に限りがあることから，停止しないプログラムを断定できる．文献 [1] では，この閾値を 100 としている．文献で扱われたプログラムでは，100 ステップを超えて終了するプログラムを書くためには，300bit 程度が必要であり，対象とされた 84bit までのプログラムでは，この閾値で十分である．

2.3　Ω の求め方

実際に Ω を求める手順を，実例を示しながら以下に示した．

2.3.1　1 文字で表せるプログラム

文献のプログラムでは，プログラムの停止を表す命令は%で表すが，1 文字で停止するプログラムは”%”だけであり，他の命令では 2.2 節の条件より，停止しないプログラムとなる．また，これを含まないものは，たとえ入力を矛盾なく処理し終わっても，終了したとはみなされない．

2.3.2 2 文字で表せるプログラム

文法の定義から%はプログラムの最後に限って出現するため，2 文字でかける (プログラム+データ) で停止するものは”\%”の 1 つだけである．

2.3.3 2 文字で表せるプログラム+データ

プログラムと入力データを区別せずにつなげて書くことで，2 文字で表せるプログラム+データ (なしでもよい) の組について考えてみる．まず，プログラムが 1 文字である場合，終了するのは”%”だけであるのは前述のとおりである．すると，それ以降は，データの列とみなされるが，このプログラムは入力を一切読まず終了するため，次の 1 文字 (7bit) がなんであれ終了する．よって，$2^7 = 128$ 個だけ終了するものがある．また，プログラムが 2 文字である場合に終了するのは，上記のとおり”\%”だけである．

このように，プログラムとデータを組として考え，7bit ずつ拡張していけば,「任意のプログラム」と「任意の入力」を同様に大きくしながら見ていくことができる．

2.3.4 n 文字で表せるプログラム+データ

前節では,

> 「2 文字の (プログラム+データ) では，1 文字の (プログラム+データ) で終了するもの (つまり”%”) が先頭に来る場合，必ず終了する」

ということがわかった．

実は,

> 2 以上の任意の整数 m と $m > n$ である正の整数 n について,「m 文字の (プログラム+データ) では，n 文字の (プログラム+データ) で終了するものが先頭に来る場合，必ず終了する」

といえる．このことは，次のように考えればわかる．

ある s 文字のプログラムがあり，そのプログラムが終了するような t 文字の入力データがつなげて書かれた，$(s+t)$ 文字の (プログラム+データ) の組があるとする．このとき，その (プログラム+データ) の組の後に任意の 1 文字を追加して $(s+t+1)$ 文字の (プログラム+データ) の組をつくる．すると，この (プログラム+データ) の組は，先頭から $(s+t)$ 文字目までを処理し，追加した文字を読まずに終了する．つまり，これ以降にどんなデータがこようと終了する．よって，終了することがわかっている (プログラム+データ) を先頭にもつ (プログラム+データ) は，必ず終了する．したがって，上記のことが示された．

また，このときに,「実行エラーとなるものは停止する」としてしまうと，文字数を増やすことでその実行エラーが解消された場合に，そのプログラムが停止しなくなる可能性があり，上記を満たさなくなってしまう．よって，実行エラーは停止しないとみなすと都合が良い．

2.3.5 Ω の求め方

実際に Ω を求めていく．求め方は単純で，n[bit] の (プログラム+データ) で停止するものを数え，それを n[bit] で表現できる全ての場合 (2^n) で割るということをすれば良い．文献のプログラムでは，7bit ずつプログラムが伸びていくため，7 の倍数の bit 数を持つ (プログラム+データ) について考えればよい．

まず，1 文字 (7bit) で表現できる (プログラム+データ) は 2^7 通りであり，停止するものは前述のとおり 1 つだけであるから，Ω_n を n[bit] でかける (プログラム+データ) の停止確率だとすると，

$$\Omega_7 = \frac{1}{2^7}(10\text{ 進数}) = 0.0000001(2\text{ 進数}) \tag{1}$$

となる．次に，2 文字 (14bit) で表現できる (プログラム+データ) は 2^{14} 通りであり，停止するものは，1 文字で終了するものを先頭に含む 2^7 個と，”\%”の 1 つを合わせれば良いから，

$$\Omega_{14} = \frac{2^7+1}{2^{14}}(10\text{ 進数}) = 0.00000010000001(2\text{ 進数}) \tag{2}$$

となる．

ここで，

$$\triangle\Omega_{14} = \Omega_{14} - \Omega_7 = 0.00000000000001(2\text{ 進数}) \tag{3}$$

のように確率の変化量を書くと，前節で示したように「終了する (プログラム+データ) が先頭にくるような (プログラム+データ) は必ず終了する」ため，それを除いたものを数えて全ての場合で割った値となっていることがわかる．つまり，

$$\triangle\Omega_k = \Omega_k - \Omega_{k-7} \tag{4}$$

は必ず 0 以上となり，それを求めるためには，文字数を増やしたことによって停止するようになった (プログラム+データ) を数えればよいとわかる．

このように考えれば，Ω を求めるためには，bit 数を増やしながらその数を加算すれば良いことがわかり，探索する対象も少なくなって計算量を減らすことができる．

これは，bit 数を増やした時に調べる対象に,「語頭条件[1]」が適用されたことを意味する．

そして，語頭条件があるために，これらを順に加算した場合，それが [0, 1] の確率となることが示され，したがって Ω を正しく確率として求めることができる．

このように，停止する確率をプログラムとデータの長さをあわせて 1 文字，2 文字，⋯ と増加させていき，それぞれで何個のプログラムが終了するかを数えることで確率の増加量を求め，その値を加算していく．論文では，この操作を繰り返し，84bit までで得られる停止確率を求めている．

2.4 論文での 64bit の求め方

停止確率の先頭から数 bit を求める方法のポイントは，このそれぞれの bit 数での確率の増分 $\triangle\Omega_n$ が，Ω に与える影響は bit 数が増加するに応じて小さくなっていくことである．例えば，今まで扱った 1 文字と 2 文字の例では，その増分はとても小さい数値であった．

具体的に Ω_n を列挙することでその様子を見てみると，2 進数で表した場合，以下のようになる．

$$\Omega_7 = 0.0000001 \tag{5}$$
$$\Omega_{14} = 0.00000010000001 \tag{6}$$
$$\Omega_{21} = 0.000000100000010000011 \tag{7}$$
$$\Omega_{28} = 0.00000010000001000000110001000 \tag{8}$$
$$\Omega_{35} = 0.00000010000001000001100010000110010 \tag{9}$$
$$\Omega_{42} = 0.00000010000001000001100010000111010001101111 \tag{10}$$
$$\Omega_{49} = 0.00000010000001000001100010000111010001111101110100 \tag{11}$$
$$\Omega_{56} = 0.00000010000001000001100010000111010001111110010110001110 \tag{12}$$
$$\Omega_{63} = 0.00000010000001000001100010000111010001111110010111011001110111001 \tag{13}$$
$$\Omega_{70} = 0.00000010000001000001100010000111010001111110010111011100111001111001001 \tag{14}$$
$$\Omega_{77} = 0.00000010000001000001100010000111010001111110010111011101000001110000010100111 1 \tag{15}$$
$$\Omega_{84} = 0.00000010000001000001100010000111010001111110010111011101000010000011110110110110111 01 \tag{16}$$

この様子からも，bit 数が増えるに従って，少しずつ先頭 bit が変わらなくなってきていることがわかる．

では，具体的に，84bit のときはこれから文字数を増やしても変わらない bit は先頭からどこまででであろうか．これを論文では計算して，文字数を 1 つ増やした時に最大でも 1×2^{-68}，つまり 68bit に加算が行われる可能性があることを示している．Ω_{84} は 66, 67, 68bit 目が 1 であるために，繰り上がりによって 65bit 目が変更される可能性がある．よって，64bit までが正しいことを示している．

---

[1] 語頭条件とは,「どの符号も他のどの符号の語頭に来ない」という条件であり，ハフマン符号などがこの条件を満たしている．例えば，記号 {a,b,c} の 3 つを 2 進数で表す場合，a を 0，b を 1，c を 10 というような割り当て方ができる．しかし，これでは，b の割り当てが 1 であり，c の割り当て符号の語頭に現れている．したがって，この割り当て方は語頭条件を満たしていない．一方，a を 0，b を 10，c を 11 という割り当てをすると，どの符号も他のどの符号の語頭になっておらず，語頭条件を満たしている．

この条件は，符号を先頭から見ていって，割り当ての中から当てはまるものがあれば，その記号に即座に翻訳できることを意味する．例えば，前者の割り当てでは，010 を ac とも，aba とも解釈出来てしまう．

3 論文の訂正と考察

3.1 訂正

論文では，64bit まで正しく求めたと述べられた．しかし，筆者らが求めた計算の過程に誤りであると思われる箇所があった．まず，文献の 7 章にある，式 (2) の第 1 項について，

$$\sum_{m=0}^{\infty}\sum_{n=13}^{\infty} 402906842 \cdot 48^{n-13} \cdot 2^{m} \cdot 128^{-(n+m+1)} = 402906842 \cdot \frac{64}{128 \cdot 48^{13}} \sum_{n=13}^{\infty}\left(\frac{48}{128}\right)^{n} \tag{17}$$

となっているが，この項は，m についての等比数列の和が正しく計算されておらず，実際には，$\frac{1}{63}$ をかけなければならない．つまり，

$$\sum_{m=0}^{\infty}\sum_{n=13}^{\infty} 402906842 \cdot 48^{n-13} \cdot 2^{m} \cdot 128^{-(n+m+1)} = 402906842 \cdot \frac{64}{63 \cdot 128 \cdot 48^{13}} \sum_{n=13}^{\infty}\left(\frac{48}{128}\right)^{n} \tag{18}$$

が正しい．ただし，この式で現れる値については，筆者らが作成した Java プログラムによって正しく求められたと仮定している．

このように考えると，第 1 項は小さくなるため，筆者が示した 2^{-68} よりも小さい数値で抑えられるのではないかと考えられるが，このことを踏まえて計算をし直すと，

$$\sum_{m=0}^{\infty}\sum_{n=13}^{\infty} 402906842 \cdot 48^{n-13} \cdot 2^{m} \cdot 128^{-(n+m+1)} + \sum_{m=1}^{\infty} 1748380 \cdot 2^{m} \cdot 128^{-(m+13)} \tag{19}$$

$$= \quad 402906842 \cdot \frac{64}{63 \cdot 128 \cdot 48^{13}} \sum_{n=13}^{\infty}\left(\frac{48}{128}\right)^{n} + 1748380 \cdot \frac{1}{63 \cdot 128^{1}3} \tag{20}$$

$$\simeq \quad 2.06655796 \times 10^{-21} + 1.12089691 \times 10^{-23} \tag{21}$$

$$= \quad 2.077766929 \times 10^{-19} \tag{22}$$

となり，2 の何乗に当たるかを調べると，

$$\log_2(2.077766929 \times 10^{-19}) = -62.061599972\cdots \tag{23}$$

となることから，2^{-62} が上界であることがわかる．つまり，文献の計算結果が間違っており，実際には，繰り上がりを考慮しても 62bit までが正しいことがわかった．

これは，筆者らの計算が間違っているか，示された途中式が異なるからであると推察される．

3.2 考察

この Ω の値は，筆者らが設定した命令セット，構文を持つプログラムでは，いくつかの定理を基に，上記のような手順を踏んでほぼ正しく求められている．しかし，このような試行を続けても，実際には Ω を完全に求めることはできず，また bit 列の規則すら見つけることができないはずである．これは，Ω が持つランダムの性質として備わった事実である．また，このような手法は今回のような Machine には適用できるが，一般的に Ω を求めることが出来ないことに変わりはない．

特に，今回は 2.2 節で示したように，100 回のステップを行っても終わらないものは，停止しないと判定すると仮定したが，プログラムの長さが長くなると，100 回を超えたステップで終了するプログラムが容易に作れてしまう．つまり，プログラムが長くなると，それに応じてこの閾値を変更する必要がある．しかし，任意の長さのプログラムにおいて，何ステップ以上であれば無限ループとみなせるかということは決定不能な問題となっている．よって，bit 数が十分大きくなると，停止するかどうかを判定できなくなるため，やはり Ω は計算できないのである．

次に Ω と関わりのある停止問題について考えてみる．まず，停止問題の決定不可能性を示しておく．背理法によって示す．

「あるプログラム x が与えられた時に，そのプログラムが停止するかを判断する関数を $H(x)$ が存在し，停止する場合は $H(x)$ =YES，停止しない場合は $H(x)$ =NO と答えるとする．このときに，あるプログラム M を，「$H(M)$ =YES なら無限ループ，$H(M)$ =NO なら停止する」と定義すると，M は停止するか」という命題を取りあげる．この問は，矛盾を含むため，そのような関数 $H(x)$ は存在しない．ゆえに，停止問題は決定不可能である．

この論文では 84bit までで表せるものまでしか取り扱っておらず，bit 数を増やしたときに停止問題の決定不可能性を示す $M(x)$ のようなプログラムが作成できる場合には上記の理由から停止問題は決定できない．逆に言えば，それまではプログラムの停止性を判定でき，よって，Ω の先頭 bit を求めることができるが，結局のところ Ω を求めきることはできない．

また，停止問題では，n[bit] のプログラムの停止問題を解くためには，Ω の最初の n[bit] が得られなければならないが，本論文では，84[bit] のプログラムを与えて，Ω の最初の 64[bit] しか得られていないため，不十分である．

一方で，この論文は，プログラミングと数学的証明をあわせて，初めて Ω を求めようとしたという点では意味がある．これは，四色問題が解かれた時のように，数学とコンピュータを組み合わせることで成果を得た例の一つであるといえよう．特に，Ω は計算不可能であることが示されているため，このような手法が新しいアプローチとして試されたことは意味があったといえるだろう．

また，この論文では，Chaitin の Ω の数 bit を求めたことよりも，むしろ ZFC という理論での停止問題の見識が，主な成果であると筆者らが主張しているが，これについては，課題の範囲を超えるため調べていない．

4　結論

本課題では，計算不可能であることがわかっている Chaitin の Ω が，ある種の問題においては最初の数 bit を，数学的証明とコンピュータで求めることができるという論文を解説した．一方で，その意義について考察し，Ω の性質や，停止問題は解決できないことを用いて,「最初の数 bit」までしか求められないことを述べた．

参考文献

[1] Calude,C.S., Dinneen,M.J. and Shu,C.-K.,“ Computing a Glimpse of Randomness, ”Experimental Mathematics, vol.11, pp.361-370, 2002.