チューリングの停止性問題

須藤賢

Disclaimer : このエッセイは独自解釈を多数含むが，著者は計算理論を専門に学んだことはないため，いくつかの点で不正確な可能性がある

目次

0. イントロダクション

この短いエッセイでは 1936 年のチューリングの記念碑的な論文である『計算可能数とその決定問題への応用(“On Computable Numbers, With an Application to the Entscheidungsproblem”)』[1]の中で初めて示された計算不能な機械の証明に関して，簡単な説明を行うことを目的とする(このエッセイでは，チューリングの論文だけでなく，[2]を大いに参考している).

本エッセイで目的とするチューリングの証明は，その後停止性問題として広く知られるようになるものであるが，そこでの多数の道具仕立て，計算機械の構成法や算術化等は，現代的に洗練された解説とは趣が異なっており，今でも一読の価値はあると思われる.

本エッセイは 2 部構成である．基本的には両方とも独立して読めるように記述している.

第 1 章では，停止性問題に関して現代的な用語を用いて一般的な解説を行う．ここでの内容は，再帰呼び出しや高階関数等を含むプログラミングの基礎的な知識のみを仮定し，具体的な例示としては Python を用いる.

第 2 章において，チューリングが実際に示した計算不能な機械の証明を行う．内容としては，証明に必要な事項のみに絞ったクラッシュコースではある．計算機械の構成に関しては，基本的にチューリングの論文を追う形で，1 つ 1 つの基礎的な定義を積み重ねて解説を行うため，あまり予備知識は必要としない．しかし，計算機械を数え上げる 2.3 以降では，集合の簡単な知識(写像，濃度等)が必要である(この辺り知識は[3]を参照).

1. 停止性問題とは

停止性問題(Halting Problem)とは，その名の通り，あるプログラムが有限時間内に停止するかどうかに関わる計算理論上の問題である．この問題は，チューリングの計算不能な機械の証明が行われた 1930 年代当時とは異なり，「"計算"とは何か」ということがコンピュータやプログラミングの普及に伴って，十分に直感的な概念になった現代においては，ごく簡単に理解可能なものである．[ここで述べている”計算”とはアルゴリズムのことであり，”計算可能”とは，ある問題を解決するアルゴリズムが存在することである．逆に，ある問題が計算不能とは，それを解決するアルゴリズムの不在を意味する]

簡単に停止性問題から導かれる結論を述べれば次のようになる：「任意のプログラムが停止する(無限ループしない)ことを判定するようなプログラムは計算可能でない」．[第 1 章では，チューリングの”機械”とプログラムを同一視する]

1.1. 停止性問題の証明

ざっくり証明の道筋を記述すると，しかるに任意のプログラムの停止性を判定するようなプログラムを計算可能と仮定すると矛盾が生じることが示せる．[背理法を忘れている人は少し記憶を辿って思い出してほしい．背理法では，真と思われる命題をあえて偽であると仮定し，推論を進めて矛盾を導くことにより，偽であると仮定した命題が真であることを証明するのであった]

これをプログラミング言語(Python)の形式に従って証明することは容易である．引数を取らない任意のプログラム f について，次のようなプログラム halt が構成可能(計算可能)であると仮定すると，矛盾を生じるプログラム G が構成できる．

$$halt(f) = \begin{cases} True & f\text{は無限ループしない} \\ False & f\text{は無限ループする} \end{cases}$$

```
G = lambda : G() if halt(G) else True
```

ここで実際に G を実行する場合を考えてみよう．G の動作は無限ループするか停止するかのいずれかである．[現実の Python では，構文的に無限ループする再帰呼び出しであっても，直ちに最大再帰深度に到達して停止するが，ここでは構文的に無限ループするならば実際に無限ループすると考える．少し言葉を選ぶと，ここでの Python は可能無限を許容するような計算モデルであると想定する]

まず G は無限ループすると仮定する．すると G は無限ループするので G の定義における halt(G) は False を返し，if 文としては True が返って G は停止する(無限ループしない)．G が無限ループすると仮定して，G が停止することが示された；これは明らかに矛盾である．同様にして，G は停止するという仮定から，G は無限ループすることが示せる．

G について考えられる動作のいずれの場合においても矛盾が生じる．そしてこれらは halt が構成可能であると仮定した結果として導かれた．故に仮定は誤りであり，halt は構成不能（計算不能）である．

1.2. 停止性問題の意義

先に簡単に証明を示した停止性問題の意義を端的に述べれば「計算できないこと」の具体例をほぼ初めて示したということになるだろう．

停止性問題は無論，否定的に解決された問題であるが(プログラムが停止するかは一般に判定できない)，もしこれが肯定的に解決されていたなら，100 万ドルの賞金がかけられている数学上の未解決問題であるリーマン予想でさえ，それを逐次的に検証するプログラムの停止性を評価すれば，瞬く間に解決され得たであろうことを思えば，それの深遠さの一端がわかるだろう．

完全に空想的ではあるが，停止性を評価することにより，ゴールドバッハ予想を証明する例を次に示そう．

```
def sieve_of_eratosthenes(n):
    a, b = [], list(range(2, n + 1))
    while b[0] < pow(n, 0.5):
        a += b[:1]
        b = [k for k in b[1:] if k % b[0] != 0]
    return a + b

def sum_of_two_primes(n):
    p = sieve_of_eratosthenes(n)
    return any([i + j == n for i in p for j in p])

def goldbach_conjecture():
    n = 4
    while sum_of_two_primes(n):
        n += 2

if not halt(goldbach_conjecture):
    print('Goldbach Conjecture is TRUE!!')
```

ゴールドバッハ予想とは，4 以上の全ての偶数は 2 つの素数の和で表すことができると主張する数学上の予想である．

ここでの例では goldbach_conjecture において，偶数が 2 つの素数で表せるかを逐次的に評価し，予想が正しければ際限なく検証を続けるようになっている．これが停止しない(halt の返り値は False)ならばゴールドバッハ予想は肯定的に，停止する(halt の返り値は True)ならば否定的に解決されるわけであるが，いずれにしても証明はなされる．

1.3. 停止性問題の応用

停止性問題には，その他にも重要な応用が存在する：多くの計算不能なプログラムは停止性問題に帰着され得る．1 つ例を挙げよう．任意のプログラムに関して”Hello,World!”を出力するかどうか判定するプログラム hw は計算不能である．

$$hw(f) = \begin{cases} True & f\text{は"Hello,World!"を出力する} \\ False & f\text{は"Hello,World!"を出力しない} \end{cases}$$

なぜならば, hw を計算可能であると仮定すると，hw を用いて次のようなプログラム HALT が構成可能である．

```
def HALT(f):
    def sub():
        f()
        print("Hello,World!")
    return hw(sub)
```

f が無限に動作するならば，sub の定義における print は永遠に実行されないため，hw は False を返し，HALT もこれに従う．f が無限に動作しないならば，print は最終的に実行されて hw は True を返し，HALT もこれに従う；この動作は halt を不可分なく実現し，計算可能としている．しかしながら，この結果は，halt が計算不能と証明さている事実と明らかに矛盾する．故に，この結果を導いた仮定は誤りであり，hw は計算不能である．[一般にこのような形で記述される問題，「任意のプログラムが特定の性質を持つかどうか判定するプログラム」は計算不能であることが証明されている．これはライスの定理と呼ばれ，現実のソフトウェアテストの”完全な自動化”が不可能である理論的な根拠となっている]

2. チューリングの停止性問題

第 1 章では総花的な内容として，停止性問題の簡易的な証明と意義，応用までを述べたが，ここからは本題のチューリングが行った計算不能な機械の証明の説明を行う．[チューリング自身は論文の中で停止性問題という用語は使っていない．Halting Problem という用語は，Martin Davis の著書『Computable and Unsolvability』に由来するらしい ([2] からの孫引き)]

『計算可能数とその決定問題への応用』の中で，チューリングは，素朴な対角線論法の適用から興味深い計算機械に関連した集合の性質を示したのちに，そこからの直接の敷衍として計算不能な機械を示している．

ここからの解説は，チューリングの論文の第 8 章までの流れを追う形で進める．まず手始めに計算機械の定義(2.1)と例(2.2)を示し，計算機械を算術化する記述数(2.3)を解説する．そこから各集合の濃度の関係を整理し(2.4)，ごく簡単に万能機械を導入する(2.5)．そして最後に計算不能な機械の証明を行う(2.6)．

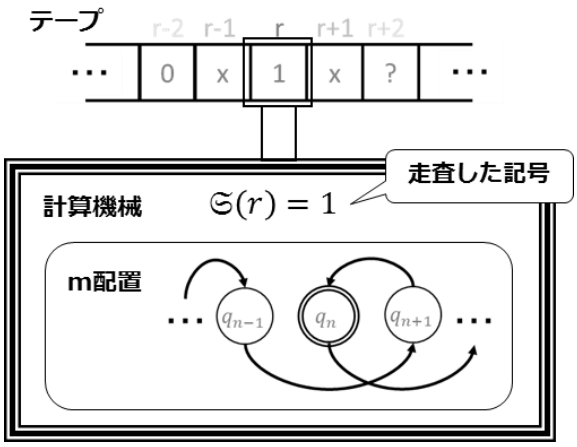
2.1. 計算機械の定義

論文のタイトルにもなっている「計算可能数(computable number)」とは，小数表現の各桁を有限の手続き(finite means)によって，機械的に計算する(書き出す)ことができる数を意味している．例えば円周率πは, 種々のよく知られた数値積分という有限の手続きによって任意の桁まで書き出すことができ，計算可能数の 1 つである．一方，仮想的な例ではあるが，無限に並ぶ数のリストを各桁として書き出す他に表現のない数が存在するならば，その手続は無限のリストを含むため，有限

的ではなく，計算可能数でない.

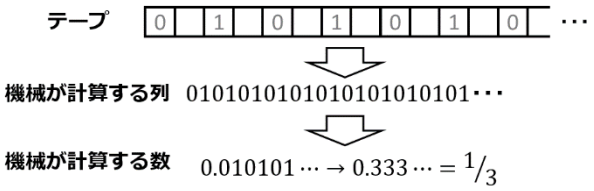
計算可能数の性質を具体的(具体的)に調べるため，チューリングは，人が紙とペンを用いて，実際に大きな桁の数の筆算を，限られた手続きに従ってステップ・バイ・ステップに計算するような状況になぞられて計算機械を定義した.

この機械にとっての計算用紙は，何らかの記号を 1 つずつ書けるマス目で区切られた「テープ」であり，また，計算を行う人(の心的状態)は，機械が取り得る「m配置(m-configuration)」と呼ばれる有限の状態$q_1, q_2, \cdots, q_r$に対応付けられる．次にテープ上の r 番目のマス目の記号を走査する(読み取る)ことを考えて，これを記号$\mathfrak{S}(r)$と表す．この機械の各時点での可能な動きはm配置q_nと走査した記号$\mathfrak{S}(r)$により定まり，これの対を「配置(configuration)」と呼ぶ．テープ上の r 番号目に紐付いて，記号の走査あるいは印字を行う機構をヘッドと呼ぶ(チューリングは，ヘッドという用語を論文の中で使っていないが，テープレコーダの比喩は十二分に利便なので，本エッセイではこの用語を使う)．［ここでの術語は正直わかり難い；後に示される計算機械の具体例と対比すれば，より容易に理解できると思われる．］



チューリングは，バッチ処理的に事前に定めた動作のみを行う機械を「自動機械(automatic machine，または a 機械)」，ユーザと対話的に動作するような，事前にすべての処理が定まらない機械を「選択機械(choice machine, または c 機械)」と呼んで区別していた．今後，計算機械と言った場合には，0 か 1 の記号(第 1 種の記号)と，その他の記号(第 2 種の記号)の 2 種類の記号をテープへ印字する a 機械を指すこととする.

計算機械が空白のテープから動作を始めて印字した第 1 種の記号を，特に*機械が計算する列*と呼び，その列の前に小数点を付けて 2 進数と解釈した実数を*機械が計算する数*と呼ぶ.



動作している計算機械のある時点での完全なスナップショットは，走査したマス番号の履歴とテープ上に印字された記号，m配置によって定まり，これら 3 つをまとめて「完全配置(complete configuration)」と呼ぶ．［ここでのスナップショットとは，現代における仮想マシンのスナップショットと等価なものである．］

第 1 種の記号を有限個しか印字をしない計算機械は「循環的(circular)」，無限に印字を行う計算機械は「非循環的(circle-free)」であると言う．［計算機械の動作の有限性に言及することから容易に想像できるように，後の証明において，この非/循環的という性質は重要な役割を果たす．］

特に非循環機械が計算できる列を「計算可能列(computable sequence)」と呼ぶ．また，既に解説をした計算可能数は，計算可能列から得られるものに限定する(ここは簡潔さを優先している；チューリングの定義的にここは厳密ではない).

2.2. 計算機械の例

ここから 2.1 で定義した計算機械の具体例と，

その動作を見てみる．次に記述するのは，チューリングの論文の中で最初に登場する計算機械であり，小数表現として 1/3 を計算する．

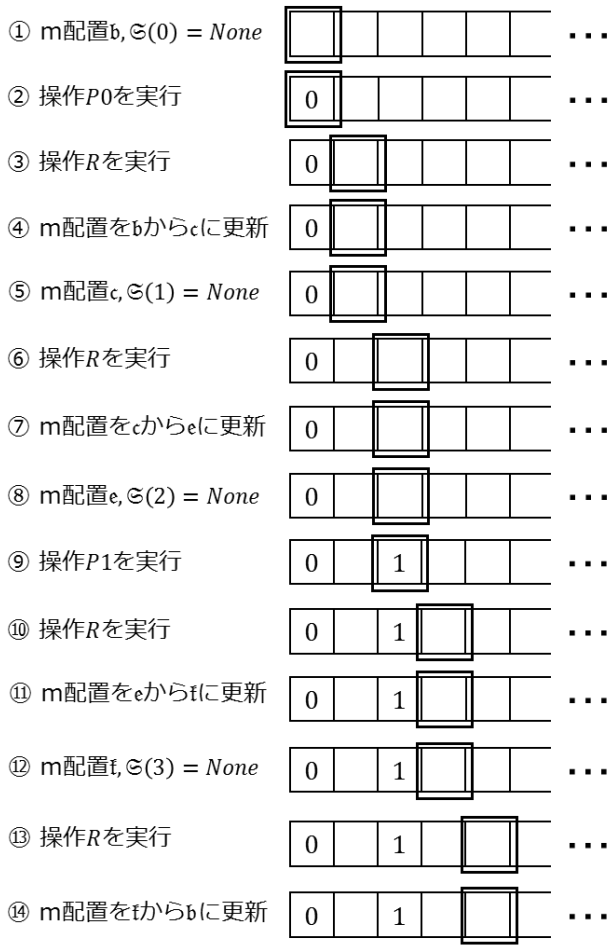
配置		動作	
m配置	記号	操作	最終m配置
𝔟	*None*	*P0,R*	𝔠
𝔠	*None*	*R*	𝔢
𝔢	*None*	*P1,R*	𝔣
𝔣	*None*	*R*	𝔟

この機械は 4 つのm配置"𝔟","𝔠","𝔢","𝔣"を持ち，動作として 0 か 1 を逐次的にテープ上に印字する．*None* は走査したマス目が空白であることを表し，*P0* と *P1* は各 0/1 をテープ上の現在のヘッド位置に印字する（一般に *Px* で記号 x を印字する）．*R* は現在のヘッド位置を右へ 1 マス移動することを表し，（この計算機械には利用されないが）*L* で左への移動を表す．

繰り返される各ステップにおける計算機械の基本動作は，まず，現状のm配置と，ヘッド位置を走査した記号が一致する配置を表から特定し，それに対応する動作における操作を実行，現状のm配置を最終m配置で更新する，という形になる．また，初期のm配置は𝔟と定まっている．

先の機械について，すべてのマスが空白のテープから始動して，m配置が一巡するまで計算を行った場合の各ステップにおける動作(左側)とテープ(右側)の遷移を次に示す. この機械の定義と合わせて順番に確認すれば，大枠としてチューリングの計算機械がどのように動作するのか概ねイメージできるようになると思う．

ここではm配置が一巡するまでを示したが，この計算機械はこれら 1～14 までの動作を無限に繰り返し，最終的に 0101010101…の*機械が計算する列*が得られる；これを*機械が計算する数*とすれば 1/3 が得られる．また，この機械は無限に動作し続けるので，定義から非循環的である．



本節では，２進数表現として 1/3 を計算する機械を例示したが，チューリングの論文では，この後，機械の定義方法として略記やマクロ的な表現などを導入し，より複雑な計算を行う機械が構成可能であることを示す．しかし，これ以上複雑な機械は本エッセイの後の節の説明に必須ではないため，計算機械の例示にこれ以上は深入りしない．

2.3. 計算機械の数え上げ

2.1 で定義した計算機械は，2.2 の例示から容易に想像されるように，適切な配置と動作の表を定義することで無数に得られる．この節では，このような無数に存在し得る計算機械を数え上げることを目標とする．最終的な結果を先取りしてしまうと，構成可能な計算機械の集合$\mathcal{M}$は可算集合である．

数え上げの方針としては，任意の計算機械をユニークに自然数へ対応付けることを考える．具体的な手順としては，まず，計算機械の定義を標準形式という分類に落とし込み，それを「標準記述(standard description / S.D)」という限られた文字のみを用いた列へ変換，標準記述に含まれる文字に数字を割り当て，計算機械とユニークに対応づく自然数である「記述数(description number / D.N)」を得る；記述数によって$\mathcal{M}$から自然数の集合$\mathbb{N}$への単射が得られ，$\mathcal{M}$が高々可算集合であることがわかる．

先に記した方針に従い，まず計算機械を標準形式の分類に落とし込む．標準形式においてm配置は番号付けされた$q_1,\cdots,q_n$(初期m配置はq_1と定める)の形で表し，機械が扱う記号も番号付けされた$S_1,\cdots,S_m$(特に *None*, 0, 1 をそれぞれS_0,S_1,S_2と定める)で表す．この表現を導入することで計算機械を定義する表の各行は次のいずれかに分類，変形できる．

m配置	記号	操作	最終m配置	
q_i	S_j	PS_k,L	q_m	(N_1)
q_i	S_j	PS_k,R	q_m	(N_2)
q_i	S_j	PS_k	q_m	(N_3)

次に(N_1)を$q_iS_jS_kLq_m$，(N_2)を$q_iS_jS_kRq_m$，(N_3)を$q_iS_jS_kq_m$の列として扱うことを考える(Pは必ず行われる動作として省略)．任意の機械はこれら列を";"で繋いで記述できることに注意する．ここでq_iを Dのあとに i個の *"A"*が連なった列，S_jを *"D"*のあとに j個の Cが連なった列と考えると，各形式の列は *"A"*, *"C"*, *"D"*, *"L"*, *"R"*, *"N"*, *";"*という有限の文字で記述でき，これを標準記述とする．

方針の最後として，*"A"*を 1，*"C"*を 2，*"D"*を 3，*"L"*を 4，*"R"*を 5，*"N"*を 6，";"を 7 という数字を割り当てれば，標準記述から直接的に 1 つの数が得られ，これが記述数になる．

2.2 で例示した計算機械を用いて, 標準形式と標準記述，記述数の具体例を示す．

まず，各m配置"ƀ", "ƈ", "ɐ", "ƭ"をq_1,q_2,q_3,q_4に，*None*, 0, 1 をS_0,S_1,S_2に置き直す．さらに，ここで右に移動するだけの操作がPS_0,Rで書けることに注意すると, 2.2 の機械は標準形式として次のようになる．

配置		動作	
m配置	記号	操作	最終m配置
q_1	S_0	PS_1,R	q_2
q_2	S_0	PS_0,R	q_3
q_3	S_0	PS_2,R	q_4
q_4	S_0	PS_0,R	q_1

ここから各行でつないだ列を構成し，各記号を所定の文字 *"A"*, *"C"*, *"D"*, *"L"*, *"R"*, *"N"*, *";"*に置き直すことで標準記述が得られる．

$$q_1S_0S_1Rq_2;q_2S_0S_0Rq_3;q_3S_0S_2Rq_4;q_4S_0S_0Rq_1;$$

$$\Rightarrow \textit{DADDCRDAA;DAADDRDAAA;}$$
$$\textit{DAAADDCCRDAAAA;DAAAADDRDA;}$$

最後に，標準記述の文字を割り当てられた数字に置き直して，次の記述数が得られる．

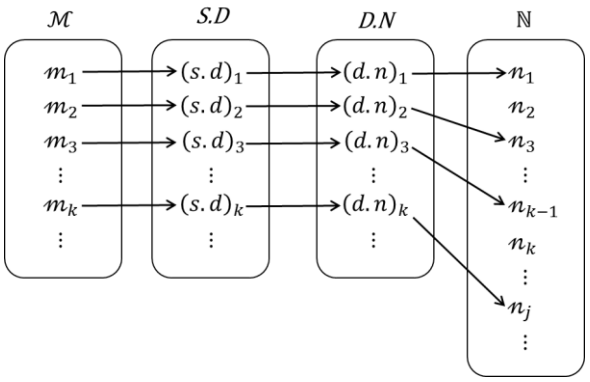
3133253117311335311173111332253111173111133531

記述数は，その構成法から明らかに任意の計算機械をユニークな自然数に対応付け，数え上げている．このことから，計算機械の集合$\mathcal{M}$から自然数の集合$\mathbb{N}$へ単射が存在し，$\mathcal{M}$が高々可算であることがわかる．また，$\mathcal{M}$は無限集合であるから，それは部分集合として可算集合を必ず含み，濃度として$\aleph_0 \leq card\,\mathcal{M} \leq card\,\mathbb{N} = \aleph_0$, $\mathcal{M}$は可算集合である．

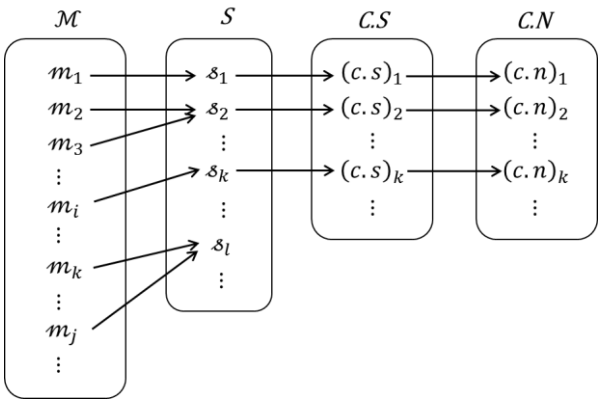
2.4. 各集合の濃度

2.3 では計算機械の集合$\mathcal{M}$が可算集合であることを示したが，この節では簡単に，ここまでに定義してきた各事項の集合の関係を整理する．

少し繰り返しになるが, 2.3 では標準記述と記述数を定義した；これらの集合には，構成法から明らかに$\mathcal{M}$からの全単射が存在し，記述数の集合は自然数の集合へ単射を持つので，最終的に，それぞれの集合は可算になる． 次にこれら関係を図示する．



次に, 2.1 で言及した計算機械の動作に対応する*機械が計算する列*の集合Sを考える；*機械が計算する列*は文字通り”機械”から生成されるので，$\mathcal{M}$からSへは全射が存在して$card\ S \leq card\ \mathcal{M}$であり，$S$は可算である(異なる複数の機械が，同一の列を書き出すことがあることに注意する)．また，定義から明らかなように, 計算可能列の集合$C.S$と計算可能数の集合$C.N$は，Sの部分集合と 1 対 1 対応を持ち，これらも可算である． 次にこれらの関係を図示する．



最終的に，ここまでに定義してきた各事項の集合の濃度は，2.3 の結論から派生して，すべて可算であることがわかる．

2.5. 万能機械

2.2 の例で示した計算機械は, 単一の機能のみを持つものであったが, チューリングは論文の中で, ある機械mの標準記述が記載されたテープを受け取り，その機械mの動作を完全に再現する機械$\mathcal{U}$が存在することを構成的に示し，これを「万能計算機械（universal computing machine）」と呼んだ（以下簡潔のため万能機械と呼ぶ）．

チューリングの論文の中でも，この万能機械を実際に構成していく部分は，最も独創的で重要な内容の 1 つではあるが，これの解説を始めると，本エッセイのボリュームが爆増してしまうため，構成に関しては，これ以上深入りはしない．

チューリングの功績として, 万能機械によって, 物理的に 1 つの機械を構築すれば，あとはテープ上の記載を変更するだけで多数の機能が実現できる，現代で言えばハードウェアとソフトウェアの可能性を”具体的”に示したことは，しばしば注目されるところである．しかし，論文中において，万能機械は，計算不能な機械の証明で”特に”重要な役割を占めていたことは述べておいて良いだろう；第 1 章で多用したように，現代のプログラミングにおいて，関数自体を通常のオブジェクトとして扱う高階関数という技法はごく当たり前のものであるが，チューリングが定義した計算機械においては，標準記述と万能機械の導入によって初めて実現され得るものであり，それらの導入は必然的な重要性を持っているのである．[万能機械は, チューリングに先立ち，不完全性定理によって形式的数学の限界の一端を示したゲーデルの仕事の流れから，自己言及性を導入するためのある種必然的な道具であったと言えるかもしれない．ゲーデルが導入したゲーデル数や証明可能性述語は,

チューリングの標準記述と万能機械に概ね符合する．]

2.6. 計算不能な機械

ここまでの節で道具仕立てが出揃ったので，計算不能な機械の証明に入る．

まず，一見健全な対角線論法に見えるが，実際はそうではない議論を行い，そこでの”見落とし”として計算可能列の集合の興味深い性質を見る．

既に検証された命題に対角線論法を適用する；計算可能列の集合$C.S$が可算であると仮定する．α_nをn番目の計算可能列，$\phi_n(m)$をα_nにおける m 番目の数字とすると，すべての計算可能列は次のようにリストとして列挙できる．

$$\alpha_1 = \phi_1(1)\phi_1(2)\phi_1(3)\phi_1(4)\cdots$$
$$\alpha_2 = \phi_2(1)\phi_2(2)\phi_2(3)\phi_2(4)\cdots$$
$$\alpha_3 = \phi_3(1)\phi_3(2)\phi_3(3)\phi_3(4)\cdots$$
$$\cdots$$

次にβを，$1-\phi_n(n)$を n 番目の数字として持つ計算可能列とする；$\phi_n(m)$は第 1 種の記号(0/1)しか取らないため，これは先のリストの対角線上の値をビット反転した列を表している．

$$\beta = \big(1-\phi_1(1)\big)\big(1-\phi_2(2)\big)\big(1-\phi_3(3)\big)\cdots$$

βは計算可能列なので，先のリストに含まれているはずである．すなわち，あるKが存在して，βは列の各数字が$1-\phi_n(n)=\phi_K(n)$となるα_Kである．

$$\beta = \alpha_K = \phi_K(1)\phi_K(2)\phi_K(3)\phi_K(4)\cdots$$

ここで，列の各数字$1-\phi_n(n)=\phi_K(n)$を，$n=K$として変形すると$1=2\phi_K(K)$となり，1 が偶数であるという矛盾した結果が得られる．これは$C.S$が可算であるという仮定から導かれた．故に仮定は誤りであり，$C.S$は可算でない(!?)．

2.4 において既に示したように計算可能列の集合は可算である．明らかに何かがおかしい．

直感的には，βはリスト上の各計算可能列に対して，容易に実現可能な減算を適用し，新たな列を生成しているだけであり，一見したところ計算不能な要素はないように思える；しかしながら，ここでの議論の誤りは，やはりβが計算可能であると考えたところにある．

根本的に計算可能列とは，非循環機械が計算できる列であったことを思い出そう．βを構成するためには，計算可能列のリストを列挙する必要があり，それは循環機械と非循環機械を見分けるのと同等の問題を暗に内包している．第 1 章を読んだ人は既に気がついているだろうが，循環機械と非循環機械を見分けることは停止性問題と同義であり，これは不可能である；先の議論のプロセスは，はじめの段階から見当違いだったのである．

ここでの議論から計算可能列/数の集合について，1 つの興味深い性質が見られる：計算可能列/数は可算集合であるにもかかわらず，一般に列挙できない．

チューリングの論文において，この段階では，証明なしに循環機械と非循環機械を見分ける一般的な方法はないことだけを言い，次に正しく対角線論法を用いることで，そのような計算不能な機械を証明できると述べる．

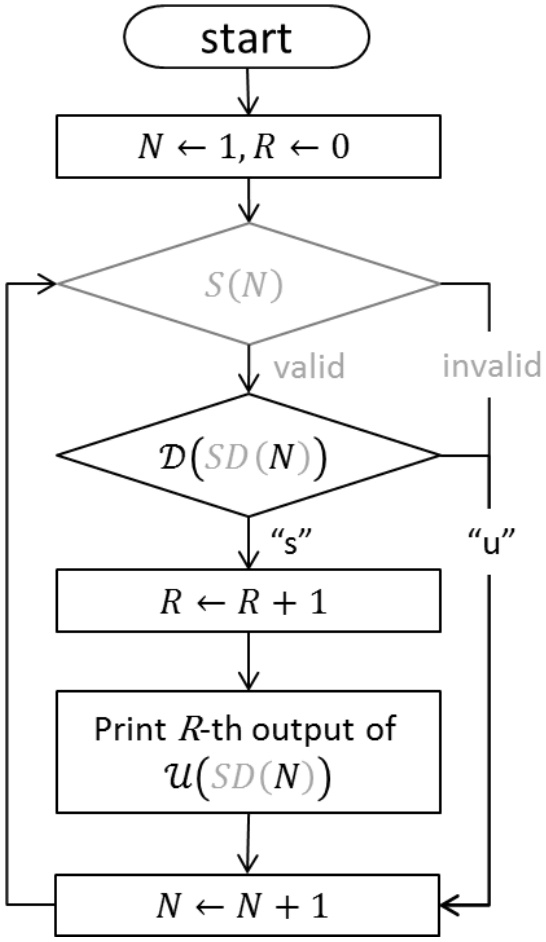
ここからが本エッセイの最後の議論である．

n 桁目が数$\phi_n(n)$であるような，βをより単純にした列β'を書き出す機械を構想する．

$$\beta' = \phi_1(1)\phi_2(2)\phi_3(3)\phi_4(4)\cdots$$

まず，任意の機械mの標準記述を受け取り，これを有限回のステップで検証，機械mが循環的ならば”u”，非循環的ならば”s”を書き出す(判定する)ような機械$\mathcal{D}$の存在を仮定する．この機械$\mathcal{D}$と万能機械$\mathcal{U}$を組み合わせることにより，列β'を計算する機械$\mathcal{H}$が構成できる．

チューリングは論述的に機械$\mathcal{H}$の動作を説明したが，幾分か見通しを良くするため，補助的な機械を伴い，各種機械を関数的に表現した場合の動作をフローチャートで次に示す．[ここで，関数への引数の受け渡しや，変数のインクリメント等は，実際にはすべてテープを介して行われることに注意する．]



補助的な機械として，機械SはNが適正な機械の記述数であるかどうかを判定し，機械SDは記述数Nを標準記述に変換する．[これら補助的な機械はチューリングの説明には登場しない．]

機械$\mathcal{H}$は，1から順番に自然数Nを生成し，機械Sの検証を経た後，それを潜在的な記述数として機械$\mathcal{D}$に渡す．その判定結果が"s"であればRを1インクリメント，機械$\mathcal{U}$で$SD(N)$の動作をR桁まで再現する；そして再現結果の最後の桁のみを，機械$\mathcal{H}$のR桁目の出力としてテープへ書き出す．

Nに対応するRを特に$R(N)$として，機械$\mathcal{H}$の各変数と動作を次に示す．

N	$R(N)$	$\mathcal{D}(N)$	β'
1	*0*		
2	*0*		
⋮	⋮	⋮	⋮
i	*1*	*"s"*	$\phi_1(1)$
⋮	⋮	⋮	⋮
j	*2*	*"s"*	$\phi_2(2)$
⋮	⋮	⋮	⋮
l	$R(l)$	*"s"*	$\phi_{R(l)}(R(l))$
⋮	⋮	⋮	⋮
m	$R(m)$	*"u"*	
⋮	⋮	⋮	⋮

機械$\mathcal{H}$は構成の方法から非循環的である．また，機械$\mathcal{D}$の仮定と機械$\mathcal{U}$の扱いから各$\phi_n(n)$の計算は有限回のステップで完了する．また，機械$\mathcal{H}$は仮定から構成可能な機械であるから，対応する記述数が存在するはずである；今これをKとしよう．ここから，1から順番に自然数Nについての各$\phi_n(n)$を計算していき，機械$\mathcal{H}$がKに到達したときに何が起こるか考える．

機械$\mathcal{H}$は非循環的であるから，記述数Kにおいて機械$\mathcal{D}$は"u"を返すことはない；しかし，一方，機械$\mathcal{D}$が"s"を返すと考えることも不適切である．このことを，Kにおける機械$\mathcal{H}$の動作を追うことで具体的に見てみる．

まず，機械$\mathcal{H}$はKを生成し，それについて機械$\mathcal{D}$は"s"を返すと考える．"s"が返されたので，次に機械$\mathcal{U}$がK(機械$\mathcal{H}$)の再現を開始する；少なくともK以前に対応するR桁目までは再現できるはずである．

しかしながら，またKに対応するR桁目まで到達したら，今再現されている機械$\mathcal{H}$の中で，再び

機械$\mathcal{U}$が K(機械$\mathcal{H}$) の再現を開始する．以下この再現の再現が際限なく行われる．このため機械$\mathcal{H}$が Kよりも大きい自然数に対応する R桁目を書き出すことはありえない．すなわち，機械$\mathcal{H}$は循環的であり，機械$\mathcal{D}$の判定は不適切である．

最終的にどちらの判定もありえないため，この結果を導いた前提は誤りであり，機械$\mathcal{D}$は存在せず，計算不能である．

3. 終わりに

このエッセイでは，前半において多少広がりのある停止性問題の全般的な解説を行い，後半でチューリングの論文を追いながら，最終的に計算不能な機械の証明を行った．

焦点を絞り，出来る限りわかりやすい説明を試みたつもりだが，著者の明らかな力量不足から理解に苦しむ点も全体を通して散在しているところと思う．そのような点にぶつかった読者には，大変申し訳ないと思う．

最後なので少し個人的な話をしよう．私がチューリング機械や停止性問題という言葉を初めて聞いてから既に十数年が経過しているかと思うが，実際に，それらがどのようなものであるのかということを具体的に知るようになったのは，ここ数年であった．

計算モデルや計算可能性の限界という何とも興味をそそられる題材に長く憧れは抱いていたが，計算理論という，なにやら難しそうな言葉に恐れをなして必要以上に近づこうとはしていなかった．

これが変わったのは参考文献[4]との出会いであった．計算理論がプログラミング言語の直接的な延長線上で議論され得ることは，自分にとってはとても新鮮であった．[後知恵ではあるが，チューリング機械やラムダ計算でなく，実在のプログラミング言語や，現実的な擬似言語を計算の定義として導入している書籍は多数存在しているようである．] ここでの観点は，第 1 章において幾らか取り入れられている．願わくは，本エッセイによって，自分が感じたのと類似の新鮮さが多少でも読者に伝わっていればと思う．

第 2 章を読んでチューリングの実際の仕事に興味が湧いた読者は，ぜひ参考文献[2]を読んでもらいたい．時間はかかると思うが，素晴らしいガイドと共に，このエッセイでは伝えきれなかったチューリングの独創性を十二分に感じてもらえることと思う．

4. 参考文献

[1] Turing, A. M. (1936). On Computable Numbers, With an Application to the Entscheidungsproblem. Proceedings of the London Mathematical Society, 42. 230-265.

[2] チャールズ・ペゾルド (著), 井田哲雄, 鈴木大郎, 奥居哲, 浜名誠, 山田俊行 (訳).(2012). チューリングを読む – コンピュータサイエンスの金字塔を楽しもう.日経 BP 社.

[3] 松坂和夫. (1968). 集合・位相入門. 岩波書店.

[4] 鹿島亮. (2008). C 言語による計算の理論. サイエンス社.