学位論文要旨 (修士 (理学))

論文著者名：伴　滉一郎

論文題名：ロールバック付き優先論法による 2-c.e. 集合の分解

定理 (Guohua Wu(2006))**.** $\mathbf{a}$ を計算不可能かつ帰納的可算次数とする．このとき以下を満たすような 1-ジェネリック次数 $\mathbf{a_0}, \mathbf{a_1}$ が存在する．

$$\mathbf{a} = \mathbf{a_0} \cup \mathbf{a_1}$$

Wu はこの定理を**有限回しか傷の付かない優先論法 (finite injury priority argument)** によって証明している．

本論文では, その巧みな優先論法を解説し, 更にその優先論法を拡張することで以下の定理を証明している．この拡張は水澤勇気および鈴木登志雄との共同研究に基づく．

定理. 2-c.e. 次数 $\mathbf{a}$ は以下のように 1-ジェネリック次数 $\mathbf{g_0}, \mathbf{g_1}, \mathbf{g_2}, \mathbf{g_3}$ に分解される．

$$\mathbf{a} = \mathbf{g_0} \cup \mathbf{g_1} \cup \mathbf{g_2} \cup \mathbf{g_3}.$$

$\mathbf{a}$ が帰納的可算次数から 2-c.e. 次数になることで, Wu の優先論法はそのままでは機能しない．Wu の優先論法は, 分解する集合が有限集合の拡大列という性質に依存しているからである．2-c.e. 次数では拡大列になるとは限らない．すなわちある元が有限集合から抜けてしまう縮小が起こり得る．

この問題を解決するために我々が研究し考案した優先論法が, **ロールバック付き優先論法**である．この手法では, 有限集合が拡大し続ける間は Wu の優先論法をシミュレートする．しかし一度でも縮小をした場合, その縮小を原因から取り除くために, 構成をロールバックする．縮小を引き起こした元にはマークを付け, ロールバックした後には縮小を引き起こすような元を判別できるようにする．

ロールバック付き優先論法による 2-c.e. 集合の分解

伴 滉一郎
首都大学東京大学院 理工学研究科 数理情報科学専攻

2017 年 1 月 6 日

概要

Wu [9] では, 優先論法により c.e. 次数を 2 つの 1-ジェネリック次数に分解している. 本論文ではその結果を拡張し, 2-c.e. 次数を 4 つの 1-ジェネリック次数に分解している. そのために考案したのがロールバック付の優先論法である. これは Wu [9] で用いられている優先論法の戦略に加え, 構成に失敗した際に失敗の原因が生まれたステージまでロールバックをして原因を取り除きながら構成を行うというものである.

目次

1 イントロダクション

計算理論では計算モデルを用いて, ある問題が計算可能かどうか, そうでなければどれぐらい複雑なものなのか, ということを研究する. 計算モデルの中でも代表的なものはチューリング機械だろう.

計算理論において, ある集合が計算可能であるとは, 与えられた数が集合に属しているか否かを正しく判定するようなチューリング機械が存在することである. そうでない集合, つまり計算不可能な集合の代表的なものとして, 停止性問題がある. 停止性問題とは, 与えられたチューリング機械 M と入力 x について, その M に x を入力した際に M が停止するかどうかを判定する問題である. そしてこの停止性問題を解くようなチューリング機械は存在しない.

このような計算不可能な集合たちを整理するために, チューリング次数 (Turing degree) が定義される. チューリング次数というのは, チューリング計算可能性によって計算不可能集合の相対的複雑さを測るものである. 具体的には, 計算したい集合とチューリング機械以外に, 外部データベースのような集合 X を用意する. そして X を利用して計算可能な集合は, X よりもチューリング次数では下に位置すると定義する. こうして整理した計

算不可能なものの中でも, 計算可能に近いものとして帰納的可算集合がある. また, 計算可能性理論の発展に大きく寄与した問題の一つに, エミール・ポストの問題がある. これは, 停止性問題よりもチューリング次数の低い計算不可能な帰納的可算集合が存在するか, というものである. そしてこの問題を解決する過程で編み出された手法が優先論法 (priority argument) である. Friedberg と Muchnik がそれぞれ編み出したこの優先論法は, 今日においても帰納的可算集合を研究する手法として代表的なものである.

また集合論において非常に有名な証明として, コーエンによる一般連続体仮説の独立性の証明がある. この証明には強制法 (forcing) という手法, ジェネリック (generic) という集合が登場する. 計算理論でもこれらと似たようなことが考えられており, その中でも重要な 1-ジェネリック集合というものがある.

本論文は, 水澤勇気および鈴木登志雄との共同研究に基づき, Wu[9] が考案した巧みな優先論法について詳細に解説した後, その優先論法を更に発展させ帰納的可算集合よりも複雑な集合を 1-ジェネリック集合に分解する手法について述べる.

2 計算可能性理論

このセクションでは, 本論文を読むにあたって必要な計算可能性理論の基本的な概念をまとめている. 特に記載が無い場合, 全て Cooper[2] を参考にしている.

2.1 再帰関数

定義 2.1. 1. 以下の 3 種類の**初期関数 (initial function)** は**原始再帰関数 (primitive recursive function)** である. ただし $\vec{m}$ は自然数の組を表わす.

（a）**零関数**

$$\mathbf{0}(n) := 0, \quad \forall n \in \mathbb{N}.$$

（b）**後者関数**

$$n' := n + 1, \quad \forall n \in \mathbb{N}.$$

（c）**射影関数**

$$U_i^k(\vec{m}) := m_i, \quad \forall k \geq 1, i = 1, \ldots, k.$$

2. $g, h, h_0, h_1, \ldots, h_l$ が原始再帰関数である場合, 以下の規則を適用して定義される f もまた原始再帰関数である.

（d）**代入**

$$f(\vec{m}) = g(h_0(\vec{m}), \ldots, h_l(\vec{m})).$$

（e）**原始再帰**

$$f(\vec{m}, 0) = g(\vec{m}),$$
$$f(\vec{m}, n+1) = h(\vec{m}, n, f(\vec{m}, n)).$$

原始再帰関数全体のクラスを PRIM と書く．PRIM には $+, -, \times, \div$ や絶対値計算, 符号関数など基本的な関数が属している．

定義 2.2. $f : A \to B$ について, $\forall x \in \mathbb{N}$ に対し $f(x)$ が定義されている (これを $f(x)\downarrow$ と書く) ときに, f は**全域関数 (total function)** であるという．またそうでないとき, すなわち $f(x)$ が定義されていない (これを $f(x)\uparrow$ と書く) ような $x \in \mathbb{N}$ が存在するときに, f は**部分関数 (partial function)** であるという．

定義 2.3. 以下のような略記法を導入する．

$$\mu m[g(\vec{n}, m) = 0] = m_0 \overset{\text{def}}{\iff} g(\vec{n}, m_0) = 0 \text{ and } (\forall m < m_0)[g(\vec{n}, m)\downarrow \neq 0].$$

この μ は $g = 0$ になる最小値 (もしあれば) を表わし, μ **作用素**とも呼ばれる．
関数が部分再帰であるとは以下のように定義される．

定義 2.4.
1. f が**部分再帰関数 (partial recursive function)** であるとは, f が定義 2.1 の零関数, 後者関数, 射影関数のいずれかである, またはそれらに定義 2.1 の代入, 原始再帰, 定義 2.3 の μ 作用素を有限回作用させたものであるときにいう．
2. 特に f が全域関数であるとき, f を**再帰関数 (recursive function)** という．

2.2 チューリング機械

チューリング機械は仮想機械だが, 現実の電子計算機の原理はチューリング機械の原理と同じである．そのため後述するチューリングの提唱では, チューリング機械で計算可能である集合は現実の電子計算機でも計算可能であるとしている．チューリング機械が電子計算機と大きくかけ離れている点は, 無限長のテープ (電子計算機でいうメモリ) を持っていることである．そのためチューリング機械で計算可能であるということは, 現実では気が遠くなるほど長い時間や, 膨大なメモリを必要とするプログラムである可能性がある．あくまで (メモリや時間の制約の無い) 理想的な状態であれば計算可能ということである．

まずは基本的なチューリング機械の構造について述べる．チューリング機械を構成するのは, 無限の長さを持ったマス目に区切られたテープ, その 1 マスを読み込むことの出来

るヘッド, ヘッドが接続されている本体, 本体内部の状態, そしてプログラムといった要素である. テープのマス目には 0 か 1 が 1 マスに 1 つ書きこまれており, ヘッドはその数字を読み取る. 本体は今の内部状態とヘッドが読み取った数字によって, 次の内部状態とヘッドが行うアクションを決定しそれを実行する. このような機械が動作する上での約束事の有限個の集まりをプログラム, またはチューリング機械そのものであるという.

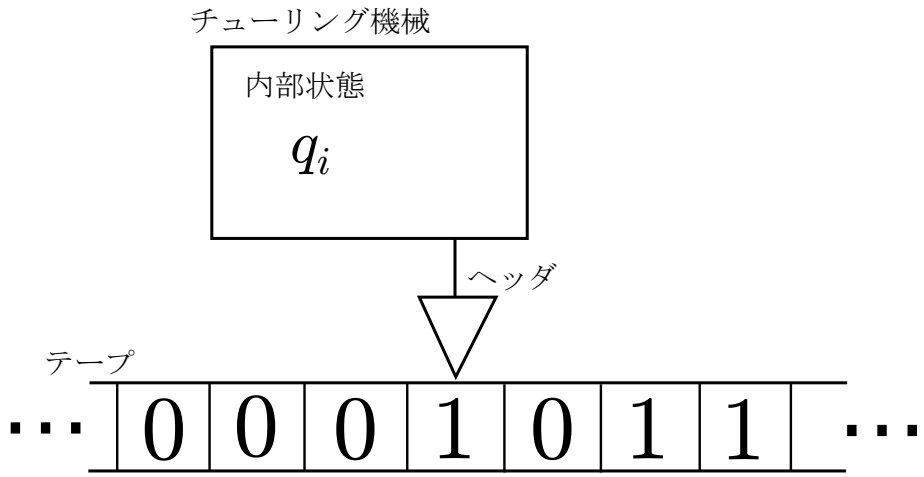

図 1　チューリング機械のイメージ

プログラムの 1 行は以下のようなクアドラプル (4 つ組) で表現される.

$$Q \to q_i S A q_j$$

S はテープシンボルといい, 読み取った 0 か 1 の数字を表わす記号である. A はアクションシンボルという. この部分が 0(resp. 1) であればヘッドが読み込んでいるマスに 0(resp. 1) に書き換える. L(resp. R) であればヘッドを 1 マス左 (resp. 右) に動かす. つまり, A はこの 4 種類のアクションのうち 1 つを表わすシンボルである. q は内部状態を表わしている. 例で言えば, 現在の内部状態が q_i であり, この状態でシンボル S を読み込んだ場合はアクション A を行う. その後移行する内部状態が q_j であり, このように約束 (プログラム) されていることをクアドラプルが表現している. クアドラプルの有限個の集まりがプログラムであり, その全てを参照しても実行すべき行動が無い場合にチューリング機械は停止する. プログラム実行開始の段階で $n+1$ 個の 1 が書きこまれているときに「n が入力された」と解釈し, 停止した際にテープに書きこまれている 1 の数を出力される値と解釈する.

定義 2.5. 1. クアドラプルの集合 X が**無矛盾**であるとは以下のようなときにいう.

$$q_iSAq_j, q_iSA'q_k \in X \Longrightarrow A = A', q_j = q_k.$$

2. 無矛盾なクアドラプルの有限集合を**チューリング機械** T(または T のプログラム)と呼ぶ.
3. f が**チューリング計算可能**であるとは, あるチューリング機械 T において任意の入力 x に対しての出力が $f(x)$ であるときにいう.

あるチューリング機械 T は 1 つの自然数で表現することが出来る. そのために**ゲーデル数**という概念を考える.

定義 2.6. まず, クアドラプルの各シンボルのゲーデル数 $\mathrm{gn}(x)$ を,

$$\begin{aligned}\mathrm{gn}(L) &= 2\\ \mathrm{gn}(R) &= 3\\ \mathrm{gn}(q_i) &= (2+2i \text{ 番目の素数})\\ \mathrm{gn}(S_i) &= (2+2i+1 \text{ 番目の素数})\end{aligned}$$

と定義する. 次に $Q \to q_iSAq_j$ というクアドラプル Q をゲーデル数で表現することを考える. これは現在の内部状態, テープシンボル, アクションシンボル, 移行する内部状態をそれぞれ異なる素数の肩に乗せて,

$$\mathrm{gn}(Q) = 2^{\mathrm{gn}(q_i)} \times 3^{\mathrm{gn}(S)} \times 5^{\mathrm{gn}(A)} \times 7^{\mathrm{gn}(q_j)}$$

といった形で Q を表現出来る.
更にチューリング機械 T のプログラムの列 $\{Q_0, Q_1, Q_2, \ldots, Q_k\}$ についても同様に,

$$\mathrm{gn}(T) = 2^{\mathrm{gn}(Q_0)} \times 3^{\mathrm{gn}(Q_1)} \times \cdots \times p_k^{\mathrm{gn}(Q_k)} \quad (\text{但し } p_k \text{は } k+1 \text{ 番目の素数})$$

とすれば 1 つのチューリング機械 T を 1 つの自然数で表現することが出来る.

1 つのゲーデル数に対応するチューリング機械が複数存在しないことは素因数分解の一意性より明らかである. このゲーデル数を使えば, いかなるチューリング機械も e **番目のチューリング機械** P_e というように自然数で番号付けすることが出来る.

2.3 チャーチ-チューリングの提唱

チャーチの提唱

f が実行的計算可能 $\Longleftrightarrow$ f は部分再帰関数

チューリングの提唱

f が実行的計算可能 $\Longleftrightarrow$ f はチューリング計算可能

これは定理ではなく提唱 (テーゼ) である. この提唱をもう少し詳しくいうと次の通りである. 「実効的に計算可能な関数」という哲学的な概念を, 今後は「部分再帰関数」という数学概念で置き換えてもよいと約束しよう. また, 「チューリング計算可能関数」という数学概念で置き換えてもよいと約束しよう.

また, 部分再帰関数全体のクラスとチューリング計算可能関数全体のクラスが等価であることは証明されており, その他の計算モデルとも等価であることが証明されている. 直観的には, ある関数が計算可能であるとは, 「C 言語でプログラミングが可能である」こととも換言できる. 以上を念頭に置いてようやく「計算可能」という言葉が定義できる.

定義 2.7. 集合 A が**計算可能 (computable)** であるとは, 任意の $n \in \mathbb{N}$ に対し, $n \in A$ または $n \notin A$ が実効的決定可能であることをいう.

この定義 2.7 と特性関数 χ_A の定義より,

$$\begin{aligned} A \text{ は計算可能} &\Longleftrightarrow \chi_A \text{が計算可能} \\ &\Longleftrightarrow \chi_A \text{は再帰関数} \end{aligned}$$

というように集合の計算可能性について換言することが出来る.

また, 計算可能集合より少しだけ計算不可能に近づいた集合として, 帰納的可算集合というものが存在する.

定義 2.8. 集合 $A \subseteq \mathbb{N}$ が**帰納的可算 (computably enumerable または c.e.)** であるとは, $A = \emptyset$ または $\mathrm{range}(f) = A$ なる計算可能関数 f が存在するときにいう.

A が帰納的可算集合であるとは, 言い換えれば A に属している n の読み上げプログラムが存在しているということである.

定義から当然であるが, 計算可能集合全体は帰納的可算集合全体に含まれる. 特に計算可能でない帰納的可算集合, つまり $n \notin A$ なる n については有限時間で答えが出せないような集合を, **真の帰納的可算集合 (proper c.e. set)** と呼ぶ. そして c.e. 集合よりも更に少しだけ計算不可能に近づいた集合が d.c.e. である.

定義 2.9. 集合 $D \subseteq \mathbb{N}$ が **d.c.e. (difference of c.e. sets)** であるとは $D = A - B$ なる c.e. 集合 A,B が存在するときにいう.

2.4 ビット列

前セクションでは抽象的に「計算不可能に近づく」などと述べたが, このセクションではそれを具体的に表現することを考える. そのためには集合のビット列による表現についていくつか定義する必要がある.

定義 2.10. 1. 0 と 1 の有限個の連なりを**有限列 (finite string)** という. たとえば 01101010 は有限列である.

2. 有限列 σ が集合 A の**始切片 (initial segment)** であるとは, $n \in \mathbb{N}$ について $\sigma = \chi_A(0)\chi_A(1)\chi_A(2)\dots\chi_A(n)$ となるときにいい, この $n(= |\sigma|)$ を σ の長さという.
   例えば $A = \{2, 3, 5\}$ のとき, 長さ 3 の A の始切片は 001 であり, 長さ 10 の A の始切片は 0011010000 である.
3. σ と τ の**結合 (concatenation)** を $\sigma^\frown\tau$ と書き, σ の後に τ を連ねたものを表わす. 例えば, $\sigma = 0111, \tau = 01$ ならば $\sigma^\frown\tau = 011101$ である.
4. τ は σ の**拡張 (extention)** であるとは, 任意の $x < |\sigma|$ について $\sigma(x)\downarrow \Rightarrow \tau(x)\downarrow = \sigma(x)$ であるときにいい, $\sigma \subseteq \tau$ と書く. 例えば $01 \subseteq 0111$ である.
5. ひとつの集合は**無限列 (infinite string)** で表現される. 例えば $A = \{2, 3, 5\}$ のとき, 0011010000000... である.

定義 2.11. c.e. 集合 A について, $\lim_{s\to\infty} A_s = A$ なる $\{A_s\}_{s\geq 0}$ を A の**近似列 (approximation sequence)** という. 但し任意の $s \in \mathbb{N}$ に対し A_s は有限集合である.

性質 2.12. 定義 2.8 と定義 2.11 より, 集合 A が c.e. であるとき,A の近似列 $\{A_s\}_{s\geq 0}$ は (集合の包含関係について) 単調増加列としてとれる.

Proof. $A = \mathrm{range}(f) = \{f(0), f(1), f(2), \dots\}$ であるので,

$$A_s := \{f(0), f(1), \dots, f(s)\}.$$

とすれば明らか. □

定義 2.11 をビット列で表現することを考える. まず, 有限集合 A_s を表わすような有限個の 1 しか持たないビット列がある. そのビット列が添え字 s を増やす毎に A を表わす無限ビット列に形が近づいていく.

更に以下では性質 2.12 もビット列で表現することを考える. そのために 1 ビットの変化についていくつかの言葉を定義する.

定義 2.13. 集合 A に収束する近似列 $\{A_s\}_{s\geq 0}$ において, $A_s(x) = 0,\ A_{s+1}(x) = 1$ であるときに, x が $s+1$ で A に**エニュメレイトされた**という.

いま, 逆に $A_s(x) = 1,\ A_{s+1}(x) = 0$ であるときに, x は $s+1$ で A から**ドロップアウトしたということにする**. また, 以降これらをまとめて, **マインドチェンジ (mind change)** と呼ぶ.

例えば $A_s = \{2, 3, 5\}$ を表現する 001101 というビット列がある. このとき $A_{s+1} = \{2, 3, 4, 5\}$ であれば, 4 は $s+1$ で A にエニュメレイトされたといい, A_{s+1} は 001111 と表現される. また $A_{s+2} = \{2, 4, 5\}$ であれば, 3 は $s+2$ で A からドロップアウトしたといい, A_{s+2} は 001011 と表現される.

性質 2.14. 性質 2.12 より,c.e. 集合のマインドチェンジは各ビット高々 1 回である. このとき単に c.e. 集合 A のマインドチェンジは高々 1 回であるという.

つまり, c.e. 集合に収束する近似列を表現するビット列は, $0 \to 1$ という変化しかしないようなものとしてとれる

性質 2.15. 定義 2.9 と性質 2.14 より, 集合 D が d.c.e. であるとき,D のマインドチェンジは高々 2 回である.

Proof. $D = A - B$ なる c.e. 集合 A,B は性質 2.14 よりマインドチェンジは高々 1 回である. このとき $D_s := A_s - B_s$ と定義し,D_s を表現する無限列の 1 ビットに着目すると, 考えられるマインドチェンジのパターンは以下で全てである.

1. A のみがマインドチェンジをする場合. D のマインドチェンジは 1 回である.

2. B のみがマインドチェンジをする場合. D のマインドチェンジは 0 回である.
3. $s \geq t$ で A が A_s でマインドチェンジをし, B は B_t でマインドチェンジをする場合. D のマインドチェンジは 0 回である.
4. $t > s$ で A が A_s でマインドチェンジをし, B は B_t でマインドチェンジをする場合. D のマインドチェンジは 2 回である.

A が先にマインドチェンジをし, その後 B もマインドチェンジするパターン 4 のみ, 着目しているビットは $0 \to 1 \to 0$ と変化するので D のマインドチェンジは 2 回になる. □

定義 2.16. ある集合 A のマインドチェンジの回数を n とする. このとき A は **n-c.e 集合**であるという. 特に性質 2.15 より以下が成り立つ.

$$A \text{ は 2-c.e. 集合} \iff A \text{ は d.c.e..}$$

2.5 算術的階層

計算可能集合や帰納的可算集合など, 今まで定義してきた集合を分類している**算術的階層 (arithmetical hierarchy)** という概念が存在する. これは集合を表現する論理式の複雑さによって分類をおこなうものである. この分類の中で最下層, すなわち最も単純な集合が計算可能集合である. 以下では計算可能集合を元に, 複雑な階層について再帰的に定義している.

定義 2.17. 1. $\Sigma^0_0 = \Pi^0_0 = \Delta^0_0 =$ 計算可能な関係全体の集まり.

以下 $n \geq 0$ について,

2. $\Sigma^0_{n+1} = (\exists \overrightarrow{y}) R(\overrightarrow{x}, \overrightarrow{y})$ という形の関係全て, 但し $R \in \Pi^0_n$.
3. $\Pi^0_{n+1} = (\forall \overrightarrow{y}) R(\overrightarrow{x}, \overrightarrow{y})$ という形の関係全て, 但し $R \in \Sigma^0_n$.
4. $\Delta^0_{n+1} = \Sigma^0_{n+1} \cap \Pi^0_{n+1}$.

R が**算術的 (arithmetical)** とは $R \in \bigcup_{n \geq 0} (\Sigma^0_n \cup \Pi^0_n)$ なるときにいう.

定理 2.18. 以下は同値である.

1. 集合 A は c.e..
2. $A \in \Sigma^0_1$.
3. ある e に対して $A = W_e$ (ここで W_e は e 番目の c.e. 集合を表す)

e はプログラムのソースコードのゲーデル数 (定義 2.6), W_e は e という自然数が表現す

るプログラムに入力した際にプログラムが有限時間で停止するような入力全体の集合である. 重要なことは, c.e. 集合は可算無限個しか存在しないということである.

2.6 チューリング還元とチューリング次数

イントロダクションで述べたように, チューリング計算可能性によって集合の相対的複雑さを測る概念がチューリング次数である. チューリング次数の定義のためにはまずチューリング還元について定義する必要がある.

定義 2.19. 集合 B が集合 A をオラクルとしてチューリング計算可能であるとき, B は A に**チューリング還元可能 (Turing reducible)** と言い, $B \leq_T A$ と書く.

オラクルは外部データベースに例えられ, ある数が外部データベースに入っているか否かは 1 ステップで決定可能である. つまり $B \leq_T A$ であるとは, 「A に n という数は属していますか？」という形の問い合わせを許可された (そしてその命令に対し 1 ステップで Yes, または No の答えが返ってくる) チューリング機械で B が計算可能であるようなときにいう.

定義 2.20. 1. A と B が**チューリング等価 (Turing equivalent)** であることを次のように定める.

$$A \equiv_T B \overset{\text{def}}{\iff} B \leq_T A \text{ かつ } A \leq_T B.$$

2. A の**チューリング次数 (Turing degree)** $\deg(A)$ を以下の通り定義する.

$$\deg(A) \overset{\text{def}}{=} \{X \subseteq \mathbb{N} \mid X \equiv_T A\}.$$

3. $\mathcal{D}$ と書いてチューリング次数全体の集まりを表わす. $\mathcal{D}$ 上に $\leq_T$ から導かれる順序 $\leq$ を以下のように定義する.

$$\deg(B) \leq \deg(A) \overset{\text{def}}{\iff} B \leq_T A.$$

$\deg(A)$ と書くように, チューリング次数はある集合を代表元とする (チューリング等価に関しての) 同値類である. 以下では単に $\mathbf{a}$ と書いて次数を表現している.

定義 2.21. 1. B が **c.e. in** A とは, B が A をオラクルとして c.e. であるときにいう. このとき単に B は **A-c.e.** と書く.

2. 次数 $\mathbf{b}$ が **c.e. in** $\mathbf{a}$ とは, ある $B \in \mathbf{b}$ が c.e. in $A \in \mathbf{a}$ なるときにいう.

3. 次数 $\mathbf{a}$ が **c.e.** であるとは, $\mathbf{a}$ が c.e. 集合を含むときにいう.

2.7 1-ジェネリック集合

イントロダクションで述べたように, 1963 年にコーエンが編み出した強制法 (フォーシング) にはジェネリック集合という概念が登場する. コーエンの専門分野は元々, どちらかとういと解析学であった. コーエンをロジックへ導いたのは, スタンフォード大学の同僚 Solomon Feferman である. Feferman (1965) と Hinman (1969) は, コーエンの結果の後に続いて, 強制法とジェネリック集合の概念を計算理論に応用した. こうした取り組みの中で生まれた概念の一つに 1-ジェネリック集合があり, その基本的な性質は Jockusch (1977) によって体系化されている.

定義 2.22. 1. $A \subseteq \mathbb{N}$ が **1-ジェネリック (1-generic)** 集合であるとは, 任意の有限列の c.e. 集合 X に対し以下のどちらかが成立するときにいう. σ, τ は有限列を表わしている.

(a) $(\exists \tau \subset A)[\tau \in X]$.

(b) $(\exists \tau \subset A)(\forall \sigma \supseteq \tau)[\sigma \notin X]$.

このようなとき, A**(または** τ**) は** X **を強制 (force) する**といい, $A \Vdash X$(または $\tau \Vdash X$) と書く.

2. 次数 $\mathbf{a}$ が **1-ジェネリック次数**とは, $\mathbf{a}$ が 1-ジェネリック集合 A を含むときにいう.

定理 2.18 より上の式は, 任意の有限列の c.e. 集合 X の部分を書き換えて

$$(\forall e \in \mathbb{N})(\exists \tau \subset A)[\tau \in W_e \text{ or } (\forall \sigma \supseteq \tau)\sigma \notin W_e]$$

とも表現出来る.

n-ジェネリックも簡単に定義が可能である. 定理 2.18 より, c.e. 集合 X は Σ_1 に属している. この Σ_1 の部分を Σ_n に置き換えれば n-ジェネリックの定義そのものになる. つまり n-ジェネリックというのは算術的階層別に強制を行う集合である.

3 Wu's finite injury priority argument

このセクションでは, Wu[9] の証明に用いられている **finite injury priority argument(有限回しか傷付けられない優先論法)** について解説する.

まずは優先論法が証明に用いられた代表的な定理を紹介する.

定理 3.1. (Friedberg-Muchnik Theorem)
$\mathbf{a} \not\leq \mathbf{b}$ かつ $\mathbf{b} \not\leq \mathbf{a}$ なる c.e. 次数 $\mathbf{a}, \mathbf{b}$ が存在する.

この定理をまず「$A \not\leq B$ なる A を構成する」ことから考える. 1 つ固定した B をオラクルにし, いかなるオラクル付きチューリング機械でも計算出来ないような A を構成することは対角線論法で可能である. しかし逆の還元が存在しないことを証明するために, 構成した A に対して「$B \not\leq A$ なる B を構成する」ことが必要になる. しかし A は最初に固定した B について構成したものであり, B を新たに構成する訳にはいかない.

これがこの定理の難しいところであり, この難関を越えるために編み出された手法が**優先論法**である. 簡単に言えば, 対角線論法を並列処理していく手法である. 目的の集合の性質を自然数で番号付けされた無限個の**要求 (requirement)** に分割し, ステージ毎に要求が満たされるか否かをチェックしながら集合を構成する. **ステージ (stage)** は漸化式でいう添え字 n に相当する概念であり, 直前のステージまでに構成された集合を利用して現在のステージにおける集合を構成する.

しかしながら, 無限個の要求は競合してしまう恐れがある. そこで優先論法では無限個の要求に対し**優先度 (priority)** を設定している. こうすることで, 競合するような要求が存在すれば優先度の高い要求を優先的に処理をすることが可能になる. しかし, 優先度の高い要求を満たしたために優先度の低い要求が満たされなくなることがある. これを**傷付け (injury)** と呼び, この傷付けが 1 つの要求については有限回しか発生しないことから finite injury priority argument という名が付いている.

Wu[9] では基本的な finite injury priority argument に加え, **サイクル**という概念が登場し, 実に巧妙な働きをする. 優先論法を用いた証明は, 構成の戦略 (strategy), 具体的な構成法 (construction), その構成法の正当性の検証 (verification) の 3 段階に分かれている.

3.1 構成の戦略

Wu[9] で示されている定理は以下の通り.

定理 3.2. c.e. 次数かつ計算可能でない $\mathbf{a}$ に対し, 1-ジェネリック次数 $\mathbf{a_0}$, $\mathbf{a_1}$ が存在して $\mathbf{a_0} \cup \mathbf{a_1} = \mathbf{a}$ が成立する.

ただしこの $\cup$ は和集合の意味ではなく, 上半束（upper semilattice）の上限演算を表している. すなわち, $\mathbf{a_0}, \mathbf{a_1} \leq \mathbf{a}$ を満たす最小のものである

$A \in \mathbf{a}$ なる真の c.e. 集合をひとつ固定する．以下の要求を満たすような集合 G_0, G_1 を構成すれば証明は完了する．

$$\mathcal{C} : A = \Gamma^{G_0 \oplus G_1}. \qquad \text{(Coding)}$$
$$\mathcal{P} : G_0, G_1 \leq_T A. \qquad \text{(Permitting)}$$
$$\mathcal{G}_{e,i} : (\exists \tau \subset A)[e \in W_e^\tau] \vee (\exists \tau \subset A)(\forall \sigma \supseteq \tau)[e \notin W_e^\sigma]. \qquad \text{(Genericity)}$$

ここで書かれている W_e^τ というのは,「自然数 e で表現されるプログラムが τ をオラクルとしたときに, 有限時間で停止するような入力全体の集合」である．Genericity は定義 2.22 とは少々異なるように見えるが, 以下の命題が成立する．

命題 3.3.

G が 1-ジェネリック集合 $\iff (\forall e \in \mathbb{N})(\exists \tau \subset G)[e \in W_e^\tau \text{ or } (\forall \sigma \supseteq \tau)[e \notin W_e^\sigma]$.

Proof. 定義 2.22 と右辺が等価であることを確かめればよい．まず, 任意の自然数 e に対し, 以下のような集合を定義する．

$$\begin{aligned} X_e :&= \{\tau : e \in W_e^\tau\} \\ &= \{\tau : (\exists s) e \in W_{e,s}^\tau\}. \end{aligned}$$

$e \in W_{e,s}^\tau$ は, e 番目の τ というオラクル付きチューリング機械に e を入力し, s ステップ以内に計算が停止することを表している．「s ステップ以内」というタイマーが付いているため, これは計算可能な関係である．定義 2.17, 定理 2.18 より, X_e は c.e. 集合である．

G が 1-ジェネリック集合の場合, 任意の自然数 e に対し, X_e を強制するような $\tau \subset G$ なる τ が存在する．このとき, 定義 2.22 と X_e の定義から以下が成立する．

$$\begin{aligned} \tau \text{が } X_e \text{を強制する} &\iff [\tau \in X_e \text{ or } (\forall \sigma \supseteq \tau)\sigma \notin X_e] \\ &\iff [e \in W_{e,s}^\tau \text{ or } (\forall \sigma \supseteq \tau) e \notin W_{e,s}^\sigma]. \end{aligned}$$

これはまさに 2 種類の 1-ジェネリック集合の定義が等価であることを示している． □

任意の e, i について $\mathcal{G}_{e,i}$ が満たされることが, G_0, G_1 が 1-ジェネリック集合となる定義そのものである．

Coding と Permitting は, $A \equiv_T G_0 \oplus G_1$ が満足されるための要求である．これらの要求がみたされるとき，$\mathbf{a_0} \cup \mathbf{a_1} = \mathbf{a}$ が成立する．

ここで以下の性質を利用して, 構成に有利な c.e. 集合 A をとってくる．

性質 3.4. 任意の c.e. 集合 A に対し, 以下を満たすような近似列が存在する.

$$(\forall s \in \mathbb{N}) \quad |A_{s+1} - A_s| = 1.$$

Proof. c.e. 集合 A に収束するような近似列 $\{\tilde{A}_s\}_{s\in\mathbb{N}}$ をひとつとる.

まず, $A_{1,0} := \tilde{A}_0$ とする.

$s \in \mathbb{N}_{\geq 1}$ に対し, $\mathrm{que}_s^+ := A_s - A_{s-1}$ と定義する. 以下では $A_{s,t}$ まで定義されているものとする.

Case 1$(\mathrm{que}_s^+ \neq \emptyset)$

$x \in \mathrm{que}_s^+$ を任意にとり, $A_{s,t+1} := A_{s,t} \cup \{x\}$ とする.
$\mathrm{que}_s^+ := \mathrm{que}_s^+ - \{x\}$ と再定義し, $t := t+1$ としてから分岐の頭まで戻る.

Case 2$(\mathrm{que}_s^+ = \emptyset)$

$t := 0$ にリセットし, $s := s+1, A_{s+1,0} := A_{s,t}$ として分岐の頭まで戻る.

このプログラムに従うと, $A_{1,0}, A_{1,1}, \ldots, A_{1,t_1}, A_{2,0}, \ldots, A_{2,t_2}, A_{3,0}, \ldots$ といった $A_{s,t}$ の列ができる. この列は添え字を順番に付け直した上で,

$$(\forall s \in \mathbb{N}) \quad |A_{s+1} - A_s| \leq 1.$$

となっている. 更に $A_{s+1} = A_s$ となるような $s \in \mathbb{N}$ を省略すれば, 求める A の近似列が得られる.

この一連の操作を**正規化 (normalization)** と呼ぶことにする. □

このような正規化を施した A_s について, $x \in A_s - A_{s-1}$ をステージ s で A に**エニュメレイトされる** x と呼び, a_s と書く.

3.1.1 Wu による構成の全体像

構成の戦略は, 基本的には各要求 $\mathcal{G}_{e,i}$ が満足されるように, ジェネリック集合 G_0, G_1 に収束するような有限列を徐々に伸ばしていくことにする. $\mathcal{P}$ のためには $\Gamma^{G_0\oplus G_1}(x)$ という A を正しく計算する関数を下から徐々に定義していく. $\mathcal{C}$ のためには $\Gamma^{G_0\oplus G_1}(x)$ を計算する過程でどれだけオラクルの領域を使用したかを表わす関数 $\gamma(x)$ を下から徐々に定義していく.

これらの戦略を一手に担うのが, サイクルという「適切な有限列探査プログラム」である. ここでいう「適切」というのはジェネリック性の 1 つ, $(\exists \tau \subset A)[e \in W_e^\tau]$ を満たす,

ということである．サイクルは適切な有限列 τ を探査しながらも, $\Gamma^{G_0 \oplus G_1}(x)$ と $\gamma(x)$ を定義していく．しかし適切な有限列 τ を発見してもすぐには近似列として確定はさせないで保留しておく．適切な有限列を G_i の近似として確定することと, A を計算するように作るべき $\Gamma^{G_0 \oplus G_1}(x)$ の修正をセットで行う．(A に x がエニュメレイトされる, すなわち $A(x) = 1$ となる際, $\Gamma^{G_0 \oplus G_1}(x) = 0$ という計算結果を一度壊し再定義する)．こうすることで, ジェネリック性の確保と同時に $\mathcal{P}$ を満足することが可能となる．しかしそのような都合の良いタイミングが訪れなかった場合の保険として, サイクルは別のサイクルを生み出し, 実行しておく．また $\mathcal{C}$ は, G_i から A を計算する必要がある．$G_{i,s}$ が長さ $\gamma(x)$ ぶんだけ収束先である G_i と一致していれば, 以降 $\Gamma^{G_{0,s} \oplus G_{1,s}}(x)$ の値に変更が起きない, すなわち $A(x)$ の値そのものである．ここで $\gamma(x)$ は実際のオラクルの使用領域を表わす関数ではなく, 構成が終わった後の検証の段階で「オラクルの使用領域を表わす関数のように振る舞う」だけであることに注意しなければならない．

しかしながら構成は真っすぐに進むわけではない．満足していない要求は注目を要求する (equires attention)．注目を受けた要求は適切な近似列を定義してもらうことで満足する．しかし初期の状態では全ての要求が満足しておらず, 同時に注目を要求することになる．しかもある要求にとっては適切な近似列であっても, 他の要求にとっては全く適切でない場合もある．こうなってしまうと構成は破綻してしまう．

そこで各要求 $\mathcal{G}_{e,i}$ に優先度 (priority) を設定する．同時に注目を要求した要求のうち, 優先度の高い要求を優先して対処することで, 秩序ある構成を行う．具体的には, 優先度の高い要求は自身より優先度の低い要求について一切気を使わない, つまり優先度の低い要求が適切な有限列を先に発見していたとしても, それをなかったことにして自分が発見した有限列を正式な近似としてしまう．これを, 優先度の低い要求が優先度の高い要求が傷付け (injury) を受けたと呼ぶ．傷付けられた要求は再度ふりだしにもどって適切な有限列を探査することをやり直す．このように優先度を定めることで, 1 つ 1 つの要求は自分より優先度の高い要求全てが満足していれば絶対に傷付けられることはない．優先度の最も高い要求は当然誰にも傷付けられないため, 各要求は優先度の高い順に徐々に満足していくことになる．

ある 1 つの要求が傷付けられっぱなしで, いつまで経っても満足することがないような状況になると, 結局ジェネリック性が確保されない, 構成がどこかで永遠に足踏みして進まなくなるといった不都合が生じる．そのため, このようなことが起こらないこと, すなわち 1 つの要求についての傷付けは有限回しか発生しないことは検証しなければならない．また, 構成を考える上で注意しなければならないのは, 各ステージがきちんと有限時間内に終了することである．1 ステージの内に, 計算不可能なクエリや, 無限回繰り返すような

命令文が入り込まないよう, 有限の資源を使用して構成を行わなければならない.

優先度は $\langle e,i\rangle < \langle e',i'\rangle$ なる $e,e' \in \mathbb{N}$ に対し, $\mathcal{G}_{e,i} > \mathcal{G}_{e',i'}$($\mathcal{G}_{e,i}$ の方が $\mathcal{G}_{e',i'}$ より優先度が高い) と設定する. つまり,

$$\mathcal{G}_{0,0} > \mathcal{G}_{0,1} > \mathcal{G}_{1,0} > \mathcal{G}_{1,1} > \mathcal{G}_{2,0} > \cdots > \mathcal{G}_{e,i} > \cdots$$

となる (ただし, $\langle\ ,\ \rangle : \mathbb{N}^2 \to \mathbb{N}$ は対関数である. これは全単射な計算可能関数である).

3.1.2 サイクルの戦略

以下には要求が複数個もつ「適切な有限列探査プログラム」である**サイクル**の戦略について詳細を記す.

各サイクルにも番号付けがされており, cycle(0), cycle(1), ... , cycle(n), ... というように記述する. 各サイクルが探査する適切な有限列のことを**ターゲット**と呼び, ターゲットよりも前の段階で定義される「適切かもしれない」ような有限列を**仮ターゲット**と呼ぶ. 以下はステージ s_n における要求 $\mathcal{G}_{e,i}$ がもつ cycle(n) の戦略についてであり, 構成中の G_i は α_{i,s_n} と書く. 現在 $\alpha_{0,s_n-1}, \alpha_{1,s_n-1}$ まで構成済みであるとする.

サイクルの戦略

(1) まず閾値 $k^n_{e,i}$ として**フレッシュ**な値を設定する. 但しここで言うフレッシュな値とは, 「今までの構成の内で現れたいかなる数よりも大きい数」として定義される. $x \leq k^n_{e,i}$ なる全ての x について, $\Gamma^{G_0\oplus G_1}_{s_n}(x)$ が未定義であれば, $\Gamma^{G_0\oplus G_1}_{s_n}(x) := A_{s_n}(x)$ とする. このときこの Γ に付随する値, use $\gamma_{s_n}(x)$ をフレッシュかつ単調増加関数となるように定義する. つまり,

$$\gamma_{s_n}(0) < \gamma_{s_n}(1) < \cdots < \gamma_{s_n}(k^n_{e,i})$$

となるようにする. このとき, cycle(n) の仮ターゲットを $\sigma^n_{e,i}$ と書き,

$$\sigma^n_{e,i} := \alpha_{i,s_n-1}{}^\frown\langle 0,\ldots,0\rangle \text{ (但し } |\sigma^n_{e,i}| = \gamma_{s_n}(k^n_{e,i})).$$

と定義する.

最後に $\alpha_{i,s_n} := \sigma^n_{e,i}, \alpha_{1-i,s_n} := \alpha_{1-i,s_n-1}$ と定義して次のステージへ進む.

(2) 各ステージ $s > s_n$ ではジェネリック性を満たすような適切な $\tau \supseteq \sigma^n_{e,i}$ を探査する. 具体的には,

$$|\tau| < s \text{ かつ } e \in W^\tau_{e,s}$$

なる τ を探す. この τ を cycle(n) のターゲットと呼び, ターゲットが見つかるまで cycle(n) はこのステップを繰り返す.

(3) ターゲット τ を発見したステージを $u > s_n$ とする. このとき $y < k^n_{e,i}$ なる y について $f_{e,i}(y)$ が未定義であれば, $f_{e,i}(y) := A_u(y)$ とする.
さらにこの要求 $\mathcal{G}_{e,i}$ がもつ cycle として新たに cycle$(n+1)$ を生み出してステップ (1) から実行する.
cycle(n) は以降, A に $k^n_{e,i}$ 未満の数がエニュメレイトされるのを待つ.

(4) $k^n_{e,i}$ 未満の数が A にエニュメレイトされるステージを $t_n > u$ とする. (つまり $a_{t_n} < k^n_{e,i}$)
このとき, α_{1-i,t_n} として, α_{1-i,t_n-1} の $\gamma_{t_n-1}(a_{t_n})$ ビット目を 1 にしたものを定義する ($\gamma_{t_n-1}(a_{t_n})$ を G_{1-i} にエニュメレイトするということ).
この操作により, $\Gamma^{G_0\oplus G_1}_{t_n}$ の計算結果は, オラクルとして $\gamma_{t_n-1}(a_{t_n})$ 以上の領域を使用したものについて全て破壊される. use γ_{t_n} は単調増加になるように定義しているため, $x \geq a_{t_n}$ なる全ての x について $\Gamma^{G_0\oplus G_1}_{t_n}(x)$ は未定義へと戻る. 最後に $\alpha_{i,t_n} := \tau$ として次のステージへ進む.

3.1.3 サイクルの動作が導く結果

cycle(n) が動作を終了したステップ毎にどのような結果が導かれるかを以下に述べる. これらは後々の ($\mathcal{G}_{e,i}$ の) 検証の段階で非常に重要になる.

(A) cycle(n) がステップ (3) までたどり着けない場合, すなわち $|\tau| < s$ かつ $e \in W^\tau_{e,s}$ なる τ を発見出来なかった場合, $\mathcal{G}_{e,i}$ は,

$$(\forall \sigma^n_{e,i} \subseteq \tau)[e \notin W^\tau_e]$$

を $\sigma^n_{e,i}$ によって満たすことになる.

(B) cycle(n) はステップ (3) に留まり続ける場合, すなわち cycle(n) はターゲット τ を発見したものの $k^n_{e,i}$ 未満の数が A にエニュメレイトされない場合, $\mathcal{G}_{e,i}$ は cycle(n) によっては満たされない. このとき, $x \leq k^n_{e,i}$ なる値について $A(x)$ を参照し $f(x)$ を定義する. そして cycle$(n+1)$ を実行する. cycle$(n+1)$ がとる閾値 $k^{n+1}_{e,i}$ は $k^n_{e,i}$ より大きくとっている. そのため cycle$(n+1)$ もステップ (3) に留まる場合, $k^{n+1}_{e,i}$ 未満の数が A にエニュメレイトされておらず, $f(x)$ の定義域は更に広がることになる.
つまりサイクルが次々に生み出されるものの, そのどれもがステップ (3) に留まっ

ている場合, A にエニュメレイトされる数が閾値 $k^{-}_{e,i}$ から逃げ続けているような状況になっている.

(C) cycle(n) はステップ (4) に辿り着いた場合. このとき $(\exists \tau \subset A)[e \in W^{\tau}_{e}]$ はまさしく探索によって発見した τ そのもので満足される.

$\mathcal{G}_{e,i}$ は自身がもつサイクルの内, どれか 1 つでも (A) または (C) の結果になっていれば良く, 以降傷付けられて振り出しに戻らない限り再び注目を要求することもないようにする.

3.2 Wu の構成方法

このセクションでは具体的な構成方法を記す. 構成を分かりやすくするために, A のエニュメレイトは奇数ステージでしか起こらないようにしておく. この操作は性質 3.4 で示した正規化を利用すれば簡単に行える.

まずはいくつかの言葉を定義しておく.

定義 3.5.

1. $\mathcal{G}_{e,i}$ の cycle(n) が発見したターゲット τ がステージ s で**リアライズ**されるとは, $a_s < k^{n}_{e,i}$ なる a_s が A にエニュメレイトされ, $\alpha_{i,s} := \tau$ と定義されたときにいう.
2. $\mathcal{G}_{e,i}$ が**初期化 (initialize)** されるとは, $\mathcal{G}_{e,i}$ より優先度の高い要求により, 自身のとっていた戦略を全てリセットし, サイクルを持たないような状態に戻ることをいう.
3. $\mathcal{G}_{e,i}$ の cycle(n) がステージ s で**リフレッシュ**されるとは, ターゲット τ を発見する前に $a_s < k^{n}_{e,i}$ なる a_s が A にエニュメレイトされたときにいう.
4. $\mathcal{G}_{e,i}$ が**注目を要求する (requires attention)** とは以下のいずれかが起きた場合にいう.
   (1) $\mathcal{G}_{e,i}$ が未だサイクルをもっていない場合.
   (2) $\mathcal{G}_{e,i}$ の cycle(n) がリフレッシュされた場合.
   (3) $\mathcal{G}_{e,i}$ の cycle(n) がターゲット τ を発見した場合.
   (4) ターゲット τ がリアライズされた場合.

G_0 と G_1 の構成

ステージ 0:

$\alpha_{i,0} := \emptyset \quad (i = 0, 1)$ として次のステージへ.

ステージ s+1:　I. $s = 2k$ (A のエニュメレイトが行われるステージ).

まず, $\langle e, i\rangle < s + 1$ なる e, i について, 注目を要求するような $\mathcal{G}_{e,i}$ を探す. このような $\langle e, i\rangle$ が見つからない場合は, $\gamma_s(a_{s+1})$ を G_0 にエニュメレイトして次のステージへ進む.

そうでなければ, 最小の $\langle e, i\rangle$ を選び, すなわち最も優先度の高い $\mathcal{G}_{e,i}$ を選び, 選んだ $\mathcal{G}_{e,i}$ のサイクルかつ注目を要求している cycle(n) の内最小の n について考える. この場合, 以下の 2 つのサブケースが考えられる.

IA. (ターゲット τ がリアライズされた場合)

まずは $\alpha_{1-i,s+1} := (\alpha_{1-i,s} \upharpoonright \gamma_s(a_{s+1}))^\frown\langle 1\rangle$ とし (すなわち $\gamma_s(a_{s+1})$ を G_{1-i} にエニュメレイトするということ), $x \geq a_{s+1}$ なる x について $\Gamma^{G_0 \oplus G_1}(x)$ を未定義にする. 続いて $\alpha_{i,s+1} := \tau$ と定義する.

IB. (ターゲット τ が未だ見つかっていない場合)

IA 同様, まずは $\alpha_{1-i,s+1} := (\alpha_{1-i,s} \upharpoonright \gamma_s(a_{s+1}))^\frown\langle 1\rangle$ とする. 次に $a_{s+1} \leq x \leq k^n_{e,i}$ なる x について, $\Gamma^{G_0 \oplus G_1}(x) := A_{s+1}(x)$ と再定義し, それに伴って $\gamma_{s+1}(x)$ もフレッシュかつ単調増加関数となるように再定義する.

更に以下のように仮ターゲット $\sigma^n_{e,i}$ を再定義し, ターゲット τ の探査を続ける.

$$\sigma^n_{e,i} := \alpha_{i,s+1}{}^\frown\langle 0 \cdots 0\rangle \text{ (但し } |\sigma| = \gamma_{s+1}(k^n_{e,i})).$$

IA,IB どちらの場合も, $m > n$ なる m について $\mathcal{G}_{e,i}$ の cycle(m) をキャンセルし, $\mathcal{G}_{e,i}$ より優先度の低い要求全てを初期化する.

II. $s = 2k + 1$ (新たなサイクルの実行, 閾値の設定などを行うステージ).

まず, $\langle e, i\rangle < s + 1$ なる e, i について, 注目を要求するような $\mathcal{G}_{e,i}$ を探す. そのうち最小の $\langle e, i\rangle$ について以下の 2 つのサブケースが考えられる.

IIA. ($\mathcal{G}_{e,i}$ がサイクルを持たない場合)

$\mathcal{G}_{e,i}$ の最初のサイクルとして cycle(0) を以下の通り実行する.

(a) まず閾値 $k^0_{e,i}$ としてフレッシュな値を設定する.

(b) $x \leq k^0_{e,i}$ なる全ての x について, $\Gamma^{G_0 \oplus G_1}_s(x)$ が未定義であれば, $\Gamma^{G_0 \oplus G_1}_s(x) :=$

$A_s(x)$ とする．このときこの Γ に付随する値, use $\gamma_s(x)$ をフレッシュかつ単調増加関数となるように定義する．

(c) $\sigma^0_{e,i} := \alpha_{i,s-1}{}^\frown\langle 0,\ldots,0\rangle$ (但し $|\sigma^0_{e,i}| = \gamma_s(k^0_{e,i})$) と定義する．

(d) $\alpha_{i,s} := \sigma^0_{e,i}, \alpha_{1-i,s} := \alpha_{1-i,s-1}$ と定義する．

IIB. ($\mathcal{G}_{e,i}$ の cycle(n) がターゲット τ を発見した場合)

$z < k^n_{e,i}$ なる z について $f_{e,i}(z)$ が未定義ならば $f_{e,i}(z) := A_{s+1}(z)$ とし, τ がリアライズされるのを待つ．更に cycle($n+1$) を生み出して以下の通り実行する．

(a) まず閾値 $k^{n+1}_{e,i}$ としてフレッシュな値を設定する．(注: $k^{n+1}_{e,i}$ は $k^n_{e,i}$ より更に大きい値である)

(b) $x \leq k^{n+1}_{e,i}$ なる全ての x について, $\Gamma^{G_0\oplus G_1}_s(x)$ が未定義であれば, $\Gamma^{G_0\oplus G_1}_s(x) := A_s(x)$ とする．このときこの Γ に付随する値, use $\gamma_s(x)$ をフレッシュかつ単調増加関数となるように定義する．

(c) $\sigma^{n+1}_{e,i} := \alpha_{i,s-1}{}^\frown\langle 0,\ldots,0\rangle$ (但し $|\sigma^{n+1}_{e,i}| = \gamma_s(k^{n+1}_{e,i})$) と定義する．

(d) $\alpha_{i,s} := \sigma^{n+1}_{e,i}$, $\alpha_{1-i,s} := \alpha_{1-i,s-1}$ と定義する．

どちらのサブケースでも自身より優先度の低い要求を全て初期化してから次のステージへ進む．

構成法については以上である．

$$G_i := \lim_{s\to\infty} \alpha_{i,s} \ (i = 0, 1).$$

と定めれば G_i こそが求める 1-ジェネリック集合となる．

3.3 検証

このセクションでは前セクションで述べた構成法の正当性の検証 (verification) を行う．正当であるとは全ての要求が満足されていることに他ならない．

3.3.1 $\mathcal{G}_{e,i}$ の検証

補題 3.6. 任意の $e, i \in \mathbb{N}$ に対して以下が成立する．

1. $\mathcal{G}_{e,i}$ は高々有限回しか初期化されない;
2. $\mathcal{G}_{e,i}$ は高々有限個のサイクルしか持たない;
3. $\mathcal{G}_{e,i}$ は高々有限回しか注目を要求しない;

4. $\mathcal{G}_{e,i}$ は満足される.

Proof. 1. を証明する.

証明には帰納法を用いる. まず, 最も優先度の高い $\mathcal{G}_{0,0}$ は絶対に初期化されない. $\langle e,i\rangle$ を 1 つ固定し, $\langle e',i'\rangle < \langle e,i\rangle$ なる $\mathcal{G}_{e',i'}$ は全て高々有限回しか初期化されないとする. ゆえに $\mathcal{G}_{e',i'}$ が最後に行動を起こした (ある有限列を α として定義した) ステージが存在する. そのステージを s とする. s 以降, $\mathcal{G}_{e,i}$ は絶対に初期化されない. 以上で 1. は示された.

2. を証明する.

証明には背理法を用いる. ゆえに $\mathcal{G}_{e,i}$ は無限個のサイクルを持つと仮定する. すると構成法より, $f_{e,i}$ が全域関数となる.
ここで更に $f_{e,i}$ が真に A を計算する関数になっていることを示す. これも背理法で証明する. まず, $f_{e,i}(x) \neq A(x)$ が成立するような最小の x が存在すると仮定する.
1. の結果より, ステージ s を $\mathcal{G}_{e,i}$ が以降初期化されないようなステージとしてとることが出来る. この時 $f_{e,i}(x)$ を定義するようなステージが s より後に必ず存在し, そのようなステージを t_0 とする. 構成法より, ステージ t_0 では $\mathcal{G}_{e,i}$ はターゲット τ を発見しており, その τ は cycle(n) によって発見され, $|\tau| < t_0, \tau \supset \sigma^n_{e,i}, W^\tau_e(e)\downarrow$ を満たすようなものである. 構成法に従い, $f_{e,i}(x) := A_{t_0}(x)$ $(x \leq k^n_{e,i}$かつ $f_{e,i}(x)\uparrow)$ と定義する.

ところで $f_{e,i}(x) \neq A(x)$ が成立するのは, $f_{e,i}(x) = 0, A(x) = 1$ となる場合しか有り得ない. もし $f_{e,i}(x) = 1, A(x) = 0$ だとすると, $f_{e,i}(x) := 1 = A_{t'}(x)$ と定義するようなステージ t' の後, A から x がドロップアウトしていることになる. この場合, $A(x)$ のマインドチェンジは 2 回発生しており, A が c.e. 集合であることに反する. ゆえに $f_{e,i}(x) = 0, A(x) = 1$ でしか $f_{e,i}(x) \neq A(x)$ は成立し得ない.

このことから, x が A にエニュメレイトされるようなステージは必ず存在し, そのステージを $t_1(> t_0)$ とする. この時, $x \leq k^n_{e,i}$ であり, $s < t_0 < t_1$ ゆえにステージ t_0, t_1 の間に初期化もされないので, ターゲット τ はリアライズされる. しかしこれは無限にサイクルを持つことに矛盾する. ゆえに, $f_{e,i}$ は A を計算する関数になっている.

しかしいかなる x を入力されても $f_{e,i}(x)$ は各ステージが有限時間で終了するような構成を十分な時間シミュレーションを行うことで計算可能である. しかしこれは A が真の c.e. 集合であることに矛盾する. 以上で $\mathcal{G}_{e,i}$ が無限個のサイクルを持つことはない.

3. を証明する.

1., 2. より簡単である. 初期化が有限回しかされず, サイクルも有限個しか持たないのならば, 注目を要求する回数も高々有限回である.

4. を証明する.
3. よりステージ t を $\mathcal{G}_{e,i}$ が最後に注目を要求するステージとする. このとき 2 つの場合が考えられる.

I. (t で新たにサイクルを生み出す, またはリフレッシュされる場合)
この注目の要求が最後ゆえ, ターゲット τ が見つからない場合である. 仮ターゲット $\sigma^n_{e,i}$ こそが

$$G_i \supset \sigma^n_{e,i},\ \forall \tau \supset \sigma^n_{e,i},\ W^\tau_e(e)\uparrow .$$

を満たすようなものであり, $\mathcal{G}_{e,i}$ は満足される.

II. (t でターゲット τ がリアライズされる場合)
このときはリアライズされたターゲット τ により,

$$G_i \supset \tau,\ W^\tau_e(e)\downarrow .$$

が満たされ, $\mathcal{G}_{e,i}$ は満足される.

最後の注目の要求が, cycle(n) がターゲット τ を発見することによるものであった場合, 以降注目の要求が無いことから, 新たに生み出した cycle($n+1$) がターゲット τ を発見出来なかった場合なので, I. に帰着する. 以上で 4. は示された.

以上で補題 3.6 は示された. □

3.3.2 $\mathcal{P}$ の検証

補題 3.7. $\mathcal{P}$ は満足される. すなわち, $G_0, G_1 \leq_T A$.

Proof. $G_0 \leq_T A$ のみ示す. G_1 についても同様である.
任意に x を 1 つとる. A をオラクルとして, $A_s \upharpoonright x = A \upharpoonright x$ なるステージ s を計算する. すると s 以降, $G_{0,s}(x)(= \alpha_{0,s}(x))$ は変化することがない. これを背理法で示す.
偶数ステージでは 1 つ前のステージまでで定義された α を拡張するのみであり, 奇数ステージのみでしか α の変更は発生しない. つまり s 以降で $\alpha_{0,s}(x)$ が変化すると仮定す

ると, それは必ず奇数ステージで発生 (それをステージ t とする), すなわち a_t が A にエニュメレイトされたことによってしか起こり得ない.

変化には 2 パターンあり, 1 つ目は $x = \gamma_{t-1}(a_t)$ がエニュメレイトされる場合である. $\gamma_{t-1}(a_t)$ は定義の仕方から, 必ず $a_t < \gamma_{t-1}(a_t)$ を満たし, 変更が起きる値はまさしく $x = \gamma_{t-1}(a_t)$ である. $a_t < x$ であるので, $s < t$ かつ $A_s \upharpoonright x = A \upharpoonright x$ かつ A は c.e. 集合であることに矛盾する.

よって $\alpha_{0,s}(x)$ の変化は奇数ステージ t でターゲット τ をリアライズしたことによってしか起こり得ない. しかしこれも閾値 $k^n_{e,i}$ をフレッシュ, γ をフレッシュかつ単調増加になるように定義していることから, $\alpha_{0,s}(x)$ に変化を起こすには, x 未満の数がステージ s 以降に A にエニュメレイトされる必要があり, $s < t$ かつ $A_s \upharpoonright x = A \upharpoonright x$ かつ A は c.e. 集合であることに矛盾する.

以上より, ステージ s 以降に $G_{0,s}(x)$ が変化しない, すなわち $G_{0,s}(x) = G_0(x)$ であることが示された. 与えられた x に対し, $G_{0,s}(x)$ を出力すれば, 任意の x に対し A をオラクルとして $G_0(x)$ が計算出来たことになる.

□

3.3.3 $\mathcal{C}$ の検証

補題 3.8. $\mathcal{C}$ は満足される. すなわち $A \leq_T G_0 \oplus G_1$.

Proof. $\Gamma^{G_0 \oplus G_1}$ が全域関数であることを示す.

1 つ x を固定し, $\langle e, i \rangle > x$ なる最小の e, i を考える. すると補題 3.6 より, $\mathcal{G}_{e,i}$ が注目を要求しなくなるステージ t が存在する. t 以降, A は $k^n_{e,i}$ 未満で変化せず, $k^n_{e,i} > \langle e, i \rangle > x$ という不等式が成立している.

更に $t' \geq t$ を, $\gamma(x)$ が定義される最小のステージとする. すると, t' 以降で $\Gamma^{G_0 \oplus G_1}(x)$ は未定義に戻ることはない. もしそうだと仮定すると, x 未満の数で A に t' 以降にエニュメレイトされるような数 y が存在し, $y < x < k^n_{e,i}$ であるので $\mathcal{G}_{e,i}$ が注目を要求しなくなることに矛盾する.

以上で $\Gamma^{G_0 \oplus G_1}$ が全域であることが示された.

次に $\Gamma^{G_0 \oplus G_1}$ が真に A を計算する関数であることを示す.

$A(x) = 1, \Gamma^{G_0 \oplus G_1}(x) = 0$ となるような x が存在すると仮定する. x が A にエニュメレイトされるようなステージを $s + 1$, すなわち $x = a_{s+1}$ とする.

$\gamma_s(a_{s+1}) \downarrow$ ならばこの値が G_i にエニュメレイトされる. しかしこのとき, オラクルの

値が変更されたことにより, $y \geq a_{s+1}$ なる全ての y について $\Gamma^{G_0 \oplus G_1}(y) \uparrow$ となり, 以降のステージで正しい値に再定義されるため矛盾する.

ゆえに $\gamma_s(a_{s+1}) \uparrow$ であるが, これも $\Gamma^{G_0 \oplus G_1}(x) \downarrow$ であることに矛盾する.

以上より $\Gamma^{G_0 \oplus G_1}$ は A を計算する関数になっている.

最後に $A \leq_T G_0 \oplus G_1$ について具体的な還元方法を与える. 任意の x について $A(x)$ の値を決定するためには以下のプロセスに従えばよい.

まず $\gamma_s(x) \downarrow$ となるようなステージ s を探す.

次に G_0, G_1 をオラクルにして,

$$(G_{0,s} \oplus G_{1,s}) \upharpoonright_{2\gamma_s(x)} = G_0 \oplus G_1 \upharpoonright_{2\gamma_s(x)}$$

となるか否かをチェックする. もし成立しなければ, 成立するまで s を徐々に大きくする. 成立したステージを t とし, t より大きいステージ t' について $\Gamma_{t'}^{G_0 \oplus G_1}(x)$ の値を出力すればその値こそが $\Gamma^{G_0 \oplus G_1}(x)$ の値である.

以上で補題 3.8 は示された. □

3.3.4 Wu の定理のまとめ

補題 3.6, 補題 3.7, 補題 3.8 の証明が完了したので, 要求は全て満足することが示された. これで定理 3.2 が証明された. □

4 主定理の証明戦略

本セクションでは主定理へ至るために必要な戦略について述べる. 以下では Wu[9] で使われている finite injury priority argument を直接拡張する方法をとる.

4.1 正規化

まずは構成を有利に進めるために真の 2-c.e. 集合 A に対して正規化を施す. 性質 3.4 で述べたような手法は 2-c.e. 集合に対しても有効である.

補題 4.1. 任意の 2-c.e. 集合 A に収束する近似列 $\{A_s\}_{s \in \mathbb{N}}$ に対し, 以下を満たすような近似列 $\{\tilde{A}_s\}_{s \in \mathbb{N}}$ が存在する.

1. $\tilde{A}_0 = \emptyset$.
2. $\forall s \in \mathbb{N},\ |\tilde{A}_s \triangle \tilde{A}_{s+1}| = 1$.
3. $A = \lim_{s\to\infty} \tilde{A}_s$.

Proof. 性質 3.4 の証明とほぼ同様. 全ての s について, $\mathrm{que}^{+}_{s+1} := A_{s+1} - A_s$, $\mathrm{que}^{-}_{s+1} := A_s - A_{s+1}$ とし, $\mathrm{que}^{\pm}_{s+1}$ の元を 1 つずつ A にエニュメレイト (またはドロップアウト) することで実現出来る. □

4.2 ロールバック

今回も Wu[9] と同様に finite injury priority argument で 1-ジェネリック集合の構成を行う. しかしまったく同じ構成法では A が 2-c.e. 集合になったことによって「A にエニュメレイトされることでリアライズを引き起こすような x が, 後のステージでもう一度マインドチェンジをし A からドロップアウトしてしまう」という不都合が起きてしまう. この現象の回避のためにロールバックという概念を導入する.

ある元 x はマインドチェンジを 2 回行う元であるとする. このとき 1 回目のマインドチェンジ, すなわち A にエニュメレイトされるようなステージで我々は Wu[9] の構成法をほぼそのまま実行する. しかし後のステージで x が 2 回目のマインドチェンジ, すなわち A からドロップアウトするようなステージでは, x がエニュメレイトされた瞬間まで全ての要求の状態, 構成中の有限列, $\Gamma^{G_0 \oplus G_1}(x)$ と $\gamma(x)$ など, ほとんどの変数の値をロールバックする. つまり, 基本的な構成方法はほぼ Wu[9] と同じだが, 間違った構成を進めてしまった場合は構成をやり直し, 間違いの原因ごと無かったことにしてしまう. そのためにこの構成方法では裏で常に全ての変数のバックアップデータを保存しておく. ロールバックはこのバックアップデータを元に行う.

構成の間違い原因把握のために, ドロップアウトしたような x には「後に A からドロップアウトする」というマークを付けておく. マークはロールバックによっても消されることは無く残り続けるようにしておき, マークが付いた x がエニュメレイトやドロップアウトをしても無視して構成を進める.

このような構成のため, ステージには真のステージと見た目上のステージの 2 種類を定義し, それぞれリアルステージとオステンシブルステージと呼ぶ. リアルステージはこの構成全体のステージを表わすものであり, オステンシブルは上述のように何度もやり直される Wu[9] の構成のステージを表わすものである. つまり構成で扱う変数はオステンシブ

ルステージに依存しているように見えるが, 実際はその変数は何度もやり直した結果であり, そのやり直しも含めて構成全体に流れる時間を表現するものがリアルステージである.

すると 1 つのリアルステージには 1 つのオステンシブルステージが対応し, 1 つのオステンシブルステージには 1 つ以上のリアルステージが対応することになる. そのため構成のある時点を表現するために r をリアルステージ, t をオステンシブルステージとして (r, t) というように記述する.

図 2 はオステンシブルステージ s_0, s_1 でエニュメレイトされた a_{s_0}, a_{s_1} がそれぞれオステンシブルステージ t_0, t_1 でドロップアウトする場合のステージの進み方を 2 次元グラフ化したものである. リアルステージの r_0, r_1 でそれぞれロールバックが発生しており, オステンシブルステージはそれぞれ t_0, t_1 から s_0, s_1 へとロールバックが発生している. この瞬間, a_{s_0}, a_{s_1} には「後に A からドロップアウトする」マークが付けられ, その他の変数の値が $s_0 - 1, s_1 - 1$ 時点のものに戻る. その後に再び a_{s_0}, a_{s_1} がドロップアウトするオステンシブルステージ t_0, t_1 がやってくるが, 2 度目以降はマークが付いているため, それを無視し更に構成を進めることができる.

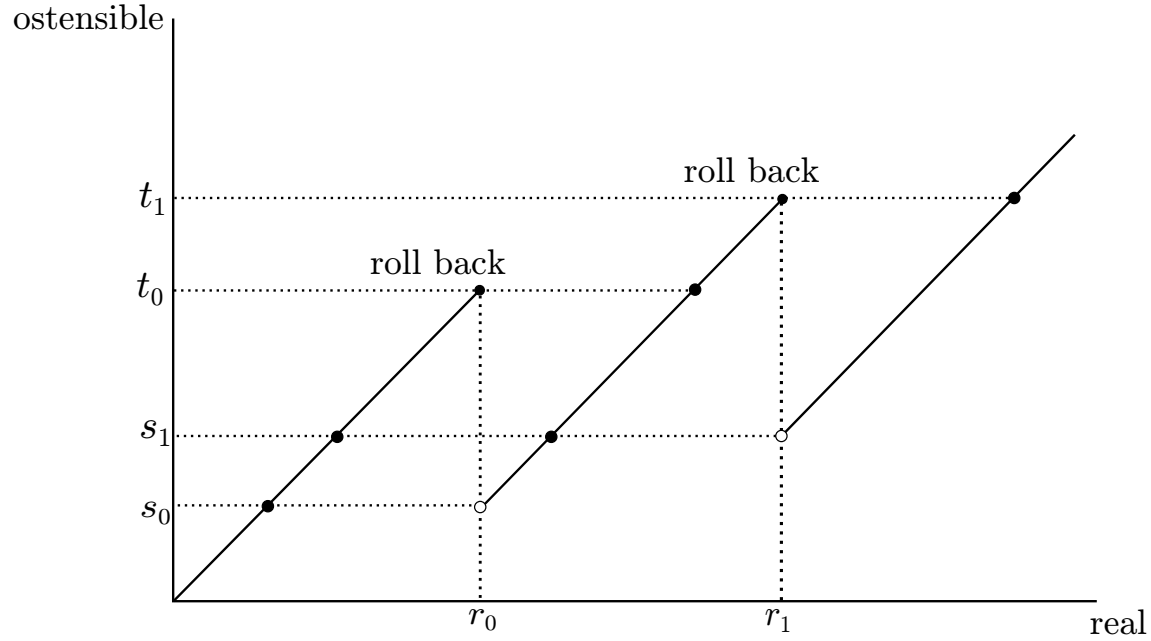

図 2　リアルステージとオステンシブルステージの進み方を表現したグラフ

4.3　証明の概略

まず以下の様な集合 H を定義する.

定義 4.2. H は以下のような自然数の集合と定義する. 但し t_i+1 と書いて, リアルステージ r_i+1 に対応するオステンシブルステージを表すものとする.

$$H := \{r_1+1 : \text{任意の } x \in A_{t_1+1} \text{に対し,「後に } A \text{ からドロップアウトする」マークがリアルステージ } r_1+1 \text{ の終わりに付いていないものついて} \\ \forall r_2 > r_1,\ x \in A_{t_2+1} \text{が成立する.}\}$$

この H についての詳しい性質は後述するが, 特に $A \geq_T H$ が成立する. ゆえに,

$$A \geq_T G_0 \oplus G_1 \oplus H.$$

である. 逆の還元については, ある x が A に属しているか否かを G_0, G_1, H をオラクルにして計算可能であればよい. 基本戦略は補題 3.8 で示した通りだが, 適切なステージ (今回の場合はオステンシブルステージ) を発見したとしても, 後にロールバックが発生, すなわち x がドロップアウトしてしまう可能性がある. そこで集合 H をオラクルにすると, 上述の適切なオステンシブルステージに対応するリアルステージが H に属するか否か計算出来る. 属していれば「対応するオステンシブルステージ以前へとロールバックすることはない」ので, 現時点での結果を出力すればよい. そうでなければ「対応するオステンシブルステージ以前へとロールバックすることがある」ので, 現時点での結果は覆る可能性が残っている. そのときは再び適切なオステンシブルステージを見つけるまで構成のシミュレーションを進め, 対応するリアルステージが H に属するか問うことを繰り返せば, いつかは正しい結果を出力することが出来る.

証明の概略は以上の通りである.

5 構成方法

我々は以下の要求を満たすように構成を進める.

$$\mathcal{C} : A = \Gamma^{G_0 \oplus G_1}. \qquad \text{(Coding)}$$
$$\mathcal{P} : G_0, G_1 \leq_T A. \qquad \text{(Permitting)}$$
$$\mathcal{G}_{e,i} : (\exists \tau \subset A)[e \in W_e^\tau] \vee (\exists \tau \subset A)(\forall \sigma \supseteq \tau)[e \notin W_e^\sigma]. \qquad \text{(Genericity)}$$

以下に具体的構成法を記す.

G_0 と G_1 の構成方法

オステンシブルステージ 0 (リアルステージ 0):

$\mathcal{G}_{0,0}$ の cycle(0) をスタートさせる. cycle(0) は以下の通りに動作する.

(a) 閾値 $k^0_{0,0}$ を a_1 より大きくフレッシュにとる.
(b) $\Gamma_0^{G_0\oplus G_1}(x) := 0$ $(x \leq k^0_{0,0})$ と定義する. それに付随する $\gamma_0(x)$ はフレッシュかつ単調増加関数となるように定義する.
(c) cycle(0) の仮ターゲット $\sigma^0_{0,0}[0]$ を以下のように定義する.([0] は現在のオステンシブルステージが 0 であることを表している)

$$\sigma^0_{0,0}[0](x) := 0 \ (x \leq \gamma_0(k^0_{0,0})).$$

最後に $\alpha_{0,0} := \sigma^0_{0,0}[0], \alpha_{1,0} := \emptyset$ と定義して次のオステンシブルステージへ (リアルステージも次へ進む).

オステンシブルステージ $s+1$ (リアルステージ $r+1$):

I. $s = 2k$ (A のエニュメレイションが行われるステージ).

Case IA: a_{s+1} は「後に A からドロップアウトする」マークが付いている場合.
なにもせずに次のオステンシブルステージへ進む (リアルステージも次へ進む).

Case IB: a_{s+1} は「後に A からドロップアウトする」マークが付いておらず, かつ $a_{s+1} \in A_{s+1} - A_s$ である場合.
注目を要求するような $\langle e, i\rangle < s+1$ なる $\mathcal{G}_{e,i}$ の中で最も優先度の高い要求についてケアする. 注目を要求していることから $a_{s+1} \leq k^n_{e,i}$ である.
以下のどちらか一方が成立しているような n の内, 最大のものを選ぶ.

(i) $\tau^n_{e,i}$ が a_{s+1} によりリアライズされた場合.
(ii) cycle(n) は a_{s+1} によりリフレッシュされた場合.

そして選んだ n について以下の様に変数を更新する.
$\alpha_{1-i,s+1} := (\alpha_{1-i,s} \upharpoonright \gamma_s(a_{s+1}))^\frown\langle 1\rangle$
$\Gamma^{G_0\oplus G_1}_{s+1} := \Gamma^{G_0\oplus G_1}_s \upharpoonright a_{s+1}$
$\gamma_{s+1} := \gamma_s \upharpoonright a_{s+1}$
更に自身より優先度の低い要求 $\mathcal{G}_{e',i'}$ を全て初期化し, 自身の $n < m$ なる cycle(m) も初期化を行う. 以下更に 2 つのサブケースに分岐をする.

Subcase IB (i): $\alpha_{i,s+1} := \tau^n_{e,i}$ とする.

Subcase IB (ii): $\Gamma^{G_0\oplus G_1}_{s+1}(x)$ を以下の通り定義する. $x < a_{s+1}$ なる x については $\Gamma^{G_0\oplus G_1}_{s+1}(x) := \Gamma^{G_0\oplus G_1}_{s}(x)$ とする. $a_{s+1} \leq x \leq k^n_{e,i}$ なる x については, 「後に A からドロップアウトする」マークが付いていれば 0 を, そうでなければ $A_{s+1}(x)$ の値を $\Gamma^{G_0\oplus G_1}_{s+1}(x)$ の値として定義する.

$\gamma_{s+1}(x)$ の値も以下の通り定義する. $x < a_{s+1}$ なる x については $\gamma_{s+1}(x) := \gamma_s(x)$ とする. $a_{s+1} \leq x \leq k^n_{e,i}$ なる x についてはフレッシュかつ単調増加関数となるように定義する.

更に $\sigma^n_{e,i} := \alpha_{i,s}{}^\frown\langle 0, \ldots, 0\rangle$ (但し $|\sigma^n_{e,i}| = \gamma_{s+1}(k^n_{e,i})$) と定義し, 更に $\alpha_{i,s+1} := \sigma^n_{e,i}$ と定義する.

そして Subcase (i),(ii) どちらの場合も次のオステンシブルステージへ (リアルステージも次へ進む.)

Case IC: a_{s+1} は「後に A からドロップアウトする」マークが付いており, かつ $a_{s+1} \in A_s - A_{s+1}$ である場合.

この場合は, $x \in A$ であるつもりで進めていた構成が間違っていたということなのでロールバックを行う.

まず $a_{t_0+1} = a_{s+1} = A_{t_0+1} - A_{t_0}$ なるオステンシブルステージ t_0 とそれに対応するリアルステージのうち最大のものを探し, そのリアルステージを r_0 とする. $(r+1, s+1)$ を**ロールバック出発地**, (r_0+1, t_0+1) を**ロールバック目的地**と呼ぶ. そして, リアルステージ $r+1$ に対応する $\alpha_{0,s+1}, \alpha_{1,s+1}, \Gamma^{G_0\oplus G_1}_{s+1}, \gamma_{s+1}$ をそれぞれ, リアルステージ r_0 に対応する $\alpha_{0,t_0}, \alpha_{1,t_0}, \Gamma^{G_0\oplus G_1}_{t_0}, \gamma_{t_0}$ と定義する. 更に要求の状態, サイクル, ターゲット, 仮ターゲットなど全てリアルステージ r_0 時点で定義されていたものを参照し, 現在の値として定義する. これが**ロールバック**に相当である. ただ一つ, 「後に A からドロップアウトする」マークは巻き戻さずにそのままにしておく. 最後に a_{s+1} に「後に A からドロップアウトする」マークを付ける.

そしてオステンシブルステージ t_0+2(リアルステージ $r+2$) へと進む (オステンシブルステージ t_0+1 は $a_{s+1} = a_{t_0+1}$ がエニュメレイトされるステージなので, 先ほど付けたマークによって Case IA に分岐して何もせずに進むため).

II. $s = 2k + 1$ (新たなサイクルの実行, 閾値の設定などを行うステージ).

$\langle e, i\rangle < s + 1$ なる $\mathcal{G}_{e,i}$ の内, 最も優先度の高いものを選ぶ. このとき以下のどちらか一方が成立している.

(i) $\mathcal{G}_{e,i}$ はサイクルを持っていない場合.
(ii) $\mathcal{G}_{e,i}$ はターゲットを発見した場合.

Subcase II (i): $\mathcal{G}_{e,i}$ の最初のサイクルとして cycle(0) を以下の通り実行する.

(a) まず閾値 $k^0_{e,i}$ としてフレッシュな値を設定する. 特に $k^n_{e,i} > a_{s+2}$ とする.
(b) $x \leq k^0_{e,i}$ なる全ての x について, $\Gamma^{G_0\oplus G_1}_s(x)$ が定義済みであればその値を $\Gamma^{G_0\oplus G_1}_{s+1}(x)$ の値とし, 未定義であり「後に A からドロップアウトする」マークが付いていれば 0, そうでなければ $A_{s+1}(x)$ と定義する.
$\gamma_{s+1}(x)$ の値も以下の通り定義する. $x < a_{s+1}$ なる x については $\gamma_{s+1}(x) := \gamma_s(x)$ とする. $a_{s+1} \leq x \leq k^n_{e,i}$ なる x についてはフレッシュかつ単調増加関数となるように定義する.
(c) $\sigma^0_{e,i} := \alpha_{i,s}{}^\frown\langle 0, \ldots, 0\rangle$ (但し $|\sigma^0_{e,i}| = \gamma_{s+1}(k^0_{e,i})$) と定義する.
(d) $\alpha_{i,s+1} := \sigma^0_{e,i}, \alpha_{1-i,s+1} := \alpha_{1-i,s}$ と定義して次のオステンシブルステージへ進む (リアルステージも次へ進む).

Subcase II (ii): このとき $\mathcal{G}_{e,i}$ の cycle(n) でターゲットを発見したものの内, 最大の n を選ぶ.

$z < k^n_{e,i}$ なる z について $f_{e,i}(z)$ が未定義ならば $f_{e,i}(z) := A_{s+1}(z)$ とし, τ がリアライズされるのを待つ. 更に cycle($n+1$) を生み出して以下の通り実行する.

(a) まず閾値 $k^{n+1}_{e,i}$ としてフレッシュな値を設定する. 特に $k^{n+1}_{e,i} > a_{s+2}$ とする. (注: $k^{n+1}_{e,i}$ は $k^n_{e,i}$ より更に大きい値である)
(b) $x \leq k^{n+1}_{e,i}$ なる全ての x について, $\Gamma^{G_0\oplus G_1}_{s+1}(x)$ と $\gamma_{s+1}(x)$ を Subcase II (i) と同様に定義する.
(c) $\sigma^{n+1}_{e,i} := \alpha_{i,s}{}^\frown\langle 0, \ldots, 0\rangle$ (但し $|\sigma^{n+1}_{e,i}| = \gamma_{s+1}(k^{n+1}_{e,i})$) と定義する.
(d) $\alpha_{i,s+1} := \sigma^{n+1}_{e,i}$, $\alpha_{1-i,s+1} := \alpha_{1-i,s}$ と定義して次のオステンシブルステージへ進む (リアルステージも次へ進む).

構成法は以上である.

$$G_i := \lim_{s\to\infty} \alpha_{i,s} \ (i = 0, 1).$$

と定めれば G_i こそが求める 1-ジェネリック集合となる.

6 基本的性質

このセクションでは構成を振り返ることで見える性質, 特にロールバックについての性質を述べる. まずは重要なステージの数について特別な表記を定義しておく.

定義 6.1. 1. 自然数 s について, 対応するオステンシブルステージが s であるリアルステージを $\varrho_0(s)$ と書く.

2. 自然数 s について, 対応するオステンシブルステージが s 以下である最大のリアルステージを $\varrho_1(s)$ と書く.
3. $r = \varrho_1(s)$ なるオステンシブルステージ s が存在するようなリアルステージ r を**ナイス**であると定義する.

性質 6.2. 以下に基本的な ϱ_i の性質をまとめておく.

(i) $t < s$ と $\varrho_0(s) < \varrho_0(t)$ は同値である.
これは $\varrho_0(s)$ が「最初にオステンシブルステージ s に辿り着いたようなリアルステージの値」であることを考えるとすぐに分かる.

(ii) $\varrho_1(s)$ は対応するオステンシブルステージが s であるようなリアルステージの内, 最大の値そのものである.
これはロールバックが発生してもリアルステージは進み続けることからすぐに分かる.

(iii) $t < s$ と $\varrho_1(s) < \varrho_1(t)$ は同値である.
これは (ii) より $\varrho_1(s)$ が「オステンシブルステージが s であるような最大のリアルステージ」であることから分かる.

(iv) オステンシブルステージが t となるようなリアルステージが一意的に存在すれば, $\varrho_0(t) = \varrho_1(t)$ である.
これは (i), (ii) よりすぐに分かる.

(v) Case IC のロールバックがロールバック出発地 $(r+1, s+1)$, ロールバック目的地 (r_0+1, t_0+1) で発生し, $r+1 = \varrho_1(t+1)$ なる t が存在する (i.e. $r+1$ はナイスである) とき, $r+1 = \varrho_1(t_0+1)$ とする. よって $r+1 \neq \varrho_1(s+1)$ であることに注意する.

以下の命題は定義 4.2 で与えた集合 H の性質についてである.

命題 6.3.

$$H := \{r_1+1 : \forall x \in A_{t_1+1}\text{に対し,「後に } A \text{ からドロップアウトする」マークが} \\ \text{リアルステージ } r_1+1 \text{ の終わりに付いていないものついて} \\ \forall r_2 > r_1, x \in A_{t_2+1}\text{が成立する.}\}$$

このとき以下が成立する.

1. H は Π^0_1 集合である.
2. r_1+1 はナイスであり, $\varrho_1(t_1+1) = r_1+1$ であるならば, $r_1+1 \in H$ である.
3. H は無限集合である.
4. $H \leq_T A$.

Proof. (1) A_{t_i+1} が有限集合ゆえ, $x \in A_{t_i+1}$ が計算可能であることと, 定義 2.17 より明らか.

(2) 背理法で示す. $\varrho_1(t_1+1) = r_1+1$ であり, $r_1+1 \notin H$ であるとする. すなわち, $x \in A_{t_1+1}$ かつ「後に A からドロップアウトする」マークが付いておらず, かつ $s+1 > t_1+1$ なる s について $x \notin A_{s+1}$ となる x が存在する. x を固定し, $r_2+1 > r_1+1$ かつ以下を満たすような最小の r_2 をとる.

$$t_2+1 > t_1+1, \quad x \notin A_{t_2+1} \text{ (但し } t_2 \text{は } r_2 \text{に対応するリアルステージとする).}$$

当然 r_2+1 で x はマークを付けられていないので, Case IC に分岐する. このときのロールバック出発地は (r_2+1, t_2+1) であり, 目的地を $(r_{0,2}+1, t_{0,2}+1)$ とする. また, $x \in A_{t_1+1}$ であることから, x がエニュメレイトされた瞬間があり, そのステージを (r_3+1, t_3+1) とする. するとロールバック目的地の定義から, $r_{0,2}+1 = r_3+1$, $t_{0,2}+1 = t_3+1$ となる. また $t_{0,2}$ の最小性により, $t_{0,2} \leq t_1$ である.

更に ϱ_1 の定義とロールバックでオステンシブルステージが $t_{0,2}$ へ戻ることから, $\varrho_1(t_1+1) \geq r_2+1$ である.

一方, $r_2+1 > r_1+1$ かつ $\varrho_1(t_1+1) = r_1+1$ であることから, $r_2+1 > \varrho_1(t_1+1)$ である. ゆえに $\varrho_1(t_1+1) \geq r_2+1 > \varrho_1(t_1+1)$ となり, 矛盾が導かれる.

(3) $x \in A$ であれば, 唯一の s について $x \in A_{s+1} - A_s$ が成立する. $r+1$ をオステンシブルステージが $s+1$ であるような最大のリアルステージであるとすると, $r+1 = \varrho_1(s+1)$ である. (2) より $r+1 \in H$ である. A が無限集合であるこ

とから, 無限に存在する x について, それぞれ唯一な $s+1$ が存在する. ゆえに $r+1=\varrho_1(s+1)$ なる $r+1$ は無限に存在する. 以上より H は無限集合である.

(4) H は以下の集合と等しい.

$$H' = \{r_1+1 : \forall x \in A_{t_1+1}\text{に対し,「後に } A \text{ からドロップアウトする」マークが} \\ \text{リアルステージ } r_1+1 \text{ の終わりに付いていないものついて} \\ x \in A \text{ が成立する.}\}$$

A をオラクルにすれば, $x \in A$ が計算可能であるため, $H = H' \leq_T A$ は明らか.

□

命題 6.4. r_0, r_1, r_2 をリアルステージとし, $r_0 < r_1 < r_2$ が成立しているとする. t_i+1 をそれぞれリアルステージ r_i+1 $(i \in \{0,1,2\})$ に対応するオステンシブルステージとする. 更に, Case IC が出発地 (r_2+1, t_2+1), 目的地 (r_0+1, t_0+1) で発生しているとする. このとき以下が成立する.

1. $a_{t_0+1} \in A_{t_1+1}$
2. $a_{t_0+1} \notin A$
3. $t_0 < t_1 < t_2$

Proof. (1) $a := a_{t_0+1}$ とする. Case IC の条件から, $a \in A_{t_0+1} - A_{t_0}$ かつ $a \in A_{t_2} - A_{t_2+1}$ である. 更に $r_1 < r_2$ であるので, リアルステージ r_1+1 では a は「後に A からドロップアウトする」マークが付けられていない.
ゆえに, もし $a \notin A_{t_1+1}$ であるとすると, r_1+1 より後で a はエニュメレイトされなければならない. すると, 目的地である r_0+1 は r_1+1 より大きくなり, 仮定に反する.

(2) A が 2-c.e. であることから, マインドチェンジは 2 回しか起こり得ない. ゆえに $r_0 < r_2$ なる r_0, r_2 について, a がリアルステージ r_0+1 でエニュメレイト, r_2+1 でドロップアウトした場合, A に属さない.

(3) (1) より, $t_0 \leq t_1 < t_2$ である. 仮に $t_0 = t_1$ であるとすると, 目的地 r_0+1 は r_1+1 と等しくなり矛盾が導かれる.

□

補題 6.5. r_0, r_1, r_2 を自然数, 特に $r_1 < r_2$ であるとする. t_i+1 をそれぞれリアルステージ r_i+1 $(i \in \{0,1,2\})$ に対応するオステンシブルステージとする. このとき Case IC が出発地 (r_2+1, t_2+1), 目的地 (r_0+1, r_0+1) で発生, かつ $r_1+1 \in H$ であるとする. こ

のとき $r_1 \leq r_0$ が成立する.

すなわち,「$r \in H$ なる r 以前のリアルステージを目的地としてロールバックが発生することはない」ということである.

Proof. 命題 6.4 の仮定と議論から, $x := a_{t_0+1}$ とすれば, x はリアルステージ $r_1 + 1$ でマークを付けられておらず, かつ $x \in A_{t_1+1}$ であり, 更に $x \notin A$ である. H の定義より, $r_1 + 1 \notin H$ である.
この補題は命題 6.4 の $r_0 < r_1$ という仮定のみを除いたものである. ゆえに以下が成立する.

$$r_0 < r_1 \to r_1 + 1 \notin H$$

よってこの対偶を考えることで,

$$r_1 + 1 \in H \to r_1 \leq r_0$$

□

命題 6.6. r_i, r'_i, t_i, t'_i は自然数で以下が成立しているものとする $(i \in \{0, 1\})$.

- r_1, r'_1 は偶数.
- $r_1 < r'_1$.
- Case IC が出発地 $(r_1 + 1, t_1 + 1)$, 目的地 $(r_0 + 1, t_0 + 1)$ で発生する.
- Case IC が出発地 $(r'_1 + 1, t'_1 + 1)$, 目的地 $(r'_0 + 1, t'_0 + 1)$ で発生する.

このとき以下が成立する.

$$r'_0 + 1 < r_0 + 1 \text{ または } r'_0 + 1 > r_1 + 1.$$

つまり 2 回続けて発生するロールバックにおいて, 2 回目の目的地のリアルステージは, 1 回目における目的地のリアルステージより前に戻る場合か, 1 回目における出発地のリアルステージより後に戻る場合しか有り得ないということである. グラフで表わすと以下のようになる.

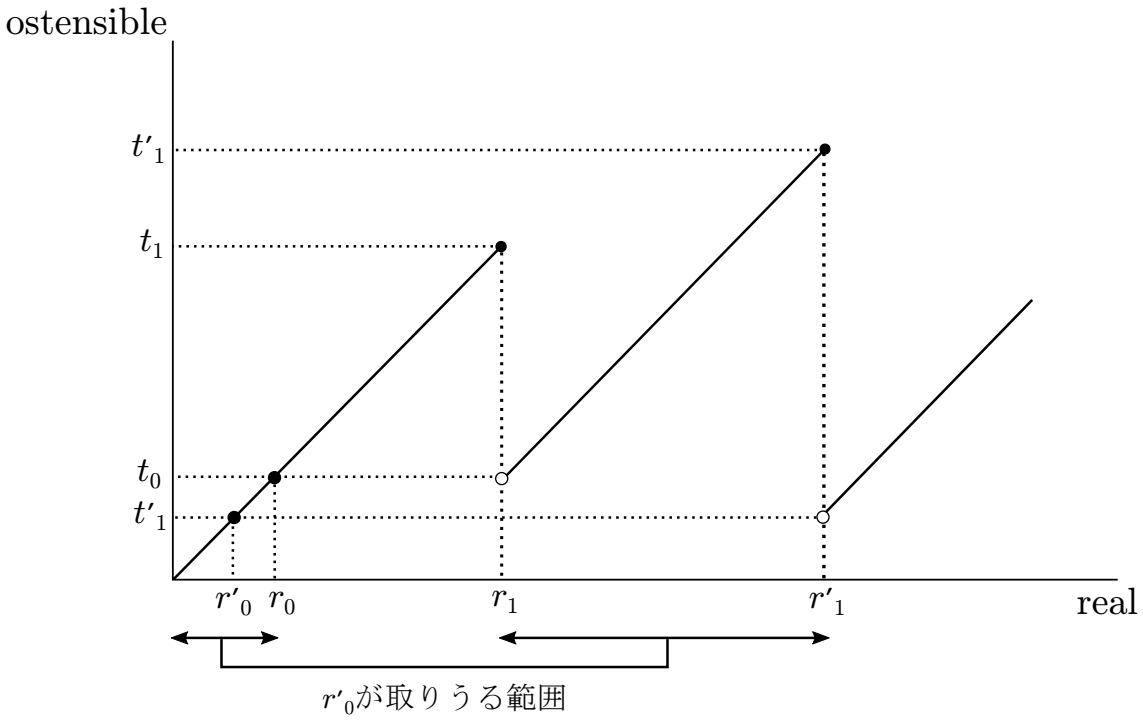

図 3　命題 6.6 で証明すべきこと

Proof. $r_1 < r'_1$ かつ r_1, r'_1 が偶数であることから, $r_1 + 2 < r'_1 + 1$ である. 背理法で示すため, 結論の否定 $r_0 + 1 \leq r'_0 + 1 \leq r_1 + 1$ を仮定する. これらの不等式より以下が成立する.

$$r_0 + 1 \leq r'_0 + 1 < r_1 + 2 < r'_1 + 1. \tag{1}$$

今 $a := a_{t'_0+1}$ とする. 命題 6.4 を利用するため, 命題 6.4 の $(r_0+1, t_0+1), (r_1+1, t_1+1), (r_2+1, t_2+1)$ を現在の $(r_0+1, t_0+1), (r'_0+1, t'_0+1), (r_1+1, t_1+1)$ と置き換えて考える.

命題 6.4 より $t_0 < t'_0$ である. それぞれ A のエニュメレイションが行われるステージのため偶数である. ゆえに $t_0 + 2 < t'_0 + 1$ が成立する.

一方この命題の仮定より, $t'_0 + 1$ は a がエニュメレイトされた最初のステージエニュメレイトである. ゆえに $a \notin A_{t_0+2}$ である. 構成の仕方から, リアルステージ $r_1 + 2$(最初のロールバック直後) でのオステンシブルステージは $t_0 + 2$ である. よって a はリアルステージ $r_1 + 2$ では A にエニュメレイトされていない.

更に, a はリアルステージ $r_1 + 2$ では「後に A からドロップアウトする」マークが付いていない. (なぜなら a がドロップアウトするリアルステージは仮定より $r'_1 + 1$ である.

ゆえにマークが付く最初のリアルステージは r'_1+1 であり, 式 (1) より $r_1+2<r'_1+1$ であるので, r_1+2 ではマークは付いていない.)

つまりリアルステージ r_1+2 でエニュメレイトされておらず, かつマークも付いていないような a がリアルステージ r'_1+1 で Case IC を引き起こすには, $r_1+2<r+1<r'_1+1$ なる $r+1$ で A にエニュメレイトされる必要がある.(i.e. $r+1$ が目的地になる.) 式 (1) より, $r'_0+1<r+1<r'_1+1$ となる. しかしこれは r'_0+1 が目的地であることに矛盾する. □

7 検証

7.1 $\mathcal{G}_{e,i}$ の検証

命題 7.1. r_1, r_0, t_1, n, e は以下の性質を満たす自然数とする.

- r_1 は偶数で, リアルステージ r_1 におけるオステンシブルステージは t_1.
- r_0+1 は, $r_0+1\leq r_1$ かつ仮ターゲット $\sigma^n_{e,i}$ が定義されるような最大のリアルステージ.
- リアルステージ r_1+1 で要求 $\mathcal{G}_{e,i}$ の cycle (n) は注目を受ける.

このとき以下が成立する.

$$\gamma_{t_1}(k^n_{e,i})=\gamma_{t_0+1}(k^n_{e,i})\ (t_i\text{は}r_i\text{におけるオステンシブルステージ}).$$

Proof. r_2 を $r_0+1<r_2+1\leq r_1$ なるリアルステージとし, そのオステンシブルステージを t_2 とする. 3 番目の仮定から $\mathcal{G}_{e,i}$ の cycle(n) は r_2+1 で初期化されることはない. つまり, リアルステージ r_2+1 では Case IB が起こったとすれば $a_{t_2+1}\geq k^n_{e,i}$ である. ゆえに r_2+1 では $\gamma(k^n_{e.i})$ の値に変化は起こらない.

Case IC が起こった場合を考える. 出発地を (r_2+1, t_2+1), 目的地を (r'_0+1, t'_0+1) とすると, やはり 3 番目の仮定から r_2+1 では $\sigma^n_{e,i}$ がキャンセルされていない, ゆえに $r'_0>r_0$. 従ってこの場合も, r_2+1 では $\gamma(k^n_{e.i})$ の値に変化は起こらない. □

補題 7.2. 任意の e, i について以下が成立する.

1. $\mathcal{G}_{e,i}$ が傷付けられるようなナイスリアルステージは高々有限回しかやってこない.
2. $\mathcal{G}_{e,i}$ が新しいサイクルを生み出すようなナイスリアルステージは高々有限回しかやってこない.

3. $\mathcal{G}_{e,i}$ が注目を要求するようなナイスリアルステージは高々有限回しかやってこない.
4. $\mathcal{G}_{e,i}$ は満足される. ゆえに G_i は 1-ジェネリック集合である.

Proof. 帰納法で証明する. 与えられた e に対し, $e' < e$ なる e' について (1)-(4) が成立しているとする. e についても, ナイスリアルステージに着目すれば (1)-(3) は補題 3.6 と同様に検証出来る. 以下 (4) について検証する.

まず以下を仮定する.

1. $r_1 + 1 > \varrho_1(s+1)$.
2. $t_1 + 1$ はリアルステージ $r_1 + 1$ のオステンシブルステージである.
3. $r_1 + 1$ は $\mathcal{G}_{e,i}$ が注目を要求する最後のリアルステージである.
4. $r_1 + 1$ はナイスとする. 言い換えれば, ある u が存在して $r_1 + 1 = \varrho_1(u+1)$ となることである.
5. リアルステージ $r_1 + 1$ では Case IB (i) が cycle (n) について発生し, α_{i,t_1+1} がターゲット $\tau^n_{e,i}[t_1]$ として定義される.

2. と 4. から $u = t_1$ であることが分かる. 命題 6.3 と 4. より $r_1 + 1 \in H$ である.

$r_1 + 1 \in H$ ということは, 今後ロールバック (Case IC) によって $(r_1 + 1, t_1 + 1)$ 以前の状態に戻ることがないということであり, 特にそのようなロールバックの出発地を (r', t'), 目的地 (r'_0, t'_0) とすれば, $r'_0 \geq r_1$ である.

ゆえにリアルステージ r' 以降も, $r_1 + 1$ で定義された α_{i,t_1+1} が Case IC によって壊されることはない. よって以下では Case IB についてのみ考察する.

Claim: $r_2 + 1 > r_1 + 1$ なるリアルステージ $r_2 + 1$ では Case IB が発生し, オステンシブルステージは $t_2 + 1$ であるとする. このとき x が α_{i,t_1+1} の定義域に属していれば, α_{i,t_2+1} における値に等しい.

Proof of claim. x が α_{i,t_1+1} に属しているとする. このとき以下の場合が考えられる.

(i) x は特に仮ターゲット $\sigma^n_{e,i}[t_1]$ の定義域に属しており, $x < \gamma_{t_1}(k^n_{e,i})$ である.
(ii) x は特にターゲットと仮ターゲットの定義域の差に属しており, $x \geq \gamma_{t_1}(k^n_{e,i})$ である.

また, t の選び方と $a_{t_2+1} \geq k^n_{e,i}$ であることから以下が成立する.

$$\gamma_{t_2}(a_{t_2+1}) \geq \gamma_{t_2}(k^n_{e,i}) \geq \gamma_{t_1}(k^n_{e,i}) \tag{2}$$

Case (i) であれば, $\gamma_{t_2}(a_{t_2+1}) > x$ であるので, $\alpha_{i,t_1+1}(x)$ の値は変更されない.

Case (ii) であれば, 構成の中で $\gamma_{t_2}(a_{t_2+1})$ はフレッシュな値をとるように定めていることから, $x \neq \gamma_{t_2}(a_{t_2+1})$ である. 加えて, 式 (2) より $\gamma_{t_2}(a_{t_2+1})$ の値はターゲットの定義域より大きい. 以上より, $\alpha_{i,t_1+1}(x)$ の値は変更されない.

Q.E.D. (Claim)

このことから, ジェネリック性の検証も補題 3.6 と似たような議論を進めればよいことがわかる. □

7.2 $\mathcal{P}$ の検証

補題 7.3. $G_0 \oplus G_1 \leq_T A$.

Proof. 命題 6.3 より $H \leq_T A$ であるので, $G_i \leq_T A \oplus H$ を示せば十分である.
与えられた x に対し, オラクル $A \oplus H$ を利用して, 以下の性質を満たすような (r_1, t_1) を探す.

- $r_1 + 1 \in H$.
- リアルステージ $r_1 + 1$ のオステンシブルステージは $t_1 + 1$.
- $A \upharpoonright (x+1) = A_{t_1+1} \upharpoonright (x+1)$.
- 任意の $i \in \{0,1\}$ について, $\alpha_{i,t_1+1}(x)\downarrow$.

任意の i について $\alpha_{i,t_1+1}(x) = G_i(x)$ であることを証明すればよい. ゆえに任意の $r \geq r_1$ について, 以下の (a), (b) どちらか一方が成立していることを証明する.

(a) リアルステージ $r+1$ の終わりにおける $\alpha_i(x)$ の値が $\alpha_{i,t_1+1}(x)$ の値に等しい.
(b) 以下の性質を満たすような $r_1^*, r_0^*, t_1^*, t_0^*$ が存在する.
  - Case IC がロールバック出発地 (r_1^*+1, t_1^*+1), ロールバック目的地 (r_0^*+1, t_0^*+1) で発生する.
  - $r_1 + 1 < r_0^* + 1 \leq r + 1 < r_1^*$, $t_1 + 1 \leq t_0^* + 1 \leq t + 1 < t_1^*$.
  - リアルステージ $r_1^* + 1$ の終わりにおける $\alpha_i(x)$ の値が $\alpha_{i,t_1+1}(x)$ の値に等しい.

証明には帰納法を使う. $r = r_1$ であれば (a) が成立しているのは自明. $r_2 > r_1$ なるリアルステージ r_2 をとり, $r_1 < r < r_2$ なる r については (a), (b) のいずれかが成立してい

ると仮定する.

まずはオラクル H を利用して $r_2+1 \in H$ かどうかを問う. $r_2+1 \in H$ であれば (b) は起こり得ないので (a) であることを示せばよい. しかしこれは補題 3.7 と全く同様に可能であり, $r_2+1 \in H$ の場合は (a) が成立していることがわかる.

$r_2+1 \notin H$ であれば今後, または直ちにロールバックが発生する. 特に r_2+1 が初めて目的地 (r_0^*+1, t_0^*+1) へのロールバックが発生したときの出発地であるとすると, $r_1+1 \in H$ であることから $r_1+1 < r_0^*+1$, $t_1+1 \leq t_0^*+1$ である. このときリアルステージ r_2+1 の終わりにおける $\alpha_i(x)$ の値は $\alpha_{i,t_0^*+1}(x)$ の値に等しい. よって仮定 (a) から $\alpha_{i,t_1+1}(x)$ とも等しいことがわかる. 2 回目以降のロールバックについても同様の議論が出来るので $r_2+1 \notin H$ の場合は (b) が成立していることがわかる. □

7.3 $\mathcal{C}$ の検証

命題 7.4. 与えられた x に対し, 有限個のリアルステージを除いて以下が成立している. 但し $t+1$ はリアルステージ $r+1$ に対応するオステンシブルステージとする.

$$\Gamma_{t+1}^{G_0 \oplus G_1}(x) \downarrow = A_{t+1}(x) = A(x).$$

Proof. 以下では $t+1$ をリアルステージ $r+1$ のオステンシブルステージであるとする.

構成方法から, いかなるリアルステージ $r+1$ においても, y が $\Gamma^{G_0 \oplus G_1}$ の定義域に属している限り以下が成立している.

$$\Gamma_{t+1}^{G_0 \oplus G_1}(y) = A(y) \text{ または } \Gamma_{t+1}^{G_0 \oplus G_1}(y) = A_{t+1}(y).$$

一方, 有限個のリアルステージを除いた $r+1$ では, $a_{t+1} > x$ のとき以下が成立している.

- $A_{t+1}(x) = A(x)$.
- $\Gamma_{t+1}^{G_0 \oplus G_1}(x) \simeq \Gamma_t^{G_0 \oplus G_1}(x)$. (左辺が定義されていることと右辺が定義されていることが等価, かつそのとき値が等しい)

特にこのようなリアルステージの中でも, Case IB が発生するようなステージ $r'+1$ を考える (オステンシブルステージは $t'+1$ とする). $a_{t'+1} > x$ ゆえに, $k_{e,i}^n > x$(e, i, n は注目を要求した $\mathcal{G}_{e,i}$の cycle(n) の値) であり, 従って x は $\Gamma_{t'+1}^{G_0 \oplus G_1}$ の定義域に属する.

以上のことから, 任意の $r \geq r'$ について以下が成立する.

$$\Gamma_{t+1}^{G_0 \oplus G_1}(x) = A(x) = A_{t+1}(x).$$

□

補題 7.5. $A \leq_T G_0 \oplus G_1 \oplus H$.

Proof. $\Gamma^{G_0 \oplus G_1}$ が全域関数であることはナイスであるリアルステージのみで考えれば, 補題 3.8 の議論がそのまま出来るので確認は容易.

$\Gamma^{G_0 \oplus G_1}$ が真に A を計算する関数であることは命題 7.4 で示した通り.

具体的な還元方法を以下に記述する. 任意の x について $A(x)$ の値を決定するためには以下のプロセスに従えばよい. 但しリアルステージ r に対応するオステンシブルステージを t とする.

まず $\gamma_t(x)\downarrow$ となるようなリアルステージ r を探す. オラクル H を利用して $r \in H$ かどうか問う. $r \notin H$ であれば最初に戻って更に大きな r を探す. $r \in H$ であれば次のステップへ進む.

G_0, G_1 をオラクルにして,

$$(G_{0,t} \oplus G_{1,t}) \upharpoonright_{2\gamma_s(x)} = G_0 \oplus G_1 \upharpoonright_{2\gamma_s(x)}$$

となるか否かをチェックする. 成立しなければ最初に戻って更に大きな r を探す. 成立したならば, そのリアルステージを r' とし, r' より大きいリアルステージ $r'' \in H$ について $\Gamma_{t''}^{G_0 \oplus G_1}(x)$ の値を出力すればその値こそが $\Gamma^{G_0 \oplus G_1}(x)$ の値である.

□

ジェネリック性と還元方法の正当性の検証は以上である. これで以下が証明されたことになる.

定理 7.6. A を真の 2-c.e. 集合とする. このとき 1-ジェネリック集合 G_0, G_1 が存在して以下が成立する.

$$A \equiv_T G_0 \oplus G_1 \oplus H.$$

ここで H に対して定理 3.2 を適用すると以下が証明されたことになる.

系 7.7. A を真の 2-c.e. 集合とする. このとき 1-ジェネリック集合 G_0, G_1, G_2, G_3 が存在して以下が成立する.

$$A \equiv_T G_0 \oplus G_1 \oplus G_2 \oplus G_3.$$

更にこのことから 2-c.e. 次数 $\mathbf{a}(\ni A)$ は 1-ジェネリック次数 $\mathbf{g_0}, \mathbf{g_1}, \mathbf{g_2}, \mathbf{g_3}$ に分解される.

$$\mathbf{a} = \mathbf{g_0} \cup \mathbf{g_1} \cup \mathbf{g_2} \cup \mathbf{g_3}.$$

8 まとめ

本論文の証明は, Wang[8] と Wu[9] で示されている定理の系の別証明になっている. 今後このロールバック付優先論法の考え方を別の場面で応用することで新たな結果が得られることが期待される.

謝辞

共同研究に際し, 多くのアドバイスとご指導戴いた鈴木登志雄准教授に厚く御礼申し上げます. 学部 4 年の頃より丁寧にご指導戴いたおかげで本論文を書き上げることができました.

共同研究者としても同研究室の先輩としても, 多くのアドバイスを戴いた水澤勇気氏にも厚く御礼申し上げます.

最後にお忙しい中本論文の副査を引き受けてくださった石谷謙介准教授, 内田幸寛准教授にこの場を借りて御礼申し上げます.

参考文献

[1] C.T.Chong, Liang Yu , “Measure-theoretic applications of higher Demuth’s theorem”, Trans. Amer. Math. Soc. 368 (2016), pp. 8249-8265.

[2] S. Barry Cooper , “Computability Theory”, Chapman and Hall/CRC (2002)

[3] S. Feferman, “Some applications of the notions of forcing and generic sets,” Fund. Math., vol. 56 (1965), pp. 325-345.

[4] P. G. Hinman, “Some applications of forcing to hierarchy problems in arithmetic,” Zeitschrift für Mathematische Logik und Grundlagen der Mathematik, vol. 15 (1969), pp. 341-352.

[5] C. G. Jockusch, Jr., “Degrees of generic sets,” pp. 110-139 in Recursion theory: its generalizations and applications, London Math. Soc. Lect. Note Series vol. 45, edited by F.R. Drake and S.S. Wainer, Cambridge University Press, Cambridge, 1980.

[6] André Nies , “Computability and randomness” , Oxford University Press (2012)

[7] Hartley Rogers, Jr. , “Theory of Recursive Functions and Effective Computability” , The MIT Press (1983)

[8] Wei Wang , “Relative enumerability and 1-genericity” , J. Symbolic Logic 76 (2011), no. 3, pp. 897-913.

[9] Guohua Wu , “1-Generic splitting of computably enumerable degrees” , Annals of Pure and Applied Logic 138 (2006), pp. 211-219.