数理解析研究所講究録
第2022巻 2017年 100-111

大学入試における数列の問題を解くための自動推論とその実装について

An automated deduction and its implementation for solving problems of sequence at university entrance examination

株式会社アトラス　和田　優未 (Yumi Wada)
Atlas Co., Ltd.
名古屋大学大学院;・工学研究科　松崎　拓也 (Takuya Matsuzaki)
Graduate School of Engineering,
Nagoya University
筑波大学・数理物質系　照井　章 (Akira Terui)
Faculty of Pure and Applied Sciences,
University of Tsukuba
国立情報学研究所・情報社会相関研究系　新井　紀子 (Noriko H. Arai)
Information and Society Research Division,
National Institute of Informatics

Abstract

"Todai Robot Project" is a project of artificial intelligence launched by National Institute of Informatics for re-unifying the artificial intelligence field subdivided in 1980 and afterwards. We focus towards attaining a high score in National Center Test for University Admissions, and use quantifier elimination over the real closed fields as a main tool for solving problems in mathematics. However, it is not applicable for problems with sequence of numbers. In this paper, we propose an algorithm for solving problems of sequences at the National Center Test for University Admissions.

1　はじめに

国立情報学研究所が中心となって進めている人工知能プロジェクト「ロボットは東大に入れるか」(以下，東ロボ) は，"総合的な知的タスクに取り組むことで，1980 年代以降，細分化された形で進展し一部では行き詰りも見えている人工知能各分野の成果を再統合すること" [3] を目的とし，2016 年度までに大学入試センター試験（以下単に，センター試験）で高得点をマークし，2021 年度までに東京大学入試を突破することを目標にしている．

東ロボでは各教科（国語，数学，英語，物理，日本史，世界史）ごとに入試問題の解答システムを開発している．数学の問題について実閉体の論理式に（書き下せることができれば）書き下した上で，量化子消去を行うことにより求解している（QE アルゴリズム）．しかし，数列に関する問題はこの手法では解くことのできない問題の 1 つである．したがって，数列に関する問題を求解するアルゴリズムを開発する必要がある．

関連する既存の研究として，(i) 漸化式（あるいは差分方程式）の求解や (ii) 帰納的定理の自動証明が挙げられる．(i) は，まず 1 つの数列に関する斉次な線形漸化式は線形代数の応用として必ず解（一般項）を求めることができる．また，非斉次なとき，非線形のとき，より一般には複数の数列に関する（連立）漸化式に対して，母関数を用いる等して漸化式ごとに工夫をし解を求めることが研究されている [1]．(ii) は，潜在帰納法や項書き換えによる自動証明等に対して，証明の失敗を減らすことや効率化が研究されている [2].

本稿では，センター試験における数列の問題（以下単に，数列の問題）を解くための自動推論のアイデアについて論ずる．センター試験の数列の問題では主に，漸化式と方程式からなる有限集合に対して，この集合に含まれる等式全てを満たす数列の一般項や変数の値を求める必要がある．(連立）漸化式を満たす数列の一般項，あるいは（連立）方程式を満たす変数の値を求めるアルゴリズムは従来研究され，その成果が主要な数式処理システム等にも実装されているが，これらを統合し数列と変数を同時に求めるようにしたことが従来とは異なっている．

本稿の流れは以下の通りである．第 2 章ではセンター試験における数学の問題のルールを述べる．第 3 章では筆者が開発した数列の問題のためのアルゴリズム達について述べ，これらが全て有限ステップで止まることを示す．

2 基本的な定義

正の整数を自然数と呼び，自然数全体を $\mathbb{N}$ で表す．集合 S の濃度を $\#S$ で表し，S の冪集合を $\mathfrak{P}(S)$ で表す．自然数から実数への 1 変数写像を数列と呼ぶ．m 個の未知の数列 $a_1, \ldots, a_m$ $(m \in \mathbb{N})$，非負整数 k と $m(k+1)+1$ 変数関数 f に対して，等式 $f(n, a_1(n), \ldots, a_1(n+k), \ldots, a_m(n), \ldots, a_m(n+k)) = 0$ を $a_1, \ldots, a_m$ に対する k 階漸化式と呼ぶ．

センター試験の数学の問題は，問題文中に現れる解答欄にあてはまる整数および英字を求める問題である．解答欄に英字が当てはまることはあまりないので，以下整数についてのみ述べる．解答欄には以下の性質がある:

1. 解答欄はカタカナで表され，その文字数は整数の桁を表している．

2. 同一のカタカナで表される解答欄には同一の整数があてはまる．

例えば，解答欄がアの場合は 1 桁の非負整数が，イウの場合は -9 以上 -1 以下の整数または 2 桁の正の整数があてはまる．

数列の問題は 2006 年から毎年，数学 IIB の第 3 問に出題される．また，制限時間があり，数学は 1 時間である．東ロボでは並列処理をすることを考え，大問ごとに制限時間一杯が与えられる．よって，数列の問題の制限時間は 1 時間である．

数列の問題の配点は 20 点である．

3 数列ソルバーについて

3.1 数列の問題の特徴

数列の問題を論理式に（言語処理ではなく人間が）書き換えた場合，以下の特徴を持つ論理式に書くことができる場合が多い:

1. 論理式に現れる関数は数列，四則演算，指数関数，数列の有限和，有限積 (Σ, Π)，数列の最大値，最小値 (max, min) である．ただし，
   (a) 数列の引数に 2 つ以上の変数は現れない．
   (b) Σ, Π は 2 変数関数であり，例えば $\Sigma(a,n)$ は $\sum_{k=1}^{n} a(k)$ を表す．
   (c) max, min は 3 変数関数であり，第 1 引数には数列を，第 2 引数には変数を，第 3 引数には原子論理式の論理積をとり，例えば $\max(a,n,n>1)$ は集合 $\{a(n) \mid n>1\}$ の最大値を表す．
2. 論理式に現れる述語は，$=$, $<$, $\leq$, $\neq$ と実数であることを表す R，自然数であることを表す N，数列であることを表す SEQ である．(R, N, SEQ はそれぞれ 1 変数述語記号である．)
3. 論理式に現れる論理記号は $\wedge$, $\rightarrow$, $\forall$ のみである．

例 1. センター試験 2007 年本試験第 3 問 (1).

> 数列 $\{a_n\}$ は，初項が -27 で，漸化式 $a_{n+1}=3a_n+60$ $(n=1,2,3,\ldots)$ を満たすとする．このとき $a_n=$ ア$^n-$ イウ である．数列 $\{a_n\}$ の初項から第 n 項までの和 S_n は $S_n=\frac{\text{エ}}{\text{オ}}(\text{カ}^n-\text{キ})-\text{イウ}\,n$ である．また，$S_n>0$ となる最小の自然数 n は ク である．

上の問題を論理式に書き換えると次のようになる:

$$\begin{aligned}\forall a(\mathrm{SEQ}(a) \rightarrow (a(1)=-27 \wedge \forall n(\mathrm{N}(n) \rightarrow a(n+1)=3a(n)+60)) \rightarrow \\ (\forall n(\mathrm{N}(n) \rightarrow a(n)=\text{ア}^n-\text{イウ}) \wedge (\forall n(\mathrm{N}(n) \rightarrow S(n)=\Sigma(a,n)) \rightarrow \\ (\forall n(\mathrm{N}(n) \rightarrow S(n)=\frac{\text{エ}}{\text{オ}}(\text{カ}^n-\text{キ})-\text{イウ}\,n) \wedge \text{ク}=\min(S,n,n>0))))).\end{aligned} \quad (1)$$

論理式 (1) は，上の特徴をすべて満たす．

このとき，

1. 量化記号は $\forall$ しか現れないので，量化記号は省略してもよい．
2. 関係記号 R, N, SEQ は量化子で縛られた変数の範囲を指定するためにのみ使われ，式の形から各変数が数なのか数列なのかは判断でき，自然数は数列の引数のみに現れるので，これらの関係記号は省略してよい．したがって実数，自然数，数

列を表す変数の加算集合をそれぞれ V_R, V_N, V_S としたとき，論理式に現れる変数は全て $V_R \cup V_N \cup V_S$ の元であるとしてよい．ここで 3 つの集合 V_R, V_N, V_S は互いに素な集合である．

3. 論理式 P, Q, R に対して $P \to (Q \wedge R) \equiv (P \to Q) \wedge (P \to R)$ および $P \to (Q \to R) \equiv (P \wedge Q) \to R$ が成り立つことを用いると，問題文は $\bigwedge_i (\bigwedge_j A_{i,j} \to \bigwedge_k C_{i,k})$ の形に書き換えることができる．ここで，$A_{i,j}$, $C_{i,k}$ は論理記号を含まない命題である．

例 2. 例 1 の論理式 (1) は次のように書き換えられる:

$$\begin{aligned}((a(1) = -27 \wedge a(n+1) = 3a(n) + 60) \to a(n) = \text{ア}^n - \text{イウ}) \\ \wedge ((a(1) = -27 \wedge a(n+1) = 3a(n) + 60 \wedge S(n) = \Sigma(a, n)) \to \\ (S(n) = \frac{\text{エ}}{\text{オ}}(\text{カ}^n - \text{キ}) - \text{イウ}\, n \wedge \text{ク} = \min(S, n, n > 0))). \quad (2)\end{aligned}$$

このとき，式 (2) のように書き換えられた論理式には，次の特徴がある場合が多い:

1. max，min の第 1 引数は単調な数列である．
2. $A_{i,j}$ の多くは等式であり，数列の一般項および変数の値を求めることができる．
3. $C_{i,k}$ は等式のみである．
4. もし $A_{i,j}$ に解答欄が含まれるならば，ある $i' < i$ と k が存在して $C_{k,i'}$ も同じ解答欄を含む．
5. 各 i に対して，$C_{i,k}$ に含まれる解答欄に当てはまる整数は，$\bigwedge_j A_{i,j}$ から求めた数列の一般項および変数の値を $C_{i,k}$ に代入することにより求めることができる．

したがって，各 i に対して，まず $\bigwedge_j A_{i,j}$ を満たす数列の一般項および変数の値を求め，解答欄を求めれば数列の問題を解くことができる場合が多い．

注意 1. 数列の問題では，解答者を答えへ誘導するために式の変形の際の途中式に解答欄が存在する場合がある．この場合，解答欄に当てはまる整数解の候補が複数存在することもある．これは解答の流れから推測しなければ解を 1 つに絞ることができないため，本ソルバーでは解の候補から 1 つ目を解として選ぶことにする．

3.2 記号処理について

問題文に言語処理を施して得られる ZF の論理式 φ からソルバーの入力を構成する．言語処理では，例えば “数列 $\{a_n\}$ の公差” を “common-diff-of(a)” と表すなど問題文をほぼ直訳するため，φ には上で述べた以外の関数，述語が現れる．したがって，まずそれらを上で述べた関数，述語に書き換える必要がある（書き換えの詳細は紙面の都合で省略する）．この書き換えにより得られる論理式を φ' とする．φ' は第 3.1 節で述べた条件を満たしている．

次に，φ' からソルバーの入力を構成する．ソルバーの入力には以下を用いる:

1. φ' に現れる V_R, V_S の元からなる集合 V, S,

2. φ' に現れる解答欄の集合 Z,

3. φ' を第 3.1 節で述べたように変形した時の $A_{i,j}$, $C_{i,k}$ たち.

例 3. 例 1 の論理式から上述の規則に従って構成したソルバーの入力は,

$$S = \{a, S\},\ V = \{\},\ Z = \{\ ア, イウ, エ, オ, カ, キ, ク\ \},$$
$$A_{1,1} \equiv a(1) = -27,\ A_{1,2} \equiv a(n+1) = 3a(n) + 60,\ C_{1,1} \equiv a(n) = ア^n - イウ,$$
$$A_{2,1} \equiv a(1) = -27,\ A_{2,2} \equiv a(n+1) = 3a(n) + 60,\ A_{2,3} \equiv S(n) = \Sigma(a, n)),$$
$$C_{2,1} \equiv S(n) = \frac{エ}{オ}(カ^n - キ) - イウ\, n,\ C_{2,2} \equiv ク = \min(S, n, n > 0)$$

である.

3.3 数式処理について

条件を満たす数列の一般項および変数の値を求めた後, 解答欄に当てはまる整数を求めることが主な方針であった. 例 2 にある数列の問題の特徴 5. を考慮して, 以下 i を固定して考える.

まず, $\{A_{i,j}\}_j$ から数列の一般項および変数の値を求めるアルゴリズムを考える. アルゴリズムの方針は次の通りである:

1. 漸化式を解き, 数列の一般項を求め,

2. 漸化式は恒等式とみなし方程式を解き, 変数の値を求める.

漸化式を解くアルゴリズムを rsolve とする. 具体的な rsolve の実装は各数式処理システムに委ねる. 例えば, Maple では `rsolve` という名で実装され, 数式処理システム Mathematica [8] では `RSolve` という名で実装されている.

本稿では, rsolve は第 1 引数に漸化式の集合を, 第 2 引数に対象となる数列を表す変数の集合をとり, 第 2 引数で指定された数列について漸化式を解くものとする. また, 解が見つからなかった場合は, そのことが判断できるとする. 例えば, Maple や Mathematica では解が見つからなかった場合, 入力されたアルゴリズム名とその引数がそのまま返されるため, 解が見つからなかったと判断できる.

漸化式の解には初期値の選び方による自由度がある. その未定定数には漸化式を解くたびに新たな変数を用意する. その変数の加算無限集合を Fst とする.

また, 方程式を解くアルゴリズムを solve とする. solve の具体的な実装も各数式処理システムに委ね, 本稿では第 1 引数に方程式の集合を, 第 2 引数に対象となる変数の集合をとり, 第 2 引数で指定された変数について方程式を解くものとする. 方程式の解に自由度が含まれていた時に, その未定定数には方程式を解くたびに新たな変数を用意する. その変数の加算無限集合を Fr とする. なお, 6 つの集合 V_R, V_N, V_S, Fst, Fr, Z はそれぞれ互いに素であるとする.

命題 A に対して，$I(A)$ を A に現れる変数全体から成る集合とし，変数の集合 B に対して，$I_B(A)$ を $I(A) \cap B$ で定義する．各式を解いた結果および解答欄の値は置換 R に記憶する．R は $S \cup V \cup Fst \cup Fr \cup Z$ のある部分集合から $\mathbb{R}^{\mathbb{N}} \cup \mathbb{R}$ への関数と見ることができる．また，命題 A に対し，$R(A)$ を A に現れる $I_{\mathrm{dom}(R)}(A)$ の各元 x を $R(x)$ に置き換えた論理式とする．(ここで $\mathrm{dom}(R)$ は関数 R の定義域である．) 例えば，ある漸化式の解が $a(n) = 2n$ であった場合，$R(a) = (n \mapsto 2n)$ とし，$A \equiv a(n+1) + b(n) = n$ のとき $R(A) \equiv 2(n+1) + b(n) = n$ となる．

アルゴリズム 1 (SUBSTITUTION).

入力: A: 命題，R: 置換;

出力: B: 次の条件を満たす命題: ある非負整数 k が存在して $R^k(A) = B$ かつ $R(B) = B$ が成り立つ;

```
1: function SUBSTITUTION(A, R)
2:     B ← A;
3:     while B ≠ R(B) do
4:         B ← R(B);
5:     end while
6:     return B;
7: end function
```

次の命題は自明である．

命題 1. アルゴリズムの入力である置換 R が次の条件を満たしているとする：(*) 任意の $x \in \mathrm{dom}(R)$ に対してある非負整数 k が存在し $R^k(x) = R^{k+1}(x)$ が成り立つ．このとき，アルゴリズム 1 は有限ステップで止まる．

以下，混乱の恐れがない限り命題 A と置換 R に対し $R(A)$ を SUBSTITUTION(A, R) の返り値とする．

アルゴリズム 2 (GETGENERALTERMS).

入力: $\mathcal{A}$: 漸化式の集合，S: 数列の集合，V: 変数の集合，R: 置換;

出力: $\mathcal{A}' \subset \mathcal{A}$: 一般項を求めるのに使われなかった漸化式の集合，$S' \subset S$: 一般項を求められなかった数列の集合，$V \cup F'$: $F' \subset F$ は数列の初項を表す変数，$R \cup R'$: R' は一般項を求めることができた S の元からその一般項への置換;

```
1: function GETGENERALTERMS(A, S, V, R)
2:     P ← 𝔓(S) \ {∅};
3:     P を濃度の小さい順に並べる;
4:     for P ∈ P do
5:         for A'' ∈ {A'' ⊂ A | ⋃_{A∈A''} I_S(R(A)) = P ∧ #A'' = #P} do
6:             if rsolve({R(A) | A ∈ A''}, P) が解 R' を求める then
7:                 V' ← ⋃_{A∈ran(R')} I_Fst(A);          ▷ ran(R') は R' の値域;
8:                 return GETGENERALTERMS(A \ A'', S \ P, V ∪ V', R ∪ R');
9:             end if
```

10: **end for**
11: **end for**
12: **return** $\mathcal{A}$, S, V, R;
13: **end function**

次の命題は容易にわかる.

命題 2. $\mathcal{A}$, S, V, R をアルゴリズム 2 の入力の条件を満たすものとし, $\mathcal{A}_r, S_r, V_r, R_r = \text{GetGeneralTerms}(\mathcal{A}, S, V, R)$ とする. また, $\mathcal{A}$, S は有限集合とする. さらに, R は命題 1 の条件 (*) を満たし, $(S \cup V) \cap \mathrm{dom}(R) = \emptyset$ が成り立つとする. このとき次が成り立つ:

1. R_r も条件 (*) を満たし, $(S_r \cup V_r) \cap \mathrm{dom}(R_r) = \emptyset$ が成り立つ,
2. $\#S_r \leq \#S$ である,
3. $S_r = S$ ならば $V_r = V$ である,
4. アルゴリズム 2 は有限ステップで止まる.

Proof. 2 は自明である.

1 を示す. 第 8 行において $S_{\mathrm{rec}} = S \setminus P$, $V_{\mathrm{rec}} = V \cup V'$, $R_{\mathrm{rec}} = R \cup R'$ とおき, R_{rec} が条件 (*) を満たし, $(S_{\mathrm{rec}} \cup V_{\mathrm{rec}}) \cap \mathrm{dom}(R_{\mathrm{rec}}) = \emptyset$ が成り立つことを示せば, 帰納的に 1 が成り立つ. まず, $\mathrm{dom}(R') = P$ と $V' \cap (\mathrm{dom}(R) \cup \mathrm{dom}(R')) = \emptyset$ であるので $(S_{\mathrm{rec}} \cup V_{\mathrm{rec}}) \cap \mathrm{dom}(R_{\mathrm{rec}}) = \emptyset$ が成り立つ. また, $I(R(A)) \cap \mathrm{dom}(R) = \emptyset$ より $\mathrm{ran}(R')$ には $\mathrm{dom}(R)$ の元は現れない. R' は明らかに条件 (*) を満たすので R_{rec} も条件 (*) を満たす.

3 は対偶を示す. $V_r \neq V$ となるとき, 第 8 行は少なくとも 1 回は実行される. したがって, $\#S_r < \#S$ であるので $S_r \neq S$ となる.

4 を示す. 1 と命題 1 から各 Substitution は有限ステップで止まる. アルゴリズム 2 の各 for 文は現れる集合が全て有限集合なので有限回で終わる. また, 再帰的にアルゴリズム 2 を呼び出しているが, P は空集合ではないのでこの再帰も有限回で終わる. □

アルゴリズム 2 は数列が 1 個しか現れない漸化式から順に求解を試みる. もし解が存在したならばその解を R に加え, 別の漸化式が解けるか試みる.

アルゴリズム 3 (GetValues).
入力: $\mathcal{A}$: 等式の集合, $\mathcal{A}_{\mathrm{IN}}$: 不等式の集合, S: 数列の集合, V: 変数の集合, R: 置換;
出力: $\mathcal{A}' \subset \mathcal{A}$: 変数の値を求めるのに使われなかった等式の集合, $V' \subset V$: 値が求められなかった変数の集合, $R \cup R'$: R' は値を求めることができた V の元からその値への置換;
1: **function** GetValues($\mathcal{A}, \mathcal{A}_{\mathrm{IN}}, S, V, R$)
2: $\mathcal{P} \leftarrow \mathfrak{P}(V) \setminus \{\emptyset\}$;

3: $\mathcal{P}$ を濃度の小さい順に並べる;
4: **for** $P \in \mathcal{P}$ **do**
5: **for** $\mathcal{A}'' \in \{\mathcal{A}'' \subset \mathcal{A} \mid \bigcup_{A\in\mathcal{A}''} I_V(R(A)) = P \wedge \bigcup_{A\in\mathcal{A}''} I_S(R(A)) = \emptyset\}$ **do**
6: **if** solve($\{R(A) \mid A \in \mathcal{A}''\}$, P) が k 個の解 $R'_1, \ldots, R'_k$ を求める ($k \in \mathbb{N}$) **then**
7: **for** $R' \in \{R'_1, \ldots, R'_k\}$ **do**
8: $R'' \leftarrow R \cup R'$;
9: **if** $\{R''(A) \mid A \in \mathcal{A}_{\mathrm{IN}}\}$ に false が存在しない **then**
10: $V' \leftarrow \bigcup_{A\in \mathrm{ran}(R')} I_{Fr}(A)$;
11: **return** GETVALUES($\mathcal{A} \setminus \mathcal{A}'', \mathcal{A}_{\mathrm{IN}}, S, (V \cup V') \setminus P, R''$);
12: **end if**
13: **end for**
14: **end if**
15: **end for**
16: **end for**
17: **return** $\mathcal{A}$, V, R;
18: **end function**

次の命題は命題 2 と同様にして示せる.

命題 3. $\mathcal{A}$, $\mathcal{A}_{\mathrm{IN}}$, S, V, R をアルゴリズム 3 の入力の条件を満たすものとし, GETVALUES$(\mathcal{A}, \mathcal{A}_{\mathrm{IN}}, S, V, R)$ の返り値を $\mathcal{A}_r, V_r, R_r$ とする. また, $\mathcal{A}$, V は有限集合とする. さらに, R は命題 1 の条件 (*) を満たし, $(S \cup V) \cap \mathrm{dom}(R) = \emptyset$ が成り立つとする. このとき次が成り立つ:

1. R_r も条件 (*) を満たし, $(S \cup V_r) \cap \mathrm{dom}(R) = \emptyset$ が成り立つ.
2. $\#V_r \leq \#V$ である.
3. アルゴリズム 3 は有限ステップで止まる.

Proof. 1 は命題 2.1 と同様にして分かる. 3 を示す. 各 SUBSTITUTION と各 for 文が有限ステップで止まることは命題 2 と同様である. また, 再帰的にアルゴリズム 3 を呼び出しているが, V' の取り方から $\#V' < \#P$ となるのでこの再帰も有限ステップで終わる. 同じ理由で, 2 も成り立つ. □

アルゴリズム 3 では方程式の解に自由度があることを許す. これは, 数列の問題には全ての変数の値が定まるとは限らない問題が存在するためである. また, アルゴリズム 3 はアルゴリズム 2 とほとんど同様だが, 不等式で解の評価をしている部分が異なる. 不等式での評価は, 方程式の解に自由度があるために必ずしも正確に行えるとは限らない. また, $\mathcal{A}''$ に漸化式 A があるときは, $I(A) \setminus (S \cup V \cup Fst \cup Fr \cup Z)$ の元に関する恒等式とみる.

アルゴリズム 2, 3 を組み合わせることにより, $\{A_{i,j}\}_j$ から数列の一般項および変数の値を求めるアルゴリズムを作れる:

アルゴリズム 4 (GETGENERALTERMSANDVALUES).
入力: $\mathcal{A} = \{A_{i,j}\}_j$: 条件の集合, S: 数列の集合, V: 変数の集合, R: Z のある部分集合から $\mathbb{Z}$ への置換;
出力: $R \cup R'$: R': $\{A_{i,j}\}_j$ から求めることができた $S \cup V$ の元からその値への置換;

```
1: function GETGENERALTERMSANDVALUES(A, S, V, R)
2:     A_IN ← {A ∈ A | A : 不等式 };
3:     A' ← A \ A_IN; S' ← S; V' ← V; R'' ← R;
4:     repeat
5:         A_RE ← {A ∈ A' | A : 漸化式 };
6:         A' ← A' \ A_RE; S_tmp ← S';
7:         A_RE, S', V', R'' ← GETGENERALTERMS(A_RE, S', V', R'');
8:         changed ← S_tmp ≠ S';
9:         A' ← A_RE ∪ A'; V_tmp ← V';
10:        A', V', R'' ← GETVALUES(A', A_IN, S', V', R'');
11:        changed ← changed ∨ V_tmp ≠ V';
12:    until changed = false
13:    return R'';
14: end function
```

ここで, 入力にある R は $A_{i,j}$ に解答欄 z がある場合に必要である. すなわち, このとき数列の問題の性質から, ある $i' < i$ が存在して $C_{k,i'}$ も z を含み, もし z の値が求められているのならば, その値を $A_{i,j}$ でも利用する必要がある.

命題 4. $\mathcal{A}$, S, V, R をアルゴリズム 4 の入力の条件を満たすものとし, S, V は有限集合とする. このときアルゴリズム 4 は有限ステップで止まる.

Proof. R は明らかに命題 1 の条件 (*) を満たすので, 命題 2, 3 より各 GETGENERALTERMS, GETVALUES は有限ステップで止まる. 命題 2.2 より, repeat 文を十分繰り返すとステップ 3-4 において $S_{\text{tmp}} \neq S'$ は常に false となる. このとき, 命題 2.3 よりステップ 3-3 において V' は変化しない. したがって命題 3.2 より, さらに十分繰り返すとステップ 3-7 において $V_{\text{tmp}} \neq V'$ は常に false となる. このとき, repeat 文は終了する. □

次の定理は自明である.

定理 1. $\mathcal{A}$, S, V, R をアルゴリズム 4 の入力の条件を満たすものとし, R_r をアルゴリズム 4 の出力, すなわち R_r = GETGENERALTERMSANDVALUES$(\mathcal{A}, S, V, R)$ とする. さらに, $\mathcal{A}' = \{A \in \mathcal{A} \mid A : \text{等式}\}$ とする. このとき, $\bigwedge_{x \in \mathrm{dom}(R^r)}(x = R_r(x)) \wedge \bigwedge_{A \in \mathcal{A}'} R_r(A)$ ならば $\bigwedge_{A \in \mathcal{A}'} R(A)$ が成り立つ. すなわち, R_r は $\mathcal{A}'$ を全て満たす数列の一般項と変数の値 (の一部) である.

次に, 上で求めた R を用いて, $\{C_{i,k}\}_k$ に現れる解答欄にあてはまる整数を求める. 原始的な方法は, 解答欄にあてはまる整数について総当たりすることである. 解答欄の

桁数が分かっているので可能であるが，各 k に対して $I_Z(C_{i,k})$ にあてはまる可能性のある整数の組み合わせは 10^9 通り以上になることもあり，制限時間 1 時間以内に解を求められない可能性がある．したがって工夫する必要がある．

各 $z \in Z$ に対して $n(z) \subset \mathbb{Z}$ を z にあてはまる可能性のある整数からなる集合とする．例えば，$z =$ ア のときは，$n(z) = \{0, \ldots, 9\}$ であり，$z =$ アイ のときは，$n(z) = \{-9, \ldots, -1, 10, \ldots, 99\}$ である．また，各等式 A に対して，$I^e(A) \subset Z$ を A に現れる指数関数の肩に現れる Z の元からなる集合とする．

アルゴリズム 5 (IntSolve).

入力: R: 置換，C: 次を満たす等式: $I^e(R(C)) = \emptyset$;

出力: R': C の解を見つけることができた時はその解を表す置換，そうでないときは空集合;

```
1: function IntSolve(R, C)
2:     Z' ← I_Z(R(C));
3:     R ← {R' : Z' → ℝ | R' : R(C) をZ'の元以外の変数に関する恒等式と見た時の解};
4:     R ← {R' ∈ R | ∀z ∈ Z', R'(z) ∈ n(z)};
5:     if R ≠ ∅ then
6:         R' ∈ R を 1 つとる;
7:     else
8:         R' ← ∅;
9:     end if
10:    return R';
11: end function
```

アルゴリズム 5 はアルゴリズム 6 のサブルーチンである．

アルゴリズム 6 (FindAnswers).

入力: $\mathcal{C} = \{C_{i,k}\}_k$: 等式の集合，$R$: 置換;

出力: $\mathcal{C}'$: 解を得られなかった等式の集合，$R \cup R'$: R': 解を求められた解答欄からその値への置換;

```
1: function FindAnswers(C, R)
2:     C' ← ∅; R'' ← R;
3:     for C ∈ C do
4:         Z' ← I_Z(R(C));
5:         if Z' ≠ ∅ then
6:             Z^e ← I^e(R(C));
7:             if Z^e = ∅ then
8:                 P ← ∅; Q ← IntSolve(C, R'');
9:             else
10:                for P ∈ ∏_{z∈Z^e} n(z) do
11:                    Q ← IntSolve(C, R'' ∪ P);
12:                    if Z' = Z^e ∨ Q ≠ ∅ then
```

```
13:                     break;
14:                 else
15:                     P ← ∅;
16:                 end if
17:             end for
18:         end if
19:         if Q ∪ P = ∅ then
20:             C′ ← C′ ∪ {C};
21:         else
22:             R″ ← R″ ∪ P ∪ Q;
23:         end if
24:     end if
25: end for
26: return C′, R″;
27: end function
```

アルゴリズム 6 は $\mathcal{C}$ が有限ならば明らかに有限ステップで止まる．このアルゴリズムは総当たりで解を見つけることができる等式に対して必ずしも解を求められるとは限らないことに注意する．したがって，最終的には総当たりで解を求める必要がある．

最後に以上のアルゴリズムを組み合わせることで，数列ソルバーを構成する．

アルゴリズム 7 (CenterSeqSolver)**.**

入力: $\{(\mathcal{A}_i, \mathcal{C}_i)\}_i$: 第 3.1 節で得た条件の集合，$S$: 数列を表す変数の集合，$V$: 数列ではない変数の集合;

出力: R: Z のある部分集合から $\mathbb{Z}$ への置換;

```
1: function CenterSeqSolver({(A_i, C_i)}_i, S, V)
2:     R ← ∅; unsolved ← ∅;
3:     for (A, C) ∈ {(A_i, C_i)}_i do
4:         R ← GetGeneralTermsAndValues(A, S, V, R);
5:         C′, R ← FindAnswers(C, R);
6:         if C′ ≠ ∅ then
7:             unsolved ← unsolved ∪ {(R, C′)};
8:         end if
9:         R ← R ∩ (Z × ℤ);
10:    end for
11:    for (R′, C′) ∈ unsolved do
12:        for C ∈ C′ do
13:            R″ ← R ∪ R′;
14:            C ← R″(C);
15:            if I_Z(C) ≠ ∅ then
16:                solves ← {P ∈ ∏_{z∈I_Z(C)} n(z) | P(C) = true};
17:                if solves ≠ ∅ then
```

```
18:                 P ∈ solves を 1 つとる;
19:                 R ← R ∪ P;
20:             end if
21:         end if
22:     end for
23: end for
24: return R;
25: end function
```

アルゴリズム 7 においてステップ 4 ではアルゴリズム 6 で解を求められなかった等式に対して総当たりで解を求めている.

参考文献

[1] Agarwal, R.P. *Difference Equations and Inequalities: Theory, Methods, and Applications*, 2nd Ed., Marcel Dekker, New York, Basel, 2000.

[2] Alan Bundy. *The Automation of Proof by Mathematical Induction.* In Handbook of Automated Reasoning (A. Robinson and A. Voronkov, Eds., 2nd Ed.), North-Holland, Amsterdam, 845–911, 2001.

[3] 新井紀子, 松崎拓也. ロボットは東大に入れるのか? — 国立情報学研究所「人工頭脳」プロジェクト —, 人工知能学会誌, Vol. 27, No. 5, 463–469, 2012.

[4] E. Luznica and M. Kohlhase. Formula Semantification and Automated Relation Finding in the On-Line Encyclopedia for Integer Sequences. Proceedings of the 5th International Congress on Mathematical Software (ICMS 2016). Lecture Notes in Computer Science 9725, Springer, 467–475, 2016.

[5] Maplesoft, a division of Waterloo Maple Inc. Maple 18 [Computer software]. Waterloo, Ontario, Canada.

[6] 松崎拓也, 岩根秀直, 穴井宏和, 相澤彰子, 新井紀子. 深い言語理解と数式処理の接合による入試数学問題解答システム. 人工知能学会全国大会, 2013.

[7] The OEIS Foundation Inc., The On-Line Encyclopedia of Integer Sequences, `https://oeis.org/`, 最終アクセス 2016 年 10 月 28 日.

[8] Wolfram Research, Inc., Mathematica [Computer software]. Champaign, Illinois, USA.