The 32nd Annual Conference of the Japanese Society for Artificial Intelligence, 2018

センター試験「数学」整数問題解答システムにおける探索空間の削減

Search Space Reduction in Automatically Solving Elementary Number Theory Problems in National Center Test

犬塚 慎也 *1　松崎 拓也 *1　佐藤 理史 *1
Shinya Inuzuka　Takuya Matsuzaki　Satoshi Sato

*1名古屋大学大学院 工学研究科 情報・通信工学専攻
Graduate School of Engineering, Nagoya University

We have been developing a number problem solver for National Center Test for University Admissions (NCTUA). The solver searches for a series of equivalence-preserving transformation of an input formula that results in a quantifier-free formula, from which the answer can be easily deduced. However, the increase of the search space hampered the capability of the solver. We introduced formula normalization and extended matching rules in the search in order to reduce the search space. Experimental results show that this method drastically reduces the search space and shorten the execution time.

1. 序論

本研究では，「ロボットは東大に入れるか」 (略称：東ロボ) プロジェクト [新井 12] の一環として開発されたセンター試験「数学 I・A」整数問題の自動解答システム [犬塚 17] における探索空間の削減に取り組んだ．センター試験の問題は図 1 に示すようにマーク式であり，論理式に翻訳すると空欄部分のみを自由変数とする論理式になる．そのため，このような問題を解くことは，論理式から限量子を除去し，自由変数のみからなる方程式・不等式を解く処理だと見なせる．

東ロボ数学解答システムのうち幾何や実代数の問題に対する処理では演繹に実閉体上の限量子除去を応用している [岩根 13]．しかし，整数領域においては限定的な体系の式を除いて限量子の除去は一般には不可能であり，これを行うアルゴリズムは存在しない．そのため，整数問題に対する解答システムにおいては問題の論理表現方法の工夫や逐次的な式変形探索を通じた発見的な方法による限量子の除去を行っている．

しかし，式変形探索においては探索空間の増大という課題があり，これを原因として解答が不能となる場合がある．この課題を解決するために，論理式を正規化して保持し，論理式と式変形規則とのマッチングの際にはこの正規化された式に対してマッチングを行うことを前提とする．また交換律や対称律を考慮するようにマッチング規則を拡張し，探索空間を削減することを試みた．

人手で作成した論理表現を入力とした評価実験の結果、本研究で改良を加えた探索手法は，得点率を低下させることなく探索空間を大幅に減少させられることが確認できた．さらに，解答システム全体については実行時間の短縮が確認できた．

2. システム概要

整数問題解答システム [犬塚 17] の構成を図 2 に示す．演繹部の入力は，一階述語論理式に特殊な述語を加えた体系の式である．これに数式処理を施し，解答を出力する．

演繹部においては、入力された問題はまずドライバによって複数の小問題に分割され、それぞれが整数問題用ソルバー (Heuristic Solver) に入力される．Heuristic Solver では，入力に対して限量子の除去および述語・関数に対応する計算処理を Algorithm 1 の通り行う．Algorithm 2 に示す限量子除去手続きや述語部分の処理においては，必要に応じて式変形探索による限量子除去 (Algorithm 3) が試みられるほか，プレスバーガー算術に対する限量子除去アルゴリズムの実装である Princess[Rummer 08] および数式処理システム Mathematica をモジュールとして用いる．処理が終了した式はドライバに返され，解答欄の桁数の条件を加えて解が求められる．なお，言語処理部との接続を念頭に，本稿では演繹部の入力を「中間表現」と呼ぶ．述語・関数の処理の詳細は [犬塚 17] に準ずる．

この構成のシステムについて，以降では 1 章で触れたマッチングの拡張を行う前のものと後のものをそれぞれ「旧システム」「新システム」と呼ぶ．

連絡先: 犬塚慎也, 名古屋大学大学院工学研究科 情報・通信工学専攻，名古屋市千種区不老町 C3-1(631)，052-789-4435，s_inuzuk@nuee.nagoya-u.ac.jp

k, l を自然数とし、$m = 6k$, $m + 56 = 44l$ と表すと、k, l について
一次不定方程式 $\boxed{\text{ケ}}(k + \boxed{\text{コ}}) + 1 = \boxed{\text{サシ}}\, l$ が成り立つ。

↓（言語処理による論理式への変換）

$$\forall k \forall l \forall m\big((m = 6k \wedge m + 56 = 44l) \rightarrow \boxed{\text{ケ}}(k + \boxed{\text{コ}}) + 1 = \boxed{\text{サシ}}\, l\big)$$

↓（限量子除去）

$$22\boxed{\text{ケ}} - 3\boxed{\text{サシ}} = 0 \wedge -2\boxed{\text{ケ}} + \boxed{\text{ケ}}\boxed{\text{コ}} + 1 - \boxed{\text{サシ}} = 0$$

↓（簡約）

$$\boxed{\text{ケ}} = 3 \wedge \boxed{\text{コ}} = 9 \wedge \boxed{\text{サシ}} = 22$$

図 1: 整数問題とその計算処理の例

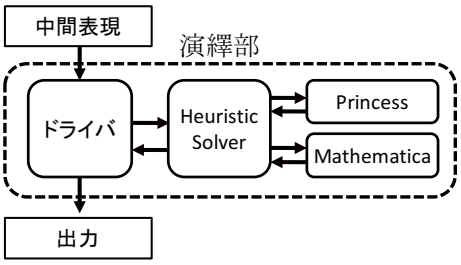

図 2: 整数問題解答システムの構成

Algorithm 1 整数問題用ソルバーが行う処理の全体像

```
procedure SOLVE(φ)                                  ▷ φ から限量子を除去する
    for all φ_i (φ の部分式) do
        φ_i ← SOLVE(φ_i)                            ▷ 部分式から処理を行う
    end for
    if φ に限量子が含まれている then
        φ ← REDUCE(φ)                               ▷ 限量子の除去を試みる
    else if φ に専用の述語が含まれている then
        φ ← 専用述語処理手続 (φ)                    ▷ 専用の述語の処理を試みる
    end if
    return φ
end procedure
```

Algorithm 2 限量子除去手続き

```
procedure REDUCE(φ)                                 ▷ φ から限量子を除去する
    if φ がプレスバーガー算術の式 then
        return PRINCESS(φ)                          ▷ Princess により限量子除去を試みる
    end if
    φ ← GRAPHSEARCH(φ)                              ▷ グラフ探索により限量子除去を試みる
    if φ が限量子を含む then
        φ ← MATH_REDUCE(φ)
                                                    ▷ Mathematica により限量子除去を試みる
    end if
    return φ
end procedure
```

Algorithm 3 式変形探索による限量子除去

```
rules ← (式変形規則の集合)
procedure GRAPHSEARCH(φ)                            ▷ 探索により φ から限量子を除去する
    try
        nodes ← {φ}                                 ▷ 探索ノード
        while nodes に未処理の式が存在 do
            φ ← (nodes のうち未処理のもの)
            for all φ_i (φ の部分式) do
                GRAPHSEARCHSUB(φ_i)
            end for
        end while
    catch SearchEnded, SearchStopped
        return (nodes のうち限量子のネストの深さが最も浅い式)
    end try
    return (nodes のうち限量子のネストの深さが最も浅い式)
end procedure
procedure GRAPHSEARCHSUB(φ)                         ▷ サブルーチン
    for all rule ∈ rules do
        φ* ←REWRITE(φ, rule)                        ▷ 式の書き換えを試行する
        if φ* ≠ φ then nodes ← nodes ∪ {φ* を部分式とする論理式 }
            if φ* を部分式とする論理式が限量子を含まない then
                throw SearchEnded
            end if
            if |nodes| ≥ Threshold then
                throw SearchStopped
            end if
        end if
    end for
    return φ
end procedure
```

3. 正規化とマッチング規則の拡張による探索空間の削減

整数問題解答システムの式変形探索における探索空間増大の原因を明らかにするため，開発用の問題 14 問分について，各式変形規則の適用回数を調べたところ，図 3 の通りであった．これにより，本質的に重要な式変形は適用回数が少なく，一方で可換性や対称性を吸収するための簡単な式変形規則が式変形試行の大部分を占めており，探索空間の増大につながっていることが判明した．

旧システムでは，探索時に論理式が式変形規則の左辺と一致するかどうかを確かめるマッチングを行うが，これは純粋な First order matching であった．このマッチングでは乗算の可換性や等号の対称性を考慮できないため，たとえば乗算の可換性 $a \times b = b \times a$ のような単純な法則についても，それらの法則自体を式変形規則に加えていた．探索の際には，たとえば可換性を持つ演算が出現するすべての箇所に関して，上記のような規則によって交換した結果をすべて生成し，それぞれが探索空間上のノードとなる．したがって，この探索手法では，入力となる中間表現が長く，可換性や対称性を考慮しなければならない箇所が多いほど探索空間の増大が顕著になる．

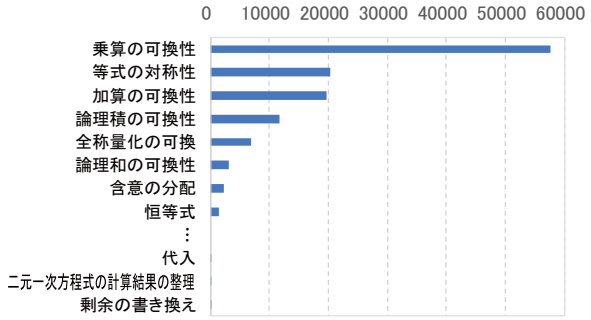

図 3: 旧システムにおける式書き換え規則の適用数

これを受けて，式の正規化とマッチング規則の拡張を組み合わせることによる探索空間の削減を試みた．この式変形探索では，式を書き換える規則を，表 1 に示す例のような「正規化規則」と，表 2 に示す「式変形規則」の 2 つに分け，式変形の探索やノードの生成は表 2 の式変形規則のみを用いて行う．各ノードの式は，正規化規則を再帰的に適用することによって正規化された形で保持される．正規化規則には，表 1 に示したもの以外に，$a - b \Leftrightarrow a + (-b)$ のような算術演算子に関する規則，$a = b \Leftrightarrow a - b = 0$ のように演算子や等号・不等号に関する規則，$\phi_1 \to (\phi_2 \to \phi_3) \Leftrightarrow (\phi_1 \wedge \phi_2) \to \phi_3$ のように論理演算子に関する規則，$\forall x \forall y \phi \Leftrightarrow \forall x, y \phi$ のように量化子に関する規則など，合計で 41 種類の規則がある．また，算術演算子や論理演算子については，$\phi_1 \wedge (\phi_2 \wedge \phi_3)$ のような結合律を満たす演算子の入れ子構造を $\phi_1 \wedge \phi_2 \wedge \phi_3$ のように順序を区別しないで保持するようにする．正規化規則の中には，複数の剰余の条件を 1 つにまとめる表中 3 番の規則や，有界な束縛変数に関する展開を行う 9 番の規則のように，演繹的な役割を持った規則も含まれているが，正規化規則を適用しても新たなノードは生成されない．

探索時のマッチングは，正規化された式に対して交換律や対称律を考慮するように拡張する．たとえば等号の対称律は，論理式に含まれる等式の右辺が 0 になるように正規化し，また式変形規則に含まれる等式も右辺が 0 になる形で作成することで，マッチング時には左辺と右辺の入れ替えを考慮する必要をなくす．また，加算や乗算の交換律は，多項式を展開し，束縛変数について同類項をまとめて整理した形で保持する．また式変形規則も束縛変数について整理された形で作成することで，その多項式の項について考えられる順列のうち，パターンとマッチするものの一つを採用する．ただし，パターンの中に束縛変数が複数含まれる場合においては，束縛変数について対称となるようにパターンを構成しているため，探索漏れは生じない．

旧システムの場合と同様に，探索では部分式に適用可能な式変形規則を再帰的に適用し，幅優先探索を行うことで限量子の除去を試みる．限量子の除去が完了したか，あるいは試行回数

表 1: 正規化規則の例

番号	変形前 *1	変形後 *1
1 *2	$(i_1 i_2 x + i_3) \bmod i_1$	$i_3 \bmod i_1$
2 *3	$(i_1 i_2 x + i_3) \operatorname{div} i_1$	$i_2 x + (i_3 \operatorname{div} i_1)$
3 *4	$\bigwedge_k -i_k + f \bmod j_k = 0$	$-i_{new} + f \bmod j_{new} = 0$
4	$i_1 x + i_2 > 0$	$\begin{cases} x > \lfloor \frac{i_2}{i_1} \rfloor & (i_1 > 0) \\ x < \lceil \frac{i_2}{i_1} \rceil & (otherwise) \end{cases}$
5	$\bigwedge_{k=1}^{j} -i_k + x > 0$	$-\max\{i_1, i_2, \cdots, i_j\} + x > 0$
6	$\bigwedge_{k=1}^{j} i_k - x > 0$	$\min\{i_1, i_2, \cdots, i_j\} - x > 0$
7	$\forall x$(x を含まない論理式)	(x を含まない論理式)
8	$\exists x$(自由変数を含まない論理式)	$\begin{cases} \top & (\text{論理式を充足する } x \text{ が存在するとき}) \\ \bot & (otherwise) \end{cases}$
9	$\forall x_1, \cdots, x_n \Big[\Big\{ \phi(x_1, \cdots, x_n) \wedge \bigwedge_{k=1}^{m} i_k \le x_k \le j_k \Big\} \to \psi(x_1, \cdots, x_n) \Big]$	$\bigwedge_{x_1^*=i_1}^{j_1} \cdots \bigwedge_{x_m^*=i_m}^{j_m} \forall x_{m+1}, \cdots, x_n \Big[\phi(x_1^*, \cdots, x_m^*, x_{m+1}, \cdots, x_n) \to \psi(x_1^*, \cdots, x_m^*, x_{m+1}, \cdots, x_n) \Big]$

*1 x は束縛変数，i, j は定数，f は多項式，h は束縛変数を含まない多項式，ϕ は論理式．
*2 $f \bmod i$ は f を j で割ったときの余りを表す．
*3 $f \operatorname{div} j$ は f を i で割ったときの商を表す．
*4 $j_{new} = \operatorname{lcm}(j_1, j_2, \cdots)$ であり，元の式と論理的に同値になるように i_{new} を決定する．

が閾値を超過した時点で探索を終了し，書き換え結果のうち限量子のネストの深さが最も浅いものを最良の結果として出力する．表 2 に示す式変形規則のうち，3 番と 5 番以外はすべて，旧システムにおいて用いていた規則を，含意の左辺に他の論理式が加わった場合や，連言で結合された部分式のうち複数のものが同じパターンにマッチする場合を扱えるように拡張したものである．また，旧システムにおいて低レベルな式変形規則による書き換えを経由して行っていた変形列は，正規化とマッチングの拡張によって，実質的に新システムでもすべて可能である．したがって，旧システムで可能だった式変形が新システムにおいて不可能になることはない．

4. 評価

システムの評価に際し，予備校マーク式問題集に掲載されている整数問題 25 問を用い，対応する中間表現を手作業により表 3 の通り作成した．この中間表現に対し，旧システムを用いた場合，および新システムを用いた場合のそれぞれについて，得点・実行時間・式変形探索の展開ノード数を調べた．

実験の結果は表 4 の通りであった．開発用データ，評価用データともに，合計得点，総実行時間，総展開ノード数が改善した．特に展開ノード数は劇的に減少した．旧システムではほとんどの論理式が交換律などの簡単な式変形規則にマッチしてしまい，いかなる子孫ノードについても本質的な式変形が不可能であるようなノードが多数生成されていた．これに対し新システムでは本質的な式変形以外では新しいノードを生成しないため，探索ノード数が大幅に減少したと考えられる．また，得点率の上昇は限定的であるが，これは，新システムには旧システムと比べて本質的に新しい式変形規則を多数追加したわけではないので，旧システムで除去できなかった限量子の多くは新システムでも除去できないためであると考えられる．

次に，各問題について旧システムと新システムそれぞれの実行時間の違いを調べた．図 4 は評価用データに対する Heuristic Solver の実行時間について，横軸を旧システムの実行時間，縦軸を新システムの実行時間にとって問題ごとにプロットしたものである．破線は縦軸と横軸の値が等しくなる直線である．

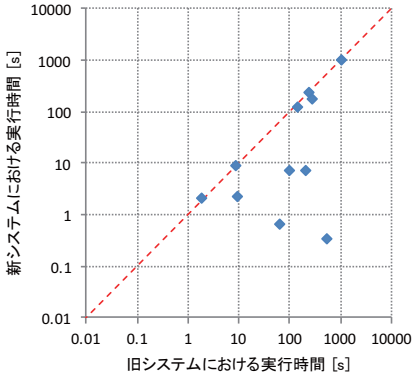

図 4: 旧システム・新システムの実行時間比較 (評価用データ)

図 4 が示すように，評価用データでは 11 問中 8 問において実行時間短縮の効果が認められた．実行時間が短縮した問題についてはすべての問題において 20 秒以上短縮されたが，実行時間が伸びたものについては，その時間変動はすべて 2 秒以内となっている．この実行時間の微小な増加は式変形探索以外の部分，主にドライバ・Princess・Mathematica における時間消費の揺らぎによるものと思われる．

最後に，評価用データのうち，新たに解けるようになった問題についてその理由を調べたところ，正規化規則 (40 番) により旧システムでは行われなかった変形が実現することで解けたものが 1 問，探索空間の削減の効果で解けるようになったものが 1 問あった．このうち後者は，旧システムにおいては探索ノード数が閾値 (実験では 1500 に設定) に達してしまい，限量子を除去できないまま探索を終了してしまうが，新システムにおいては図 5 に示す変形手順によりノード数 3 で限量子の除去を完了し，探索が終了する．

以上のように，低レベルな式変形は保持する式の正規化とマッチング規則によって吸収することで，より本質的な式変

表 2: 式変形規則

番号	変形前 *1	変形後 *1
1	$\forall x\,[(-i_1 + x \bmod i_2 = 0 \wedge \phi(x)) \to \psi(x)]$	$\forall y\,[\phi(i_1 + i_2 y) \to \psi(i_1 + i_2 y)]$
2	$\exists x_1, \cdots, x_m \left[\bigwedge_{k=1}^{m} f_k(x_1, \cdots, x_m) = 0\right]$	$\begin{cases}\top & (\text{充足する } x_1, \cdots, x_m \text{の組が存在}) \\ \bot & (otherwise)\end{cases}$
3	$\forall x_1, \cdots, x_m \left[\bigwedge_{k=1}^{m} (h_k + \sum(f(x_1, \cdots, x_m) \bmod i)) = 0\right]$	$\bigwedge_{i_1,\cdots,i_m \in \{0,1,\cdots,i-1\}} \bigwedge_{k=1}^{m} (h_k + \sum(f(i_1, \cdots, i_m) \bmod i)) = 0$
4	$\forall x, y \left[\left\{\phi(x,y) \wedge \bigwedge_{k=1}^{m} f_k(x,y) = 0\right\} \to \psi(x,y)\right]$	$\bigwedge_{(i_x,i_y) \in \left\{(i_x^*, i_y^*) \mid \bigwedge_{k=1}^{m} f_k(i_x^*, i_y^*) = 0\right\}} [\phi(i_x, i_y) \to \psi(i_x, i_y)]$
5	$[\exists x \phi_1(x) \wedge \phi_2] \to \phi_3$	$\forall x\,[\{\phi_1(x) \wedge \phi_2\} \to \phi_3]$
6	$\forall x_1, \ldots, x_n \left[\left\{\phi_1(x_1, \ldots, x_n) \wedge \bigwedge_{k=1}^{m} - i_k + x_k = 0\right\} \to \psi_1(x_1, \ldots, x_n)\right]$	$\forall x_{m+1}, \cdots, x_n$ $\left[\phi_1(i_1, \ldots, i_m, x_{m+1}, \ldots, x_n) \to \psi_1(i_1, \ldots, i_m, x_{m+1}, \ldots, x_n)\right]$
7*2	$\forall x \left[\phi(x) \to \bigwedge_k \left(\sum_l h_{k,l} x^l = 0\right)\right]$	$\bigwedge_{k,l} h_{k,l} = 0$
8*3	$\forall x_1, x_2 \left[\phi(x_1, x_2) \to \bigwedge_k \left(\sum_{l_1,l_2 \in \{0,1,2\}} h_{k,l_1,l_2} x_1^{l_1} x_2^{l_2} = 0\right)\right]$	$\bigwedge_{k,l_1,l_2} h_{k,l_1,l_2} = 0$
9*4	$\forall x_1, x_2 [\{i_0 + i_1 x_1 + i_2 x_2 = 0 \wedge \phi(x_1, x_2)\} \to \psi(x_1, x_2)]$	$\forall y [\phi(k_1 + l_1 y, k_2 + l_2 y) \to \psi(k_1 + l_1 y, k_2 + l_2 y)]$
10*5	$\forall x_1 \left[\phi(x_1) \to \exists x_2 \left\{\bigwedge_k h_{k1} + i_{k1} x_1 + h_{k2} x_2 = 0\right\}\right]$	$\bigvee_d \left[\left(\bigwedge_k \frac{i_{k1}}{d} + h_{k2} = 0\right) \wedge \exists y \left(\bigwedge_k h_{k1} + \frac{i_{k1}}{d} y = 0\right)\right]$

*1 x, y は束縛変数，i, j, k, l は定数，f は多項式，h は束縛変数を含まない多項式，ϕ は論理式．

*2 $\phi(x)$ を充足する x が上限あるいは下限を持たない場合のみ書き換えを行う．

*3 $\phi(x_1, x_2)$ が三次以上の項を含んでおらず，$\phi(x_1, x_2)$ を充足するすべての x_1, x_2 を解に持つ二元二次方程式・二元一次方程式がいずれも存在しない場合のみ書き換えを行う．

*4 $i_0 + i_1 x_1 + i_2 x_2 = 0 \Leftrightarrow \exists y\,[x_1 = k_1 + l_1 y \wedge x_2 = k_2 + l_2 y]$ となるように k_1, l_1, k_2, l_2 を設定する．

*5 $\phi(x_1)$ が x_1 について上限あるいは下限の高々一方のみを条件として与え，それ以外のいかなる条件をも与えない場合にのみ書き換えを行う．d はすべての i_{k1} の公約数．

表 3: 実験データ

-	問題数	配点
開発用	14	269
評価用	11	195
合計	25	464

表 4: 実行時間と展開ノード数

-	-	旧システム	新システム
開発用	得点	87/269	106/269
	平均実行時間 [s]	115.14	109.90
	平均展開ノード数	10357	1.29
評価用	得点	88/195	96/195
	平均実行時間 [s]	235.58	140.13
	平均展開ノード数	21224	2.55

$$\forall a \left[a \bmod 7 = 2 \to \forall b \left\{b \bmod 7 = 3 \to ab \bmod 7 = \boxed{\text{ケ}}\right\}\right]$$
$$\updownarrow \text{(正規化)}$$
$$\forall a \left[-2 + a \bmod 7 = 0 \to \forall b \left\{-3 + b \bmod 7 = 0 \to -\boxed{\text{ケ}} + ab \bmod 7 = 0\right\}\right]$$
$$\updownarrow \text{(式変形規則 1)}$$
$$\forall a \left[-2 + a \bmod 7 = 0 \to \forall k_b \left\{-\boxed{\text{ケ}} + (3a + 7ak_b) \bmod 7 = 0\right\}\right]$$
$$\updownarrow \text{(正規化)}$$
$$\forall a \left[-2 + a \bmod 7 = 0 \to \forall k_b \left\{-\boxed{\text{ケ}} + (3a) \bmod 7 = 0\right\}\right]$$
$$\updownarrow \text{(式変形規則 1)}$$
$$\forall k_a, k_b \left\{-\boxed{\text{ケ}} + (6 + 21ak_a) \bmod 7 = 0\right\}$$
$$\updownarrow \text{(正規化)}$$
$$-\boxed{\text{ケ}} + 6 = 0$$

図 5: 評価用データにおける式変形手順の例

形のみによる探索ノードの展開が可能になった．したがって，新システムでは探索空間の増大の影響を抑えつつ，式変形規則をさらに追加することが可能となったと予想される．今後は，現時点で限量子除去ができていな論理式の簡約に必要な式変形規則の精査，ヒューリスティックな解法パターンの抽出・蓄積が必要である．

5. 結論

本研究では，センター試験「数学」整数問題解答システムの探索処理を改良し，探索空間の削減に取り組んだ．式の正規化と拡張されたマッチング規則を用いた式変形探索手続きを構成し，開発用データ，評価用データのそれぞれに対して得点・実行時間・展開ノード数を調査した．その結果，低レベルな式変形規則による探索空間の増大が抑止されることで，実行時間が平均的に減少することが確かめられた．

今後は，現時点で限量子を除去できていない形の論理式については，簡約にどのような式変形規則が必要なのか，また式変形規則の個数・粒度はどの程度が適切か，より詳細な調査を行うことが課題である．

参考文献

[新井 12] 新井 紀子, 松崎 拓也: ロボットは東大に入れるか?—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, pp. 463–469 (2012)

[犬塚 17] 犬塚 慎也, 松崎 拓也, 佐藤 理史: センター試験「数学」整数問題解答システムの開発, 人工知能学会全国大会論文集, ROMBUNNO.3O1-5 (2017)

[岩根 13] 岩根 秀直, 松崎 拓也, 穴井 宏和, 新井 紀子: 数式処理による入試数学問題の解法と言語処理との接合における課題, 人工知能学会全国大会論文集 (CD-ROM) Vol. 27, ROMBUNNO.2A4-2 (2013)

[Rummer 08] Philipp Rummer: A constraint sequent calculus for first-order logic with linear integer arithmetic, LNCS, vol. 5330, pp. 274–289. Springer (2008)