一般社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report
MSS2012-78,SS2012-78(2013-3)

整数解を導出するための単体法とゴモリーカットの合成について

伏見　政晃† 　西田　直樹†† 　酒井　正彦†† 　草刈圭一朗†† 　坂部　俊樹††

†,††名古屋大学大学院情報科学研究科
〒464–8601 名古屋市千種区不老町
E-mail: †fushimi@sakabe.i.is.nagoya-u.ac.jp, ††{nishida,sakai,kusakari,sakabe}@is.nagoya-u.ac.jp

あらまし　与えられた有理数上の線形制約を充足する割り当てを求める手法として単体法がある．また，有理数解を求める手法と，ゴモリーカットをはじめとする切除平面法を組み合わせることで整数解を求められることが知られている．しかし，単体法を適用した後，必ずしもゴモリーカットが適用可能であるとは限らない．本稿では，ゴモリーカットの合成に必要な単体法における不変条件を示し，単体法とゴモリーカットを合成した手続きを示す．また，ゴモリーカットで追加する制約について，より単純な実装を行うための制約の形式を述べる．これらに基づいて，上記２つの手法を合成したソルバを実装し，評価する．

キーワード　単体法，ゴモリーカット，SMT ソルバ

On Composing the Simplex Method and Gomory Cut for Deriving Integer Assignments

Masaaki FUSHIMI†, Naoki NISHIDA††, Masahiko SAKAI††,
Keiichirou KUSAKARI††, and Toshiki SAKABE††

†,††Graduate School of Information Science, Nagoya University
Furo-cho, Chikusa-ku, 4648603 Nagoya, Japan
E-mail: †fushimi@sakabe.i.is.nagoya-u.ac.jp, ††{nishida,sakai,kusakari,sakabe}@is.nagoya-u.ac.jp

Abstract　The simplex method is one of the methods to derive rational assignments from linear constraints on ratiolals. It is known that we can derive integer assignments by composing the methods to derive rational assignments and cutting plane methods such as Gomory cut. However, Gomory cut is not applicable to all formulas resulted from the simplex method. In this paper, we first reveal an invariant property of the internal state of the simplex method in order to compose the simplex method and Gomory cut. Then we show a procedure of the composition of these methods, and we propose the form of constraints that are added into the initial constraint by applying Gomory cut so as to implement more simply. Finally, we compare our implemented solver with other solvers.

Key words　simplex method, Gomory cut, SMT solver

1. はじめに

プログラム検証手法の一つとして，制約付き項書換え系（制約付き TRS）の定理自動証明を利用した検証手法 [9] [11] [13] が研究されている．その検証手法では，証明の際に制約充足可能性を何度も判定する必要があり，その判定に毎回 SMT ソルバが用いられる．SMT ソルバとは指定された理論の下で論理式の充足可能性を判定するツールであり，近年，プログラムの停止性証明や定理自動証明などにおいて盛んに利用されている．前述の検証法では，充足可能性が決定可能なプレスブルガー算術式を制約として扱う．自由変数をすべて存在限量子で束縛し，Cooper の限量子除去アルゴリズム [1] を適用する手法であれば，理論上はプレスブルガー算術式の充足可能性が決定できる．この手法を実装したソルバと，既存のソルバとの比較を行った先行研究 [8] では，制約付き TRS の定理自動証明の例題に対して，全ての論理式の判定に成功し，実行速度においても優れていることが示された．しかし，すべての自由変数を束縛しなければ判定できない手法であり，束縛された変数の数に応じて論理式のサイズが指数関数的に増加してしまう．また，充足可能な場合でも自由変数への値の割り当て（整数解）までは求められない．そのため，この手法を用いては前述の検証法において，充足可能な際の整数解を利用することで証明能力を強化できる

推論規則 [10] の適用を自動化できない.

本研究では，前述の検証法での利用を目的とし，プレスブルガー算術式の充足可能性を判定し，かつ充足可能な際の整数解を導出するソルバを開発する．具体的には，Cooper の限量子除去アルゴリズムとゴモリーカット [2]（2 章）を導入した単体法を逐次実行することで目的のソルバを実現する．単体法は，有理数上の線形制約の有理数解を求める手法であり，単体法の一つに，最大化または最小化するための目的関数をとらない一般単体法 [4]（2 章）がある．ゴモリーカットは，得られた有理数解をもとに適当な制約を追加し，その有理数解を解領域から除外する切除平面法の一つである．また，有理数解を求める手法と切除平面法を組み合わせることで整数解を導出できることが知られている．しかし，一般単体法の場合，適用した後にゴモリーカットの適用条件を満たすとは限らない.

本稿では，一般単体法における不変条件を示し，一般単体法とゴモリーカットを合成した手続きを提案する（4 章）．これらに基づいて実装したソルバと既存の SMT ソルバで性能を比較し，評価する（5 章）．なお，Cooper の限量子除去アルゴリズムの適用により得られた論理式から単体法が適用可能な形式への変換は示されている [12].

2. 準　　備

本章では，一般単体法 [4] とゴモリーカット [2] を紹介する.

2.1 一般単体法

一般単体法は，最大化または最小化する目的関数をとらずに有理数上の線形制約を充足する有理数解を導出する手法である．以降では，この一般単体法のことを指して単体法と呼ぶ.

まず，単体法を適用するためには，制約を一般形と呼ばれる形式に変換する必要がある．一般形の文法は次の通りである．ただし，a は有理数，x は有理数上の変数である.

$$\begin{array}{rcl} \phi & ::= & \top \mid \bot \mid \phi \wedge \phi \mid atom \\ atom & ::= & sum = 0 \mid x \leq a \mid x \geq a \\ sum & ::= & t \mid sum + t \\ t & ::= & a\,x \end{array}$$

次に，単体法のアルゴリズムについて述べる．まず，論理式 ϕ_0 が与えられ，これが一般形 ϕ に変換されたとする．このとき，単体法のアルゴリズムを以下に示す.

アルゴリズム 2.1 単体法

入力: 一般形の論理式 ϕ

出力:“充足可能”（ϕ_0 が有理数解をもつ場合）もしくは “充足不能”（それ以外の場合）

（1） ϕ のみに含まれる，一般形への変換によって追加された変数を基底変数とし，基底変数リスト V_b に加える.

（2） ϕ 中の変数の内，V_b に含まれない変数を非基底変数とし，非基底変数リスト V_{nb} に加える.

（3） 図 1 のようなタブローを用意する．ただし，m は V_b の長さ，n は V_{nb} の長さ，$a_{i,j}$ $(0 \leq i \leq m-1, 0 \leq j \leq n-1)$ は有理数とする．図 1 のタブローでは，s_i $(0 \leq i \leq m-1)$ が

	x_0	x_1	$\cdots$	x_{n-1}
s_0	$a_{0,0}$	$a_{0,1}$	$\cdots$	$a_{0,n-1}$
s_1	$a_{1,0}$	$a_{1,1}$	$\cdots$	$a_{1,n-1}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
s_{m-1}	$a_{m-1,0}$	$a_{m-1,1}$	$\cdots$	$a_{m-1,n-1}$

図 1　単体法で用いるタブロー

基底変数，x_j $(0 \leq j \leq n-1)$ が非基底変数であり，各行 i は $s_i = \sum_{j=0}^{m-1} a_{i,j} x_j$ という原子論理式を表している.

（4） 変数に固定の順序を与える.

（5） すべての変数へ 0 を割り当てる割り当て α を用意する．以降，$\alpha(x)$ と書いて変数 x への割り当て値を表す.

（6） すべての基底変数への割り当てがそれぞれの定義域を満たしているならば “充足可能” を出力し，そうでなければ定義域を満たしていない基底変数のうち，(4) で定めた変数の順序が最も小さい変数を s_i とする.

（7） 適当な非基底変数のうち，最も変数の順序が小さいものを x_j とする．ここで，変数 x_j が適当であるとは，以下のように定義される.

- $\alpha(s_i)$ が s_i の定義域の上限を超えている場合.

$a_{i,j} > 0$ かつ $\alpha(x_j) > l_j$，もしくは $a_{i,j} < 0$ かつ $\alpha(x_j) < u_j$ であれば，変数 x_j は適当である.

- $\alpha(s_i)$ が s_i の定義域の下限よりも小さい場合.

$a_{i,j} > 0$ かつ $\alpha(x_j) < u_j$，もしくは $a_{i,j} < 0$ かつ $\alpha(x_j) > l_j$ であれば，変数 x_j は適当である.

適当な非基底変数がなければ “充足不能” を出力し，終了.

（8） 次のように θ を定義する.

$$\theta = \begin{cases} \dfrac{u_i - \alpha(s_i)}{a_{i,j}} & \left(\begin{array}{l}\alpha(s_i) \text{ が } s_i \text{の定義域の上限を} \\ \text{超えている場合}\end{array}\right) \\ \dfrac{l_i - \alpha(s_i)}{a_{i,j}} & \left(\begin{array}{l}\alpha(s_i) \text{ が } s_i \text{の定義域の下限よりも} \\ \text{小さい場合}\end{array}\right) \end{cases}$$

$\alpha(x_j) := \alpha(x_j) + \theta$ とすることで，$\alpha(s_i)$ が s_i の定義域を満たすようにする.

（9） s_i，x_j についてピボット操作を行い，(6) に戻る.

次に，アルゴリズム 2.1 におけるピボット操作を以下に示す.

アルゴリズム 2.2 ピボット操作

入力: 基底変数 s_i，非基底変数 x_j，タブロー T，基底変数リスト V_b，非基底変数リスト V_{nb}，変数への割り当て α

出力: タブロー中の s_i の行が表す等式を x_j に関する式に変形し，他の行が表す等式から変数 x_j を消去することで得られた，タブロー T'，基底変数リスト V_b'，非基底変数リスト V_{nb}'，変数への割り当て α'

ただし，x_j は適当なものを選んでいるので，$a_{i,j} \neq 0$ が前提となっている．この条件の下で以下の操作を行う.

（1） タブローの i 行目が表す等式の左辺が x_j のみになるように移項し，両辺を $a_{i,j}$ で割る.

（2） $l \neq i$ である全ての行 l について，(1) で得られた等式を用いて x_j を消去する．

（3） s_i を V_b から取り除き，V_{nb} へ加える．また，x_j を V_{nb} から取り除き，V_b へ加える．これにより得られた基底変数リストを V_b'，非基底変数リストを V_{nb}' とする．

（4） (1)，(2) の操作により得られた論理式と現在の変数の割り当てをもとに，x_j 以外の基底変数への割り当てを変更する．新たに得られた割り当てを α' とする．

（5） (1)，(2) の操作により得られた論理式に合わせてタブローの値を変更する．得られたタブローを T' とする．

（6） T'，V_b'，V_{nb}'，α' を出力する．

2.2 ゴモリーカット

ゴモリーカット [2] は，有理数上の線形制約式を解くことである有理数解が得られたとき，適当な制約を追加することでこの有理数解を解領域から除外する切除平面法の一つである．有理数解を求める手法と切除平面法を組み合わせることで整数解を導出できることが知られている．

ゴモリーカットのアルゴリズムは以下の通りである．

アルゴリズム 2.3 ゴモリーカット

入力: 基底変数リスト V_b，非基底変数リスト V_{nb}，変数の定義域リスト B，タブロー T，有理数解を表す変数への割り当て α

出力: 入力の有理数解を解領域から除外する制約を追加することで更新された，基底変数リスト V_b'，非基底変数リスト V_{nb}'，変数の定義域リスト B'，タブロー T'，有理数解を表す変数への割り当て α'

（1） タブロー T において，次の 2 つの条件を満たす行 i を選択．条件を満たす行がなければゴモリーカットは行わない．

- $x_i \in bv \wedge \alpha(x_i) \notin \mathbb{Z}$
- $\forall j. 0 \leqq j \leqq (nbv\_length - 1) \wedge x_j \in nbv$

$\Longrightarrow \left(a_{i,j} \neq 0 \Longrightarrow \alpha(x_j) = l_j \vee \alpha(x_j) = u_j\right)$

ただし，$a_{i,j}$ はタブローにおける i 行 j 列目の値，nbv_length は V_{nb} の長さを表す．

このアルゴリズムにおいては，以降 i は (1) で選んだ i とする．

（2） $x_i (\in V_b)$ について $f_0 = \alpha(x_i) - \lfloor \alpha(x_i) \rfloor$ とする．

（3） タブローの列番号 j について，次のように分類する．

- $J^+ = \{j \mid x_j \in V_{nb} \wedge \alpha(x_j = l_j) \wedge a_{i,j} > 0\}$
- $J^- = \{j \mid x_j \in V_{nb} \wedge \alpha(x_j = l_j) \wedge a_{i,j} < 0\}$
- $K^+ = \{j \mid x_j \in V_{nb} \wedge \alpha(x_j = u_j) \wedge a_{i,j} > 0\}$
- $K^- = \{j \mid x_j \in V_{nb} \wedge \alpha(x_j = u_j) \wedge a_{i,j} < 0\}$

（4）

$$\sum_{j \in J^+} \frac{a_{i,j}}{1 - f_0}(x_j - l_j) - \sum_{j \in J^-} \frac{a_{i,j}}{f_0}(x_j - l_j) + \sum_{j \in K^+} \frac{a_{i,j}}{f_0}(u_j - x_j) - \sum_{j \in K^-} \frac{a_{i,j}}{1 - f_0}(u_j - x_j) \geqq 1 \qquad (1)$$

式 (1) を追加する論理式として V_b, B, T を更新する．ゴモリーカットで追加する制約を単体法における一般形に変換する際に導入する新しい変数 s には，右辺を構成する変数に α の値を代入することで得られる定数を割り当てる．これにより得られた基底変数リスト，タブロー，変数の定義域，変数への割り当てをそれぞれ V_b'，T'，B'，α' とする．

（5） V_b'，V_{nb}，T'，B'，α' を出力．

	$x_0:0$	$x_1:0$	$x_2:0$	$x_3:0$
$s_0:0$	3	0	-2	1
$s_1:0$	-3	0	2	-1
$s_2:0$	2	-3	1	0
$s_3:0$	-2	3	-1	0

	下限	上限
s_0	$-\infty$	5
s_1	$-\infty$	-5
s_2	$-\infty$	6
s_3	$-\infty$	-6

図 2 単体法を適用する前のタブローと変数の定義域

3. 単体法とゴモリーカットの合成の問題点

単体法の適用後のタブロー等の状態がゴモリーカットの適用条件を満たすとは限らないため，この 2 つの手法を単純に組み合わせて利用することはできない．この章では，単体法とゴモリーカットの合成における問題点について述べる．

3.1 ゴモリーカットの適用条件を満たさない例

まず，単体法を適用するために，それが適用可能な一般形へと変換することを第 2 章で述べた．もとの論理式から一般形に変換すると，論理式の形式は式 (2) のようになる．

$$\bigwedge_{i=1}^{m} \left(-s_i + \sum_{j=1}^{n} a_j x_j = 0 \ \wedge \ l_i \leqq s_i \leqq u_i \right) \qquad (2)$$

ただし，a_j は有理数，x_j は有理数上の変数の列である．また，変数 s_i について，$l_i = -\infty$ の場合は $s_i \leqq u_i$ と表し，$u_i = \infty$ の場合は $l_i \leqq s_i$ と表す．一般形への変換では，変換によって追加される変数 s_i のみが上限 u_i，下限 l_i のいずれか，もしくは両方をもち，変換前の論理式に含まれていた変数 x_j は定義域をもたない．ここで，定義域をもたないとは，その変数のとり得る値が $-\infty$ から ∞ で定義されることをいう．また，単体法の適用において，初めは s_i が基底変数，x_j が非基底変数となる．ここで，単体法を適用することによって得られた解が整数解でなかったとする．この場合は，ゴモリーカットの適用による制約の追加と単体法の適用を繰り返すことで整数解を導出できると考えられる．しかし，単体法を適用した後のタブローの状態はゴモリーカットの適用条件を満たさない場合がある．

例 3.1 次の論理式を考える．

$$\begin{array}{llllllll} & 3x_0 & & -2x_2 & +x_3 & -s_0 & = 0 & \wedge\ s_0 \leqq 5 \\ \wedge & -3x_0 & & +2x_2 & -x_3 & -s_1 & = 0 & \wedge\ s_1 \leqq -5 \\ \wedge & 2x_0 & -3x_1 & +x_2 & & -s_2 & = 0 & \wedge\ s_2 \leqq 6 \\ \wedge & -2x_0 & +3x_1 & -x_2 & & -s_3 & = 0 & \wedge\ s_3 \leqq -6 \end{array}$$

この論理式は単体法における一般形であり，初めは $s_0, \ldots, s_3$ が基底変数，$x_0, \ldots, x_3$ が非基底変数である．上記の論理式をもとにタブローをつくると図 2 のようになる．ただし，以降ではタブローにおいて $x:n$ と書くことで，変数 x への割り当てが n であることを表すものとする．次に単体法を適用していく．ここでは，変数の順序を式 (3) とする．

$$s_0 < \cdots < s_3 < x_0 < \cdots < x_3 \qquad (3)$$

	$s_3:-6$	$s_0:5$	$x_2:0$	$x_3:0$
$x_1:-8/9$	1/3	2/9	7/9	-2/9
$x_0:5/3$	0	1/3	2/3	-1/3
$s_2:6$	-1	0	0	0
$s_1:-5$	0	-1	0	0

	下限	上限
s_0	$-\infty$	5
s_1	$-\infty$	-5
s_2	$-\infty$	6
s_3	$-\infty$	-6

図 3　有理数解導出後のタブローと変数の定義域

この条件のもとで単体法を適用すると有理数解が求まり，タブローは図 3 のようになる．図 3 より，導出された解は

$$\left\{ \begin{array}{llll} x_0 \mapsto 5/3, & x_2 \mapsto 0, & x_1 \mapsto -8/9, & x_3 \mapsto 0, \\ s_0 \mapsto 5, & s_1 \mapsto -5, & s_2 \mapsto 6, & s_3 \mapsto -6 \end{array} \right\}$$

であり，これは整数解ではない．ただし，このとき解が整数解であるとは，すべての基底変数に整数値が割り当てられていることをいう．したがって解領域を制限するために，ゴモリーカットの適用を考える．第 2 章で述べたゴモリーカットの適用条件は次の 2 つの条件を満たすことであった．

- 割り当てが整数でない基底変数が存在する．
- すべての非基底変数に対し，それぞれの定義域の上限または下限の値が割り当てられている．

図 3 において，非基底変数は s_3, s_0, x_2, x_3 である．このとき，s_3，s_0 にはそれぞれ各変数の定義域の上限である -6，5 が割り当てられている．一方，x_2，x_3 は定義域をもたないため，それぞれに割り当てられた 0 という値は，上限と下限のいずれでもない．したがってゴモリーカットを適用することができない．このように単体法の適用後のタブロー，変数の割り当ては必ずしもゴモリーカットの適用条件を満たすとは限らない．

4. 単体法とゴモリーカットの合成

本章では，単体法とゴモリーカットを合成した手続きを示す．

4.1 単体法とゴモリーカットを合成した手続き

前章では，単体法の適用後の状態が，ゴモリーカットの適用条件を満たすとは限らないということを述べた．本節では，これら 2 つの手法を組み合わせ，整数解を導出する手続きを示す．

まず，第 2 章で述べた単体法についてであるが，このアルゴリズムにおいて次のような不変条件が知られている．

- 非基底変数への割り当ては常にその定義域を満たす．

しかし，このアルゴリズムでは，次の条件も不変条件である．

- 定義域をもつ非基底変数への割り当ては常にその変数の上限または下限である．

これが不変条件であることについて述べる．

- 初め，すべての変数には 0 が割り当てられる．
- このときの非基底変数は定義域をもたず，条件を満たす．
- 単体法のアルゴリズム (6) において，定義域を満たしていない基底変数に s_i を選び，適当な非基底変数に x_j を選ぶ．
- 単体法のアルゴリズム (8) で，$\alpha(x_j)$ の値に θ を加えると，$\alpha(s_i)$ は s_i が上限を超えていた場合はその上限の値になり，下限よりも小さかった場合はその下限の値になる．
- ピボット操作により，s_i と x_j がタブロー上で入れ替えられる．このとき，非基底変数に変わった s_i はその上限または下限の値が割り当てられており，条件を満たす．
- 定義域をもたない x_j のような変数はどのような値が割り当てられても良いため，一度基底変数になると，再び非基底変数になることはない．
- 以降，基底変数から非基底変数に変わる変数には単体法のアルゴリズム (8) による割り当ての修正により，非基底変数になるときには必ずその変数の上限または下限の値が割り当てられることになる．

以上より，上述の条件は単体法の不変条件である．また，ゴモリーカットの適用条件の 1 つは，すべての非基底変数にその変数の上限または下限の値が割り当てられていることであった．よって上述の不変条件は，ゴモリーカット適用における必要条件でもある．

ここで，ゴモリーカットが適用できない場合，単体法適用後のタブロー等は必ず次の状態になっている．

- 0 が割り当てられた，定義域のない非基底変数が存在．

上述したように，定義域をもたない変数は，一度基底変数になるとピボット操作の対象になることはない．また非基底変数の割り当てが変わるのは，ピボット操作の対象になったときのみである．定義域をもたない変数は，初めは非基底変数であり，有理数解が求まった状態においても非基底変数として残っている場合，その変数は割り当てが初めの状態から変わっていないことを表している．したがってその変数の割り当ては 0 である．

以上より，これらの変数の定義域の上限または下限が 0 であれば，ゴモリーカットを適用できる．そこで単体法とゴモリーカットの合成のために，次のような手法を提案する．

> 定義域をもたない非基底変数 x に $x \geq 0$，$x \leq 0$ という仮の定義域を与えて場合分けすることで，ゴモリーカットを適用できるようにする．

この手法の具体的な手続きについて以下に示す．

手続き 4.1　単体法とゴモリーカットの合成
入力：単体法における一般形の論理式 ϕ
出力：充足可能（ϕ が充足可能な場合）もしくは充足不能（それ以外の場合）

（1）　入力論理式に対し，単体法を適用する．出力が “充足不能” ならば “充足不能” を出力して終了し，“充足可能” ならば (2) へ進む．

（2）　整数解導出手続きを行い，出力が “充足不能” ならば “充足不能” を出力して終了し，“充足可能” ならば “充足可能” を出力して終了する．

次に，上記の整数解導出手続きについて述べる．

手続き 4.2　整数解導出手続き
入力：タブロー T，基底変数リスト V_b，非基底変数リスト V_{nb}，変数の定義域リスト B，変数への割り当て α，

（1）　すべての基底変数への割り当てが整数ならば “充足可能” を出力．

（2）　割り当てが整数でない基底変数を一つ選び，x とする．

	$s_3:-6$	$s_0:5$	$x_2:0$	$x_3:0$
$x_1:-8/9$	1/3	2/9	7/9	-2/9
$x_0:5/3$	0	1/3	2/3	-1/3
$s_2:6$	-1	0	0	0
$s_1:-5$	0	-1	0	0

	下限	上限
s_0	$-\infty$	5
s_1	$-\infty$	-5
s_2	$-\infty$	6
s_3	$-\infty$	-6
x_2	0	∞
x_3	0	∞

図 4 仮の定義域を追加した後のタブローと変数の定義域

（3） すべての非基底変数への割り当てがその変数の上限，または下限の値ならば（8）へ進む．

（4） 定義域をもたない非基底変数を一つ選び，y とする．この変数 y の定義域を $y \geqq 0$ として B を更新する．

（5） 整数解導出手続きを行い，その出力が “充足可能” ならば “充足可能” を出力し，“充足不能” ならば（6）へ進む．

（6） （4）で選んだ非基底変数 y の定義域を $y \leqq 0$ とする．

（7） 整数解導出手続きを行い，その出力が “充足可能” ならば “充足可能” を，“充足不能” ならば “充足不能” を出力する．

（8） 選択した基底変数 x のタブロー中の行についてゴモリーカットを行った後，単体法を適用する．単体法を適用した結果，出力が “充足可能” ならば再び整数解導出手続きを行い，“充足不能” ならば “充足不能” を出力する．

4.2 ゴモリーカットで追加する制約の形式

前節では，単体法とゴモリーカットを合成するための手続きについて述べた．それにより，単体法で有理数解が導出された状態であればゴモリーカットにより制約を追加することができるようになった．しかし，ここで得られた制約をそのまま単体法における一般形に変換した場合，その変数に要求される条件はゴモリーカット適用前から存在する変数と異なる．具体的には，整数解が求まった状態とは，ゴモリーカットの適用前から存在する変数すべてに整数が割り当てられている状態であり，ゴモリーカットで追加する制約を一般形に変換することで導入される新しい変数への割り当ては整数でなくても良い，ということである．以下に例を示す．

例 4.3 図 3 のタブローおよび変数の定義域，割り当てを考える．これは第 3 章で示した，ある論理式に単体法を適用した後の状態である．このとき，整数解として例えば次の割り当てが存在する．

$$\left\{ \begin{array}{llll} x_0 \mapsto 0, & x_1 \mapsto 0, & x_2 \mapsto 6, & x_3 \mapsto 17, \\ s_0 \mapsto 5, & s_1 \mapsto -5, & s_2 \mapsto 6, & s_3 \mapsto -6 \end{array} \right\} \quad (4)$$

ここで，前節で提案した手続きを適用することを考える．まず，定義域を持たない非基底変数 x_2，x_3 が存在するため，順に $x_2 \geqq 0$，$x_3 \geqq 0$ という定義域を与える．このとき，定義域は図 4 のようになる．図 4 の状態はゴモリーカットの適用条件を満たしている．割り当てが整数でない基底変数として，x_1 を選ぶと，ゴモリーカットで追加する制約は次のようになる．

$$-3s_3 - 2s_0 + \frac{7}{8}x_2 + 2x_3 \geqq 9 \quad (5)$$

式（5）を一般形に変換すると，新しい変数 s_4 が導入され，式（6）となる．

$$-3s_3 - 2s_0 + \frac{7}{8}x_2 + 2x_3 - s_4 = 0 \wedge s_4 \geqq 9 \quad (6)$$

式（4）の割り当てを式（6）へ代入すると，式（7）が得られる．

$$s_4 = \frac{189}{4} + \mathbb{Z} \quad (7)$$

式（7）より，ゴモリーカットの適用後に追加された変数への割り当てが整数であることが，もとの論理式の整数解であることの必要条件であるとすると，追加される制約はもとの論理式の整数解の一つである式（4）を除外してしまう．

したがって，ゴモリーカットの適用後に追加された変数への割り当てが整数であることは，もとの論理式への割り当てが整数解であることの必要条件ではない．よって，第 2 章のゴモリーカットのアルゴリズムでは，得られた解が整数解であることを「すべての基底変数に整数が割り当てられた状態」として判定していたが，「ゴモリーカット適用以前から存在する基底変数すべてに整数が割り当てられた状態」とする必要がある．つまり，整数解導出の手続きとして，ゴモリーカット適用によって導入された変数かどうかを区別しなければならない．

そこで本節において，ゴモリーカットで追加される制約の形式を「すべての係数，定数項が整数である形式」とする．このために，以下の操作を行う．

> ゴモリーカットで追加される制約を単体法における一般形へ変換する前に，すべての係数および定数項の分母の最小公倍数を両辺にかける．

一般形に変換する前にすべての係数，定数項を整数にすることにより次のことが言える：ゴモリーカットの適用後に追加される変数への割り当てが整数であることは，もとの論理式の整数解であることの必要条件である．これにより，整数解が得られた状態を第 2 章の通りに定めることができる．したがって，ゴモリーカットの適用にともなって導入された変数かどうかを区別する必要がなくなり，より単純で効率的な実装が可能になると考えられる．

5. 実験と評価

Cooper の限量子除去アルゴリズムと本稿で提案した手法を逐次実行するソルバを実装した．本章では，実装したソルバと既存の SMT ソルバとの性能を比較するために行った実験と評価について述べる．

5.1 実験方法

本節では，比較のために行った実験について述べる．

まず，比較の対象とする SMT ソルバは Z3 (ver. 3.2) [7], Yices (ver. 1.0.29) [6], CVC3 (ver. 2.4.1) [3], 先行研究 [8] で実装されたソルバの 4 つである．ただし，[8] のソルバは論理式の充足可能性のみ求めることができる．また，実験環境は，OS： Windows 7 (32bit)，CPU： Intel Core 2 Duo，クロック周波数： 3.16 GHz，主記憶： 4.00 GB である．

次に，実験の手順について以下に示す．

表 1 例題 (a), (b), (c) における各ソルバの判定結果と実行時間

(a) の判定結果	本研究	Z3	Yices	CVC3	先行研究 [8]
判定可	2310	2295	2295	2270	2310
不明	0	15	15	40	0
合計実行時間 (秒)	13.24	7.35	6.66	6.36	11.44

(b) の判定結果	本研究	Z3	Yices	CVC3	先行研究 [8]
判定可	3149	3134	3134	3109	3149
不明	0	15	15	40	0
合計実行時間 (秒)	17.10	9.79	8.83	8.70	15.79

(c) の判定結果	本研究	Z3	Yices	CVC3	先行研究 [8]
判定可	4467	4450	4444	4417	4467
不明	0	17	23	50	0
合計実行時間 (秒)	24.98	14.22	11.89	12.63	22.42

(1) 制約付き TRS の定理自動証明の例題を 3 題用意.

(2) 用意した 3 つの例題について定理自動証明を行う過程で, 充足可能性を判定すべき論理式をすべて SMT-LIB ver. 1.2 [5] の AUFLIA に準拠した文法で記述した個別のファイルに変換する. 論理式ファイルの数は, 例題 (a) が 2310 個, 例題 (b) が 3149 個, 例題 (c) が 4467 個である. ただし, 証明において判定が必要な論理式をすべて別ファイルとして扱っているため, 論理式の個数は重複を含んだ上での数である.

(3) ソルバの出力は, "充足可能", "充足不能", "不明".

(4) 上記設定の下, 用意した各例題ごとに, その例題で充足可能性を判定すべきすべての論理式ファイルの判定を行う. その際, 各ファイルの判定ごとに実行時間を計測し, その合計時間を求める. この操作を各ソルバごとに 5 回行い, 合計実行時間の平均を求める.

5.2 実験結果と評価

本節では, 前節の実験を行った結果とその評価について述べる. まず, 実験を行った結果を表 1 に示す. 表の見方であるが, "判定可", "不明" の各行は各ソルバが判定した論理式の合計数を表しており, "判定可" は "充足可能", "充足不能" と判定されたものを指す. 本研究の実装の判定結果は先行研究 [8] の実装と同じであった. 最下行の合計実行時間は, その例題で解くべき論理式をすべて解くのに要した時間を表す. これらの表をもとにソルバの比較を行う.

a) 判定できた論理式の数

表 1 より, 本研究の実装と, Cooper の限量子除去アルゴリズムに基づいた先行研究の実装では (a), (b), (c) の 3 つの例題において, 判定すべきすべての論理式の充足可能性を判定することができた. これに対し, それ以外のソルバでは充足可能性を判定できずに不明を出力する論理式があった. 既存のソルバで不明を出力する論理式についてであるが, 今回の実験ではタイムアウトの時間は設定していない. いずれのソルバも即座に不明を出力しており, 判定ができない理由はタイムアウトやメモリオーバーフローによるものではなかった.

b) 実行速度

実験結果の各表より, 本研究で実装したソルバは, 広く使われている Z3, Yices, CVC3 と比較して実行速度がおよそ 2 倍遅かった. また先行研究で実装されたソルバに対しても, およそ 10%ほど遅かった. ただし今回の実験では, 実装したソルバが判定にかかる時間は, 論理式 1 個あたり 5 ミリ秒程度であり, 実用的な実行時間であるといえる.

6. おわりに

本稿では, 整数上の線形制約の整数解を導出するための手法として, 単体法とゴモリーカットの合成のための手続きについて提案した. Cooper の限量子除去アルゴリズムと提案した手法を逐次実行することで, プレスブルガー算術式の整数解を導出するソルバを実装した. そして制約付き TRS の定理自動証明の例題を用いて, 実装したソルバと Z3, Yices, CVC3, および先行研究 [8] との比較実験を行った. その結果, 本手法により整数解を導出することはできたが, 実行速度の点では他のソルバよりも遅かった. 今後の課題としては, まず実装上の効率化を図ることが挙げられる. また, ゴモリーカットを組み合わせた場合の停止性は保証されていないため, より収束しやすい論理式の形式や条件についても考える必要がある.

謝辞 : 本研究は, 一部, 文部科学省科学研究費#20300010, #20500008, #21700011 の助成を受けたものである.

文 献

[1] A. R. Bradley, Z. Manna.: The Calculus of Computation, Decision Procedures with Applications to Verification. Springer, 2007.
[2] B. Dutertre and L. de Moura.: Integrating simplex with DPLL(T). Technical Report SRI-CSL-06-01, Stanford Research Institute (SRI), 2006.
[3] CVC3 page: http://www.cs.nyu.edu/acsys/cvc3/
[4] D. Kroening, O. Strichman.: Decision Procedures: An Algorithmic Point of View, Springer, 2008.
[5] SMT-LIB: http://www.smtlib.org/
[6] The Yices SMT Solver: http://yices.csl.sri.com/
[7] Z3: SMT solver: http://research.microsoft.com/en-us/um/redmond/projects/z3/ml/z3.html
[8] 小林裕幸 : 整数上の線形制約に対する充足可能性判定ツールの実装と評価, 名古屋大学工学部, 卒業論文, 2011.
[9] 坂田翼, 西田直樹, 坂部俊樹, 酒井正彦, 草刈圭一朗 : 制約付き項書換え系における書換え帰納法, 情報処理学会論文誌 プログラミング, Vol. 2, No. 2, pp. 80–96, 2009.
[10] 中林直生 : 制約付き項書換え系の書換え帰納法における補題等式の自動生成, 名古屋大学, 修士論文, 2010.
[11] 中林直生, 西田直樹, 草刈圭一朗, 坂部俊樹, 酒井正彦 : 制約付き項書換え系の書換え帰納法における補題等式の自動生成法, コンピュータソフトウェア, Vol. 28, No. 1, pp. 173–189, 2011.
[12] 伏見政晃, 西田直樹, 坂部俊樹, 酒井正彦, 草刈圭一朗 : Cooper の限量子除去アルゴリズムと単体法を逐次合成するための論理式変換, 平成 24 年度電気関係学会東海支部連合大会講演論文集, A4-2, 2012.
[13] 古市祐樹, 西田直樹, 酒井正彦, 草刈圭一朗, 坂部俊樹 : 制約付き項書換え系の潜在帰納法を利用した手続き型プログラム検証の試み, 情報処理学会論文誌 プログラミング, Vol. 1, No. 2, pp. 100–121, 2008.