OUKA
The University of Osaka
Institutional Knowledge Archive

Title	凹多面体併合を用いた有理数プレスブルガー文真偽判定アルゴリズムの実装と形式的設計検証への適用
Author(s)	柴田, 直樹; 岡野, 浩三; 谷口, 健一
Citation	電子情報通信学会論文誌D. 2001, J84-D1(7), p. 999-1008
Version Type	VoR
URL	https://hdl.handle.net/11094/27415
rights	Copyright © 2001 IEICE
Note	

The University of Osaka Institutional Knowledge Archive : OUKA

https://ir.library.osaka-u.ac.jp/

The University of Osaka

論　文

凹多面体併合を用いた有理数プレスブルガー文真偽判定アルゴリズムの実装と形式的設計検証への適用

柴田　直樹[†]　　岡野　浩三[†]　　谷口　健一[†]

A Decision Algorithm for Rational Presburger Sentences Using a Merge Routine for Polyhedra and Its Application to Formal Verification

Naoki SHIBATA[†], Kozo OKANO[†], and Kenichi TANIGUCHI[†]

あらまし　加算をもつ有理数の理論（有理数変数，有理数定数，+，−，=，<，∧，∨，∀，∃からなる理論）の上の閉論理式（RP 文）の真偽判定ルーチンは通信プロトコルのテスト，ハードウェアのタイミング検証などに利用できる．筆者らは以前，計算幾何学の手法を利用し，時間計算量を改善した RP 文真偽判定アルゴリズムを提案した．本論文では，まずそのアルゴリズムに対する凹多面体併合を用いた高速化法について述べる．次に，その手法を実装した真偽判定ルーチンを MC68030 バス上で非同期バスマスタ転送を行う時間オートマトンの適合性試験系列生成に適用し，評価した結果について述べる．高速化により，比較的簡単な時間制約をもつ時間オートマトンの実行可能性判定の例に対し，実際の検証で現れる変数の数が 16 個，不等式の数が 20 個程度の RP 文を CPU 時間数秒程度（Pentium III 600 MHz）で判定できるようになった．

キーワード　有理数プレスブルガー文，真偽判定，凹多面体，適合性試験

1. まえがき

加算をもつ有理数の理論（有理数変数，有理数定数，+，−，=，<，∧，∨，∀，∃からなる理論）の閉論理式（以降有理数プレスブルガー文または RP 文と呼ぶ）の真偽判定ルーチンは通信プロトコルのテスト，ハードウェアのタイミング検証などに利用できる [1]．

これまで，加算をもつ整数の理論（プレスブルガー文）やそのサブクラス，有理数プレスブルガー文のサブクラス，bit vector arithmetic の真偽判定ルーチンが実際に検証系に用いられてきた [2], [3]．しかし，実用性が問題となり，種々の応用例が考えられるにもかかわらず有理数プレスブルガー文の真偽判定ルーチンを検証に使用した例はない．

筆者らは文献 [4] で解説されている，従来知られていた最も速いアルゴリズムである Ferrante と Rackoff のアルゴリズム [5] よりも時間計算量オーダの小さいアルゴリズムをこれまでに提案した [6]．Ferrante と Rackoff のアルゴリズムの時間計算量は $r5^{\epsilon d}n^{\zeta d(2b+1)^a}$（$n$，$d$，$a$，$b$，$r$ はそれぞれ入力の RP 文に含まれる不等式の個数，変数の個数，限定子交替数，同じ限定子が続く最大の個数，入力の式の各係数と定数の分母，分子の最大のビット数，ϵ，ζ は定数）であり，筆者らの提案したアルゴリズムの時間計算量は $r\alpha^{\beta d}n^{\gamma d(b+1)^a}$（$\alpha$，$\beta$，$\gamma$ は定数）である．双方のアルゴリズムの計算量で支配的なのは 2 重指数の部分であり，提案したアルゴリズムの計算量は Ferrante と Rackoff のアルゴリズムのものに比べてこの部分が改善されている．

本論文では，筆者らが提案したこのアルゴリズムに対して，更に隣接する凹多面体の併合等を用いて高速化する手法について述べる．本手法で高速化した真偽判定ルーチンをモトローラ社製プロセッサ MC68030 [7] のバス上で非同期バスマスタ転送を行う時間オートマトンの適合性試験系列生成に適用したところ，実際の検証で現れる変数の数が 16 個，不等式の数が 20 個程度の RP 文を CPU 時間数秒程度（Pentium III 600 MHz）で判定できた．従来の RP 文真偽判定ルーチンでは，その例に対し実用上判定不可能であった．また，本高速化により，すべての係数のうち 7 割が 0 であるような，ランダムに生成した式に対し，変数の数が 7，不等式の数が 15 程度で数百秒程度で判定でき

[†] 大阪大学大学院基礎工学研究科情報数理系専攻，豊中市
Graduate School of Engineering Science, Osaka University, Toyonaka-shi, 560–8531 Japan

ることがわかった．

提案する高速化法として，1. 投影後に同じ真偽値が割り当てられた（共有部分をもつ，あるいは隣接した）多面体を併合し，その結果できる凹多面体も扱えるようにすることで，処理する多面体の数を最小限にすること，2. 投影後に凹多面体の重なり部分を除去する際の凹多面体同士の交差判定の回数を減らすことなどが挙げられる．

以後，**2.** では以前提案したアルゴリズムの概要を述べ，**3.** では今回提案する高速化手法について述べる．**4.** では実装した高速化手法を評価した結果及び考察について述べる．

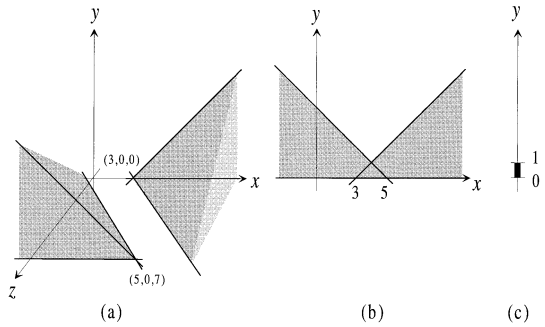

図 1　アルゴリズムの適用によって変化していくアレンジメント
Fig. 1　Process of successively applying projections onto an arrangement.

2. 基本アルゴリズムの概略

本章では，筆者らが既に提案した基本アルゴリズムの真偽判定の概要と必要な概念について述べる．

3 次元の場合を例にして述べる．一つの平面で空間をその平面の一方の空間，他方の空間，その平面の三つに分割できる．したがって，複数の平面により空間は，点，両端を含まない線分，外周を含まない多角形，外面を含まない多面体に分割できる．こうして分割されたそれぞれの部分をフェースと呼ぶ．点，両端を含まない線分，外周を含まない多角形，外面を含まない多面体をそれぞれ 0-フェース，1-フェース，2-フェース，3-フェースと呼ぶ．空間全体を平面の集合 H でフェースの集合 A に分割したとき，A を H のアレンジメントと呼ぶ．また，点を 0-フラット，直線を 1-フラット，平面を 2-フラット，空間全体を 3-フラットと呼ぶ．以上 3 次元で説明したが一般に d 次元に拡張して $(d-1)$-フェース，$(d-1)$-フラット，などという用語も使う．d 次元空間における $(d-1)$-フラットを超平面と呼ぶ．これらの定義は文献 [9] に従う．

真偽判定では，まず入力された RP 文が冠頭形でない場合，冠頭形に変換する．以降，RP 文が $F = \exists y \forall x \exists z \{y \geqq 0 \wedge [(z \geqq 0 \wedge x - y - z \geqq 3) \vee (z \leqq 7 \wedge x + y - z \leqq -2)]\}$ である場合を例に，真偽判定の概要について述べる．入力の F の母式（限定子をすべて取り除いてできる式）に含まれるすべての等式と不等式に対応する超平面すべての集合で空間を分割し，アレンジメント A を作る．F には 3 個の変数が含まれるので，3 次元で考える．F の母式 E が真になる領域（点の集合を領域と呼ぶ）は図 1 (a) の灰色の部分で表され，この領域はアレンジメント A 中のいくつかのフェースからなる．

次に，アレンジメント A の各フェース f に，f に含まれる任意の点の座標を母式に代入して得られる真偽値を割り当てる（この操作を A に E の真偽を割り当てると呼ぶ）．これによって，空間の任意の点 p に対し，p の座標を E に代入して得られる真偽値と p が含まれるフェースに割り当てられた真偽値が一致する．図 1 (a) 中の灰色のフェースが，真が割り当てられたフェースである．以降，二つのアレンジメントに対し，真が割り当てられたフェースすべての和集合の領域が一致するなら，二つのアレンジメントは等価と呼ぶ．

次いで，最も内側の限定子である $\exists z$ を消去する操作を行う．これは E を真にする z が存在する x，y の値の領域を x-y 平面上に図示する操作である．これによって得られた図が図 1 (b) である．これは，図 1 (a) に z 軸に平行な光を当ててできる真になる部分の影である．この影を表す 2 次元のアレンジメントは，まず図 1 (a) 中の 1-フラット（直線）の x-y 平面への投影（直線）すべての集合から 2 次元のアレンジメントを作り，次いで図 1 (a) の真になる部分の影に含まれるフェースにのみ真を割り当てることによって作る．

［定義 1］ 投影

点 $p = (V_1, V_2, \ldots, V_d)$ の $(v_1, v_2, \ldots, v_{d-s})$ 空間（ただし $s \geqq 0$）への投影 $\mathbf{Pr}_{(v_1,v_2,\ldots,v_{d-s})}p$ は $(V_1, V_2, \ldots, V_{d-s})$ である．領域 R の $(v_1, v_2, \ldots, v_{d-s})$ 空間への投影 $\mathbf{Pr}_{(v_1,v_2,\ldots,v_{d-s})}R$ は領域 $\{\mathbf{Pr}_{(v_1,v_2,\ldots,v_{d-s})}q | q \in R\}$ である．　□

次に内側の $\forall y$ を消去して，1 次元の真偽を割り当てたアレンジメント（図 1 (c)）を得る．$\exists$ を消去する操作が，真になる領域に座標軸に平行な光を当ててで

きる影を真とする領域を求める操作であるのに対し，$\forall$ を消去する操作は，偽になる領域に座標軸に平行な光を当ててできる影を偽とする領域を求める操作である．残った $\exists x$ を消去し，真が割り当てられた一つの頂点からなる，0 次元の真偽が割り当てられたアレンジメントを得る．ここから，式全体は真と判定される．

3. アルゴリズムの高速化

以降，**2.** のアルゴリズムに対する多面体併合を用いた高速化法について述べる．

2. のアルゴリズムでは，技術的な理由により，定義どおりに，投影前のアレンジメントを構成する $(d-2)$-フェース（d はアレンジメントの次元とする，$(d-2)$-フェースは 3 次元空間においては線分）を延長してできる $(d-2)$-フラットの投影すべての集合を求め，それらで空間を細分化することでアレンジメントの投影を作る．この方法では真偽判定をする上では細分化する必要のない空間まで細分化するので，アレンジメントの投影に含まれるフェースの数がばく大になり，計算時間がかかる(注 1)．したがって，各フェースの投影を直接求めることで，高速化する．また，投影後に，最小のフェースからなるアレンジメントを再構成することにより，次回の投影の計算を簡単にする．

3.1 フェースの直接投影と最簡アレンジメントの構成

真偽判定に必要なフェースは，真と偽の領域の境界が表現できるだけのフェースである．投影の前にアレンジメントに含まれるフェースの数を減らせば，これによって短縮される判定時間は減らしたフェースの数の指数乗に比例する．高速化法を導入したアルゴリズムでは，まず，(i) 投影前のアレンジメント（図 2 (a)）の各フェースの投影すべての集合を求め（投影）（図 2 (b)），次に，(ii) すべての重なったフェースの組合せを，一方のフェースのみに含まれるフェース，他方のみに含まれるフェース，両方に含まれるフェースに置き換える処理を行い（凹多面体分割）（図 2 (c)），その後，(iii) 隣り合う同じ真偽値が割り当てられたフェースを併合し，最小数のフェースからなる等価なアレンジメントを構成する（アレンジメント簡単化）（図 2 (d)）．併合した結果，凹多面体のフェースができるので，これもアルゴリズム中で扱えるようにする．

以降，フェース等の概念を拡張する．

［定義 2］ 閉包，境界，内部

集合 $\{x|d(x,p) < \rho\}$ を中心 p，半径 ρ の開球という（$d(x,p)$ は点 x，p 間のユークリッド距離）．S を d 次元ユークリッド空間（以降 E^d と表す）内の領域とする．S が k-フラット fl に含まれてあらゆる $(k-1)$-フラットに含まれず，S のすべての点 p に対し，p を中心として S に含まれるような，部分空間 fl 内の開球が存在するならば，S は開であるという．S が閉であるとは，S の補集合が fl 内で開であることである．領域 S の閉包は，S を含む最小の閉集合である．領域 S の内部は，S に含まれる最大の開集合である．領域 S の境界は，S の閉包から内部を引いた集合である． □

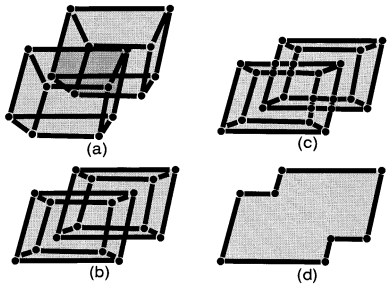

図 2 フェース同士の重なりがなくなるようにする過程
Fig. 2 Process of eliminating overlaps of faces.

点，両端を含まない線分，外周を含まない三角形，外面を含まない四面体をそれぞれ 0-凸フェース，1-凸フェース，2-凸フェース，3-凸フェースという．同様に，n-凸フェースを定義する(注 2)．同一の次元の複数の凸フェースの領域の和集合が拡張されたフェースである．ただし，拡張されたフェースは連続(注 3)であって，同一の次元の一つのフラットに含まれる．拡張されたフェースはすべて開領域である（0 次元の点は除外）．

［定義 3］ 拡張された k-フェース

拡張された k-フェースは以下の二つの規則のみを有限回適用してできるものである．

1. k-凸フェースは拡張された k-フェースである．

(注 1)：**2.** のアルゴリズムの時間計算量は **1.** で触れたとおり $r\alpha^{\beta d_n}\gamma^{d(b+1)^a}$ であるが，例えば **3.2** の高速化により定数 γ の値が減少する．

(注 2)：$P = \{\vec{p_0}, \cdots, \vec{p_k}\}$ を E^d 内の有限点集合とする．$x = \Sigma_{i=0}^{k}\lambda_i\vec{p_i}$ かつ $\Sigma_{i=0}^{k}\lambda_i = 1$ であるとき，x は P のアフィン結合であるという．$p_i \in P$ が $P - \{p_i\}$ のアフィン結合になるような p_i が存在しないとき，P はアフィン独立であるという．

E^d 内の $k+1$ 個（$k > 0$）のアフィン独立な点 $\vec{p_0}, \cdots, \vec{p_k}$ に対して，$\{y|y = \Sigma_{i=0}^{k}\lambda_i\vec{p_i}, 0 < \lambda_i < 1, \Sigma_{i=0}^{k}\lambda_i = 1\}$ で定義される領域を k-凸フェースという．また，点を 0-凸フェースという．

(注 3)：ある領域のどのような 2 点をとっても，その 2 点を結ぶ，その領域に含まれる曲線が存在するときその領域は連続であるという．

2. 同一の k-フラット上の，領域を共有する二つの拡張された k-フェースの和集合の領域は拡張された k-フェースである． □

以降，単に k-フェースと書いた場合は拡張された k-フェースを指す．

［定義 4］ サブフェース，スーパフェース

ある k-フェース f が $(k+1)$-フェース g の境界の領域に含まれるとき，f は g のサブフェースであるといい，g は f のスーパフェースであるという． □

［定義 5］ 拡張されたアレンジメント

空間全体を分割して得られるフェースの集合 S が以下の条件を満たすとき，これらのフェースすべての集合を拡張されたアレンジメントと呼ぶ：S のいずれの要素 f についても，f の境界の領域 r に対して S の部分集合 S' が存在して，S' の要素のフェースすべての和集合が r に一致すること． □

［定義 6］ 内部サブフェース

あるフェース f とフェース s に対して，s に含まれる点を中心とする，f に含まれる f と同じ次元の開球が存在するとき，s は f の内部サブフェースと呼ぶ．s が f の内部サブフェースでなく，f のサブフェースであるとき，s は f の外部サブフェースと呼ぶ． □

以降，単にアレンジメントと書いた場合は拡張されたアレンジメントを指す．

図 3 は 2 次元の，拡張されたアレンジメントの例である．図 3 のフェース B の境界は E，F，G，H，I，J，K，L，M，N，P，Q，R，S，T，U，V，W，X，Y，Z の和集合である（C，D，O が含まれないことに注意）．また，同様に E の境界は Q，R の和集合である．また，X は O のサブフェースであるので，M，X，K をまとめて一つの 1-フェースとすると，O の境界 Z，X を表

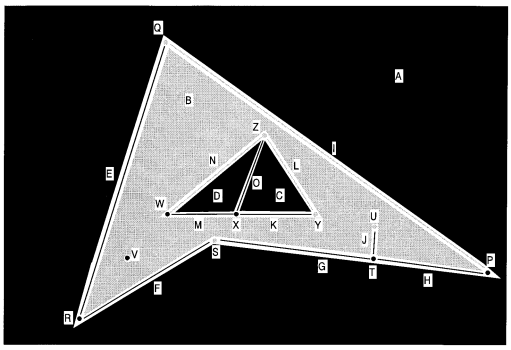

図 3 拡張されたアレンジメントの例
Fig. 3 An example of an extended arrangement.

すために必要な X がなくなってしまい，拡張されたアレンジメントではなくなる．G，T，H をまとめても，同様である．

実際のデータ構造では，アレンジメントは一つのグラフで表される．グラフの各ノードは一つのフェースに相当し，0-フェースに相当するノードには任意精度有理数のベクトルで表された座標が含まれる．フェース f がフェース g のスーパフェースであるときに限り f に相当するノードと g に相当するノード間にエッジが存在する．また，各ノードには真偽値が割り当てられている．図 3 の黒いフェースには真が，灰色のフェースには偽が割り当てられている．

付録 **1.** のアルゴリズム中のサブルーチン SIMPLIFY は前述のステップ (iii) のアレンジメント簡単化を行うが，それによって最小限のフェースの数から構成される等価なアレンジメントを構成することができる．その証明の概要は付録 **2.** で述べる．

3.2 フェースの交差判定回数の削減

3.1 で述べた高速化法を適用したアルゴリズムにおいて，フェース同士が重なっているかの判定は二つのフェースの組合せすべてに対して行う．この判定回数を平均的に減らすために次のような手法により高速化する．空間を一つの超平面 H で区切り，空間 U と他方の空間 D の二つに分割し（H は D に含める），それぞれのフェースに対し「U のみと領域を共有する」，「D のみと領域を共有する」，「両方の空間と領域を共有する」のいずれであるかあらかじめ調べておく．U のみと領域を共有するフェースと D のみと領域を共有するフェースは交差しないので，これらのフェース同士の交差判定は省略できる．空間を更に細かく分割することで，最良の場合フェース同士の交差判定の回数はフェースの数 n に対して $O(n \log n)$ 程度にまで少なくできる [10]．

また，フェースの交差判定の前処理として，それぞれのフェースを含む最小の直方体を求め，それらが交わっているかを調べる [11]．これらの直方体が交わってなければ，二つのフェースは交わっていない．

3.3 入力式の構文上の簡単化

入力式の中で，表 1 の左側のいずれかに当てはまる部分をその右側の部分に書き換える．これを，表のいずれにも当てはまらなくなるまで繰り返す．これ以外にも二つの簡単化規則がある．これによって，更にアレンジメントの分割ルーチン（**3.1** で述べたステップ (ii) を行う）に入力されるアレンジメントの次元数を

表 1　入力式の簡単化規則
Table 1　Simplifying rules for input sentences.

適用前	適用後
変数を含まない不等式	true または false
$\exists x[f(y_1,\ldots,y_k) \wedge g(x,y_{k'},\ldots,y_{k''})]$	$f(y_1,\ldots,y_k) \wedge \exists x g(x,y_{k'},\ldots,y_{k''})$
$\forall x[f(y_1,\ldots,y_k) \vee g(x,y_{k'},\ldots,y_{k''})]$	$f(y_1,\ldots,y_k) \vee \forall x g(x,y_{k'},\ldots,y_{k''})$
$\exists x f(y_1,\ldots,y_k)$	$f(y_1,\ldots,y_k)$
$\forall x f(y_1,\ldots,y_k)$	$f(y_1,\ldots,y_k)$
$\exists x[f(x,y_1,\ldots,y_k) \vee g(x,y_{k'},\ldots,y_{k''})]$	$\exists x f(x,y_1,\ldots,y_k) \vee \exists x g(x,y_{k'},\ldots,y_{k''})]$
$\forall x[f(x,y_1,\ldots,y_k) \wedge g(x,y_{k'},\ldots,y_{k''})]$	$\forall x f(x,y_1,\ldots,y_k) \wedge \forall x g(x,y_{k'},\ldots,y_{k''})]$
$x = f(y_1,\ldots,y_k) \wedge g(x,y_{k'},\ldots,y_{k''})$	$x = f(y_1,\ldots,y_k) \wedge g(f(y_1,\ldots,y_k),y_{k'},\ldots,y_{k''})$
$x \neq f(y_1,\ldots,y_k) \vee g(x,y_{k'},\ldots,y_{k''})$	$x \neq f(y_1,\ldots,y_k) \vee g(f(y_1,\ldots,y_k),y_{k'},\ldots,y_{k''})$
$\exists x\{f(x,\ldots) \wedge [g(x,y_1,\ldots,y_k) \vee h(x,y_{k'},\ldots,y_{k''})]\}$ ただし $k \neq k'$, f 中の変数の集合は $\{x,y_1,\ldots y_k\}$ または $\{x,y_{k'},\ldots y_{k''}\}$ を含まない	$\exists x\{f(x,\ldots) \wedge g(x,y_1,\ldots,y_k)\} \vee$ $\exists x\{f(x,\ldots) \wedge h(x,y_{k'},\ldots,y_{k''})\}\}$
$\forall x\{f(x,\ldots) \vee [g(x,y_1,\ldots,y_k) \wedge h(x,y_{k'},\ldots,y_{k''})]\}$ ただし $k \neq k'$, f 中の変数の集合は $\{x,y_1,\ldots y_k\}$ または $\{x,y_{k'},\ldots y_{k''}\}$ を含まない	$\forall x\{f(x,\ldots) \vee g(x,y_1,\ldots,y_k)\} \wedge$ $\forall x\{f(x,\ldots) \vee h(x,y_{k'},\ldots,y_{k''})\}\}$

減らすことができる．

3.4　高速化法を導入した判定手続きの実装

これらの高速化手法を導入した判定ルーチンを C++ 言語を使って作成した．ソースコードの行数は約 15000 行となった．判定ルーチンに使用できるような多次元凹多面体の分割アルゴリズムは筆者らの調べた限り存在せず，凹多面体の分割ルーチンはアルゴリズムを考案し，実装した．また，任意精度有理数演算ライブラリについては既存のものも使えるが，既存のものの多くは大きなビット長の演算に対して最適化されているのに対して，考えられる実用例題で用いられる，大半の演算のビット長は小さい．小さなビット長の演算に対しても最適化した任意精度有理数演算ライブラリを自作し組み込むことで，判定時間を短縮している．

4.　評価実験と考察

4.1　評 価 実 験

以下では，**3.** で述べた高速化法を導入した判定ルーチンの評価実験の結果について述べる．次の二つの評価を行った．(1) 一般の形の式に対し，どの程度のサイズまで現実的な時間内に判定できるか調べる．また，式のサイズと判定時間の関係を調べる．(2) 実用例に対する有用性を調べるために MC68030 バス上で非同期バスマスタ転送を行う時間オートマトンの適合性試験系列生成への適用を行い，評価する．各測定は Pentium III 600 MHz，メインメモリ 512 MByte の PC 上で行った．

(1-1) 入力式に含まれる異なる変数の数が 1 から 4 の場合に，不等式の数を 1 から 20 まで（変数が 4 個の場合は 15 まで）変化させ，各係数を 0 以外の分子，分母がそれぞれ 4 ビット程度のランダムな有理数にし，

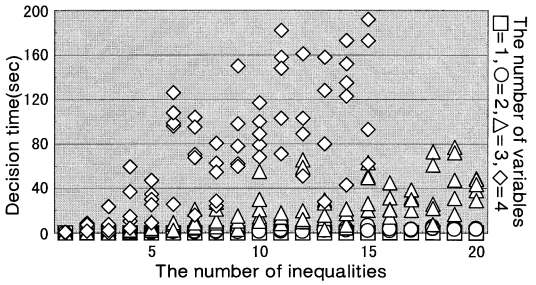

図 4　測定結果 1
Fig. 4　Experimental results (1).

ランダムな式の形で限定子は冠頭形で毎回入れ換わる並びとした．それぞれ 5 回ずつ判定時間を測定した．この結果を図 4 に示す．実験で本プログラムが使用したメモリは最大で 5 MByte 程度であった．また，紙面の都合上計測結果を省略するが，不等式の数が増えてもアレンジメントのサイズはほとんど増加しなかった．これは，入力式がランダムに生成されるため，アレンジメント中の大半のフェースを，真になる領域に含むような不等式と論理和がとられることなどによりアレンジメントのサイズが大幅に減少することがあるからである．

(1-2) 入力式に含まれる異なる変数の数が 3 から 7 の場合に，不等式の数を 1 から 20 まで（変数が 6 個以上の場合は 15 まで）変化させ，係数のうち全体の 3/10 を 0 以外の分子，分母がそれぞれ 4 ビット程度の 0 以外のランダムな有理数にし，残りは 0 とした．式の形はランダムにし，限定子は冠頭形，$\exists$, $\forall$ が毎回入れ換わる並びとした．それぞれ 5 回ずつ判定時間を測定した．計測は 300 秒で打ち切った．この結果を図 5 に示す．実験で本プログラムが使用したメモリは

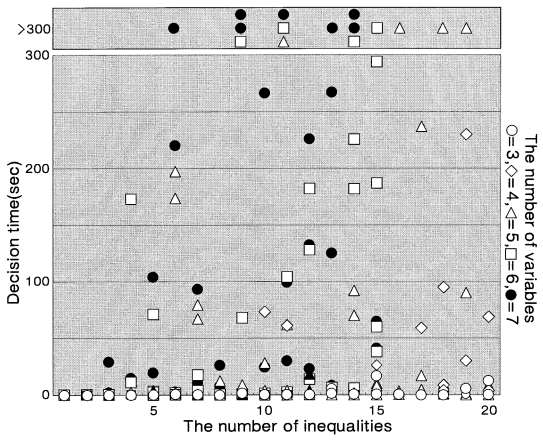

図 5　測定結果 2
Fig. 5　Experimental results (2).

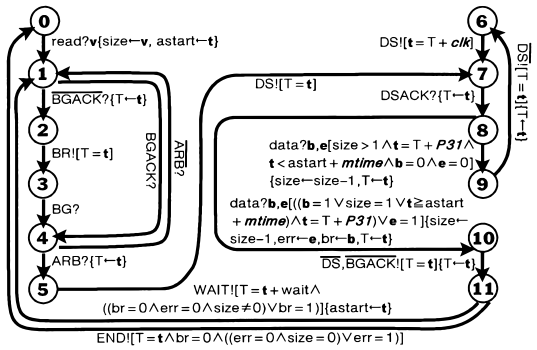

図 6　非同期バスマスタ転送を行う時間オートマトン
Fig. 6　An asynchronous busmaster transfer controller represented in a timed automaton.

最大で 8 MByte 程度であった．

(2) MC68030 バス上で非同期バスマスタ転送を行う時間オートマトンの適合性試験系列生成に適用し，評価を行った．

通信ソフトウェアの信頼性を高める一つの手法として適合性試験がある．一般に，時間オートマトンモデルでの試験では，オートマトン上で指定された遷移系列を実行しようとしても（時間制約が満たせなくなって）必ずしも実行できるとは限らない．すなわち，実時間プロトコルの試験では，テスタから IUT（Implementation Under Test）への入力は希望するタイミングで与えることができても，テスタへの出力は IUT の出力タイミングを制御できないので，そのタイミングをあらかじめ固定できない．また，その実行タイミングに依存して，後続の入出力動作の実行可能時刻が変化する可能性がある．このため，試験に際しては，その系列に含まれる出力動作が時間制約内のどの時点で行われても，与えられた試験系列を実行可能とするような入力動作の実行タイミングが存在する（must トレースが可能である）ことが望ましい．

適合性試験系列生成の手法は文献 [8] で述べられている手法を用いる．ただし，時間オートマトンのクラスは，遷移条件に線形不等式の論理積だけではなく，論理和を含むことができる．これにより，時間オートマトンのサイズを小さくすることができ，理解しやすいオートマトンを作ることができる．また，must トレース可能性判定を行う系列の長さを短くすることができる．must トレース可能性判定は定義の式を今回提案する真偽判定ルーチンを使って直接判定することによって行う．must トレース可能性の判定式において，入力タイミングと出力タイミングの変数はそれぞれ存在記号と全称記号で束縛される．判定する系列において入力と出力が交互に行われる場合は，存在記号と全称記号が交互に現れる．

図 6 は MC68030 バス上で非同期バスマスタ転送を行うプロトコルを上述のクラスのオートマトンでモデル化したものである．これは MC68030 バスに接続され，読み込むデータのサイズを受け取り，そのサイズのデータを非同期バスマスタ転送で読み込む．

適合性試験系列生成において判定する RP 文のサイズと判定時間の例として，文献 [8] で述べられている系列集合 $V.A.U_\circ$（初期状態から各状態に至って，その後状態を確認するための状態に遷移する系列）に対する must トレース可能性判定の結果を表 2 に挙げる．適合性試験系列生成において判定する他の RP 文に関してもほぼ同程度の時間で判定が可能であった．図 6 の例に対する適合性試験系列生成において判定する RP 文全体に対するべき判定時間はおよそ 20 分であった．

4.2　各高速化の効果

3. で述べた各高速化手法の効果を調べるため，すべての高速化手法を用いた場合と前述の高速化手法をそれぞれ用いなかった場合について表 2 における系列 $s10$ に対する must トレース可能性判定時間を比較した．結果を表 3 に示す．

4.3　考　　察

(1) の実験結果から，すべての係数が 0 ではない場合，変数が 4 個，不等式が 15 個程度までならおおよ

表 2　$V.A.U_0$ の各系列に対する must トレース可能性判定時間
Table 2　Decision times of must traceability for transition sequences in $V.A.U_0$.

状態	系列長	変数の数	不等式数	判定時間（秒）
$s0$	2	8	4	0.01
$s1$	3	10	8	0.08
$s2$	4	12	10	0.25
$s3$	5	9	5	0.34
$s4$	4	9	5	0.03
$s5$	4	13	11	0.3
$s6$	6	13	19	14.18
$s7$	7	14	21	17.88
$s8$	8	15	22	18.45
$s9$	8	15	19	1.59
$s10$	9	16	22	1.87

表 3　各高速化手法の効果
Table 3　Effect of each technique.

条件	判定時間（秒）
すべての高速化手法を使用	1.87
フェースの交差判定回数の削減をしなかった場合	23.24
式の構文上の簡単化をしなかった場合	> 1800
アレンジメント簡単化をしなかった場合	8.92

そ 200 秒以下で判定できるといえる．また，すべての係数のうち 7 割が 0 である場合，変数が 7 個，不等式が 15 個程度でも数百秒程度で判定できることがわかった．また，判定時間は「不等式の数」と「変数の数の指数」の積にほぼ比例しているといえ，最悪時間計算量解析の結果を大幅に下回っている．

(1-1) の実験では不等式の係数がすべて 0 でないため，入力式の構文上の簡単化がほとんど行われない．このため，「変数の数」次元のアレンジメントの分割処理が「不等式の数」回行われており，判定に時間がかかっている．付録 **1.** の各ルーチンの中で，判定時間のほとんどは，凹多面体分割を行うサブルーチン（付録 **1.** のサブルーチン DIVIDE）で費やされていた．凹多面体分割ルーチンの処理時間はだいたい「変数の数の指数」に比例する．

また，表 3 の結果から，本論文で述べた高速化手法がいずれも大きな効果をもつことがわかる(注 4)．

5. むすび

本論文では筆者らが以前提案した RP 文真偽判定アルゴリズムに対する更なる高速化手法とその評価について述べた．その高速化により，Pentium III 600 MHz において，すべての係数のうち 7 割が 0 である場合，変数の数が七つ，不等式が 15 個程度でも数百秒で判定できることがわかった．また，比較的簡単な時間制約をもつ時間オートマトンの実行可能性判定の例に対し，実際の検証で現れる，変数が 16 個，不等式が 20 個程度の RP 文を CPU 時間数秒程度で判定できるようになった．今後，他の例での評価も必要であるが，本高速化は，実際の検証に現れる式に対してはかなり有効であると思われる．

文　献

[1] 東野輝夫，北道淳司，谷口健一，“整数上の線形制約の処理と応用，” コンピュータソフトウェア，vol.9, no.6, pp.31–39, 1992.

[2] T.R. Shiple, J.H. Kukula, and R.K. Ranjan, “A comparison of presburger engines for EFSM reachability,” LNCS, vol.1427, p.280, 1998.

[3] C.W. Barett, D.L. Dill, and J.R. Levitt, “Validity checking for combinations of theories with equality,” LNCS, vol.818, pp.187–201, 1996.

[4] J.E. Hopcroft and J.D. Ullmann, Introduction to automata theory, languages and computation, Addison-Wesley, 1979.

[5] J. Ferrante and C. Rackoff, “A decision procedure for the first order theory of real addition with order,” SIAM J. Comput., vol.4, pp.69–76, 1975.

[6] 柴田直樹，岡野浩三，東野輝夫，谷口健一，“冠頭標準形有理数プレスブルガー文の真偽判定アルゴリズムの提案，” 信学論（D-I），vol.J82-D-I, no.6, pp.691–700, June 1999.

[7] 日本モトローラ株式会社，MC68030 ユーザーズマニュアル，CQ 出版，1990.

[8] 深田敦史，中田明夫，東野輝夫，谷口健一，“あるクラスの時間オートマトンに対する適合性試験系列生成の一手法，” 情処学論，vol.40, no.1, pp.85–94, 1999.

[9] H. Edelsbrunner, Algorithms in Combinatorial Geometry, Springer-Verlag Berlin Heidelberg, 1987.

[10] 柴田直樹，岡野浩三，谷口健一，“多面体分割を用いた有理数プレスブルガー文真偽判定アルゴリズムとその実装，” 信学技報，COMP99-49, 1999.

[11] S. Suri, P. Hubbard, and J. Hughes, “Collision detection in aspect and scale bounded polyhedra,” Proc. 9th Annual Symposium on Discrete Algorithms, San Francisco, 1998.

付　録

1. 以下では，高速化したアルゴリズムを挙げる．MAIN はメインルーチンである．PROJECT, DIVIDE(注 5), SIMPLIFY はそれぞれ **3.1** で述べた投影，凹多面体分割，アレンジメント簡単化を行う．判定

(注 4)：SIMPLIFY に関しては，アレンジメントのフェース数が更に増えれば効果も大きくなる．

(注 5)：サブルーチンのうち実装が最も複雑なのは DIVIDE である．DIVIDE の詳しい解説は `http://www-tani.ics.es.osaka-u.ac.jp/research/tech_report/rpres.ps` を参照のこと．

する式の形が冠頭形のみならば MAIN は PROJECT, DIVIDE, SIMPLIFY を順に繰り返し呼び出せばよいが，冠頭形でない式も判定できるようにするため，再帰的サブルーチン DECIDESUB を設けた．

Algorithm MAIN
◇ 入力：RP 文 F
◇ 出力：F の真偽値
1 $F' :=$ **3. 3** で述べた方法により F を簡単化した式;
2 $A :=$ DECIDESUB(F');
▷ A は一つの点だけで構成される
3 **return** A の原点の真偽値;
4 **end**

Algorithm INEQTOARNG
◇ 入力（不）等式 in
◇ 出力 次の性質を満たすアレンジメント A：in が真になる領域と A の真が割り当てられたフェースすべての和集合が一致する

Algorithm DECIDESUB
◇ 入力 自由変数をもつ RP 文 E
◇ 出力 次の条件を満たすアレンジメント A：A に含まれる真が割り当てられたフェースすべての和集合が，E が真になる領域と一致する
1 E の先頭にある限定子すべての並びを $Q_1, \ldots, Q_k$，E から $Q_1, \ldots, Q_k$ を取り除いた部分を M とする．
2 **if** M が単一の（不）等式 in である
3 $A :=$ INEQTOARNG(in);
4 **else**
▷ M は $L \wedge R$ か $L \vee R$ の形をしている
5 **if** M が $L \wedge R$ の形をしている **then** $q :=$ '$\forall$' **else** $q :=$ '$\exists$' **endif**
6 $A_L :=$ DECIDESUB(L);
7 $A_R :=$ DECIDESUB(R);
▷ 実際のアルゴリズムではここで A_L と A_R の次元を合わせる
8 $A :=$ DIVIDE($A_L \cup A_R$,q);
9 **endif**
▷ i は残っている限定子の数
10 $i := k$;
11 **while** $i \geqq 1$ **do**
12 $s :=$ 最も内側の連続した Q_i の個数;
13 $A :=$ PROJECT(A,s);
14 $A :=$ DIVIDE(A,Q_i);
15 $A :=$ SIMPLIFY(A);
16 $i := i - s$;
17 **endwhile**
18 **return** A;
19 **end**

Algorithm PROJECT
◇ 入力：アレンジメント A，整数 s
◇ 出力：A に含まれるフェースの（$v_1, \ldots, v_{d-s}$）空間への投影（ただし割り当てられた真偽値はもとのフェースに割り当てられた真偽値と同じ）すべての集合

Algorithm DIVIDE
◇ 入力 k 個のフェース $f_1, \ldots, f_k$ からなるフェースの集合 S，限定子 q
◇ 出力 $v_j(p)$ を，p が f_j に含まれていれば 1，そうでなければ 0 とするとき，$(v_1(p), \ldots, v_k(p))$ の同値類分割によって定義されるアレンジメント A．A に含まれる各フェース f に割り当てられている真偽値は，q が $\exists(\forall)$ なら f を含む S の要素であるフェースに割り当てられた真偽値すべての論理和（論理積）．
▷ **3. 2** で述べた高速化法はこのルーチンの中で使われる

Algorithm SIMPLIFY
◇ 入力：アレンジメント A
◇ 出力：A と等価で，最小のフェースから構成されるアレンジメント A'
1 $A' := A$;
2 **while** 併合可能な k 次元のフェース $f \in A'$ がある **do**
if A' において，f のスーパフェースが存在しなくて，閉包が f と領域を共有するフェース g が存在し，f と g に同じ真偽値が割り当てられている **then** 閉包が f と領域を共有するようなフェースすべてと f を併合する **endif**
if A' において，f のスーパフェースが二つ以上存在して，すべて同じ $(k+1)$-フラット fl に含まれていて，すべて f と同じ真偽値が割り当てられている **then**
3 f と f のスーパフェースすべてを併合する
4 **endif**
5 **endwhile**
6 **return** A';
7 **end**

2. 以下では，SIMPLIFY によって最小のフェースから構成される等価なアレンジメントが求まることの証明を示す．

（証明の概要） 以下では，(1) アレンジメントが最小のフェースから構成されていない場合に以下の性質 (P) が成り立つことを示し，次いで (2) SIMPLIFY によって，入力のアレンジメントと等価な，最小のフェースから構成されるアレンジメントを構成できることを (1) を用いて示す．

［性質 (P)］ 次の条件のうちいずれかを満たす k-フェース f が存在する．

1. f のスーパフェースは存在しない．また，f と同じ真偽値が割り当てられていて，かつ境界が f を含むようなフェースが存在する．

2. f のスーパフェースが二つ以上存在し，これらのスーパフェースすべてに f と同じ真偽値が割り当てられていてかつこれらのスーパフェースすべてが同じ $(k+1)$-フラット fl に含まれている． □

アレンジメントの次元を d とする．以下では，まずアレンジメントが最小のフェースから構成されない場合に性質 (P) を満たすことを証明する．

アレンジメントに対し，T'_k, FT_k を以下のように定義する．T'_d，FT_d を基底段階とし，順に T'_{d-1}，FT_{d-1} から T'_0，FT_0 までを帰納的に定義する(注6)．T'_d をアレンジメントの真が割り当てられたフェースすべての和集合とする．FT_d をそれぞれ d 次元空間全体の d-フラットを唯一の要素とする集合とする．

帰納的に T'_k，FT_k が定義されているとする．次のように T'_{k-1}，FT_{k-1} を定義する．FT_k の各要素 $ft_{k,0},\ldots,ft_{k,m}$ に対し，$ct_{k,j}$ を $ft_{k,j}$ と T'_k との積集合とする．また，$ct_{k,j}$ の境界を $ot_{k,j}$ とする(注7)．$T'_{k-1} = ot_{k,0} \cup \ldots \cup ot_{k,m}$ である．

$ct_{k,j}$ は k-フラットに含まれる領域であり，$ot_{k,j}$ は $ct_{k,j}$ の境界であるから，$ot_{k,j}$ に対し $(k-1)$-フラットの集合が存在し，これらすべての和集合は $ot_{k,j}$ を含む．したがって，T'_{k-1} に対して $(k-1)$-フラットの集合が存在し，これらすべての和集合に T'_{k-1} が含まれる．このようなフラットの集合（ただし，集合の要素が $(k-1)$-フェースを含まない場合は $(k-1)$-フラットではなく必要十分な次元の（複数の）フラットを代わりに要素とする集合）の中で要素数が最小のものを FT_{k-1} とする．F'_{k-1}，FF_{k-1} も真を偽と読み替え，同様に定義する．

$T_k = T'_k - T'_{k-1}$ $(k>0)$，$T_0 = T'_0$ とする(注8)．F'_k も同様に定義する．ここで，明らかに T'_k，F'_k に含まれるような $k+1$ 以上の次元のフェースは存在しない．T_k，F_k を一意に最小の連続な領域に分割することができて(注9)，これらは k-フェースである．

$T_d,\ldots,T_0$，$F_d,\ldots,F_0$ をそれぞれ最小の連続な領域に分割したものすべての集合(注10)は，アレンジメントであり，これを A とする．構成方法より A は任意のアレンジメントに対し一意に定まり，A 中のフェース数は最小である．後述の補題1から，T_k，F_k の領域の一部を k よりも大きい次元のフェースが含むような，A と等価なアレンジメントは存在しない．したがって，A と等価なすべてのアレンジメントにおいてすべての T_k，F_k の領域が k 以下の次元のフェースの集合に一致する．

A と異なる（最小のフェースから構成されない），A と等価なすべてのアレンジメント A'' に対して，A の k-フェース f が存在し，f の占める領域は A'' の二つ以上のフェースの集合 S の要素すべての和集合の領域と一致する．S が二つ以上のフェースを含む場合，拡張されたアレンジメントの条件等より S は $k-1$ 以下の次元のフェースを含む．S が $(k-1)$-フェースを含む場合，A'' は性質 (P) の2を満たし，S が $(k-1)$-フェースを含まず，$k-2$ 次元以下のフェースを含む場合，性質 (P) の1を満たす．よって，アレンジメントが最小のフェースから構成されなければ (P) を満たす (*)．

次いで，(2) を示す．アレンジメントが性質 (P) を満たすとき，SIMPLIFY の while ループ内のどちらかの if 文の条件が成立する．このとき，while ループを1回回る前後でアレンジメントは等価であり，また必ずフェース数が減少する．よって，アルゴリズムの停止性は保証され，アルゴリズムより SIMPLIFY の実行が終了したとき性質 (P) は満たされていない．(*) の対偶より，アレンジメントが (P) を満たさなければそのアレンジメントは最小のフェースから構成される．

よって，SIMPLIFY によって最小のフェースから構成されるアレンジメントが求まる． □

［補題1］ 次の性質を満たす A と等価なアレンジメント A' は存在しない：A 中のある m-フェース f に含まれる領域 g に対し，g を含むような A' の要素の k-フェース h が存在する（$m<k$）． □

紙面の都合により，補題1の証明は省略する．

（平成12年9月5日受付，13年1月4日再受付）

(注6)：アレンジメントの表す真の領域 T'_d から，順に $d, d-1, \ldots, 1$ フェースで表せる領域を除いていく．除いた結果が $T'_{d-1},\ldots,T'_0$，k-フェースで表せる領域が（後述の）T_k である．FT_k 等はこれらを定義するために補助的に使う．図3の例では，T'_2 = A∪C∪D∪E∪F∪G∪H∪J∪K∪M∪O∪P∪R∪T∪V∪W∪X，T'_1 = E∪F∪G∪H∪J∪K∪M∪P∪R∪T∪V∪W∪X，T'_0 = P∪R∪T∪V∪W．

(注7)：$ot_{k,j}$ が簡単化後のアレンジメントのフェースの集合で表せることが，定義5の条件3を満たす上で必要である点に注意．

(注8)：図3の例では，T_2 = A∪C∪D∪O，T_1 = E∪F∪G∪H∪J∪K∪M∪X，T_0 = P∪R∪T∪V∪W．T_1 に X が含まれ，T が含まれないことに注意．

(注9)：T_k，F_k は一般に不連結な領域であるが，この一部のひと続きの領域をそれぞれ一つのフェースとする．

(注10)：同じく，A = {A, C∪D∪O, E, F, G, H, J, K∪M∪X, P, R, T, V, W, B, I, N, L, Q, S, U, Y, Z}．

柴田　直樹

平8阪大・基礎工・情報中退．現在，同大大学院博士後期課程在学中．ハードウェア，ソフトウェアの形式的検証等に興味をもつ．情報処理学会会員．（現 奈良先端科学技術大学院大学情報科学研究科助手）．

岡野　浩三　（正員）

平2阪大・基礎工・情報卒．平5同大大学院博士後期課程中退．同年同大助手．平11同大学院基礎工学研究科情報数理系専攻講師，現在に至る．工博．代数的手法によるプログラム開発，分散システムなどの研究に従事．情報処理学会会員．

谷口　健一　（正員）

昭40阪大・工・電子卒．昭45同大大学院博士課程了．同年同大・基礎工・助手．現在，同大学院基礎工学研究科情報数理系専攻教授．工博．この間，計算理論，ソフトウェアやハードウェアの仕様記述・実現・検証の代数的手法及び支援システム，関数型言語の処理系，分散システムや通信プロトコルの設計・検証法等に関する研究に従事．