幾何学的手法を用いた
有理数プレスブルガー文真偽判定アルゴリズム

柴田 直樹

平成13年1月

内容梗概

本論文は，筆者が大阪大学大学院基礎工学研究科に在学中に行なった計算機科学の研究のうち，有理数プレスブルガー文真偽判定アルゴリズムに関する研究をまとめたものである．

ハードウェアやソフトウェアは複雑化の一途を辿っており，正しさを保証するための手法として形式的設計検証が注目されている．定理証明に基づく形式的設計検証において，整数や実数といった抽象データタイプの変数と加減算や大小比較といった演算を含む論理式の真偽判定手続きに帰着して検証を行なう方法が近年特に注目され，成果を上げて来ている．定理証明に基づくハードウェア，ソフトウェアの形式的設計検証を行なう手法の一つとして代数的手法が挙げられる．代数的手法によって有理数を含む仕様等が扱えると，実時間プロトコルのテストやハードウェアのタイミング検証などが扱えるようになる．しかし，これまで有理数を扱うことのできる検証系は事実上存在しなかった．有理数を扱うことのできる検証系を実現する上で課題となるのは，有理数変数を含む論理式の高速な真偽判定ルーチンの実現である．

加算を持つ有理数の理論 (有理数変数, 有理数定数,$+,-,=,<,\wedge,\vee,\forall,\exists$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) は決定可能である．

これまで，加算を持つ整数の理論 (プレスブルガー文) やそのサブクラス，RP 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．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 RP 文の真偽判定ルーチンを検証に使用した例はない．

本研究では有理数を扱うことのできる検証系の実現を目指し，(1) 従来知られていた Ferrante, Rackoff の RP 文真偽判定アルゴリズムよりも最悪時間計算量オーダの少ない同真偽判定アルゴリズムを考案した．両真偽判定アルゴリズムの最悪時間計算量オーダは二重指数であるが，この一重指数部を 1/2 の大きさに改善した．Ferrante, Rackoff のアルゴリズムが式変形に基づくアルゴリズムであるのに対し，提案したアルゴリズムは幾何学的手法に基づくアルゴリズムである．(2) また，このアルゴリズムに対して，さらに隣接する凹多面体の併合等を用いて高速化する方法を考案，実装し，評価を行なった．

アルゴリズムの概要は次のとおりである．3 次元空間では，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 3 つに分割できる．複数の平面により空間全体を，

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したものを(3次元の)アレンジメントと呼び，同様に一般のd次元に対してd次元のアレンジメントが定義できる．入力のRP文に含まれる不等式の集合からd次元のアレンジメントを作る．分割で得られる個々の部分空間(フェイスと呼ぶ)内ではRP文の母式(限定子を全て取り除いてできる式)の真偽は変わらない，という性質があるので，各フェイスに母式と真偽が同じになるように真偽を割り当てる．次に，内側から連続する同じ限定子に束縛された変数分だけ小さな次元の空間に，このアレンジメントを(限定子が存在記号なら真偽値の論理和を，全称記号なら論理積をとって)投影する．例えば，母式に対応するアレンジメントが3次元で，限定子の並びが$\exists x \forall y \exists z$である場合，アレンジメントの真が割り当てられたフェイス全てに対して，z軸に平行な光をあてて，(x, y)平面にできる影のアレンジメントを作る．これは，もとのアレンジメントに含まれる線分のフェイスを全て抜き出し，これらを(x, y)平面に投影してできる線分を延長したもので(x, y)平面を細分することでできる．同様に投影をくりかえし，最後に得られた0次元のアレンジメントに割り当てられた真偽値から式全体の真偽を得る．

また，このアルゴリズムに対して，さらに凹多面体併合等を用いて高速化する手法を考案した．提案する高速化法として，(i) 同じ真偽値が割り当てられた多面体を併合し，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少限にすること，(ii) 投影の領域を計算する際の凹多面体同士の交差判定の回数を減らすこと，(iii) 入力式の構文上の簡単化などが挙げられる．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のバス上で非同期バスマスタ転送を行な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，この例題において現れた変数の数が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 600MHz)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，全変数が出現する不等式の係数全てのうち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ることが分かった．

関連発表論文

学術論文誌・国際会議録

[1] 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", 電子情報通信学会論文誌 Vol.J82-D-I, **6**, pp.691-700, 1999.

[2] N.Shibata, K.Okano, T.Higashino, K.Taniguchi : "A decision algorithm for prenex normal form rational Presburger sentences based on combinatorial geometry," Proc. of 2nd International Conference on Discrete Mathematics and Theoretical Computer Science and the 5th Australasian Theory Symposium (DMTCS'99+CATS'99), pp.344-359, 1999.

[3] 柴田直樹, 岡野浩三, 谷口健一 : "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", 電子情報通信学会論文誌, 採録決定.

研究会発表

[4] 柴田直樹, 岡野浩三, 東野輝夫, 谷口健一 : "Tarski 算術における冠頭標準形の閉論理式の真偽判定アルゴリズムの提案", 電子情報通信学会技術研究報告, Vol.97-COMP, No.356, pp.17-24, 1997.

[5] 柴田直樹, 岡野浩三, 東野輝夫, 谷口健一 : "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とその高速化手法", 電子情報通信学会情報基礎論ワークショップ LA シンポジウム報告, pp.4.1-4.6, 1998.

[6] 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", 電子情報通信学会技術研究報告, Vol.99-COMP, No.432, pp.1-8, 1999.

Contents

1 緒論

1.1 研究の背景

ハードウェアやソフトウェアは複雑化の一途を辿っており，正しさを保証するための手法として形式的設計検証が注目されている．定理証明に基づく形式的設計検証において，整数や実数といった抽象データタイプの変数と加減算や大小比較といった演算を含む論理式の真偽判定手続きに帰着して検証を行なう方法が近年特に注目され，成果を上げてきている．定理証明に基づくハードウェア，ソフトウェアの形式的設計検証を行なう手法の一つとして代数的手法が挙げられる．筆者の所属する研究室でも従来から項書換えや後述のEPP文と呼ぶ整数上の制約論理式の真偽判定問題への帰着など代数的手法を用いて形式的検証を行なう手法を提案し，設計支援システムを構築し，評価してきた．これにより，マックスソートプログラム，KUE-CHIP2やアウトオブオーダ型パイプライン方式CPUの検証等に成功している [15,16,17,18,19,20].

代数的手法による検証系において重要なのは論理式の高速な真偽判定手続きである．EPP文の充足可能性判定問題はNP完全問題であり [32]，式のサイズが大きくなると判定時間は指数的に増大する．筆者の所属する研究グループではかつてEPP文の真偽判定の高速化を行ない，変数が数十個程度のEPP文を数十秒で判定できるようにすることに成功している [7,8,9,12].

代数的手法によって有理数を含む仕様等が扱えると，実時間プロトコルのテストやハードウェアのタイミング検証などが扱えるようになる．しかし，これまで有理数を扱うことのできる検証系は事実上存在しなかった．有理数を扱うことのできる検証系を実現する上でキーとなるのは，有理数変数を含む論理式の高速な真偽判定ルーチンの実現である．

1.2 関連研究

加算を持つ整数の理論 (整数変数, 整数定数,+,−,=,<,∧, ∨,∀,∃からなる理論) の閉論理式はプレスブルガー文と呼ばれる [28]．また，全ての変数が存在記号で束縛されたプレスブルガー文を以降EPP文と呼ぶ．加算を持つ有理数の理論 (有理数変数, 有理数定数,+,−,=,<,∧, ∨,∀,∃からなる理論) の閉論理式を有理数プレスブルガー文またはRP文と呼ぶ．

一般のプレスブルガー文の真偽判定問題の時間計算量の下界はFischerとRabinらによって非決定性 $2^{2^{cl}}$ (cは定数，lは入力の長さ) という結果が得られている [33]．また，上界に関

しては Oppen によって，定数 d に対して決定性 $2^{2^{2^{dl}}}$ の結果が得られている [30].

良く知られているプレスブルガー文真偽判定アルゴリズムとして Cooper のアルゴリズム [29] と Omega のアルゴリズム [46,47,48] がある.

Cooper のアルゴリズムの概要は次のようになる．判定するプレスブルガー文を $F = Q_n x_n ... Q_2 x_2 Q_1 x_1 E(x_1, x_2, ..., x_n)$ とする (但し，Q_k は $\forall$ か $\exists$ のいずれか，x_k は整数変数，E は線形 (不) 等式の論理結合)．次のようにして，F の最も内側の限定子を消去する．Q_1 が $\forall$ の場合は，$Q_1 x_1$ を $\neg\exists x_1 \neg$ に書き換える．x_1 に対応する新しい変数 I_{x1} を導入する．I_{x1} はとり得る範囲が $1 \leq I_{x1} \leq \lambda_{x1}$ に定まっている (λ_{x1} は E より定まるある正の整数定数)．次に，x_1 のいくつかの値 val_1〜val_k に対し，$\exists x_1 E$ を $\bigvee_1^{\lambda_{x1}} E(val_1, x2, ..., x_n) \vee ... \vee E(val_k, x2, ..., x_n)$ に書き換える．ここで，val_k は x_1 以外の変数，I_{x1} および定数の一次結合である．同様にして順に限定子を消去していき，最終的に得られる限定子を含まない式から F の真偽を判定する.

Omega のアルゴリズムは判定するプレスブルガー文の母式を積和形に直し，それぞれの積項があらわす凸多面体の集合を作る．これらに対し，Fourier-Motzkin elimination[49] により，さらに少ない次元に投影を作る処理を繰り返すことにより，限定子を消去していき，真偽を判定する.

EPP 文はプレスブルガー文のサブクラスであるため，Cooper のアルゴリズムを使って真偽判定を行なうことができる．直井らは Cooper のアルゴリズムを使った EPP 文の判定を高速化する方法を提案している [13]．これは，上記の Cooper のアルゴリズムにおいて，連続する同じ限定子は順番を入れ換えても式全体の真偽が変わらないことを利用して，判定する式を次のように変形する.

$$
\begin{array}{l}
F' = \exists I_{x1} ... \exists I_{xn} E'(I_{x1}, ..., I_{xn}) \wedge \\
\lambda_1 \quad | \quad \mu_{11} i_1 + \mu_{12} i_2 + \cdots + \mu_{1n} i_n + \nu_1 \wedge \\
\lambda_2 \quad | \quad \mu_{21} i_1 + \mu_{22} i_2 + \cdots + \mu_{2n} i_n + \nu_2 \wedge \\
\vdots \qquad\qquad\qquad \vdots \\
\lambda_n \quad | \quad \mu_{n1} i_1 + \mu_{n2} i_2 + \cdots + \mu_{nn} i_n + \nu_n
\end{array}
\tag{1}
$$

次に，ガウス消去／後退代入と同様の方法で連立一次合同式の部分の解を求め，これを E' に代入し，真になるか調べる.

この，Cooper・直井の判定法に対し，筆者の所属する研究グループでは以下のヒューリス

ティックを追加することによってさらにEPP文の判定を高速化する方法を考案した[7,8,9,12]. (i) 変数消去の度に，冗長な不等式をなるべく除去する，(ii) 構文木の根に近い位置に出現する変数から先に消去する，(iii) 変数が数個程度に減少したとき，実数上の線形計画法を利用して充足不能性を直接調べる，(iv) ガウス消去／後退代入の際に解の数が少ない順に合同式を並べる等.

有理数プレスブルガー文の真偽判定の時間計算量，領域計算量については数々の研究がなされている[34,35,36,37,38]. 時間計算量の下界はFischerとRabinらによって非決定性2^{cl}(cは定数, lは入力の長さ) という結果が得られている[33]. 上界に関しては，1975年に文献[31]でFerranteとRackoffが時間計算量$2^{2^{dl}}$(dは定数) の決定性のRP文真偽判定アルゴリズムを提案している. この時間計算量を，入力のRP文が冠頭形の場合に，含まれる不等式の個数n，異なる変数の個数d，限定子交替数[1]a，同じ限定子が続く最大の個数b，入力の式の各係数と定数の分母，分子の最大のビット数rを用いて精密に評価すると，$r5^{\epsilon d}n^{\zeta d(2b+1)^a}$($\epsilon, \zeta$は定数) となる.

実数上の線形計画問題は数学的最適化理論において研究されている基本的な問題である. 最も広く用いられている方法はDantzigによって導入されたシンプレックス法である. KleeとMintyは，シンプレックス法の最悪時間計算量は入力のサイズに対して指数関数的であることを示した[39]. Khachiyanは最悪時間計算量が多項式オーダで表されるアルゴリズムを提案し[40]，このアルゴリズムはKarmarkar, Renegar, Vaidyaによってさらに改良されている[41,42,43]. Khachiyanのアルゴリズムは入力に対応する多次元凸多面体内の一点を含む楕球を求め，アルゴリズム内の繰り返しで楕球を小さくしていくことで，解を求める.

1.3 本研究の内容

本研究では有理数を扱うことのできる検証系の実現を目指し，時間計算量が$r\alpha^{\beta d}n^{\gamma d(b+1)^a}$($\alpha, \beta, \gamma$は定数，入力が冠頭形の場合) の計算幾何学の手法を利用したRP文真偽判定アルゴリズムを考案した[4,1,2]. 前述のFerranteとRackoffのアルゴリズムが筆者の知る限り最も時間計算量の少ないRP文真偽判定アルゴリズムであるが，双方のアルゴリズムの計算量で支

[1]連続する複数の限定子において，隣り合う2つの限定子が異なる数

配的なのは二重指数の部分であり，提案するアルゴリズムの計算量は Ferrante，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．

アルゴリズムの概要は次のとおりである．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 3 つに分割できる．複数の平面により空間全体を，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したものを (3 次元の) アレンジメントと呼び，同様に一般の d 次元に対して d 次元のアレンジメントが定義できる．入力の RP 文に含まれる不等式の集合から d 次元のアレンジメントを作る．分割で得られる個々の部分空間 (フェイスと呼ぶ) 内では RP 文の母式 (限定子を全て取り除いてできる式) の真偽は変わらない，という性質があるので，各フェイスに母式と真偽が同じになるように真偽を割り当てる．次に，内側から連続する同じ限定子に束縛された変数分だけ小さい次元の空間に，このアレンジメントを (限定子が存在記号なら真偽値の論理和を，全称記号なら論理積をとって) 投影する．例えば，母式に対応するアレンジメントが 3 次元で，限定子の並びが $\exists x \forall y \exists z$ である場合，アレンジメントの真が割り当てられたフェイス全てに対して，z 軸に平行な光をあててできる影を (x, y) 平面に投影してできる影のアレンジメントを作る．これは，もとのアレンジメントに含まれる線分のフェイスを全て抜き出し，これらを (x, y) 平面に投影してできる線分を延長したもので，(x, y) 平面を細分することでできる．同様に投影をくりかえし，最後に得られた 0 次元のアレンジメントから式全体の真偽を判定する．

また，このアルゴリズムに対して，さらに凹多面体併合等を用いて高速化する手法を考案した [5,6,3]．提案する高速化法として，(i) 同じ真偽値が割り当てられた多面体を併合し，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少限にすること，(ii) 投影の領域を計算する際の凹多面体同士の交差判定の回数を減らすこと，(iii) 入力式の構文上の簡単化などが挙げられる．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[25] のバス上で非同期バスマスタ転送を行な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，この例題において現れた変数の数が 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 600MHz) 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，全変数が出現する不等式の係数全てのうち 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ることが分かった．

2 基本アルゴリズム

2.1 序言

RP 文の真偽判定アルゴリズムは，実時間プロトコル，ハードウェアの実時間タイミング検証などに利用できる [14,11]．本研究では計算幾何学の手法を利用した時間計算量が $r\alpha^{\beta d}n^{\gamma d(b+1)^a}$ (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子分母の最大のビット数，α, β, γ は定数) の冠頭形の RP 文 (以下 PRP 文) 真偽判定アルゴリズムを提案する．

Ferrante と Rackoff のアルゴリズム [31,24] が筆者の知る限り最も時間計算量の少ない RP 文真偽判定アルゴリズムであるが，双方のアルゴリズムの計算量で支配的なのは二重指数の部分であり，提案するアルゴリズムの計算量は Ferrante,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．

また，Ferrante と Rackoff のアルゴリズムと提案するアルゴリズムを実装し，真偽判定時間の測定を行った．測定の結果，だいたいにおいて Ferrante と Rackoff のアルゴリズムよりも提案するアルゴリズムのほうが高速に真偽判定が行うことができ，また限定子交替数が大きいほどその差は大きくなることが分かった．

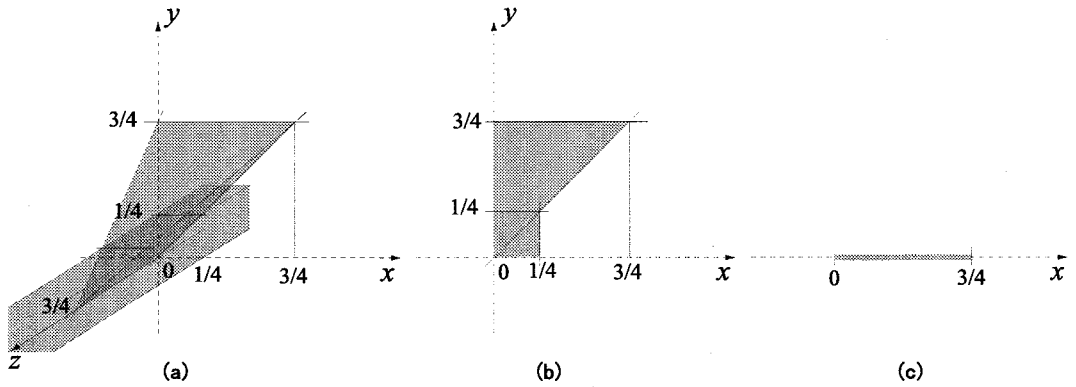

図 1 アルゴリズムの適用によって変化していくアレンジメント

2.2 基本アルゴリズムの概要

以下では，$F = \forall x \exists y \exists z \{x \geq 0 \wedge y \geq 0 \wedge [(y - x \geq 0 \wedge y + z \leq 3/4) \vee (x \leq 1/4 \wedge z \leq 1/4)]\}$ の真偽判定の様子を例に，このアルゴリズムの真偽判定の概要について説明する．また，必要な概念を適宜定義していく．

有理数変数, 有理数定数,$+, -, =, <, \wedge, \vee, (,)$ からなる論理式を PRP 式と呼ぶことにする．また，冠頭形の閉論理式である PRP 式を PRP 文と呼ぶ．

最初に入力の PRP 文の母式から空間を分割するアレンジメントを作る．まず，3 次元の場合について必要な概念を定義する．それぞれの平面で空間を平面の一方，もう一方，そして平面に含まれる空間の 3 つに分割することを考える．これにより空間は，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される．こうして分割されたそれぞれの領域をフェイスと呼ぶ．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-フェイス，1-フェイス，2-フェイス，3-フェイスと呼ぶ．空間全体を平面の集合 H でフェイスの集合 S に分割したとき，S 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 $(d-1)$-フェイス，$(d-1)$-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．PRP 文 F には 3 つの変数が含まれるので，3 次元の図で考える．F の母式である PRP 式 $E = x \geq 0 \wedge y \geq 0 \wedge [(y - x \geq 0 \wedge y + z \leq 3/4) \vee (x \leq 1/4 \wedge z \leq 1/4)]$ が真になる領域は図 1(a) の灰色の部分で表され，この領域は E 中の各不等式の表す平面で 3 次元空間を分割してできるアレンジメントのある部分集合の要素全ての和集合である．

以上のことがらを一般次元に拡張して定義すると，次のようになる．以下の定義は文献

[26] に従う.

定義 1 k-フラット

$P = \{\vec{p_0}, \cdots, \vec{p_k}\}$ を d 次元ユークリッド空間 (以降 E^d と表す) 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 のアフィン結合の集合をアフィン空間と呼ぶ. $p_i \in P$ が $P - \{p_i\}$ のアフィン結合になるような p_i が存在するとき, P はアフィン従属, そうでないときアフィン独立であるという. $k+1$ 個のアフィン独立な点からできるアフィン空間を k-フラットと呼ぶ. 特に, E^d において 1-フラットを直線, $(d-1)$-フラットを超平面と呼ぶ. □

定義 2 軸に垂直なフラット, 平行なフラット

直交座標 $v_1, ..., v_d$ に対してフラット fl が v_i 軸に垂直なある直線を含んでいるならば, fl は v_i 軸に垂直であるという. フラット fl が v_i 軸に平行なある直線を含んでいるならば, fl は v_i 軸に平行であるという. □

定義 3 k-フェイス

一般性を失うことなく v_d 軸に垂直でない超平面 h を考える. h 上の任意の点 $x = (x_1, x_2, \cdots, x_d)$ について $x_d = \eta_d + \sum_{i=1}^{d-1} \eta_i x_i$ が成り立つというような実数の組 $\eta_1, \cdots, \eta_d$ が唯一つ存在する. 点 $p = (\pi_1, \cdots, \pi_d)$ に対し π_d が $\eta_d + \sum_{i=1}^{d-1} \eta_i \pi_i$ より大きい, 等しい, 小さい場合にそれぞれ, p は h より上にある, h に乗っている, h より下にあるということにする. h^+ は h より上にある点の集合を, h^- は h より下にある点の集合を表す. 超平面の集合 $H = \{h_1, \cdots, h_n\}$ に含まれるどの超平面も全て v_d 軸に垂直でないとする. 超平面 h_i および点 p に対し $v_i(p)$ を以下のように定める.

$$v_i(p) = \begin{cases} +1 & (p \in h_i^+) \\ 0 & (p \in h_i) \\ -1 & (p \in h_i^-) \end{cases}$$

$(v_1(p), \cdots, v_n(p))$ の値が同じ点 p の集合をフェイスという. k-フラットに含まれて, $(k-1)$-フラットに含まれないフェイスを k-フェイスと呼ぶ. 特に 0-フェイスを頂点と呼ぶ. □

定義 4 アレンジメント

d 次元空間内の超平面の有限集合 H は d 次元以下の種々の次元のフェイスに空間を分割する. このフェイスの集合を H によって E^d を分割してできる (または単に H の) アレンジメントと呼ぶ. H のアレンジメント A に含まれるフェイスを A を構成するフェイスと呼ぶ. ある k-フラット fl がアレンジメント A を構成する k-フェイス f を含むとき, fl は A から

定まるフラットであるという．　□

次に，Eより得られたアレンジメントAの各フェイス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が含まれるフェイスに割り当てられた真偽値が一致する．図1(a) 中の灰色のフェイスが，真が割り当てられたフェイスである．

$\exists z$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軸に並行な光を当ててできる真になる部分の影である．この影の部分を表す真偽を割り当てたアレンジメントを真偽を割り当てたアレンジメントの投影と呼ぶ．

投影を作る操作は図1(a) 中の1-フラット (直線) の(x, y)平面への投影 (直線) 全ての集合から2次元のアレンジメントを作り，図1(a) の真になる部分の影に含まれるフェイスにのみ真を割り当てることで行う．

定義 5 点の投影

点$p = (V_1, V_2, ..., V_d)$の$(v_1, v_2, ..., v_{d-s})$空間への投影は$(V_1, V_2, ..., V_{d-s})$である．　□

定義 6 フェイスの投影

フェイスfの$(v_1, v_2, ..., v_{d-s})$空間への投影はfに含まれる点の$(v_1, v_2, ..., v_{d-s})$空間への投影全ての集合である．　□

定義 7 フラットの投影

フラットflの$(v_1, v_2, ..., v_{d-s})$空間への投影はflに含まれる点の$(v_1, v_2, ..., v_{d-s})$空間への投影全ての集合である．　□

定義 8 アレンジメントの投影

アレンジメントBの$(v_1, v_2, ..., v_{d-s})$空間への投影は，Bから定まる$(d-1-s)$-フラット ($(v_1, v_2, ..., v_{d-s})$空間での超平面) の$(v_1, v_2, ..., v_{d-s})$空間への投影fl'のうちfl'が$(d-1-s)$-フラットとなるfl'の集合のアレンジメントである．　□

一般に，あるアレンジメントAに属するフェイスfの投影に対応する集合$\mathcal{F}$があって，次の条件を満たす: $\mathcal{F}$はAの投影 **pr** A上のフェイスの集合であり，かつfの投影に含ま

れる点の集合と $\mathcal{F}$ に含まれるフェイスから構成される点の集合は一致する (2.4.3 の補題 3 参照).

定義 9 真偽を割り当てたアレンジメントの限定子 Q による投影

真偽を割り当てたアレンジメント A の $(v_1, v_2, ..., v_{d-s})$ 空間への Q による投影 $\mathbf{pr}\ A$ は定義 8 の A の $(v_1, v_2, ..., v_{d-s})$ 空間への投影 A' に属する各フェイスに次のように真偽を割り当てたものである:

$Q =$'$\exists$' **の場合** A' に属するフェイス f' に対して，A に属する真が割り当てられたあるフェイス f が存在して f の投影が f' を含むときかつそのときのみ，f' の値は真である.

$Q =$'$\forall$' **の場合** A' に属するフェイス f' に対して，A に属する偽が割り当てられたあるフェイス f が存在して f の投影が f' を含むときかつそのときのみ，f' の値は偽である.

□

入力の論理式の限定子の並びは $\forall x \exists y \exists z$ なので，同様に $\exists y$ を消去して，1 次元の真偽を割り当てたアレンジメントを得る．これは図 1(c) で表される．実際のアルゴリズムでは連続する同じ限定子で束縛された変数は一度に消去する．すなわち，図 1(a) から直接図 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並行な光を当ててできる影を真とする領域であるのに対し，$\forall$ を消去する操作は，偽になる領域に座標軸に並行な光を当ててできる影を偽とする領域である (定義 9 参照)．残った $\forall x$ を消去し，0 次元の真偽が割り当てられたアレンジメントを得る．これは偽が割り当てられた 1 つの頂点からなる．したがって，式全体は偽と判定される.

2.3 アルゴリズムの詳細

まずサブルーチンとなる ARRANGE, ASSIGN, PROJECT について順に述べる．ここで，ARRANGE はアレンジメントを表すデータ構造を作るものであり，メインルーチン MAIN と PROJECT に用いられるサブルーチンである．ASSIGN はアレンジメントと PRP 式から真偽を割り当てたアレンジメントを作る．PROJECT は真偽を割り当てたアレンジメントの投影を作るものである．2.3.4 において主ルーチンである MAIN について述べる．

2.3.1 手続き ARRANGE

d 次元空間上の超平面を表すデータ構造として，超平面上にある d 個のアフィン独立な点[2]の座標の集合を使う．

アレンジメントのデータ構造を次に述べる．

それぞれのフェイス f は定数サイズの付加情報と f に接続する次元が 1 つ上のフェイスへのポインタ群，次元が 1 つ下のフェイスへのポインタ群で表される．図 2 のアレンジメント全体のデータは図 3 のようになる．ただし，図 3 のそれぞれの長方形がフェイスをあらわす．フェイス同士を結ぶ枝は，結ばれた両方のフェイスが接続していることをあらわす．

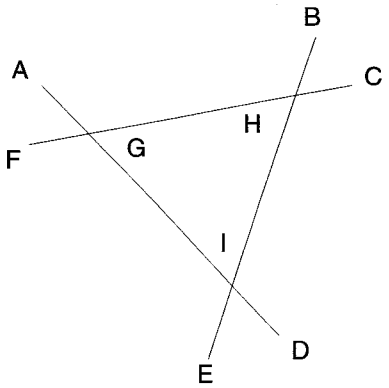

図 2 アレンジメントの例

付加情報には次のような情報が含まれる．

- フェイスの次元数
- 0-フェイスの場合は，頂点の座標
- フェイスに割り当てられた真偽値

[2]場合によってはそれらに加えて超平面上のいくつかの点

なお有界でないフェイス (図2のフェイスAG等) については十分大きな値を頂点の座標として用いる (例えばAの座標として半直線GA上の十分Gから離れた点をとる).

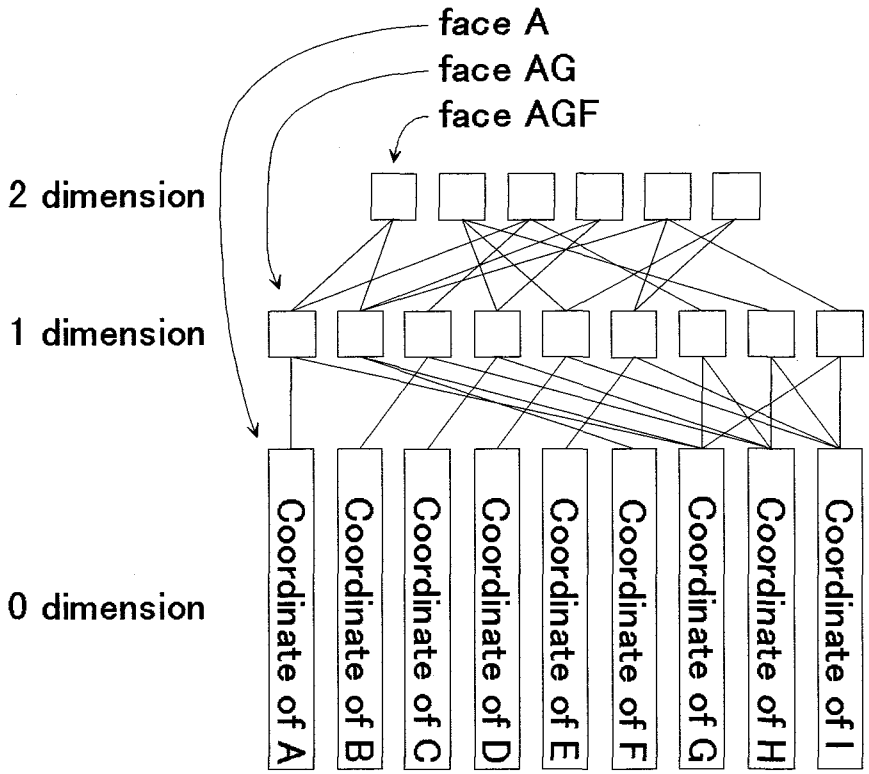

図 3 図2のアレンジメントに対するデータ構造

Algorithm ARRANGE
◇ 入力 超平面の集合 H
◇ 出力 H のアレンジメント

アルゴリズムは文献 [26] 第7章参照

2.3.2 手続き ASSIGN

サブルーチン ASSIGN は入力のアレンジメントに入力の PRP 式の真偽を割り当てるルーチンである. 以下に用いる関数 INNER は (頂点の集合で表される) フェイス f から, f の内部にある点 (例えば重心) の座標を求めるものである.

Algorithm ASSIGN
◇ 入力 与えられた入力論理式の母式 E,E の不等式の表す超平面集合を含む超平面集合のアレンジメント A
◇ 出力 A に E の真偽を割り当てたアレンジメント
▷ A に含まれるフェイスに対して E の真偽を割り当てる

```
1    for each f ∈ A
2        f に E(INNER(f)) の真偽値を割り当てる;
3    next
4    return A;
5  end
```

2.3.3 手続き PROJECT

サブルーチン PROJECT はアレンジメントの投影のアレンジメントを作るルーチンである.

以下に用いる関数 EXT はフェイスからフラットを作るものである.

Algorithm PROJECT
◇ 入力 次の条件を満たす真偽を割り当てたアレンジメント A: 全ての座標軸に対してその軸に垂直でありかつ A から定まる超平面 hp がある, 整数 s, 限定子 q
◇ 出力 A の投影
1 $S := \{\mathrm{EXT}(f) | f \in A, f$ は $(d-1-s)$-フェイス $\}$;
2 $A' := \mathrm{ARRANGE}(S)$;
3 **if** $q =$ '∃' **then** $b :=$ true **else** $b :=$ false;
▷ 各 $f' \in A'$ に対し最初に $\neg b$ に初期設定する
4 **for** each $f' \in A'$ **do** f' に $\neg b$ を割り当てる **next**
▷ A' に含まれるフェイスに対して, 真偽を割り当てる
5 **for** $f \in A$ s.t. f に b が割り当てられている
6 **for** each $f' \in A'$
7 $\vec{x_1}, ..., \vec{x_k}$ を f の頂点の投影とする.
▷ f' に含まれる適当な一点 *(INNER(f')* が **pr** f に含まれるかどうか調べる
8 **if** $(\mathrm{INNER}(f') = \sum_{m=1}^{k} a_m \vec{x_m} \wedge 0 < a_m \wedge$
$a_m < 1 \wedge \sum_{m=1}^{k} a_m = 1)$ を満たす
$a_1, ..., a_k$ が存在する
9 **then** f' に b を割り当てる. **endif**
10 **next**
11 **next**
12 **return** A';
13 **end**

2.3.4 メインルーチン MAIN

MAIN は提案するアルゴリズムのメインルーチンで, 与えられた PRP 文の真偽を判定する.

以下に用いる関数 AFFINE は不等式 in の表す超平面 h の上にあるような点であって, かつ h を表現する十分な数のアフィン独立な点の集合に in を変換するものである.

Algorithm MAIN
◇ 入力 : PRP 文 F
◇ 出力 : F の真偽値
1 入力の PRP 文の限定子の並びを $Q_1, ..., Q_d$ とする. それぞれの限定子が束縛する変数を $v_1, ..., v_d$ とする.
2 $S :=$ (F に含まれる不等式の集合) $\bigcup$ $(\bigcup_{i=1}^{d} \{v_i = 0\})$;
3 $H := \emptyset$;
4 **for** each $h \in S$
5 $t := \mathrm{AFFINE}(h)$;
6 $H := H \bigcup \{t\}$;
7 **next**
8 $A := \mathrm{ARRANGE}(H)$;
9 $A :=$ ASSIGN(A,F の母式);
▷ i は残っている限定子の数
10 $i := d$;

```
11      while i ≥ 1 do
12
13          if Q_i = '∀' then q := '∃' else q := '∀'
14
            ▷ 内側の連続した Q_i の個数を s 個に代入.
15          if q ∈ {Q_1, ..., Q_i} then
16              determine s, s.t. Q_{i-s} = q∧ for each i - s + 1 ≤ k ≤ i, Q_k ≠ q;
17          else
18              s := i;
19          endif
20
21          A := PROJECT(A,s,Q_i);
22
23          i := i - s;
24      endwhile
25
        ▷ A は 1 つの点だけで構成される
26      return A の原点の真偽値;
27  end
```

2.4 提案アルゴリズムの正当性の証明と時間計算量の解析

以降の議論では以下の性質を用いる.

性質 1

d 次元空間内で,「$m(\leq d)$ 個の超平面の交わりが $(d-m)$-フラットであること」と「それらの m 個の超平面の法線ベクトルが線形独立であること」とは等価である. □

性質 2

fl を $(v_1,...,v_d)$ 空間上の k-フラットとする. $v_{d-s+1},...,v_d$ の中で fl に平行な座標軸の集合を S とする. fl の $(v_1,...,v_{d-s})$ 空間への投影が k'-フラットとなるとき, k' に対して $k-k'=|S|$ が成り立つ. □

性質 3

fl をアレンジメント A から定まる d 未満の次元のフラットであるとする.「A から定まる, fl を含む超平面が全て j 軸に平行であること」と「fl は j 軸に対して平行であること」とは等価である. □

性質 4

fl を $(v_1,...,v_d)$ 空間上のフラットとする. $v_{d-s+1},...,v_d$ の中で fl に平行な座標軸の集合を S とする. S に含まれる任意の座標軸 v_i に対して垂直な超平面と fl の交わりを fl' とする. $(v_1,...,v_{d-s})$ 空間への fl の投影と fl' の投影は一致する. □

性質 5

d 次元のアレンジメント A から定まる $k \leq d-1$ なる k-フラット fl に対し, A から定まる $d-k$ 個の法線ベクトルが線形独立な超平面があって, fl はそれらの超平面全てに含まれている. □

性質 6

fl_1,fl_2 を $(v_1,...,v_d)$ 空間上の同次元のフラットとする. fl_1,fl_2 の $(v_1,...,v_{d-s})$ 空間への投影をそれぞれ $\mathbf{pr}\ fl_1$, $\mathbf{pr}\ fl_2$ とする. $fl_1 \neq fl_2$ かつ $\mathbf{pr}\ fl_1 = \mathbf{pr}\ fl_2$ が成り立てば, fl_1 と fl_2 の両方を含むフラットは v_{d-s+1} 軸 ,..., v_d 軸のいずれかまたはいくつかに対して平行である. □

性質 7

fl と fl' が同次元のフラットであるとする. $fl \subseteq fl'$ が成り立てば, $fl = fl'$ である. □

ここでは，各ルーチンの仕様を再掲し，それぞれの正当性の証明，時間計算量，およびその解析について述べる．

以下，明示しない限り左の表記で右の事柄をあらわす．
n　入力の PRP 文に含まれる不等式の数
d　入力の PRP 文に含まれる異なる変数の数
l　入力のサイズ
a　入力の PRP 文の限定子交替数
b　入力の PRP 文の同じ限定子が続く最大の個数

2.4.1　手続き ARRANGE の正当性と時間計算量

仕様:
入力　超平面の集合 H
出力　H のアレンジメント A

正当性の証明: 文献 [26] 参照．

時間計算量: H に含まれる各超平面は超平面を表す頂点の集合の各座標の分母，分子がそれぞれたかだか g ビットの整数で表されているとする．時間計算量及び出力のサイズは定数 γ に対し $O(g \cdot \gamma^d |H|^d)$ である [26]．

時間計算量の解析: アレンジメントに含まれるフェイスの数は $O(|H|^d)$ 個 [26]．出力の各頂点の座標は入力の超平面を表現するための各有理数を係数とする d 元連立一次方程式の解である．したがって，出力の各頂点の座標はたかだか $g \cdot \gamma^d$ ビットの整数を分母，分子に持つ有理数である．時間計算量はフェイスの数と出力の各頂点の座標のビット数の積であるから，時間計算量及び出力のサイズは $O(g \cdot \gamma^d |H|^d)$ となる．

2.4.2　手続き ASSIGN の正当性と時間計算量

仕様:
入力　PRP 式 E, E の不等式の表す超平面集合を
　　含む超平面集合のアレンジメント A
出力　E の真偽を割り当てたアレンジメント A'

正当性の証明: 定義 4，PRP 式の定義，真偽を割り当てたアレンジメントの定義から明らか．　□

時間計算量: A のサイズ l に対して時間計算量は l の多項式オーダ，出力のサイズは $O(l)$．

時間計算量の解析: 1 行目から 3 行目までのループを一回回るのに必要な時間計算量は l に対してたかだか多項式オーダである．ループを回る回数は $O(l)$ であるから，1 行目から 3 行目までの時間計算量は l の多項式オーダとなる．したがって，このアルゴリズム全体の時間計算量 l の多項式オーダである．出力のサイズは，A のサイズと等しいので，$O(l)$ である．

2.4.3 手続き PROJECT の正当性と時間計算量

仕様:

入力　各軸に垂直な超平面が定まる，真偽
を割り当てたアレンジメント A,
投影前と投影後のアレンジメントの次元の差
s, 限定子 Q

出力　A の次元を d' とするとき，A の
$d'-s$ 次元空間への限定子 Q による
投影 A'

正当性の証明: 後述の補題3より，f' が **pr** f に含まれることと f' に含まれる一点が **pr** f に含まれることは等価である．

8行目の正当性について述べる．凸多面体 C の頂点の座標ベクトルの集合を $\{\vec{x}_1,...,\vec{x}_k\}$ とするとき，ある座標 $\vec{p}$ が C に含まれることは，式(2)を満たすような $a_1,...,a_k$ が存在することと同値である．ここで，$\{\vec{x}_1,...,\vec{x}_k\}$ の代わりに，$\{\vec{x}_1,...,\vec{x}_k,\vec{y}_1,...,\vec{y}_l\}$(ただし，$\vec{y}_1,...,\vec{y}_m$ は全て C に含まれる) に対しても同様のことがいえる．

$$\vec{p} = \sum_{m=1}^{k} a_m \vec{x_m} \wedge 0 < a_m < 1 \wedge \sum_{m=1}^{k} a_m = 1 \tag{2}$$

したがって，成り立つ．その他に関しては定義4,5,9から明らか．　□

時間計算量: A を構成するフェイスの数を m，A の各頂点の座標がたかだか r ビットの整数を分母，分子に持つ有理数，α,η,θ を定数とするとき，出力 A' のデータサイズはたかだか $r\alpha^{b+1}m^{b+1}$．全体の時間計算量はたかだか $r\alpha^{\eta(b+1)}m^{\theta(b+1)}$．

時間計算量の解析: A を H のアレンジメントとする．A を構成するフェイスの数は $O(|H|^d)$ である．また，A から定まる $d-s-1$-フラットの数はたかだか ${}_{|H|}C_{s+1}$ である．$s \leq b$ より，A から定まる $d-s-1$-フラットの数はたかだか $|H|^{b+1}$ である．A の投影 **pr** A は A から定まる $(d-s-1)$-フラット全てと同じ数の超平面の集合 H' のアレンジメントであるので，**pr** A にはたかだか $O(|H|^d)^{b+1}$ 個のフェイスが含まれている．したがって，入力 A を構成するフェイスの数 $O(|H|^d)$ を m とおくと，出力 A' を構成するフェイスの数はたかだか m^{b+1} となる．

A の各頂点の座標がたかだか r ビットの整数を分母，分子に持つ有理数であるとすると，**pr** A の各頂点の座標の分母，分子のビット数はたかだか $r\alpha^{b+1}$ である．

8行目の処理にかかる時間に関しては，頂点の数がたかだか m^{b+1} 個なので，解が存在す

るかどうかを判定する式中の不等式と等式の数はたかだか $d+2m^{b+1}+1$ 個，変数の数はたかだか m^{b+1} 個になる．これは線形計画問題なので，式の判定にかかる時間は $O((m^{b+1})(d+2m^{b+1}+1)^3 r)$ となる [40].

5行目から11行目までの処理は A に属するフェイスの数が m，A' に属するフェイスの数が m^{b+1} であるから，m^{b+2} の多項式オーダとなる．

したがって，全体の時間計算量は $r\alpha^{\eta(b+1)}m^{\theta(b+1)}$ となる．

以下では補題1，2，3を導入し，証明する．ここで補題2は補題3を証明するためのさらなる補題である．

補題 1

$$\vec{x_{k+1}} = \sum_{m=1}^{k} b_m \vec{x_m} \wedge 0 < b_m < 1 \wedge \sum_{m=1}^{k} b_m = 1 \tag{3}$$

$$\vec{x} = \sum_{m=1}^{k} a_m \vec{x_m} \wedge 0 < a_m < 1 \wedge \sum_{m=1}^{k} a_m = 1 \tag{4}$$

$$\vec{x} = \sum_{m=1}^{k+1} c_m \vec{x_m} \wedge 0 < c_m < 1 \wedge \sum_{m=1}^{k+1} c_m = 1 \tag{5}$$

与えられた $\vec{x}, \vec{x_1}, ..., \vec{x_k}$ に対して式(3)を満たす $b_1, ..., b_k$ が存在するとき，「式(4)を満たす $a_1, ..., a_k$ が存在すること」と，「式(5)を満たす $c_1, ..., c_{k+1}$ が存在すること」とは等価である．

証明

$\Rightarrow$ の証明

式3と式5より，次の式が成り立つ．

$$\vec{x} = \sum_{m=1}^{k} (c_m + c_{k+1}b_m)\vec{x_m}$$

$1 \leq m \leq k$ を満たす各 m に対して $a_m = c_m + c_{k+1}b_m$ とおく．$c_m = a_m - c_{k+1}b_m$ となる．$\sum_{m=1}^{k} c_m = \sum_{m=1}^{k} a_m - c_{k+1}\sum_{m=1}^{k} b_m = 1 - c_{k+1}$ が成り立つ．したがって，$\sum_{m=1}^{k+1} c_m = 1$ が成り立つ．$1 \leq m \leq k$ を満たす m の中で，a_m/b_m が最小となる m を min とおくと，$0 < c_{k+1} < a_{min}/b_{min}$ を満たす時，$0 < c_m = a_m - c_{k+1}b_m < 1$ が満たされる．したがって，成り立つ．

$\Leftarrow$ の証明

$\Rightarrow$ の証明と同様に $a_m = c_m + c_{k+1}b_m$ とおく．$\sum_{m=1}^{k} a_m = \sum_{m=1}^{k} c_m + c_{k+1}\sum_{m=1}^{k} b_m = 1 - c_{k+1} + c_{k+1} \cdot 1 = 1$ となる．また，$0 < c_m + c_{k+1}b_m < c_1 + c_{k+1} < 1$ が成り立つ．したがって，成り立つ． □

以下補題2，3の証明の中ではアレンジメントには各座標軸に平行でない超平面が含まれているとする．また，単に投影と書いた場合 $(v_1, ..., v_d)$ 空間から $(v_1, ..., v_{d-s})$ 空間への投影を表し，明示しない限り下記の表の左の表記で右の事柄をあらわす．

A	真偽を割り当てた H のアレンジメント
pr A	A の投影
H'	**pr** A が H' のアレンジメントであるような $(v_1, ..., v_{d-s})$ 空間上の超平面の集合
f	A に属するフェイス
ext f'	あるフェイス f' を含む同次元のフラット
pr f'	A に属するあるフェイス f' の投影
pr fl'	A から定まるあるフラット fl' の投影
ext pr f	**pr** f と同次元で，**pr** A から定まってかつ **pr** f を含むフラット

補題 2 S を $|S| \leq d$ を満たす超平面の集合とする．(A):「S に含まれている超平面の法線ベクトルが線形独立であること」と (B):「$S_1 \subseteq S, S_2 \subseteq S, S_1 \neq S_2$ なる S_1 に含まれる超平面全ての交わりと S_2 に含まれる超平面全ての交わり同士は一致しないこと」は等価である．

証明

(A) $\Rightarrow$ (B) の証明: S_1 と S_2 に含まれる超平面の個数が違えば両者の交わりは次元の違うフラットになるから，一致しない．超平面の個数が一致する場合について背理法を使って証明する．両方の交わりが一致すると仮定する．そのような交わりを fl とする．$S_1 \bigcup S_2$ に含まれる超平面の交わりも fl になる．$S_1 \neq S_2$ より S_1，S_2 の一方には他方に含まれていない超平面が含まれているから，$S_1 \bigcup S_2$ に含まれている超平面の数は S_1 または S_2 に含まれている超平面の数よりも多い．したがって，$S_1 \bigcup S_2$ に含まれる超平面の交わりは S_1 に含まれる超平面の交わりよりも次元の低いフラットになるはずである．これは仮定に矛盾する．

(A) $\Leftarrow$ (B) の証明: (B) $\wedge \neg$ (A) から矛盾が生じることを証明する．$S = \{h_1, h_2, ..., h_{|S|}\}$ とする．(B) の S_1 に h_1 を，S_2 に $h_1 \bigcap h_2$ を代入すると，$h_1 \neq h_1 \bigcap h_2$ がいえる．$fl_i = h_1 \bigcap ... \bigcap h_i$ とすると，同様にして，$1 \leq i \leq |S| - 1$ に対して $fl_i \neq fl_{i+1}$ がいえる．fl_i と fl_{i+1} が同じ次元のフラットであると仮定すると，性質1より，$fl_i = fl_{i+1}$ となって矛盾す

るので，fl_i と fl_{i+1} が同じ次元のフラットではない．また，fl_i の定義と $fl_{i+1} \neq fl_i$ の仮定より $fl_{i+1} \subset fl_i$ であるから，fl_{i+1} は fl_i より次元の低いフラットである．したがって，fl_1 は $d-1$-フラット，fl_2 は $d-2$-フラット,...,$fl_{|S|}$ は $d-|S|$-フラットとなる．性質1より $fl_{|S|}$ は線形独立であることになって，¬ (A) に矛盾する． □

補題 3 各座標軸に平行でない超平面がアレンジメント A から定まる場合に，A を構成する各フェイス f の $(v_1,...,v_{d-s})$ 空間への投影 **pr** f と **pr** A を構成するフェイスの集合 S に対し **pr** f と (点の集合として) 一致する S の部分集合が存在する．

証明 (S1) から (S5) までの5つのステップに分けて証明を行う．

(S1) fl を A から定まるフラットとする．**pr** fl を $i(\leq d-1-s)$-フラットとする．**pr** fl = **pr** c を満たす A から定まる i-フラット c があることを示す．

S を v_{d-s+1} 軸 ,..., v_d 軸の中で fl に平行な座標軸の集合とする．性質2より fl は $i+|S|$-フラットである．アレンジメントには各座標軸に垂直な超平面が含まれているので，S の各要素 j に対し，fl は $j=0$ で表される超平面と交わっている．そのような超平面の集合を P とする．性質3より，fl を含み，A から定まる $(d-1)$-フラット α は全て S の各要素と平行である．したがって，α または P に含まれる $(d-1)$-フラットの法線ベクトルは線形独立である．P に含まれる全ての超平面と fl の交わり c が存在し，性質1より c は i-フラットである．c を含み，かつ A から定まる $(d-1)$-フラットの集合を α' とおく．$\alpha' = \alpha \bigcup P$ である．また，v_{d-s+1} 軸 ,..., v_d 軸の中で c と平行な座標軸はない．性質4より，**pr** fl は **pr** c に一致する．

(S2) フラット c を含む $(d-1)$-フラットのうちうまく選んだ s 個の交わりは v_{d-s+1} 軸 ,..., v_d 軸のいずれにも平行でない $(d-s)$-フラットであることを示す．

v_{d-s+1} 軸 ,..., v_d 軸の中で c と平行な座標軸はないので，性質3から，次のことが言える: v_{d-s+1} 軸 ,..., v_d 軸のなかのそれぞれの軸 j に対して，c を含み，かつ A から定まる超平面 hp が存在し，hp は j に平行ではない．そのような hp 全ての集合を β' とおく．v_{d-s+1} 軸 ,..., v_d 軸は s 本の軸なので，β' は高々 s 個の超平面の集合である．フラット c が i-フラットであることと性質5からフラット c を含む $(d-1)$-フラットは $d-i$ 個ある．仮定より $d-i > s$．したがって，フラット c を含む $(d-1)$-フラットのうち β' を含む s 個の交わり c' は v_{d-s+1} 軸 ,..., v_d 軸のいずれにも平行でない．これらの s 個の $(d-1)$-フラットの集合を β とおく．c' が求めるフラットである．

(S3) フラット c を含む $(d-1-s)$-フラットのうち (うまく選んだ)$d-i-s$ 個のフラットの各投影が，$\mathbf{pr}\ A$ から定まる法線ベクトルが線形独立な $d-i-s$ 個の $(d-1-s)$-フラットとそれぞれ一致することを示す．

c を含む $d-i-s$ 個の $(d-1-s)$-フラットは次のようにして求めることができる．定義から $\beta \subset \alpha'$ である．$\gamma = \alpha' - \beta$ とおく．α'，β はそれぞれ $d-i$ 個，s 個の超平面の集合であるから，γ には $d-i-s$ 個の超平面が含まれる．γ に含まれる各超平面と β に含まれる各超平面との交わりは v_{d-s+1} 軸 ,...,v_d 軸のいずれにも平行でない $(d-1-s)$-フラットであり，これらのフラットの集合を δ とおく．β に含まれる各超平面との交わりは $(d-s)$-フラットであり，γ の要素の各超平面と β に含まれる全ての超平面は法線ベクトルが線形独立なので，性質2より δ には $d-i-s$ 個のフラットが含まれる．α' に含まれる超平面の法線ベクトルが線形独立であることから δ の要素は (d 次元空間上で) 重ならない．δ に含まれるフラットの投影の法線ベクトルが線形独立であることは次のようにして分かる．α' に含まれる超平面の法線ベクトルが線形独立であることから，δ に含まれる任意の二つのフラットの交わりは全て一致しない．また，δ に含まれるフラットの交わりの投影も全て一致しない．なぜなら，もし δ に含まれるフラット同士の交わり fl_1 と fl_2 に対して $\mathbf{pr}\ fl_1 = \mathbf{pr}\ fl_2$ が成り立つと仮定すると，明らかに fl_1 と fl_2 は同じ次元のフラットであり，$fl_1 \neq fl_2$ であるから，性質6より fl_1 と fl_2 を含むフラットすなわち β に含まれる全ての超平面の交わりが v_{d-s+1} 軸 ,...,v_d 軸のいずれかに平行であることになる．これは矛盾する．したがって，補題2より，δ に含まれるフラットの投影の法線ベクトルは線形独立である．

(S4) A を構成する各フラット fl の投影 $\mathbf{pr}\ fl$ に対し $\mathbf{pr}\ A$ から定まるフラットの集合の中に $\mathbf{pr}\ fl$ と (点の集合として) 一致するあるフラットが存在することを示す．

H' の要素は全て $(d-1-s)$-フラットである．δ の各要素は A から定まる $(d-1-s)$-フラットであり，δ の各要素の投影は全て $(d-1-s)$-フラットであるから，δ の各要素の投影は全て H' の要素である．(δ の各要素を定義するときに使った) γ に含まれる全ての超平面と β に含まれる全ての超平面の集合は α' であり，α' に含まれる超平面全ての交わりは c であるから，δ に含まれる $(d-1-s)$-フラットの交わりは i-フラット c であり，δ に含まれるフラットの投影の法線ベクトルが線形独立であることから，δ に含まれるフラットの投影の交わり c' は i-フラットである．$\mathbf{pr}\ c$ は δ に含まれるフラットの投影全てに含まれるので，$\mathbf{pr}\ c \subseteq c'$ がいえる．性質7より，$c = c'$ がいえる．したがって，δ に含まれる各フラットの

投影の交わりが **pr** c に一致する．したがって，**pr** c が **pr** A から定まることがいえる．したがって，(S4) が示された．

(S5) A **を構成する任意のフェイス** f **の投影** **pr** f **に対し** **pr** f **に (点の集合として) 一致する，** **pr** A **を構成するフェイスの集合が存在することを示す．**

f の各サブフェイス f' のうち **pr** $f' \subseteq$ **pr** f を満たさないものの集合を ϵ' とする．$\epsilon =$ **cl pr** $f -$ **pr** f とする．

フェイスの境界は開であること，f の各サブフェイス f' に対して **pr** $f' \subseteq$ **cl pr** f であることから「**pr** $f' \bigcap$ **pr** $f = \emptyset$」または「**pr** $f' \bigcap$(**cl pr** $f -$ **pr** $f) = \emptyset$」のいずれかがなりたつ．このことと，**pr** f に含まれる点および f の全てのサブフェイスの投影に含まれる点からなる集合は **cl pr** f に点の集合として一致することなどから $\epsilon = \epsilon'$ である．

$\epsilon = \epsilon'$ なる ϵ' が存在することと (S4) までで証明したこと及び **pr** f が凸であることから題意が成り立つ． □

なお補題 3 中に用いられる集合の直感的な意味をまとめると，次のようになる．

c	**pr** $fl =$ **pr** c となる i-フラット
α'	c を含み，A から定まる $(d-1)$-フラットの集合
β	c' すなわち (S2) で求めるフラットを含み， A から定まる $(d-1)$-フラットの集合
δ	δ に含まれる $(d-1-s)$-フラットの投影は **pr** c を 含み，**pr** A から定まる

2.4.4 提案アルゴリズム全体の正当性と時間計算量

仕様:

入力　PRP 文 F
出力　F の真偽値

正当性の証明: MAIN の正当性は下の項目が証明されれば十分である．

定義 10 関数 $\mathcal{VAL}$ 関数 $\mathcal{VAL}$ は真偽を割り当てたアレンジメント A と A に属する一点の座標 p を引数に取り，p が含まれる A に属するフェイスに割り当てられた真偽値を返す．

定義 11 関数 $\mathcal{FML}$ A を真偽を割り当てたアレンジメント，b をブール変数，$Q_1, ..., Q_i$ を限定子とする．関数 $\mathcal{FML}$ を次のように定義する．

$$\mathcal{FML}(A, (Q_1, ..., Q_i))$$

$$= \quad Q_1 v_1, ..., Q_i v_i \mathcal{VAL}(A, (v_1, ..., v_i))$$

命題 1 最初に MAIN の 12 行目に到達したときに
$\mathcal{FML}(A,(Q_1,...,Q_i))$ の真偽が F の真偽と一致している，かつ A に各座標軸に垂直な超平面が含まれていること．

命題 2 MAIN の 12 行目において
$\mathcal{FML}(A,(Q_1,...,Q_i))$ の真偽が F の真偽と一致している，かつ A に各座標軸に垂直な超平面が含まれていること．

命題 3 最初に MAIN の 25 行目に到達したとき A が一つの点だけで構成され，$i=0$ であり，t が入力の PRP 文の真偽と一致していること．

命題 4 while 文からいつか抜け出ること．

命題 1 の証明 アレンジメント A 内の，同一のフェイスに属する座標では，E の真偽が変わらないことと，関数 $\mathcal{FML}$ の定義，MAIN の 2 行目で A に各軸に垂直な超平面を付加していることから，題意は成立する． □

命題 2 の証明 a_1,a_2 をそれぞれ 12 行目，22 行目の時点の $\mathcal{FML}(A,(Q_1,...,Q_i))$ の値とする．サブルーチン PROJECT の正当性と補題 4 より $a_1=a_2$ が成り立つ．

各座標軸に垂直な超平面をアレンジメントから定まるフラットとして持つアレンジメントの投影は各座標軸に垂直な超平面をアレンジメントから定まるフラットとして持つことは補題 3 より明らか．したがって，題意が成立する． □

補題 4 各軸に垂直な超平面を定めるアレンジメント A の限定子 Q による $(v_1,...,v_{i-s})$ 空間への投影を **pr** A とすると，任意の $V_1,...,V_{i-s}$ に対し下の式が成り立つ．

$$\begin{aligned}&\mathcal{VAL}(\mathbf{pr}\ A,(V_1,...,V_{i-s}))\\ =\ & Q\ v_{i-s+1},...,Q\ v_d\\ & \mathcal{VAL}(A,(V_1,...,V_{i-s},v_{i-s+1},...,v_d))\end{aligned}$$

証明 補題 3 より，**pr** f が含む **pr** A 上のフェイスの集合を $\mathcal{F}$ とすると，$\mathcal{F}$ に含まれるフェイスに含まれる点全ての集合と，**pr** f は一致する．Q ='∃' の場合について述べる．アルゴリズムより f に真が割り当てられていれば，$\mathcal{F}$ に含まれるフェイスには真が割り当てられるので，真が割り当てられたフェイス f に含まれる点 p に対して，**pr** p は真が割り当てられたフェイスに含まれている．f に偽が割り当てられていれば，$\mathcal{F}$ に含まれる各フェイ

ス f' に真が割り当てられる場合と偽が割り当てられる場合がある．f' に偽が割り当てられる場合は $f' \subseteq \mathbf{pr}\ f''$ となるような A を構成するフェイス f'' には全て偽が割り当てられている．f' に真が割り当てられている場合，ある f'' に対して真が割り当てられている．$\mathbf{pr}\ p'$ が偽が割り当てられたフェイスに含まれていれば，$\mathbf{pr}\ p'' = \mathbf{pr}\ p'$ となるような p'' は全て偽が割り当てられたフェイスに含まれている．

また，Q ='∀' の場合にも同様に証明できる．

以上により，題意が成り立つ． □

命題 3 の証明 最後に 22 行目を通った時点で A は 0 次元である． □

命題 4 の証明 22 行目の時点で $s > 0$ である．したがって，while 文を一回回る毎に必ず i の値は減少する．よって，題意は示される． □

時間計算量: 入力の式の各係数と定数の分母，分子のビット数 r，定数 α, β, γ に対して，$r \cdot \alpha^{\beta d} n^{\gamma d (b+1)^a}$．

時間計算量の解析: 5 行目の時間計算量はたかだか l に対して多項式オーダである．8 行目の時間計算量および代入される A のサイズは $O(r \cdot \gamma^d n^d)$ である．9 行目の ASSIGN の返り値 A のサイズは $O(l \cdot \gamma^d n^d)$ となる．

11 行目から 24 行目までのループの時間計算量について考える．12 行目の時点での A を構成するフェイスの数を m，各頂点の座標がたかだか r' ビットの整数を分母，分子に持つ有理数であらわされているとする．21 行目の時間計算量は定数 α, η, θ に対し，$O(r' \alpha^{\eta(b+1)} m^{\theta(b+1)})$ であり，返り値の A のサイズは $r' \alpha^{b+1} m^{b+1}$ である．ループに入る前の A に含まれるフェイスの個数を m_s，A の各頂点の座標の分母，分子のビット数を r_s とおくと，ループを回る回数は a であるので，a 回目のループの 21 行目の PROJECT の時間計算量は $r_s \alpha^{\eta ab} m_s^{\gamma d (b+1)^a}$ である．$d = O(ab)$ より，全体の時間計算量は定数 α, β, γ に対して，$r \alpha^{\beta d} n^{\gamma d (b+1)^a}$ となる．

□

2.5 関連研究

以降 Ferrante と Rackoff のアルゴリズム [31] とその時間計算量について解説する.

真偽を判定する PRP 文を F, F の母式を E とする. F の限定子の並びを $Q_1v_1\ Q_2v_2...Q_dv_d$ とする. E に含まれる v_d を含む各不等式は x_d **op** t_i の形で表すことが出来る. ただし **op** は $<, =, >$ のいずれかであり, t_i は適当な有理数 c_j に対し

$$t_i = \sum_{j=1}^{d-1} c_j v_j$$

という形の式である. $v_1, ..., v_{d-1}$ へのある与えられた値に対するこのような各不等式を x_d **op** $t_i, 1 \leq i \leq n$ とおき, $t_1 \leq t_2 \leq, ..., \leq t_n$ と仮定する. $t_i < v_d < t_{i+1}$ を満たす全ての v_d に対し, 各アトムは同じ真理値を持つ. すなわち, E の真偽は v_d がこの区間に属する限り v_d の値とは独立に定まる. このことから次のことが導かれる. $t_1, ..., t_n$ は実数軸を有限個の部分に分割し, E の真偽は v_d が属する部分のみによって定まり, v_d の値そのものにはよらない. したがって, 図 4 に示すような有限個の部分のそれぞれから v_d の値として一つを選んで試すことにより, E の真偽を調べることができる.

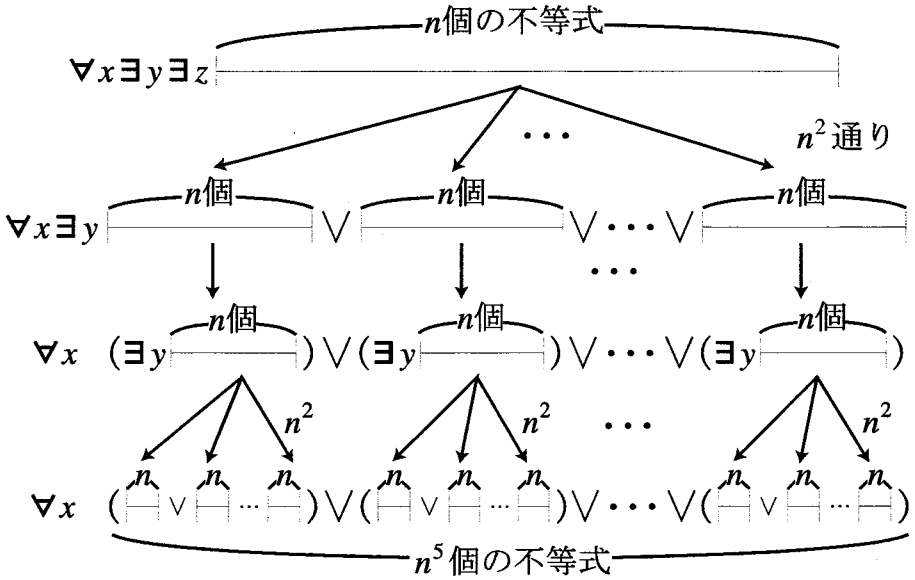

図 4 限定子消去に伴う不等式の数の増加

実際には $v_1, ..., v_{d-1}$ の値が変わるので, $t_1, ..., t_n$ の順序関係がどうであるかは分からない. しかし, 全ての i, j に対し $x_d = (t_i + t_j)/2$ とおき, さらに $x_d = \pm\infty$ とおいて調べるこ

とにより, t_i の間の順序がどうであろうとも E の各区間を代表する値と t_i 自身とを x_d に与えることができるのは確かである.

以上のことから Q_d ='∃' の場合, $\exists v_d E$ を

$$E' = \bigvee \begin{array}{lll} v_d & = & t_i \\ v_d & = & (t_i + t_j)/2 \\ v_d & = & \pm\infty \end{array} E$$

によって, すなわち F の中の v_d に代入を行って得られる $n(n+1)/2+2$ 個の項の論理和によって置き換えることが出来る. Q_d ='∀' の場合も $\vee$ を $\wedge$ にかえるだけで上と同様の置き換えが可能である. $Q_d = Q_{d-1}$ の場合, 次のようにして二つの限定子を一度に消去することができる.

E' に含まれる各不等式は上と同様に x_{d-1} **op** t'_i の形で表すことができ, t'_i は適当な有理数 c_j に対し

$$t'_i = \sum_{j=1}^{d-2} c_j v_j$$

という形の式である. $Q_d = Q_{d-1}$ ='∃' の場合, $\exists v_{d-1}\exists v_d E$ を

$$E'' = \bigvee \begin{array}{lll} v_d & = & t_i \\ v_d & = & (t_i + t_j)/2 \\ v_d & = & \pm\infty \end{array} \bigvee \begin{array}{lll} v_{d-1} & = & t'_i \\ v_{d-1} & = & (t'_i + t'_j)/2 \\ v_{d-1} & = & \pm\infty \end{array} E$$

によって置き換えることができる. E に不等式が n 個含まれているとすると, E' は $n[n(n+1)/2+2]$ 個のアトムを, すなわち $n \geq 3$ の場合高々 n^3 個の不等式を含む. 同様に, E'' は高々 n^5 個の不等式を含む.

上記の操作を繰り返すことにより限定記号と変数を全て取り除くと, 最終的に論理演算子,$=$,$<$ および定数だけからなる式が得られる. そこに現れる定数は分母, 分子が各々 $5^d r$ ビットの整数であるような有理数であり, また, a を限定子交替数, b を同じ限定子が続く最大の個数とすると, 不等式の個数は高々

$$\underbrace{(\ldots((n^{2b+1})^{2b+1}\ldots)^{2b+1}}_{a\text{times}} = n^{(2b+1)^{a+1}}$$

である. したがって, 真偽判定にかかる時間計算量は $r \cdot 5^{\epsilon d} n^{\zeta d(2b+1)^a}$ となる.

2.6 評価実験と考察

提案するアルゴリズムによる判定ルーチンと Ferrante,Rackoff のアルゴリズムによる判定ルーチンのそれぞれを実装し，評価実験を行った．それぞれのアルゴリズムの実装に際して，インプリメント上の最適化は一切行わなかった．各測定は SunOS 5.6 の動作するワークステーション (UltraSPARC-II 248MHz，メモリ 512M バイト，スワップ領域 1G バイト) 上で行った．

提案するアルゴリズムの 4 章で述べた時間計算量オーダが実際の判定時間とどれほど一致するか，また Ferrante,Rackoff のアルゴリズムとの実際の判定時間の差について調べた．

双方のアルゴリズムは入力の式中に出てくる各不等式の形と限定子にのみ計算量が依存し，各不等式がどのように論理演算子で結合されているかには依存しないという特徴がある．したがって，入力の PRP 文の母式の形をいくつかの不等式の論理積に限定しても，そうでない場合と同じ結果が得られる．

最初に，入力の PRP 文の係数と定数のビット長のみを変化させ，判定時間の変化を調べた．

実験は下の式の係数と定数 A から L までの分子，分母それぞれのビット長を 5 ビットから 30 ビットまで変化させ，判定時間を測定した．係数と定数 A から L をランダムで生成した 5 個の PRP 文に対して判定時間を測定した．結果を図 5 に示す．

$$\exists x \forall y \{ Ax + By < C \wedge Dx + Ey < F \wedge \\ Gx + Hy < I \wedge Jx + Ky < L \}$$

実験の結果から，双方のアルゴリズムの判定時間は入力の PRP 文の係数と定数のビット長にほぼ比例しており，これは解析により求めた時間計算量から予想される結果と一致している．次に，入力の PRP 文の限定子交替数と，含まれる変数の数，不等式の数を変化させ，判定時間の変化を調べた．母式は不等式の単純な論理積であり，係数と定数のビット長は 7 ビットに統一した．

まず，最も外側の変数のみ $\forall$，その他の変数は全て $\exists$ で束縛された場合 (限定子交替数 =1) について，変数の数を 2 から 5 まで，不等式の数を 1 から 4 まで変化させ，判定時間を測定した．各係数と定数を 10 進数で分母二桁，分子二桁のランダムな値として，5 回測定した平均値を図 6 に示す．

次に，外側から二番目の変数のみが $\forall$，その他の変数は全て $\exists$ で束縛された場合 (限定子

交替数 =2) の場合について，変数の数を 3 から 5 まで，不等式の数を 1 から 4 まで変化させて判定時間を測定した．上の場合と同様に 5 回の測定値の平均を図 7 に示す[3]．なお，限定子の並び方，変数の数，不等式の数が同じであれば，係数と定数が変わっても，いずれのアルゴリズムでも判定時間は数倍程度しか変わらないようである[4]．

双方のアルゴリズムで変数の数と不等式の数が増えるに従い判定時間が指数的に増えているが，提案するアルゴリズムでは判定時間の増え方が Ferrante,Rackoff のアルゴリズムよりも少ないといえる．

また，双方のアルゴリズムの判定時間に関して次のようなことが観測された [1]．

まず，入力の PRP 文の各不等式に現れる変数の数が少ない場合は，Ferrante,Rackoff のアルゴリズムによって高速に判定できる．これは，Ferrante,Rackoff のアルゴリズムの性質からも分ることである．

次に，限定子交替数や同じ限定子が連続する最大の個数が同じ場合でも内側の限定子が交替する間隔が短い[5]といずれのアルゴリズムでも判定に時間がかかった．例えば，変数の数が 4，限定子交替数が 1，不等式の数が 3 の場合，提案するアルゴリズムでの判定時間が約 1500 秒，Ferrante,Rackoff のアルゴリズムだと 3000 秒を越えた[6]．これは，次の理由によるものであると思われる．外側の限定子が交替する間隔が短いと，次元の数が急激に減るので変数の数が多い式や次元の多いアレンジメントを処理する回数が少ない．変数の数が少ない式や，次元の少ないアレンジメントを処理する過程で，全く同じ不等式やフェイス，フラットが数多く出現し，重複する不等式やフェイス，フラットは一回だけ処理されるため，その分計算時間が減る．したがって，外側の限定子が交替する間隔が短いと計算時間が減る．

以上の実験から次のことが言える．まず，アルゴリズムの実際の判定時間の変化は計算で求めた結果とだいたい一致する．また，外側の限定子が交替する間隔が短い場合には提案するアルゴリズムの方が Ferrante,Rackoff のアルゴリズムよりも短い時間で判定できる．

---

[3]全ての組合せについて測定を行ったが，それぞれについて判定時間が 3000 秒を越えた時点で測定を打ち切った．また，その場合は結果は載せていない

[4]ただし，係数と定数が全て同じであるなど，特殊な係数と定数の組合せに対してはこの限りではない

[5]例えば ∃∃∀∃∀ という限定子の並びは，∀∃∀∃∃ よりも内側の限定子が交替する間隔が短い．

[6]係数と定数を変えて 5 回測定した平均値

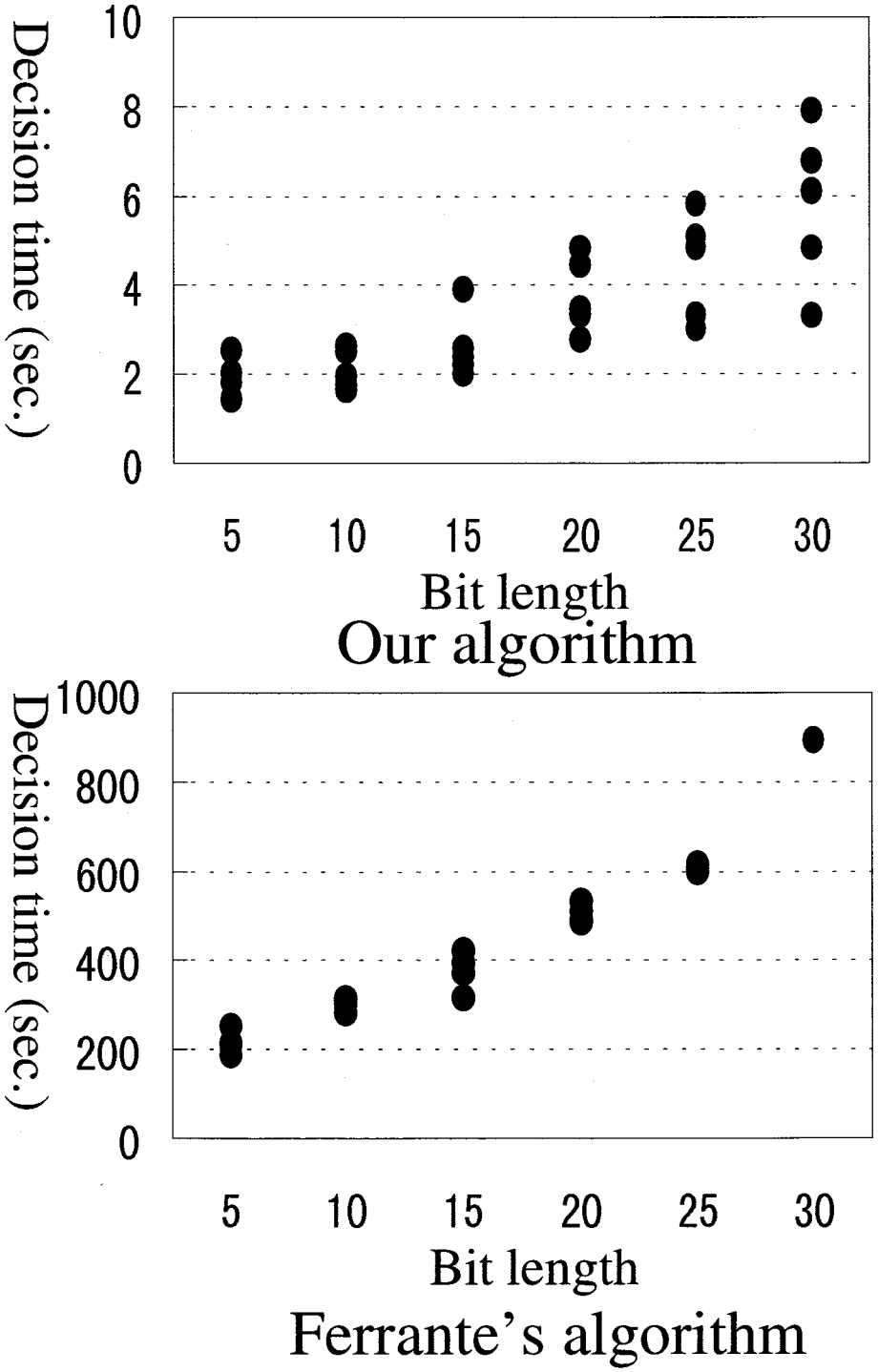

図 5 入力の係数と定数のビット長と判定時間

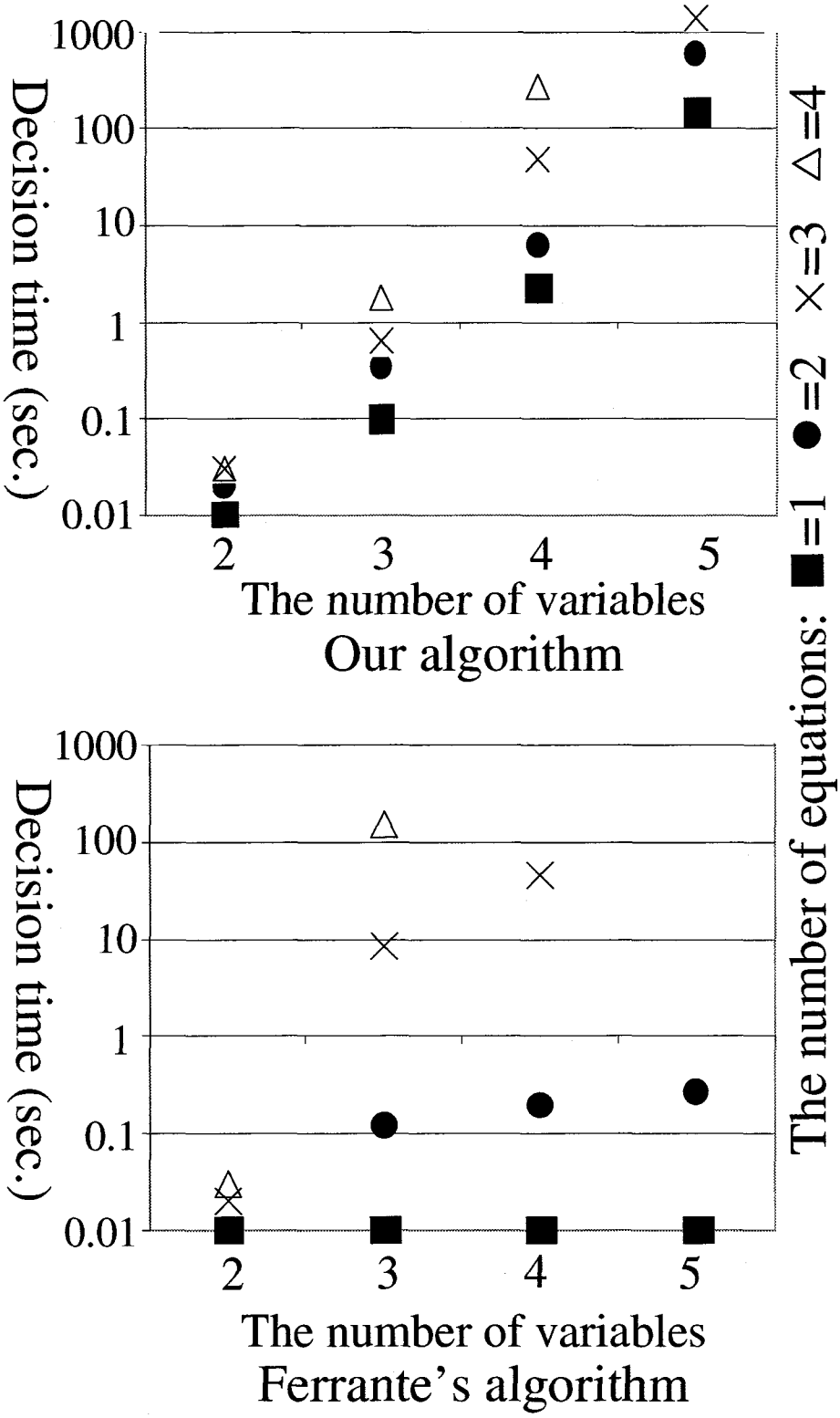

図 6 限定子交替数が 1 の場合の判定時間

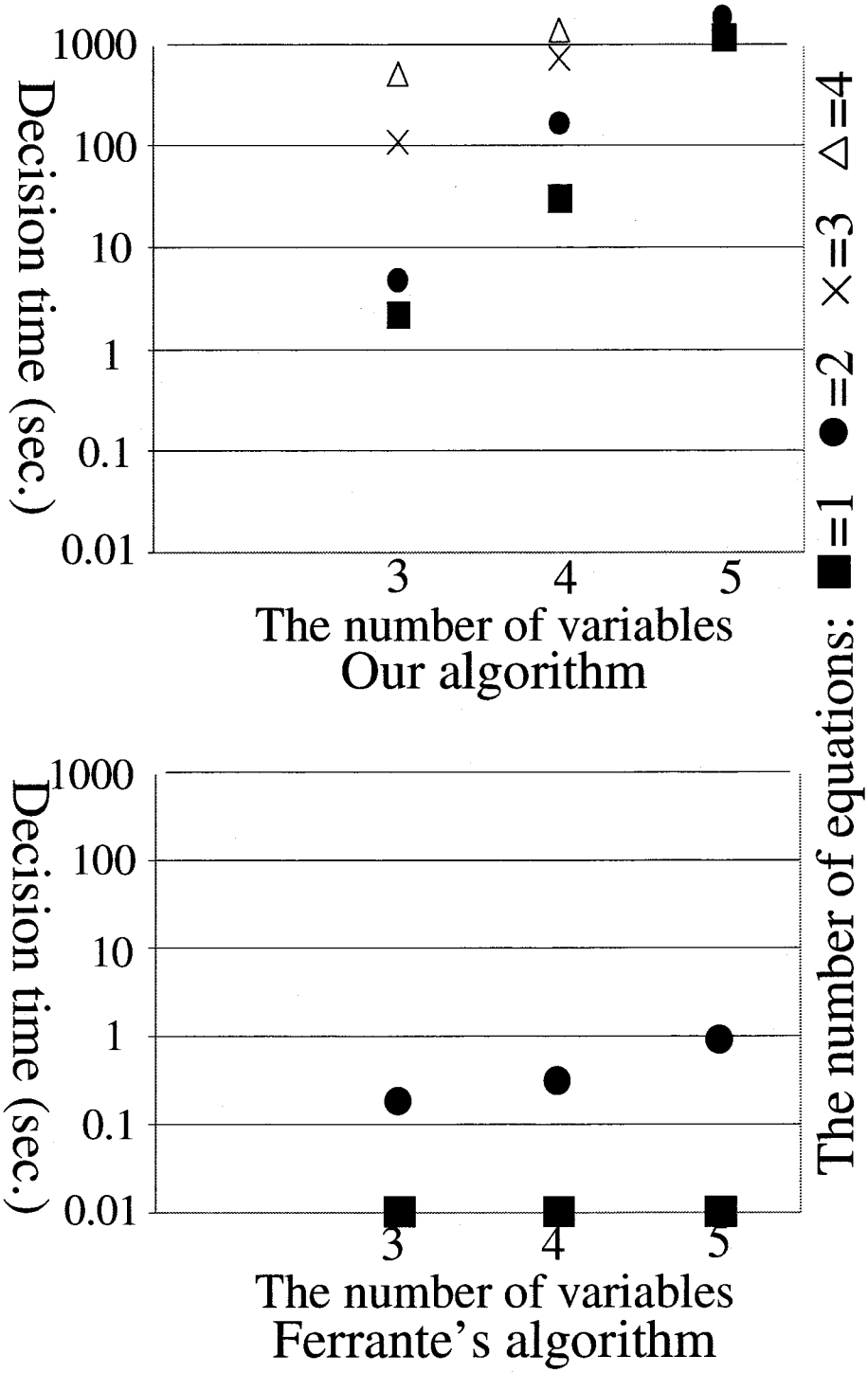

図 7 限定子交替数が 2 の場合の判定時間

2.7 結言

本研究では時間計算量が $r\alpha^{\beta d}n^{\gamma d(b+1)^a}$ (α, β, γ は定数，n は入力の PRP 文に含まれる不等式の個数，d は変数の個数，a は限定子交替数，b は同じ限定子が続く最大の個数，r は入力の式の各係数と定数の分母，分子のビット数) の計算幾何学の手法を利用した PRP 文真偽判定アルゴリズムを提案した．

評価実験から，提案したアルゴリズムの実際の判定時間の変化は計算で求めた結果とだいたい一致すること，外側の限定子が交替する間隔が短い場合には提案するアルゴリズムの方が Ferrante,Rackoff のアルゴリズムよりも短い時間で判定できることが分かった．

3　凹多面体併合を用いたアルゴリズムの高速化

3.1　序言

本研究では，著者が提案したこのアルゴリズムに対して，さらに凹多面体併合等を用いて高速化する手法について述べ，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[25] のバス上で非同期バスマスタ転送を行なう時間オートマトンの適合性試験系列生成などに適用し，評価した結果について述べる．試験系列生成は従来の RP 文真偽判定ルーチンでは，その例に対し実用上不可能であった．

提案する高速化法として，1. 同じ真偽値が割り当てられた多面体を併合し，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少限にすること，2. 投影の領域を計算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．

3.2 凹多面体併合を用いた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，定義通りに，投影を取る前のアレンジメントを構成する $(d-2)$-フラット (アレンジメントの次元を d とする．$(d-2)$-フラットは3次元空間においては線分) を延長してできるフラットの投影全ての集合を求め，それらで空間を細分化することで投影のアレンジメントを作る．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，投影のアレンジメントに含まれるフェイスの数が莫大になり，計算時間がかかる．したがって，各フェイスの投影を直接求めることで，高速化する．また，投影後に，投影のアレンジメントと等価であり，かつ最少のフェイスから構成されるアレンジメントを構成することにより，次回の投影の計算を簡単にする．

この高速化は2のアルゴリズムより平均的に少ない時間計算量で判定することを目指したものであるが，高速化後のアルゴリズムの最悪時間計算量は2のアルゴリズムと同じ $r\alpha^{\beta d}n^{\gamma d(b+1)^a}$ である[7]．時間計算量はフェイスの数の定数 γ 乗に比例する．2のアルゴリズムにおいて時間計算量がフェイスの数の定数乗に比例する箇所は2.3.3において投影前のアレンジメントのフェイス f と投影後のアレンジメントのフェイス f' の全ての組合せに対し f' が f の投影に含まれるか調べる処理であり，およそアレンジメントのフェイスの数の二乗に比例する時間計算量になる．高速化後のアルゴリズムでは，これに相当する処理は後述の3.2.2で述べる高速化によりフェイスの数 n に対して $O(n\log n)$ 程度にまで時間計算量が改善されており，これによって γ の値は減少する．

3.2.1 フェイ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イスは，真と偽の領域の境界が表現できるだけのフェイスである．投影の前にアレンジメントに含まれるフェイスの数を減らせば，これによって短縮される判定時間は減らしたフェイ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8(a)) の各フェイスの投影全ての集合を求め (**投影**)(図8(b))，次に，(ii) 全ての重なったフェイスの組合せを，一方のフェイスのみに含まれるフェイス，他方のみに含まれるフェイス，両方に含まれるフェイスに置き換える処理を行ない

[7] 3.5で述べる通り，高速化を行なったアルゴリズムは，評価実験において判定時間が定数 δ に対し $n^{\delta d}$ に比例する振舞いを見せている．最悪計算量オーダが2のアルゴリズムより下がっている可能性もあるが，解析的には明らかになっていない．

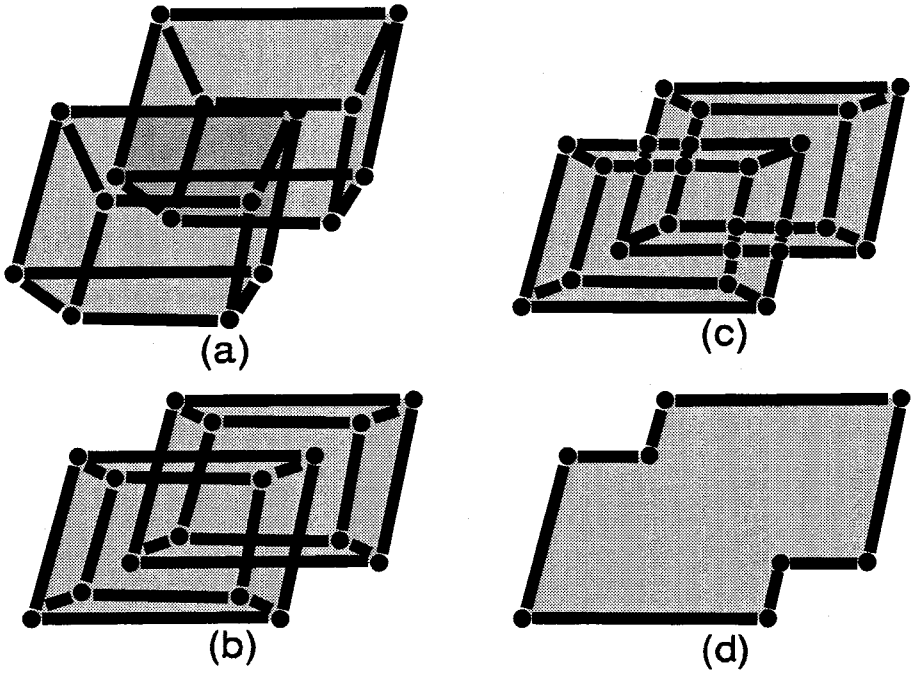

図 8 フェイス同士の重なりがなくなるようにする過程

(**凹多面体分割**)(図 8(c)), その後, (iii) 隣り合う同じ真偽値が割り当てられたフェイスを併合し, 最少数のフェイ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 8(d)). 併合した結果, 凹多面体のフェイスができるので, これもアルゴリズム中で扱えるようにする.

凹多面体を扱うために, フェイス等の概念を拡張する.

定義 12 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 の全ての点 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. □

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-凸フェイス, 1-凸フェイス, 2-凸フェイス, 3-凸フェイスという. 同様に, n-凸フェイスを定

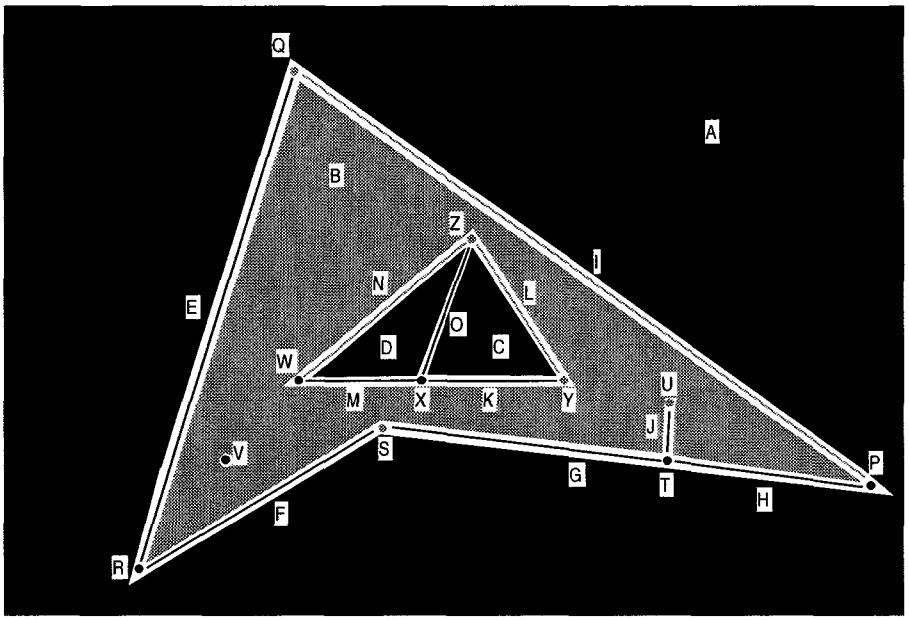

図 9 拡張されたアレンジメントの例

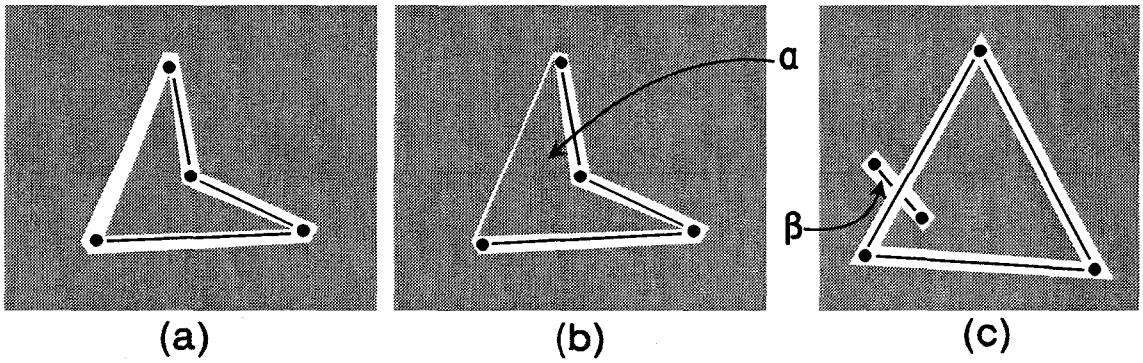

図 10 拡張されたアレンジメントではないフェイスの集合の例

義する[8]．同一の次元の複数の凸フェイスの領域の和集合が拡張されたフェイスである．但し，拡張されたフェイスは連続[9]であって，同一の次元の一つのフラットに含まれる．拡張されたフェイスは全て開領域である (0 次元の点は除外)．

定義 13 拡張された k-フェイス

拡張された k-フェイスは以下の 2 つの規則のみを有限回適用してできるものである．

1.k-凸フェイスは拡張された k-フェイスである．

---

[8] $P = \{\vec{p_0}, \cdots, \vec{p_k}\}$ 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-凸フェイスという．また，点を 0-凸フェイスという．

[9] ある領域のどのような 2 点を取っても，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．

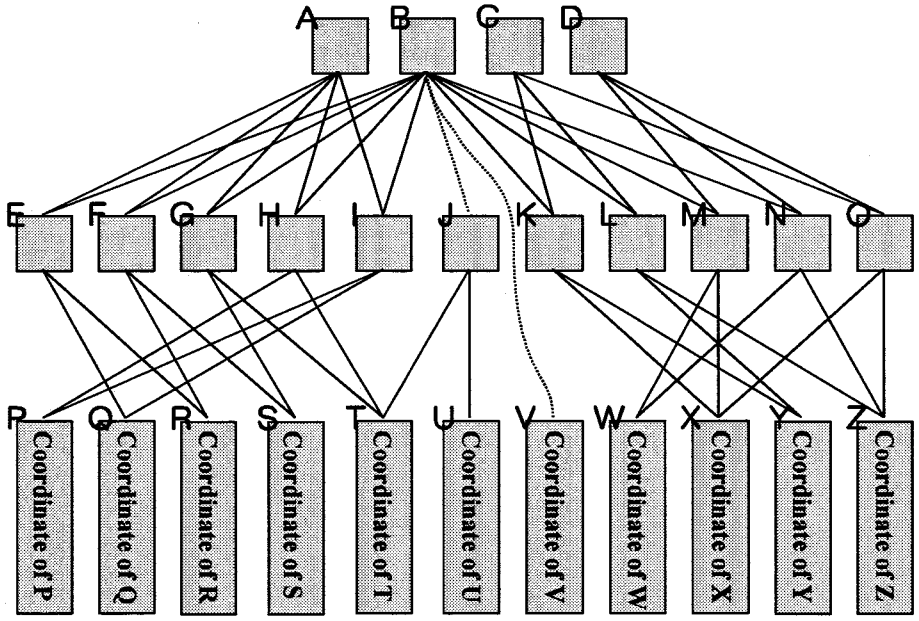

図 11 図 9 のデータ

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

以降，単に k-フェイスと書いた場合は拡張された k-フェイスを指す．

定義 14 サブフェイス, スーパーフェイス

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

定義 15 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イスの集合 S が以下の条件を満たす時，これらのフェイス全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' の要素のフェイス全ての和集合が r に一致すること． □

定義 16 内部サブフェイス

あるフェイス f とフェイス 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時，s は f の内部サブフェイスと呼ぶ．s が f の内部サブフェイスでなく，f のサブフェイスであるとき，s は f の外部サブフェイスと呼ぶ． □

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 9 は 2 次元の，拡張されたアレンジメントの例である．図 9 のフェイス B の境界は

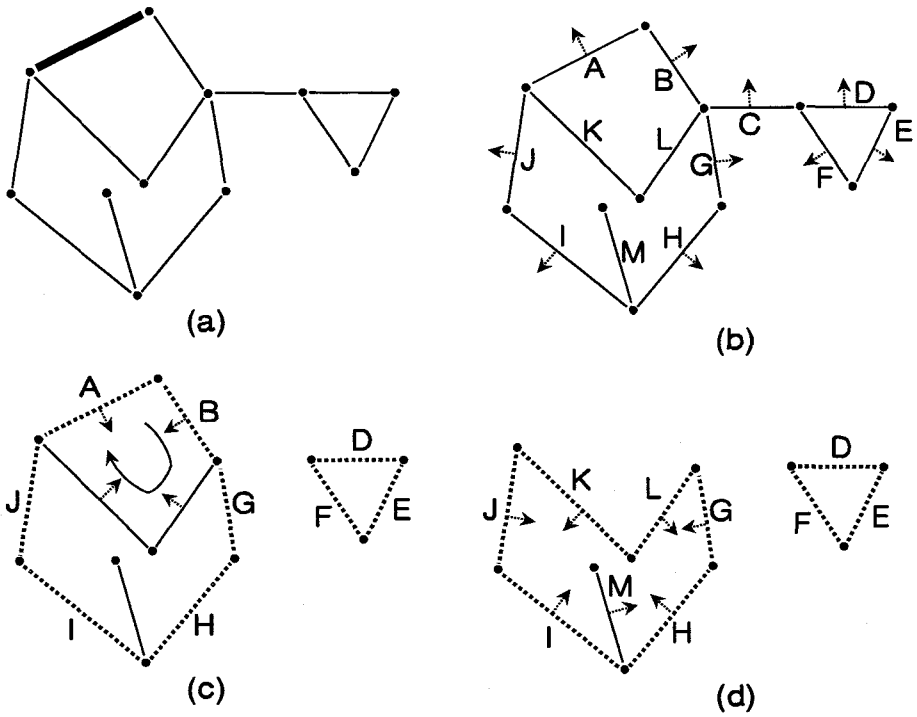

図 12 2次元アレンジメントの例

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 のサブフェイスであるので，M,X,K をまとめて一つの 1-フェイスとすると，O の境界 Z,X を表すために必要な 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．

図 10 は 2 次元の，拡張されたアレンジメントの条件を満たさない，領域の集合の例である．図 10(a) は，集合に含まれるフェイス全ての和集合が空間全体にならない．図 10(b) は，フェイスではない領域 α が集合に含まれる．図 10(c) は，フェイス β に対し，要素のフェイス全ての和集合が境界に一致するような，部分集合が存在しない．

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イスに相当し，0-フェイ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．フェイス f がフェイス g のスーパーフェイスである時に限り 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 9 の黒いフェイスには真が，灰色のフェイスには偽が割り当てられている．

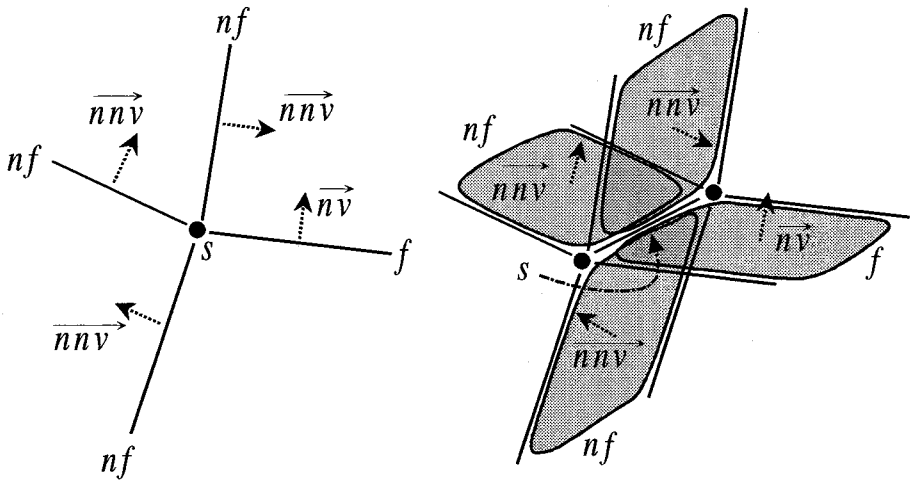

図 13 次に探索するフェイスの決定過程の例

以下では，フェイスの分割・併合処理の概要について述べる．

まず，アレンジメントのデータ構造について述べる．アレンジメントはグラフで表される．図 9 のアレンジメントは図 11 のグラフで表される．グラフの各ノードは一つのフェイスに相当し，0-フェイ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．フェイス g がフェイス f の外部サブフェイスである時に限り f に相当するノードと g に相当するノード間に外部サブフェイスを表すエッジ (図 11 の実線で表されたエッジ) が存在する．また，フェイス g がフェイス f の内部スーパーフェイスである時に限り f に相当するノードと g に相当するノード間に内部サブフェイスを表すエッジ (図 11 の破線で表されたエッジ) が存在する．また，アレンジメントの各フェイスには真偽値が割り当てられている．図 9 の黒いフェイスには真が，灰色のフェイスには偽が割り当てられている．

最後に，アルゴリズムの概要について述べる．分割処理は k 個のフェイス $f_1, .., f_k$ からなるフェイスの集合 S を入力とし，点 p が f_j に含まれていれば $v_j(p) = 1$, そうでなければ $v_j(p) = 0$ であるとき，$(v_1(p), ..., v_k(p))$ の同値類であるアレンジメント A を出力する．併合処理は入力されたアレンジメントと等価で，最少のフェイスから構成されるアレンジメントを出力する．アルゴリズムは次のようになる．

1. 入力されたフェイスの集合から最初に新たな 0 次元のフェイス (頂点) を作る処理を行なう．

1-1. 全てのフェイスの組合せに対してその交わりが頂点になるか調べ，そうであれば

そこに頂点を作る.

1-2. 1-1 の処理で，1-フェイス (辺) との交わりに新しい頂点ができた場合，その辺を 2 つに分割する.

2. 1. と同様に，全てのフェイスの組合せなどに対して，その交わりにフェイスができるかどうか調べ，できるのであれば作る処理を行なう.

3. 2-フェイスから順に高い次元のフェイスを作っていく．作っているフェイスの次元を i とする．アレンジメントに含まれる i-フェイスを含む各 i-フラット $cmnfl$ について，$cmnfl$ に含まれる i-フェイスを作っていく.

3-1. $cmnfl$ 内の，最も外側の $(i-1)$-フェイス f を一つ勝手に選んで求める[10](図 12(a) の太線で示されたフェイス).

3-1-1. A の，$cmnfl$ に含まれるの $(i-1)$-フェイスの集合を F とする．F の，最も外側の 0-フェイス f_0 を求める．これは，適当に選んだ座標軸 ax に対し，最も小さな座標を持つ点のうちいずれかを勝手に選ぶ.

3-1-2. 最も外側の 1-フェイスを求める．これは，f_0 の各スーパーフェイスと座標軸 ax のなす角度を求め，最大の角度を持つもののうちの一つを勝手に選ぶ．同様に最も外側の一つ大きい次元のフェイスを次々に求めていく.

3-2. $cmnfl$ 上で，アレンジメント中の最も外側にある，$(i-1)$-フェイスを全て求める.

3-2-1. f の法線ベクトルのうち，座標軸 ax の正の側を向くベクトルとの内積が小さい方のベクトルを求め，これを $\vec{nv}$ とする.

3-2-2. f と $\vec{nv}$ から，f のサブフェイスのスーパーフェイスでかつ最も外側にあるフェイスとその外側を向く法線を求める処理をくりかえし，全ての最も外側のフェイスを求める (図 12(b)，求めた法線ベクトルが矢印で示されている)．これは，f と $\vec{nv}$ とサブフェイスを共有する f と同じ次元のフェイスが図 13(a) のようであったとすると，$\vec{nv}$ は図の左周りの方向を向いているので，f からみて左周りに最も近い nf を次に探索する．nf の法線として，右

[10]あるフェイス g が最も外側にあるとは g に含まれる点と，原点から十分はなれた点を結ぶ曲線 l が $cmnfl$ 内に存在し，l と共有点を持つ，g と同じかそれ以下の次元の g 以外のフェイスは存在しないことである.

周りの nnv を用いる．さらに大きな次元のアレンジメントに関してもどうように処理する (図 13(b))．求めた最も外側のフェイスにマークをつける．

3-3. F から，一つの i-フェイスの内部サブフェイスとなるようなフェイス (図 12(b) のフェイス C) を取り除き，求めた法線ベクトルを全て反転する (図 12(c))．

3-4. F の最も外側の $(i-1)$-フェイス f を一つ勝手に選んで求める．

3-5. F に含まれる，f のスーパーフェイスのサブフェイスとなるフェイス全ての集合 S を求める．f と $\vec{nv}$ から，f のサブフェイスのスーパーフェイスでかつ f のスーパーフェイスのサブフェイスとなるフェイスと，法線を求める処理をくりかえし，求める (図 12(c))．

3-6. S から新しい i-フェイスを作り，適当な真偽値を割り当てる．

3-7. S に含まれる，外側にマークされたフェイスを F から取り除き，S に含まれるそれ以外のフェイスの法線ベクトルを反転する (図 12(d))．

3-8. F が空集合でなければ，3-4 へ．

3-9. 各 $cmnfl$ に対して 3-1 から 3-8 の処理を行なう．

3-10. 各 i に対して 3-1 から 3-9 の処理を行なう．

4. フェイスの併合を行なう．

4-1. アレンジメント中に以下のようなフェイス f が存在すれば，閉包が f と領域を共有するようなフェイス全てと f を併合する：f のスーパーフェイスが存在しなくて，閉包が f と領域を共有するフェイス g が存在し，f と g に同じ真偽値が割り当てられている．

4-2. アレンジメント中に以下のようなフェイス f が存在すれば，f と f のスーパーフェイス全てを併合する：f のスーパーフェイスが 2 つ以上存在して，全て同じ $(k+1)$-フラット fl に含まれていて，全て f と同じ真偽値が割り当てられている．

4-3. 4-1 または 4-2 の条件に当てはまるフェイスが存在しなくなるまで 4-1 から 4-2 の処理を行なう．

3.3 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

ト簡単化を行なうが，それによって最少限のフェイ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 3.4 で述べる．

3.2.2 フェイスの交差判定回数の削減

3.2.1 で述べた高速化法を適用したアルゴリズムにおいて，フェイス同士が重なっているかの判定は 2 つのフェイスの組合せ全てに対して行なう．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 1 つの超平面 H で区切り，空間 U と他方の空間 D の 2 つに分割し (H は D に含める)，それぞれのフェイスに対し「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イスと D のみと領域を共有するフェイスは交差しないので，これらのフェイス同士の交差判定は省略できる．空間をさらに細かく分割することで，最良の場合フェイス同士の交差判定の回数はフェイスの数 n に対して $O(n \log n)$ 程度にまで少なくできる [6].

また，フェイスの交差判定の前処理として，それぞれのフェイ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 [27]．これらの直方体が交わってなければ，2 つのフェイスは交わっていない．

3.2.3 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あてはまる部分をその右側の部分に書き換える．これを，表のいずれにもあてはまらなくなるまで繰り返す．これ以外にも 2 つの簡単化規則がある．これによって，さらにアレンジメントの分割ルーチン (3.2.1 で述べたステップ (ii) を行なう) に入力されるアレンジメントの次元数を減らすことができる．

3.2.4 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

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

図 1 入力式の簡単化規則

3.3 アルゴリズムの詳細

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, SIMPLIFYはそれぞれ3.2.1で述べた**投影, 凹多面体分割, アレンジメント簡単化**を行なう．判定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，サブルーチンDECIDESUBが必要になる．

Algorithm MAIN
◇ 入力：RP文 F
◇ 出力：F の真偽値
1 F' := 3.2.3で述べた方法により F を簡単化した式;
2 A := DECIDESUB(F');
▷ A は *1* つの点だけで構成される
3 **return** A の原点の真偽値;
4 **end**

Algorithm DECIDESUB
◇ 入力 自由変数を持つ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イス全ての和集合が,E が真になる領域と一致する
1 E の先頭にある限定子全ての並びを $Q_1, ..., Q_k$,E から $Q_1, ..., Q_k$ 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**
10
▷ i は残っている限定子の数
11 $i := k$;
12 **while** $i \geq 1$ **do**
13 s := 最も内側の連続した Q_i の個数;
14 A := PROJECT(A,s);
15 DIVIDE(A,Q_i);
16 SIMPLIFY(A);
17 $i := i - s$;
18 **endwhile**
19 **return** A;
20 **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イス全ての和集合が一致する

Algorithm PROJECT

◇ 入力：アレンジメント A, 整数 s

◇ 出力：A に含まれるフェイスの $(v_1, ..., v_{d-s})$ 空間への投影 (但し割り当てられた真偽値はもとのフェイスに割り当てられた真偽値と同じ) 全ての集合

Algorithm SIMPLIFY

◇ 入力：アレンジメント A

◇ 出力：A と等価で, 最少のフェイスから構成されるアレンジメント A'

1

2 **while** 以下の 2 つの if 文の条件のいずれかを満たす k 次元のフェイス $f \in A$ がある **do**

if f のスーパーフェイスが存在しなくて, 閉包が f と領域を共有するフェイス 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イス全てと f を併合する **endif**

if f のスーパーフェイスが 2 つ以上存在して, 全て同じ $(k+1)$-フラット fl に含まれていて, 全て f と同じ真偽値が割り当てられている **then**

3 f と f のスーパーフェイス全てを併合する

4 **endif**

5 **endwhile**

6 **end**

Algorithm DIVIDE

◇ 入力 k 個のフェイス $f_1, .., f_k$ からなるフェイスの集合 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, $(v_1(p), ..., v_k(p))$ の同値類分割によって定義されるアレンジメント A. A に含まれる各フェイス 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 の要素であるフェイスに割り当てられた真偽値全ての論理和 (論理積).

1 A から重複するフェイスを取り除く;

2 A 内のフェイスに全て「古いフェイス」のマークをつける;

3 PHASE_A(A,Q);

4 PHASE_B(A,Q);

5 PHASE_C(A,Q);

6 A 内の「古いフェイス」のマークのついたフェイスを A から取り除く;

Algorithm PHASE_A

◇ 入力 拡張されたアレンジメント A, 限定子 Q

◇ 出力 A の 1 次元以下のフェイスを分割した, 拡張されたアレンジメント

1 分割によってできる 0-フェイス v を全て作る. また, Q 等によって適切な真偽値を v に割り当てる. また, その 0-フェイスが 1-フェイス上にある場合はその 1-フェイスを分割する

2 **for** A 内のフェイスの組合わせ (f_1, f_2) 全てに対して **do**

▷ 実際に実装したルーチンではここに *3.2.2* で述べた工夫がなされている

3 **if** flatIncludingFace(f_1) と flatIncludingFace(f_2) の交わりが直線にならない **then continue** ;

4 f_1, f_2 のサブフェイス, サブフェイスのサブフェイス, ... の頂点全ての集合を V_1, V_2 とする.

5 **if** $V_1 \cap V_2 = \emptyset$ **then continue** ;

6 $V := V_1 \cap V_2$;
7 V から勝手に 2 つの異なる要素を選び，それらの座標を $\vec{e_1}, \vec{e_2}$ とする．
8 V のある要素の座標を $\vec{v}$ としたとき，$(\vec{e_1} - \vec{e_2}) \cdot (\vec{v} - \vec{e_2})$ の値の昇順に V の要素全てを整列し，これをリスト L とする．
9 **for** each $e'_1 \in L$, 但し最後の要素を除く **do**
10 $e'_2 :=$ e'_1 の次の要素
11 **if** e'_1 と e'_2 の中点の座標が f_1 と f_2 の両方に含まれていない **then continue** ;
endif
12 **if** e'_1 と e'_2 をサブフェイスとする 1-フェイスがない **then**
13 サブフェイスを e'_1 と e'_2 とするフェイスを作り，そのポインタを ls に代入する．
14 Q と f_1, f_2 に割り当てられた真偽値から ls に適切な真偽値を割り当てる．
15 **else**
16 e'_1 と e'_2 をサブフェイスとする 1-フェイスへのポインタを ls に代入する．
17 **endif**
18 $f'_1 := f_1$;
19 **while** true **do**
20 **if** f'_1 のサブフェイス f_s のなかで，
flatIncludingFace(f) と f_2 の交わりが
flatIncludingFace(f_1) と
flatIncludingFace(f_2) の交わりと等しく，かつ f_s が e'_1 と e'_2 の中点を含む，という条件を満たす f_s がある **then**
21 $f'_1 := f_s$;
22 **else**
23 **break** ;
endif
24 **endwhile**
25 **while** true **do**
26 **if** f'_2 のサブフェイス f_s のなかで，
flatIncludingFace(f) と f_1 の交わりが
flatIncludingFace(f_1) と
flatIncludingFace(f_2) の交わりと等しく，かつ f_s が e'_1 と e'_2 の中点を含む，という条件を満たす f_s がある **then**
27 $f'_2 := f_s$;
28 **else**
29 **break** ;
endif
30 **endwhile**
31 ls のスーパーフェイスに f'_1, f'_2 を加える．
32 **next**
33 **next**
34 **end**

Algorithm DUF
◇ 入力 フェイスの集合 A, A 内のフェイスへのポインタ f, フラット $cmnfl$, ベクトル $\vec{nv}$, フェイスの集合 F_d,F_a,F_i へのポインタ
◇ 出力 穴が見つかったかを示すブール値
▷ F_d は新しくできるフェイスの外部サブフェイスに確定したフェイス
▷ F_i は新しくできるフェイスの内部サブフェイスに確定したフェイス
▷ F_a は未確定のフェイス
1 **for** each f のサブフェイス s **do**
2 $nof :=$ s のスーパーフェイスの数;

```
3    配列 nfaces に s のスーパーフェイスを 3.2.1 のアルゴリズムの概要 3-2-2 で述べられ
     たようにソートした結果を代入;
4    for ni := 1 to nof do
5        nf := nfaces[ni];
6        cret := false;
7        if nof == 1 then
8            return true;
9        endif
10       アルゴリズムの概要 3-2-2 で述べた方法により，f と n⃗v，s から nf と nn⃗v を求
         め，nf の nr⃗m にセットする;
11       if s が nf の内部サブフェイスである then
             ▷ T 字形の下から上の場合
12           hole := 1;
13       endif
14       if nf ∈ Fd ∨ nf ∈ Fa then
             ▷ nf ∈ Fd であれば nf は必ず外側
15           if nf の nr⃗m ≠ nn⃗v then REWRITE(nf); endif
16           break ;
17       endif
18       if nf ∈ Fi then
             ▷ nf は必ず内側
19           if s が f の内部サブフェイスでない then cret := true; endif
20           continue ;
21       endif
22       if nf に外側のマークが付いている then
23           DUF(A,nf,cmnfl,nn⃗v,Fd, Fa, Fi);
24           break ;
25       endif
26       if nof = 2 then
27           DUF(A,nf,cmnfl,nn⃗v,Fd, Fa, Fi);
28           break ;
29       endif
30       if s が f の内部サブフェイス then
31           F'a := Fa;
32           DUF(A,nf,cmnfl,nn⃗v,Fd, F'a, Fi);
33           Fi := Fi ∪ (F'a − Fa);
34           continue ;
35       endif
36       F'a := Fa;
37       dret := DUF(A,nf,cmnfl,nn⃗v,Fd, F'a, Fi);
38       selfloop := アルゴリズムの概要 3-2-2 で述べられている意味で nf の次のフェイ
         スが 37 行の DUF で探索されたか？
39       if selfloop then ni にアルゴリズムの概要 3-2-2 で述べられている意味で nf に
         続く，37 行の DUF で探索されたフェイスの次のフェイスを代入 endif
40       if dret then
41           Fi := Fi ∪ (F'a − Fa);
42           cret := true;
43       else
44           cret := selfloop;
45           if f が外側にマークされている then
```

```
46              F_d := F_d ∪ (F'_a − F_a);
47          else
48              if selfloop then
49                  F_d := F_d ∪ (F'_a − F_a);
50              endif
51          endif
52          if selfloop でない then break ; endif
53      endif
54  next
55  ret := ret ∨ cret;
56 next
57 return ret;
```

Algorithm PHASE_BC_SUB

◇ 入力 アレンジメント A, 整数 i, $(i-1)$-フェイスの集合 F, i-フラット $cmnfl$, 限定子 Q

◇ 出力

1 F の部分空間 $cmnfl$ 上でのもっとも外側のフェイス (of) と法線ベクトル $\vec{nv}$ をアルゴリズムの概要 3-1,3-2-1 で述べた方法で求める;
2 $F_d := \emptyset$;
3 $F_i := \emptyset$;
4 $F_a := \emptyset$;
5 $dret :=$ DUF(A,of,$cmnfl$,$\vec{nv}$,F_d, F_a, F_i); **if** $dret = 0$ **then** $F_d := F_d \cup F_a$; **else** $F_i := F_i \cup F_a$; **endif**
6 $F := F$ - F_i;
7 F_d の要素全てに外側のフェイスのマークをつけ, 各フェイスの $n\vec{r}m$ を反転する;
8 **while** $F \neq \emptyset$ **do**
9 F の部分空間 $cmnfl$ 上でのもっとも外側のフェイス (of) と法線ベクトル $\vec{nv}$ をアルゴリズムの概要 3-1,3-2-1 で述べた方法で求める;
10 $F_d := \emptyset$;
11 $F_i := \emptyset$;
12 $F_a := \emptyset$;
13 $dret :=$ DUF(A,of,$cmnfl$,$\vec{nv}$,F_d, F_a, F_i);
14 $F := F - F_i$;
15 F_d を外部サブフェイス, F_i を内部サブフェイスとする i-フェイスを作り, Q などから適切な真偽値を割り当てる;
16 F_d のうち, 外側にマークされているフェイスを F から取り除く;
17 F_d のうち, 外側にマークされていないフェイスの $n\vec{r}m$ を反転し, 外側のフェイスにマークする;
18 **endwhile** ;
19 **next**
20 **for** each 新しくできた i-フェイス f **do**
21 **for** each f に含まれる, f 以外の新しくできた i-フェイス g **do**
22 f の外部サブフェイスに g の外部サブフェイスを追加する;
23 f の内部サブフェイスから g の内部サブフェイスを削除する;
24 **next**
25 **next**

Algorithm PHASE_B

◇ 入力 アレンジメント A, 限定子 Q

◇ 出力

```
1  for i := 2 to A の次元 d do
2     for A 内のフェイスの組合わせ (f1, f2) 全てに対して do
3        if f1 と f2 の領域の積集合が i-フェイス f になり，f が A に含まれない then
4           cmnfl := f1 と f2 の積集合を含む i-フラット;
5           F := A 中の，cmnfl 上にある (i − 1)-フェイス全ての集合;
6           PHASE_BC_SUB(A, i, F, cmnfl, Q);
7        endif
8     next
9  next
```

Algorithm PHASE_C

◇ 入力 1 次元以下のフェイスの重なりのない，アレンジメント A，限定子 Q

◇ 出力 フェイス同士の重なりのない，拡張されたアレンジメント

```
1  for i := 2 to A の次元 d do
2     S := A のいずれかの i-フェイスを含む i-フラット全ての集合;
3     for each cmnfl ∈ S do
4        F := A 中の，cmnfl 上にある (i − 1)-フェイス全ての集合;
5        PHASE_BC_SUB(A, i, F, cmnfl, Q);
6     next
7  next
```

3.4 手続きSIMPLIFYによって最少の等価なアレンジメントが求まることの証明

以下では，SIMPLIFYによって最少のフェイ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 以下では，(1) アレンジメントが最少のフェイスから構成されない場合に次に定義する性質 (P) が成り立つことをを示し，次いで (2)SIMPLIFYによって，入力のアレンジメントと等価な，最少のフェイ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-フェイス f が存在する．

1. f のスーパーフェイスが存在しなくて，f と同じ真偽値が割り当てられていて，かつ境界が f を含むフェイスが存在する

2. f のスーパーフェイスが2つ以上存在し，全て同じ $(k+1)$-フラット fl に含まれていて，全て f と同じ真偽値が割り当てられている

以下では，まずアレンジメントが最少のフェイ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．

アレンジメントの次元を d とする．

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[11]．T'_d をアレンジメントの真が割り当てられたフェイス全ての和集合とする．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 $ft_{k,0}, .., ft_{k,m}$ 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[12]．$T'_{k-1} = ot_{k,0} \cup ... \cup ot_{k,m}$ 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，$ot_{k,j}$ に対し有限の $(k-1)$-フラットの集合が存在し，これら全ての和集合は $ot_{k,j}$ を含む．従って，T'_{k-1} に対して $(k-1)$-フラットの集合が存在し，これら全ての和集合に T'_{k-1} が含まれる．この

[11] アレンジメントの表す真の領域 T'_d から，順に $d, d-1, ..., 1$ フェイスで表せる領域を除いていく．除いた結果が $T'_{d-1}, ..., T'_0$，k-フェイスで表せる領域が (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9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.

[12] $ot_{k,j}$ が簡単化後のアレンジメントのフェイスの集合で表せることが，定義15の条件3を満たす上で必要である点に注意．

ようなフラットの集合 (但し，集合の要素が $(k-1)$-フェイ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) フラットをかわりに要素とする集合) の中で要素数が最少のものを FT_{k-1} とする．

F'_{k-1}, FF_{k-1} も真を偽とよみかえ，同様に定義する．

$T_k = T'_k - T'_{k-1}(k > 0), T_0 = T'_0$ とする[13]．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 $k+1$ 以上の次元のフェイスは存在しない．T_k, F_k を一意に最少の連続な領域に分割することができて[14]，これらは k-フェイスである．

$T_d, ..., T_0, F_d, ..., F_0$ をそれぞれ最少の連続な領域に分割したもの全ての集合[15]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 中のフェイス数は最少である．

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イスが含むような，A と等価なアレンジメントは存在しない．したがって，A と等価な全てのアレンジメントにおいて全ての T_k, F_k の領域が k 以下の次元のフェイスの集合に一致する．

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

ついで，(2) を示す．アレンジメントが性質 (P) を満たす時，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 ループを一回回る前後でアレンジメントは等価であり，また必ずフェイス数が減少する．よって，アルゴリズムの停止性は保証され，アルゴリズムの性質より SIMPLIFY の実行が終了するのは性質 (P) を満たさなくなった時である．(*) 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少のフェイスから構成される．

---

[13] 図9の例では，$T_2 = \mathsf{A \cup C \cup D \cup O}$, $T_1 = \mathsf{E \cup F \cup G \cup H \cup J \cup K \cup M \cup X}$, $T_0 = \mathsf{P \cup R \cup T \cup V \cup W}$. T_1 に X が含まれ，T が含まれないことに注意．

[14] T_k, F_k は一般に不連続な領域であるが，この一部のひと続きの領域をそれぞれ一つのフェイスとする．

[15] 同じく，$A = \{\mathsf{A, C \cup D \cup O, E, F, G, H, J, K \cup M \cup X, P, R, T, V, W, B, I, N, L, Q, S, U, Y, Z}\}$.

よって，SIMPLIFYによって最少のフェイ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-フェイス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-フェイスhが存在する$(m < k)$．

□

補題1の証明

以降，アレンジメントBと領域rに対してフェイス$e \in B$が存在し，eが領域rを含むことを$\mathrm{inc}(B,r)$と表記する．

以下の1から4に場合分けする．

1. fのスーパーフェイスが存在しない場合: Aの構成法から，あらゆるgに対し，gとの和集合が連続な領域になるようなfより大きい次元のフェイスはすべてfと違う真偽値が割り当てられているので，gとそれらのフェイスの和集合の一部との和集合rに対して$\mathrm{inc}(A',r)$であるようなA'は，Aと等価ではない．よって，アレンジメントA'は存在しない．

2. fのスーパーフェイスが一つだけある場合: Aの構成法から，fのスーパーフェイスにはfと同じ真偽値が割り当てられている．また，あらゆるgに対し，gとの和集合が連続になるような，fと同じ真偽値が割り当てられた，fより大きい次元のフェイスはfのスーパーフェイスのみである．fのスーパーフェイスの任意の一部とgの和集合rは，gがrの境界に位置することから，開領域ではないので，rはフェイスにならない．したがって，rに対して，$\mathrm{inc}(A',r)$を満たす，Aと等価なA'は存在しない．

3. fのスーパーフェイスが二つあって，fと同じ真偽値が割り当てられていて，それらが一つの，fのスーパーフェイスと同じ次元のフラットに含まれる場合: これは，Aの構成法からありえない．

4. 上記以外の場合，すなわちfのスーパーフェイスが三つ以上あるか，fのスーパーフェイスが二つあっていずれかにfと違った真偽値が割り当てられているか，fのスーパーフェイスが二つあって，それらが一つの，fのスーパーフェイスと同じ次元のフラットに含まれない場合:

以降，まず$m = d-1$の場合にA'が存在しないことを証明し，次に帰納的に$m = 0$の場合まで証明する．

$m = d-1$の場合: fのスーパーフェイスが三つ以上あることはない．二つのfのスー

パーフェイスは必ず一つの f のスーパーフェイスと同じ次元のフラットに含まれる．二つのスーパーフェイスに違う真偽値が割り当てられていると仮定する．(2. の場合と同様) あらゆる g に対し，f のスーパーフェイスの一部と g の和集合を r とすると，g は r の境界に位置するので，r は開集合ではない．よって，r に対して，$\mathrm{inc}(A', r)$ を満たす A' は存在しない．

$m = i + 1$ のときに題意が成り立っていると仮定し，$m = i$ のときにも成り立つことを証明する．

$k > m + 1$ の場合: フェイスの定義から，f のスーパーフェイスの一部も h に含まれる．この場合，補題 1 が，スーパーフェイスが f である場合に満たされていることになり，帰納法の仮定から，そのような h は存在しない．

$k = m + 1$ の場合: 4. の条件より，あらゆる g に対し，f のスーパーフェイス全ての和集合の一部と g の和集合 r が $(m+1)$-フラットに含まれれば，g は r の境界に位置するので，r は開集合ではない．よって，2. の場合と同様 r に対して，$\mathrm{inc}(A', r)$ を満たす A' は存在しない．r が $(m+1)$-フラットに含まれず，$\mathrm{inc}(A', r)$ を満たす A' が存在すると仮定すると，フェイスの定義より r は f のスーパーフェイスの一部を含み，$k > m + 1$ の場合と同様，帰納法の仮定と矛盾する．よって，アレンジメント A' は存在しない． □

3.5 評価実験と考察

3.5.1 評価実験

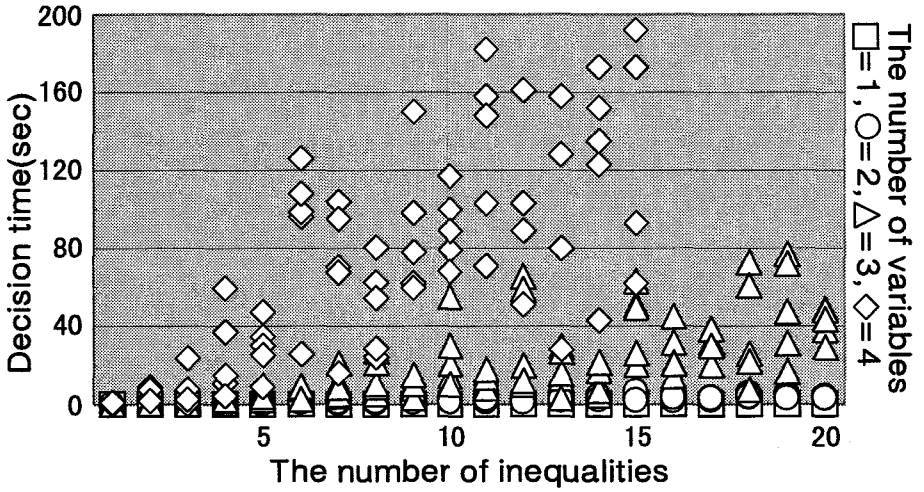

図 14 測定結果 1

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 2 つの評価を行なった．(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な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ない，評価する．各測定は Pentium III 600MHz，メインメモリ 512MB の PC 上で行なった．

(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 つの場合は 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，ランダムな式の形で限定子は冠頭形で毎回入れ替わる並びとした．それぞれ 5 回ずつ判定時間を測定した．この結果を図 14 に示す．同様に使用したメモリを図 16 に示す．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イスを，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．実験で本プログラムが使用したメモリは最大で 5MB 程度であった．

(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 つ以上の場合は 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

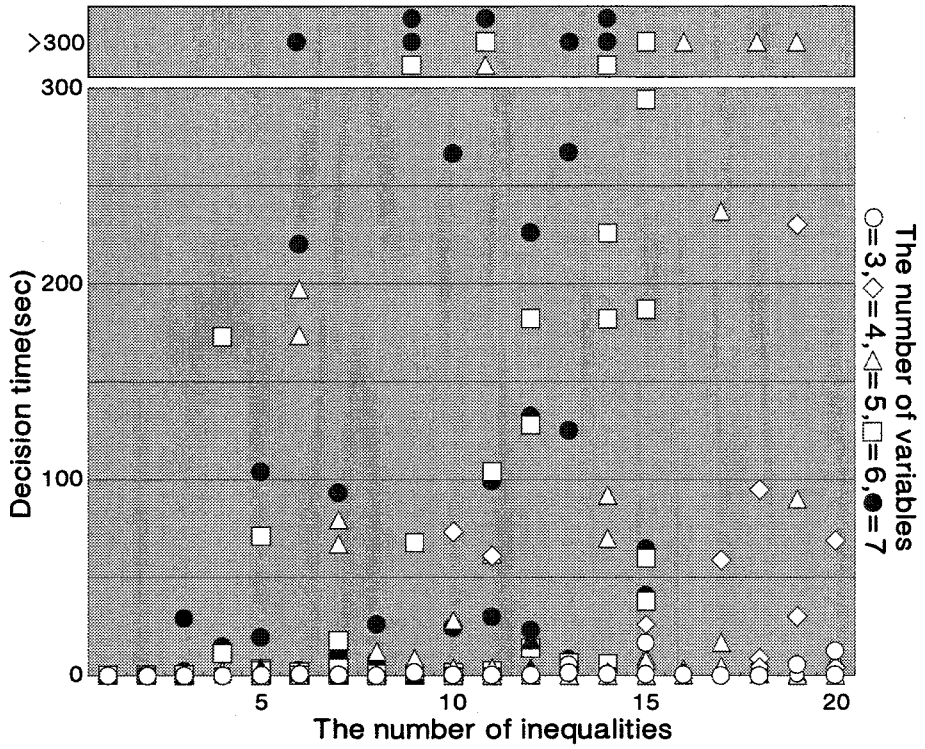

図 15 測定結果 2

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，∃,∀が毎回入れ替わる並びとした．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 15 に示す．実験で本プログラムが使用したメモリは最大で 8MB 程度であった．

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

通信ソフトウェアの信頼性を高める 1 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，テスターから 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，テスターへの出力は 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なわれても，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

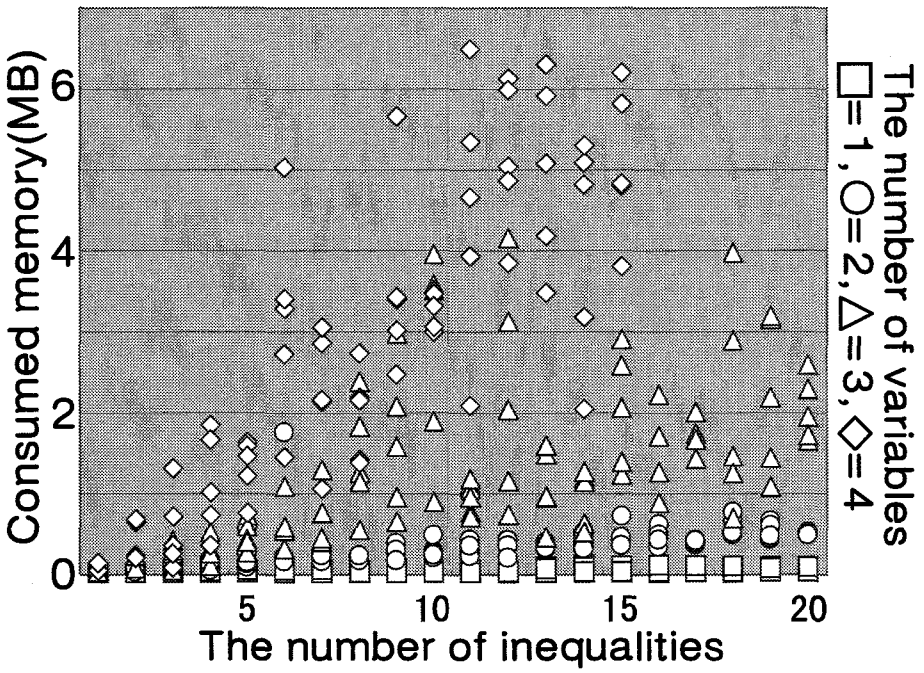

図 16 測定結果1におけるメモリ使用量

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 [10] で述べられている手法を用いる. 但し, 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, 適合性試験系列生成の過程において must トレース可能性判定を行なう系列の長さを短くすることができる. このクラスの時間オートマトンに対する 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, 入力タイミングの変数は存在記号で束縛され, 出力タイミングの変数は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なわれる場合は, 存在記号と全称記号が交互に現れる.

図 17 は MC68030 バス上で非同期バスマスタ転送を行な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, 読み込み指令と読み込むデータのサイズを受けとり, 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.

状態 0 で読み込み指令を受けとると, 状態 1 から 4 でバス要求を行ない, 外部調停を経てバスマスタになる. 既に CPU 以外のデバイスがバスマスタになっていれば, その間待ち続け

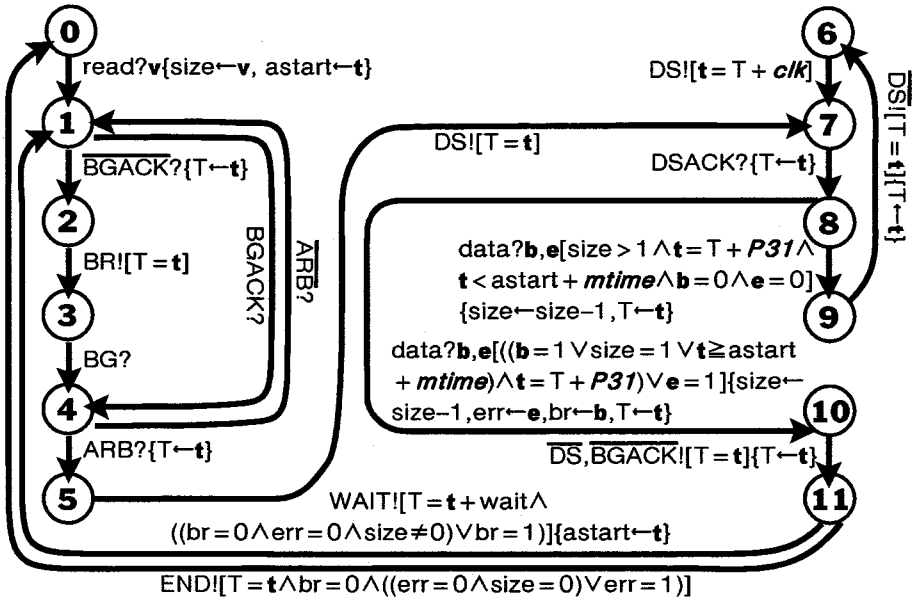

図 17 非同期バスマスタ転送を行なう時間オートマトン

る．状態 5 から 7 においてアドレスバス上にアドレスなどのデータを出力し，スレーブデバイスが DSACK を返すのを待つ．バスの占有を始めてから mtime の時間が経過していなくて，読み込んだデータのサイズが指定された量に達していなくて，バスエラーがなくて，外部調停機構からのバス要求がなければ，状態 8 で DSACK を受け取ってから P31 経過後．状態 9 でデータを読み込み，読み込むデータサイズのカウンタをデクリメントする．これらの条件を満たしていなければ読み込むデータサイズのカウンタをデクリメントして状態 10 に進む．状態 9 からは，アドレスバスからアドレスなどのデータを取り除いて clk の時間が経過後再び次のデータを読む動作を行う (状態 6,7)．状態 11 でバスを解放する．その後，エラーが起こっていなくて，読み込みが終わっていなければ状態 1 に遷移する．そうでなければ，状態 0 に戻る．

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 [10] で述べられている系列集合 $V.A.U_0$[16] に対する must トレース可能性判定の結果を図 2 に挙げる．また，これらの must トレース可能性判定式を付録で挙げる．測定は Pentium III 600MHz，メインメモリ 512MB の PC 上で行なった．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 17 の例に対する適合性試験系列生成において判定する RP 文全体に対する判定時間はおよそ 20 分であった．

[16] 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までの系列．

また，3. で述べた各高速化手法の効果を調べるため，全ての高速化手法を導入した場合と 3.2，3.3 で述べた各手法のみを使わなかった場合について図 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．結果を図 18 に示す．

state	sequence length	number of variables	number of inequalities	time (sec.)
$s0$	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

図 2 *V.A.U.*。の各系列に対する must トレース可能性判定時間

条件	判定時間 (秒)
全ての高速化手法を使用	1.87
フェイ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

図 18 各高速化手法の効果

3.5.2 考察

(1) の実験結果から，全ての係数が 0 ではない場合，変数の数が 4，不等式の数が 15 程度までならおおよそ 200 秒以下で判定できるといえる．また，全ての係数のうち 7 割が 0 であるばあい，変数の数が 7，不等式の数が 15 程度でも数百秒程度で判定できることが分かった．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全て 0 でないため，入力式の構文上の簡単化がほとんど行なわれない．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」回行なわれており，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，凹多面体分割を行なうサブルーチン (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18の結果から，高速化手法がいずれも大きな効果を持つことが分かる[17]．

[17]SIMPLIFYに関しては，アレンジメントのフェイス数がさらに増えれば効果も大きくなる．

3.6 結言

本研究では著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さら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 600MHzにおいて，全ての係数のうち7割が0であるばあい，変数の数が7つ，不等式の数が15個程度でも数百秒で判定できることが分かった．また，比較的簡単な時間制約を持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．

4 あとがき

本研究によって得られた成果をまとめる.

本研究では有理数を扱うことのできる検証系の実現を目指し，時間計算量が $r\alpha^{\beta d}n^{\gamma d(b+1)^a}$ (α, β, γ は定数) の計算幾何学の手法を利用した RP 文真偽判定アルゴリズムを考案した.

また，このアルゴリズムに対して，さらに凹多面体併合等を用いて高速化する手法を考案した.

今後の課題として，完成した真偽判定ルーチンを検証系に組み込んで有理数拡張を行うこと，有理数拡張を行った検証系を用いて，実際の仕様や設計例題に対する検証実験を行い，適用可能な問題の種類, サイズ等を見極めることなどが挙げられる.

謝辞

本研究を行なうにあたり，常日頃より適切なご指導を賜り，また，忍耐強く励ましてくださいました大阪大学大学院基礎工学研究科情報数理系専攻 谷口健一 教授に心から深謝申し上げます．

本研究をまとめるにあたり，有益なご助言を与えてくださいました情報数理系専攻の東野輝夫 教授，柏原敏伸 教授，増澤利光 教授に深謝申し上げます．

著者が大阪大学基礎工学部情報工学科および同大大学院に在籍中に御指導頂いた情報数理系専攻の故 西川清史 教授， 故 藤井護 教授， 都倉信樹 教授，菊野亨 教授，宮原秀夫 教授， 橋本昭洋 教授， 今井正治 教授， 藤原融 教授，井上克郎 教授，萩原兼一 教授， 村田正幸 教授， 首藤勝 教授 (現在，大阪工業大学), 大阪大学産業科学研究所の 北橋忠宏 教授，大阪大学医学部の 田村進一 教授に深謝申し上げます．

本研究の全過程を通じ終始, 適切な御指導, 御助言を賜わりました大阪大学基礎工学研究科情報数理系専攻 岡野浩三 講師に深く感謝いたします．

著者が本研究を始める前の研究においてプレスブルガー文真偽判定手続きを作るにあたって，適切な御指導, 御助言を賜わりました森岡澄夫 氏 (現在，日本 IBM 株式会社) に深く感謝いたします．

本研究をすすめるにあたり，適切なご助言を頂きました滋賀大学経済学部情報管理学科安本慶一 助教授，大阪大学基礎工学研究科情報数理系専攻 山口弘純 助手に深謝いたします．

常日頃より御討論頂いた谷口研究室の方々に心から感謝いたします．

参考文献

[1] 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", 電子情報通信学会論文誌 Vol.J82-DI, **6**, pp.691-700, 1999.

[2] N. Shibata, K. Okano, T. Higashino, and K. Taniguchi : "A decision algorithm for prenex normal form rational Presburger sentences based on combinatorial geometry," Proceedings of 2nd International Conference on Discrete Mathematics and Theoretical Computer Science and the 5th Australasian Theory Symposium (DMTCS'99+CATS'99), pp.344-359, 1999.

[3] 柴田直樹, 岡野浩三, 谷口健一 : "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", 電子情報通信学会論文誌, 採録決定.

[4] 柴田直樹, 岡野浩三, 東野輝夫, 谷口健一 : "Tarski 算術における冠頭標準形の閉論理式の真偽判定アルゴリズムの提案", 電子情報通信学会技術研究報告, Vol.97-COMP, No.356, pp.17-24, 1997.

[5] 柴田直樹, 岡野浩三, 東野輝夫, 谷口健一 : "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とその高速化手法", 電子情報通信学会情報基礎論ワークショップ LA シンポジウム報告, pp.4.1-4.6, 1998.

[6] 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", 電子情報通信学会技術研究報告, Vol.99-COMP, No.432, pp.1-8, 1999.

[7] 柴田直樹, 森岡澄夫, 東野輝夫, 谷口健一 : "プレスブルガー文真偽判定手続きにおける多元連立 1 次合同式の求解処理の高速化", 第 53 回情報処理学会全国大会予稿集 2M−07, 1996.

[8] 森岡澄夫, 柴田直樹, 東野輝夫, 谷口健一 : "プレスブルガー文真偽判定手続きを用いた算術演算回路の正しさの証明", 電子情報通信学会技術研究報告, VLD96-61, pp.49-56, 1996.

[9] 森岡澄夫, 柴田直樹, 東野輝夫, 谷口健一 : "全ての変数が存在記号で束縛された冠頭標準形プレスブルガー文の真偽判定の高速化手法", 情報処理学会論文誌, Vol.38, **12**, pp.2419-2426, 1997.

[10] 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", 情報処理学会論文誌, Vol.40, **1**, pp.85-94, 1999.

[11] A. Nakata, T. Higashino and K. Taniguchi : "Time-action alternating model for timed LOTOS and its symbolic verification of bisimulation equivalence," Proceedings of FORTE/PSTV'96, pp.279-294, 1996.

[12] 森岡澄夫, 東野輝夫, 谷口健一 : "全ての変数が存在記号で束縛された冠頭標準形プレスブルガー文の真偽判定プログラム", 電子情報通信学会技術研究報告, SS95−18, pp.63-70, 1995.

[13] 直井邦彰, 高橋直久 : "プレスブルガー算術を用いた Infeasible Path 検出", 電子情報通信学会論文誌, Vol.J80-DI, **11**, pp.898-906, 1997.

[14] 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, **6**, pp.31-39, 1992.

[15] 嵩忠雄, 谷口健一, 杉山裕二, 関浩之 : "代数的言語 ASL /*-意味定義を中心に-", 電子情報通信学会論文誌, Vol.J69-D, **7**, pp.1066-1074, 1986.

[16] 東野輝夫, 関浩之, 谷口健一 : "代数的仕様から関数型プログラムの導出とその実行", 情報処理, Vol.29, **8**, pp.881-896, 1988.

[17] 岡野浩三, 北道淳司, 東野輝夫, 谷口健一 : " 順序機械型プログラムの階層的設計法と在庫管理プログラムの開発例", 電子情報通信学会論文誌, Vol.J76-DI, **7**, pp.354-363, 1993.

[18] 北嶋暁, 森岡澄夫, 島谷肇, 東野輝夫, 谷口健一 : "代数的手法を用いた CPU KUE-CHIP2 の段階的設計およびその正しさの自動証明", 電子情報通信学会論文誌, Vol.J79-DI, **12**, pp.1017-1029, 1996.

[19] 島谷肇, 北嶋暁, 森岡澄夫, 東野輝夫, 谷口健一 : "代数的手法を用いた in-order 実行パイプライン CPU の自動設計検証", 情報処理学会論文誌, Vol.39, **6**, pp.1999-2008, 1998.

[20] 竹中崇, 北道淳司, 谷口健一 : "あるクラスの Out-of-Order 型パイプライン CPU の設計の正しさの十分条件とその形式的検証", 情報処理学会論文誌, Vol.40, **4**, pp.1587-1596, 1999.

[21] T.R. Shiple, J.H. Kukula and R.K. Ranjan : "A comparison of Presburger engines for EFSM reachability," LNCS **1427**, pp.280-292, 1998.

[22] T. Bultan, R. Gerber and W. Pugh : "Symbolic model checking of infinite state systems using Presburger arithmetic," LNCS **1254**, pp.400-411, 1997.

[23] C.W. Barett, D.L. Dill and J.R. Levitt : "Validity checking for combinations of

theories with equality," LNCS **818**, pp.187-201, 1996.

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

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

[26] H. Edelsbrunner : "Algorithms in combinatorial geometry," Springer-Verlag Berlin Heidelberg, 1987.

[27] S. Suri, P. Hubbard and J. Hughes : "Collision detection in aspect and scale bounded polyhedra," Proceedings of 9th Annual Symposium on Discrete Algorithms, pp.382-392, 1998.

[28] M. Presburger : "Uber die vollstandigkeit eines gewissen systems der arithmetik ganzer zahlen," in welchen die Addition als einzige Operation Hervortritt, in Comptes-Rendus du ler Congress des Mathematiciens des Pays Slavs, 1929.

[29] D.C. Cooper : "Theorem Proving in arithmetic without multiplication," Machine Intelligence, **7**, pp.91-99, 1972.

[30] D.C. Oppen : "A $2^{2^{2^{pn}}}$ upper bound on the complexity of Presburger arithmetic," Journal of Computer System Science, **16**, pp.322-332, 1978.

[31] J. Ferrante and C. Rackoff : "A decision procedure for the first order theory of real addition with order," SIAM Journal of Computation, **4**, pp.69-76, 1975.

[32] B. Scarpellini : "Complexity of subcases of Presburger arithmetic," American mathematical society, Vol 284, **1**, pp.203-281, 1984.

[33] M.J. Fischer and M.O. Rabin : "Super exponential complexity of Presburger arithmetic," SIAM-AMS Proceedings, **VII**(AMS, Providence, RI), 1974.

[34] A.R. Bruss and A. Meyer : "On time-space classes and their relation to the theory of real addition," Thoretical Computer Science, **11** pp.59-69, 1980.

[35] L. Berman : "The complexity of logical theories," Theoretical Computer Science, **11** pp.71-77, 1980.

[36] V. Weispfenning : "The complexity of linear problems in fields," Journal of Symbolic Computation, **5**, pp.3-27, 1988.

[37] C. Hosono and Y. Ikeda : "A formal derivation of the decidability of the theory SA," Theoretical Computer Science, **127** pp.1-23, 1994.

[38] 細野千春, 池田靖雄 : "有理 Presburger 算術の決定性について", コンピュータソフトウェア, Vol.9, **5**, pp.54-61, 1992.

[39] V. Klee and G.L. Minty, : "How good is the simplex algorithm?," O.Shisha ed., Inequlities III, Academic Press,New York, pp.159-179, 1972.

[40] L.G. Khachiyan : "Polynomial algorithms for linear programming," Dokl. Akad. Nauk SSSR, **244**, pp.1093-1096, 1979.

[41] N. Karmarkar : "A new polynomial time algorithm for linear programming," Combinatorica, **4**, pp.373-395, 1984.

[42] J. Renegar : "A polynomial-time algorithm based on Newton's method, for linear programming," Technical Report, MSRI 07118-86, Mathematical Science Research Institute, Berkeley, CA, 1986.

[43] P.M. Vaidya : "An algorithm for linear programming which requires $O(((m+n)n^2 + (m+n)^{1.5}n)L)$ arithmetic operations," Proceedings of 19th Annual ACM Symposium on Theory of Computing, pp.29-38, 1987.

[44] R.E. Shostak : "On the SUP-INF method for proving Presburger formulas," Journal of the Association for Computing Machinery, Vol.24, **4**, pp.529-543, 1977.

[45] W.W. Bledsoe : "A new method for proving certain Presburger formulas," In Advance Papers, 4th International Joint Conference on Artificial Intelligence, Tibilisi, Georgia, U.S.S.R, 1975.

[46] W. Pugh : "The Omega test: a fast and practical integer programming algorithm for dependence analysis," Communications of the ACM, 35, **8**, pp.102-114, 1992.

[47] W. Pugh : "A practical algorithm for exact array dependence analysis," Communications of the ACM, vol.35, **8**, pp.102-114, 1992.

[48] W. Pugh : "Counting solutions to Presburger formulas: how and why," Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation, 1994.

[49] G.B. Dantzig and B.C. Eaves : "Fourier-Motzkin elimination and its dual," Journal

Combinatorial Theory(A), **14**, pp.288-297, 1973.

[50] E. Gradel : "Subclasses of Presburger arithmetic and the polynomial-time hierarchy," Theoretical Computer Science, 56, **3**, pp.289-301, 1988.

[51] H. Comon and Y. Jurski : "Multiple counters automata, safety analysis, and Presburger arithmetic," LNCS **1427**, pp.268-279, 1998.

[52] C.R. Reddy and D.W. Loveland : "Presburger arithmetic with bounded quantifier alternation," Conference Record of the Tenth Annual ACM Symposium on Theory of Computing, pp.320-325, 1978.

[53] T. Amon, G. Borriello and J. Liu : "Making complex timing relationships readable : Presburger formula simplification using don't cares," Proceedings of the International Conference on Design Automation '98, pp.586-590, 1998.

付録

3.5.1 における must トレース可能性判定式を以下に挙げる.

状態 s0 の must トレース可能性判定式

```
forall mtime forall p31 forall wait (
(mtime = 100 and p31 = 1 and wait = 100) imply
(exists size exists breq exists err
(
  (exists t0 (true))
)))
```

状態 s1 の must トレース可能性判定式

```
forall mtime forall p31 forall wait (
(mtime = 100 and p31 = 1 and wait = 100) imply
(exists size exists breq exists err
(
  (exists t0 (true) and
    (exists t1 (t0 <= t1))
  )
)))
```

状態 s2 の must トレース可能性判定式

```
forall mtime forall p31 forall wait (
(mtime = 100 and p31 = 1 and wait = 100) imply
(exists size exists breq exists err
(
  (exists t0 (true) and
    (exists t1 (t0 <= t1 and true) and
      ((exists t2 (t1 <= t2 and t2 = t1)) and
       (forall t2 ((t1 <= t2 and t2 = t1) imply true))
      )
    )
  )
)))
```

状態 s3 の must トレース可能性判定式

```
forall mtime forall p31 forall wait (
(mtime = 100 and p31 = 1 and wait = 100) imply
(exists size exists breq exists err
(
  (exists t0 (true) and
    (exists t1 (t0 <= t1 and true) and
      ((exists t2 (t1 <= t2 and t2 = t1)) and
       (forall t2 ((t1 <= t2 and t2 = t1) imply
         (
           exists t3 (t2 <= t3)
         )
       ))
      )
    )
  )
)))
```

状態 s4 の must トレース可能性判定式

```
forall mtime forall p31 forall wait (
(mtime = 100 and p31 = 1 and wait = 100) imply
(exists size exists breq exists err
(
  (exists t0 (true) and
    (exists t1 (t0 <= t1 and true) and
      ((exists t2 (t1 <= t2 and true))
      )
    )
  )
)))
```

状態 s5 の must トレース可能性判定式

```
forall mtime forall p31 forall wait forall clk (
(mtime = 100 and p31 = 1 and wait = 100 and clk = 2) imply
(exists size exists breq exists err
(
  (exists t0 (true) and
    (exists t1 (t0 <= t1 and true) and
      (exists t2 (t1 <= t2 and true) and
        (
         (exists t3 (t2 <= t3 and t3 = t2 + clk)) and
         (forall t3 ((t2 <= t3 and t3 = t2 + clk) imply true))
        )
      )
    )
  )
)))
```

状態 s6 の must トレース可能性判定式

```
forall mtime forall p31 forall wait forall clk (
(mtime = 100 and p31 = 1 and wait = 100 and clk = 2) imply
(exists size exists breq exists err
(
  (exists t0 (true) and
    (exists t1 (t0 <= t1 and true) and
      (exists t2 (t1 <= t2 and true) and
        (
         (exists t3 (t2 <= t3 and t3 = t2 + clk)) and
         (forall t3 ((t2 <= t3 and t3 = t2 + clk) imply
           (exists t4 (t3 <= t4 and true))
         ))
        )
      )
    )
  )
)))
```

状態 s7 の must トレース可能性判定式

```
forall mtime forall P31 forall wait forall clk (
(mtime = 100 and P31 = 1 and wait = 100 and clk = 2) imply
```

```
(exists size
(
  (exists t0 (true) and
    (exists t1 (t0 <= t1 and true) and
      (exists t2 (t1 <= t2 and true) and
        (
         (exists t3 (t2 <= t3 and t3 = t2 + clk)) and
         (forall t3 ((t2 <= t3 and t3 = t2 + clk) imply
           (exists t4 (t3 <= t4 and
             (
               (exists t5
                 (t4 <= t5 and size > 1 and
                  t5 = t4 + P31 and t5 < t0 + mtime)) and
               (forall t5
                 ((t4 <= t5 and size > 1 and t5 = t4 + P31 and
                   t5 < t0 + mtime) imply true))
             )
           ))
         ))
        )
      )
    )
  )
)))
```

状態 **s8** の **must** トレース可能性判定式

```
forall mtime forall P31 forall wait forall clk (
(mtime = 100 and P31 = 1 and wait = 100 and clk = 2) imply
(exists size
(
  (exists t0 (true) and
    (exists t1 (t0 <= t1 and true) and
      (exists t2 (t1 <= t2 and true) and
        (
         (exists t3 (t2 <= t3 and t3 = t2 + clk)) and
         (forall t3 ((t2 <= t3 and t3 = t2 + clk) imply
           (exists t4 (t3 <= t4 and
             (
               (exists t5
                 (t4 <= t5 and size > 1 and t5 = t4 + P31 and
                  t5 < t0 + mtime)) and
               (forall t5
                  ((t4 <= t5 and size > 1 and t5 = t4 + P31 and
                    t5 < t0 + mtime) imply
                   (exists t6 (t5 <= t6 and t6 = t5))
             ))
           ))
         ))
        )
      )
    )
  )
)))
```

状態 **s9** の **must** トレース可能性判定式

```
forall mtime forall P31 forall wait forall clk (
(mtime = 100 and P31 = 1 and wait = 100 and clk = 2) imply
(exists size
(
  (exists t0 (true) and
    (exists t1 (t0 <= t1 and true) and
      (exists t2 (t1 <= t2 and true) and
        (
         (exists t3 (t2 <= t3 and t3 = t2 + clk)) and
         (forall t3 ((t2 <= t3 and t3 = t2 + clk) imply
           (exists t4 (t3 <= t4 and
             (
               (exists t5
                  (t4 <= t5 and (size = 1 or t5 >= t0 + mtime) and
                   t5 = t4 + P31)) and
               (forall t5
                  (t4 <= t5 and (size = 1 or t5 >= t0 + mtime) and
                   t5 = t4 + P31) imply true)
             )
           ))
         ))
        )
      )
    )
  )
)))
```

状態 s10 の must トレース可能性判定式

```
forall mtime forall P31 forall wait forall clk (
(mtime = 100 and P31 = 1 and wait = 100 and clk = 2) imply
(exists size
(
  (exists t0 (true) and
    (exists t1 (t0 <= t1 and true) and
      (exists t2 (t1 <= t2 and true) and
        (
         (exists t3 (t2 <= t3 and t3 = t2 + clk)) and
         (forall t3 ((t2 <= t3 and t3 = t2 + clk) imply
           (exists t4 (t3 <= t4 and
             (
               (exists t5
                 (t4 <= t5 and (size = 1 or t5 >= t0 + mtime) and
                  t5 = t4 + P31)) and
               (forall t5
                 (t4 <= t5 and (size = 1 or t5 >= t0 + mtime) and
                  t5 = t4 + P31) imply
                 ((exists t6 (t5 = t6+wait)) and
                  (forall t6 (t5 = t6+wait) imply true)
                 )
               )
             )
           ))
         ))
        )
      )
    )
```

```
    )
)))
```