Omega test and beyond

酒井 政裕

Proof Summit 2012

目次

- 自己紹介とか
- 算術とは
- Omegaの中身
- まとめ

自己紹介

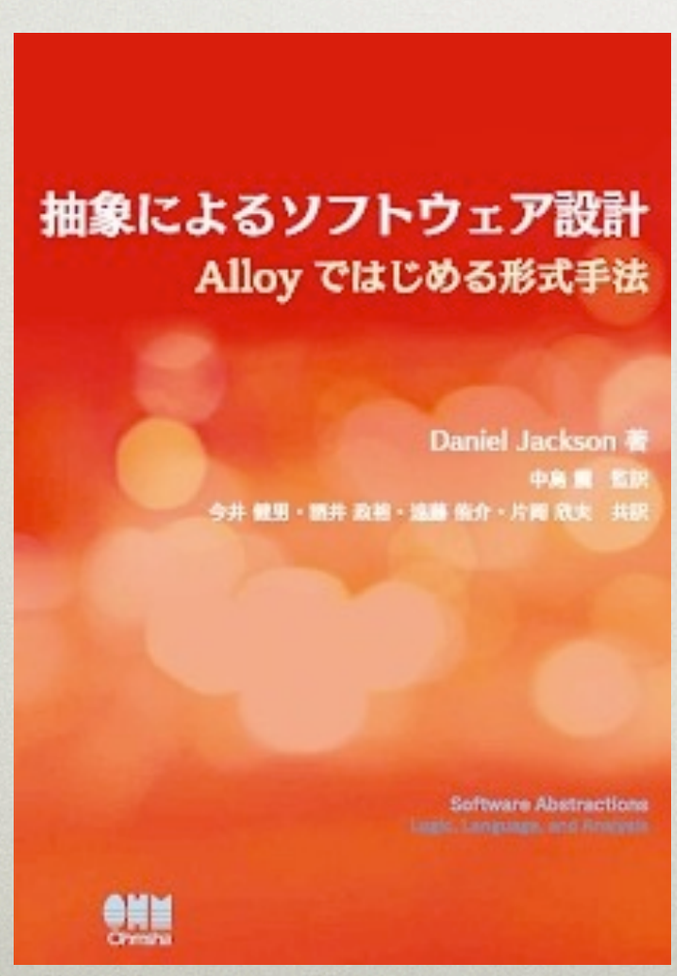
酒井 政裕

- Blog: ヒビルテ
- Twitter: @masahiro_sakai

- Haskeller
- Agda: Agda1のころ遊んでた。Agda2は少しだけ。
- Coq: 初心者 (多分、Coq力 0.3 くらいの雑魚)
- 昨年、Alloy本の翻訳を出版 (共訳)
- Proofsummi2011 では自動定理証明の簡単な紹介

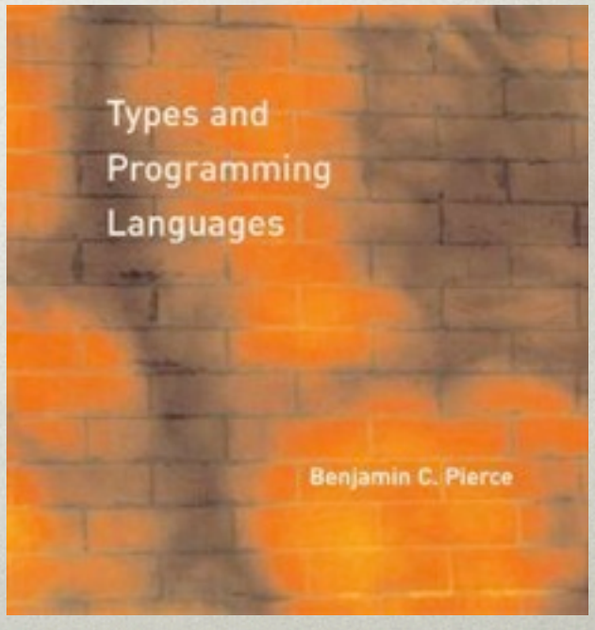
宣伝: Alloy本 & TAPL

- 抽象によるソフトウェア設計（Daniel Jackson 著）
  - 中島震監訳 今井健男・酒井政裕・遠藤侑介・片岡欣夫共訳
    オーム社



Types and Programming Language (TAPL)
Benjamin C. Pierce 著

現在翻訳中



最近の興味:

色々な決定手続きのオモチャ実装

- Presburger Arithmetics
  - Omega Test
  - Cooper's Algorithm
- Real Arithmetics
  - Fourier-Motzkin variable elimination
  - Simplex method
  - Gröbner basis (Buchberger)
- (Mixed) Integer Programming
  - Branch-and-bound
  - Gomory's Cut
  - Conti-Traverso
- SAT / MaxSAT / Pseudo Boolean (DPLL, CDCL)
- Uninterpreted function (Congruence Closure)
- 実装したいと思ってるもの
  - CAD (Cylindrical Algebraic Decomposition)
  - cut-sat (Cutting to the Chase)

github.com/msakai/toysolver

最近の興味:

色々な決定手続きのオモチャ実装

- Presburger Arithmetics
  - Omega Test
  - Cooper's Algorithm
- Real Arithmetics
  - Fourier-Motzkin variable elimination
  - Simplex method
  - Gröbner basis (Buchberger)
- (Mixed) Integer Programming
  - Branch-and-bound
  - Gomory's Cut
  - Conti-Traverso
  - CAD (Cylindrical Algebraic Decomposition)
  - cut-sat (Cutting to the Chase)

本当は実数には限らないけれど。
あと、きっと@erutuf13さんが、
熱く語ってくれるはず

github.com/msakai/toysolver

今日の話: Omega

- そのなかの Omega Test

  ≒ Coq の omega タクティック

- 自然数や整数に関する自動証明

今日の話: Omega

- そのなかの Omega Test

  ≒ Coq の omega タクティック

  自然数や整数に関する自動証明

Q. omegaを使ったことある人？

Omega使用例

```
Require Import Omega.
Open Scope Z_scope.

Goal forall m n, 1 + 2 * m <> 2 * n.
  intros.
  omega.
Qed.
```

目次

- 自己紹介とか
- 算術とは
- Omegaの中身
- まとめ

算術

- 算術 = 自然数に関する理論
  - 自然数ではなく整数を対象にすることもあるが、本質的には変わらない
- 様々な算術の体系
  - ペアノ算術 PA (1889)
    - ゲーデルが決定不能性を証明 (1930)
  - ロビンソン算術 Q (1950) ≒ PA – 帰納法
    - PAより相当弱いが決定不能
  - Presburger 算術 (1929) ≒ PA – 掛け算
    - 決定可能！

算術の体系と問題クラス(?)

真の算術

多分、色々
間違ってるけど

Peano Arithmetics

Presburger Arithmetics
(掛け算なし = 線形)

Robinson's Q
(帰納法無し)

QF_LIA

整数線形計画

ディオファントス
方程式

∀無し (Σ_1?)

Presburger 算術

- 公理
  - $\neg(0 = x + 1)$
  - $x + 1 = y + 1 \rightarrow x = y$
  - $x + 0 = x$
  - $(x + y) + 1 = x + (y + 1)$
  - $(P(0) \wedge \forall x(P(x) \rightarrow P(x + 1))) \rightarrow \forall y\ P(y)$.
- 注: 定数との掛け算は、足し算の繰り返しで書けるので、表現可能 (e.g. $3 \times a = a+a+a$) 。**線形**多項式は扱える。

目次

- 自己紹介とか
- 算術とは
- Omegaの中身
- まとめ

Omegaの中身

- Presburger 算術は決定可能 ⇒ 自動証明可能
- Coqのomegaタクティックは Presburger算術の自動証明をする
  - 本体はCoqのプラグイン (omega.ml)
  - Coqの証明項(Gallina)を生成
  - 生成された証明項で使う補題等 (OmegaLemmas.v)

Omegaの中身

- Presburger 算術は決定可能 ⇒ 自動証明可能
- Coqのomegaタクティックは Presburger算術の自動証明をする
  - 本体はCoqのプラグイン (omega.ml)
  - Coqの証明項(Gallina)を生成
  - 生成された証明項で使う補題等 (OmegaLemmas.v)

Q. omegaの中身を知っている人いる?

Omegaの元ネタ

- "The omega test: a fast and practical integer programming algorithm for dependence analysis," by W. Pugh
  - in *Proceedings of the 1991 ACM/IEEE conference on Supercomputing*, ser. Supercomputing '91.
  - http://www.cs.umd.edu/~pugh/papers/omega.pdf
  - 並列化のため、ループの依存性を、添字に関するPresburger算術の論理式の真偽で判定

Omega Test の仕組み

- 実数の場合の量化子除去手法

  Fourier-Motzkin variable elimination を応用して実現

- 量化子除去 (Quantifier elimination; QE)

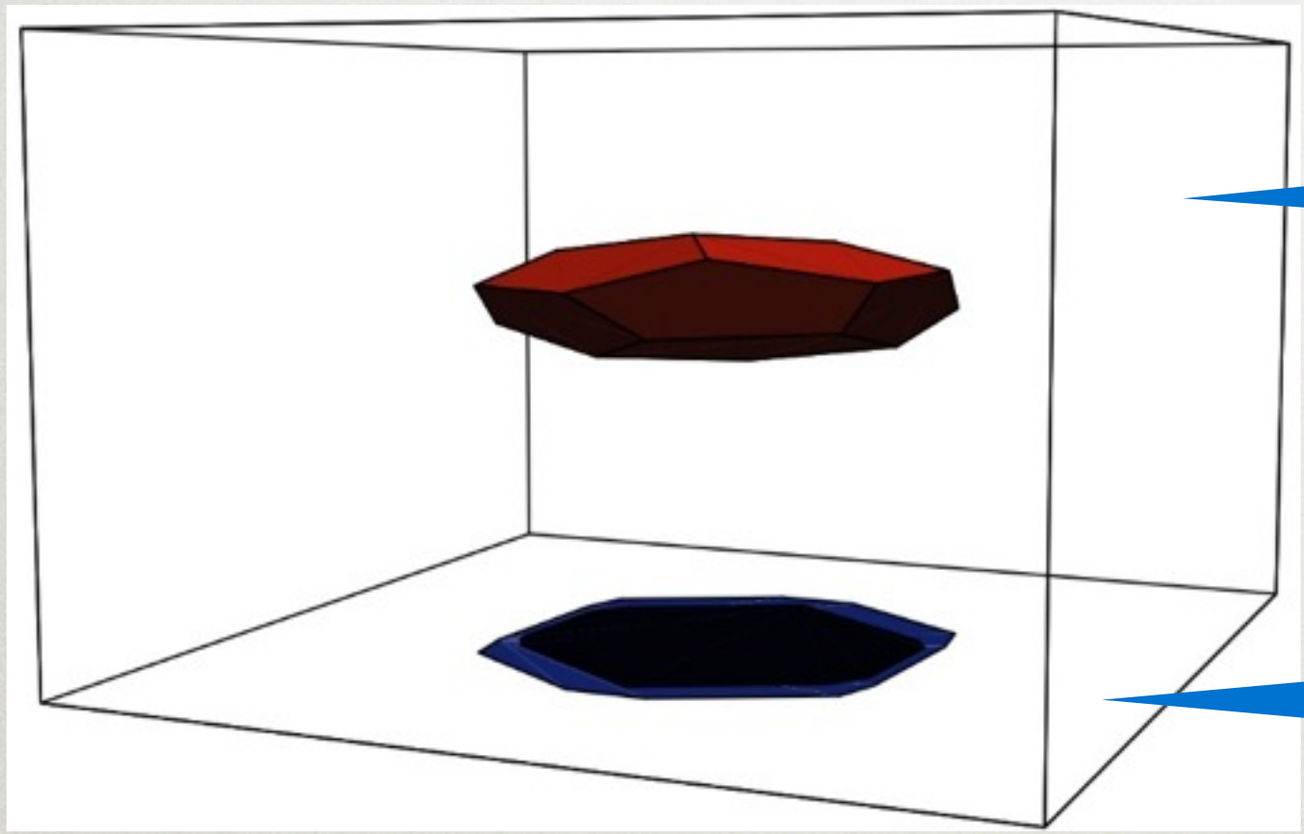
  - 量化子を含む論理式と**同値**で**量化子を含まない**論理式を求める。

  - 例: $\forall x : R.\ 2 < x \vee x \leqq 3 \Leftrightarrow \neg\exists x : R.\ 2 \geqq x \wedge x > 3 \Leftrightarrow \neg(2 > 3)$

  - 量化子を含まない論理式は真偽が自明

Fourier-Motzkin variable elimination: 射影

$$\exists x.\ \exists y.\ \exists z.\ P(x,y,z) \quad \Leftrightarrow \quad \exists x.\ \exists y.\ Q(x,y)$$



P(x,y,z) が真になる領域

影

= ∃z. P(x,y,z) が真になる領域

= Q(x,y) が真になる領域

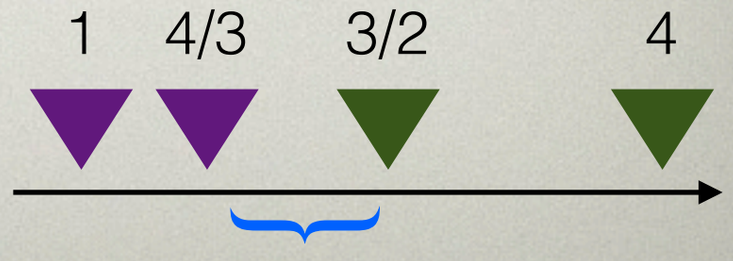
The Omega Test: a fast and practical integer programming algorithm for dependence analysis

射影を繰り返していくと、変数が消えていく

Fourier-Motzkin variable elimination の基本

1. $P = \exists x.\ Q_1(x) \wedge \ldots \wedge Q_n(x)$ で $Q_i(x)$ が x を含む不等式とする
   - ただし不等号は≦,≧に限る
   - 例: $\exists x.\ x \geqq 1 \ \wedge\ 3x \geqq 4 \ \wedge\ 2x - 3 \leqq 0 \ \wedge\ x \leqq 4$
2. 各 $Q_i(x)$ を x について解いて $\exists x.\ (L_1 \leqq x \wedge \ldots \wedge L_p \leqq x) \wedge (x \leqq U_1 \wedge \ldots \wedge x \leqq U_q)$
   - 例: $\exists x.\ 1 \leqq x \ \wedge\ 4/3 \leqq x \ \wedge\ x \leqq 3/2 \ \wedge\ x \leqq 4$
3. $\max\{ L_i \}_i \leqq x \leqq \min\{ U_j \}_j$ の範囲のxが条件を満たすが、

   この範囲が空でないのは $\bigwedge_i \bigwedge_j\ L_i \leqq U_j$ と同値
   - 例: $1 \leqq 3/2 \ \wedge\ 1 \leqq 4 \ \wedge\ 4/3 \leqq 3/2 \ \wedge\ 4/3 \leqq 4$



Fourier-Motzkin variable elimination: 一般の場合(1)

- 厳密不等号 (<, >)を扱うには、

  下界 $L_i \lhd x$ と上界 $x \lhd' U_j$ を組み合わせるとき

  - $\lhd = <$ もしくは $\lhd' = <$ ならば $L_i < U_j$ にする。
    - 例: $(\exists x.\ 2 < x \wedge x \leqq 3) \Leftrightarrow 2<3$
  - $\lhd = \lhd' = \leqq$ ならば、これまで通り $L_i \leqq U_j$ にする。
- 本体 Q_i にxが現れない場合: くくりだして帰着
  - $\exists x.\ Q(x) \wedge Q_i \quad \Leftrightarrow (\exists x.\ P(x)) \wedge Q_i$

Fourier-Motzkin variable elimination: 一般の場合(2)

- 本体がDNFの場合:

  $\exists x.\ (Q_{1,1} \wedge \dots) \vee \dots \vee (Q_{n,1} \wedge \dots)$

  $\Leftrightarrow$

  $(\exists x.\ Q_{1,1} \wedge \dots) \vee \dots \vee (\exists x.\ Q_{n,1} \wedge \dots)$

  なので、量化子を分配すれば帰着できる

- DNF = Disjunctive Normal Form (選言標準形)

  原子論理式の連言(and)の選言(or)

Fourier-Motzkin variable elimination: 一般の場合(3)

- toDNF : Formula → DNF
  toDNF = …

- elim : Var × DNF → DNF
  elim(x, P) = (∃*x*. *P*)を量化子除去した結果

- FMVE : Formula → DNF
  FMVE(P)　= toDNF(P)　if P が量化子を含まない

  FMVE(C[∃x. P]) = FMVE(C[elim(x, FMVE(P))])
  FMVE(C[∀x. P]) = FMVE(C[¬∃x. ¬P])

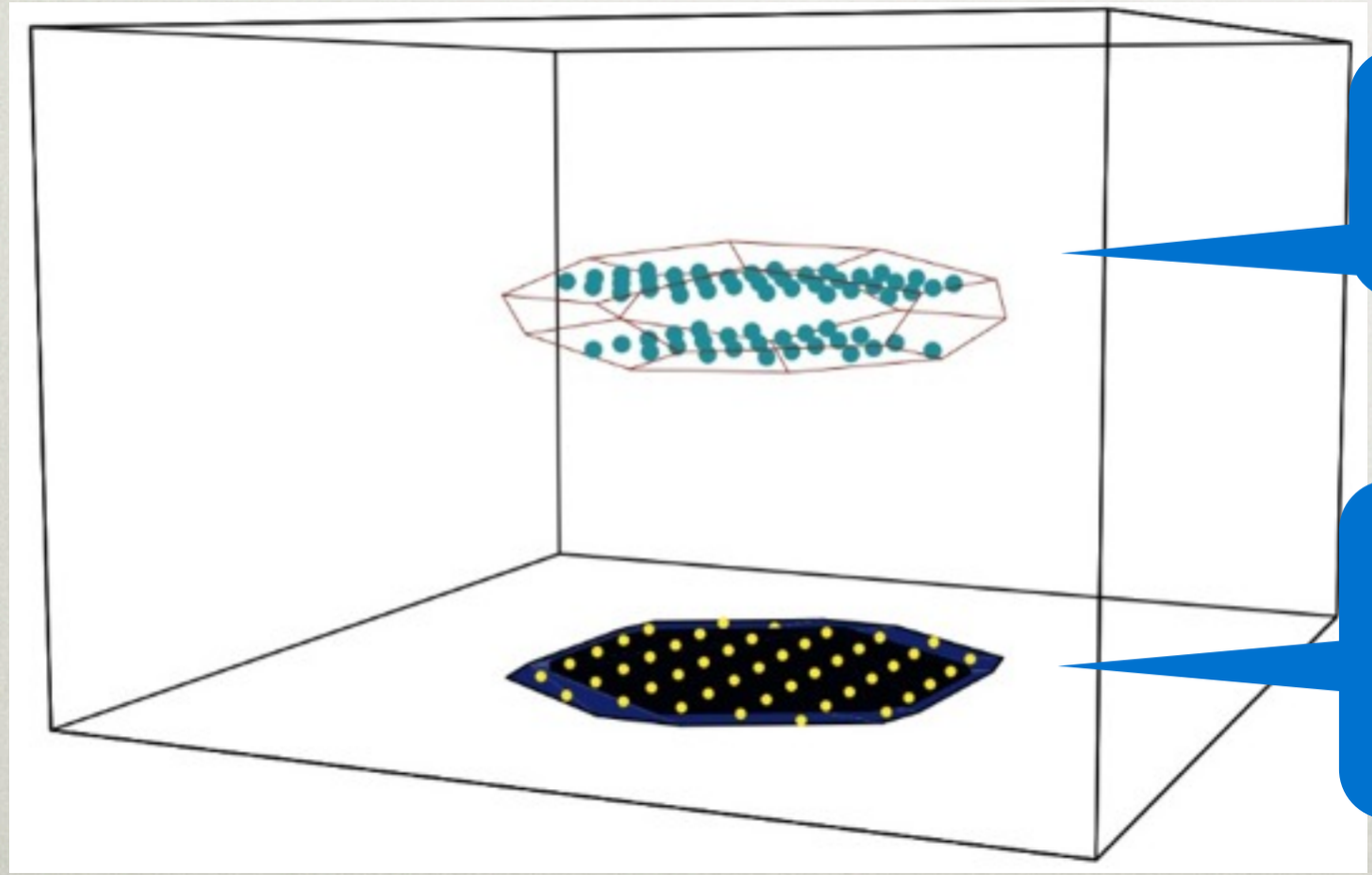
Fourier-Motzkin variable elimination: Coq

- Coq にも fourierタクティックがあるよ。
  - omegaほどには知られてないけど
  - 使用例:

```
Require Import Fourier.

Goal forall x y : R, x-y>1 -> x-2*y<0 -> x>1.

  intros.

  fourier.

Qed.
```

Omega Test: 概要

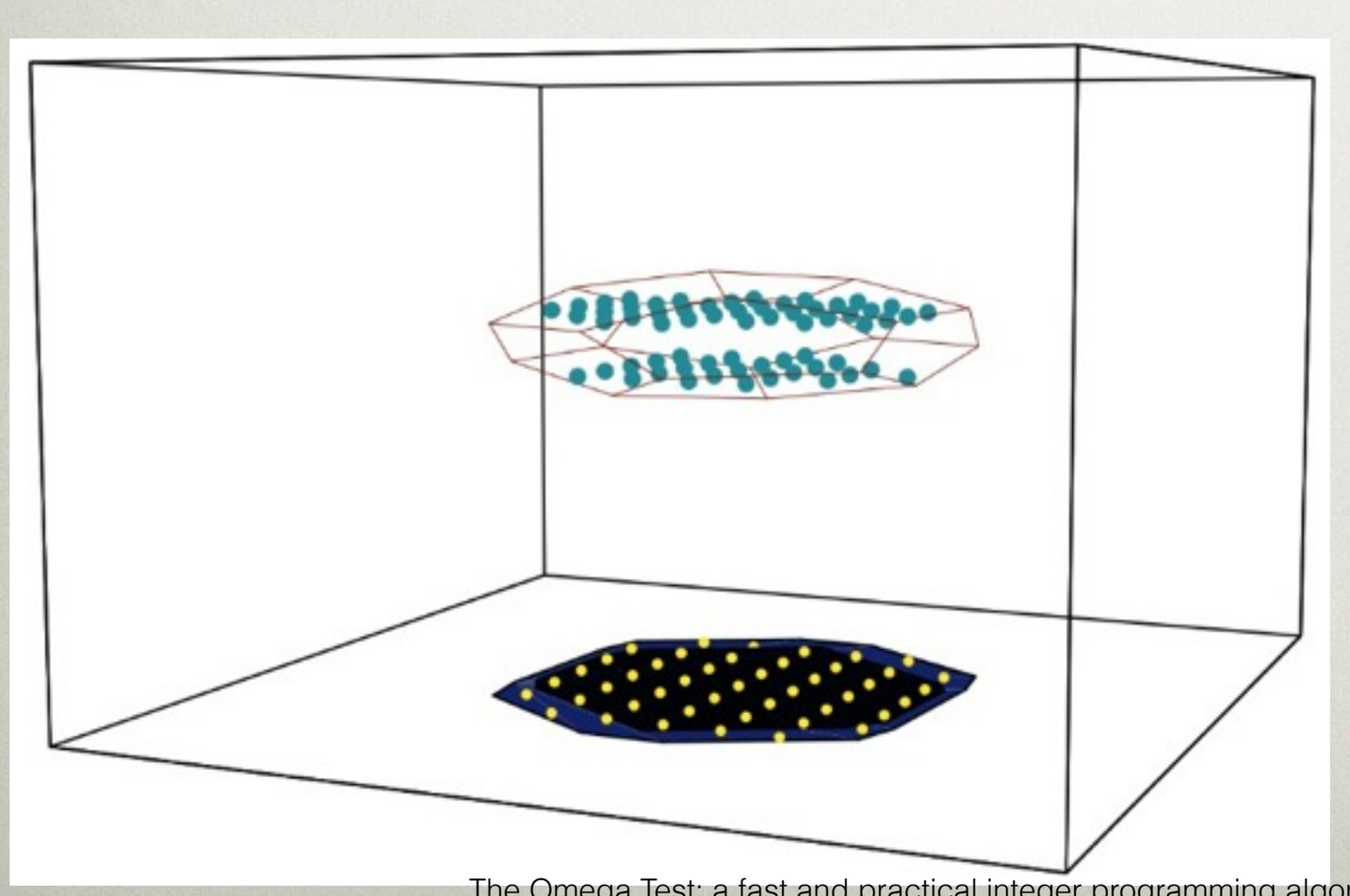
- Fourier-Motzkin variable elimination で射影
- ただし、 射影結果に格子点が含まれても、
  射影元に格子点が含まれるとは限らない

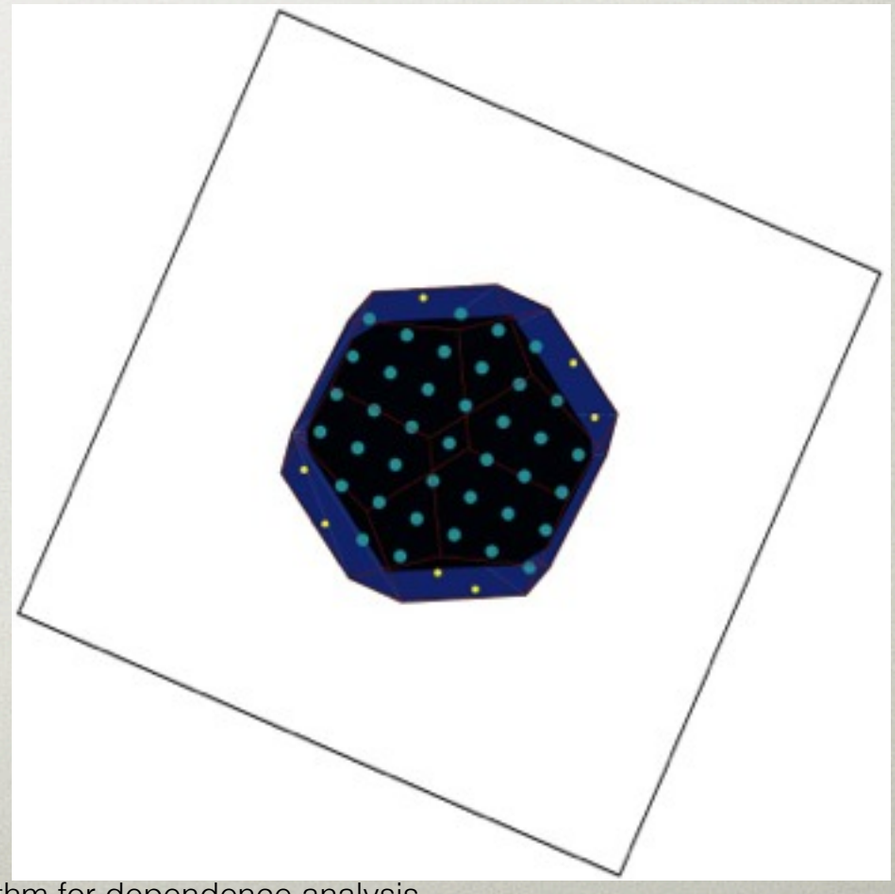


The Omega Test: a fast and practical integer programming algorithm for dependence analysis

Omega Test: 工夫

- Omega Test の工夫
  - 射影元が充分厚い部分なら、影の格子点の上には、必ず格子点がある (影の黒い部分 dark shadow)
  - 定理1: $\underline{(a-1)(b-1) \leq ad - bc} \Rightarrow (\exists x : Z.\ \underline{c \leq a*x \wedge b*x \leq d})$
    dark shadow を表す式　　射影元の領域





The Omega Test: a fast and practical integer programming algorithm for dependence analysis

Omega Test: 流れ

∃x : Z. P(x) の真偽を判定したい

1. P の real shadow と dark shadow を計算
2. real shadow に格子点が含まれない ⇒ 偽
3. real shadow = dark shadow ⇒ FMVEと同じ
4. dark shadow に格子点が含まれる ⇒ 真
5. それ以外 (Omega Test Nightmare) ⇒
   real shadow の格子点の上の格子点を網羅チェック

Omega Test: その他

- 網羅的チェックは説明が面倒なので略
- 量化子除去: $\exists x.\ P(x) \Leftrightarrow$ (dark shadow $\vee$ 網羅的な場合の式)
- 様々な工夫
  - 等号の方が効率的に処理可能なので、できるだけ等号を使う
  - GCD Test: $a_1x_1 + \ldots + a_nx_n = c$ は c が $\gcd\{a_i\}_i$ で割り切れないなら充足不能
  - $\exists\ x\ y\ z.\ P(x,y,z)$ なら x, y, z のうち
    real shadow と dark shadow が一致する変数から除去
    (網羅的なチェックによる分岐を出来るだけ先送り $\Rightarrow$ 組み合わせ爆発を回避)
  - などなど

CoqのOMEGAタクティック

- Coq のomegaタクティック
  - 残念ながら Omega Test Nightmare の場合の処理は実装されてない。 (名前負けしてるぞ〜)
  - なので、完全な決定手続きではなく、解けない場合も……

```
Require Import Omega.
Open Scope Z_scope.
Goal forall x y,
      27 <= 11 * x + 13 * y <= 45 ->
      -10 <= 7 * x - 9 * y <= 4 ->  False.
 intros.
 omega.
```

⇒ *Error: Omega can't solve this system*

CoqのOMEGAタクティック

- omegaの代替としてPsatzがあり、

  そのliaタクティックなら解けるらしいが...

```
Require Import Psatz.
Require Import ZArith.
Open Scope Z_scope.
Goal forall x y,
      27 <= 11 * x + 13 * y <= 45 ->
      -10 <= 7 * x - 9 * y <= 4 ->  False.
 intros.
 lia.
```

CoqのOMEGAタクティック

- omegaの代替としてPsatzがあり、

  そのliaタクティックなら解けるらしいが...

```
Require Import Psatz.
Require Import ZArith.
Open Scope Z_scope.
Goal forall x y,
     27 <= 11 * x + 13 * y <= 45 ->
     -10 <= 7 * x - 9 * y <= 4 ->  False.
 intros.
 lia.
```

⇒

Anomaly: Uncaught exception Unix.Unix_error(1, "open", "lia.cache").
Please report.

CoqのOMEGAタクティック

- omegaの代替としてPsatzがあり、

  そのliaタクティックなら解けるらしいが...

```
Require Import Psatz.
Require Import ZArith.
Open Scope Z_scope.
Goal forall x y,
     27 <= 11 * x + 13 * y <= 45 ->
     -10 <= 7 * x - 9 * y <= 4 ->  False.
 intros.
 lia.
```

(´・ω・`)

⇒

Anomaly: Uncaught exception Unix.Unix_error(1, "open", "lia.cache").
Please report.

CoqのOMEGAタクティック

- omegaの代替としてPsatzがあり、

  そのliaタクティックなら解けるらしいが...

```
Require Import Psatz.
Require Import ZArith.
Open Scope Z_scope.
Goal forall x y,
     27 <= 11 * x + 13 * y <= 45 ->
     -10 <= 7 * x - 9 * y <= 4 ->  False.
 intros.
 lia.
```

⇒

Anomaly: Uncaught exception Unix.Unix_error(1, "open", "lia.cache").
Please report.

(´・ω・`)

【追記】 CoqIDEを実行しているカレントディレクトリが書き込み不能という原因でした

まとめ

- 実数・整数に関する線形式の決定手続と量化子除去を紹介
  - Fourier-Motzkin variable elimination
  - Omega Test
- タクティックの意義
  - 決定不能な論理の表現力と
  - 決定手続による自動化の
  - いいとこ取り
- 決定手続き自体も面白いので一緒に遊びませんか？

	一般の論理	決定可能な論理
表現力	高い	低い
自動証明	半決定可能(爆発しがち)	決定可能(それなりにスケール)

まとめ

Happy

Theorem

Proving

おまけ: 非線形の場合

- 実数に関する線形多項式を扱うFourier-Motzkin
- 整数に関する線形多項式を扱うOmega Test
- を紹介したけど、実数なら非線形でも実は解ける (実閉体)

	実数	整数
線形	Fourier-Motzkin (double-exponential)	Omega Test, Cooper's method (triple-exponential)
非線形	Cylindrical Algebraic Decomposition (double-exponential)	(undecidable)