LMNtal によるプレスバーガー算術の制約ソルバの実装に向けて

発表日： 2022 年 10 月 26 日

早稲田大学 基幹理工学研究科
情報理工・情報通信専攻

学籍番号：5122F015-3
今川　連

第 1 章

概要

1.1 本発表の概要

- ハイパーリンクを用いて，制約ソルバを書く．
- プレスバーガー算術の範囲の制約を解くためのソルバを LMNtal で実装することを目指す．
- まずは，自前で簡単なソルバを書いた．
- これからの課題：decision procedure を読むなりして， presburger arithmetic のソルバを書く．

1.2 注意

今から紹介する方法は，一般とあまりにもかけ離れている可能性があります．去年の春学期に decision procedure を読んだはずですが，それも含めて知識があまりにも抜けていることに気づきました．なので，この発表の後は，decision procedure を読んで，まともなソルバを書こうと思っています．

これからの話は，全て間違っていると思って聞いて，違和感のある部分を指摘してください．

第 2 章

設計

2.1 今回書いたソルバの概要

事実 (fact) を宣言し，入力した質問 (question) の真偽を判定する．この時，事実が充足不可能なら必ず真を返す．
$\sim$ 全ての変数について，事実 $\Longrightarrow$ 質問 を判定する．

- 扱う制約は，自然数，変数，$+$，$=$，$(\leq)$ からなる．

$$t \quad : \quad \mathbb{N} \;\Big|\; \mathrm{var} \;\Big|\; t+t \qquad \text{(term)}$$

$$af \quad : \quad (t \leq t) \;\Big|\; t = t \qquad \text{(atomic formula)}$$

$$f \quad : \quad af \;\Big|\; f \wedge f \qquad \text{(formula)}$$

2.2 実装

今回の実装では，変数はハイパーリンクで表現して，同じハイパーリンクに属する場合は同じ値を持つということを表現している．例えば，

$$\text{facts: } x_1 = x_2 \wedge x_3 = x_4$$

が与えられた場合，x_1 と x_2，x_3 と x_4 はそれぞれ同じハイパーリンクに属する．また，数値が決定できる場合は，n というアトムをかませて，int 型の値をハイパーリンクに登録

する．

```
hh_eq@@ eq(!X,!Y) :- !X >< !Y.
hi_eq@@ eq(!X,N) :- int(N) | n(N, !X).
ih_eq@@ eq(N,!X) :- int(N) | n(N, !X).
```

同じハイパーリンク内に異なる 2 つの自然数が存在する場合は false.

```
n(N,!X), n(M,!X) :- N =\= M | antecedent(false).
```

異なる 2 つのハイパーリンクが同じ自然数を持っていた場合は併合する．

```
n(N,!X) \ n(M,!Y) :- N =:= M | !X >< !Y.
```

decision procedures p.86

等式だけではなくて，未解釈の関数についても同じように併合している．足し算も式のまま併合するべき？

ここまでは普通に見えるか意見が頂きたいところ

実装の理想形：性質（公理？）をそのまま書く

pertial order constraint

```
leq_duplicate@@ leq(!X,!Y), leq(!X,!Y) :- leq(!X,!Y).
leq_reflexivity@@ leq(!X,!X):- .
leq_antisymmetry@@ leq(!X,!Y), leq(!Y,!X) :- eq(!X,!Y).
leq_transitivity@@ leq(!X,!Y), leq(!Y,!Z)\ :-uniq(!X,!Y,!Z)
    | leq(!X,!Z).
```

とか，

```
less_duplicate@@ less(!X,!Y) \ less(!X,!Y) :- .
less_reflexivity@@ less(!X,!X):- false.
less_antisymmetry@@ less(!X,!Y), less(!Y,!X) :- false.
less_transitivity@@ less(!X,!Y), less(!Y,!Z)\ :-uniq(!X,!Y,!
    Z) | less(!X,!Z).
```

みたいな形．`https://en.wikipedia.org/wiki/Inequality_(mathematics)`

decision procedures p.86

等式理論において，

$$x_1 = x_2 \lor (x_2 = x_3 \land x_4 = x_5 \land x_5 \neq x_1 \land \cancel{F(x_1) \neq F(x_3)})$$

の論理和の部分は，ケースを分けて考えると書いてある．→ LMNtal だったらどうやる？
例えば，

$$(x_1 = x_2 \lor x_1 = x_3) \land (x_2 = x_4 \land x_1 \neq x_2)$$

は，以下に変形できる（Disjunction Normal Form）

$$(x_1 = x_2 \land x_2 = x_4 \land x_1 \neq x_2) \lor$$
$$(x_1 = x_3 \land x_2 = x_4 \land x_1 \neq x_2)$$

ここで，今までの方法で単に

1. !X1 >< !X2
2. !X2 >< !X4
3. より，!X1 = !X2 = !X4
4. !X1 = !X2 と neq(!X1,!X2) は矛盾するので 第 1 節が unsatisfiable.

とすると，第 2 節も unsatisfiable になる（併合が残ってしまうので）

- 解決策の案

DNF（Disjunction Normal Form） を仮定して，disjunction 毎に独立な変数名を宣言する．つまり，上の式は以下のようにする．

$$(x_1 = x_2 \land x_2 = x_4 \land x_1 \neq x_2) \lor$$
$$(x_{1_1} = x_{1_3} \land x_{1_2} = x_{1_4} \land x_{1_1} \neq x_{1_2})$$

さらに，同じ節の等式は，同じハイパーリンクに属するとする．すると，以下のようにプログラムを書き換えると，できそう？ or の埋め込み（3 節以上の or ）もできるようになっている．（できるというのは，充足可能か判定するということ．量化子はまだ考えていない←まだ読めていないが，量化子はうまく除去できるのかな〜という期待 quantifier elimination という言葉をよく見るので）

```
eq(!X1,!X2,!H1), eq(!X2,!X3,!H1), neq(!X1,!X2,!H1).
eq(!X1_1,!X1_3,!H2), eq(!X1_2,!X1_4,!H2), neq(!X1_1,!X1_2,!H2).
ans = or(!H1,!H2).

```

```
%or
Ret = or(!X,C), unsat(!X) :- Ret = or(unsat,C).
Ret = or(C,!X), unsat(!X) :- Ret = or(C,unsat).
Ret = or(unsat,unsat):- Ret = unsat.
Ret = or(sat,C) :- ground(C) | Ret = sat.
Ret = or(C,sat) :- ground(C) | Ret = sat.

%neq
neq(!X,!Y,!H) :- !X = !Y | unsat(!H).

% one variable has 2 or more integer
n(N,!X,!H), n(M,!X,!H) :- N =\= M | unsat(!H).

% two variable has same integer
n(N,!X) \ n(M,!Y) :- N =:= M | !X >< !Y.

%equality
hh_eq@@ eq(!X,!Y,!H) :- !X >< !Y.
hi_eq@@ eq(!X,N,!H) :- int(N) | n(N,!X,!H).
ih_eq@@ eq(N,!X,!H) :- int(N) | n(N,!X,!H).

%sat
%全ての伝播が終わって,unsat でなければ sat
Ret = or(!X,C) :- ground(C) | Ret = sat.
Ret = or(C,!X) :- ground(C) | Ret = sat.
```

さっきまでは，DNF に対する標準的な処理＋ equality, inequality
ここからは，線形論理が入る．（decision procedures p.97 ）

線形論理のシンタックスは以下の通り．ドメインは整数

$$formula : formula \wedge formula \mid (formula) \mid atom$$

$$op : = \mid \leq \mid <$$

$$sum : term \mid sum + term$$

$$term \ : \ identifier \mid constant \mid constant \, identifier$$

$identifier$ は，整数上の変数，$constant$ は整数定数.

decision procedures p.98 Linear Arithmetic の ソルバ

- Simplex: 数値最適化のための最も古いアルゴリズムの 1 つ．実数上の線形制約の論理積を与えて，目的関数の最適値を求める．
- general Simplex: 目的関数を与えない Simplex．充足可能かどうかを決定する．
- Fourier-Motzkin variable elimination: Simplex より効率は劣るが，実装が比較的容易．実数上の線形制約の論理積の充足可能性を決定.
- Omega test: Simplex より効率は劣るが，実装が容易．整数上の線形制約の論理積の充足可能性を決定.

omega test について考えていく.

入力は以下の形の論理積.

$$\sum_{i=1}^{n} a_i x_i = b \text{ or } \sum_{i=1}^{n} a_i x_i \leq b$$

例えば以下のような式が当てはまる.

$$2x_1 + 3x_2 = 10 \wedge x_1 + -1x_2 \leq 0$$

係数 a_i は整数を仮定.（最小公倍数をかけて全部整数係数になるように変形する.）

手順 1：equality の左辺の係数（$a_1, \ldots, a_n$）の最大公約数 g で両辺を割る．この時，g が b を割り切れなかったら unsat.

補足：

$$\sum_{i=1}^{n} a_i x_i = b$$

について

$$b \, mod \, gcd(a_1, \ldots, a_n) = 0 \Leftrightarrow \text{整数解が存在する}$$

手順 2：inequality の左辺の係数（$a_1, \ldots, a_n$）の最大公約数 g で両辺を割る．右辺は小数点以下切り捨て（普通の int 型の割り算）.

正規化部分.

```
%%mul(constant, !Identifier, !Hlink)
%% hlink is unique for each equality
gcd0@@ Ret = mul(N,!X,!E) \:- int(N), uniq(!X,!E) | gcd(N,!E
```

```
        ).

    gcd1@@ gcd(0,!H) :- .
    gcd2@@ gcd(N,!H) \ gcd(M,!H) :- N =< M, G = M-N | gcd(G,!H).

    %normalization using gcd of identifiers
    %omega test の入力の右辺は整数定数のはずなので,等式,不等式ごとに n(N,!E) が 1つ繋がってるはず
    norm_eq_unsat@@ eq(C,n(N,!E),!H), gcd(G,!E) :- ground(C), N mod G =\= 0 | unsat(!H).
    norm1@@ gcd(G,!E) \ Ret = mul(N,!X,!E):- A = N / G, uniq(G,!X,!E) | Ret = mul(A,!X,!E).
    norm2@@ Ret = n(N,!E), gcd(G,!E) :- A = N / G | Ret = n(A,!E).
```

手順 3：係数が 1, -1 となるような変数を探す．一般性を損なわないために，$n = j, |a_j| = 1$ とすると，

$$x_n = b/a_n - \sum_{i=1}^{n-1} a_i/a_n\, x_i$$

で，全ての x_n を右辺に置き換える．→ LMNtal だったらどうやる？？？？

例えば，

$$3x_1 + 3x_2 = 9 \wedge 2x_1 + 5x_2 = 12$$

なら，正規化して

$$x_1 + x_2 = 3 \wedge 2x_1 + 5x_2 = 12$$

1 つ目の等式を変形して

$$x_1 = 3 + -1x_2$$

だから，2 つ目の式に代入して，

$$6 + -2x_2 + 5x_2 = 12 \tag{2.1}$$
$$3x_2 = 6 \tag{2.2}$$
$$x_2 = 2 \tag{2.3}$$

より充足

ここで、疑問．変数を消去した後の式は，再び正規形にするべきか．→するべき

これは，mul, mul_n, mul_p という 3 種類のアトムを用意することによって，解決．mul になったら gcd を求めて正規化する．mul_n は正規化済みであることを表す（gcd アトムをそれ以上生成しないため）elimination が起こったら，一旦 mul_n を mul_r に変換して，全部変換した後に mul_p を mul に一斉に変換する（mul に変換しながら進めると上にある gcd ルールが先に発火して大変なことになる）elimination が起こったら token_1 アトムが生成されて，全部変換したら token_1 アトムが ready_1 アトムに変換されて，gcd アトムの生成が終わったら ready_1 アトムは削除される．

不定方程式は，上に書いたように，gcd で割り切れたら整数解を持つし，割り切れなかったら整数解を持たないので，式の conjunction が全て処理し終わった後（式が 1 つになったとき）に解の判定をすれば，充足可能か決定できる．ということで変数消去によって式を減らしていっている．

手順 4:無理やり係数が 1 か -1 の式を作る
係数が 1, -1 のどちらかであるような変数がなくなってしまったら，

1. 絶対値が最も小さい非ゼロの係数を持つ変数を選ぶ
2. いくつかの係数が 1 か -1 になるまで繰り返し式変形を行う
3. 係数の絶対値が 1 になった変数は先ほどのように削除する

symmetric mod を以下のように定義

$$a \,\widehat{mod}\, b = \begin{cases} a \,mod\, b & : \ a \,mod\, b < b/2 \\ (a \,mod\, b) - b & : otherwise \end{cases}$$

$$m = a_n + 1 \; (a_n\text{は全項の中で最小の係数})$$

として，次の数式を追加．（LMNtal で実装するときは，変換後の式を基にしてアトムを追加？）

$$\textstyle\sum_{i=1}^{n} (a_i \,\widehat{mod}\, m) x_i = m\sigma + b \,\widehat{mod}\, m$$

参考文献