ソフトウェア基礎論24—2
(1988. 2. 12)

CAL: 制約論理プログラミングの理論と実例

CAL: Theory and Examples of Constraint Logic Programming

坂井　公　　相場　亮
Ko SAKAI　　Akira AIBA

(財) 新世代コンピュータ技術開発機構
ICOT

概要

制約論理プログラミング(CLP)とは，ある領域における制約を書いたり解いたりする能力を加えて，論理プログラミングを拡張しようとするものである。CAL は，CLP 言語の一例で多項式による方程式の形で制約を記述することができる。従来，提案されてきた，CLP 言語は，線形方程式や線形不等式解くことに努力を集中してきたのに対し，CAL は非線形方程式の形の制約を解くところにその特長がある。本論文では，CAL を含めたCLP の一般的な意味論を与え，その枠組みの上での CAL の正当性を示す。

1．はじめに

Jaffer と Lassez は，制約論理プログラミング (CLP) のパラダイム [JaL86] を提唱した。類似のパラダイム（あるいは言語）は，Colmerauer [Col 82], [Col87] や Dincbus [Din 87] も提唱している。Prolog に代表される論理プログラミング(LP) 言語は，単化(unification)に計算を行う言語であるといってよいが，CLP は，単化を制約というより広い概念に置き換えることでさらに言語の記述力を向上させようとするものである。Jaffarらは，CLP を提唱するにあたって，3つの基準を挙げた。

(1) 複雑な問題をなるべく簡単にかつ自然に記述できる。

(2) このような問題の記述に対して，実用的な速度を持つ処理系が作れる。

(3) 言語の宣言的意味論および操作的意味論の間に統一的な枠組みが存在する。

基準(1)の答えとして，CLP は問題を記述するために「制約」を採用した。基本的なアイデアは，新しいものとは言えない [Ste 81], [Fik 70], [StS 78]。しかしながら，LPの枠組みの上に展開していることが新しい。これによって得られる利点は多い。問題を宣言的に記述できることが LP の特長であったが，これは制約という形で問題を記述する場合さらに強い要求となる。また，そのためにこそ基準(3)が重要になる。問題を解くために細かい制御情報を記述する言語では，(3)は期待できない。これには，操作的モデルの単純さが必要である。LP を利用したことで，さまざまな制御構造を導入するのではなしに，ゴール・リダクションの一般化として操作モデルを定義できる。

LPの特徴は，そのさまざまな意味論にある。論理的意味論，関数的意味論，操作的意味論などであるが，これらはすべてエレガントで単純なだけではなく，互いに一致している [ApE 82], [EmK 76], [Jaf 85], [Llo 84]。Jaffer らは，これらの意味論が，CLPでも，古典的なLPの場合の拡張になっているということを示した，あるいは，示そうとしている。また，彼らは，加えてCLPの代数的意味論を提唱した。

CLPでは，プログラムの実行ステップが与えられた領域における「制約の可解性の決定」に依存している。これは，通常の単化手続きが，（エルブラン領域における）等式の可解性を決定し，可能であれば最汎単化子を求めるというのと酷似している。等式というのは，制約の一種と考えられるので，CLP の操作的モデルは，単化に基づいた（LPの）モデルの拡張になっている。

また，「制約の可解性の決定」という問題の設定は，基準(2)の処理系が果たすべき役割を明確化するので，さまざまな領域における制約解消の研究成果を取り込んだ，効率の良い処理系の研究を促す。

筆者らは，この Jaffer らの考えに無条件にに与するものではないが，上にあげた考え方には十分な論拠を見つけることができる。本論文では，我々が独自に開発しつつあるCAL (Contrainte avec Logique) という CLP 言語の基礎，インプリメント，応用について，主として Jaffar らの基準に沿って概観する。

2. 多ソート代数上の CLP とその論理的解釈

まず，CLPの意味論を展開するための基礎概念を手短に述べる。これは，Jaffar らの定義[JaL 87]に沿ってはいるが，細かい点で様々な相違がある。

変数の集合Ｖ，関数記号の集合Ｆ，述語記号の集合Ｐ，制約記号の有限集合Ｃ，ソートの集合Ｓが与えられているものとする。各変数，関数記号には，ソートが１つ割り当てられている。また，各関数記号，述語記号，制約記号には，ソートの有限列（空列でもよい）が割り当てられていて，それをシグネチャと呼ぶ。変数 vが ソートS を持つことを v:s と書く。関数記号 f がシグネチャ s1,s2,...,sn ソート s を持つことをf:s1,s2,...,sn→sと書く。述語記号または制約記号pがシグネチャs1,s2,...,sn を持つことを p:s1,s2,...,sn と書く。

項とそのソートを次のように帰納的に定義する。

(1) ソートsの変数は，ソートsの項である。

(2) f が関数記号でf:s1,s2,...,sn→s であり，t1,t2,...,tn がそれぞれソートs1,s2,...,snの項ならば，f(t1,t2,...,tn) はソートsの項である。

また，素論理式，素制約を次のように定義する。

(3) p が述語記号でp:s1,s2,...,sn であり，t1,t2,...,tn がそれぞれソートs1,s2,...,snの項ならば，p(t1,t2,...,tn) は素論理式である。

(4) c が制約記号でc:s1,s2,...,sn であり，t1,t2,...,tn がそれぞれソートs1,s2,...,snの項ならば，c(t1,t2,...,tn) は素制約である。

項 t がソートs を持つことを t:s と書く。項，素論理式，素制約の全体を，それぞれ，T(F,V)，A(P,F,V)，A A(C,F,V)と書く。また，変数を持たない項，素論理式，素制約を，それぞれ定項，定素論理式，定素制約と呼び，それらのの全体を，それぞれ，T(F)，A(P,F)，A(C,F)と書く。制約とは，素制約の（空かもしれない）有限集合である。直観的には，制約とは素制約の論理積である。

各ソートsに対してシグネチャ s,s の制約記号＝ₛが有るものとする。（これに対しては間置記法を用い，特に必要がない限り添字 s も省略する）。

次を満足する集合族 {D(s)|s∈S}，関数族 {D(f)|f∈F}，関数族 {D(c)|c∈C}の組を構造と呼ぶ。これが，通常の LP における Herbrand 空間に対応する。

(1) f が関数記号で f:s1,s2,...,sn→s なら，D(f) は D(s1)×D(s2)×...×D(sn) から D(s) への関数である.

(2) c が制約記号で p:s1,s2,...,sn なら，D(c) は D(s1)×D(s2)×...×D(sn) から{0,1} への関数である。

いま，構造を固定して考える。D(=ₛ) は，D(s)×D(s) から {0,1} への関数であるが，

$$D(=_s)(x,y) = \text{if } (x=y) \text{ then } 1 \text{ else } 0$$

が，成り立つものとする。=ₛは，通常のLP における単化の役割を演ずる。

次を満足する関数族 {I(p)|p∈P} を解釈と呼ぶ。これが，通常の LP での Herbrand解釈に対応する。

(3) p が述語記号で p:s1,s2,...,sn なら，I(p) は D(s1)×D(s2)×...×D(sn) から{0,1} への関数である。

付値とは，次を満足するVから ∪D(s) への関数Θである。

(4) v:s ならば，vΘ∈D(s)
（慣例にしたがってΘには，後置記法を用いる。）

付値は簡単にT(F,V)上の関数に拡張でき，そのとき t:s ならば，tΘ∈D(s) である。同様に付値は，A(P,F,V)，A(C,F,V)から {0,1} への関数に拡張できる。

C を制約とする。ある付値Θがあって，全ての c∈C に対し

て $c\Theta=1$ のとき C は可解であるといい，Θのことを C の解という。

プログラム節（これは確定節の拡張である）とは，p :- p1, p2,...,pn （n=0 でも良い）という形の式である。ここで p は素論理式，各 pi は素制約か素論理式である。プログラム節の有限集合を（制約論理）プログラムという。

次に，プログラムPのモデルを定義する。今，解釈 {I(p)| p∈P}が与えられたものとする。これが，P のモデルであるとは，すべてのプログラム節 (p :- p1,p2,...,pn)∈Pとすべての付値Θに対して，$p1\Theta=p2\Theta=\ldots=pn\Theta=1$ ならば $p\Theta=1$ であることである。

3．プログラムの関数的解釈

次に示す定義は，Van Emden とKowalskiによって与えられたものに対応する[EmK 76]。プログラム P が与えられたとする。このとき，1つの解釈 {I(p)|p∈P} をもとにして別の解釈{J(p)|p∈P} を次のように定義する。

J(p)(d1,d2,...,dn) =
  if　プログラム節 p(t1,t2,...,tn) :- p1,p2,...,pm
  　と付値Θがあって$p1\Theta=p2\Theta=\ldots=pm\Theta=1$ かつ
  　$d1=t1\Theta$, $d2=t2\Theta$, ..., $dn=tn\Theta$
  then 1
  else 0

上で定義したJ をT(P,I) と書くことにすると，T(P,_) は，解釈から解釈への写像となる。解釈I が解釈J より小さい（I≦J と書く）ということを次のように定義する。すべての述語記号 p:s1,s2,...,sn とすべての元 d1∈D(s1), d2∈D(s2), ..., dn∈D(sn)に対して，I(p)(d1,d2,...,d2)=1ならば J(p)(d1,d2,...,d2)=1。このとき，次を証明するのは易しい。

<命題 3.1> 解釈の全体は，≦に関して完備束となり，T(P,_) はその上で連続である。つまり次が成立する。
(1) I≦J ならば，T(P,I)≦T(P,J)
(2) I1≦I2≦... ならば，sup T(P,Ii) = T(P,sup Ii)

T(P,_) を用いて，次のような解釈を自然に定義できる。ただし，αは順序数である。

$T\uparrow\alpha=$ if　αが successor ordinalで $\alpha=\beta+1$
  then $T(P,T\uparrow\beta)$
  else sup $\{T\uparrow\beta \mid \beta<\alpha\}$

$T\downarrow\alpha=$ if　αが successor ordinalで $\alpha=\beta+1$
  then $T(P,T\downarrow\beta)$
  else inf $\{T\downarrow\beta \mid \beta<\alpha\}$

$\alpha=0$ の場合にも else 以下の定義をそのまま当てはめる。$T\uparrow 0$ は≦に関する最小元になる。つまり，すべてのの述語記号p:s1,s2,...,snとすべての元d1∈D(s1),d2∈D(s2), ..., dn∈D(sn)に対して，$T\uparrow 0$ (p)(d1,d2,...,d2)=0 である。逆に$T\downarrow 0$は≦に関する最大元になる。つまり，すべての述語記号p:s1,s2,...,snとすべての元 d1∈D(s1), d2∈D(s2),..., dn∈D(sn)に対して，$T\downarrow 0$ (p)(d1,d2,...,dn)=1 である。

また，次を示すのは易しい。

$$T\uparrow 0 \leqq T\uparrow 1 \leqq T\uparrow 2 \leqq \ldots$$
$$T\downarrow 0 \geqq T\downarrow 1 \geqq T\downarrow 2 \geqq \ldots$$

命題3.1 の(1)と完備束上の順序順同型に関する不動点定理より，T(P,_) は，最小不動点と最大不動点を持つ。それぞれ，lfp(T,P), gfp(T,P) と書くと，ある十分大きな順序数αとβがあって $lfp(T,P)=T\uparrow\alpha$, $gfp(T,P)=T\downarrow\beta$ である。実は，命題3.1の(2) より，$lfp(T,P)=T\uparrow\omega$ であることが簡単に示される。最大不動点 gfp(T,P)は$T\downarrow\omega$とは一般には異なる。

<補題 3.2>
任意のプログラムPに対して次が成立する。

(1) I が P のモデルであるとき，かつそのときに限り，T(P,I)≦Iである。特に最大元$T\downarrow 0$ は≦に関して最大のPのモデルである。

(2) lfp(T,P)はモデルであり，任意のPのモデルIに対して lfp(T,P)≦Iである。このゆえにlfp(T,P)をPの最小モデルという。

ここで，この関数T(P,_) の構文的な対応物を定義する。素論理式 p と 可解な制約 Cの対を考える。便宜上この対をp:- C と書き，QA-対と呼ぶ。QA-対の全体をＱＡと考く。ＱＡの1つの部分集合 S をもとにして別の部分集合 T を次のように定義する。

T = { p(s1,s2,...sn) :- C |
  　プログラム節p(t1,t2,...,tn) :- p1,p2,...,pm が

あって，

(1) 各piについて piが素論理式で (pi :- Ci)∈S
　　または piが素制約で Ci＝{pi}，

(2) C＝{s1＝t1, s2=t2, ..., sn=tn}
　　∪C1∪C2∪...∪Cm

(3) C は可解　　}

上で定義したTをQ(P,S) と書くことにすると，Q(P,_) は，ＱＡの部分集合から部分集合への写像となる。このとき，Q(P,_) は集合の包含関係⊆に関して先とまったく同様の性質を持つ。

<命題 3.3>　Q(P,_) は集合の包含関係⊆に関して連続である。つまり次が成立する。

(1) S⊆T ならば，Q(P,S)⊆Q(P,T)

(2) S1⊆S2⊆... ならば，∪Q(P,Si) ＝ Q(P,∪Si)

Q↑α，Q↓αも同様に定義する。

Q↑α＝ if　αが successor ordinalで α＝β+1
　　then Q(P,Q↑β)
　　else ∪{Q↑β| β＜α}

Q↓α＝ if　αが successor ordinalで α＝β+1
　　then Q(P,Q↓β)
　　else ∩{Q↓β| β＜α}

Q↑0＝φ（空集合），Q↑0＝ＱＡ である。また，

Q↑0 ⊆ Q↑1 ⊆ Q↑2 ⊆ ...
Q↓0 ⊇ Q↓1 ⊇ Q↓2 ⊇ ...

Q(P,_) は，最小不動点lfp(Q,P) と最大不動点gfp(Q,P) を持つ。ある十分大きな順序数αとβがあって lfp(Q,P)＝Q↑α，gfp(Q,P)＝Q↓β である。実は，lfp(Q,P)＝Q↑ω であるが，gfp(Q,P) は Q↓ωとは一般には異なる。

S⊆ＱＡ に対して解釈 {I(p)|p∈P} を次のように定義する，

I(p)(d1,d2,...dn) ＝
　if　QA-対(p(t1,t2,...,tn) :- C)⊆Sと付値Θがあって
　　d1＝t1Θ，d2＝t2Θ，...，dn＝tnΘ かつ ΘはCの
　　解である
　then　1
　else　0

この I を [S] と書くことにする。

<補題 3.4> 任意のプログラム P と，任意の順序数αに対して次が成立する。

(1) T↑α＝[Q↑α]

(2) T↓α＝[Q↓α]

上の補題より，特にlfp(T,P)＝[lft(Q,P)]，gfp(T,P)＝[gft(Q,P)] である。

4. プログラムの操作的解釈

ここで，CLP の操作的モデルを定義する。p1,p2,...,pn;C という形の式をゴールという。各 pi は素制約か素論理式であり，C は可解な制約である。n＝0 でも良く，このときゴールは succesful であるという。Pをプログラムとするとき，（拡張された）SLD-導出とは，ゴール p1,p2,...,pn;C から次の形のゴールを得ることである。

(1) p1が素制約で D＝{p1}∪C が可解なら，ゴールp2,...,pn;D を得る。

(2) p1＝p(s1,s2,...,sm)が素論理式でプログラム節
　(p(t1,t2,...,tm) :-q1,q2,...,qk)∈P があり，
　D＝{s1＝t1, s2＝t2, ..., sm＝tm}∪C が可解なら，
　ゴール q1,q2,...,qk,p2,...,pn;D を得る。

ゴールの列 G0, G1, G2, ..., Gn が SLD-導出列であるとは，各Gi+1 が Gi から SLD-導出によって得られることである。n をこのSLD-導出列の長さという。SLD-導出列がsuccessfulであるとは，その最後の要素が successful であることである。successfulな SLD-導出の最後の要素の制約を解制約と呼ぶ。

ここで，success set S(P) を定義する。

S(P)＝{ (p :- C)∈ＱＡ |
　　ゴール p;φ からの successful なSLD-導出列で解制約 C を持つものが存在する。}

<定理 4.1>
任意のプログラムに対して lft(Q,P)＝S(P) である。

5. 制約解消と標準形

前節のCLP の操作的モデルに基づけば，（拡張された）SLD-導出により実際に計算を行っていくには，制約の可解性の判定が必要かつ十分である。しかし，解制約が得られてもプログラムに書かれた内容のままでは，計算結果として受け容れるには，多くの場合に不満が残る。例えば，通常の数の領域で {x+y=3, x-y=1} は可解であり，前節の定義でみる限り解制約として何の不都合もないが，実際に欲しいのは {x=2, y=1} という解である。この意味で「制約を解く」というのは，単なる可解性の判定というよりも，我々にとって好都合な結果を得るための手続きであることが望ましい。

２つの制約 C, D が同値であるとは，付値Θが C の解であるとき，かつそのときに限りD の解であることをいい，C～D と書く。例えば，{x+y=3, x-y=1}～{x=2, y=1} である。明らかに～は制約の上の同値関係を定義する。各同値類 Eに，代表元 E↓が定義されているとする。制約 C の属する同値類の [C] の代表元 [C]↓を C の標準形という。次のアルゴリズムを↓に関する制約解消系と呼ぶ。

(1) 任意の制約に対して，その可解性を判定する。
(2) 可解な制約に対して，その標準形を求める。

このような制約解消系が有るとき，前節の SLD-導出は，単に制約の和集合 D を作るのではなく，D の標準形を求めるという形にさらに改善される。実際，通常の LP における単化は，Herbrand 空間上の等式制約の標準形を求めることに他ならない。

6. CAL (Contrainte avec Logic)

CLP 言語の実例として CLP(R)と呼ばれる言語が，Monash大学で実現されている [JaM 85], [HJL 86]。clp(R) においては，線形の方程式と不等式を，制約として扱うことができる。代数的制約を扱うことのできる，もうひとつの重要な CLP言語が Colmerauerの Prolog III である [Col 87]。Prolog IIIにおいては，有理数上の線形の制約や，Boolean等の制約を扱うことができる。本節では，CAL (Contrainte avec Logique) と呼ぶ CLP 言語について述べる。CAL は，非線形方程式も扱うことができるところにその特長がある。

6.1 CAL の言語，領域

CAL の言語と領域とを２節に則って述べる。

S = {AN},
V = 大文字で始まる英字列,
F = {*, +} ∪ 有理数の全体,
P = 小文字で始まる英字列,
C = {=}

とする。ここでは，簡単のためソートは，AN (Algebraic Numberの意) 1つだけとするが，実際の CAL 処理系は，通常のProlog との互換性を保つために Herbrand空間のソートが入っている。ソートが1つしかない場合，各記号の持つソートは明らかだし，シグネチャはarity だけを示せばよいが，上の CAL言語についてはわざわざ述べるまでもあるまい。

また，構造を次のように定義する。

(1) D(AN) = 代数的数（整係数の整方程式の解になりうる複素数）の全体
(2) D(*) = 積，D(+) = 和，D(有理数) = その数自身

言語と領域から，制約として書けるものが多項方程式であることは，明らかであろう。

6.2 CALの制約解消系: Buchberger アルゴリズムと Groebner 基底

Buchbergerは，Groebner 基底の概念を導入し，与えられた多項式系の Groebner 基底を求めるアルゴリズムを示した [Buc 83] 。このアルゴリズムは，近年，とくに数式処理の分野において，非常に広い範囲にわたって応用されている。Groebner 基底は，５節で述べた制約の標準形としての要件をほとんど完全に満たしている。そこで CAL インタープリタは，Buchberger アルゴリズムを制約解消系として採用した。まず，Groebner 基底と Buchberger アルゴリズムの理論的背景について述べる。

一般性を失わずに，多項方程式がp=0という形をしていると仮定することができる。E = {p1=0,...,pn=0} を多項方程式系とする。I を {p1,...,pn}から生成される多項式環のイデアルとする。I の要素とE の解との間には，次のように深い関係がある。

<定理 6.1> Hibertの零点定理 [Hil 1890]
p を多項式とする。E のすべての解が p=0 の解でもあるときは，ある自然数 n に対して p^n が I の元であるとき，かつそのときに限る。

さらに，次の系は，制約の可解性を調べる上で，重要なもの

である。

〈系 6.2〉
Eは，1∈ Iであるとき，かつそのときに限り，解を持たない。

さて，これによって制約を解くという問題は，生成されたイデアルに属するか否かという問題に帰着されたことになる。Buchberger は，多項式がイデアルに属するかどうかを決定する，次のようなアルゴリズムを与えた。

いま，多項方程式系（制約）があるとする。その各方程式は，単項間のある順序が与えられたとき，その順序のもとでの最大の単項をそれ以外の多項式へと書き換えるような書換え規則であるとみなすことができる。たとえば，辞書式順序を仮定すると，多項方程式 Z-X+B=A は，書換え規則 Z→X-B+A とみなすことが可能である。２つの任意の書換え規則の左辺が互いに素ではない場合，これら２つの規則は重なりを持つという。そのような場合，それらの左辺の最小公倍数（LCM）は，それぞれの規則によって，２通りに書き換えられる。この２通りの結果を要対と呼ぶが，これらはさらに書き換えていっても，合流しない場合がある。すなわち，それらの既約形は異なるわけである。このような場合にこの対を方程式系に新たな方程式として付け加える。これを繰り返すことにより，合流性のある書換え系が得られるのである。この合流性を持った書換え系をもとの方程式系のGroebner 基底と呼ぶ。次の定理は，イデアルと Groebner 基底との関係を述べたものである。

〈定理 6.3〉 Buchberger
Rを方程式系 {p1=0,...,p2=0} の Groebner 基底，Iを {p1,...,pn} から生成されるイデアルとする。このとき，多項式pは，pがRによって0に書き換えられるとき，かつそのときに限りIに属する。

さらに，次の定理が，既約な Groebner 基底を制約の標準形と考えることの妥当性を保証してくれる。Groebner 基底が既約というのは，その中の２つの書き換え規則が互いに他を書き換えることがないということである。

〈定理 6.4〉
単項間のある順序が固定されているとする。E，Fを方程式系とする。このとき，E，Fから生成されるイデアルが等しければ，E，Fの既約な Groebner 基底は等しい。

定理6.1の解とイデアルの関係が完全でないため，残念ながら，既約な Groebner 基底は５節の制約解消系の要件を完全には満足していない。例えば，制約 {X=0} と {X*X=0}とは，全く同じ解を持つが，既約な Groebner 基底はそれぞれ {X→0} と {X*X→0}であり，異なる。しかしながら，この程度の傷はほとんど問題にならないであろう。

6.3 CAL プログラムの実行

CAL プログラムの実行の様子を，例を用いて示す。前節においても述べたように，CALの特徴は，制約が非線形である場合に最も顕著に現れる。次の例は，幾何学の定理を証明するものである。この定理とは，四辺形の各辺の中点を順に結んで得られる図形は平行四辺形であるというものである。プログラムは次のようになる。

```
mid(AX,AY,BX,BY,CX,CY) :-
        AX+CX=2*BX,
        AY+CY=2*BY.
para(AX,AY,BX,BY,CX,CY,DX,DY) :-
        (AX-BX)*(CY-DY) == (AY-BY)*(CX-DX).
```

これらの節は，それぞれ中点および平行について述べたものである。節midは，点 (BX,BY) が線分 (AX,AY)-(CX,CY)の中点であることを述べている。また，節paraは，線分 (AX,AY)-(BX,BY)と線分(CX,CY)-(DX,DY) が平行であるということを述べている。単化だけで様々なプログラムを自由に書くのは難しいので，Prolog で多くのメタ述語を利用するように，CLP でも，実際の応用プログラムを書くためには，論理的な枠組みに完全にはそぐわない述語を導入する必要がある。節paraの本体に現れる記号 == もその１つで，その右辺と左辺とがその時点であつめられられている制約のもとで等しいかどうかを判定する述語である。この式は，次の式の変形により，得られるものである。

```
(AY-BY)/(AX-BX) == (CY-DY)/(CX-DX)
```

この両辺はそれぞれ２つの線分の勾配を表わしている。上の問題をこのプログラムを用いて解くには，次のようなゴール列を評価すればよい。

```
?- mid(0,0,x4,y4,x1,y1),mid(x1,y1,x5,y5,x2,y2),
   mid(x2,y2,x6,y6,x3,0),mid(x3,0,x7,0,0,0),
   para(x4,y4,x5,y5,x7,0,x6,y6),
   para(x4,y4,x7,0,x5,y5,x6,y6).
```

上のゴールは，次のように評価される。

(1) ゴールmid(0,0,x4,y4,x1,y1)が，mid の節の頭部に単化され，制約 0+x1=2*x4, 0+y1=2*y4 が得られる。

(2) ゴールmid(x1,y1,x5,y5,x2,y2)が，mid の節の頭部に単化され，制約 x1+x2=2*x5, y1+y2=2*y5 が得られる。

(3) ゴールmid(x2,y2,x6,y6,x3,0) が，mid の節の頭部に単化され，制約 x2+x3=2*x6, y2+0=2*y6 が得られる。

(4) ゴールmid(x3,0,x7,0,0,0)が，mid の節の頭部に単化され，制約 x3+0=2*x7, 0+0=2*0 が得られる。

(5) この時点で集められた制約は，以下の通りである。ただし，これらは標準化されている。

```
x1=2*x4,
y1=2*y4,
x1+x2=2*x5,
y1+y2=2*y5,
x2+x3=2*x6,
y2=2*y6,
x3=2*x7
```

(6) ゴールpara(x4,y4,x5,y5,x7,0,x6,y6)が，paraの節の頭部に単化される。ここで，集められた制約のもとで等式(x4-x5)*(0-y6) == (y4-y5)*(x7-x6)がチェックされる。この場合，両辺共，現時点での Groebner 基底によって，x2*y2に書き換えられるので，これらは制約のもとで等しい。

(7) ゴールpara(x4,y4,x7,0,x5,y5,x6,y6)が，paraの節の頭部に単化される。ここで，集められた制約のもとで等式(x4-x7)*(y5-y6) == (y4-y5)*(x7-x6)がチェックされる。この場合，両辺共，現時点での Groebner基底によって，y1*(x1-x3)に書き換えられるので，これらは制約のもとで等しい。

CAL プログラムの実行に際して，制約に出会うたびに制約評価系が呼び出される。新しい制約が既に得られている制約に対して矛盾する場合には，実行は失敗し，バックトラックが発生する。

上の例は， Groebner 基底を述語 == を通じて間接的に利用するものであったが，次のものは Groebner 基底を直接的に利用するような例である。

```
horse&man(Horses,Men,Horses+Men,2*Men+4*Horses).
```

このプログラムは，「馬と人間」問題を解くためのものである。

```
horse&man(H,M,5,14)
```

をゴールとする。すると，本体の制約の Groebner 基底が求められ，{Horses=3, Man=2}が得られ，表示される。

7. Boole 式のためのCAL

上の節で述べたCAL は，代数的数の領域における多項方程式のためのものであった。我々はまた，CAL の別バージョンの実現を行なった。これは，Boole 式を制約として記述できるものであり，その典型的な領域は，真偽値の集合である。その制約解消系では，Boolean Groebner 基底を求めるための，もとの Buchbergerアルゴリズムとは少し異なったアルゴリズムを採用している[SaS 88]。

CAL のこのバージョンにおいては，論理的評価を必要とするプログラムを容易に記述することができる。たとえば，論理回路の検証を行なうプログラムを書くことができる。

7.1 Boolean CAL の言語，領域

Boolean CAL の言語と領域とを２節に則って述べる。

S = {Boole},
V = 大文字で始まる英字列,
F = {*,+,0,1}
P = 小文字で始まる英字列,
C = {=}

また，構造を次のように定義する。

(1) D(Boole) = 任意のBoole代数
(2) D(*) = 論理積，D(+) = 排他的論理和，D(0) = 偽，
D(1) = 真

他の論理演算，例えば，論理和，含意，否定なども上の言語で記述できることは，明らかである。実際のシステムは，F の中に始めからそれらを含んでいるが，本論文では，複雑になるのを避けた。

7.2 Boolean Groebner 基底

Boole 方程式の可解性の判定法は多く知られている。その中で典型的なものは，意味単化(semantic unificatioh) によるものであろう。例えば Dincbus は，それを制約解消系として採用している [Din 87]。

ところが，Boole 代数に対しても，一般の多項式と同様に Groebner 基底によるアプローチが可能である。このアプローチは，意味単化によるものより次の点で優れていると我々は考えている。

(1) プログラムまたはゴールに書かれた以外の変数を制約解消系が勝手に導入するということがないので，出力として，得られる解制約が分りやすい。
(2) 5節の意味での制約の標準形が存在し，その定義も明確である。

Boolean Groebner 基底の存在，それを求めるアルゴリズムは，文献 [SaS 88] に譲り，ここでは幾つかの重要な性質をあげるにとどめる。

EをBoole方程式系，p を Boole 多項式とする。IをEから生成されるイデアル，RをEのBoolean Groebner基底とする。

<定理 7.1> 零点定理
Eのすべての解が p=0 の解でもあるのは，p がIの元であるとき，かつそのときに限る。

<系 7.2>
Eは，1∈ Iであるとき，かつそのときに限り，解を持たない。

<定理 7.3>
p は，Rによって 0 に書き換えられるとき，かつそのときに限りIに属する。

<定理 7.4>
単項間のある順序が固定されているとする。E，FをBoole方程式系とする。このとき，E，Fから生成されるイデアルが等しければ，E，Fの既約な Boolean Groebner 基底は等しい。

通常の多項式の場合と異なり，解とイデアルの関係が完全なので，既約な Boolean Groebner 基底は5節の制約解消系の要件を完全に満足する。

8. 拡張の構想

現在，CAL の機能を拡張する計画がいくつかある。CAL の現バージョンにおいては，制約中の各変数の（仮想的な）値は代数的数，すなわち整係数多項方程式の解になりうる複素数である。しかしながら，ある変数が実数しか取りえないことが分かっているとすれば，x^2+1=0 のような制約があれば，矛盾がただちに結論付けられる。したがって，小さな領域についての知識を持った，強力な制約解消系があれば，計算時間は劇的に減少可能である。あるいは，sin(X)=1とか，e^x =π といった，代数的でない制約を書きたいこともあるかもしれない。そのような場合には，領域を，すべての複素数からなる集合へと拡張する必要がある。

制約を書き，あるいは解くにあたっては，この種のさまざまな要求が考えられる。このような要求を満たすためには，制約解消系が完全に利用者によって自由に変更可能となっていなければならない。この方針に従って，システムは，利用者自身の目的のために，制約解消系を再定義できるように，設計されいる。ただ，利用者に要求されるのは，2節で述べたように言語と意味領域を明確化することと，5節で述べたような制約解消系を用意することである。（あるいは少なくとも制約の可解性の判定アルゴリズムをインプリメントすることである。）

［参考文献］

[ApE 82] Apt, K. R., and van Emden, M. H., "Contributions to the Theoryof Logic Programming," JACM, 29, 3, Jul. 1982, 841-862.

[Buc 83] Buchberger, B., "Groebner Bases: an Algorithmic method in Polynomial Ideal Theory," Thechnical Report, CAMP-LINTS, 1983.

[Col 82] Colmerauer, A., "PROLOG II - Reference Manual and Theoretical Model," Internal Report," Groupe Intelligence Artificialle, Universite Aix-Marseille II, Oct. 1982.

[Col 87] Colmerauer, A., "Opening the Prolog III Universe," BYTE, August, 1987.

[Din 87] Dincbas, M., Simonis, H., and van Hentenryck, P., "Extending Equation Solving and Constraint Handling in Logic Programming," ECRC Internal Reoprt IR-LP-2203, Feb. 1987.

[EmK 76] van Emden, M. H. and Kowalski, R. A., "The Semantics of Predicate Logic as a Programming Language," JACM, 23, 4, Oct. 1976, 733-742.

[Fik 70] Fikes, R. E., "REF-ARF: A System for Solving Problems stated as Procedures," Artificial Intelligence, 1, 1970, 27-120.

[Hil 1890] Hilbert, D., "Ueber die Theorie der algebraischen Formen," Math. Ann, 36, 1890, 473-534.

[HJL 86] Heintze, N., Jaffar, J., Lim, C.S., Michaylov, S., Stuckey, P., Yap, R., and Yee, C.N., "The CLP Programmer's Manual, Version 1.0," Department of Computer Science, Monash University, 1986.

[JaL 86] Jaffar, J. and Lassez, J-L., "Constraint Logic Programming," IBM Watson RC Internal Memo, 1986.

[JaM 85] Jaffar, J. and Michaylov, S., "Methodology and Implementation of a Constraint Logic Programming System," TR 54, Department of Computer Science, Monash University, June 1985.

[Llo 84] Lloyd, J. W., "Foundations of Logic Programming," Springer-Verlag, 1984.

[SaS 88] Sato, Y. and Sakai, K. "Boolean Groebner Base", LA-シンポジウム, Feb. 1988.

[Ste 81] Stefik, M., "Planning with Constraints," Artificial Intelligence, 16, 2, 1981.

[StS 78] Steele, G.L. and Sussman, G.J., "Constraints," MIT AI Lab Memo 502, Cambridge, Massachusetts, 1978.