特　集「第五世代コンピュータ」

制約論理プログラミング言語 CAL について

Constraint Logic Programming Language CAL

相場　亮*
Akira Aiba

* (財)新世代コンピュータ技術開発機構
Institute for New Generation Computer Technology.

1989年2月10日　受理

Keywords: constraint logic programming, Buchberger algorithm, Gröbner base, Boolean Gröbner base.

1. はじめに

制約論理プログラミング言語 (constraint logic programming language) は，論理型プログラミング言語を拡張したパラダイムであり，Prolog と同様 Colmerauer によって提唱されたものである[3]．Prolog に代表される論理型言語が単一化 (unification) によって計算の行われる言語であるのに対して，制約論理プログラミング言語は，単一化を制約 (constraint) というより広い概念で置き換えることによって，その記述力をさらに高めようとするものである．

また，Jaffar と Lassez は CLP (X) と呼ばれる制約論理プログラミング言語のスキーマを提唱した[7]．彼らはこの提唱にあたって次のような三つの基準をあげた．

(1) 複雑な問題をできるだけ簡単かつ自然に記述できること．
(2) このようにして作られた「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こと．
(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が存在すること．

基準(1)の答えとして，彼らは，問題を記述するために「制約」を採用した．制約という考え方自体は新しいものではない[9]が，これを論理型言語の枠組の上に展開している点は新しいものである．このことによる利点は多い．論理型言語の特徴として問題を宣言的に記述できるということがあったが，これは制約という形で問題を記述しようとする場合より強い要請となる．また，そのためにこそ基準(3)が重要なものとなる．問題を解決するために細かい制御情報を必要とするような言語では，(3)は期待できない．論理型言語を利用したことにより，さまざまな制御構造を導入することなしにゴール・リダクションの一般化として操作モデルが定義される．

制約論理プログラミング言語では，プログラムの実行ステップは与えられた領域における「制約の可解性」に依存している．これは，通常の単一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，可解であれば最汎単一化子 (mgu) を求めるという状況と酷似している．実際，等式は制約の一種であると考えられるので，制約論理プログラミング言語の操作モデルは単一化に基づいた (論理型言語の) 操作モデルの拡張となっている．

本稿では，ICOT において現在研究・開発の行われている制約論理プログラミング言語 CAL (Contrainte Avec Logique) について概説する．意味論その他の詳細に関しては我々の文献 (1) を参照されたい．

2. CAL

現在までにいくつかの制約論理プログラミング言語が提唱されている[3]-[5][7]が，これらは主として制約の記述領域によって特徴づけられる．例えば，Dincbas の CHIP[5] は実数上の線形等式，線形不等式，ブール代数上の等式，有限領域上の等式，不等式，非等式などを制約として扱うことができる．

CAL では，扱うことのできる制約を評価する機構 (以降，制約評価系と呼ぶ) をこれらのように言語自体に固定的に組み込むのではなく，制約評価系を論理

型言語と組み合せるときのための枠組を提供することを目標としている．このため，現在のCALはいくつかの種類の制約を扱うことができるが，これは固定的なものではなく将来的には拡張可能な構成をとっている．

現在CALにおいては次の2種類の制約を扱うことができる．

(1) 複素数上の代数等式（線形には限らない）

(2) ブール代数上の等式

以下に，これらについて少し詳しく述べる．

3．CALの制約

3・1 制約の評価

制約論理プログラミング言語の操作的モデルにおいては（拡張された），SLD導出によって計算を行っていく際に制約の可解性の判定が必要かつ十分である．しかしながら，実際問題としてプログラムの実行結果としてプログラムに書かれたままの制約が出力されるのでは不満である．このような意味で「制約を解く」というのは単なる可解性の判定だけではなく，利用者にとってより有益な結果を得るための手続きであることが望ましい．

今，C，Dを二つの制約とするとき，これらが同値であるとは，ある付値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，$C\sim D$と書くことにする．このとき明らかに$\sim$は制約の上の同値関係を定義する．各同値類Eに代表元$E\downarrow$が定義されているとする．制約Cの属する同値類$[C]$の代表元$[C]\downarrow$をCの標準形と呼ぶ．このとき次のアルゴリズムを$\downarrow$に関する制約評価系と呼ぶ．

(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

このような制約評価系があるとき，SLD導出は単に制約の和集合を求めるだけでなく，その標準形を求めることになる．実際，通常の論理型言語における単一化は，エルブラン空間上の等式制約の標準形を求めることにほかならないのである．

3・2 制約としての代数等式とその評価

前節にも述べたように，CALにおいては代数等式を制約として扱うことができる．この「代数等式」とは，有理数を係数に持つ，複素数上の任意次数の代数的等式のことである．我々はこのような制約の標準形を求めるためのアルゴリズムとして，ブフバーガ・アルゴリズム（Buchberger Algorithm）[2]を採用した．

ブフバーガ・アルゴリズムは，リンツ大学のBuchbergerによって提唱されたアルゴリズムであって，近年，数式処理，図形定理証明などに応用されている．Buchbergerは，多項式系のグレブナ基底（Gröbner Base）の概念を導入し，それを求めるアルゴリズムを示した．グレブナ基底は前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おり，CALにおいてはこれを制約評価アルゴリズムとして利用している．

制約を評価する（方程式を解く）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るが，Buchbergerは，ある多項式がイデアルに属するかどうかを決定するアルゴリズムを与えた．以下にアルゴリズムを示すが，詳細については文献(2)を参照されたい．このアルゴリズムによって等式の集合Eのグレブナ基底が，書換え規則の形でRに求まる．

① $R\leftarrow\phi$

② E中の等式$l=r$のすべてについて，$l-r$をR中の書換え規則と算術計算によって簡略化し，式eを得る．$e\equiv 0$であればこの式を捨て，さもなければE中の等式$l=r$を$e=0$で置き換える．

③ $E=\phi$であれば終了．

④ E中から等式$e=0$を選択する．

⑤ E中の最大の単項l'とする．$e=0$をl'について解き，式$l'=r'$を得る．

⑥ 書換え規則$l'\rightarrow r'$をRに付け加える．

⑦ R中の発散する要対を等式としてEに付け加える．

⑧ ②へ行く．

3・3 制約としてのブール等式とその評価

この節では，ブール等式を制約として扱うために，CALに導入されたアルゴリズムについて述べる[8]．

ブール方程式の可解性の判定法は多く知られているが，その中の代表的なものは意味ユニフィケーション（semantic unification）によるものであろう[5]．一方，CALにおいて採用した方法は一般の多項式の場合と同様にグレブナ基底によるものである．このアプローチには次のような利点がある．

(1) ゴールに書かれた変数以外の変数を導入することがないので，出力として得られる評価結果がわかりやすい．

(2) 制約の標準形が存在し，その意味も明確である．

CALにおいては与えられた制約をブール多項式に翻訳し，それのブーリアン・グレブナ基底（Boolean Gröbner Bases）[8]を求めることで制約の評価を行っている．

ブール多項式が通常の代数多項式と著しく異なる点の一つは，制約評価がイデアルによって完全に記述できるということである．

ブーリアン・グレブナ基底を求めるアルゴリズムは，複素数多項式環の場合とほぼ同様であるが，要対を等式として等式集合に付加する際，自己要対も求めて付加する点が異なっている．詳細については文献(8)を参照されたい．

4．CALプログラム

この節においてはプログラム例を示す．

4・1　代数等式制約を利用したプログラム例

この例はCALで三角形の三辺の長さとその面積との関係を求めたものである．

任意の三角形について，その底辺の長さをl，高さをh，面積をsとすると，これらの間には$l \times h = s$という関係がある．これをsurという述語名で記述する(下記の節①)．

次にピタゴラスの定理をrightという述語名で記述する．x, yを直角を挟む2辺とし，zを斜辺とするとき，$x^2+y^2=z^2$である(下記の節②)．ただし，プログラム中ではx^2を$x\hat{}2$と表記している．

さらに，任意の三角形が二つの直角三角形に分割されること(図1参照)をtriという述語名で記述する(下記の節③)．

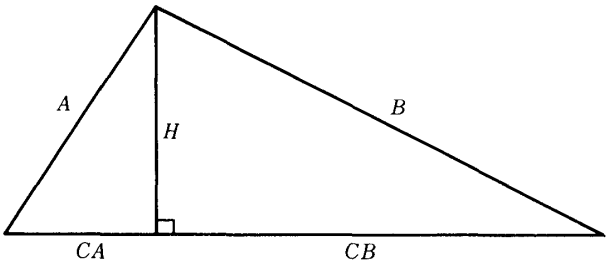

図1　ヘロンの公式

これらの節本体における制約(等式)の後ろにある「:alg」は，これらの制約が代数的なものでありブフバーガ・アルゴリズムで扱われるべきものであることを指定している．

```
sur (L,H,S)   :- L*H=2*S:alg.          ①
right (X,Y,Z):- X^2+Y^2=Z^2:alg        ②
tri (A,B,C,S) :- C=CA+CB:alg,
                 right (CA, H, A),
                 right (CB, H, B),
                 sur (C, H, S).        ③
```

このプログラムに対し，すべての引数がフリーであるようなゴールtri (a,b,c,s)を評価すると，次のようなa, b, c, sのみからなるような式が得られるが，これはヘロンの公式の展開形にほかならない．

```
s^2= (-c^4 +-1*a^4 +2*(2*b^2*a^2)+
      -1*b^4 +2*(2*c^2*a^2)+-1*c^4 +
      2*(2*c^2*b^2))/16
```

むろんこのプログラムは，確定した値を引数として評価することもでき，例えばゴールtri (3,4,5,s)を与えれば$s\hat{}2=36$が得られる．

4・2　ブール等式制約を利用したプログラム例

ブール等式制約を利用したプログラムとして簡単な論理回路の検証を行ってみる．次のプログラムは，図2の回路をブール等式を使って直接的に記述したもの

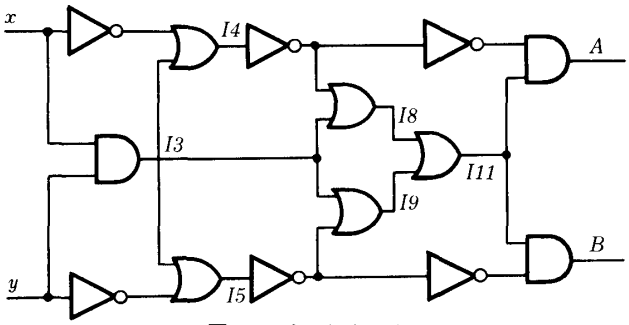

図2　平面交差回路

である．プログラム中，制約の後ろにある「:bool」は前節の「:alg」と同様，これらの制約がブール等式を扱うためのアルゴリズムによって扱われることを指定している．

```
circuit (X,Y,A,B):-
        I4=~X∨I3:bool,   I3=X∧Y:bool,
        I5=~Y∨I3:bool,   I8=~I4∨I3:bool,
        I9=~I5∧I3:bool,  A=I4∧I11:bool,
        I11=I8∨I9:bool,  B=I5∧I11:bool.
```

このプログラムに対してすべての引数をフリーにした次のようなゴールの評価を行う．

```
circuit (x,y,a,b).
```

すると次のような結果が得られる．

$x = b$

$y = a$

この結果，この回路が平面交差回路であることが記号的に検証できたわけである．

5．おわりに

CALは現在開発中の言語であり，最終的な姿については現時点では軽々しく判断することはできない．現状のCALは次の三つの方向に向けての研究を考慮

している.

(1) 新しい制約評価系の実装

例えば，線形不等式制約のための制約評価系や限量子除去アルゴリズム (Quantifier Elimination Algorithm) の部分アルゴリズムの実装など.

(2) 応用からのフィードバック

幾何学の定理証明系を CAL で記述する計画がある．また，効率向上を目指して制約評価系を階層化することも考えられる．さらに，複数の制約評価系間の関係（例えば，複素数領域における制約評価系と実数領域における制約評価系との間の包含関係など）の整合性をとることも今後の課題である.

(3) 並　列　化

我々の周辺には，並列言語提唱・開発の経験として GHC がある．CAL を並列化する際には，論理型言語を並列化して GHC が得られたときと同様の課題と制約言語特有の課題とがあると考える．前者は例えば，and 並列，or 並列，ストリーム並列といったような選択や実装上の問題などがあり，後者としては制約言語と探索の間の関係の明確化などがある．いずれにしてもこの研究はまだ始まったばかりであり，今後多くの応用を通じて知見を得た上で決定をしていく予定である.

制約論理プログラミングにおいては，引数間の関係を結果として出力することがよくある．特にゴール中の多くの引数がフリーのまま残されることがある．これは部分評価 [10] の効果と非常に似たものがあり，さらに unfolding の技法 [11] にも近いものである．これらとの関連についても今後の課題である.

謝　　辞

本稿の作成にあたり，この研究の共同研究者である ICOT の坂井公氏，佐藤洋祐氏，D. J. Hawley 氏，本稿作成の機会を与えていただいた古川康一研究担当次長，および本研究の機会を与えていただいた渕一博所長，長谷川隆三第 1 研究室長に謝意を表する.

◇ 参　考　文　献 ◇

(1) Aiba, A., Sakai, K., Sato, Y., Hawley, D. J. and Hasegawa, R. : Constraint Logic Programming Language CAL, Proc. of the FGCS '88, Vol. 1, pp. 263-276 (1988).

(2) Buchberger,B. : Gröbner Bases ; An Algebraic method in Polynomial Ideal Theory, Technical Report, CAMP-LINZ (1983).

(3) Colmerauer, A. : PROLOG-Ⅱ ; Reference Manual and Theoretical Model, Internal Report, Groupe Intelligence Artificille, Universite Aix-Marseille Ⅱ, Oct. (1982).

(4) Colmerauer, A. : Introduction to Prolog-Ⅲ, ESPRIT '87, Proc. of the 4th Annual ESPRIT Conference, North-Holland (1987).

(5) Dincbas, M., Van Hentenryck, P., Simonis, H., Aggoun, A., Graf, T. and Berthier, F. : The Constraint Logic Programming Language CHIP, Proc. of the FGCS '88, Vol. 2, pp. 693-702 (1988).

(6) Hilbert, D. : Über die Theorie der algebraischen Formen, *Math. Ann.*, Vol. 36, pp. 473-534 (1890).

(7) Jaffar, J. and Lassez, J-L. : Constraint Logic Programming, IBM T. J. Watson R. C., Internal Memo (1986).

(8) Sato, Y. and Sakai, K. : Boolean Gröbner Bases, LA-Symposium (Feb. 1988).

(9) Steele, G. L. and Sussman, G. J. : Constraints, MIT AI Lab. Memo 502, Cambridge, Massachusetts (1978).

(10) Takeuchi, A. and Furukawa, K. : Partial evaluation of Prolog Programs and Its Application to Meta-Programming, Information Processing 86, pp. 415-420, North-Holland (1986).

(11) Tamaki, H. and Sato, T. : Unfold/Fold transformation of Logic Programs, 2nd International Logic Programming Conference (1984).

著　者　紹　介

相場　亮

1986 年慶應義塾大学大学院数理工学専攻博士課程修了．工学博士．同年，日本電気（株）入社．C&C システム研究所に勤務．1987 年より（財）新世代コンピュータ技術開発機構に出向中．制約論理プログラミング言語の研究に従事．情報処理学会，日本ソフトウェア科学会各会員.