制約付き論理型プログラミング言語の構想と実例

4Q− 3

相場 亮　坂井 公

新世代コンピュータ技術開発機構

1.はじめに

J.Jaffar等は、制約付き論理型言語CLP(X)という言語スキーマを提案した[1]。このスキーマは、より柔軟な論理型言語の枠組みを与える。我々は、これに基づき、言語CALを試作した。CALは、制約として、任意の多項式からなる方程式を記述することが可能である。制約解消系としては、近年数式処理の分野において利用されつつあるBuchbergerのアルゴリズム[2]を採用した。我々はさらに、複数の制約解消系が共存するような言語を設計中である。制約解消系の使い分けはタイプの導入によって行う。本報告においては、試作言語であるCALと、我々の構想について述べる。

2.制約付き論理型言語CAL

制約という用語は、さまざまな文脈で用いられ、それに応じて微妙に意味が異なっている。CALにおける制約とは、代数的な等式である。この処理は構文上の単化(unification)のみで行うことは出来ない。

我々が試作を行った言語CALは、制約として任意の多項式からなる方程式を制約として記述出来るものであり、この解消のために、Gröbner基底を求めるためのBuchbergerアルゴリズムを採用している。これを採用することにより、制約をインクリメンタルに解くことが可能となり、特に制約が矛盾する場合の検出が早い段階で可能となっている。また、方程式の本数が未知変数の数と比較して少ない場合にも、これらの未知変数間の関係式が制約解消の結果得られると言う利点を持っている。また、不等式の形で与えられた制約の記述を可能とし、それを受動的制約[3] として扱えるような拡張は容易である。

3.Gröbner基底とBuchbergerアルゴリズム

任意の多項式からなる方程式(以下、単に等式と書く)を、ある順序の元での最大の単項(これを頭項と呼ぶ)について解き、これを書換え規則とみなす。このとき、各書換え規則は、その左辺の最小公倍数がそれらの積とは異なる場合、重なりを持つという。この場合、頭項の最小公倍数は2通りに書き換えられ、その結果は一般には異なる。2通りの書き換えの結果得られた2つの多項式を要対(critical pair)と呼ぶ。要対の生成を繰り返すことにより、与えられた任意の方程式系は、合流性のある書換え系へと変換される。この変換のためのアルゴリズムがBuchbergerアルゴリズムであり、その結果得られた合流性のある書換えシステムがGröbner基底である。以下、このアルゴリズムを示す。

Eを与えられた等式の集合、Rを書換え規則の集合とする。次のアルゴリズムにより、EのGröbner基底が、書換え規則の形でRに求まる。

(1) R←φ
(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演算とによって単純化して、式eを得る。e=0であれば、この等式を捨てる。そうでなければ、E中の等式l=rをe=0で置き換える。
(3) E=φであれば、終了。
(4) E中から等式e=0を選ぶ。
(5) e中の、ある順序に従った場合の頭項をl'とする。e=0をl'について解き、式l'=r'を得る。
(6) 規則l'→r'をRに加える。
(7) R中の規則の要対を等式としてEに付け加える。
(8) (2)へ行く。

4.CALプログラムの評価

以下に例を用いて、CALプログラムの動作を示す。CALにおいてはBuchbergerアルゴリズムを制約解消系として用いている関係上、非線形連立方程式の形の制約の解消において、最もその能力を発揮するが、ここでは簡単のため、線形の制約を例としてあげることにする。

```
trkm(C,T,C+T,2*C+4*T).        (1)
```

これは、いわゆる鶴亀算を行うためのCALプログラムである。Cが鶴の数、Tが亀の数、C+Tが頭の数、2*C+4*Tが足の本数である。ここで、(1)はCALシステムの前処理部において、次のように展開される。

```
trkm(C,T,H,P) ←
        H=C+T,P=2*C+4*T.    (2)
```

これに対して、次を入力したとする。

```
←trkm(C,T,5,14).            (3)
```

1) まず(3)が(2)の頭部に単化され、H=5、P=14となる。
2) 制約5=C+T、14=2*C+4*Tが制約解消系に与えられる。
3) Buchbergerアルゴリズムにより、これらのGröbner基底C=3、T=2が得られる。

また、入力が次であったとする。

```
←trkm(X+1,X,Y,14).          (4)
```

An Instance of Constraint Logic Programming and its Future Extensions
Akira AIBA, Kô SAKAI
Institute for New Generation Computer Technology

1)(4)が(2)の頭部に単化され、制約(X+1)+X=Y,2*(X+1)+4*X=14が得られる。
2) Buchbergerアルゴリズムにより、これらのGröbner基底X=2、Y=5が得られる。

この例の場合、制約が解消されないまま凍結されるという事態は起きないが、一般には未解消の制約は、実際には凍結されるのではなく、常にその時点のGröbner基底が求められながら、後ろ送りされる。最終的に、<変数>=<数値>の形の制約へと変形された時点で、初めてこの<変数>と<数値>とは単化される。したがって、このような形に変形されなかった制約は、CALプログラム評価終了時に出力される。これは、当然Gröbner基底である。例えば、上記の鶴亀算のプログラムに対して、次を入力したとする。

←trkm(c,b,a,14).

この場合、制約は完全には解消されず、次が出力される。

b=−a+7
c=2a−7

5.CALインタープリタの構成

CALインタプリタは、前処理部、推論エンジン、制約解消系からなる。これらは、次図のように、関連している。

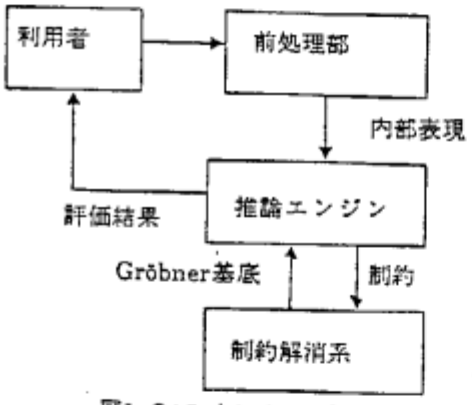

図1: CALインタープリタの構成

6.拡張の構想

本稿では、制約に現れる変数は、任意の代数的数を値としてとれると考えている。しかし、もし変数が有理数以外の値をとらないことがわかっているとしたら、$x^2=2y^2$からx=y=0という結論が導けるし、変数が虚数値をとらないとすると、$x^2+1=0$からは矛盾が導かれる。逆に任意の(代数的数とは限らない)実数値をとる変数xを考えて、sin(x)=1というような制約を書きたい場合もあろう。このように、変数の領域を広くしたり狭くしたりすると、一般には制約解消系も変える必要がある(その制約解消系がつくれるかどうかは別の問題である)。

上で見たように、CALにおいては、制約解消系は、その他の部分とは独立に機能しうる。必要なことは、推論エンジンとのインターフェスがとられているという点だけである。この性質を利用して、利用者定義の制約解消系を任意に導入出来るようにしようというのが、我々の第一の拡張構想である。

現在実現を考慮している別の制約解消系の1つは、ブール代数式のためのものであり、デジタル回路の検証等への応用を想定している。また、完備な項書換え系の存在を仮定し、その下での等式を制約として記述出来るようなシステムの実現も考慮中である。

また、1つのプログラムの中にブール値を表すbという変数と代数的数を表すxという変数とを導入し、それぞれについて制約を記述したい場合もあろう。この場合、問題はどの制約をどの制約解消系を用いて解消するかということである。我々は、この目的のためにタイプの導入を行うことを考えている。例えば、利用者がpという述語のタイプをbool×algebraicであると定義したとしよう。このとき、p(¬X,Y+1)←... は、前処理部において、

p(B,A)←B=¬X:bool, A=Y+1:algebraic, ...

と展開されるのである。等式の後ろの:bool、:algebraicは、その制約を解くのに利用術き制約解消系を示している。つまり、利用者がbool、algebraicの制約解消系としてそれぞれブール式制約解消系、Buchbergerアルゴリズムを作成しておけば、それらが呼び出されるという仕組みである。これが第二の拡張構想である。

もっと複雑な状況も考えられる。例えば、Xはある代数的数、Aは代数的数からなる行列を表すとする。この場合、XAに関する制約の解消を考えると、それがXの制約解消と全く独立に行われうるとは考えにくい。このような場合、異なる制約解消系間のインターフェースが必要になるかもしれない。これをエレガントに解決することも重要な課題であろう。これについては、我々は具体的な案を持っていない。しかし、これはあくまでも制約解消系間の問題であり、将来拡張があってもCALの前処理部と推論エンジンは大きな変更を受けないように工夫出来るであろうと考えている。

[参考文献]

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

[2] Buchberger, B., "Gröbner Bases: An Algebraic Method in Polynomial Ideal Theory", CAMP-LINZ Internal Memo. 83-29.0, Nov. 1983.

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