C1-1

制約指向 Committed-Choice 型言語 Fleng/C の構想
A Plan of Constraint Oriented Committed-Choice Language Fleng/C

小島 浩 Hiroshi Kojima
小池 汎平 Hanpei Koike
田中 英彦 Hidehiko Tanaka

東京大学工学部
Faculty of Engineering, Tokyo University

本稿では、Committed-Choice 型言語 Fleng に対して、制約指向の概念を取り入れた言語、Fleng/C の提案を行う。直接型と間接型の2種類の制約の表現形式を導入する。直接型の制約としては線形計画問題、ブール値上の方程式などを直接プログラムに表現できるようにする。また、間接型の制約としては集合を使って制約を扱うことや、求める解の個数についての指示を可能にする枠組を提案する。

1 はじめに

制約指向プログラミングのよい点は問題を宣言的に表現できることである。つまり、プログラムを書く人から見れば、解く方法を示さないでただ目標を示すだけで問題が解けてしまうのである。

本研究の目的は Committed-Choice 型言語 Fleng に対して、制約指向の概念を取り入れた言語 Fleng/C を設計することである。

Fleng/C では、さまざまな種類の制約を統一的な枠組で扱うことを目指している。また、Fleng/C は Fleng に対して完全な上位互換性を持たせた。制約の表現形式として大きく分けて、直接型と間接型という2種類を用意することにした。間接型は制約を集合の形で扱うためのものである。それでも、直接型と間接型はまったく独立のものではなく、集合を扱う上で役割分担をしている。

2 Fleng

ここでは拡張の元になった言語 Fleng[Nilsson89] について説明する。Fleng は Committed-Choice 型言語の範疇に属する言語である。Fleng のプログラムはヘッドとボディゴールでできた次のような節の集合である。

$$H : -G_1, G_2, \ldots, G_n.$$

Fleng のプログラムの実行サイクルは次のようなものである。

- プログラムの実行はゴールの集合 Q を与えることで開始される。すべてのゴールが独立に任意の順序に実行される。Q の中のあるゴール G が Q から取り除かれ、プログラム中のすべての節のヘッド部とマッチング (matching) される。最初にマッチングに成功したヘッド部を持つ節が選ばれて、その節のボディゴールが集合 Q に追加される。

- G の中の変数を具体化するようなマッチングはすべて中断される。そしてその変数がどこか別のところで具体化するとそのマッチングは再開される。

Fleng ではゴールの実行は論理的な真理値とは無関係に行なわれる。つまり、1つのゴールが失敗しても、他のゴールは影響を受けず、無関係に処理が進められる。同じ節のゴール間の論理的な関係が必要なときは、ゴール間の共有変数によってその関係を記述することができる。

3 Fleng/C

Fleng/C においてはおなじ節のボディ部に述語と制約を混在させることは許されない。

3.1 Fleng/C のプログラム

Fleng/C のプログラムは節 (clause) の集合である。Fleng と異なる点は節に2つの種類があることであ

る。一つは述語節 (predicate clause) であり、もう一つは制約節 (constraint clause) である。

- 述語節は Fleng における節と同じものであり、ボディ部には通常の Fleng の述語しか現れない。

$$h : -p_1, \ldots, p_n.$$

- 制約節には一つの種類の制約しか現れない。制約節は次の形をしている。

$$h : -constraint(c), c_1, \ldots, c_n.$$

ここで c はこの節に現れる制約の種類の名前であり、アトムである。constraint(c) は述語ではなく、ユーザが Fleng/C の処理系に対して与える情報である。

3.2 直接型の制約と間接型の制約

Fleng/C においては大きく分けて次の 2 通りの制約の表現方法が可能である。直接型と間接型である。

- 直接型の制約
  直接型の制約とはこれまで述べてきたような制約節を使って制約を表現する方法である。制約節のボディ部の中には述語 (Fleng の) や他の種類の制約を書くことは許されない。したがってこの節からユーザの書いた他の節を呼び出すことはあり得ない。述語節と制約節の間のデータの受け渡しはヘッドの単一化のみによって行なわれる。直接型の制約の表現は Fleng/C の処理系によって、Fleng のプログラムに書き換えられる。また、直接型の制約の表現法の中の一つ、集合による制約の表現は、次に説明する間接型の制約を評価することを処理系に指示するために用意されたものである。

- 間接型の制約
  この表現方法は制約を集合の形で表現するものである。集合の定義、集合から演算子をつかって他の集合を作ることなどが、述語節を使って表現される。間接型の制約は Fleng/C の処理系に利用される。

4 直接型の制約の種類

直接型の制約には次のようなものがある。

- 線形計画問題

  線形計画問題 (Lenear Programming Probrem) が次のような形で与えれるものとする。
  制約条件

$$a_{11} * X_1 + \ldots + a_{1n} * X_n \leq b_1$$
$$\ldots$$
$$a_{m1} * X_1 + \ldots + a_{mn} * X_n \leq b_m$$

  の下で線形形式

$$f(X_1, \ldots, X_n) := c_1 * X_1 + \ldots + c_n * X_n,$$

  が取り得る最大値を求めること。ただし、

  $c_1, \ldots, c_n$ は実定数
  $X_1, \ldots, X_n$ は実数変数

  である。

  制約を解くアルゴリズムとしては Sup-Inf 法 [Shostak77] を用いることを考えている。それは、Sup-Inf 法は小規模な不等式の問題についてはマトリックスを作る計算のオーバーヘッドがないためその分だけ速く、また、記号的計算であり、また、並列化がしやすいと考えられるからである [Kawagishi89]。

- AC matching 問題。AC matching を使って解けるものは AC matching を使って解く。AC matching 問題の例としては、次のようなものが考えられる。

```
p(X) :-
    (sin(1)*sin(1))+X*X=1.
```

  解は $X = cos(1)$ となればよい。ここではルールとして、$cos(A)*cos(A)+sin(A)*sin(A)-1 = 0$ を用意しておかなければならない。問題点としては AC matching だけて解ける問題は少ないと思われることである。

- ブール値上の方程式 (boolean equation)
  ブール値上の方程式は値域が真偽値 {0,1} であり、係数も真偽値である。ブール値上の方程式 (Boolean equations) を解くのに、制約を Buchberger アルゴリズムを用いて評価して、Boolean Grobner 基底 [SakaiSato89] を求める方法を考えている。

- 集合による制約
  これはある集合で表された制約を解くことを処理系に指示するものである。その際、求めるべき解の個数を指示することができる。この制約

の識別子は set である。有限個 (n 個とする) の解を求める場合には制約として

```
some(X,a,n)
```

を書き、すべての解を求めて欲しい場合には制約として、

```
all(X,a)
```

を書く。ここで a はユーザが間接的な表現の制約のところで定義した集合である。制約節全体の例として次のような節をあげることができる。

```
p(X) :-
    constraint(set),
    some(X,a,1).
```

これは集合 a で表現される制約を満たす解を一つ求めることを処理系に指示するものである。all の場合も some の場合も帰されるのは制約を満たす解のリストである (たとえ 1 つであっても)。

このように、求める解の個数をどうするかを (全部にするかどうかも含めて) ユーザか指示できる。これは、問題によっては解を 1 つ見つけてくるのが容易な問題でも、すべての解を見つけてくることができるとは限らないし、解の個数が有限とは限らず、無限にある場合があるからである [Sakai89]。

5 間接型の制約

間接的な制約とは基本的にはユーザが制約集合を作っていくことである。

5.1 必要な定義

次にここで必要な定義をしておく。基本ユニバースの集合 U_0、ユニバースの集合 U、2 項集合演算の集合 F_2、単項集合演算の集合 F_1、基本制約集合の集合 C_0、制約集合の集合 C は次のように定義される。

基本ユニバースの集合 U_0 は次のように定義される。

$$U_0 = \{\mathbf{Z_2}, \mathbf{N}, \mathbf{Z}, \mathbf{R}\}$$

ここで $0 \in \mathbf{N}$, $\mathbf{Z_2} = \{0,1\}$ である。U の各要素には名前が付けられていて、それぞれ boolean, natural, integer, real である。

ユニバースの集合 U は次のように定義される。

1. $U_0 \subset U$
2. $x \in U, y \in U$ ならば $x \times y \in U$ である。

2 項集合演算の集合 F_2 は次のように定義される。

$$F_2 = \{\text{union}, \text{intersection}, \text{difference}\}$$

単項集合演算の集合 F_1 は次のように定義される。

$$F_1 = \{\text{complement}\}$$

基本制約集合の集合 C_0 は次のように定義される。$X \in U$, $y \in X$, $c(y)$ が y に対する制約とすると、$\{y|c(y)\} \in C_0$ である。要するに基本制約集合はユニバースとそのユニバースの上の制約の組として表されるということである。ここでもちろん $C_0 \subset X$ である。

制約集合の集合 C は次のように定義される。

1. $C_0 \subset C$
2. $X, Y \in C$, $f \in T_2$ ならば $f(X,Y) \in T$ である。
3. $X \in T$, $f \in R$ ならば $f(X) \in T$ である。ただし $\exists Z \in U, X \subset Z, Y \subset Z$ でなければならない。要するに X と Y がある共通のユニバースの部分集合でなければならないということである。

5.2 Fleng/C のプログラムにおける間接型の制約

では次に実際の Fleng/C のプログラムの中で間接型な制約がどのように表されるかを示す。

間接的な制約はすべて述語節の形で表現される。ユーザがプログラム中で間接型の制約として書くことができるのは、ユニバースの定義、基本制約集合の定義、制約集合の定義である。

- ユニバースの定義これは基本ユニバースやユーザが定義したユニバースを用いて新しくユニバースを定義することである。

```
universe(u,U1,U2) :-
    U1 = u1,
    U2 = u2.
```

ここで u,u1,u2 はいずれもユニバースの名前である。ユニバースの名前は Fleng の普通のアトムである。この節の意味は u1 と u2 の直積を u とするということである。例として $\mathbf{R^3}$ を定義するための節の集まりは次のようになる。

```
universe(r2,X,X) :-
    X = real.
universe(r3,X,Y) :-
    X = r2,
    Y = real.
```

- 基本制約集合の定義

```
basicset(s,X,D,C) :-
    X = x,
    D = d,
    C = c.
```

s はユーザが付けた基本制約集合の名前である。x は s が属するユニバースの要素を変数のリストの形で表したものである。d は s が属するユニバースの名前である。c は d 上の制約のリストである。制約は x の中の変数の関係式として表される。s,d,c およびリスト x の各要素は Fleng のプログラムの中ではアトムである。(Fleng の変数とは区別しなければならない。) 具体的な例をあげると、たとえば次のようなものである。

```
basicset(a,X,D,C) :-
    X = [x,y,z],
    D = r3,
    C = [x+y+z=0].
```

これは $\mathbf{R}^3$における平面を表す制約の集合による表現である。

- 制約集合の定義
これは基本制約集合や制約集合を使って新しく制約集合を定義することである。2 項集合演算を使う場合は次のようになる。

```
set(c,F,A,B) :-
    F = f,
    A = a,
    B = b.
```

ここで a と b は別の節で定義された制約集合 (基本制約集合を含む) の名前であり、c はここで定義しようとしている制約集合の名前である。f は単項集合演算の名前である。

単項集合演算を使う場合は次のようになる。

```
set(b,F,A) :-
    F = f,
    A = a.
```

ここで a は別の節で定義された制約集合 (基本制約集合を含む) の名前であり、b はここで定義しようとしている制約集合の名前である。f は単項集合演算の名前である。

5.3 間接型の制約の利点と問題点

間接型の制約は完全な Fleng の節として通用するものであり、Fleng/C の処理系はそれを書き換える必要はない。制約の評価をするときには、自然な形で利用することができる。

また、間接型の制約が基本制約集合の定義と制約集合の定義に分離しているため、基本制約集合の中身を評価する前に集合として処理することができる。どう処理するかは Fleng/C の処理系の裁量に任されるが、例えば基本制約集合を最小の単位とした連言標準形に変換することなどが考えられる。

集合の内部表現として、その集合に存在することがわかっている要素や存在しないことがわかっているものをデータ構造の一部として持たせることにすると、この情報に対しても集合演算を適用することができる。

問題点としては複数の種類の制約が混在してしまうことである。この点では処理系の負担が重くなってしまう。

参考文献

[Nilsson89] Nilsson,M., *Parallel Logic Programming for SIMD Supercomputers and Massively Parallel Computers,* 東京大学 大学院 工学系研究科 博士論文 (1989)

[Kawagishi89] 川岸太郎, 線形不等式を解く制約ソルバーについて, サマーチュートリアル「制約プログラミング」, 日本ソフトウェア科学会 (1989), pp117-125

[Sakai89] Sakai,K., 処理系の一般論, サマーチュートリアル「制約プログラミング」, 日本ソフトウェア科学会 (1989), pp41-62

[SakaiSato89] Sakai,K., Sato,Y., *Boolean Grobner Bases,* 計算アルゴリズムと計算量の基礎理論, 数理解析研究所講究録 666

[Shostak77] Shostak,R.E., *On the SUF-INF Method for Proving Presburger Formulas,* J. ACM, Vol.24 (1977), pp529-543