制約論理型言語のための制約集合の依存性解析手法の制約アプリケーションへの適用

永井保夫*

1　はじめに

制約論理型言語は論理型言語への制約概念の導入により、1）対象となる構成要素やその属性間で成立する関係を関係表現や関数表現を用いてより宣言的に記述でき、2）プログラムの実行制御に関する表現もより宣言的に記述できるという特徴をもつ［4,8,15,2］。これにより、ユーザは問題の本質的な性質をのみを記述し、その問題の解法について記述する必要がなくなるということである。

しかしながら、制約論理型言語の処理系は、制約の対象領域によっては必ずしも効率のよい処理を実現しているとはいえない場合がある。一般に、制約集合に関する制約評価の効率は制約評価系に入力される制約の順序に依存する。制約論理型言語における制約評価系は推論機構がSLD-導出［12］によって集められた制約を解くため、その順序はプログラム中の順序によって決定される［2,13］。特に、われわれの対象としている制約論理型言語CALの代数制約評価系においてはグレブナ基底の計算に時間を要し、Buchbergerアルゴリズムは最悪の場合の計算量は変数の数の二乗のべきの複雑さとなる［1,7,2］。

一方、プロセスシステム工学の分野におけるシミュレーションや最適化問題では問題解決を非線形代数方程式系の解法に帰着させる場合が多い。このような代数方程式系を効率良く解くために、有効グラフを用いて方程式系のもつ代数構造情報を明確にし、その情報を用いて解法を決定する方法が研究されている［6,22］。また、幾何学的定理の証明問題の代数的手法であるWuの方法やグレブナ基底法による方法を効率的に実行するために、代数方程式系として表現された仮説ならびに証明すべき命題から変数の依存関係情報を求め、この情報に基づき定理証明手続きの処理性能を改善する研究がおこなわれている［10］。

そこで、われわれは、このようなプロセス工学や幾何学の定理証明で取り上げられている問題を非線形方程式により表現される制約アプリケーションとして定式化し，制約集合の依存性解析により求められる構造情報に基づいた制約評価系の効率化手法を適用する。本効率化手法では、グラフ論的手法により制約集合の変数の依存関係情報を求め、得られた情報から制約の評価順序ならびに変数の優先順位を変更し、代数制約評価系の処理効率を向上させることを試みる。ここでは、制約集合の依存性解析に基づいた効率化手法を制約論理型言語CALで記述されたプログラムに対して適用し、制約の評価順序ならびに変数の優先順位に関する考察をおこなう。

本稿では、まず制約論理型言語CALにおける代数制約集合の依存性解析手法の概要ならびにその主要な3つの処理について述べる。次に、制約論理型言語CALの代数制約集合の依存性解析について問題を用いて説明し、いくつかの問題に対する効率化手法の適用結果を示す。最後に、依存性解

*東京情報大学総合情報学部情報システム学科助教授

2002年11月12日受理

析による効率化手法の問題点および課題について考察する。

2 制約論理型言語

2.1 制約論理型言語の計算モデル

制約論理型言語の計算モデルは, 以下の図1のように定義する。

入力：制約論理プログラムPとゴールG

出力：PからGのsuccessfulな導出列が得られたとき解制約$C\theta$を出力；失敗したときは、失敗を出力する

アルゴリズム：

1 リテラル部と制約部からなる導出節R=〈RL;RC〉を入力ゴールGに初期化する；すなわち、R_n=〈G;ϕ〉とする；
2 **while** リテラル導出節RLが空でない **do**
3 $mgu\,\theta$により、L_1とPが単一化できるようなリテラルゴールL_1をリテラル導出節から選択する；
4 確定節P←P_1,P_2,…,P_n;C_1,C_2,…,C_lをPから選択する；
5 リテラル導出節からL_1を取り除き、P_1,P_2,…,P_nを加える；
6 **if**（$RC\cup C_2\cup C_2\cup\ldots C_l$）$\theta$が可解（solvable）
then これをあらたに制約導出節RCとする；
7 θを新しい導出節R_n（=〈RL;RC〉）に適用する；
8 **endwhile**
9 **if** リテラル導出節が空
then 解制約$C\theta$を出力する
10 **else** 失敗を出力する。

図1：制約論理言語の計算モデルの概要

制約論理型言語の大きな特徴として、制約評価系に対して与えられた制約が充足しているかどうかを調べるために、制約をインクリメンタルに評価し、冗長な計算を減らして、計算量をなるべく少なくする機能がある。

2.2 制約論理型言語CAL

制約論理型言語CALは複素数上の線形・非線形代数方程式（代数制約）、真偽値上の方程式（ブール制約）、実数上の線形方程式・不等式（線形制約）、集合とその要素に関する関係式（集合制約）という4種類の制約を取り扱うことができる［20］。

CALの代数制約評価系は、グレブナ基底を解として求める。グレブナ基底は多項式イデアルの計算をするために、B. Buchbergerによって提案された概念であり、その計算方法はBuchbergerアルゴリズムとして知られている［1,7,10］。Buchbergerアルゴリズムは、等式（制約）集合の各等式を簡約化するための、項書き換え系の適用操作とみなせる。つまり、制約集合が与えられ、さらにその要素である各制約の単項間に対して順序がつけられたとき、その順序のもとで最大の単項をそれ以外の多項式へ書き換え基底を求める。

図2はBuchbergerアルゴリズムの概要を示している。入力である等式集合Eを多項式の有限集合$F = \{f_1,\ldots,f_n\}$、出力であるEをFによって生成されるイデアルのグレブナ基底$G(F)$とする。

まず、$E \leftarrow F$, $R \leftarrow 0$とする。Eのすべての要素について、規則集合Rを用いて簡約化をおこない、規則集合Rを更新する（**ReduceAll**）。その結果から**NewBasis**によって新しい基底を求める。Eのなかから互いに等しくない2個の要素からなるペア $\{f_1, f_2\}$ をすべて求め、その集合をBとする。

次に、集合Bから任意のひとつの要素 $\{f_1, f_2\}$ を取りだし、S-多項式hを求め、Bから $\{f_1, f_2\}$ を消去する。さらに、Eに関するhの正規表現h'を計算する。もし、$h'=0$でなければ、新たなペア $\{\{g, h'\} \mid g \in G\}$ をBに加え、さらにEにh'を加え、Eのすべての要素について、規則集合Rを用いて簡約化をおこない、基底を更新する。このような操作をBの要素がなくなるまで繰り返す。**Normal Form/2**は正規表現を求める手続きを、**SPolynomial/2**はS-多項式を求める手続きをそれぞれ示す。

3　制約集合の依存性解析

制約論理型言語CALにおける制約集合の依存性解析ならびにBuchbergerアルゴリズムに基づいた代数制約評価系への適用による効率化について説明する。

われわれは、トップダウンなインタプリタの実行により求められる制約集合を2部グラフとして表現し、グラフ論的手法を用いて制約評価系を効率化する手法を提案した［16,18］。本手法では、制約集合がもつ代数的構造情報を抽出し、その情報に基づいて求められた制約間の依存関係情報から、プログラム中のゴールやサブゴールの並び換えならびに変数の優先順位を決定し、制約評価系を制御する。なお、現時点では解析の結果、複数の制約集合が存在する場合には、それぞれに対してプログラム中のゴールやサブゴールの並び換えならびに変数の優先順位の決定をおこなう。

制約集合の依存性解析は、トップダウンなインタプリタの実行による制約集合の導出、導出された制約集合の構造解析、制約の評価順序および変数の優先順序の決定という3つの処理からなる。以下では、それぞれの処理の内容について説明する。

```
1    E ← F;
2    R ← 0;
3    ReduceAll(E, R);
4    NewBasis(E, R);
5    B ← {{f1, f2} | f1, f2 ∈ E, f1 ≠ f2};
6        while B ≠ 0 do
7          choose a pair {f1, f2} ∈ B;
8          B ← B − {f1, f2};
9          h ← SPolynomial(f1, f2);
10         h' ← NormalForm(h, E ∪ R);
11         if h' ≠ 0 then
12           B ← B ∪ {{e, h'} | e ∈ E};
13           R ← R ∪ {h'};
14           ReduceAll(E, R);
15           NewBasis(E, R);
16         endif
17       endwhile
```

図2：Buchberger アルゴリズムの概要

3.1 トップダウンなインタプリタの実行による制約集合の導出

トップダウンなインタプリタの実行による制約論理プログラムの制約集合の導出処理では、トップレベルのゴールに対して、制約論理型言語のSLD導出に基づいた計算モデル（実行機構）から可能となる計算経路をトップダウンにトレースし、プログラムのふるまいに関する特徴、特にここでは制約集合を求める。

具体的な処理では、入力であるプログラムPとゴールG（$P \cup \{G\}$）に対するプログラムの計算経路に対応する探索木T_rをたどりながら、プログラム中でのデータ定義と参照の関係、すなわちデータフロー情報を解析し、出力として探索木上の成功路における制約集合Cと対応する代入θを求める。その場合、制約評価系が制約評価をおこなわない形で、制約論理プログラムを実行し、最終的に制約集合の代入例$C\theta$を求める。また、複数の成功路が存在する場合には、全解探索によりすべての制約集合、代入集合ならびに代入例を求める。

このような制約集合の導出アルゴリズムの概要は以下の図3に示される。

main/4はトップダウンなインタプリタを用いた制約集合の導出アルゴリズムのトップレベル手続きをあらわす。ゴールに関する解析**analyze_goal/3**では、実行可能なすべての述語呼びだしについての変数束縛に関する情報を、各述語に対応するすべての節に伝播し、各述語の静的なフロー情報を求める。節に関する解析**analyze_clause/4**では、節とゴールとのヘッドユニフィケーションをおこない、生成された代入情報をボディ部に伝播し、さらにサブゴールやヘッド間の変数共有情報を考慮してボディ部の解析**analyze_body/3**をおこなう。なお、再帰計算の場合には、無限ループによる無駄な計算をしないように、ループの検出をおこなう。

[アルゴリズム]
入力: プログラム P ならびにゴール $G(P \cup \{G\})$
出力: 探索木 T_r 上のすべての成功路における制約集合 C と代入 θ および代入例 $C\theta$ を求める

```
procedure main(Goal, Subst, Constr, Instance)
  begin
    Subst ← {}; % Subst is substitution. %
    Constr ← {}; % Constr ia constraint. %
    analyze_goal(Goal, Subst, Constr);
    Apply_subst_to_constr(Subst, Constr, Instance);
      % Instance is an instance of constraint by substitution. %
  end;
```

トップレベルの手続き

```
procedure analyze_goal(Goal, Subst, Constr)
  begin
    for each clause Cl_i of Goal do;
    analyze_clause(Cl_i, Goal, Subst, Constr);
    endfor
  end;
```

ゴール (述語) 解析アルゴリズム

```
procedure analyze_clause(Cl, Goal, Subst, Constr)
  begin
    let Cl be of the form Head :- Body;
    if unify(Head,Goal,Subst1);
      then Subst ← Subst ∪ Subst1;
           analyze_body(Body, Subst, Constr);
      else failed_unification(Head, Goal, Subst) ;
  end;
    節に関する解析アルゴリズム

procedure analyze_body(Body, Subst, Constr)
  begin
    let Body be of the form Literal_Body;Constraint_Body ;
    if Literal_Body is empty
      then return Constr ← Constr ∪ Constraint_Body;
      else let Literal_Body be of the form p(X), LBody_Tail;
    generate_goal(p, Subst, Cp);
    analyze_goal(Cp, Subst, Constr1);
    analyze_body(LBody_Tail, Subst, Constr2);
    Constr ← Constr ∪ Constr1 ∪ Constr2;
  end;
    ボディ部に関する解析アルゴリズム
```

図3：トップダウンなインタプリンタ実行による制約集合の導出

［問題1］

非線形代数制約を対象とした問題1 について考える。

```
?-   f1(x2,x5), f2(x2,x5,x7), f3(x3,x6,x7), f4(x1,x7),
     f5(x3,x5,x8), f6(x1,x4,x7), f7(x6,x8), f8(x1,x4).

f1(X,Y) : −2 * X^2 + Y = 2.
f2(X,Y,Z) : −X + Y^2 + Z = 1.
f3(X,Y,Z) : −X + Y + 2 * Z^3 = 4.
f4(X,Y) : −X + Y = 3.
f5(X,Y,Z) : −X^2 + 2 * Y + Z = 2.
f6(X,Y,Z) : −X^3 + Y + 3 * Z = 1.
f7(X,Y) : −X + Y^3 = 2.
f8(X,Y) : −X + Y = 4.
```

図4には問題1のトップダウン実行に基づいたインタプリタから求められた探索木ならびに制約集合と代入集合（代入例）を示す。

この例では、制約評価系による制約評価（*solve*($C \cup \cdots \cup RC$)）をおこなわないで、解制約集合（$C \cup \cdots \cup RC$）を求める。つまり、問題1では、各ゴール**f1(x2,x5)**，**f2(x2,x5,x7)**，**f3（x3,x6,x7)**，**f4(x1,x7)**，**f5(x3,x5,x8)**，**f6(x1,x4,x7)**，**f7(x6,x8)**，**f8(x1,x4)** のリダクションにより制約および代入が順次集められ、制約集合と代入集合 $\{\theta_1, \theta_2, \ldots, \theta_8\}$ が求められる（図4）。最終的には、制約評価系による制約の評価はおこなわれずに、制約集合Cの $\boldsymbol{\theta}$[1] **= {2*x2² + x5 = 2, x2 + x5² + x7 = 1, . . . , x1 + x4 = 4}** による代入例$C\theta$が求められる。本問題では、単一の制約集合のみを求めてい

[1] 代入 θ はそれぞれの代入の合成、すなわち $\theta_1 \circ \theta_2 \circ \cdots \circ \theta_8$から求められる

るが、問題によっては複数の制約集合を求めることが必要となる。

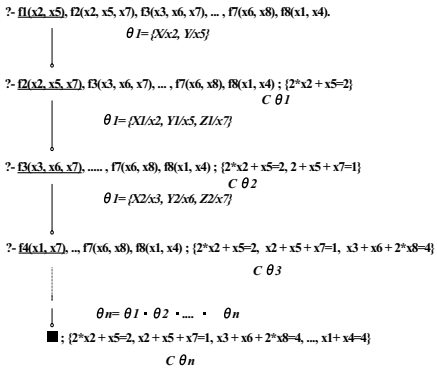

図4：問題1のトップダウン解析

3.2　2部グラフのDM分解を用いた制約集合の構造解析

3.2.1　制約ならびに制約集合のグラフ表現

制約とは対象の構成要素およびその属性間で成立する関係を宣言的に記述したものである。制約は関係や関数によって表現される。

われわれは、制約論理型言語CALの代数制約集合をグラフ表現することにする［16,17,18］。ここでは、特に制約集合を2部グラフ［16］表現し、制約集合のもつ代数的構造情報を抽出する。

図5は、問題1の解析により求められた制約集合の代入例$C\theta$を2部グラフ表現したものである。

3.2.2　2部グラフのDM分解

2部グラフ$G=(V^+,V^-;E)$におけるDM分解［5］とは既約成分への分解であり、半順序構造$\prec$をもつ$V(=V^+\cup V^-)$の部分集合の族である $\{G_-,G_+\} \cup \{G_i\}_{i=1}^{n}$に完全正準分解することである。前者の $\{G_-,G_+\}$ を不整合部、後者の $\{G_i\}_{i=1}^{n}$を整合部とよぶ。

制約集合を2部グラフ$G=(V^+,V^-;E)$として表現し、2部グラフのDM分解をおこなうことにより、制約集合を表現するグラフが構造的に可解であるかどうかを判定し、可解であれば部分問題$G_i=(W_i^+,W_i^-;E_i)$（i=1,…,n）に分割し、解を求めるための順序を決定できる。

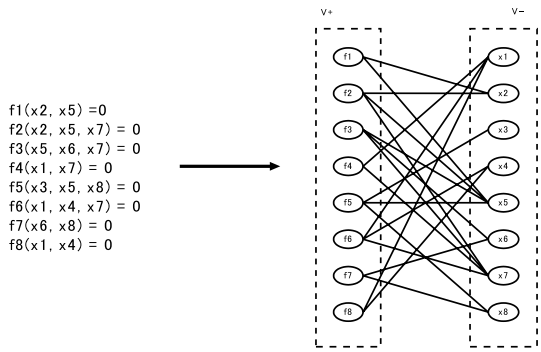

図5：問題1の2部グラフ表現

次に、2部グラフのDM分解の処理概要について説明する。具体的なアルゴリズムは、図6に示される。3行目では、2部グラフ$G=(V^{+},V^{-};E)$の最大マッチングMを求める。4行目では、最大マッチングMの各エッジの逆向きエッジをGに追加して得られる補助グラフ$G_M=(V^{+},V^{-};\tilde{E})$（$\tilde{E}=E\cup\{(u,v)\mid(v,u)\in M\}$）を求める。5行目では、不整合部 $\{G_{-},G_{+}\}$ に対する処理をおこなう。$V^{+}-\partial^{+}M$の点を始点とする補助グラフG_Mの有向道の終点全体を$U_{(-)}$とし、$V^{-}-\partial^{-}M$の点を終点とするG_Mの有向道の始点全体を$U_{(+)}$としたとき、$U_{(+)},U_{(-)}$によりGの部分グラフG_{-},G_{+}を誘導する。6行目では、$G_M-(U_{(-)}\cup U_{(+)})$を強連結成分$G_i=(W_i^{+},W_i^{-};E_i)$、（$i=1,2,\ldots,n$）に分解する。$G'=\{G_i\}_{i=1}^{n}$とする。7行目では、得られた $\{G_{-},G_{+}\}\cup G'$を半順序関係と矛盾しないように並べ換える。

```
1 procedure DM_decomposition(G:input, G':output, G'':output)
2   begin
3     find_maximum_matching(G, M);
4     make_auxiliary_graph(G, M, G_M);
5     induce_auxiliary_graph_and_handle_two_tails(G_M, G');
6     find_strongly_connected_component(G', G'');
7     permute_subgraph_merge(G'');
8   end;
```

図6：DM分解のアルゴリズムの概要

3.2.3　制約集合の依存関係情報

DM分解により求められた制約集合の構造情報は、次節の制約評価の処理効率の向上のために制約集合の依存関係情報として利用される。制約集合の依存関係情報はこの構造情報$\mathcal{G}=\{G_i\}$を半順序$\prec$に矛盾しないように並べ換え、$W_{-}^{-}=W_{+}^{+}=\phi$かつ$\mathcal{G}$によって得られる有向グラフをブロック三角化行列の形式（スパース構造）に表現することで求められる［3］。

つまり、この構造情報がスパース構造を有している場合、ブロック三角化行列が生成され、生成されたブロック三角化行列の順序に基づき部分問題（部分グラフ）(W_n^{+},W_n^{-}), $(W_{n+1}^{+},W_{n+1}^{-}),\ldots,(W_2^{+},W_2^{-})$, (W_1^{+},W_1^{-})に分割された制約間の依存関係情報が生成される。このような制約間の依存関係情報を求めることができれば、Buchbergerアルゴリズムの制御情報とみなした効率的な制約評価が期待できる［16,17,18］。

たとえば、図7は図5で表現された2部グラフをDM分解した結果、G_1, G_2, G_3という3つの部分グラフに構造分解されたことを示す。この結果から、各部分問題に関する制約の集合ごとに制約評価していけば、効率的な処理が期待される。

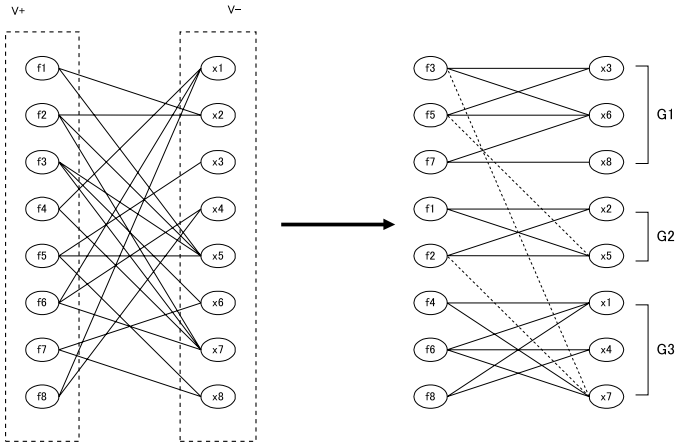

図7：2部グラフをDM分解した結果

3.3 制約の評価順序ならびに変数の優先順位に基づいた代数制約評価系の効率化

制約論理型言語における導出処理は第2節に示したようにリテラルに関する処理と制約に関する処理にわけられる。制約論理型言語の処理系を効率化するためには、両者の導出処理をどのように制御するかが重要な問題となる。たとえば、リテラルに関する導出処理ではリテラルゴールの選択規則ならびに選択されたリテラルと単一化する確定節の選択規則と確定節のボディ部のゴールの順序が全体の処理効率に影響を与える。一方、制約に関する処理では、リテラルの処理により集められた制約を制約評価系が解くので、その処理効率は制約の評価順序に依存する。

特に、後者の場合にはわれわれの対象としている制約論理型言語CALの代数制約評価系がBuchbergerアルゴリズムに基づくため、アルゴリズムは次のような性質を有する。1）非線形制約に対する基底計算で求められる多項式の次数が、最悪の場合で多項式環における変数の数の二乗のべきとなるような複雑さとなり、2）計算時間が制約の評価順序ならびに単項の優先順位に依存し、3）解の係数成長が計算時間に非常に影響を及ぼす。このような性質から、制約評価系における効率的な処理の実現が要求される［2,16］。

そこで、制約集合の依存性解析から求められた制約の評価順序ならびに変数の優先順位に基づいて制約評価系に対する制御情報の付与ならびにサブゴールの並び換えをおこなう。その結果、変換されたプログラムの実行時には、Buchbergerアルゴリズムに基づいた代数制約評価において冗長な計算（S-多項式）を減らし、解制約であるブレブナ基底の係数成長をなるべく抑制することで無駄な計算を避けた制約評価の効率化が期待できる［16,17,18］。

以下では、制約の評価順序ならびに変数の優先順位の決定方法を説明する。まず、DM分解によってブロック三角化をおこない、ブロック上三角化行列を求める。求められた依存関係情報に基づき、部分問題ごとに制約評価ができるようにゴール列を並べ変える。

次に、CALでは、変数間の優先順位を指定する述語pre/1が組み込み述語として提供されている。pre述語を用いることにより、部分問題G_nの変数集合 ≪ 部分問題G_{n-1}の変数集合 ≪ … ≪ 部分問題G_1の変数集合となるように変数間の優先順位を指定できる。Buchbergerアルゴリズムでは制約評価系を合流性を保証した項書き換え系とみなせるので、CALの制約評価系では、このようなpre述語を用いて決定した変数に対する優先順位により書き換え規則の変更が可能であり、制約の評価

順序を制御できる［21］。

なお、現時点では解析の結果、複数の制約集合が存在する場合には、それぞれに対して効率化手法を適用することが必要である。

4　制約集合の依存性解析による制約評価系の効率化手法の適用実験

非線形制約を取り扱った問題1から問題4までの問題に対して、依存性解析による効率化手法の適用実験をおこなった。

4.1　問題1（非線形制約求解問題）

問題1において、制約の評価順序を変更したプログラムを図8に示す。図8は求められた依存関係情報に基づき、部分問題G_3からはじめて、G_2, G_1という順序に従って、部分問題ごとに制約評価がおこなわれるように並べ換えたゴール列を示す。

```
:- f4(x1,x7),
   f6(x1,x4,x7),
   f8(x1,x4),          } G3
   f1(x2,x5),
   f2(x2,x5,x7),       } G2
   f3(x3,x6,x7),
   f5(X3,X5,X8),
   f7(x6,x8).          } G1
```

図8：制約の評価順序の指定（ゴールの並び替え）

また、依存関係情報に基づき求められた変数の優先順序は以下に示される。

$$[x_3, x_6, x_8] \gg [x_2, x_5] \gg [x_1, x_4, x_7]$$

4.2　問題2（熱交換器設計問題）

問題2では、図9に示される三連向流熱交換器の設計問題［14,6］を取り上げる。

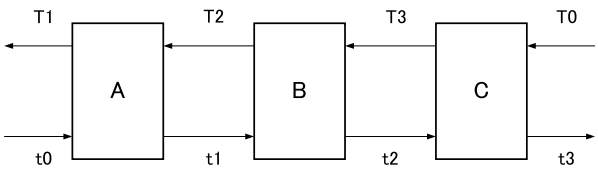

図9：三連熱交換器

単一の熱交換器では、高温側流体の温度をT、低音側のそれをt、熱交換器の入口・出口を添字i、oであらわすと、向流側での入口・出口の関係は次式で与えられる。

$$P=\frac{t_0-t_i}{T_i-t_i}(0<P<1) \qquad (1)$$

$$R=\frac{T_i-T_0}{t_0-t_i}=\frac{gc}{GC} \qquad (2)$$

$$\frac{UA(R-1)}{gc}=\frac{1-P}{1-PR}^{2)}(R\neq 1) \quad (3)$$

われわれが対象とする三連向流熱交換器は、単一の熱交換器に対応する式（1)、(2)、(3）の組に入口、出口温度に所定の変数を対応させて3つ接続させたものである。この三連向流熱交換器の設計問題では、未知変数間の関係は最終的には線形になり、有理数解を求めることができる。図10にはゴールとCALプログラムを、図11には制約を並べ換えたCALプログラムを示す。

ゴール G

```
heat_exchanger([[20,t1,t2u,t1u,20],
                [t1,t2,t3u,t2u,10],
                [t2,t3,300,t3u,15]]).
```

プログラム P

```
:- public heat_exchanger/1.

heat_exchanger([]).
heat_exchanger([[Ti,To,Tiu,Tou,A]|T]) :-
          (Tiu-Ti)*P=(To-Ti),
          (To-Ti)*5000000=(Tiu-Tou),
          Z=80*A*4999999/4000,
          (1-P*5000000)*Z=(1-P),
          heat_exchanger(T).
```

図10：ゴールとCALプログラム

また、依存関係情報に基づき求められた変数の優先順序は以下に示される。

［*tu*1］≫［*tu*2］≫［*tu*3］≫［*t*3］≫［*z*3］≫［*t*2］≫［*z*2］≫［*t*1］≫［*z*1］

```
:- public heat_exchanger/1.

heat_exchanger([]).
heat_exchanger([[Ti,To,Tiu,Tou,A]|T]) :-
          Z=80*A*4999999/4000,
          (1-P*5000000)*Z=(1-P),
          (Tiu-Ti)*P=(To-Ti),
          (To-Ti)*5000000=(Tiu-Tou),
          heat_exchanger(T).
```

図11：制約が並び換えられたCALプログラム

---

2) ここでは $exp\{\frac{UA(R-1)}{gc}\}=\frac{1-P}{1-PR}(R\neq 1)$ の近似式を用いている

4.3　問題3および問題4（幾何定理証明）

問題3ならびに問題4では、非線形代数制約を対象とした幾何学的定理の証明問題を取り上げる。この問題では、座標系を設定し、その座標系に基づいて仮説と結論からなる幾何定理を代数多項式に変換し、代数的手法であるグレブナ基底による方法を用いて証明をおこなう。効率的に幾何学的な定理を証明するためには、そこで取り扱われる多項式集合（仮説集合*Hyp*と結論*c*）のグレブナ基底を効率的に求める求めるかが重要な問題となる。

グレブナ基底による方法を用いた定理証明では、結論を示す代数多項式が、仮説を示す代数多項式から生成されるイデアルに属するかどうかを決定するイデアルのメンバシップ問題とみなし、定理が成立するかどうかを決定する。この方法には、直接的なアプローチと反駁的なアプローチがあり、本実験では仮説のグレブナ基底（標準形）を求め、求められた基底を用いて結論が正しいかどうかを判定する［11,10］アプローチを適用する。

ここでは幾何学的な証明問題の例として、まず3垂線定理の証明問題［11］を取り上げ、次に、Hilbertの9点円定理の証明問題［11］を取り上げる。

4.3.1　3垂線定理の証明問題

3垂線定理は「三角形*ABC*においてその高さ*BE*と*CF*の交点を*H*としたとき、直線*AH*は直線*BC* と直交する」（図12）という命題をあらわす。座標系として、$A=(0,0)$, $B=(u_1,0)$, $C=(x_1,x_2)$, $E=(u_2,u_3)$, $H=(x_1,x_3)$ とする。

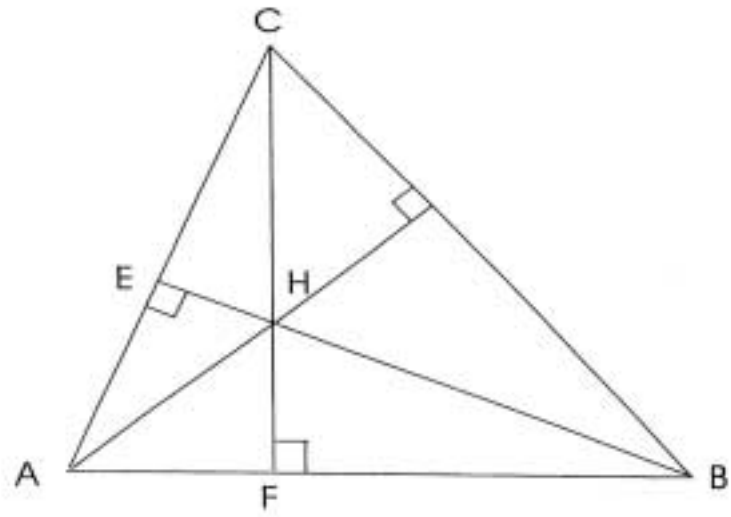

図12：3垂線の定理

仮説を示す多項式制約*Hyp*は次のようになる。

$x_2u_3+(u_2-u_1)x_1=0$；*BE*は*AC*と直交する（1）

$-u_2x_2+u_3x_1=0$；*A, E, C*は一直線上にある（2）

$(u_1-u_2)x_3+u_3x_1-u_1u_3=0$；*B, E, H*は一直線上にある（3）

また、本定理では、（$x_2\neq 0$かつ$u_1\neq 0$）、または$u_3\neq 0$という補助条件*S*が必要となり、そのために新しい変数z_1,z_2を導入し、次のように表現する。

$(z_1x_2u_1-1)(z_2u_3-1)=0$（4）

定理の結論をあらわす代数制約*Conc*は

$x_2x_3=-x_1(x_1-u_1)$；*AH*は*BC*と直交する　(5)

のようになる。

直接的なアプローチでは、まず仮説集合$Hyp=\{x_2u_3+(u_2-u_1)x_1=0, -u_2x_2+u_3x_1=0, (u_1-u_2)x_3+u_3x_1-u_1u_3=0\}$ に補助条件（または非退化条件）$S=\{(z_1x_2u_1-1)(z_2u_3-1)=0\}$ を加えた代数制約集合のグレブナ基底を計算する。次に、結論$Conc=(x_2x_3=-x_1(x_1-u_1))$が正しいかどうかを、求められたグレブナ基底を用いて判定する。すなわち、基底の要素である多項式を用いて結論*Conc*を書き換えていき、既約になるまで簡約化をおこない、その正規形がゼロになるかどうかを判定する。

図13は、図12の問題の代数制約の構造情報をあらわす。変数の優先順位、特に従属変数間の順位は、代数制約の構造情報の解析により求められる制約間の依存関係（変数の共有関係）情報から決定される。問題3では、図13の構造情報の解析結果に基づき、x_1,x_2,x_3,u_1,u_2を従属変数、u_3,z_1,z_2を独立変数とみなし、次のような変数の優先順位を決定した。

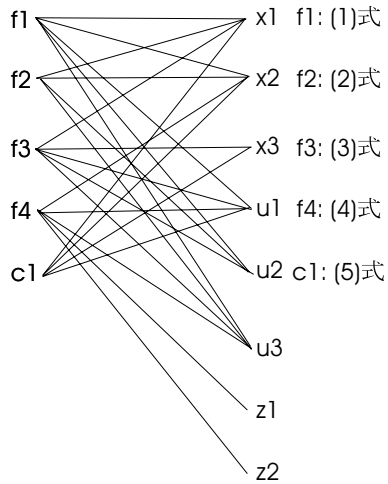

図13：3垂線の定理の代数制約集合の依存関係情報

$[x_1,x_2,x_3,u_1,y_2] \gg [z_1,z_2] \gg u_3$

または

$[x_1,x_2,x_3,u_1,y_2] \gg [u_3,z_1,z_2]$

図14は直接的なアプローチにより記述したCALプログラムを示す。図15は構造解析の結果を反映して制約の評価順序を変更するためにゴールを並べ換えたプログラムを示す。図14および図15ではプログラムの実行により定理が証明された結果がそれぞれ0として表示される。なお、evalwrite/1は引数である多項式（代数制約）をグレブナ基底の要素である多項式を用いて既約になるまで書き換えた結果を表示する述語である。また、変数の順序付けはCALの組み込み述語preをもちいておこなった。

```
:- public tp_orthocenter/0.
%% Theorem of Orthocenter  %%
tp_orthocenter  :-
      x2*u3+(u2-u1)*x1=0,
      -u2*x2+u3*x1=0,
      (u1-u2)*x3+u3*x1-u1*u3=0,
      (z1*x2*u1-1)*(z2*u3-1)=0,
      evalwrite(x2*x3+x1*(x1-u1)).
```

図14：直接的なアプローチにより記述したプログラム

```
:- public tp_orthocenter1/0.
%% Theorem of Orthocenter  %%
tp_orthocenter1  :-
      x2*u3+(u2-u1)*x1=0,
      -u2*x2+u3*x1=0,
      (z1*x2*u1-1)*(z2*u3-1)=0,
      (u1-u2)*x3+u3*x1-u1*u3=0,
      evalwrite(x2*x3+x1*(x1-u1)).
```

図15：ゴールを並べ換えたプログラム

4.3.2　Hilbertの9点円定理の証明問題

本節で取り上げているもう一方の例としてはHilbertの9点円定理（図16）の証明をとりあげ、効率化手法を適用した。

9点円定理は「三角形ABCの三頂点から対辺に下ろした垂線の足をD、E、F、垂心をHとして、AH、BH、CHの中点をL、M、N、3辺の中点をP、Q、Rとすれば、9点D、E、F、L、M、N、P、Q、Rは同一円周上にある」という命題である。図17には直接的なアプローチにより記述したプログラムを、図18にはゴールが並べ換えられたプログラムをそれぞれ示す。

なお、変数の優先順位は次のように決定された。

$$[x_h] \gg [x_g, y_g] \gg [x_f, y_f] \gg [x_a, x_b, y_c] \gg [z_1, z_2, z_3, z_4]$$

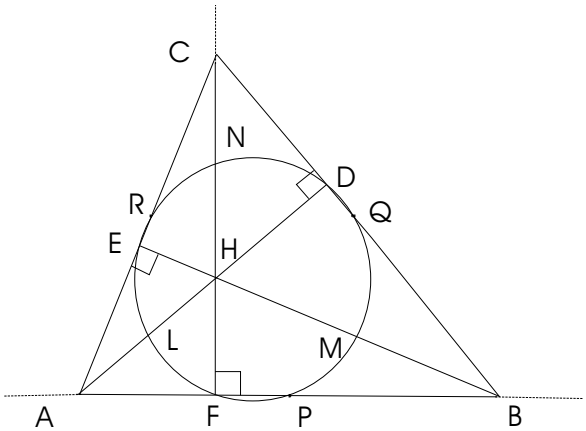

図16：9点円の定理証明問題

```
:- public tp_ninepoint/0.
%% Theorem of Ninepoint  %%
tp_ninepoint  :-
    z1*(xa*yc-xb*yc) -1=0,
    z2*(xa-xb)-1=0,
    z3*(xb^2+yc^2)-1=0,
    xa*xb-xb*xf+yc*yf=0,
    xb*yf+yc*xf-yc*xb=0,
    z4*(xa^2+yc^2)-1=0,
    xa*xb-xa*xg+yc*yg=0,
    xa*yg+yc*xg-yc*xa=0,
    xa+xb-2*xh=0,
    evalwrite(-xh*yf*yg^2+xh*yf^2*yg
                        -xf*xh^2*yg+xf^2*xh*yg+xg*xh^2*yf-xg^2*xh*yf).
```

図17：直接的なアプローチにより記述したプログラム

```
:- public tp_ninepoint1/0.
%% Theorem of Ninepoint  %%
tp_ninepoint1  :-
    z1*(xa*yc-xb*yc) -1=0,
    z2*(xa-xb)-1=0,
    z3*(xb^2+yc^2)-1=0,
    z4*(xa^2+yc^2)-1=0,
    xa*xb-xb*xf+yc*yf=0,
    xb*yf+yc*xf-yc*xb=0,
    xa*xb-xa*xg+yc*yg=0,
    xa*yg+yc*xg-yc*xa=0,
    xa+xb-2*xh=0,
    evalwrite(-xh*yf*yg^2+xh*yf^2*yg
                        -xf*xh^2*yg+xf^2*xh*yg+xg*xh^2*yf-xg^2*xh*yf).
```

図18：ゴールを並べ換えたプログラム

5 実験結果

問題1から問題4までの非線形制約を取り扱った問題を記述したプログラムに対して、制約集合の依存性解析に要する時間ならびに効率化手法を適用しない場合と適用した場合のそれぞれのプログラムの実行時間を求め、評価をおこなった。制約集合の導出ならびに依存関係解析に関する処理はSUN4上のSicstus Prologにより実現されたシステムを用いた［19］（表1）。また、プログラムの実行はSUN4上で、制約論理型言語CALバージョン1.4を用いた（表2）。

問題1から問題4まで適用した結果、表1 のように約1/3から1/300まで処理時間の改善がみられた。特に、問題3や4のようなグレブナ基底法を用いた幾何定理証明では、変数の優先順位が処理時間に非常に影響を与えるため、その設定をいかにおこなうかが問題になる。われわれが提案した効率化手法では、代数制約の構造情報を解析し、得られた構造情報から制約の不足（不十分）状態や制約の矛盾または競合した状態を判定し［16,17］、独立変数ならびに従属変数への変数集合の分類をおこなっている。問題1から問題4までの問題は、制約集合のグラフ表現がスパース（疎）な構造をと

るものであり、効率化手法が有効な場合である。しかしながら、グラフ構造がスパースでない（密である）場合には、本手法はあまり有効ではなかった。次に、制約集合の依存性解析に要する時間と効率化手法によるプログラムの実行時間の改善度を比較する。問題1 はプログラム実行時間の改善度に対して解析に要する時間がかかりすぎているので、本効率化手法の有効性を十分に示しているとはいえなかった。また、問題2から問題4では、制約集合の依存性解析に要する時間とプログラムの実行時間の改善度を比較した結果、われわれが提案した効率化手法の有効性が示せた。

表1：プログラム解析に要した時間（単位:msec）

対象とした問題	解析時間
問題 1 (8 制約, 8 変数)	1051
問題 2 (熱交換器設計: 12 制約, 10 変数)	1151
問題 3 (3 垂線定理: 5 仮説，8 変数)	768
問題 4 (9 点円定理: 9 仮説，12 変数)	901

表2：非線形制約を対象としたプログラム実行時間（単位:msec）

対象とした問題	適用前	適用後
問題 1 (8 制約, 8 変数)	610	68
問題 2 (熱交換器設計: 12 制約, 10 変数)	281279	1322
問題 3 (3 垂線定理: 5 仮説，8 変数)	7565	2371
問題 4 (9 点円定理: 9 仮説，12 変数)	3524098	12753

6　問題点と今後の課題

本効率化手法では、次の項目については十分対処されておらず、今後の課題としての検討が必要である。

- 制約集合の構造分解において、処理の効率化という点から制約評価系に適した部分問題の大きさ（粒度）を明確にすることが必要である。
- 効率化手法の適用による実行時間の改善度は部分問題間の依存関係（インタラクション）の度合いに依存するので、なんらかの尺度の設定が必要である。
- 依存関係情報に基づいた効率化手法では、変数の共有関係だけでなく、共有する変数の次数に関しても考慮した制約の評価順序ならびに変数の優先順位を決定する必要がある。
- 部分問題を評価する（解く）順序は部分問題間の半順序関係を用いて決定しているが、上記の部分問題間の関係の度合いや変数の次数などの情報を考慮した決定方式を考えることが必要である。

また、われわれの提案した効率化手法では、制約評価をおこなわないで実際の領域上でプログラムを実行するため、プログラムの大規模化によりその取り扱いが非効率になる。このような問題に対処するためには、プログラムを実際に実行するのではなく、近似計算による実行過程からプログラムの性質を求める手法が必要である。そこで、今後の課題としては以下のような項目について検討していく必要がある。

- 抽象解釈の枠組の適用、特に制約ならびに制約評価の抽象化［9］
- 他領域（例えば、ブール領域）の制約評価に対する制約の構造解析手法の有効性
- 複数の制約集合が求められた場合の有効な取り扱い

7 まとめ

制約論理型言語の代数制約により表現された制約アプリケーションに対して、制約集合の依存性解析を用いた効率化手法を適用した。その結果、制約集合のグラフ表現がスパース（疎）な構造をとる問題に対しては、本効率化手法が特に有効であることを示した。今後は、より実用的な制約アプリケーションに対して、提案した効率化手法を適用し、プログラムの解析時間ならびに実行時間に関する評価をおこなう予定である。

参考文献

［1］Buchberger, B.: Grőbner Bases: An Algorithmic Method in Polynomial Ideal Theory, Publications/ Reports, RISC-Linz Series no. 83-29.0（1983）.

［2］Cohen, J.: Constraint Logic Programming Languages, ***Comm. of the ACM,*** Vol.33, No.7, pp.52-68（1990）.

［3］Coleman, T.F. et al.: Predicting Fill for Sparse Orthogonal Factorization, ***J. ACM,*** Vol.33, No.3, pp.518-532（1986）.

［4］Dincbas, M.: Constraint Logic Programming and Deductive Database, ***Proc. of France-Japan Artificial Intelligence and Computer Science Symposium 86***（1986）.

［5］Dulmage, A.L. and Mendelsohn, N.S.: Two Algorithms for Bipartite Graphs, ***J. SIAM, Vol.11, No.1,*** pp.183-194 March（1963）.

［6］Henry, E.J. and Williams, R.A.: ***Graph Theory in Modern Engineering, Computer Aided Design, Control, Optimization, Reliability Analysis,*** Academic Press（1973）.

［7］Hoffmann, C.M.: Grőbner Bases Techniques, in Chapter 7, ***Geometric & Solid Modeling, An Introduction,*** Morgan Kaufmann Publishers, Inc.（1989）.

［8］Jaffar, J. and Lassez, J.-L.: Constraint logic programming, In ***Procs. of the Fourteenth ACM Symposium of the Principles of Programming Languages***（1987）.

［9］堀内謙二：論理プログラムの抽象解釈を用いた解析、ICOT Technical Memorandumn（1992）.

［10］Kapur, K. and Mundy, J.L.: Special Volume on Geometric Reasoning, ***Artificial Intelligence*** Vol. 37, No.1-3, December（1988）.

［11］Kutzler, B.: Algebraic Approaches to Automated Geometry Theorem Proving, PhD Thesis,Johannes Kepler Unversity, RISC-LINZ Series no. 88-74.0（1988）.

［12］Lloyd, J.W.: ***Foundations of Logic Programming***（邦訳：論理プログラミングの基礎、佐藤雅彦他訳）, Springer-Verlag（1984）.

［13］Marriott, K. and Sondergaard, H.: Analysis of Constraint Logic Programs, ***Proc. of the North American Conf. on Logic Programming,*** pp.531-547（1990）.

［14］三井東圧EQM研究会：パソコンのための方程式解法ソフト-EQUATRAN-M入門、（財）省エネルギーセンター（1987）.

［15］淵一博監修、溝口文雄他編：制約論理プログラミング、知識情報処理シリーズ別巻2、共立出版（1989）.

［16］永井保夫：制約グラフの構造分解および整合性解析による代数制約評価系の効率化についての検討、人工知能学会研資SIG-F/H/K-9001-12（1990）.

［17］永井保夫：代数制約の構造情報を用いた幾何定理証明の効率化手法の検討、情報処理学会第42回全国大会6F-1, pp.224-225（1991）.

［18］永井保夫、長谷川隆三：データフロー解析による制約論理型言語の処理系の効率化検討、1991年度人工知能学会全国大会12-5, pp.501-504（1991）.

［19］永井保夫、長谷川隆三：制約論理型言語におけるプログラム解析システムの実現検討、日本ソフトウエア科学会第8回大会、B2-2,（1991）.

［20］Sakai, K. and Aiba, A.: CAL: A Theoretical Background of Constraint Logic Programming and its Application, ***J. Symbolic Computation 8,*** pp. 589-603（1989）.

［21］佐藤洋祐：制約プログラミング−処理系の具体例（Grobner Base）、日本ソフトウエア科学会サマーチュートリアル（1989）.

［22］高松武一郎、橋本伊織、冨田重幸：有向グラフによる代数方程式系の構造解析とその応用、化学工学論文集、第8 巻、第4号、pp.500-506（1982）.