視覚的制約プログラミングについて

2K−10

町田 和浩　川越 恭二*　相場 亮**

日本電気技術情報システム開発（株）　*日本電気（株）
**(財)新世代コンピュータ技術開発機構

1 はじめに

制約論理プログラミング言語は，それまでの論理プログラミング言語と比較してより宣言的に問題を記述できる高レベルなプログラミング言語である．しかし，現状ではプログラミング言語特有の問題でもある，変数の名称や，述語の名称などを定義する必要があり，問題自体の記述に集中できないという問題がある．

視覚的制約プログラミングは，問題を視覚化することにより記述しやすい表現形式と，視覚的な解の表現方法を提供し，問題の本質に集中することのできる問題解決系を実現するものである．

本稿では，ICOTで開発された制約論理プログラミング言語CAL[3]における視覚的なプログラミング環境の実現方法について報告する．

以下，制約論理プログラミングの問題点と，視覚的制約プログラミングの概要について述べる．

2 制約論理プログラミング

制約論理プログラミング[2]では，問題領域の各対象間における関係を制約として記述することによりその問題を表現することができる．このように問題を制約で表現するため，「制約が実際に成立しているか」，「制約を満たすような変数の値は何か」などをユーザは知る必要はなく，問題を表現するためには制約にのみ集中すれば良い．

しかし，制約は式の形で表現しなければならず，複雑な問題の記述や理解は容易ではない．

また，現状の制約論理プログラミング言語は，一般には制約解消系と呼ぶ機構を論理プログラミング言語の処理系の一部として構築することにより制約の記述や制約の解消を実現している．しかし，制約解消系では，与えられた問題領域での制約の充足可能性を判定し，制約の標準形を求めるが，これだけでは結果の確認が困難な場合があり，実用的なシステムとするためには具体的な答えを導くような機能が必要となる．

3 視覚的制約プログラミング

前節に述べたように，問題の本質を捉えやすく，かつ記述しやすい表現方法と，制約解消結果の確認のしやすさを実現するために，制約論理プログラミング言語の視覚的プログラミング機能を開発した．

視覚的プログラミングでは，基本的に以下の機能が考えられる．

- 視覚的シンボルとしての制約の入力機能
  制約集合と視覚的シンボルを一対一に対応させて，制約を視覚的シンボルとして入力することにより，問題とは直接関係のない制約論理プログラミング言語におけるリテラルや変数を考慮せずに，より問題領域の対象に近い形で記述することができる．

- 制約評価結果の視覚的表現機能
  本視覚的プログラミング機能では，制約を式と視覚的シンボルとして記述することができる．式として記述された制約は，ユーザが記述した視覚的情報とは独立に記述できるため，必ずしも全ての制約が視覚的情報に反映されているとは限らない．また視覚的シンボルに影響を与えるような属性値(直線の長さなど)を変化させた時，視覚的情報もそれに対応して変化させなければ制約解消結果と視覚的情報との間に矛盾が生じてしまう．

  そこで，制約解消された結果を視覚的情報についての制約と考えると，視覚的情報と制約解消結果の間で矛盾の生じるものについて，制約の範囲内で任意の値に変更することにより，視覚的情報と制約解消結果との同期をとることができ，制約の標準形だけでは確認が困難であった具体的な答えを，視覚的な情報として容易に確認することができる．

対象問題領域を幾何問題に見られるような幾何図形を対象とする問題領域に限定した場合では，図形という視覚的な情報がそのまま制約となるので，上記の機能を実現するのは他の問題領域と比較して容易である．

さらに，対象問題領域を有向グラフで問題記述が可能な領域にした場合，有向グラフのノードが幾何図形の点の概念を，アークが直線の概念を拡張したものとすると，有向グラフでの視覚的表現は幾何図形の視覚的表現を拡張したものと考えられる．このような有向グラフが扱えれば，かなり一般的な問題領域を抽象的なグラフで表現できる．

このような理由から，本視覚的制約プログラミング機能では，扱える視覚的シンボルを，幾何学図形(点，直線，角，円)と有向グラフとした．この結果，幾何問題のような制約集合と視覚的情報が対応しやすい問題と，一般的な問題領域を抽象的なグラフ表現で表す問題の両者を扱えるようにした．

具体的には，制約の記述性を高めるために以下の機能を加えた．

- 対象領域の特化(制約の抽象化)
  複雑な問題を扱うためには，制約論理プログラミング言語レベルの基本的な制約式を組み合わせて記述しなければならない．また，現在の制約論理プログラミング言語では，ユーザが問題にあった制約解消系の選択をしなければならない．

  そこで，視覚的表現形式で記述する段階と制約論理プログラミング言語で記述する段階の間に，問題領域に特化した解釈系(ドメイン)を組み込む機能をつけた．これにより，ユーザはドメインを選択することで，問題を対象に近い形の制約として扱うことができるようにした．

- 対象領域の階層化
  ユーザはあらかじめ定義されているプリミティブな視覚的シンボル(点，直線...)を組み合わせて問題領域を記述することができる．しかし，問題が複雑になるとシンボル間の関係を記述する量が増え，解りにくいものになってしまう．

  本視覚的制約プログラミング機能では，ユーザが定義した視覚的シンボルの集合とその集合内で満足すべき制約との対応関係を定義できる枠組を作ることにより，シンボル間の制約記述量を抑え，階層的に問題を定義，編集することができるようにした．

4 実現方法

図1に視覚的制約プログラミング機能の基本構成を示す．

---

[0] On Visual Constraint Programming,
Kazuhiro Machida, Kyoji Kawagoe*, Akira Aiba**,
NEC Scientific Information System Development ,Ltd, *NEC Corporation,
** ICOT.

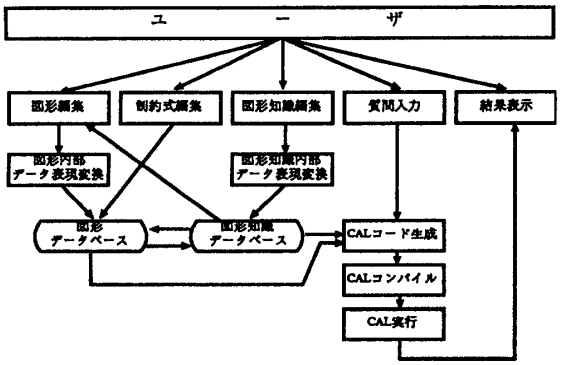

図 1: 視覚的制約プログラミング機能の基本構成

扱える図形には点，直線，角，円，ノード，アークそしてユーザ定義図形がある．点は位置情報のみを持つ最もプリミティブな図形である．直線，角，円はそれら固有の属性値と，制約を持つ．ノード，アークは複数の属性値を持つことができる．ユーザ定義図形は，上記のプリミティブな図形およびユーザ定義図形を組み合わせたものである．ユーザ定義図形は名前，制約式と固有の属性値を持つことができる．

処理の流れの概要は以下の通りである．

(1) ユーザはウィンドウ上に，視覚的シンボルである図形を用いて求めたい問題領域を記述していく．この時，あらかじめ登録された図形知識データベースに対象問題領域の要素を用いて，階層的に問題を記述することができる．

(2) システムは，与えられた図形に対応する抽象的な制約を図形データベースから取りだし，制約解消系で実行可能な形に変換した後，制約論理プログラミング言語の CAL を用いて制約解消を行なう．

(3) そして，現在表示されている図形の表示情報を，得られた制約の一般形と比較して，矛盾する所だけを修正する．この結果，ユーザは制約解消結果の具体的な答えを視覚的に確認することができるようにする．

5 実行例

ここでは 2 つの例をあげて，視覚的制約プログラミングの実行イメージを説明する．

5.1 幾何問題

以下の問題を解くことにする．

直角三角形 abc の斜辺 ab を 3 等分する点を d,e とし，線分 cd,ce の長さをそれぞれ x,y とする．

辺 bc=8, 辺 ac=6 の時，x,y を求めよ．

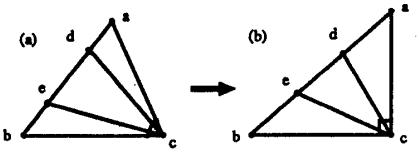

図 2: 例 1

本視覚的制約プログラミング機能では，図 2(a) の様な図と次のような制約を入力することにより，問題の記述を終了する．

```
ab = 3*de. de = ad. ad = be.
point_on_line(ab,ae,be). point_on_line(ab,ad.bd).
```

point_on_line(a,b,c) は，直線 a が直線 b と直線 c からなることを定義する制約 [1] で，ドメインに登録されている．

この問題について，CAL の Buchberger アルゴリズムを用いた制約解消系を利用して制約を解消すると，求める結果 ($x^2 = 208/9, y^2 = 292/9$) となる，結果は図 2(b) のように，制約解消により得られた具体的値 (座標) を画面上の図形座標に変換して図形を表示させる．

5.2 PERT

グラフタイプの図形を用いて次のような PERT 図 (最早日程表) の各ノードでの最早ノード時刻を求める問題を解くことにする．

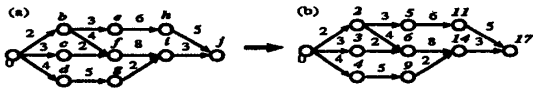

図 3: 例 2

図中の b ～ j は，未設定の最早ノード時刻を表す．

各ノードの最早ノード時刻を計算するために，そのノードを終点とするアークの値 (作業時間) の最大値を求める制約式 (max/3) をドメインに登録する．制約解消系として，単一化アルゴリズムを利用する．

```
max([], Tmp, Max) :- Tmp = Max.
max([Car|Res], Tmp, Max) :- Car =< Tmp,
    max(Res, Tmp, Max).
max([Car|Res], Tmp, Max) :- Car >= Tmp,
    max(Res, Car, Max).
```

始点ノードと，終点ノードの関係に注目すると，上図はこの関係をもとにして階層的に記述することができる．

以上の問題設定を行ない制約評価を行なう．結果は，図 3(b) のように未設定であった各ノードの最早ノード時刻を計算して図形表示される．

6 まとめ

視覚的制約プログラミング機能を PSI-II の上に実現した．前節の例で示したように制約論理プログラミングにおいて，視覚的シンボルとしての制約の入力や，制約評価結果の視覚的表現が有効であることが確認できた．また，問題領域に対応した制約の抽象化を行なうことにより，ユーザが制約の記述する量を抑えることで，記述ミスを少なくすることができる．しかし，有向グラフにより一般的な問題領域を表すとき，抽象的な表現にならざるを得ない．アイコンを組み合わせて問題対象を表現する方法 [4] 等の，よりわかり易い視覚的表現方法を考えるのが今後の課題である．

謝辞

本研究は，第五世代コンピュータ・プロジェクトの一環として行なわれた．御支援頂きました長谷川 隆三 ICOT 研究部長代理を初めとする方々に深く感謝いたします．

参考文献

[1] B.Kutsler. *"Algebraic Approaches to Automated Geometry Theorem Proving.* PhD thesis, Johannes Kepler Univ.,Austria, 1990.

[2] J-L.Lasses 溝口, 古川 (編). 「制約論理プログラミング」. 共立出版, 1989.

[3] 相場 坂井. 「cal: 制約論理プログラミングの理論と実例」. 情報処理学会, ソフトウェア基礎論 *24-2*, 1988.

[4] 岩戸 小山田. 「リアルタイムソフトウェアの視覚的な設計手法」. *IPA*, 第 *9* 回 *IPA* 技術センター発表会, 10 1990.