制約プログラミングを用いたクラス・ライブラリの設計

野村　紀匡　（沼田　一道　助教授）

1　はじめに

制約プログラミングは，制約充足問題 (Constraint Satisfaction Problems) を解くための手法である．制約充足問題とは，変数 x_i（$i = 1, \ldots, n$），変数の領域 d_i（x_i の取り得る値），制約 c_{ij}（x_i と x_j 間の可能な組み合わせ．$j = 1, \ldots, n.$）のすべての変数に対して制約に整合する値の割り当てを求める問題である [2]．制約プログラミングの特徴は，制約充足問題を変数と変数間の制約の組み合わせとして表現することと，解法であるアルゴリズムの選択とを分けることとにある．

制約プログラミングに関する研究は，1980 年代中旬に論理プログラミングの応用である制約論理プログラミング (Constraint Logic Programming) の研究により始まった．1980 年代後半から，Prolog 言語の拡張版として制約プログラミング・ライブラリが開発された．1990 年以降は商用化が進み，スケジューリング問題のように特定の問題領域に特化した制約の形式化と，C++ のような汎用的手続き型言語による実装の研究が行われ，現在では製造，通信，金融等様々な業種の，広範囲の問題に適用されている．

しかし，一般のソフトウェア・エンジニアが制約プログラミングを利用した応用プログラムを開発するのは依然困難である．なぜなら，制約プログラミング・ライブラリの多くが実務で使用される機会の少ない Prolog 言語の拡張として実装されているからである．また，JCL（Java Constraint Library[1]）という Java で実装され，かつソースが公開されている制約プログラミング・ライブラリがあるが，実際に JCL を使用してみたところ機能は豊富であるものの，使い勝手が良くなかった．そこで本研究では，一般のソフトウェア・エンジニアが制約プログラミングを利用した応用プログラムをより容易に開発できるようにするために，Java や C++ のような一般的なオブジェクト指向プログラミング言語で実装することができる制約プログラミング・ライブラリを設計する．設計は JCL と Roy[3] を参考に行い，JCL よりもシンプルで拡張性の高い制約プログラミング・ライブラリの設計を目標にする．

2　要件定義及び設計方針

本研究で設計する制約プログラミング・ライブラリ（以下本ライブラリ）はクラス・ライブラリである．想定する利用者はソフトウェア・エンジニアであり，応用プログラムを開発するときに本ライブラリをツールとして使用する．

本設計では，n-クイーン問題を制約充足問題のサンプル（以下サンプル問題）として用いる．n-クイーン問題とは $n \times n$ の盤上に，チェスのクイーン n 個を互いに取り合わないように

配置する問題 [2] である．本ライブラリに要求される機能要件は，サンプル問題のような制約充足問題をコンピュータ・プログラムとして表現することと，解を導出することである．

また本設計では，JCL における次の問題点を改善する．１つ目は，単項制約条件と二項制約条件しか定義することができないことである．JCL ではサンプル問題の n が増えれば増えるほど，問題をプログラム上に表現することは困難になる．２つ目は，利用者独自の制約条件を定義することが容易ではないことである．本設計では”拡張性”に配慮し，利用者が独自のクラスを追加できるよう，主要なクラスは階層的に設計する．

3　機能設計及びクラス設計

本ライブラリの機能は問題定義機能と解探索機能の２つの大機能から構成する．問題定義機能は制約充足問題をプログラム上に表現する機能，解探索機能はその解を探索する機能である．図 1 に，これらの機能を実装するクラスを示す．

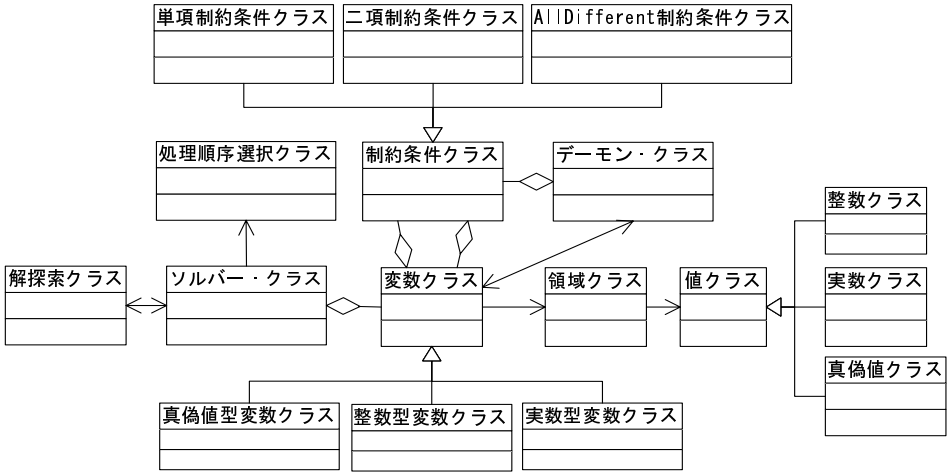

図 1　クラス図

3.1　問題定義機能

問題定義機能は変数定義機能，制約条件定義機能の２つの中機能から構成する．変数定義機能は，制約充足問題における変数と，領域，値をプログラム上に表現する機能であり，変数クラス，領域クラス，値クラスで実装する．制約条件定義機能は，制約充足問題における制約条件を定義する機能であり，基底クラスである制約条件クラスとその派生クラス，単項条件クラス，二項条件クラス，AllDifferent 制約条件クラスで実装する．AllDifferent 制約条件は，各変数に割り当てられる値がお互いすべて異なることを要求する制約条件である．なお，制約条件クラスを派生すれば，利用者独自の制約条件を定義できる．

3.2 解探索機能

解探索機能は，制約充足問題の解を導出する機能であり，制約伝播機能，処理順序選択機能，探索制御機能の３つの中機能から構成する．本機能を実装するクラスはソルバー (solver)・クラス，デーモン（daemon）・クラス，処理順序選択クラス，解探索クラスである．制約伝播機能は探索空間を絞り込む機能であり，変数間で整合する可能性のない値の組み合わせを予め排除する機能である．処理順序選択機能は，解探索時に次に処理する変数，または次に処理する値を選択する機能である．探索制御機能は，処理順序選択機能で選択された処理対象変数，及び値に対して制約条件を充たす解を探索する機能である．

4 本ライブラリを用いた *n*-クイーン問題の表現

最後に，本ライブラリを用いてサンプル問題を表現する．プログラミング言語として，Visual Basic.NET を使用した．

■変数の生成 クイーンの個数をサイズとする配列を生成する．

Dim *VarArray*(aQueenCount −1) As CIntegerVariable

CIntegerVariable は整数型変数のクラスである．ここで配列のインデックスを盤上の列番号，変数に割り当てられる値を盤上の行番号とする．従って，配列のインデックスはすべて異なるから，変数に割り当てられる値がすべて異なれば，クイーンが縦横の利き筋に置かれていないということになる．

■領域を設定する 整数型変数の領域を設定するためのメソッドはいくつかあるが，ここで用いたのは領域の上限値と下限値を指定すれば，その間公差 1 の領域を生成するメソッドである．

VarArray(*i*).setDomain(0, aQueenCount−1)

■制約条件を生成する 前述の縦横の利き筋にクイーンを置かない，という制約を表現するために AllDifferent 制約条件クラス（*CAllDifferent*）を使用する．2 行目は各変数を制約条件に追加する処理である．

Dim *CtAllDiff* As New CAllDifferent

CtAllDiff.add(*VarArray*)

また，対角線上の利き筋にクイーンを置かないという制約は二項差制約条件の派生クラス (*CIntSubtractNEConstraint*) を使用して次のように表現できる．*CCPSolver* クラスは探索制御クラスである．

Dim *MySolver* As New CCPSolver
Dim *varCt* As CIntSubtractNEConstraint

```
For j As Integer = VarArray.GetLowerBound(0) To VarArray.GetUpperBound(0)
    For k As Integer = (j + 1) To VarArray.GetUpperBound(0)
        varCt = New CIntSubtractNEConstraint(VarArray(j), VarArray(k), k − j)
        MySolver.add(CType(varCt, CConstraint))
        varCt = New CIntSubtractNEConstraint(VarArray(j), VarArray(k), j − k)
        MySolver.add(CType(varCt, CConstraint))
    Next
Next
```

5　考察とまとめ

以上，制約プログラミング・ライブラリの機能設計，クラス設計を行った．また問題定義生成機能を開発し，サンプル問題をプログラムに表現する過程を示した．本ライブラリを使用しないでサンプル問題を解く場合，クイーンの位置のすべてのパターンを制約充足するかどうか虱潰しに調べる方法と，Wirth[4] が示すようなアルゴリズムを用いて解く方法がある．前者の場合はコストの問題があり，後者の場合はそのアルゴリズムを他の問題に利用しづらいという問題がある．これらに比べて本ライブラリを用いる場合，本ライブラリにて実装済みのアルゴリズムを使用するのであれば，利用者は制約充足問題をプログラム上に表現しさえすれば，解探索は本ライブラリが行ってくれる，という利点がある．利用者独自のアルゴリズムを使用したい場合でも，本ライブラリの基底クラスを派生することにより追加が可能である．そればかりか，そのアルゴリズムを他の問題に再利用することもできる．この拡張性という点で，本ライブラリは JCL に対して利点がある．本ライブラリは上記のようにアルゴリズムの拡張のみならず，制約条件，領域，値の拡張も可能であるからである．また，階層的に設計した結果，JCL に比べて構造が分かりやすくなったということも利点である．

6　今後の課題

現時点で，設計した機能のうち問題定義機能を開発した．今後の課題は，解探索機能を含む本ライブラリの完成と，スケジューリング問題等の実際問題への応用である．

参考文献

[1] http://liawww.epfl.ch/~torrens/JCL/.

[2] 石塚満. 知識の表現と高速推論. 丸善, 1996.

[3] Pierre Roy, Anne Liret, and François Pachet. The framework approach for constraint satisfaction. *ACM Computing Surveys*, Vol. 32, No. 1es, pp. 13–16, 2000.

[4] Nikulaus Wirth. アルゴリズム＋データ構造＝プログラム. 日本コンピュータ協会, 1979. 片山卓也訳.