制約指向フロアプランニングシステムの設計

3H−8

大和田勇人 *　本多一賀 †　溝口文雄 *

*東京理科大学理工学部　†東京ガス（株）

1 はじめに

制約論理プログラミングは変数間に成り立つ制約の集合を記述することでプログラムを構成するもので，通常のプログラミング言語における代入，手続き，制御構造等の概念を持たない．プログラムは，言語処理系内部の制約充足システムによって実行され，変数に対する無矛盾な値の割り当てが実行結果として返ってくる [1]．

本論文では，以上のような特徴を持つ制約論理プログラミングを住宅間取りを対象としたフロアプランニングに応用することを試み，特に部屋の自動レイアウトを行う．ここでは，与えられた部屋の位置的制約（例えば，キッチンの隣には台所がある）を充足するように間取りを決める方法を述べる．

2 問題記述

問題は以下のように与えられる．

- 部屋は長方形である．
- 各部屋には大きさの下限が設定される．
- 部屋数は固定されている．
- 部屋の間の隣接関係が部分的に与えられる．

3 空間表現

上の問題を制約論理プログラミングに基づいて記述する．シンタックスは以下のようである．

$$H \leftarrow c_1, \ldots, c_n, B_1, \ldots, B_m$$

ここに，H, B_iはアトム，c_jは制約である．ここでは，制約として線形方程式，不等式を取り上げ，変数はエルブランもしくは実数領域上の値をとるとする．部屋を次のように表現する．

$$[North, South, West, East]$$

ここで，各変数は最終的には実数値に具体化される．また，フロア全体の大きさは次のように与えられる．

$$[0, h*, 0, w*]$$

ここで，$h*$はフロアの南北方向の長さ，$w*$は東西方向の長さを表す．

3.1 部屋の大きさの下限

部屋の大きさの下限を表す述語は以下のように定義できる．

$$area([N, S, W, E], Width, Height) \leftarrow$$
$$S - N \geq Height,\ E - W \geq Width,$$
$$0 \leq N,\ S \leq h*,\ 0 \leq W,\ E \leq w*.$$

ここに，変数 $Width, Height$ には実数値が割り当てられる．

3.2 部屋の方位

部屋の方位は，部屋が東西南北いずれかの面に接することを表す．それは以下のように記述できる．

```
north([0, _, _, _]).   % 部屋が北側にある
south([_, h*, _, _]).  % 部屋が南側にある
west([_, _, 0, _]).    % 部屋が西側にある
east([_, _, _, w*]).   % 部屋が東側にある
```

3.3 部屋どうしの隣接関係

隣接関係は選言制約集合で表すことができる．その定義は以下のようになる．

$$adjacent(R1, R2) \leftarrow west\_of(R1, R1).$$
$$adjacent(R1, R2) \leftarrow east\_of(R1, R1).$$
$$adjacent(R1, R2) \leftarrow south\_of(R1, R1).$$
$$adjacent(R1, R2) \leftarrow north\_of(R1, R1).$$

A Constraint-oriented Floor Planning System

Hayato OHWADA　Kazuyoshi HONDA　Fumio MIZOGUCHI
Science Univ. of Tokyo　Tokyo Gas　Science Univ. of Tokyo

この中で，例えば $west\_of(R1, R2)$ は以下のように定義されている．

$$west\_of([N1, \_, W1, E1], [\_, S2, W2, E2]) \leftarrow$$
$$N1 = S2 + 1,\ overlap(W1, E1, W2, E2).$$

$$overlap(A, B, C, D) \leftarrow A \leq C,\ B \geq C.$$
$$overlap(A, B, C, D) \leftarrow C \leq A,\ D \geq A.$$

3.4 部屋どうしの重なり

“各部屋は重ならない” という制約は以下のように表される．

$$disjoint([N1, S1, \_, \_], [N2, S2, \_, \_]) \leftarrow$$
$$non\_overlap(N1, S1, N2, S2).$$
$$disjoint([\_, \_, W1, E1], [\_, \_, W2, E2]) \leftarrow$$
$$non\_overlap(W1, E1, W2, E2).$$

$$non\_overlap(A, B, C, D) \leftarrow D \leq A - 1.$$
$$non\_overlap(A, B, C, D) \leftarrow B \leq C - 1.$$

4 プラン生成

プラン生成の手順は，制約充足を行い，次に部屋の位置を決める．

4.1 制約充足

制約集合は，例えば以下のように定義される．

$$plan([Dining, Kitchen, Bath, \ldots]) \leftarrow$$
$$area(Dining, 12, 10),$$
$$area(Kitchen, 8, 10), \ldots,$$
$$north(Dining),$$
$$west(Kitchen), \ldots,$$
$$adjacent(Dining, Kitchen), \ldots,$$
$$disjoint(Dining, Bath), \ldots.$$

ここで，問合せ

$$?- plan(X).$$

を与えると，すべての制約を充足する各部屋の配置可能領域が解として返ってくる．

4.2 選好解の生成

部屋の位置を決定するには，新たな制約を追加しなければならない．例えば，部屋をできるだけ広くするような制約を与えるには，次のようにプログラムする．

$$prefer([N, S, W, E]) \leftarrow$$
$$inf(N, N1),\ N = N1,\ sup(S, S1),\ S = S1,$$
$$inf(W, W1),\ W = W1,\ sup(E, E1),\ E = E1.$$

ここで，$inf/2, sup/2$ はそれぞれ変数の下限と上限を求める述語である．

5 論理的推論による効率化

論理的に充足可能であることが明らかな制約を制約充足の前に除くことにより，効率化を計ることができる．例えば，論理的関係

$$disjoint(R1, R2) \leftarrow adjacent(R1, R2).$$

より，$adjacent(R1, R2)$ が成立するときは $disjoint(R1, R2)$ に関する制約充足は行う必要がない．これは，論理式が充足可能であることが判明したなら，その情報をメモしておくことで実現できる．

さらに，制約充足の前に部屋間の位置関係として考えられ得るものをあらかじめ列挙しておくことも重要である．例えば，フロアサイズの制限から東西方向に並べられる部屋の組合せを求めておく．これを，西から順に $R1, R2, \ldots, Rn$ とおく．このとき，論理的関係

$$disjoint(R1, Rn) \leftarrow$$
$$west\_of(R1, R2),\ west\_of(R2, R3),$$
$$\ldots,\ west\_of(Rn - 1, Rn).$$

より，$R1$ と Rn は重ならないことは明らかである．本システムでは，このように部屋の定性的関係をまず求め，その後制約充足を行っている．

6 おわりに

本論文では，制約論理プログラミングに基づいてフロアプランニングシステムの設計を試みた．制約論理プログラミングによると，空間的関係の維持や変更管理は制約の追加，削除によって容易に行うことができる．さらに，空間的関係を定性的に扱うことにより，論理的推論と数理的処理を融合することも可能になる．これにより，制約充足を効率よく実現することができた．

参考文献

[1] Jaffar, J. and Lassez, J-L.: Constraint Logic Programming, *Proc. of the 14th ACM Principles of Programming Languages Conference*, 1987.