記 号 処 理 52－4
（1989. 9. 29）

グラフィカルユーザインタフェースのモデル化に関する一検討

高田 久靖
NTT ヒューマンインタフェース研究所

オブジェクト指向を用いたグラフィカルユーザインタフェースの研究が盛んである。本論文では、オブジェクトモデルに制約を導入した グラフィカルユーザーインターフェースモデルを提案する。 本モデルの特徴は、オブジェクト間の関係を制約の形で自然に表現できることの他、宣言的にかつ双方向に制約を記述可能なこと、グラフィカルオブジェクト間の幾何学的関係を代数方程式をもとにして表現可能であること、数値・文字列などのデータオブジェクト間の制約関係も導入したことである。

A Study on a Graphical User Interface Model

Hisayasu Takada
NTT Human Interface Laboratories
1-2356, Take, Yokosuka-shi, Kanagawa, 238-03, Japan

Graphical user interface based on object-oriented model is a current popular research theme. I present a graphical user interface model based on constraints with objects. Users can describe geometric and other kinds of object relations declaratively with constraints. Constraints based on numerical or text data are supported.

1 はじめに

高帯域な人間ー機械間の情報伝達を目指して、マルチメディア、特にグラフィックを用いたユーザインタフェースの実現が盛んである。グラフィカルユーザインタフェースにおいては、人間の直感に訴え、理解度を高めるために、現実世界の事物のシミュレーションを通して、人間ー機械間の情報授受が行われる。

複数のメディアを同時に扱う場合、これらを統一的に扱うことが望ましい。このためには、オブジェクト指向 [1] を用いることが良いと思われる。なぜなら、メディアの差による細部の意味の相違は全てオブジェクト内部に隠され、オブジェクトに対する操作の意味を統一できるためである。

オブジェクト指向言語 / モデルを用いて、既に多くのグラフィカルユーザインタフェース作成パッケージが開発されている [2] が、これらの殆どはツールキットと呼ばれる少数のユーザインタフェース部品のためのライブラリである。ユーザの要求に特化されたユーザインタフェースはオブジェクト指向言語または一般のプログラミング言語で記述するしかない。このためには、グラフィカルユーザインタフェースの基本モデルが必要である。このモデルはグラフィカルユーザインタフェースの統一化および個別のユーザインタフェースの評価のためにも使用される。

しかし、オブジェクトモデルでは現実世界の事物のシミュレーションに十分に対応しきれない。これは事物が必ずしも独立した機能を有した物では無く、相互の関係に依存して機能し、かつそれらの関係が複雑で手続き的に記述するのが難しいためである。

一方、事物の間の関係を主に記述することを目指した制約指向の考え方が有る [3]。制約指向では、制約対象および制約関係の種類を一般に制限することにより、高い記述力を提供している。この制約をオブジェクト指向言語 / モデルと融合することで両者の長所を有する言語 / モデルも提案されている [4][5][6][7]。

本稿で、筆者は事物に対応するオブジェクト間に制約を持ち込んだグラフィカルユーザインタフェースモデルを提案する。本モデルの特徴は、オブジェクト間の関係を制約の形で自然に表現できることの他、宣言的にかつ双方向に制約を記述可能なこと、グラフィカルオブジェクト間の幾何学的関係を代数方程式をもとにして表現可能であること、数値・文字列などのデータオブジェクト間の制約関係も導入したことである。

2章ではモデルの概要とその特徴を述べ、3章ではモデル中に現れる制約の表現方法に付いて述べ、4章では例を、5章では他のモデルとの比較分析を行う。6章では、本モデルをもとにしたオブジェクト指向言語上での制約機能と、その実現イメージについての検討結果を述べる。

2 グラフィカルユーザインタフェースモデル

2.1 背景

オブジェクトモデルにおいては、オブジェクトはシステムの構成要素であり、独立性が強く、その間をメッセージにより疎に結合される。図形やウインドウのようなグラフィック要素は複雑な構造を有し、まとまって扱われることも多いため、オブジェクトとして表現すると便利である。本稿では、このようなオブジェクトをグラフィカルオブジェクトと呼ぶ。グラフィカルオブジェクトは、しかし他方では、相互に緊密な関係を持つことが有る。例えば、直線間の結線関係では各直線はそれぞれ独立しているにも関わらず、強く相互に束縛されている。これら相互の関係が固定的な物であれば、全体を一つのオブジェクトして表現すれば良い。しかし、高度に対話的にグラフィック環境においては、グラフィカルオブジェクトで表現される対象間の関係を非常に動的に変化させることが良く見受けられる。良い例がLSIのCADシステムである。

オブジェクトシステムでこのような動的な関係を表現する方法は、関係を表現するような複合オブジェクトを作成し、これを介して相互に影響を及ぼしあうことである。しかし、この方法は動的に制約を設定・変更していくような場合には負荷が大きく、また実体ではなく関係を表現する余分なオブジェクトを導入することになり、混乱のもとである。

このようなオブジェクト間の関係を記述するために制約を導入する。制約とはシステムにより維持される関係である。これにより、あるオブジェクトの状態が変動することで、制約関係にある他のオブジェクトの状態が変更される。しかし、制約のみで全てを記述することを目指すものでは無い。オブジェクトと制約は

相互に補完関係にあると見なす。

また、グラフィカルユーザインタフェースのモデル化においては、グラフィカルオブジェクトとは別に、それらの表示/データ入力を司るオブジェクトをも導入する。但し、グラフィカルオブジェクトとこれらの間の関係はかなり緊密なので、相互に制約が存在するものと考える。

2.2 モデル

ユーザインタフェースモデルは、部品オブジェクト、入力オブジェクト、ビューオブジェクトおよびそれらの間の制約から構成される(図1参照)。

部品オブジェクトは、表現しようとしている問題のモデルの部品を表す。上記でグラフィカルオブジェクトと述べていたものは、これに相当する。ビューオブジェクトは部品オブジェクトに従い、現実に表示操作を行うものであり、ディスプレーやウィンドウに対応する。入力オブジェクトは部品オブジェクトに対する入力機能を表す。制約はこれら3種のオブジェクトの間に成立する。具体的には、以下のものが存在する。

まず、部品オブジェクトの入出力に関する制約がある。これは、オブジェクトの内部状態とその表示とは常に等しいと言う制約と、入力を受け付ける各部品オブジェクトは専用の入力オブジェクトを有し、前者の状態は後者の状態(即ち、入力値)と常に等しいという制約である。

更に、部品オブジェクト相互間の制約がある。これにはオブジェクトが表現している問題のモデル内部における制約と、グラフィック属性に関する制約が存在する。

本ユーザインタフェースモデルは、以下のように形式的に表現できる。

$$(M, V, I, C_{cv}, C_{cc}, C_{ci})$$

但し、

M: 部品オブジェクトの集合

V: ビューオブジェクト

I: 入力オブジェクトの集合

C_{cv}: 部品オブジェクト・ビューオブジェクト間の制約の集合

C_{cc}: 部品オブジェクト同士の制約の集合

C_{ci}: 部品オブジェクト・入力オブジェクト間の制約の集合

制約はモデルの仕様により設定される他、ユーザが動的に指定/解除できるものが考えられる。すなわち、制約設定を宣言的プログラミングとして独立に扱うことを示唆する。このような制約プログラミングは、APのカスタマイズやAPの結合を可能にする。

2.3 入出力に関わるオブジェクト間の制約

ビューオブジェクトと部品オブジェクトとの間には、部品オブジェクトの状態とそれに対応する表示(ビューオブジェクトの状態)は等しいという制約関係(出力制約)である。これは、あるビュー内部に表示される部品オブジェクト群とビューオブジェクトの間に、N対1に存在する制約である。制約は部品オブジェクトからビューオブジェクトへ伝播する。

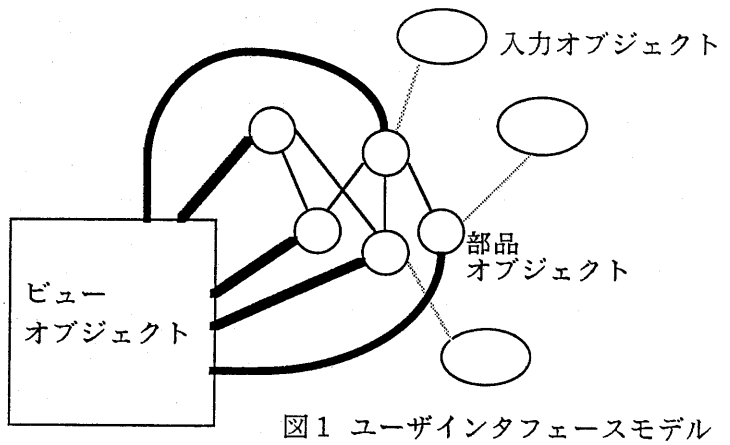

図1 ユーザインタフェースモデル

また、入力オブジェクトと部品オブジェクトの間には、前者からの入力値が後者の状態に等しいという制約(入力制約)が存在する。これは、部品オブジェクトとそれに1対Nに対応する入力オブジェクトとの間に存在し、伝播方向は双方向である。

これにより、部品オブジェクトの状態の変化がビューオブジェクトの状態を変化させ、(エンドユーザ操作による)入力オブジェクトの状態の変化が部品オブジェクトの状態を変化させることになる。

すなわち、入力と出力の詳細は各々、入力オブジェクトとビューオブジェクトに隠される。入出力(入力オブジェクトとビューオブジェクト)と部品オブジェクトとの間に本質的な関係、すなわち意味的な等価性のみを指定すれば良い。

2.4 部品オブジェクト間の制約

部品オブジェクトの多くはグラフィック属性を基本的に有している。個々のオブジェクトを特徴付ける属性はオブジェクト内部で実現されるが、オブジェクト相互間にまたがる幾何学的関係(e.g., 上下関係、平行性)は制約で表現される。これを幾何的制約と呼ぶ。

これとは別に、部品オブジェクト間には表現しようとしている問題のモデルに含まれている制約がある。この種の制約は、オブジェクトに依存した形をとる。これを記述するための一般的記述方法はここでは考えない。しかし、次節で述べる理由により、基本的なデータ型に依存した幾つかの原始的な制約とその組合せを考え、この種の制約の記述にも用いる。

3 制約関係の表現方法

3.1 方針

制約充足とは、あるオブジェクトの状態が変動することで制約関係にある他のオブジェクトの状態が、関係に従って変更されることである。これは、状態が変動したことを制約関係にある他のオブジェクトへ伝達すること(制約伝播)と、変動に従って他のオブジェクトの状態を変更(状態値再計算)することの二段階からなる。この二つの処理をある制約記述に従って行うのが制約解消系である。

モデルに従い、オブジェクト間の関係を極力、制約のみで記述する場合、それに適合した制約解消系と制約記述法が必要になる。しかし制約解消系(特に、幾何的制約)を、対象であるグラフィカルオブジェクトの実現法に強く依存して作成することは一般性を損なう。

そこで、より原始的的なシステム組み込みの制約解消系を使用し、高次の制約を記述できるような制約記述法を目指す。このために、幾つかの原始的な制約解消系の導入および状態値再計算手順の定義を可能にする。

組み込み制約解消系として、数値データ間の代数等式制約の他、文字列データ間の連結関係をもとにした文法的制約を採用する。これより、幾何的制約はグラフィカルオブジェクトの座標値間の代数等式制約によって、入出力制約は扱うデータの型毎に代数等式制約(数値(座標値も含む))と文法的制約(文字列データ)によって表現される。

組み込み制約では制約伝播と状態値再計算が解の導出という形で混然と行われるが、それでは対応しきれない状態値再計算手順は手続きとして別にユーザ定義させる。これは一種のデーモンである。具体的には、制約関係にあるオブジェクト(およびインスタンス変数)を宣言した後、再計算対象オブジェクト毎に計算手順を手続きの形で書き表す。

3.2 制約解消系の特徴

制約伝播を引き起こすのは、制約関係にあるオブジェクトの状態の変動、より具体的には、ユーザ入力やプログラム実行により引き起こされたオブジェクト状態の確定(変更を含む)である。これに対し、未確定の状態が制約により決定される。しかし、確定されたオブジェクトの状態同士が制約に反することが有り得る。この様な矛盾状態に陥った場合、システムは矛盾発生要因となった状態変更をエラーとしてはじくなどの例外処理を行う必要が有る。

また制約関係を成すオブジェクト(の状態)には、

1. 自らの状態の変更により制約伝播を引き起こすが、他で引き起こされた制約伝播に対して再計算対象にならないもの; (例) レシートの(合計価格に対する)単品価格
2. 自らの状態の変更により制約伝播は引き起こさず、他で引き起こされた制約伝播に対して再計算対象になるだけのもの; (例) レシートの(単品価格に対する)合計価格

3. 自らの状態の変更により制約伝播を引き起こし、かつ他で引き起こされた制約伝播に対して再計算対象になるもの;

の3種類が存在する。入力オブジェクトの状態は1. または3.、ビューオブジェクトの状態は2. に対応する。

制約伝播および状態値再計算は、再計算対象にならないまたは制約伝播契機になった変数をその時点での値に置換した制約式により行い、残りのオブジェクトの新しい状態を求める。

3.3 組み込み制約解消系

1. 代数等式制約

   代数等式制約は、制約関係にあるオブジェクトの状態(数値)を変数とする代数方程式によって、規定される。スプレッドシートのセル(オブジェクト)間の関係が例として上げられる。それに使用される原始的な演算子は四則演算子である。

   代数等式制約においては、変化した(オブジェクトの状態を表す)変数を含む制約式中の他変数で表されたオブジェクトに制約が伝播する。これに伴う状態値再計算は、当該制約式を方程式として解くことにより行われる。なお、制約を記述する等式が複数個存在する場合は、制約式群を連立方程式と見なして解く。制約伝播は双方向に行われる。

   また、計算手順の宣言のため、演算子を定義できる事が望ましい。これらの演算子は、引数の型による処理の相違を隠蔽できるオブジェクト指向の世界での実現が容易であると考えられる。

2. 文法的制約

   グラフィカルユーザインタフェースにおいても、テキストの表示/入力は必須である。特に、入力されたテキストは分析加工され、種々の用途に使用される。ここで、テキストを合成する際の各テキストの連結関係を制約として捉える。例えば、エンドユーザ設定可能な環境情報(ユーザ名、所属名など)は一旦変更されると、それを含む多くのメッセージは変更される。

   そこで、テキスト情報間の制約の記述機能を導入する。テキスト間の連結関係も等式制約的に記述し、制約評価の過程でテキスト情報の分解合成を行わせる。制約式は、左辺は文字列変数または文字列定数一個、右辺は文字列変数及び定数の連結からなる。等号は左辺の文字列を右辺の変数への分解または右辺の連結文字列から左辺の変数への合成を促す。この時、文字列の連結を '+' 演算子を用いて表現する。

   (例1) `"BEFORE-AFTER" = VARforBEFORE + "-" + VARforAFTER`

   (例2) `VARforEXAMPLE = "EXAM" + "PLE"`

この他、局所的に宣言された制約に属する全てのオブジェクトへ制約伝播が起き、各オブジェクトの状態値を再計算する場合もある。

4 例

本稿で提案したモデルを適用した例題を考える。例題は平面上の3点A,B,Cの座標からその重心を求めるものである(図2参照)。図2のように、3点A,B,CのXおよびY座標値を表の形で数値表示すると共に、各点を図示する。各点は表のセル内でX/Y座標値をキー入力するか、または図上の点を(マウスなどで)直接動かす事により、位置を変える事ができる。
これらの間の主な制約関係は以下のようになる。

1. 表内の座標データと図上の3点の座標値は等しい。
2. 重心点は任意の時点の3点A,B,Cの座標値から一意に求まる。

各点を表す3個の部品オブジェクト(点オブジェクト)A,B,Cに対して、上記の制約1は、表内の各XおよびY 座標値用セル/図上の各点に対応する入力オブジェクトと各点オブジェクトとの間の入力制約と、ビューオブジェクトと各点オブジェクトとの間の出力制約の形で表現できる。更に制約2は、重心の部品オブジェクトと3個の点オブジェクトとの間の制約(重心と各点の座標値間の代数等式制約)の形で表現できる。これらは図2に示す通りである。

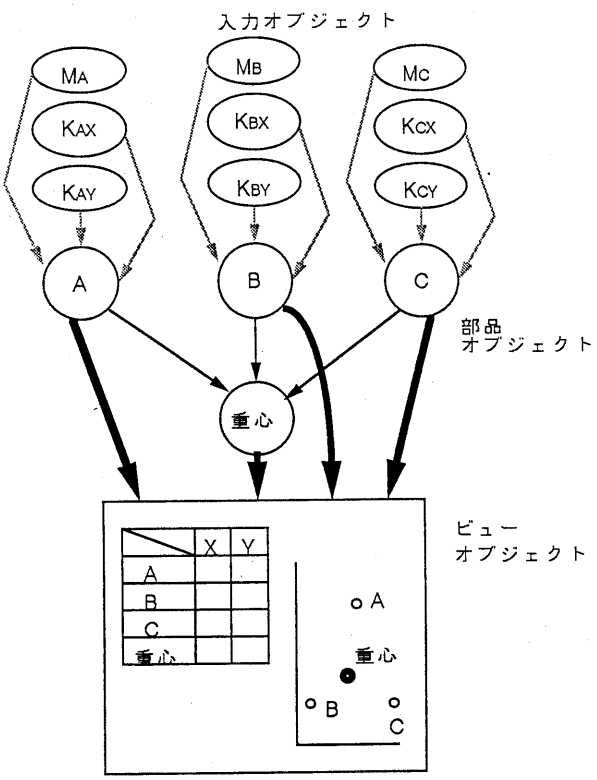

図２　例題

KAX/Y,KBX/Y,KCX/Y：キーからの点A,B,Cの座標X/Y入力

MA,MB,MC：マウスからの点A,B,Cの座標入力

⟶：制約伝播方向を示す。

5　既存モデルとの比較

グラフィカルユーザインタフェースのモデルとしては、Smalltalk ウィンドウシステムのモデルである MVC(Model-View-Controller) パラダイム [1] が有名である。

MVC パラダイムでは、グラフィカルユーザインタフェースは三種類のオブジェクト (モデルオブジェクト、ビューオブジェクト、コントローラオブジェクト) により実現される。モデルオブジェクトはグラフィックの意味的実体、ビューオブジェクトは意味的実体の表示を司る。モデルオブジェクトはビューオブジェクトを制御し、グラフィックを表示する。コントローラオブジェクトは入力を司り、モデルオブジェクトの状態を変更する。しかし、モデルおよびビューオブジェクトはそもそも同じものの二面を表現するものであり、コントローラオブジェクトをその間を橋渡しする手続きを表すオブジェクトである。

MVC パラダイムにおいては、以下の問題点がある。まず、モデルオブジェクトはあくまでビューの内部に表示される物の全体を表現するものであり、モデル内部の細かい部品およびそれらの間の関係を扱う事には向かない。また、本来は同一物体の意味とその表現である筈のモデルおよびビューオブジェクト相互の関係が手続き的にしか記述できず、繁雑である。

これに対し、本稿で提案したモデルではオブジェクト間を制約で宣言的に関係づけることにより、オブジェクト間の手続き的なやり取りを少なくできる。また制約の宣言的性質により、オブジェクト間の関係は直感的に理解し易く、誤りにくい。しかし、オブジェクト間の制約関係は宣言的に、すなわち静的に記述することが難しいものもあるので、手続き的な記述を許す事も必要である。

6　計算機言語への制約の導入

本稿で述べたモデルを素直に表現するためには、言語側にも制約を表現する機能を組み込む事が要請される。本節では、言語に組み込まれるべき制約機能およびその実現イメージを述べる。オブジェクト指向言語 CLOS を前提に制約機能を考える。

6.1　オブジェクト指向言語への制約の導入

実現に当たって、オブジェクトと制約を互いに全く別な概念としてではなく、できるだけ融合した形で実現する事を方針にする。これは、制約を手続き的に記述することを許すなど、相互の利点を引き出して使用できるようにするためである。また、ベースはオブジェクト指向とする。

まず、制約は動的に定義可能にする。すなわち、複数の制約がどの順で定義されるかは定まっていない。制約を言語機能の一部として実現する場合、制約で束縛されるオブジェクトの状態はインスタンス変数 (CLOS ではスロット) である、と考える。

次に、具体的な制約の種類として、代数等式制約、幾何的制約、文法的制約機能を考える。

1. 代数等式制約

   四則の連立線形方程式を扱う事にする。以下の例は、直線の両端座標と中点座標との間の制約を定

義している。

```
(def-constraint points (((x y) EndPoint1)
((x y) EndPoint2) ((x y) MiddlePoint))
((x MiddlePoint) * 2 =
(x EndPoint1) + (x EndPoint2))
((y MiddlePoint) * 2 =
(y EndPoint1) + (y EndPoint2)))
```

2. 幾何的制約

   グラフィカルオブジェクトの座標値間の代数等式制約によって表現する。このような原始的な形式で表現された制約を、関係子で代表させる事にする。すなわち、ユーザ定義された制約を表す関係子を定義する機能を用意する。以下の例は二直線の平行性を示している。

```
(def-constraint parallel (Line1 Line2)
(Line1 // Line2))
```

3. 文法的制約

   右辺の文字列の('+'演算子による)連結と左辺の文字列とが等しいことを、両辺を'='で結んで表現する。

```
(def-constraint parse ((value Filename)
(value Devicename) (value Pathname))
((value Filename) = (value Devicename)
+ ":" + (value Pathname)))
```

手続きによる状態値再計算定義の例を以下に示す。これはオブジェクトA(のインスタンス変数a)とオブジェクトB(のインスタンス変数b)との制約関係および、AとBでの状態値再計算手順の定義を示している(この例では、aとbは互いに契機になって無条件に他方に制約を伝播し、状態再計算を促す)。

```
(def-constraint ex ((a A) (b B)))
(def-constraint-satisfaction (a A) (..))
(def-constraint-satisfaction (b B) (..))
```

6.2 実現イメージ

全ての物をオブジェクトで表す事を要請するオブジェクト指向の考え方から制約自体もオブジェクトで実現する。しかし、素直に制約を(クラスのインスタンスとしての)オブジェクトで表現すると、個々の制約が独立のオブジェクトと化してしまうため、解こうとする問題に現れないオブジェクトが出現し、混乱し易い(図3参照)。

図3では、青、黄および赤信号オブジェクトの他、それらの間の制約を表現する制約オブジェクトが導入されている。ユーザが意図した信号オブジェクト配置の中に、意図されていない制約オブジェクトが現れ、ユーザがその存在を意識せざるをえなくなる事は望ましくない。また、信号間の制約も信号の属性と考えられるため、それが信号オブジェクトの外の別オブジェクトの形で実現されるのも好ましくない。

ここでは、クラスによる制約実現を考える(図4参照)。制約定義時に、制約対象になっているオブジェクトのクラスを新しいものに変更する。そして、新しいクラスは元のクラスのサブクラスであり、かつ制約を表現するクラスを抽象スーパークラスとするようなクラスとする。制約伝播及び状態値再計算はこの新しいクラスのメソッドとして実現する。すなわち、制約関係の充足は、関係を有するオブジェクト間でのメッセージの交換により行われる。但し、このメッセージ交換の制御はユーザ定義クラスのユーザ定義メソッドからは直接的には行われない。ユーザ定義メッセージの交換の結果、制約関係にあるオブジェクトの状態(インスタンス変数の値)が変更された場合に、代入機構の裏側で制約伝播及び状態値再計算を促すメッセージ交換が行われるものとする。

制約伝播機構の実現方法について述べる。

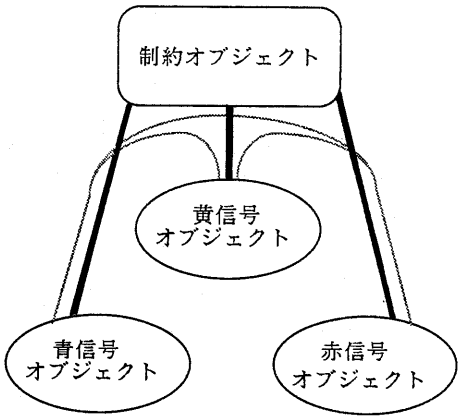

図3 オブジェクトとしての制約
(—線は制約関係を表し、制約オブジェクトを介して実現される)

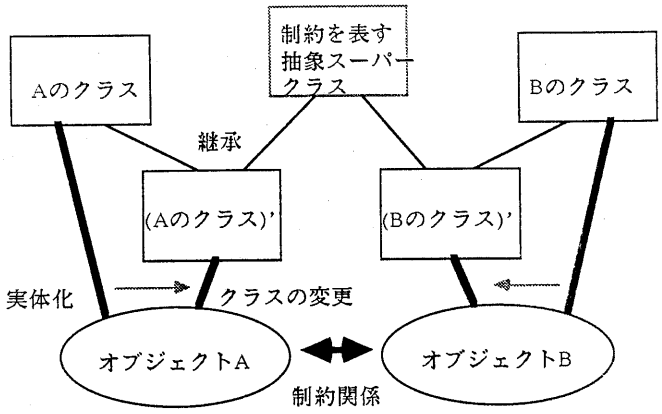

図4 抽象スーパークラスとしての制約

代数等式制約では、基本的に制約伝播の契機となったインスタンス変数を含む (連立) 方程式を状態未定義インスタンス変数について解くことが制約伝播になる。求められたインスタンス変数値が制約伝播前と異なっており、当該変数について定義された状態値再計算手続きがあれば、それが起動される。方程式が矛盾する場合、すなわち確定された値を用いて方程式の等号が成立しない場合、制約伝播契機になったインスタンス変数の値の確定は取り消される。

幾何的制約は、グラフィカルオブジェクトの座標データ間の代数等式制約に変換され、代数等式制約解消系が起動される。

文法的制約での制約伝播の概略アルゴリズムを以下に示す。

1. 制約伝播の契機になったインスタンス変数を含む制約式を集め、未定義変数の個数の昇順に制約式をソートする。

2. ソートされた制約式を順番に解いていく。

上記の実現方法においては、言語側にクラスの再定義機能が必要になり、また解くべき問題のモデルには現れないクラスが出現する。しかし、(クラスのインスタンスとしての) オブジェクトの形での実現方法に比べて、オブジェクトの情報隠蔽特性から、制約に関する情報 / メソッドはユーザ定義情報 / メソッドに紛れてしまい、このクラスの存在は殆ど意識されない。使用するオブジェクトの数も少なくなり、また同種類の制約を使用する場合に制約を表すクラスを流用できる。更に、制約伝播および状態値再計算を行う際に送られるメッセージ数も少なくなる。その他、制約を分散管理できる利点もある。

7 おわりに

本稿では、制約を導入する事により、アプリケーションレベルのモデルをオブジェクトを用いて実現する上で、オブジェクトの意味と表示の関係およびオブジェクト同士の関係を自然に記述できるようなグラフィカルユーザインタフェースモデルについて述べた。また、本モデルに基づいてアプリケーションを記述するために必要な言語機能とその実現方式についての検討結果も合わせて述べた。

現在、本稿で述べた制約言語機能を CLOS の上で実現中である。今後は、本モデルに基づいたグラフィカルユーザインタフェース管理システムを試作する予定である。

[参考文献]

[1]A.Goldberg, et al. : "Smalltalk-80:The Language and its Implementation",Addition-Wesley, 1983

[2]The X Toolkit Documents, M.I.T, 1988

[3]G.L.Steele.Jr, G.J.Sussman: "CONSTRAINTS", APL Conference Proceeding Part 1, APL Quote Quad, 208-225(1979).

[4]A.Borning, et al. : "Constraint Hierarchies", OOPSLA '87 Proceedings, 48-60(1987)

[5]M.Grosman, R.K.Ege: "Logical composition of Object-Oriented Interfaces", OOPSLA '87 Proceedings, 295-306(1987)

[6]P.A.Szekely, B.A.Myers: "A User Interface Toolkit Based on Graphical Objects and Constraint", OOPSLA '88 Proceedings, 36-45(198

[7]中島: "制約伝播機構を内蔵するオブジェクト指向言語 : COOL", 情報処理学会論文誌, Vol.30, No.1, 101-107(Jan.1989)