ICOT Technical Memorandum: TM-0579

TM-0579

意思決定支援システムにおける
モデル構築支援

平石邦彦(富士通)

July, 1988

ICOT

Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan

(03) 456-3191~5
Telex ICOT J32964

Institute for New Generation Computer Technology

意思決定支援システムにおけるモデル構築支援

富士通国際情報社会科学研究所　　平石　邦彦

1．はじめに

意思決定支援システムは人間が行う様々な問題解決を支援するシステムである．問題解決のための方法の1つに問題のモデル化がある．構築したモデルを解析することにより，モデル化された対象がもつ特質を推定することができる．ORや経営科学の分野ではモデル化による問題解決が主流であり，LP(線形計画法)，PERTなど様々な意思決定モデルが提案されてきた．しかし，現実の問題が既存のモデルに適用可能かどうかを判断し，その記述形式にしたがって問題をモデル化する過程は，人間が行う作業として残されている．

本研究では，問題のモデル化を支援する方法を提案する．この方法は，関係データモデルに手続き的な表現を取り入れることによりモデル化を行うものであり，spread sheet*1の概念の拡張と考えることもできる．モデルは統一された形式で記述されるため，モデルどうしの比較，複数のモデルの結合など，モデルの統合的な利用が容易に行えるようになる．

以下，2節ではモデルの記述形式について，3節ではインプリメンテーションについて述べる．

2．モデルの記述形式

2.1　モデルの概念

(1)　モデル(model)とモジュール(module)

意思決定支援システムは通常，様々な問題解決を支援するためのソフトウェア・モジュール―たとえば，線形計画，非線形計画，多変量解析，統計処理など―を備えている．ここでモジュールとは，文献1)と同様に，"単独で，あるいは，他のモジュールとともに，より広汎なモデルを構成するために使用されるモデル"であり，具体的には，"入出力が定義された手続きの集まり"として扱う．

一方モデルとは，現実対象の構成要素がもつ関係の表現であり，モジュールよりも広い概念として扱う．モデルの入出力は明確ではなく，モデルをどう使用するかで変化する．たとえば，数式で記述されたモデルにおいて，どの変数を決定変数にするか，目的関数をどのように設定するかなどはモデルの使用目的により変化し，さらに，それらの選び方により，線形計画法により解くか，非線形計画法により解くかなど，解析手段も異なってくる．モデルに対して入出力を定めることにより，そのモデルは1つのモジュールとして扱うことができる．逆に，モジュールは，より大きなモデルを構成する要素となる．

(2)　モデル管理システム(Model Management System)

モジュールを問題解決に利用するためには，問題をモジュールが利用可能な形式にモデル化－具体的には，入出力パラメータと実際の値との対応―を行う必要がある．このとき，モジュールの実行に必要な入力データの生成が困難，出力結果の解釈が難しい，などの問題点があり，これらを克服する手段としてモデル管理システムの重要性が指摘されている[2]．モデル管理システムは，構築したモデルをモデルベースに蓄積し，必要に応じてモデルを取り出し，問題解決に利用するシステムである．モジュールの入出力と現実対象との対応をモデルの形で蓄積することにより，モジュールをより容易に使用することができる．このとき，既存のモデルをそのまま利用するだけでなく，問題に合うように変更したり，部品となる複数のモデルの結合などの操作も必要になる．

(3)　モデルのモデル

モデル管理のためには，統一した形式でモデルが記述されていると都合がよい．この記述形式は，"モデルのモデル"として考えられる．このアプローチの研究には文献3)のStructured Modelingなどがある．

モデルには，モデルの構成要素がモデル化された対象の意味を保持しているような抽象度の低いモデル(これを対象の固有モデルとよぶ)から，数式で表現されるような抽象度の高いモデルまで様々なものが存在する．既存のモデルを利用するためには，数式で記述されたモデルの各変数・係数を現実の対象に対応させる，固有モデルから構成要素の関係を数式の形で抽出する，数式を行列の形に変換するなど，記述形式相互の変換が必要になる．

また，問題のモデル化には，まず大まかなモデルを作り，それを詳細化していくトップダウン的なアプローチ，逆に，対象を分割してサブモデルを作り，それらを統合していくボトムアップ的なアプローチなど，様々なアプローチが考えられる．

2.2では，これらの操作が柔軟に行なえる固有モデルの記述形式として，関係の内包的記述によるモデル化について述べる．

2.2　関係の内包的記述によるモデル化

データベース理論におけるデータモデル論では，実世界にある構造をもったデータを，どのように形式化(あるいは抽象化)して計算機内部の表現に移すかを論じているが[4]，本研究の目的は，データだけでなく，データが得られる過程，あるいは，どのような制約条

*1 Visicalc，Multiplanなどに代表される表計算言語

件を満たすデータが実世界に存在しうるのかをモデル化することにある．ここでは，関係データモデルを基本にしたモデル化を考える．その理由としては，表計算言語の普及に見られるように，表形式で表現されるデータが数多く扱われること，関係データベースとの結合が容易であること，数学的な取り扱いに適していることが挙げられる．

関係データベースでは，関係はすべて外延的に(すなわち，データの集合として)記述されるが，ここで提案するモデルでは，関係を内包的に(すなわち，データが満たすべき条件として)記述し，条件を満たす値の集合の外延を求めることを目的とする．この集合をモデルの解とよぶ．

関係Rの属性の集まりX，Yに対し，Xの任意の属性値に対してYの属性値がたかだか1個しかないとき，YはXに関数従属であるといいX→Yであらわす．関係Rのいかなる属性の集まりXに対してもつぎのことが成立するとき，Rは第3正規形であるという：“Xに含まれない属性がXに関数従属であるならば，Rのすべての属性がXに関数従属である”．モデルが表現する関係は第3正規形に分解されているものとする．すなわち，関係にある関数従属性は，キーと全属性間にある従属性ただ1種である．

関係Rに関数従属性X→Yがあるとき，Yは，XおよびXをキーとして得られる他の関係の関数として一意に決定される．この関数を陽に記述したものをモデルとして定義する．なお，以下の記述で，P(X)は集合Xのべき集合をあらわす．

モデルを4項組M＝(E，S，F，π^0)により表現する．

i) Eは型の集合である．型e∈Eに対し，値集合$\underline{e}$が存在する．型eと値vが与えられたとき，v∈$\underline{e}$かどうか（値vが型eをもつかどうか）は，明確に定まっているものとする．

ii) S＝{s_1，…，s_n}は関係スキーマ（以後，単にスキーマ）の集合である．各スキーマs_iは，Eに属する型を属性とする組〔x_1，…，x_n，y_1，…，y_m〕であらわされる．ここで，x_1，…，x_nはキー属性，y_1，…，y_mは非キー属性である．値集合の直積$\underline{x_1}\times\cdots\times\underline{x_n}\times\underline{y_1}\times\cdots\times\underline{y_m}$を$s_i$の定義域とよび$\underline{s_i}$であらわす．また，$P(\underline{s_1})\times\cdots\times P(\underline{s_n})$をSの定義域とよび，$\underline{S}$であらわす．

iii) Fは関数の集合である．スキーマs＝〔x_1，…，x_n，y_1，…，y_m〕（x_1，…，x_nはキー属性）に対し，その関数f_sはつぎの形で定義される．

$$f_s : \underline{x_1}\times\cdots\times\underline{x_n}\times P(\underline{s_{z1}})\times\cdots\times P(\underline{s_{zw}}) \rightarrow \underline{y_1}\times\cdots\times\underline{y_m} \quad (1)$$

ここで，s_{z1}，…，s_{zw}はSの他のスキーマであり，その集合をI_sであらわす．1つのスキーマに対して関数はたかだか1つしか定義されない．ただし，すべてのスキーマが関数をもつ必要はない．f_sの定義域を$dom(f_s)$であらわす．

iv) $\pi^0\in\underline{S}$は初期値である．関数が関係の内包的な記述を行うのに対し，初期値は関係の外延的な記述を行う．初期値には2つの意味がある．1つは変化しない値の記述，もう1つはモデルへの入力である．

$\pi\in\underline{S}$および$s\in S$に対し，π_sはπのP($\underline{s}$)への射影をあらわす．もしπ_sにキー属性の値が等しいタプルが複数存在しないならば，π_sは無矛盾であるという．すべてのs∈Sについてπ_sが無矛盾ならば，πは無矛盾であるという．

Mに対し，$\pi\in\underline{S}$がつぎのi)～iii)を満たすとき，πをMの解とよぶ．

i) πは無矛盾

ii) 任意のs∈Sについて，$\pi_s\supset\pi^0_s$

iii) $s=〔x_1,\cdots,x_n,y_1,\cdots,y_m〕\in S$に(1)式で表される関数$f_s$が存在するとき，以下が成り立つ．ただし，$dom(f_s,\pi)$は，$dom(f_s)$の$\pi$への制限である：

任意の$(a_1,\cdots,a_n,r_1,\cdots,r_w)\in dom(f_s,\pi)$について，$(b_1,\cdots,b_m)=f_s(a_1,\cdots,a_n,r_1,\cdots,r_w)$ならば，$(a_1,\cdots,a_n,b_1,\cdots,b_m)\in\pi_s$である．

モデルの解は，モデル内のデータに関数を適用していったときの不動点である．まず，関数の適用関数δ：F×$\underline{S}$→$\underline{S}$をつぎのように定義する：

s＝〔x_1，…，x_n，y_1，…，y_m〕∈Sおよび(1)式で表されるsの関数f_sについて，

$$\begin{cases}\delta(f_s,\pi)_s=\pi_s\cup\Delta\pi_s\\ \delta(f_s,\pi)_{s'}=\pi_{s'}\quad(s'\neq s)\end{cases}$$

$$\Delta\pi_s=\{(a_1,\cdots,a_n,b_1,\cdots,b_m)\mid {}^\exists r_1,\cdots,{}^\exists r_w:(a_1,\cdots,a_n,r_1,\cdots,r_w)\in dom(f_s,\pi)\ \wedge\ (b_1,\cdots,b_m)=f_s(a_1,\cdots,a_n,r_1,\cdots,r_w)\}$$

Mの到達値集合Π_Mをつぎのように定義する．

i) $\pi^0\in\Pi_M$

ii) $\pi\in\Pi_M$ならば，任意の$f_s\in F$について，

$\delta(f_s,\pi)\in\Pi_M$

任意の$f_s\in F$について$\delta(f_s,\pi)=\pi$であるような$\pi\in\Pi_M$をMの不動点とよぶ．無矛盾な不動点は，モデルの解になる．

つぎに，不動点の計算方法について考える．i)，ii)によりS上の2項関係＞を定義する．

i) 任意のs∈Sについて，$s'\in I_s\Rightarrow s>s'$

ii) ＞は，推移性を満たす．

任意のs，s'∈Sについて，$s>s'\Rightarrow not\ s'>s$が成り立つとき，Mは非循環(acyclic)であるという．モデルの解は一般には無限に存在するので，最小(極小)な解が意味をもつ．モデルが非循環なとき，つぎのアルゴリズムにより最小不動点が求められる．

［アルゴリズム1］：

1) S'を関数をもつスキーマの集合，$\pi:=\pi^0$とする．

2) S'＝φならば終わり．そうでなければ，S'で＞に関して極小なsを選び3)へ．

3) sの関数 f_sについて，$\pi := \delta(f_s, \pi)$とする．
S′:=S′－{s} として2)へ．

非循環なモデルに対しアルゴリズム1を実行することをモデルの評価とよぶ．モデルが非循環でないときは，一般に単純に計算を行うだけでは不動点は求められず，数理計画法など，他の制約解消系を用いる必要がある．

2.3 モデルの例

feedmixモデル[3]を取り上げる．これは，動物の飼料に栄養分をどのように混合させるかを表現したモデルで，栄養分としては，Protein，Calciumの2種類があり，それぞれに1日の必要最小量が決まっている．原料には，Standard Feed，Additive の2種類があり，それぞれに単価と使用量が与えられている．また，分析により，各栄養分の原料1単位当たりの混合量が示される．栄養分の必要最小量を満たし，かつ，総コストをできるだけ低く抑える原料の使用量を求めることが目的となる．

型としては，

nutr：栄養分，pounds：重量(ポンド)，material：原料，dollars：金額(ドル)，boolean：論理値

スキーマとしては（下線部がキー属性），

MIN = [nutr, pounds]（必要最小量）
UCOST = [material, dollars]（単価）
Q = [material, pounds]（使用量）
ANALYSIS = [nutr, material, pounds]（分析）
NLEVEL = [nutr, pounds]（栄養分のレベル）
T:NLEVEL = [nutr, boolean]（レベルの判断）
TOTCOST = [dollars]（総コスト）

関数としては，

$f_{TOTCOST}$: P(UCOST)×P(Q)→dollars
TOTCOST := {[v] | v = sum({u q | ∃m: [m, u] ∈ UCOST ∧ [m, q] ∈ Q})}
（sum(X)は集合Xの要素の和を求める関数）

f_{NLEVEL} : nutr×P(ANALYSIS)×P(Q)→pounds
NLEVEL := {[n, v] | v = sum({a q | ∃m: [n, m, a] ∈ ANALYSIS ∧ [m, q] ∈ Q})}

$f_{T:NLEVEL}$: nutr×P(MIN)×P(NLEVEL)→boolean
T:NLEVEL := {[n, v] | [n, x] ∈ MIN ∧ [n, y] ∈ NLEVEL ∧ ((y ≧ x ∧ v = true) ∨ (y < x ∧ v = false))}

初期値としては，

π^0_{MIN} = {[Protein, 16], [Calcium, 4]}
π^0_{UCOST} = {[Standard Feed, 1.2], [Additive, 3]}
π^0_Q = {[Standard Feed, 2], [Additive, 0.5]}
$\pi^0_{ANALYSIS}$ = {[Protein, Standard Feed, 4], [Protein, Additive, 14], [Calcium, Standard Feed, 2], [Calcium, Additive, 1]}

が定義される．

スキーマの従属関係を調べると，Fig. 1に示すように非循環となり，アルゴリズム1により最小不動点が求められる．

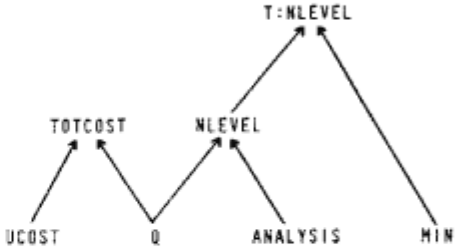

Fig. 1 スキーマの従属関係

S′ = {TOTCOST, NLEVEL, T:NLEVEL} で極小なTOTCOST, NLEVELのうちでTOTCOSTを選び計算する（どちらを選んでも，最終的な結果は変わらない），

$\pi' = \delta(f_{TOTCOST}, \pi^0)$,

$\pi'_{TOTCOST} = \pi^0_{TOTCOST} \cup \{[3.9]\}$
$\pi'_s = \pi^0_s$ （s ≠ TOTCOST）

S′ = {NLEVEL, T:NLEVEL} で極小なNLEVELを計算する．

$\pi^2 = \delta(f_{NLEVEL}, \pi')$,

$\pi^2_{NLEVEL} = \pi'_{NLEVEL} \cup \{[Protein, 15], [Calcium, 4.5]\}$
$\pi^2_s = \pi'_s$ （s ≠ NLEVEL）

S′ = {T:NLEVEL} なので，T:NLEVELを計算する．

$\pi^3 = \delta(f_{T:NLEVEL}, \pi^2)$,

$\pi^3_{T:NLEVEL} = \pi^2_{T:NLEVEL} \cup \{[Protein, false], [Calcium, true]\}$
$\pi^3_s = \pi^2_s$ （s ≠ T:NLEVEL）

S′ = φなので終わり，π^3は無矛盾なので解である．

2.4 モデルの解析

構築したモデルに対して，以下のような解析方法が考えられる．

(1) 解の計算による解析

①What-if分析：初期値を変更したときの解の変化を見る．

②感度分析：初期値を段階的に変化させたときの解の変化を見る．

③直接探索法の利用：モデルが非循環なとき，初期値の一部を入力と考え，それを変化させることにより，対応した最小解が求められる．このとき，制約なし最適化手法である直接探索法を最適化に利用する．

④代替案の生成と選択：初期値の一部を入力と考え，それを許容範囲で変化させて複数の実行可能解(代替案)を生成する．多目的決定法[5]などを利用して選好解を選択する．

(2) 初期値の探索

初期値の一部を入力と考え，与えられた制約条件を

満たす解を生成するような入力を求める．feedmixモデルで，TOTCOSTをある値以下にするためのMINの値を求める場合などがこれに相当する．制約プログラミング[6]の手法が有効であると考えられる．

(3) モデルの変換によるモジュールの利用

構築したモデルを既存のモジュールが使用可能な形に変換し，解析を行う方法である．feedmixモデルで線形計画法モジュールにより最適解を求める場合，つぎのように行う．初期値をπ^*，解をπとする．

①目的関数：$[z] \in \pi_{TOTCOST}$ を最小化

②決定変数：$[\text{Standard Feed}, x_1] \in \pi^*_Q$，
$[\text{Additive}, x_2] \in \pi^*_Q$

③制約条件：$[\text{Protein}, true] \in \pi_{T:NLEVEL} \wedge$
$[\text{Calcium}, true] \in \pi_{T:NLEVEL}$

④数式に変換する．

$$\text{Minimize} \quad z = 1.2x_1 + 3x_2$$
$$\text{subject to} \quad 4x_1 - 14x_2 \geq 16$$
$$2x_1 + x_2 \geq 4$$

⑤線形計画モジュールで解く．

2.5 モデルに対する演算

関係に対する集合演算（結合，射影，制約など）と同様に，モデルに対する演算を定義することができる．これにより，既存のモデルの変更や，複数のモデルの統合的な利用が容易になり，より柔軟なモデル化が可能になる．

(1) 結合(join)

複数のモデルのスキーマ，関数の和集合により新しいモデルを作る機能である．このとき，i)名前は異なるが，同一のものを表している型名，スキーマ名の同一視，ii)名前は同じだが，異なったものを表している型名，スキーマ名の付替え，などが必要になる．

(2) 射影(projection)

キーに対し属性値を指定し，その値をもつ関係だけを取り出したモデルを作る．たとえば，feedmixモデルで"material = Standard Feed"に関する射影は，つぎのようになる．

```
MIN       = [nutr, pounds]
UCOST     = [dollars]
Q         = [pounds]
ANALYSIS  = [nutr, pounds]
NLEVEL    = [nutr, pounds]
T:NLEVEL  = [nutr, boolean]
TOTCOST   = [dollars]
f TOTCOST : P(UCOST)×P(Q)→dollars
  TOTCOST := { [v] | v = sum( {u q |
   [u] ∈ UCOST ∧ [q] ∈ Q} ))
f NLEVEL : nutr×P(ANALYSIS)×P(Q)→pounds
  NLEVEL := { [n, v] | v = sum( {a q |
   [n, a] ∈ ANALYSIS ∧ [q] ∈ Q} ))
f T:NLEVEL : nutr×P(MIN)×P(NLEVEL)→boolean
  T:NLEVEL := { [n, v] | [n, x] ∈ MIN ∧
   [n, y] ∈ NLEVEL ∧ ((y ≥ x ∧ v = true)
   ∨ (y < x ∧ v = false)))
π*MIN      = { [Protein, 16] , [Calcium, 4] }
π*UCOST    = { [1.2] }
π*Q        = { [2] }
π*ANALYSIS = { [Protein, 4] , [Calcium, 2] }
```

(3) 拡張(extension)

指定したスキーマに対して，そのモデルの識別子となるべき属性を追加する．feedmixモデルが豚(pig)についてのものならば，以下のように属性animal，属性値Pigを付加する．UCOSTは，原料にのみ依存するので，属性は付加されない．

```
MIN       = [animal, nutr, pounds]
UCOST     = [material, dollars]
Q         = [animal, material, pounds]
ANALYSIS  = [animal, nutr, material, pounds]
NLEVEL    = [animal, nutr, pounds]
T:NLEVEL  = [animal, nutr, boolean]
TOTCOST   = [animal, dollars]
f TOTCOST : animal×P(UCOST)×P(Q)→dollars
  TOTCOST := { [p, v] | v = sum( {u q | ∃m:
   [m, u] ∈ UCOST ∧ [p, m, q] ∈ Q} ))
f NLEVEL : animal×nutr×P(ANALYSIS)×P(Q)
        →pounds
  NLEVEL := { [p, n, v] | v = sum( {a q |
   ∃m: [p, n, m, a] ∈ ANALYSIS ∧
       [p, m, q] ∈ Q} ))
f T:NLEVEL : animal×nutr×P(MIN)×P(NLEVEL)
        →boolean
  T:NLEVEL := { [p, n, v] | [p, n, x] ∈ MIN ∧
   [p, n, y] ∈ NLEVEL ∧ (( y ≥ x ∧ v = true)
   ∨ (y < x ∧ v = false))}
π*MIN      = { [Pig, Protein, 16] , [Pig, Calcium, 4] }
π*UCOST    = { [Standard Feed, 1.2] , [Additive, 3] }
π*Q        = { [Pig, Standard Feed, 2] ,
               [Pig, Additive, 0.5] }
π*ANALYSIS = { [Pig, Protein, Standard Feed, 4] ,
               [Pig, Protein, Additive, 14] ,
               [Pig, Calcium, Standard Feed, 2]
               [Pig, Calcium, Additive, 1] }
```

スキーマの拡張の目的としては，つぎのことが挙げられる．

i) 汎用的なモデルを対象に合うように詳細化する．

ii) 特定の状況を想定して作成したモデルを，その状況を表現する属性を追加することにより一般化する．

たとえば，上のモデルでは，属性animalを追加したことにより，他の家畜についても同一のモデルで扱うことを可能にする．

(4) 制約(restriction)

属性値に対する制約条件を与え，その条件を満たす初期値からなる新しいモデルを作る．

(5) 抽出(extraction)

指定したスキーマに関係したスキーマ，関数からなるモデルを作る．

3．モデル構築支援システムの試作

2節で述べたモデル記述形式に基づき，モデル構築支援システムをPrologにより試作した．

3.1 システム構成

システムの構成をFig. 2に示す．ユーザーはモデル記述言語によりモデルを作る．このとき，既存のモジュールが組み込み関数として利用できる．記述されたモデルに対して，解の計算，各種演算，数式への変換などを行うことができる．

3.2 モデル記述言語

モデルはモデル記述言語により編集ウインドウ上に記述し，コンパイルされる．feedmixモデルの記述例をFig. 3に示す．型，スキーマはtype文により定義される．関数はデータベース問い合わせ言語であるSQLに類似の形式で記述する．

3.3 機能

(1) データの表示：データは，各スキーマに対応したウインドウ上に表示される（Fig. 4）．ユーザーは，注目するデータに対応するウインドウを作れば，そのデータだけを見ることができる．

(2) 解の計算：2つの方法がある．

①指定した属性値の計算：スキーマ（およびその属性値）を指定し，その値を求める．このとき，つぎの4つの求め方がある．

a. 最初の1つのみを求める．

b. すべての値を求める．

c. 計算に必要なデータが存在しないときは，そのデータを入力するようにユーザーに要求してくる．また，最終的に不足しているデータを表示する．

d. 結果を値ではなく，計算式で表示する．これにより，モデルが表現していた関係を数式の形で取り出すことができる．

②モデルの評価：モデルが非循環なとき，スキーマの従属関係により計算順序を決定し，不動点を計算する．

(3) モデルの演算：結合，射影，拡張，制約，抽出を行う．

(4) 感度分析：指定した属性値を段階的に変化させることにより，他の値がどのように変化するかを見る．

(5) 制約なし最適化手法の適用：Simplex法[7]により制約なし最適化を行う．これは，変化させる属性値をその範囲とともに指定し，目的関数を最大にする点を求めるものである．

(6) モジュールの検索：モジュール（組み込み関数）は，入出力に対しデータ型が定義されている．また，データ型には，データ集合の包含関係に基づくネットワークが作られており，その上の探索によりモジュールを検索する[8]．

4．まとめ

本研究では，モデルを関係の内包的な記述として定式化し，この記述形式によりモデル構築を行うシステムを試作した．今後の課題としては，以下のことが挙げられる．

・解析モジュール，制約解消系とのインタフェース．特に，relationalなデータ構造に対する制約の記述に適した，ＣＬＰ[9]などの制約論理プログラム言語との結合が有効であると考えられる．

・作成したモデルを蓄積するモデルベースの構築方法

・動的なモデル（状態が時間とともに変化するモデル）の表現方法

本研究を行うにあたり，貴重な御意見をいただいた当研究所戸田光彦部長に感謝いたします．なお，本研究は，第５世代コンピュータ・プロジェクトの一環として行われたものである．

〔参考文献〕

1) B.H.Bonczek et al.: A Generalized Decision Support System Using Predicate Calculus and Network Data Base Management, Operations Research, 29-2, 263/281(1981).

2) R.H.Sprague, E.D.Carson : Building Effective Decision Support Systems, Prentice-Hall(1982).

3) A.M.Geoffrion : An Introduction to Structured Modeling, WP-338, Western Management Science Institute, Univ. of Calfornia, Los Angeles(1986).

4) 有澤：データベース理論，情報処理叢書2.情報処理学会(1981).

5) F.Szidarovszky et al.: Techniques for Multi-objective Decision Making in System Management, Elsevier(1986).

6) W.Leler: Constrained Programming Languages, Addison-Wesley(1988).

7) W.Murry(ed.): Numerical Methods for Unconstrained Optimization, Academic Press, 24/28(1972).

8) 平石：DSSにおけるモデル管理，情処学会研究報告 87-IS-17(1987).

9) N.C.Heintze et al.: Constraint Logic Programming: A Reader, 4th IEEE Symp. on Logic Programming(1987).

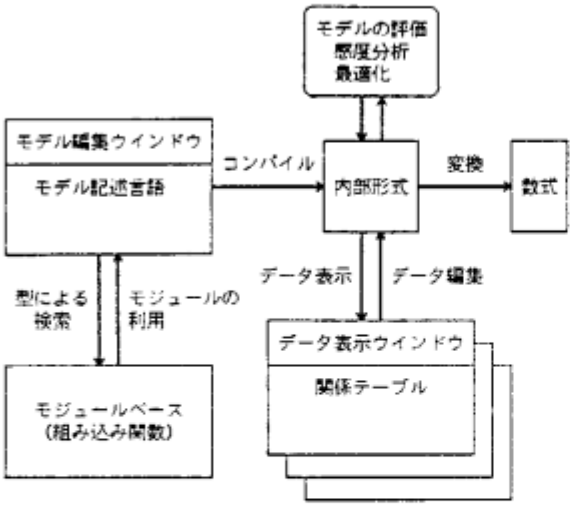

Fig. 2　モデル構築支援システム構成図

```
modelschema feedmix
  typedef
    type nutr super chr#X;
    type material super chr#X;
    type pounds super real#X;
    type dollars super real#X;
    type min record [nutr#X] : [pounds#Y] ;
    type ucost record [material#X] : [dollars#Y] ;
    type analysis record [nutr#X, material#Y] : [pounds#Z] ;
    type q record [material#X] : [pounds#Y] ;
    type nlevel record [nutr#X] : [pounds#Y] ;
    type t_nlevel record  [nutr#X] : [boolean#Y] ;
    type totcost record  [] : [dollars#X]
  functiondef
    function totcost# [] : [V]
     procedure
      V:=select sum(U*Q) from  [ucost# [M] : [U] , q# [M] : [Q] ] ;
    function nlevel# [N] : [V]
     procedure
      V:=select sum(A*Q)
          from [analysis# [N,M] : [A] , q# [M] : [Q] ]  group [N] ;
    function t_nlevel# [N] : [V]
     procedure
      V:=select call((Y>=X, R=true;Y<X, R=false), R)
          from [min# [N] : [X] , nlevel# [N] : [Y] ]  group [N]
  datadef
    data min# [ [protein] : [16] ] , [calcium] : [4] ] ;
    data analysis# [ [protein, standard] : [4] ] ,
                     [protein, additive] : [14] ] ,
                     [calcium, standard] : [2] ] ,
                     [calcium, additive] : [1] ] ] ;
    data ucost# [ [standard] : [1.2] ] , [additive] : [3] ] ;
    data q# [ [standard] : [2] ] , [additive] : [0.5] ]
end
```

Fig. 3　モデル記述言語

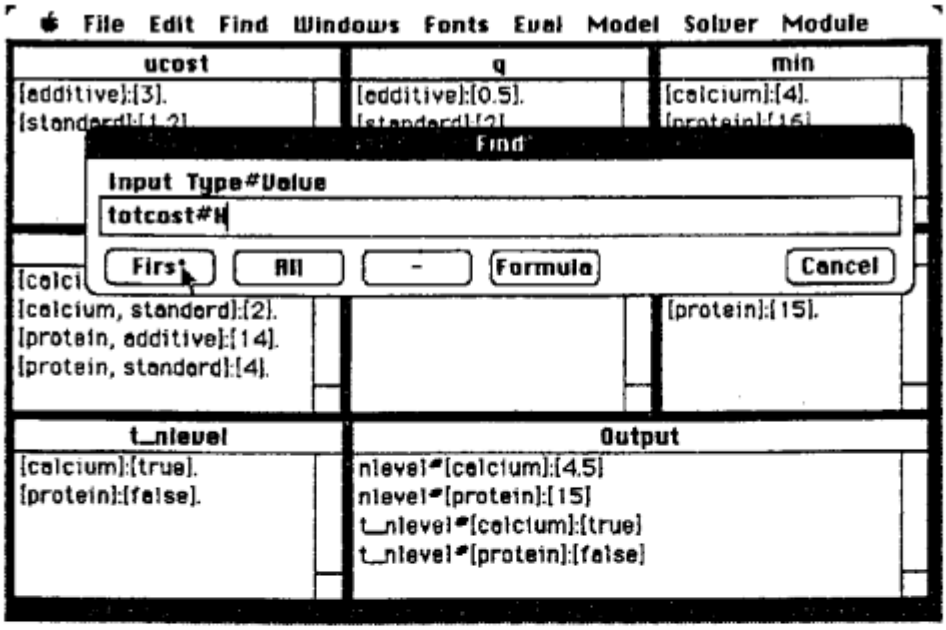

Fig. 4　実行中の画面