情報システム 17－6
（1987. 11. 17）

ＤＳＳにおけるモデル管理

── モデルネットワークによるモデルの統合的利用の支援 ──

平石 邦彦
富士通㈱国際情報社会科学研究所

意思決定支援システムにおいて重要な機能の一つに，様々なモデルを蓄積し，ユーザーに提供するモデル管理機能がある．モデル管理には，モデルがもつ機能を抽象化した情報が必要となるが，本研究では，モデルの入出力に対して定義されたデータ型を基にしてモデル管理を行う．これらのデータ型は階層構造をもっており，この階層構造に，モデルを入出力関係で結合することにより，モデルネットワークを構成する．モデルネットワーク上の探索により，あるモデルの出力を別のモデルの入力にするといった，複数のモデルの統合的な利用の支援が可能になり，また，複数のモデルを使用する過程を集約モデルとしてモデルネットワークに追加していくことにより，モデルの使用方法に関する知識を蓄積していくことができる．

Model Management in Decision Support Systems
－ Integrated Utilization of Models by a Model Network －

Kunihiko Hiraishi
International Institute for Advanced Study of Social Information Science, FUJITSU LIMITED
140, Miyamoto, Numazu, Shizuoka, 410-03 Japan

Model Management is an important function for decision support systems(DSS). It provides DSS users with capability for finding and constructing appropriate models to make decisions on their problems.

The notion of hierarchical data type is defined for inputs and outputs of each model. It is introduced as the information to specify functions of the model. A model network is constructed by joining the hierarchical structure of the data types with the input/output relation of models. Integrated utilization of models can be attained by matching their input/output data types and combining them on the model network. A procedure of using models is added to the model network as an aggregate model, which represents the knowledge about how to use the models.

1．はじめに

意思決定支援システム（ＤＳＳ：Decision Support System）では，その基本的機能として，

ｉ）データベース管理（Data Base Management），

ii）モデル管理（Model Management），

iii）対話管理（Dialog Management）

の３つが挙げられることが多い［Sprague 82］．過去において提案されたＭＩＳ（Management Information System）では，情報提供を行うデータベース管理機能およびレポート作成機能が中心であり，モデルの使用は重要視されていなかった［Keen 78］．一方，ＯＲあるいは経営科学の分野では，モデリングこそ意思決定のための最も強力な道具であるという考え方が強く，ＬＰ（線形計画法）やＰＥＲＴなどに代表される様々な意思決定モデルが考案されてきた．このようなモデルをデータベースなどの他のＤＳＳサブシステムと結合し，すぐれたユーザー・インタフェースのもとで問題解決に利用していくことは，ＤＳＳの重要な機能の１つである．しかしながら，データベース管理と比べ，モデル管理のための方法論はそれほど確立されているわけではない．これは，モデル管理という考え方が比較的新しいものであり［Elam 79］，また，モデル自体の概念も明確でなく，その形態も多様であることが影響している．

モデル管理システムの機能を考える１つの方向は，データベース管理と対比させることである［Dolk 84］．データベース管理システムはデータの追加，削除，更新，検索などの機能をもつが，モデル管理についても，モデルをデータとして考え，モデルベース上でそれらの機能を実現しようとするものである．このような機能を実現するための方法論として，今までに，一階述語論理，フレーム，意味ネットワーク，関係モデルなどの利用が提案されている［Applegate 86］．

モデル管理を行うには，モデルがもつ機能を抽象化した情報が必要である．本研究では，このような情報として，モデルの入出力に対し定義されたデータ型を用いる．このとき，整数型，浮動小数点型，配列型などの物理的なデータ型だけでなく，費用，支出，収入といった，利用目的で規定されるデータ型を用いることにより，モデルのもつ機能を特徴付けることができる．さらに，モデルの検索を，入出力のデータ型を指定することにより行うことが可能になる．

これらのデータ型は階層構造をもっており，この階層構造に，モデルを，その入出力データ型で結合することにより，モデルネットワークを構成する．モデルネットワーク上の探索により，あるモデルの出力を別のモデルの入力にするといった，複数のモデルの統合的な利用の支援が可能になる．

以下，２節では，モデルの概念およびＤＳＳにおけるモデルの利用について，３節では，モデルネットワークとその上でのモデルの検索について述べる．

2．ＤＳＳにおけるモデルの利用

2.1　モデルの概念

［Elam 86］では，モデルとは，特定の現実世界についての解釈を持ち，その世界についての推論をユーザーに提供するような計算表現（computational representation）であるとしている．すなわち，現実に存在する対象を必要な属性について抽象化し，数式，論理式，関係，グラフなどの形式で表現したものである．たとえばＰＥＲＴでは，プロジェクトにおける各活動の前後関係を非循環な有向グラフであらわし，その上でクリティカル・パスなどの解析を行うが，これは，プロジェクトを各活動の前後関係および所要時間に関して抽象化を行ったモデルと考えることができる．

このようなモデルを解析することで，現実の対象がもつ性質を推定することができるが，解析には，何らかの計算が必要になる．この計算過程を中心に考えると，モデルとは，必要な入力を与えることにより，計算を行い，結果を出力する手続きの集まりとしてとらえることができる．［Little 70］が，モデルを問題解決を行うための数値手続きの集まり（a set of numerical procedures）として扱っているのは，この考えに近い．

本研究では，このような計算手続き（モジュール）１つ１つをモデルと考える．そして，モデルどうしを結合して新しいモデルを構成することにより，複雑なモデルを記述していく．このようなモデルを集約モデル(aggregate model）とよぶ．集約モデル自体も，他の集約モデルを構成する要素になることが可能で，これにより，階層的な構造をもつモデルを表現できる．

モデルに対し入出力を考えることで，モデル利用者は，モデルの内部を知らなくても，必要な入力さえ与えれば結果を得ることができるようになる．システムがもつのはモデルの雛型であり，モデル使用者は，パラメータを与えることで現実の問題のモデル化を行う（すなわち，モデルのインスタンスを生成する）．

ＤＳＳでは，このようなモデルとして，数理計画法，多変量解析，統計解析，計量経済モデルなどがよく扱われる．

2.2　モデルを利用した問題解決

モデルを利用した問題解決には，つぎのような段階がある．

① 問題を解決するために有効であると思われるモデルを選び出す.
② 問題をモデルの記法に従って定式化する.
③ モデルを実行する.
④ モデルの出力結果を解釈し，問題解決に利用する.

さらに，あるモデルの出力結果を別のモデルの入力とするような，複数のモデルを統合的に使用する場合もある.

ＯＲや経営科学では，モデルに対する関心の中心はその構造にある．入出力はモデルの構造により決定されるものであり，モデルの構造を知らないと，どのような入力をモデルに与えればよいかがわからず，結果の解釈を行うこともできない．[Sprague 82] では，このような従来のモデルに対する概念に基づいたモデル管理がもつ問題点として，次の６つを挙げている.

ｉ）必要な入力データやパラメータは，入手しにくかったり，生成するのが困難なことが多い.
ii）モデルの出力は，使用するのが困難なことが多い.
iii）複雑な問題に対し，広範囲にわたるモデルを作成し，維持することは困難である.
iv）大きく，複雑なモデルは，管理者にとって理解しにくく，それゆえ信用されない.
v）これらの問題のいくつかに対する回答として，ＴＳＳ環境上で実現される小さなモデル・ライブラリの利用がある．各モデルは固有のデータ・フォーマットを必要とし，また，意思決定のほんの１つの局面しか支援しない．意思決定者や問題解決者には，モデル化の段階と実際の活動との統合が手動で行う過程として残された.
vi）一般的に，意思決定者とモデルの対話は最小限しかない．たとえモデルがＴＳＳのような対話的環境の中で動いていても，対話的に行えるのは必要なデータやパラメータの供給，実行や出力形式のオプション選択にとどまる.

2.3 モデル管理システム

従来のモデル管理に対する反省として生まれてきたものがモデル管理システム(Model Management System)の概念である．モデル管理システムでは，モデルを単独に存在するものとして考えずに，他のモデルやデータベースとの関連を意識している．その機能として，次のことが挙げられる.

ｉ）モデル選択：　問題解決に必要なモデルを選択する機能．これを実現するためには，モデルをその機能により検索する機構が必要となる．また，モデルの利用方法についての知識（モデル化知識）が重要な役割をはたす.

ii）モデルの登録，削除：　問題解決に必要なモデルがシステム内に存在しないとき，ユーザー自身がモデルを定義する機能，および，不要となったモデルを削除する機能.

iii）モデルの編集：　既存のモデルの内容の一部を変更する機能.

iv）問題の変換・構造化：　モデルが利用可能な形式に問題を変換する機能．従来からあるモデルは構造化された問題に対してのみ有効なものが多い．しかし，モデルが直接利用可能なほど問題は構造化されていないのが普通である.

v）結果の解釈：　モデルの出力結果がどのような意味をもつかをユーザーに説明する機能．モデルの出力は，単なる数値やパラメータなど，専門家でなければ理解できないものであることが多い．これらを，ユーザーが理解できるような形に翻訳する.

vi）モデルの統合的な利用：　ある問題を解決するために複数のモデルを使用する場合，それらのモデルの実行順序を決定し，さらに，モデルの実行順序自体を１つのモデルとして登録する．この機能により，汎用的なモデルを組み合わせ，特定の領域の問題に対するモデルを構築することができる．このような操作を，モデルの集約(aggregation) という.

ｉ）～iii）は，データベース管理に類似した機能である.

2.4 モデル利用の例

ＤＳＳにおけるモデル利用の例として，[Bonczek 81] に述べられている一階述語論理によるモデル管理をとりあげる．これは，過去の収支データから回帰分析を使って販売量，収益および配当の予測を行うもので，つぎの４つのモデルが使用される.

① 回帰モデル (REGRESS) ： 回帰係数を計算する.
② 予測モデル (PREDICT) ： 回帰係数により予測値を求める.
③ 配当計算モデル（O_1）： 収益と保有株数から配当を求める.
④ 収益計算モデル（O_2）： 支出と収入から収益を求める.

これらのモデルに対するモデル化知識をつぎのような形で表現している．なお，下線を引いたのは，入力変数である．入力変数は，モデルが実行される前に，値がinstantiate されていなければならない.

(1) REGRESS(Y, X, β) ∧ PREDICT (β,x,r)
∧ SALES(Y,YR1) ∧ SALES-IND(X,YR1)
∧ SALES-IND(x,yr) ⇒ REV(r,yr)
Xが YR1年（過去）の販売指数，ｘがyr年（将来）

の予測販売指数，YがYR1年の販売高であり，回帰分析モデルではXとYを入力として回帰係数βを出力とし，予測モデルではβとxを入力として，yr年の予測収入rを出力する．

(2) PR(π,yr) ∧ SH(shares) ∧ O_1(π,shares,d) ⇒ DIV(d,yr)

πがyr年の収益，sharesが保有株数，モデルO_1ではπとsharesを入力としてyr年の配当dを出力する．

(3) REGRESS(V,U,τ) ∧ PREDICT(τ,u,e) ∧ EXP(V,YR2) ∧ EXP-IND(U,YR2) ∧ EXP-IND(u,yr) ⇒ EXP(e,yr)

(1)と同様に，yr年の予測支出eを得る．EXPは支出，EXP-INDは支出指数をあらわす．

(4) EXP(e,yr) ∧ REV(r,yr) ∧ O_2(e,r,p) ⇒ PR(p,yr)

eがyr年の支出，rがyr年の収入，モデルO_2ではeとrを入力として，yr年の収益pを出力する．

さらに，ユーザーが要求するデータ型と，モデル化知識で使われている述語の引数の関係づけを行うものとして，つぎのメタ公理が定義される．

(5) ～DIV(d,yr) ∨ DIVIDEND(d)

(6) ～DIV(d,yr) ∨ YEAR(yr)

(7) ～O_1(π,sh,d) ∨ SH(sh)

このようなモデル，公理を用いて，たとえば，ユーザーの『１００株に対する１９９０年（将来）の配当は？』といった質問に対し，導出過程（resolution process)により解を見出す．

3．モデル管理システムの設計

[Bonczek 81] が用いた一階述語論理によるモデル管理では，モデルに対応する述語の変数はデータ型をもたず，それらはすべて公理の形で与えていた．また[Dutta 84] では，モデルの入出力のデータ型を陽に表現するためにmany sorted logicを用いている．

本研究で提案するモデル管理は，モデルの入出力に対しデータ型を厳密に規定することにより，2.3 で述べたモデル管理システムの機能のうちで，i）のモデル選択機能，および，vi）のモデルの統合的な利用の実現を主な目標とする．

3.1 データ型

(1) データ型の定式化

ユーザーがモデルを捜し出すためには，モデルがもつ機能を検索可能な形式で特徴付けなくてはならない．そのための方法として，モデルがどのようなデータを入力とし，計算の結果どのようなデータを出力するかという，入出力のデータ型によるモデルの特徴付けを行う．ここでいうデータ型とは，モデルの入出力となるデータの取り得る値の集合を決定するものである．このとき，利用目的により規定されるデータ型を用いることにより，モデルがもつ機能を入出力のデータ型で，ある程度は表現できる．たとえば，2.4 のモデルの例において，モデルO_1は，データ型『収益』と『保有株数』を入力とし，データ型『配当』を出力とするモデルと考えることができるが，モデル使用者は，この記述だけでO_1が何を行うモデルであるかを推定することができる．

データ型は，つぎの２種類の階層構造をもつ．

i）データ型の継承関係： データ型が表現するデータ集合の包含関係に基づいた関係である．たとえば，『収益』と『配当』は，『金額』の形であらわされるが，データ型『収益』および『配当』に属するデータは，データ型『金額』のもつ性質をすべてそなえている（すなわち，『金額』のもつ性質を継承している）．言い換えれば，『収益』と『配当』は，『金額』のより詳細(specific)な記述になっている．２つのデータ型がこのような関係にあるとき，包含されるデータ型を子データ型，包含するデータ型を親データ型とよぶ．

ii）レコード構造関係： 収益のデータを使う場合，何年度の収益かがわからなければ意味がない．また，収益を計算する基となる収入と支出のデータは，同一年度でなければならない．この関係を表現するためには，収益，収入，支出を金額と年度の２つの属性をもつデータ型としておくと都合がよい．このように，複数のデータ型をまとめて１つのデータ型にした構造をレコード構造とよぶ．

データ型の集合をTであらわす．そして，データ型t∈Tを２項組t＝（p，S_t）により定式化する．ここで，p∈Tは親データ型，S_t ＝（A，τ_r）はレコード構造であり，Aは属性集合，τ_r ：A→Tである．

pにより定められるデータ型の継承関係を，

R_{is}＝｛（t，p）｜t＝（p，S_t）∈T｝

であらわす．（t，p）∈R_{is}のとき，pのもつレコード構造はtに継承される．すなわち，

tのレコード構造＝

〔pのレコード構造＋tが固有にもつレコード構造〕

となる．したがって，tは上位のデータ型がもつレコード構造をすべてもつことになる．このレコード構造を，S_t であらわす．

R_{is}は，データ構造は同じだが，もつ意味が異なるデータ型の定義であり，Ada におけるderived type [Gehani 83] の考えに近い．

つぎの構文で，データ型を定義する．

type データ型名
　［*super* 親データ型名 ［*if* 制約条件］］
　［*record* 属性１：ﾃﾞｰﾀ 型；…；属性ｎ：ﾃﾞｰﾀ 型］
end .

［*if* 制約条件］は，親データ型をもつデータのうちで，制約条件を満たすものが子データ型をもつことをあらわす．たとえば，非負整数はつぎのように定義できる．

type NonNegativeInteger *super* Integer
　　if Integer>=0 *end*.

(2) データ型定義の例

2.4 で述べたモデルの例におけるデータ型の定義を示す．

```
type Year super Integer if Integer>=1970 end.
type Money super Integer end.
type Index super Real end.
type Money-Year
     record money:Money ; year:Year end.
type Index-Year
     record index:Index ; year:Year end.
type Shares
     record  shares:Integer ; year:Year end.
type Regression-Coefficient super Real end .
type Sales super Money-Year
    if Money-year:money>=0 end.
type Sales-Index super Index-Year end.
type Expense super Money-Year
    if Money-year:money>=0 end.
type Expense-Index super Index-Year end.
type Divident super Money-Year
    if Money-year:money>=0 end.
type Revenue super Money-Year
    if Money-year:money>=0 end.
type Profit super Money-Year end.
```

Revenue，Expense，Sales-Index などは，年度により値を区別する必要があるので，値と年度の２つの属性をもつレコード構造として定義してある．

3.2　データオブジェクト

データを扱うときには，データの値とデータ型を組みにしたデータオブジェクトの形で扱う．こうすることにより，データに対し，データ型を一意に与えることができる．

データオブジェクトの集合をD，データ型 $t \in T$ をもつデータオブジェクトの集合を D_t であらわす．

3.3　モデル

モデルを，$m=(U, V, \tau_m, C, f)$ により定式化する．ここで，

$U=\{u_1, \cdots, u_k\}$ は入力ポートの集合，

$V=\{v_1, \cdots, v_l\}$ は出力ポートの集合，

$\tau_m : U \cup V \rightarrow T$ は，入出力ポートにデータ型を与える関数，

$C : D_{\tau_m(u_1)} \times \cdots \times D_{\tau_m(u_n)} \rightarrow \{true, false\}$
は実行可能条件，

$f : D_{\tau_m(u_1)} \times \cdots \times D_{\tau_m(u_n)} \rightarrow D_{\tau_m(v_1)} \times \cdots \times D_{\tau_m(v_n)}$ は

モデルの計算を行う関数である．f の定義域は，

$$dom(f) = \{d^n \mid d^n \in D_{\tau_m(u_1)} \times \cdots \times D_{\tau_m(u_n)} \wedge C(d^n) = True\}$$

である．

モデルは，つぎの i），ii）を満たすとき実行可能であるという．

i）モデルのすべての入力ポートに対し，ポートのデータ型と同じデータ型をもつデータオブジェクトが存在する．

ii）それらのデータオブジェクトがモデルの実行可能条件Cを満たす．

Cとしては，前述の，収益を求めるときの収入と支出は同一年度であるといった制約の他に，解くことのできる問題の大きさによる制約などが考えられる．

モデルの実行により，各出力ポートのデータ型と同じデータ型をもつデータオブジェクトが生成される．また，入力となったデータオブジェクトは失われることはなく，参照されるだけである．

3.4　モデルネットワークによるモデルベースの構築

(1) モデルネットワーク

データ型の集合T，モデルの集合M，データオブジェクトの集合Dの３項組N＝（T，M，D）をモデルネットワークとよぶ．モデルネットワークは，データ型の階層構造に，モデルを入出力関係により結合したネットワークである．データ型およびモデルを定義していくということは，このモデルネットワークの構成を行っていることに他ならない．

モデルネットワーク上でのモデルの検索は，基本的には，結合している節点をたどっていくだけなので，複雑なパターンマッチングを必要としない．これは，モデルネットワーク中のモデル数が増加したときに，計算効率の上で大きな利点となる．

(2) モデルネットワーク上でのモデルの検索

モデルの検索を，モデルベースに対して『このような結果を出力するモデルが欲しい』という要求を出したとき，モデルベースからその要求を満たすモデルを捜し出す動作として考える．モデルネットワークはモデルの入出力関係を記述したものなので，検索は入出

力のデータ型により行う．具体的には，つぎのような機能を実現する．

ｉ）求めたいデータを与えたとき，モデルの実行により，現在のデータで求められるかどうかを示す．求められないならば，さらに，どのようなデータが必要かを示す．

ii）実行させたいモデルがあったとき，現在あるデータで実行可能かどうかを示す．実行可能でないならば，さらに，どのようなデータが必要かを示す．

iii）モデルの実行により，現在あるデータからどのようなモデルが実行可能で，それらの実行によりどのようなデータが得られるかを示す．

いずれの機能も，モデルネットワーク上で，データ型の階層構造，および，モデルの入出力関係をたどることにより実現される．これを，つぎの３つの検索ルールにまとめる．この検索ルールを組み合わせて適用することにより，モデルの検索を行う．

《検索ルール１》モデルによるデータの生成：　データ型ｔを出力とするモデルｍが存在し，かつ，ｍのすべての入力ポートについて同じデータ型をもつデータオブジェクトが存在すれば，ｍの実行によりデータ型ｔのデータオブジェクトを求めることができる．

《検索ルール２》レコード構造の合成・分解：　データ型ｔのすべての属性について対応するデータオブジェクトが存在するならば，データ型ｔをもつデータオブジェクトが存在するとみなす（データオブジェクトの合成）．逆に，データ型ｔのデータオブジェクトが存在するとき，S_t の各属性のデータ型をもつデータオブジェクトが存在するとみなす（データオブジェクトの分解）．

《検索ルール３》継承関係の探索：（ｔ，ｐ）$\in R_{is}$ のとき，データ型ｔをもつデータオブジェクトが存在すれば，データ型ｐをもつデータオブジェクトが存在するとみなす．逆に，ｔがレコード構造をもたなければ，*if*以下に定義された制約条件，あるいは，ユーザーの判断により，データ型ｐをもつデータオブジェクトが存在すれば，データ型ｔのデータオブジェクトが存在するとみなすこともできる．

検索ルール２は，レコード構造の各属性を別々に求めたり，レコード構造をもつデータオブジェクトの一部分の属性だけを取り出して使用する場合に対応している．

モデルの入出力となるデータ型を定義されたものに限定してしまうと，汎用的に使用可能なモデルは使用されるデータ型ごとに定義しなおさなければならない．これを避けるために，検索ルール３では，データ型の継承関係における子あるいは親データ型もモデルの入出力にできるようにしている．汎用的なモデルを上位のデータ型と結合させておくことにより，下位のデータ型をそのモデルの入出力とすることができる．

一階述語論理によるモデル管理では，モデルの検索と実行を並行して行っていたが，つぎの理由で，モデルを実行することなくモデルの検索を行うべきであると考える．

ｉ）モデルには実行に非常に多くの時間を要するものがあり，必要な結果を得るためのモデルの実行方法が複数存在するときには，それらのうちから何らかの基準で最適なものを選び出し，その後に実行を行った方が効率的である．

ii）単にモデルどうしを結合しても，計算結果は出るが，それが無意味な場合がある．この判断は，ユーザーに委ねることになるので，実行前に，どのようなモデルが使用されるかをユーザーが知る必要がある．

モデルを起動（すなわちｆを評価）し，実際にデータオブジェクトを生成しないことには，モデルの起動可能条件Ｃを調べることができない．一方，Ｃ，ｆの情報をまったく使用せずにデータ型のみの探索を行うと，多数の無意味な結合が生じる可能性がある．そこで，モデルを実行しなくてもわかる範囲の入出力の関係ｆ’を定義することにより，モデルの結合にある程度制限を加える．つぎのような構文でモデルの入出力関係の定義を行う．

model モデル名
　（入力ﾃﾞｰﾀ 型ﾘｽﾄ，出力ﾃﾞｰﾀ 型ﾘｽﾄ）
　［*suchthat*　入出力ﾃﾞｰﾀ の制約条件ｆ’］
end.

入力（出力）データ型リストは，ポート名：データ型のリストである．

2.4 の例では，モデルO_1 の入出力のデータ型であるProfit，Shares，Dividendのレコード構造は属性yearをもつが，それらの値はすべて等しくなくてはならない．これを，入出力データの制約条件として記述している．また，vector(データ型) は，そのデータ型の不定長の系列をあらわしており，不定個の属性をもつレコード構造として考えることができる．

```
model   Regress
 ( [value1:vector(Number), value2:vector(Number)] ,
   [regression-coef:Regression-Coef] ) end.
model   Predict
 ( [regression-coef:Regression-Coef, value1:Number] ,
   [value2:Number] )  end.
model   O1
 ( [profit:Profit, shares:Shares] ,
   [dividend:Dividend] )
 suchthat
   profit:year=shares:year and
   profit:year=dividend:year  end.
```

```
model   O2
 ( [expense:Expense, revenue:Revenue] ,
   [profit:Profit] )
 suchthat
    expense:year=revenue:year and
    expense:year=profit:year  end .
```

以上，定義したデータ型の階層構造およびモデルの入出力により構成されるモデルネットワークを図示したものが図１である（なお，この図には(3)で定義する集約モデルも入れてある）．

(3) プロセスネットによる集約モデルの表現

複数のモデルを統合的に使用する場合，モデルの実行順序を表現する最も簡単な方法は，モデルのシーケンシャルな順序を記憶しておく方法である．しかし，この方法では，どのモデルの出力がどのモデルの入力になるかという，モデルの実行に関する因果関係を必ずしも反映させることができない．これを示すために，Petri Net 理論におけるプロセス [Goltz 86] の概念を導入したプロセスネットを定義する．これは，モデルネットワーク上の探索経路を非循環な(acyclic)ネットワークで表現するものであり，モデルの実行に関する半順序関係をあらわす（詳細な定義は付録参照）．

プロセスネットを１つのモデルとして登録したモデルを集約モデルとよぶ．与えられたデータに対してモデルをどのように実行していくかを示す集約モデルは，モデル利用に関する知識として見ることもできる．

2.4 のモデルにおけるモデル化知識は，プロセスネットとして表現できる．例えば(1)の予測収入を得るモデル化知識は，図２のプロセスネットで表現される．vector(Sales)は，レコード構造の分解によりvector(Money)とvector(Year)に分解され，vector(Money)は継承関係をたどりvecrtor(Number)としてモデルRegressの入力になる．vector(Sales-Index)についても同様である．このとき，vector(Sales)とvector(Sales-Index)の年度が等しいという制約条件が付く．Sales-IndexとRevenueの年度についても同様である．

このプロセスネットは，つぎのような集約モデルとして定義できる．

```
aggregate-model Pred-Rev
 ( [sales-indexes:vector(Sales-Index),
    saleses:vector(Sales),
    predict-sales-index:Sales-Index ] ,
   [predict-revenue:Revenue ] )
  suchthat
   sales-indexes:year = saleses:year and
   predict-sales-index:year
              = predict-revenue:year end.
```

(2)の予測支出を得るモデル化知識についても同様である．

```
aggregate-model Pred-Exp
 ( [expense-indexes:vector(Expense-Index),
    expenses:vector(Expense),
    predict-expense-index:Expense-Index ] ,
   [predict-expense:Expense ] )
  suchthat
    expense-indexes:year = expenses:year and
    predict-expense-index:year
              = predict-expense:year end.
```

集約モデルは，実行時にはプロセスネットに展開され，個々のモデルごとに実行可能条件が調べられ，実行される．したがって，集約モデルのすべての入力ポートにデータオブジェクトが揃ったとしても，内部のモデルの起動可能条件により，実行が完了しない場合も存在する．

(4) モデル検索の例

集約モデルを取り入れたモデルネットワークに，つぎのデータオブジェクトを配置し，１９９０年の配当を求める場合を考える．

```
shares(100,1990)
vector(Expense)  [(1300,1970),(1400,1975),…]
vector(Sales)  [(2500,1970),(2800,1975),…]
vector(Expense-Index)  [(80,1970),(90,1975),…]
vector(Sales-Index)  [(120,1970),(125,1975),…]
Expense-Index(110,1990)
Sales-Index(140,1990)
```

求めたいデータオブジェクトは，dividend(d,1990)である．モデルネットワーク上の探索により，図３に示すプロセスネットが生成される．このプロセスネットもまた，配当を過去のデータから配当を計算するモデルとして登録することが可能である．

4．おわりに

本論文では，モデル管理システムにおけるモデルベースを構築するための枠組みとして，モデルを入出力のデータ型により結合したモデルネットワークを提案した．これにより，つぎの機能が実現される．

i）モデルの検索を，その入出力関係をたどることにより，モデルネットワーク上で行うことができる．

ii）必要なデータを得るためのモデル起動順序の候補を，実際にモデルを起動することなく得られる．

iii）複数のモデルの統合的な使用方法を，集約モデルとしてネットワーク上に追加していくことにより，モデルの使用法に関する知識を蓄積することができる．

モデルネットワークはモデルベースを構築するための枠組みだけを示したものである．モデルの入出力のデータ型だけでは，そのモデルの機能を特徴付けるには不十分であり，また，モデルの利用方法に関する知識も集約モデルによるものだけでは，不十分である．モデル管理システムには，これらの機能を補充する知

識ベースの他，モデルの検索過程を視覚的に表示するグラフィック・インタフェースなども必要である．

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]

[Applegate 86] L.H.Applegate, B.R.Konsynski et al.: Model Management Systems : Design for Decision Support, Decision Support Systems, Vol.2 (1986)
[Bonczek 81] B.H.Bonczek et al.: A Generalized Decision Support System Using Predicate Calculas and Network Data Base Management, Operations Research, Vol.29, No.2 (1981)
[Dolk 84] D.R.Dolk, B.R.Konsynski : Knowledge Representation for Model Management Systems, IEEE Trans. on Software Engineering, SE-10, No.6 (1984)
[Dutta 84] A.Dutta, A.Basa : An Artificial Intelligence Approach to Model Management in DSS, IEEE Computer, No.9 (1984)
[Elam 79] J.J.Elam : MODEL MANAGEMENT SYSTEMS - AN OVERVIEW, Working paper 79-12-04, Dept.of Decision Sciences, The Wharton School, Univ. of Pensylvania (1979)
[Elam 86] J.J.Elam, R.M.Lee : Guest Editors' Introduction, Decision Support Systems, Vol.2(1986)
[Gehani 83] N.Gehani : Ada - An Advanced Introduction, Prentice-Hall (1983)
[Goltz 86] U.Goltz, Y.C.Yi : SYNCHRONIC STRUCTURE, Lecture Note in Computer Sciense 222, Springer(1986)
[Keen 78] P.G.Keen, M.S.Morton, DECISION SUPPORT SYSTEMS - AN ORGANIZATIONAL PERSPECTIVE, Addison- Wesley (1978)
[Sprague 82] R.H.Sprague, E.D.Carson : BUILDING EFFECTIVE DECISION SUPPORT SYSTEMS, Prentice-Hall(1982)

[付録] プロセスネットの定義

モデルネットワーク$N=(T, M, D)$に対し，以下のようにプロセスネット$F=(Q, E, p)$を定義する．

(1) $p : Q \cup E \to T \cup M$をプロセスとよぶ．

(2) Qは*condition*の集合で，

i) 任意の$q \in Q$について，$p(q) \in T$

(3) $E = E_{is} \cup E_{dec} \cup E_{com} \cup E_m$は*event*の集合である．$E_{is}$は継承関係の探索に，$E_{dec}$はレコード構造の分解に，$E_{com}$はレコード構造の合成に，$E_m$はモデルに対応する．

a．$e_{is} \in E_{is}$は2項組$e_{is}=(q_1, q_2)$であらわされる．ここで，$q_1 \in Q$をe_{is}の入力，$q_2 \in Q$を出力とよぶ．このとき，

ii) $(p(q_1), p(q_2)) \in R_{is} \cup R_{is}^{-1}$

b．$e_{dec} \in E_{dec}$は3項組$e_{dec}=(q, A, \rho_{dec})$であらわされる．ここで，$q \in Q$は$e_{dec}$の入力，Aは属性の集合，$\rho_{dec} : A \to Q$は単射な写像である．$\exists a \in A : \rho_{dec}(a) = q$のとき，qを$e_{dec}$の出力とよぶ．このとき，

iii) $\underline{S_{p(q)}}=(A, \tau_r)$であり，任意の$a \in A$について，$p(\rho_{dec}(a)) = \tau_r(a)$

c．$e_{com} \in E_{com}$は3項組$e_{com}=(q, A, \rho_{com})$であらわされる．ここで，$q \in Q$は$e_{com}$の出力，Aは属性の集合，$\rho_{com} : A \to Q$は単射な写像である．$\exists a \in A : \rho_{com}(a) = q$のとき，qを$e_{com}$の入力とよぶ．このとき，

iv) $\underline{S_{p(q)}}=(A, \tau_r)$であり，任意の$a \in A$について，$p(\rho_{com}(a)) = \tau_r(a)$

d．$e_m \in E_{md}$は3項組$e_m=(U, V, \rho_m)$であらわされる．ここで，Uは入力ポートの集合，Vは出力ポートの集合，$\rho_m : U \cup V \to Q$は単射な写像である．$\exists w \in U : \rho_m(w) = q$のとき，qを$e_m$の入力とよび，$\exists w \in V : \rho_m(w) = q$のとき，qを$e_m$の出力とよぶ．このとき，

v) $p(e_m) = (U, V, \tau_m, C, f)$であり，任意の$w \in U \cup V$について，$p(\rho_m(w)) = \tau_m(w)$

(4) Eの入出力関係を$A \subset Q \times E \cup E \times Q$とし，$s \in Q \cup E$に対し，${}^{\cdot}s = \{r \mid (r, s) \in A\}$，$s^{\cdot} = \{r \mid (s, r) \in A\}$とすると，

vi) 任意の$q \in Q$について，$|{}^{\cdot}q| \leq 1$

vii) 任意のr，$s \in Q \cup E$について，$(r, s) \in A^+ \Rightarrow r \neq s$ (acyclic)

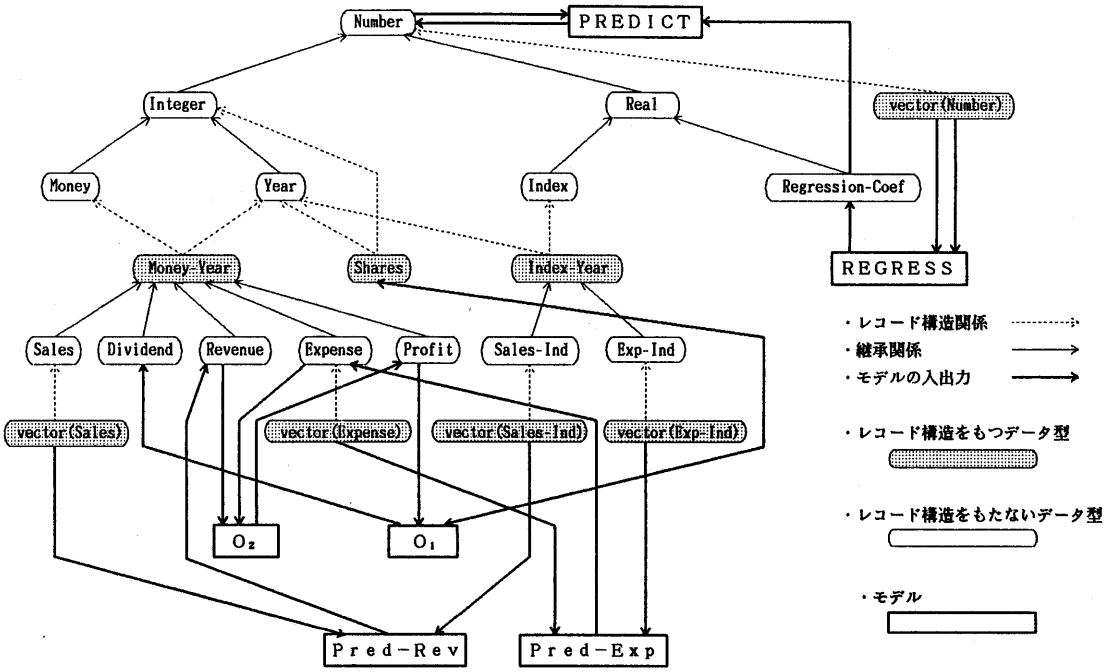

図1　モデルネットワーク

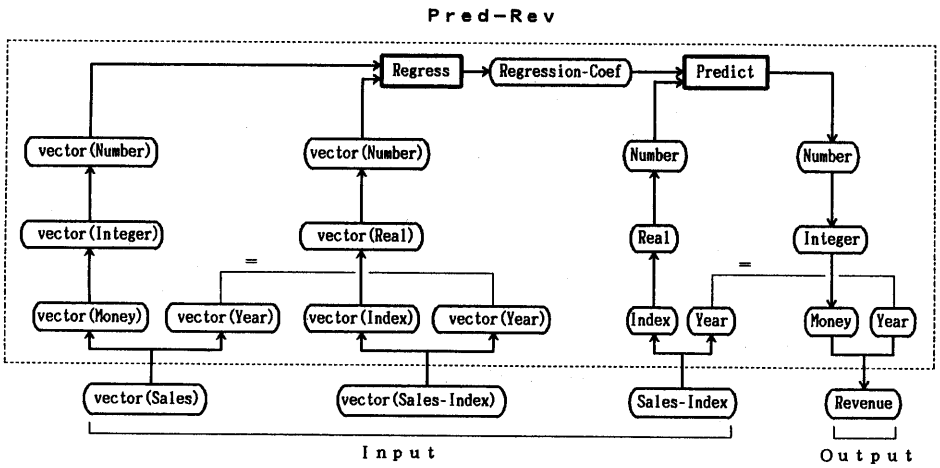

図2　予測収入を得るプロセスネット

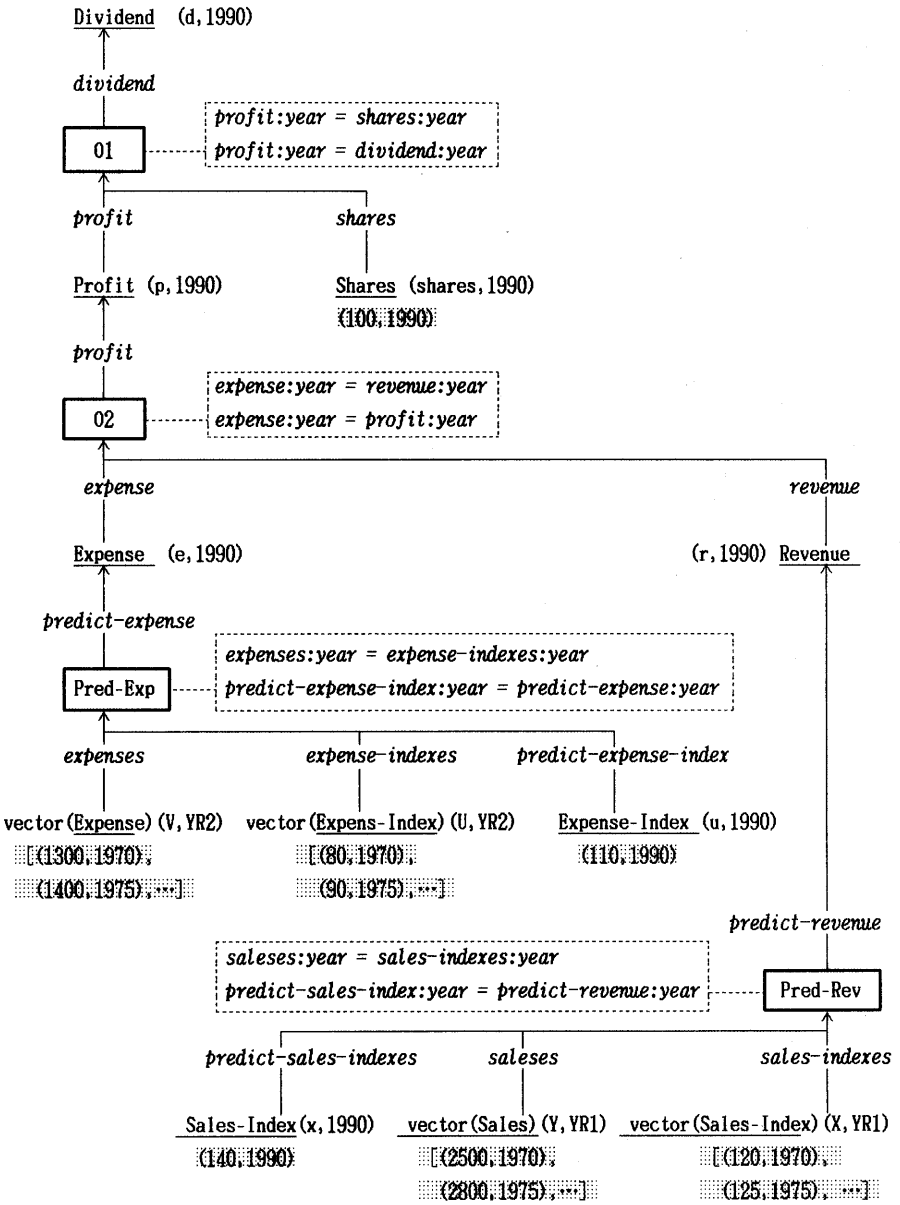

図3　探索経路を示すプロセスネット