Vol. 27 No. 2　　情　報　処　理　　Feb. 1986

解　説

データ構造とモデリング技術†

大須賀　節雄††

1. はじめに

データ構造は情報処理分野でこれまで何度も議論の対象とされてきたし，データ構造に関する書物も数多く出版されてきた[2),16),18),34)]．計算機内でのデータ構造の形式という点ではすでに計算機ソフトウェア技術のかなり初期の頃にほとんど確立されており，表現形式が定まればその操作アルゴリズムは目的に応じてほとんど自動的に定まってしまう[5)]．以後はそれぞれの表現ごとに現れる操作面での特徴が，たかだか順序性程度の意味的（というよりは数学的）な側面を伴って論じられればデータ構造の形式上の議論は終ってしまう．事実，多くの書物はこの議論を中心に置き，それ以外は，これらの形式的な扱いをするための手続きの表し方と，形式ごとに二，三の応用例を挙げて終っていた[33),38)]．

このように形式上の議論に限ればデータ構造の議論は範囲も限定され，内容も明確であるにもかかわらず，この話題が繰り返し現れるのは，データ構造の持つ意味的な側面が明らかにされないままに残されているからであろう．計算機プログラムの手続きの部分に関してはセマンティクスに関する議論が比較的しやすいが，データ構造に関してはセマンティクスが定義しにくいため，一向明らかにされないままに残されていた．

しかし近年，この状況を変化させるいくつかの傾向が現れてきた．それらはデータベースにおけるデータ・モデル論[6)]，抽象データ型の議論[13),19),29)]，最近流行のオブジェクト指向の概念[12)]，そしてこれも最近の顕著な傾向の一つである知識処理における知識表現の問題である[7)]．これらはデータベース，プログラミング言語，人工知能といった異なった分野から発し，それぞれの分野ごとに定義された名称で呼ばれているが，いずれも計算機内の情報として，手続きという動的な表現とは別の，静的な存在としてのデータの意味にも立ち入ろうとするものである点が共通である．

情報の意味を正しく定義することが容易なことではないことはもちろんであるが，以下では不毛な議論に陥ることをできる限り避けてデータ構造の意味の問題を考えてみる．この議論は必然的に，最近関心を持たれ始めている概念モデル論に導かれる[3)]．したがって，以下で述べるのはモデル論の立場からデータ構造を考察する一つの試論である．

以下，第2節では情報の形式と意味の関係について考察し，そこでのデータ構造の持つ意味について論ずる．第3節ではデータ構造の意味表現に関して大きな影響を与えた2つの概念，抽象データ型とデータ・モデルの考え方とその特徴などについて述べる．第4節では宣言型言語を設計する上で実際的なアプローチとしてモデル化技法を挙げ，それが広い応用範囲を持つことを示す．最後に第5節ではこのモデル化という技術内でのデータ構造の役割を述べ，階層構造とグラフ構造が実現可能であると同時に，有用であることを示す．

2. 情報の意味

記述系として，情報の新しい形式と意味を定義すること，すなわち新しい言語の設計が一般論として困難なことは言うまでもないが，特に計算機言語に関しては，非手続き的言語の意味定義には多くの困難を含む．このことはたとえば知識処理のための知識表現言語が多くの関心を集めているにもかかわらず，いまだにその多くは設計の根拠が必ずしも納得されたものではないという状況にも見ることができる．データ構造の意味解釈の難しさも，非手続き的表現であるという点においてこれと共通するところがある．

この点をもう少しはっきりさせるため，情報処理における実問題とその計算機内の情報表現の関係を考えてみる．これを図-1のように表す．この図は（1）実世界で生ずる問題が人により一定の前処理を受けて

† Data Structure in Modelling by Setsuo OHSUGA (Institute of Interdisciplinary Research, Faculty of Engineering, University of Tokyo).

†† 東京大学工学部境界領域研究施設

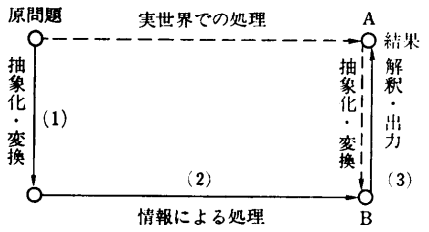


図-1　情報処理のモデル

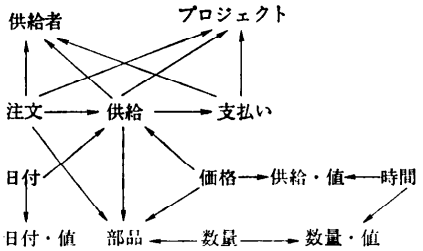
整理・抽象化された後，計算機内表現に変換される部分，(2)計算機内での処理部分，(3)処理結果の表現と解釈の部分から成り，このプロセスが実世界での問題解決プロセスにちょうど対応すること，すなわち，情報の世界での処理結果Bが，実世界で処理を行ったと仮定したときの結果Aに上記(1)で用いたものと同じ抽象化・変換の規則を適用した結果と一致するものであるべきことを表している．図中，実線部分は実際に実行される変換を，点線は仮想的な変換を示す．

計算機内の情報のセマンティクスは表現の形式と実世界の概念との対応関係と考えれば良いが，実際には計算機の機構を駆動せねばならないという制約があるため，この対応関係は必ずしも直接的なものにはなり得ない．ここにさまざまな表現形式を定義する可能性が生ずる．

今日，計算機内の情報表現に対し形式的な意味記述を与える方法として次の3種があるといわれている．操作型 (Operational) セマンティクス，表示型 (Denotational) セマンティクス，公理論的 (Axiomatic) セマンティクスである．しかし以下では前記知識処理を含めて，今日，生じつつある情報処理の変化を操作型（手続き型）言語から宣言型（非手続き型）言語への変化としてとらえ，この2種の型の言語およびその変化とデータ構造の関わりについて考察する．

操作型言語は今日の言語の主流であり，手続きによってアルゴリズムひいては問題の意味を表す言語である．この言語において手続きによって記述された部分と実世界の概念とは直接的には対応せず，図-1 において，“操作手続きを実行して得られた結果(B)が人の解釈のもとで実世界での処理結果(A)に対応する”ことまでしか言えない．

一方，宣言型言語のセマンティクスは，語のレベルで（記号列としての）言語表現にたいして記述しようとする世界内の対象あるいは関係などの概念を対応づけることによって定義づけられる[9]．記述される世界



注) 各矢印には意味を表す記号が与えられるがここでは省略している

図-2　セマンティックネットワークの例

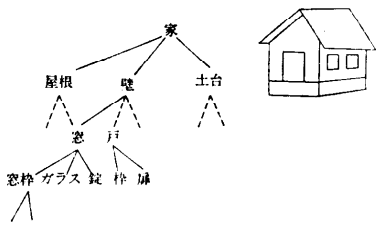


図-3　「家」の構造の表現例

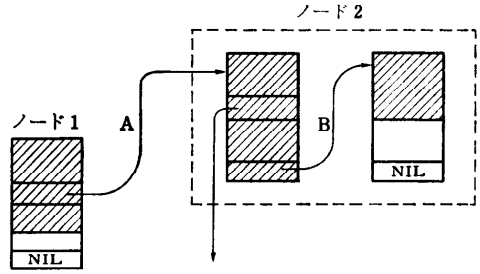
は実世界，あるいは数学的表現のように人により一部抽象化された実世界のモデル，であるが，宣言型言語表現の構造はこの概念構造にほぼ同形で対応する．たとえば概念構造をセマンティック・ネットワークで表す場合を考えていただきたい（図-2）．

このいずれの言語にもデータ構造が定義されるが，言語の本体部分とデータ構造のかかわり方は両言語で異なってくる．操作型言語におけるデータ構造は言語本体の手続きの部分とは異なる非手続き表現であり，当然手続き部分のセマンティクスとは異なる意味定義が与えられている．非手続き表現として，これは宣言型言語のセマンティクスと類似し，データ構造の一部の要素は記述される世界（これを以下記述世界と略称する）の概念構造に一対一の対応を示す．しかし同時にデータ構造は，手続きがそこに作用するという形で，手続き部分の意味を表現する上で便利なように構造化されている．この意味ではデータ構造には操作型の言語の一部として，非手続き表現のもつ意味表現とは関係しない部分も含まれる．たとえば同じ木構造でも，実世界で“家は屋根と壁と土台から構成され，壁は窓，戸その他を含み，窓は窓枠，ガラス，錠を含む”(図-3)などのように実体の構造にほぼ一対一に対応するものもあると同時に，B-Tree や Quadtree のように探索の効率化を重視し，各中間ノードは現実の事物との対応を持たず，操作型の表現としてのプログラムの実行上の便宜から定義されるようなものもある[30]．

さらにデータ構造は計算機の記憶装置の構造に依存する部分を持っている．例としてセマンティック・ネットワークを表現する場合を考えよう．このネットワーク内で各ノードは基本的な概念を表現し，それらの概念間の関係をリンクで表すことにより意味の構造が作られている．このリンクとしてポインタを用いるものとしよう．このときのポインタは実世界での人間の理解としての意味を表している．一方，個々の概念を表すノードには記憶管理の便宜上，固定長のメモリ・セルを与えるものとしよう．もし概念の表現が1セルに与えられたメモリ長では不足の場合，ポインタで結合された複数セルにわたってデータを格納するという技法はしばしば用いられる（図-4）．このときのポインタは計算機の構造上の理由で用いられるものであり，実世界内での概念間関係の表現とはまったく無関係である．

このように計算機内の情報の表現については，前記宣言型と操作型の区別に加え，計算機の構造に依存するか否かという点は，情報の理解や記述のしやすさという観点から非常に重要である．データ構造は上述のように宣言型のセマンティクス（実世界の対象と直接対応する部分），操作型のセマンティクスとともに記憶構造すなわち機械に依存する部分を含み，結局これらすべての要素を含んでいる．同一形式内に異種の意味表現が混在することは情報の理解を困難にする．今日の操作型言語に含まれているデータ構造はこの意味で理解しにくく，またプログラムの自動生成などの試みに際してもデータ構造部分の取扱いが大きな障害となってきた（図-5）．

操作型言語と異なり，宣言型言語では言語の本体部分とデータ構造はともに非手続き表現として類似の意味定義がなされている上，データ構造に操作型の意味定義の補助的役割も不要になる．したがって言語の本体部分とデータ構造の相違は操作型言語ほど顕著ではなく，むしろ表現の効率などの，意味表現以外の理由で構造が用いられるという傾向が強くなる．



A：意味を表すポインタ
B：記憶のためのポインタ

図-4　異なった目的にポインタが用いられる例

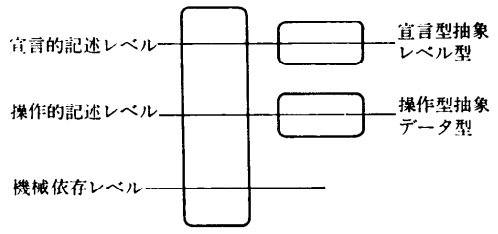


図-5　抽象データ型の効用

しかし，プログラマないし利用者のレベルでは宣言型言語が用いられても，これが実際に処理される段階では計算機のハードウェアを駆動し得る形式に変換されねばならない．この変換結果は手続き型の表現であり，その表現に固有のデータ構造を含む．宣言型言語はこの変更の可能性のもとで成り立つものである．

3. 抽象データ型とデータ・モデル

ソフトウェア技術は，より理解しやすい情報表現を求めて発展してきたと言える．この努力は図-1 に則して述べれば，実世界で人が認知する問題の表現形式とその計算機内表現形式間の距離を縮めることである．同図(1)の部分がソフトウェア開発に相当するから，この距離を縮めることはきわめて重要であり，そのためには計算機内表現をできる限り実世界での表現に近いものにすることが必要である．

この努力は大きな流れとして捕えると，(1)機械依存からの脱皮，(2)操作型記述から宣言型記述へ，という形で変化している．前述のように操作型言語から出発した計算機言語においては当初から手続き部分とデータ部分が分かれており，この両方の部分の変遷の過程が異なっていた．

手続き部分については，機械依存からの脱皮は比較的早期に達成された．それは (i)機械語からアセンブラへ，そしてコンパイラへという言語の高級化と，(ii)オペレーティング・システムの高度化，という二つの面で達成されてきた．現在は操作型言語から宣言型言語への移行過程にあると考えられる．オブジェクト指向言語の流行もその一つの現象とみなされるが，これにも見られるように，宣言型セマンティクスへの移行はデータ構造と本体の記述部分の一体化とも言える．今後はデータを含む宣言型意味表現を明らかにし

てゆくことが必要である.

一方，データ部分については機械依存からの脱皮が遅れた．機械レベルでのデータ表現の問題から利用者を解放することは，特定の問題向きの言語であればその問題向きの組込みのデータ型を定義するという方式で可能となるが，一般的にデータ表現の抽象化を可能にする方式として出現した概念に，抽象データ型と，データベースにおけるデータ・モデルがある.

3.1 抽象データ型

抽象データ型そのものについては本特集号にも含まれているから，ここでは計算機における情報表現というグローバルな枠組内で考察する．抽象データ型は個個の問題解決の分野で有用な対象データの新しいデータ型を導入するために用いられる．利用に際して，プログラマはこれらの対象の振舞いや特性，これを表すために記憶する情報とそれから得られる情報について知っているが，データがいかに記憶装置に貯えられ，そのためにどのようなアルゴリズムが用いられるかには関与しない[20]．対象の振舞いは記述の組で表される．これには対象データを生成し，そこから情報を取り出し，またそれを変更する操作も含まれる．このように抽象化データは対象を表すデータ（構造）と，対象の振舞いなどを表す一組の記述をカプセル（encapsulate）したもので，プログラマは定義された記述を通してこれにアクセスする.

抽象データ型のこのような構造を高級言語に組込むことによって，プログラマは問題処理のアルゴリズムの記述を機械に非依存の形で行うことが可能となる．すでに多くの言語（Ada, Alphard, Clu, Euclid 他）が抽象データ型を定義する機能を与えられている.

抽象データ型の実現法としては，対象の実体に関する記述を操作型で行うか，宣言型で行うかの2つの可能性があり，これにより定義されるデータ型自体の外から見たセマンティクスが操作型か宣言型となる.

このことを前提として，抽象データ型のもつ意義を考察してみよう．以下，抽象データに対する操作記述が手続き的に行われ，外から見たときに操作型に見える抽象データ型を操作型データ型，宣言的に行われるものを宣言型データ型と呼ぶことにしよう．どちらのデータ型にしても，データ構造の表現は抽象データ型としてカプセルに封じ込められ，データ構造が本来含んでいた機械依存の部分は外には現れない．これが抽象データ型の第一義的な目的であった.

操作型抽象データ型について言えば，さらにデータ構造が部分的に含んでいた非手続き的セマンティクスの部分も含めて，外から見たデータがすべて手続きと同等のものとなる．すなわち，操作型言語において，従来，異質のものとして分離されてきた手続き記述とデータ記述が抽象データ型化によって一元化し，見かけ上すべて操作型セマンティクスで統一されることになる（図-5 参照）．現実のプログラムの作成に際してプログラマへの負担軽減の効果に関する評価はともかくとして，意味表現を一元化しているこの効果は大きい.

宣言型抽象データ型についてもこの状況は同様である．宣言型の抽象データ型が手続き型言語内で用いられる場合はコンパイル時に宣言型から操作型への変換が必要になるほかは上述の場合とまったく同様であるが，宣言型言語とともに用いられるとき，表現の理解のしやすさは著しく向上することが期待される．しかし宣言型の言語開発が多くの困難を含むことは2節の最初に知識表現言語の例で述べた通りである．この難しさは宣言型の言語はその構文一意味の関係が自然言語や図形など，われわれが日常的に用いる言語（図形その他を含む広義の言語）と類似していること，したがってこのような言語を用いる汎用システムを開発するとしたら（自然言語そのほかをそのまま利用できるなら問題はないが，その全体系の処理は今日の技術ではいまだ困難なため），ある一定の限定された表現の形式，すなわちその表現の範囲内で多くの分野の異なった型の問題でも記述でき，しかも計算機処理可能な形式，を求める必要があるが，それは多種類の応用の分析を含む困難な作業であるためである．事実，前述の知識表現言語に関して，このような立場からの要求を，整理された形で求めることすらなされていないのが現状である.

3.2 データ・モデル

操作手順の記述からも機械の構造からもまったく独立に非手続き情報を定義する方法として確立されたもう一つの概念としてデータベースにおけるデータ・モデルがある．しかもこれはすでに実用化された技術として広く利用されている．データベースは周知のごとく記憶情報を複数のユーザで共用することを目的とし，このため応用プログラムからの独立，機械からの独立を基本条件においた[6]．これを実現するために，実際のデータの記憶構造（物理構造）と別に，利用者に見える仮想構造ともいうべき論理構造を定義し，それに基づいてデータ操作を行う方式を実現した．論理

構造はそれによって問題領域における記述世界を表すデータモデルである.

論理構造はそれへの操作命令と一体化されて初めて意味を持つ. この操作命令はデータが実際に記憶されている物理構造の操作プログラムに対応する. したがって, この構成は前記の抽象データ構造と類似である. というより原理的には抽象データ構造そのものと言ってよい[21]. ただしデータベースを管理するデータベース管理システムは, この他に質問言語を解析して, 操作命令の列から成る最適な問合せ手順を生成したり, データの論理的および意味的な矛盾をチェックする一貫性制約の実施, ユーザ・ビューの設定などの数多くの仕事をする. しかしデータベースの議論をすることが目的ではないから, これらについてはこれ以上立入らない.

データベースがこのように非手続き的にかつ機械独立に情報の意味を表現し得るシステムとして成功したのは, 実は応用の分野を限定し, したがってモデルとして表現すべき範囲を, 比較的単純な論理構造で表現し得るものに制限したからである. この範囲はいわゆる事務処理と呼ばれる人事, 在庫, 仕入れ, 会計などの分野である. これらの分野ではすでに事務処理の手順が確立されており, 情報表現の形式化も十分になされていた. データベースはこの形式を導入することによって前記方式を確立したものである. この意味でデータベースは問題向きシステムとも言えるが, この形式化の範囲内で事務処理一般が可能なので, 特定の問題向きシステムとも異なり, 準汎用システムとでも呼ぶのがふさわしい.

最近になって, 事務処理以外の分野でもデータベースを利用する機運が高まってきた[14),15),31)]. たとえば工学の分野で設計・開発あるいは研究の対象のモデル化にデータベースを利用しようという試みがなされてきた. このような工学分野でのモデル化手段を目指したデータベース・システムに対する要求は, 従来の事務処理分野を対象に開発されたデータベース・システムに比し著しく高度のものであるため, 従来のものを転用するという方法と別に, 工学向けのデータベース・システムを開発する必要があるという認識が強まってきた[31)]. 工学目的のデータベースはエンジニアリング・データベースと呼ばれるが, このための新しい管理システムへの要望が高まっているわけで, この特徴は設計対象のモデル化に必要な構造表現が可能であること, およびインタラクティブに人と計算機が協力して高度の要求を満たすような複雑な対象のモデルを見出す, 試行錯誤の動的なプロセスを支援し得ること, その他である.

データベースにより複雑な構造表現能力を与えることは現在各地で積極的に進められている[23),31)]. これには関係データベースの関係スキーマ自身に階層構造を含めると同時に, データベース言語でこの構造に対応する階層性を記述することにより構造要素にアクセスする, というように, データ構造を外に見せるもの[23)], データベース自身を一つの大きな抽象データ型とし, 言語としてはこのようなデータ構造の詳細な記述はしないで操作指令のみを与えるようにするもの[31)], などの考え方が出されている.

データベースを構造化することはこのような方向で可能であるが, 動的な性能を与えるにはこれらは必ずしも十分なものではない. 大量のデータとその構造情報を記憶することと, それを動的に処理することは両立しにくいので, これを同一機構で行う代りに, 2つの機構に分ける方が実現性は高いように思える. このときは上述の構造化されたデータベースに加えて, ここから必要なデータを読み出し, 要求・仕様を満たすようにモデル構造を処理する部分を組合わせることになろう. この部分はエンジニアリング分野でのノウハウを利用できること, したがって前記知識ベースを持つことが望ましい.

一方, 現行データベースの技術をより完全なものにする上でも, このような二層構造を採用する考え方が広がりつつある. これはデータベースに推論機能と演繹規則を記憶する部分を付加することより (1)演繹的な検索と, (2)一貫性制約や質問の最適化その他の機能をより完全なものにすることを目的とし, 論理データベースとか演繹的データベースと呼ばれている[11),36)].

抽象データ型とデータベースにおける上記の新しい傾向は興味深い. どちらも宣言型のシステムを目指しているが前者は情報の表現法という一般的な観点からのトップダウン型のアプローチであるのに対し, 後者は応用分野を拡大するというボトムアップ型のアプローチである. このどちらもが結果的に知識処理という情報処理の新しい方式を指向しているのである.

4. 問題解決とモデル化技法

3.1 節で宣言型言語の仕様を定めるためには応用分野の分析が必要であると述べた. 理想的には分析を通

して個々の問題の解法を探り，それらすべてを包含する問題解決手法を見出して，それを記述し得るように宣言型の言語仕様を定めることである．しかしこのような手順をとることが困難なことは言うまでもない．

データベースからの上位の発展形式を求めるというアプローチはもう少し具体的である．この目的は対象のモデルを表現し，かつ目標に向けそれを操作するプロセスを支援することである．この枠組の中で可能な限り一般化をはかるために，モデルの表現手法を一般化し，広範囲な問題向きのモデル表現を可能にする形式を求めることが必要である．

仮にこのような方式が実現したとしても，モデル化という一つの枠組を設定している以上，方法論という立場では制約的であることは免れない．しかし現実に可能なアプローチとしてはこれしかなく，またモデル化手法は実質的には十分な一般性を持つ方式である．このことを簡単に示した後，モデル表現のためのデータ構造について次章で考察する．

モデル化法が広い範囲の応用に対して普遍性を備えた方法論であることは，問題解決の手順を分析することにより示される．一言で問題と言ってもさまざまのものがあり，そのすべてを一括に論ずるわけにはゆかないので，ここでは問題形成の特徴に基づいて，すべての問題を3つのクラスに分類しておこう．

将来の計算機が扱う問題としては，その内容や目的，用いられる手法などに基づいて，解析・予測・診断・設計・計画・監視・制御など多岐にわたるが，これらをさらに分析すると，いずれもある考察される対象の実体と，ある環境のもとでその実体が示す属性・性質・機能・振舞など，その実体に伴って生ずる機能（と呼んでおく）の間の関係を求めるという形で表すことができる．問題は実体，環境，機能のうちの1つが未知であって，2つの既知の部分から未知の部分をどのように見出すかという観点から3つに分けられる．第1は実体が与えられ，これが一定の環境の下でどのような機能（未知）を示すかというもので，これを解析型問題と呼んでおく．第2は実体は未知である代りにその実体が一定の環境の下で示す機能が与えられ，そのような機能を発揮する実体を求めるもので，合成型問題と呼ぶことにする．可能性としては第3の型として実体とその機能が与えられ，実体がそのような機能を発揮する環境（未知）を求めるものがある．これを環境推定型問題と呼んでおこう．

例として（航空機の）翼という「実体」wが一定の環境eのもとで発生する「機能」としての空気力（「揚力」）lの関係で考えよう．「揚力」という特性以外にはこの実体はさまざまな特性，たとえば「抗力」，「モーメント」その他を有するから，記述に際して「揚力」という特性項目の指定が必要である．これを述語形式で，

LIFT (w, e, l)；実体（翼）wの，環境eの下での単位面積あたりの揚力（LIFT）の値はlである

と表すことにしよう．eは問題に応じて異なるが，一般にはベクトルである．この例では$e=(u, \rho, \alpha)$とする．ここでu；空気流速，ρ；空気密度，α：流れに対する翼の角度（迎え角）を表す．

この形式はかなり一般的なものであり，この述語内のw, e, lの3項目のうちのどれが未知量であるかによって上記の3つの型の問題が定まり，その未知量を求める手順は問題の型ごとにまったく異なってくる．しかしいずれの型であれ，計算機が問題解決を支援し得るためにはこれを表現できねばならない．特に対象のモデルとして構造を持つ実体が，環境eのもとでこの述語で表される機能（特性）を持つという形式の表現が可能でなければならない．

このように考えると，対象モデルの表現手段を確立しようという，前記データベースからの発展的アプローチは実用上は十分に一般性のある方式を目指しているといえる．

以下ではこのようなモデル化の方式に基づいてデータ構造を考察する．

5. モデル記述のためのデータ構造

宣言型言語の設計条件を前述のように“一般モデルの表現に適しているもの”とすると，データ構造の問題は一般モデルを表現するために必要な構造ということになる．

なお，モデルに基づく問題解決システムではモデルを静的なものとして記述するのみでは不十分であり，それを目標に近づけてゆくプロセスを計算機が支援し得ることが望ましい．しかしこの機能は言語の問題というよりはシステム設計の問題として，ここでは議論しない．ただしこのプロセスはインタラクティブ・プロセスであり，言語はマン・マシン・コミュニケーションに適したものであることが必要で，これは宣言型言語に対するもう一つの要求となるが，この点も本稿の範囲外であり，これも省く．

もとの議論に戻り，宣言型言語のデータ構造に関する条件をここまで絞ることができて始めて具体的な設計に入ることができる．これがボトムアップ・アプローチの効果である．宣言型言語では手続き型言語のようにデータ構造と記述本体の区別が明確でない．いわば言語全体が抽象データ型のようなものであるから，上記のように表現対象を具体化しない限り，設計条件は“あらゆる対象を記述し得ること”という漠然としたものになろう．

ここで次に“一般的なモデルの表現”という問題がある．宣言型の言語は「実体」とその「機能」，すなわち個々の実体の性質や実体どうしの関係，を直接表現するものである．この言語の体系がいかなるものであるかはここでは問わないが，以下の記述の便宜上，述語論理式（以下単に式と言う）を用いるとしよう．

このような言語の性格から，宣言型言語では構造を含めたすべての記述を宣言（この場合は述語）として表現することができる．図-6 に示した構造関係を例にとり，これを立体一面一線一頂点と構造化された構造体と考えよう．面は立体の構成要素，線は面の構成要素，点は線の構成要素であるから，述語として“x は y の構成要素である”を意味する COMPONENT $(y\ x)$ を用いれば立体構造をこの述語で表された個々の関係の組で表される．あるいはこの構造は，立体一面一頂点の階層関係とし，頂点どうしは連結関係にあるものとして表すこともできる．このとき，連結関係を述語 CONNECTED (xy)：“x と y は連結している”で表すものとすると，立体は2種類の述語で表された関係表現の組で表される．

モデルとしては問題解決に必要なすべての情報を保有する必要があり，これらを加えて，たとえば図-7

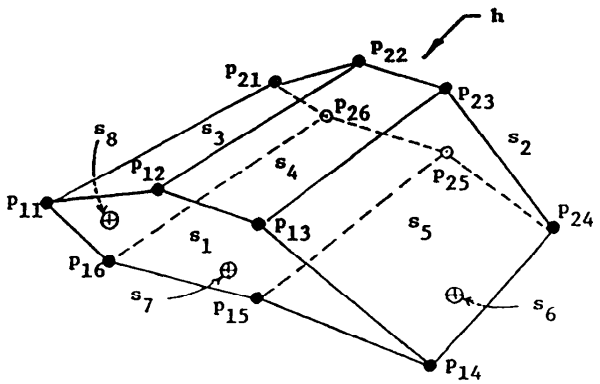


図-6 対象モデル構造

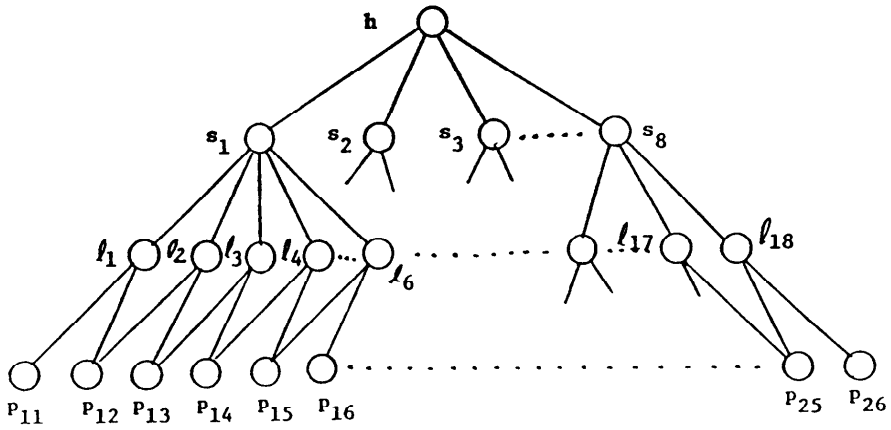
SHAPE $(m\ h)$：（対象）m の形状は h である
COMPONENT $(h\ s_i)$：s_i, $i=1, 2, \cdots, r$ は h の構造要素である
COMPONENT $(s_k\ p_{ij})$：p_{ij}, $i=1, 2, \cdots, m$, $j=1, 2, \cdots, n$ は s_k, $k=1, 2, \cdots, r$ の構造要素である
CONNECTED $(p_{ij}\ p_{kl})$：p_{ij}, $i=1, 2, \cdots, m$, $j=1, 2, \cdots, n$ は p_{kl}, $k=1, 2, \cdots, m$, $l=1, 2, \cdots, n$ と連結している
COORDINATE $(p_{ij}\ x_{ij}\ y_{ij}\ z_{ij})$：$p_{ij}, i=1, 2, \cdots, m, j=1, 2, \cdots, n$ の座標値は $(x_{ij}\ y_{ij}\ z_{ij})$ である
AREA(s_{100})：s_3 の面積は 50 cm² である
ANGLE$(s_3\ s_8\ 30)$：s_3 と s_8 間の角度は 30° である
$(\exists x/h)(\forall y/x)$ DISTANCE $(y\ p\ 5)$：h の中にはある面 x がありその x 内の全の頂点は p 点から5 cm の距離にある（そのような面 x がある）

図-7 モデル記述の一例

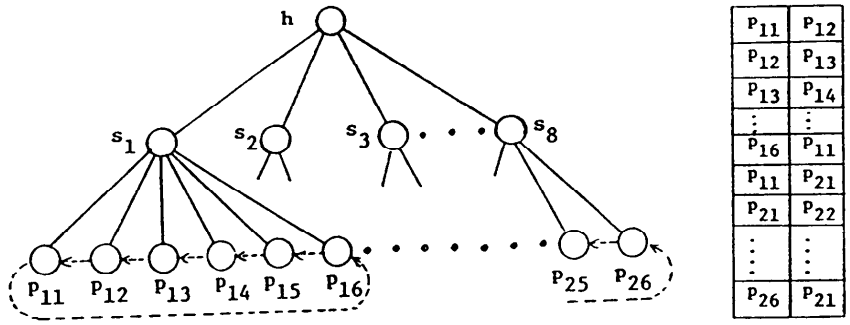
のような述語式の組で表される．このように原理的には構造を含めたすべての記述を同一の形式で行える点が宣言型言語の一つの特徴であるが，このような表現方法は人にとっての理解のしやすさ，処理の効率の両面で非現実的なものである．実世界において人が対象モデルを記述する場合でも，（狭義の）言語のほかに図形その他，構造を直接表現する手段を用いているのは，それが少なくとも（構造の表現に関しては）理解と処理効率が著しく優れているからである．図-7 のモデルの記述例において，構造部分の表現に関しては述語 COMPONENT と CONNECTED をデータ構造要素として図-8 のような表現が可能である．さらにモデルとしてはこのように構造を持った実体に関してさまざまな記述がなされるから，この構造全体あるいはその部分に関するさまざまな「機能」（実体の性質や実体間の関係）を表す述語が与えられている．この述語に含まれる実体は構造内のノードで表されているから，各述語が関係するノードに連結した図-9 のような構造に至る．これは抽象データ型（構造）の一種であるが，特に階層構造がモデル記述に重要なことからこれを抽象階層構造と呼んでおこう[20),21)]．

抽象階層構造には COMPONENT と CONNECTED で表される2種類の構造要素が含まれ，これらは階層構造とグラフ構造を形成する．これらがモデル記述における基本構造として必要であり，かつ有効であることは明らかであるが，これで十分か否かはいまだ明らかではない．今後，多くの問題のモデル化を通してこの問題を明らかにしてゆくことが必要である．

一方，任意の構造要素を宣言型言語に含め得るわけではない．宣言型言語による表現は直接



(a)



(b)

図-8 階層構造とグラフ構造

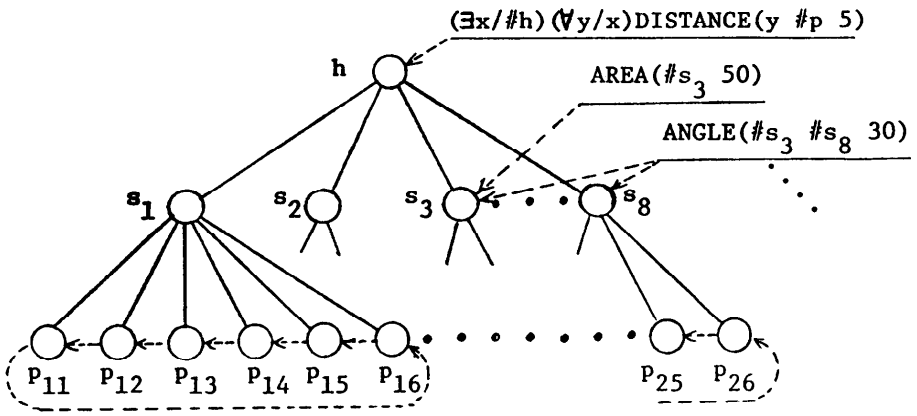


図-9 抽象階層構造

ハードウェアを駆動するわけにはゆかず，最終的には手続き型言語に変換されねばならないし，宣言型の特徴として，この型内での表現の変換，すなわち推論が可能であり，かつ必要でもある．構造表現を許すためにはその構造に基づいて上記変換のアルゴリズムを作ることが可能でなくてはならない．宣言型言語として述語論理を選んだとき，階層構造とグラフ構造は上記の意味で言語内に含めることが可能であることが示され，したがって抽象階層構造までは実現可能である．

とは言え抽象階層構造の記述力は十分高く，これを

適切に用いれば実用上ほとんど支障ないように思える．抽象階層構造を用いてさまざまな問題のモデルを表すことは興味深い．われわれは抽象階層構造を含む述語論理言語を宣言型言語（知識表現言語）としたシステムの開発とともに，これによる各種のモデルの表現，たとえば機械部品の構造，化学分子構造，制御系その他で用いられるブロック図，そしてソフトウェアのモデルなどの記述を試み，あるいは検討している．これらを示すにはシステムのより詳細な記述を必要とするので，ここには記載しない．いずれ別の機会に述べたい．

6. むすび

データ構造の問題を情報の意味の側面から考察した．データ構造はプログラムの一部として定義されるが，従来の言語のもとでは機械に依存する部分，操作型（手続き的）意味記述のための補助的役割を担う部分，問題の表現に直接対応している部分が混在していた．このためデータ構造の意味理解がしにくく，プログラム合成などの新しい試みの際でも障害の一つになっていた．

ここに新しい視点を導入したのが抽象データ構造とデータ・モデルの考え方で，いずれもデータに対する統一的な意味づけを可能にするものであった．

一方，言語自体が，操作型（手続き型）から宣言型（非手続き型）に移行しようとする情勢にあり，このときのデータ構造の定義が変化してくる．本稿では宣言型言語におけるデータ構造を対象モデル表現に必要な（あるいは便利な）構造概念と定義し，階層構造とグラフ構造が有用であると同時に処理可能なものであることを述べた．対象モデルの概念は問題解決の重要な方法の一つであるが，未解明の部分も多く，今後の研究にまつところが大きい．

参考文献

1) Berge, C.: Graphs and Hypergraphs, North-Holand (1973).
2) Berztiss, A. T.: Data Structures, Theory and Practice, Academic Press Inc. (1971).
3) Brodie, M. L., Mylopoulos, J. and Schmidt, J. W. (eds.): On Conceptual Modelling-Perspectires from Artificial Intelligence, Databases and Programming Languages, Springer-Verlag (1984).
4) Chang, C. L. and Lee, R. C. T.: Symbolic Logic and Mechanical Theorem Proving, Academic Press (1972).
5) Claybrook, B. G.: A Facility for Defining and Manipulating Generalized Data Structures, ACM Trans. on Database Systems, Vol. 2, No. 4, pp. 370-406 (1977).
6) Data, C. J.: An Introduction to Data Base System, Adison Wesley (1976).
7) Davis, R. and Lenat, D.: Knowledge Based Systems in Artificial Intelligence, McGraw-Hill (1982).
8) Encarnacao, J. and Krause, F. L. (eds.): File Structures and Data Bases for CAD, North-Holland (1982).
9) Enderton, H. B.: Mathematical Introduction to Logic, Academic Press (1972).
10) Gallaire, H. and Minker, J. (eds.): Logic and Databases, Plenum Pub. Co. (1978).
11) Gallaire, H., Minker, J. and Nicolas, J. M.: Logic and Databases: A Deductive Approach Computing Surverys, Vol. 16, No. 2 pp. 153-185 (June 1984).
12) Goldberg, A. and Robson, D.: Smalltalk-80, The Language and its Implementation, Adison-Wesley Pub. Co. (1983).
13) Gries, D. and Gehani, N.: Some Ideas on Data Types in High-Level Languages, Comm. ACM, Vol. 20, No. 6, pp 414-420 (June 1977).
14) Guttag, J.: Abstract Data Types and the Development of Data Structures, CACM (June 1977).
15) Guttman, A. and Stonebraker, M.: Using a Relational Dtatabase Management System for Computer Aided Design Data, Data Base Engineering (June 1982).
16) Harrison, M. C.: Data Structures and Programming, Scott, Foresman & Co. (1973).
17) Haskings, R. and Lorie, R.: On Extending the Fanctions of a Relasional/Database System, Proc. 1982 ACM-SIGMOD Conf. on Management of Data, Orlando, FL (June 1982).
18) Knuth, D. E.: Fundamental Algorithm, The Art of Computer Programming, Vol. 1, Addison-Wesley Pub. Co. (1973).
19) Ledgard, H.F. and Taylor, R.W.: Two Views of Data Abstraction, Comm. ACM, Vol. 20, No. 6, pp. 382-384 (June 1977).
20) Liskov, B., Snyder, A., Atkinson, R. and Schaffert, C.: Abstraction Mechanisms in CLU Comm. ACM, Vol. 20, No. 8, pp. 564-576 (Aug. 1977).
21) Lockemann, P.C., Mayr, H. C., Weil, W. H. and Wohlleber, W. H.: Data Abstraction for Database Systems, ACM Trans. on Database Systems, Vol. 4, No. 1, pp. 60-75 (1979).

22) Lorie, R. A.: Issues in Database for Design Applications, in File Structure and Data Bases for CAD (eds. Encarnacao, J. and Krause, F. L.), North-Holland Pub. Co., IFIP (1982).
23) Lum, V., Dadam, P., Erbe, R., Guenauer, J., Pistor, P., Walch, G., Werner, H. and Woolfill, J.: Design of an Integrated DBMS to Support Advanced Applications, Proc. Int. Conf. on Foundations of Data Organization, Kyoto, pp. 21-31 (May 1985).
24) McLeod, D.: Abstraction in Database, Proc. of ACM Workshop on Data Abstraction, Database and Conceptual Modelling, pp. 19-25 (June 1980).
25) Ohsuga, S.: Perspectives on New Computer Systems of the Next Generation -- A Proposal for Knowledge-Based Systems, J. of Information Processing, No. 3, pp. 171-185 (1980).
26) Ohsuga, S.: A New Method of Model Description -- Use of Knowledge Base and Inference, in CAD System Framework (ed. K. Bo and F. M. Lillehagen), North-Holland, pp. 285-312 (1983).
27) Ohsuga, S.: Predicate Logic Involving Data Structure as a Knowledge Representation Language, Proc. '83 Eight Int. Joint Conf. on Artificial Intelligence, pp. 391-394 (1983).
28) Ohsuga, S.: A Consideration to Knowledge Representation -- An Information Theoretic View, Bulletin of Informatics and Cybernetics, Vol. 21, No. 1-2, pp. 121-135 (1984).
29) Rowe, L. A.: Data Abstraction from Programming Language Viewpoint, Proc. Workshop on Data Abstraction, Databases and Conceptual Modelling, pp. 29-39 (June 1980).
30) Samet, H.: The Quadtree and Related Hierarchical Data Structure, Computing Surveys. Vol. 16, No. 2, pp. 187-260 (June 1984).
31) Stonebraker, M., Rubenstein, B. and Guttman, A.: Application of Abstract Data Types and Abstract Indices to CAD Data Bases, Proc. Engineering Design Applications of ACm-IEEE Data Base Week, pp. 107-113 (1983).
32) Van Ender, M. H. and Maiddaum, T. S. E.: Equations Compared with Clauses for Specification of Abstract Data Types, Advances in Database Theory, Vol. 1, Plenum Pub. Co., pp. 159-193 (1981).
33) Williams, R.: A Survey of Data Structures for Computer Graphics Systems, Computing Surveys, Vol. 3, No. 1 (Mar. 1971).
34) Wirth, N.: Algorithm+Data Structure=Programs, Prentice-Hall Inc. (1976).
35) Yamauchi, H. and Ohsuga, S.: KAUS as a Tool for Model Building and Evaluation (To appear in Proc. 5th International Workshop on Expert Systems and Their Applications, to be held in Avignon, France, May 13-15, 1985).
36) Yazdanian, K.: Use of the Relational Model for Data Representation in a Deductive Augmented Data Base Management System, Proc Int. Conf. on Foundations of Data Organization, Kyoto, pp. 32-38 (May 1985).
37) Zilles, S. N.: Types, Algebras and Modelling, Proc. of workshop on Data Abstraction, Database and Conceptual Modelling, pp. 207-209 (1980).
38) 大須賀節雄：データ構造, bit. Vol. 8, No. 5～Vol. 8, No. 13 (1976).

(昭和60年11月11日受付)