ソフトウェア基礎論　17－15
（1986. 7. 4）

モデルに基づくプログラム合成方式の基本思想と合成過程

BASIC THOUGHT AND SYNTHESIS PROCESS OF THE MODEL-BASED PROGRAM SYNTHESIS METHOD

間野　暢興
Nobuoki MANO
電子技術総合研究所
ElectroTechnical Laboratory

あらまし　本論文では、データ処理ソフトウエアに関する、要求定義、仕様、プログラム、および知識のすべてを統一したモデルの形式で表わす．それらは、主に対象物およびそれらの間の関係（およびそれらの属性）を中心に記述される．データ集合対象物の使用により、仕様記述は簡潔に表わされ、仕様とプログラムの対応付けが可能となる．モデルの操作には、宣言的規則も使用されるが、モデルの構造をアルゴリズミックに辿る手続き的処理が活用される．その合成過程は、出力仕様から入力仕様へのゴール指向的問題解決による分析過程、および、その結果の整構造プログラムモデルへの変換組み立て過程、の２段階からなる．この特性を利用したプログラム合成方式は、ファイル処理、入力のフィルタリングとデータ構造の更新、複合データ構造処理などの問題に適用される．

Abstract　We represent all of the information about data-processing software -- from requirements definition, specification, program, and knowledge -- by an unified model which is desribed mainly by objects and relationships between them. Use of data-set object makes specification compact and corresponding representaion between specification and program possible. Synthesis process consists of two phases -- analysis process by goal-oriented problem-solving process and program construcion process by transforming the analysis result. We apply this method to problems such as file processing, input filtering with data strucure updating, and processing compound data strucures.

１．　はじめに

一般に、「モデル」という言葉は、対象とする問題の世界の性質を（特にその構造に注目して）抽象化概念化して表現したものを指し、問題の性質を研究したり問題を解く際に利用される．ＣＡＤや人工知能では昔から計算機内に対象のモデルを形成しそれを用いて問題を解くことが行なわれてきた．プログラムの生成（あるいは合成）においても最近その重要性が徐々に認識されてきている．しかし、現存するシステムでは、データ処理全般に拡張可能な十分一般性のある形式化は未だなされていない．筆者は、ファイル処理に関して、意味モデルと呼ぶ一般性のある基礎的なモデルに基づくプログラム合成方式を発表してきた[1],[2],[3],[4] が、紙面の都合もあり、何故このような表現形式をとるのかという説明が十分行なえなかった．第２章では、モデルの表現形式について、そのような形式を考えるようになった経緯を踏まえて、その記述力と一般性を説明する．意味モデルは、述語論理や集合式の表現を基礎とし、主に対象物(object)とそれらの間の関係(relationship)およびそれらの属性により記述される（この場合の「対象物」とは、最近のメッセージの受渡しを主体とするオブジェクト指向言語の対象物と同一ではないことに注意）．計算機の内部ではモデルは人工知能におけるフレーム(frame) の表現形式で表わされるが、説明のためには簡易化されたラベル付き有向グラフの図式表現（意味ネットワークの表現）を用いる．意味モデルは、事実や規則を宣言的に表現できるだけでなく、データ構造内に表現された情報の構造をアルゴリズミックに辿ることにより手続き的な処理が可能となる．これは一昔前人工知能の分野で交わされた論争における結論：－宣言的知識と手続き的知識のいずれも必要－ということに対応している．第３章では、このモデルに基づくプログラム合成方式の基本原理を述べ、モデルの構造表現機能がいかに活用されるかをInsertion-sortおよび複合データ構造の例題を用いて説明する．入出力データの構造記述からの変換プログラムの合成例などは、参考文献を参照されたい．

２．　意味モデルの表現形式

２．１　一階述語論理式の図式表現（図１）

最初に述語論理式の表現法を定める．一般に、関係（述語）を英小文字ラベル付きの黒点、その引数を○で表わすと、ｎ項関係は同図（ａ）のように表わされる．二項関係は、関係の黒丸は省略しエッジは直線で表わす．整式（英大文字で表わす）は、一般に、部分グラフを内に持つ部分空間で表わされる．ＯＲはＯＲノードで示される（ｃ）．グラフは一般に＆を表わすので＆記号は省略する（ｄ）．存在限量子は省略しその束縛変数はスコーレム定数（図では？を冠頭に持つ）で表わす（ｅ）．全称限量子を含む式は、結合子EQUIV あるいはIMPLY により結合される前提、帰結部分の各部分空間に、共通の？を冠頭に持つ変数で束縛変数を表わす（ｆ）．

２．２　データ集合およびデータ集合間の対応関係（図２）

（ａ）に示すように、具体データA1,A2, およびB1,B2, 間の関係の集まりを、それぞれが属する定義域ａおよびｂのラベルを持つデータ集合AおよびB間の単一の関係として表わす．このとき一方のデータ集合A（B）の一つの要素に対応する他方の集合B（A）の要素数の最小値min および最大値max を、AからB（BからA）への矢印のついた[min,max] の形式の対応関係で表わす．AからB、BからAへの順で対応を示すものとして、AB間の対応が1対1対応、および多対1対応の場合は、それぞれ[1,1] と[1,1] 、および[1,1] と[,] 、BがAの部分集合の場合は[0,1] と[1,1] 、となる．max が のときは、データ集合が無限集合の場合無限、有限集合の場合不定個、を表わす．[c,c] は丁度ｃ個ということを表わす．これは、集合の基数や、配列などのデータの個数を表わす場合にも使用される．ループ不変量、データ型不変量もこの対応関係を用いて簡潔に表現できる（図１０参照）．

データ集合は、意味モデルにおける対象物の型の一つである．意味モデルでは、対象物および関係の内部表現は、（ｂ）に示すように、フレーム表現を用いて表わす．これは図式表現（ｃ）の左側の図に対応するが、図を用いて説明するときは右の簡略化した図を用いる．ここで、in-rels およびout-relsのスロットは、その対象物に入って来る関係および出て行く関係を表わす．関係x-ofは、一般に関係x の逆の関係を表わす．個々の関係はユニークな名前（英大文字）を持つが図示はしない．

なお、データ集合の定義域は、一般には、述語論理式により定義されるが、データ集合を基に定義することもでき

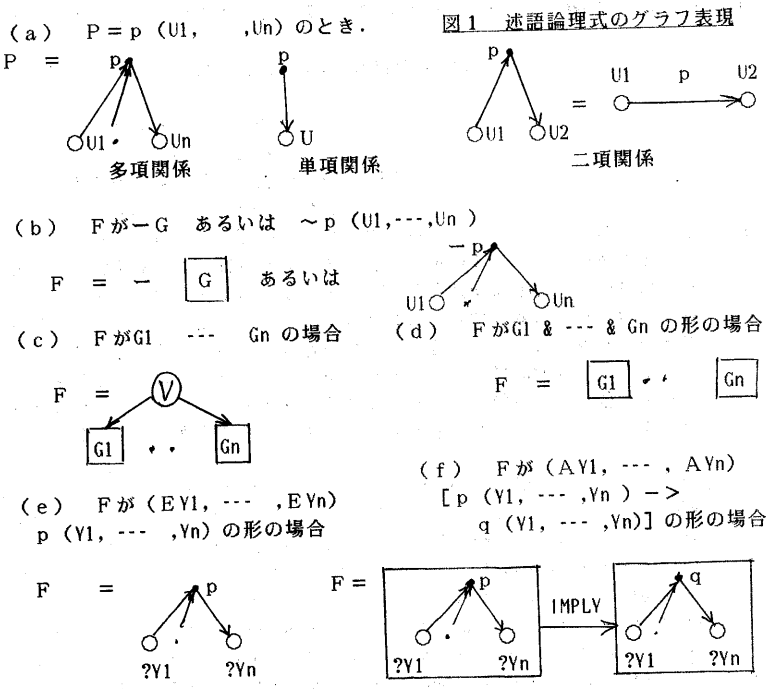


図１ 述語論理式のグラフ表現

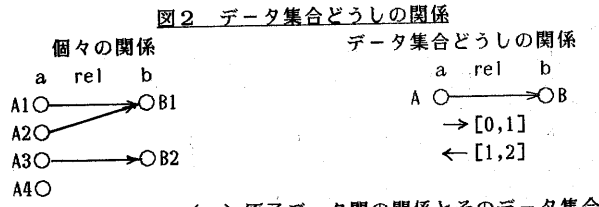


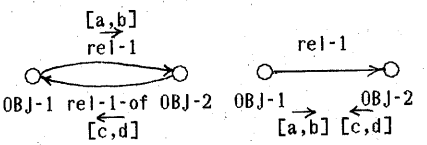
図２ データ集合どうしの関係

（ａ）原子データ間の関係とそのデータ集合どうしの関係

```
(OBJ-1
  (instance-of: obj-1)
  (out-rels (REL-1))
(OBJ-2
  (instance-of: obj-2 )
  (in-rels (REL-1-OF))
(REL-1
  (instance-of: rel-1)
  (cor: [a,b]))
(REL-1-OF
  (instance-of: rel-1-of)
  (cor: [c,d]))
```

（ｂ）データ集合どうしの関係の内部表現

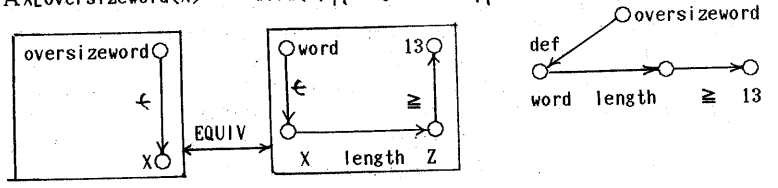
（ｃ）その図式表現



（左を右のように簡略化して表わす）

（ｄ）データ集合の定義

∀X[oversizeword(X) <=> word(X)∧length(X,Z) ∧Z≧13]

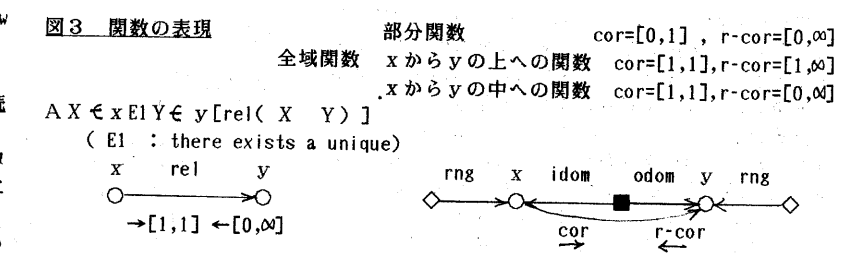


る．例えば、oversize-word の定義の例を（ｄ）に示す．

２．３　関数および手続きの表現（図３）

意味モデルでは、プログラムを記述するためには、対象物の型として、算譜構成要素（図において■の印を用いる）、状態（図においては部分グラフを含む長方形で表わす）、および、状態の記述において変数などを表わす識別子（図において◇の印を用いる）、を導入する．入出力仕様記述には、述語論理の記法を用いるのではなく、データ集合とその間の関係表現を用いる．関数あるいは手続きを表わす算譜構成要素は、関係idomおよびodomによりそれぞれ入力および出力のデータ集合と結ばれる．データ集合と変数とは関係rng で結ばれる（データの場合は、関係valで結ばれる）．関数の場合、図３に示すように入力と出力のデータ集合間の対応関係を用いて、その特性を定義することができる．このように、データ集合を用いて入出力を表わすことは、後で示すように、仕様の記述が簡潔化するだけでなく、データの構造の陽な記述、入出力データ要素間の対応付け、入力データの要素とプログラム（の構造）を構成する要素（算譜構成要素）との対応付け、などが可能となる利点がある．

図３　関数の表現



状態の記述は、副作用のないプログラムの場合、識別子対象物（変数）とデータ対象物との結びつきを用いた表明で記述される．命令型（副作用利用）プログラムの場合には、さらに場所という型を導入し、その対象物を介して識別子対象物（変数）とデータ対象物の結合を表わすところが関数型プログラムの場合と異なる[3]．

関数や手続きのプログラムの内部構造は、繰り返し文型、判定文型、連接文型の制御構造の算譜構成要素を基本とし、関数や手続きの適用を使用して構成される（図５、図６参照）．

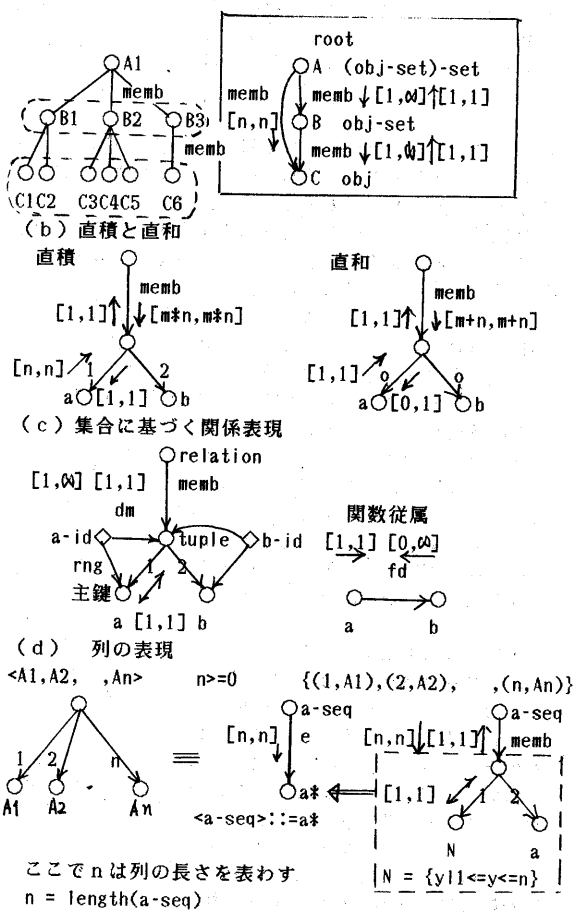
２．４　概念世界と現実世界

このような、対象物と関係中心の表現により全ての情報を統一的に表現する長所として、概念の上位下位構造と自然に整合することが挙げられる．この上位下位構造には、対象物の型に関するもの、関係に関するもの、操作に関するものなどがある．知識ベースにある概念の構造は、実際の問題を反映した現実世界の概念構造に繋がっている．これらに関しては、他の論文に書いたので、ここでは省略する．

実際の問題の記述に用いられる対象物は概念世界にあるgeneric 対象物の写し（instance）である．抽象レベルで実際問題のデータ構造を考える場合には、実体関連(Entity-Relationship) モデルの見方が役立つ．データの構造を正規文法表現で表わした場合、非終端記号は実体に相当する．実体の各要素は一つ一つがユニークな固有名を持つ．また終端記号は、実体を表わす場合と属性を表わす場合とがある．たとえば、電信文の問題において、各電報に出て来る単語の一つ一つは別個の実体要素で、その属性としてラベルを持つと考えられる．属性の値の集合は有限集合の場合と無限集合の場合とがある．

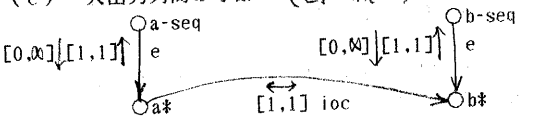
２．５　抽象データ型の構造記述（図４）

図４　種々のデータのモデル表現

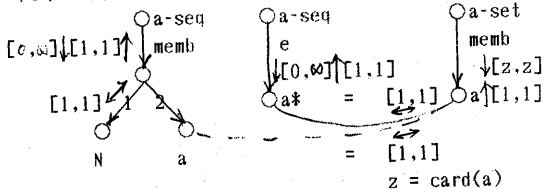


集合の基数(cardinal number)は次のように親子関係を基本として決められる．データ構造の骨格を表わす木の根となる集合は、部分空間から関係rootで指される単元からなる（即ち基数１の）集合である．その単元集合から対応 [n,n]の関係membで指されるデータ集合は基数ｎの集合を表わす．さらにこのノードから対応 [m,m]の関係membで指されるデータ集合は、基数ｎ＊ｍの集合である．このように、現実のデータ集合は、親子関係を辿ることによりその基数が決められる．これらの関係において、逆向きの対応[1,1] は、その集合の元には必ずユニークな親が存在ことを表わしている．（ａ）はobj の集合の同値類への分割を表わす．また、関数の入出力仕様において、入力データ集合と出力データ集合の間に等号の関係（１対１となる）あるいは部分集合の対応関係があるときは、前者はデータの保存、後者はフィルタリングの計算を表わしている．

（ｅ）　入出力列間の写像　（図千鏡き）



（ｆ）　列と集合の関連



それぞれ基数m、ｎのデータ集合ａ，ｂの直積および直和は、（ｂ）に示すように表わされる．これらは、連接および選択を表わす分岐を含んでいる．

関係(relation)は（ｃ）図に示すように、異なるtuple の集合として集合を基に表わすことができる（直積は関係表現の一種である）．関係に付随した概念として主鍵および関数従属の概念が登場する．主鍵のデータ集合はその関係のtuple の集合と１対１に対応する．欄a-idの指すデータ集合の要素一つを定めると欄b-idの指すデータ集合の要素が一つ定まるとき、欄b-idは欄a-idに関数従属であるという．

列(sequence)は、以下に示すように集合を基に表わすことができる．集合ａの要素を用いる長さｎの列<A1,A2, ,An> は、　A1,A2, ,An　（ただし、AiはAjと同じでもよい）

1　2　n

のように、N={x　1<=x<=n}と１対１に対応をつけることにより、{(1,A1),(2,A2),　,(n,An)}と表わすことができる．これは図（ｄ）に示すように、集合１からｎまでの整数の集合Ｎと集合ａのtuple からなる基数ｎの集合として表現される．図のように、三つのノードを一つのノードにまとめ、関係をｅとつけ直し、その対応関係は、membのそれをそのまま引き継いで（[n,n] [1,1] となる、[1,1] は列の要素にはユニークな親が存在することを意味する）表わすことにする．正規表現の<a-seq> ::= <a> （あるいは、<a-seq> ::= <a> ）を表わすには、関係ｅの対応は[0,] [1,1] （あるいは [1,] [1,1]）となる．入出力の列のノードを表わすtuple 間に１対１の対応が成り立つ場合、写像の関係として関係input-ouput-correspondence（ioc と略す）を用いる（ｅ）．列とその要素の集まりからなる集合との間には、（ｆ）のように、等号関係（１対１の対応）が付られる．

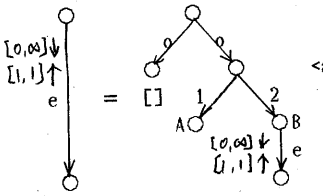
２．６　再帰構造とグラフ処理

入力のデータ構造と対応するプログラムの構造とは、それぞれの対応するノードを関係input-program-correspondence（ipc)により対応付ける．対応付けられた算譜構成要素と、その入力データの前始末、対応する始末、後始末を行なう算譜構成要素とは、各々関係before(bf)、at、after により結ばれる．

（１）　リスト構造（図５）

再帰データ構造は、要素間に順序があるので、やはり列を表わす関係（ｅなど）を用いて表わされる．長さｎの列のデータ構造（ｎ＞＝０）は、（ａ）図のようにＢＮＦ表現に対応した再帰形のモデル表現で表わすことができる．３．２でその合成を述べるinsertion-sortのプログラムのsortの場合、問題解決により得られた結果を入力の

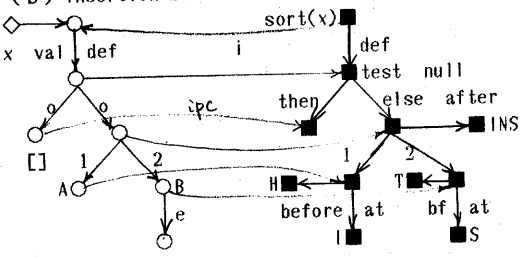
図５　再帰表現



<a-seq> ::=<a>* ::= [] | <a>.<a>*
::= [] | <a>.<a-seq>

（ａ）繰り返しデータ構造とその再帰データ構造

（ｂ）Insertion Sortにおける入力データ構造とプログラム構造の対応



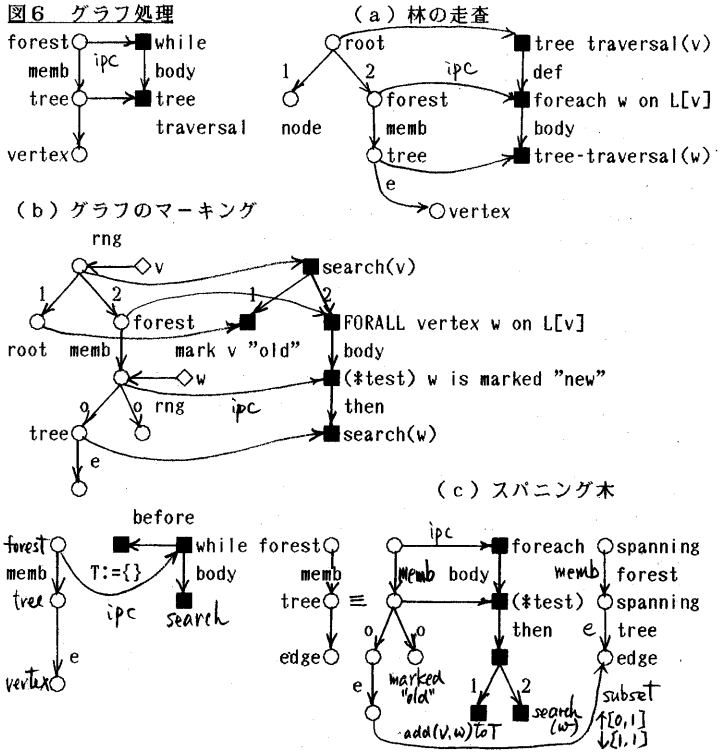
再帰形に対応させてそのプログラムの木を形成すると、（ｂ）のようになる．

（２）　グラフ（図６）

グラフ処理のアルゴリズムの場合、木の走査プログラム－＞頂点のマーキングおよび縦型探索－＞強連結および二連結のアルゴリズム、とその発展がなされていく．

（ａ）図は、木をその頂点と林（それは再帰的に木の繰り返しとなる）からなるデータ表現で表わし、エッジの両端のノードを別別の変数で指す．グラフのマーキングは、（ａ）図に、①場合のあてはめ、②新しい場合分けの導入による基本となるアルゴリズム上の路の条件化（マークが"new"かどうか、など）、③必要な操作の追加、を施すことで新しいアルゴリズムを作り出している（ｂ）．スパニング木を求めるアルゴリズムは、この頂点のマーキングのプログラムの判定文とエッジのフィルタリングプログラムの（未決定の）判定文を同一化することにより得られる（ｃ）．強連結および二連結のアルゴリズムでは、スタックなどのデータ構造、部分木の属性の導入、などを行ない、入力部分木の後始末（および前始末）に新に必要な操作文や判定文の追加を行なって、アルゴリズムが作り出される．

図６　グラフ処理



２．７　問題の構造と仕様

ユーザは、図７に示すような形式で、問題の型に応じてその構造と仕様を与える．

（ａ）変換問題

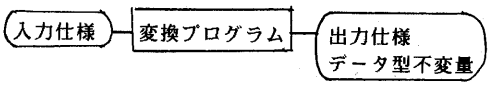
入力を出力に変換する問題で、ジャクソン法の対象とする問題（構造変換、属性計算、構造変換＋属性計算）、および関数型のプログラムを合成する問題、などがある．後者の例としては、２．６および３．２で述べるinsertion-sortの例題がある．

（ｂ）フィルタリング＋更新問題

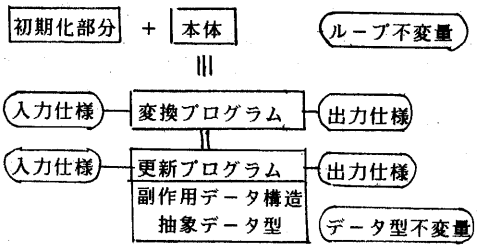
本体部分と初期設定部分の二つの処理単位に分かれる．本体部分は、ゴールは複数のサブゴールからなる．出力は入力をフィルタリングして得られる（プログラム①）．またデータ構造の内容の更新を行なう（プログラム②）．出力とデータ構造の間の関係を表わすループ不変量の関係が存在する．初期設定部分では、データ構造の初期設定を行なう．例題としては、いずれもグラフのスパニング木を求める問題として次のものがある．集合の観点によるアルゴリズム[3]や、縦型探索とマーキングによるものが存在する（２．６参照）．

図７　問題の構造と仕様の与え方

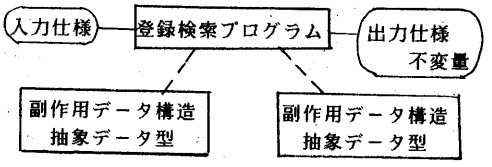
（ａ）　変換問題



（ｂ）　変換（フィルタリング）と副作用データ構造更新



（ｃ）　副作用複合データ構造登録検索



（ｃ）　複合データ構造問題

直積結合している複数のデータ構造や、データ構造名自身をデータ要素としてもつ場合で、外部仕様あるいはシステム仕様に基づいて指定された関数あるいは操作の内容を合成する．データ構造どうしの欄間の意味的結合を表わす不変量関係を指定する．３．２に簡単な例題を示す．

図７でユーザが与える情報としては、次のものがある．

①　入力仕様では、入力の構造とその拘束条件．

②　出力仕様では、出力の構造と拘束条件および出力の要素の入力の要素との対応．

③　副作用利用データ構造の抽象データ型の定義．

④　複合データ構造の場合、それらの欄の意味的結合により成立する不変量．

３．　モデルに基づくプログラム合成方式

３．１　合成過程の概要

２．７で述べたように、問題には、その構造および不変量関係の有無、用いられる仕様の性質の相違などに基づく特性があり、それを無視した一様な方法で扱えるものとは思われない．しかしほとんどすべての問題において適用可能な指導原理が存在するものと思われる．それは、図８に示すように、二つのフェーズからなる．

（１）　ゴール指向的な問題解決による問題の分析フェーズでは、出力仕様をプログラムによって満たされるべきゴールと考える．これは通常、同時に満たされるべき複数の目的（すなわちサブゴール）の集まったものから構成されている．入力仕様はその際に前提となる条件を表わす．このゴール指向的な問題解決は、変換問題における処理構造の分割、処理単位のプログラム合成（ジャクソン法の自動化）、関係データ処理、再帰関数の合成、実現変換後の具体レベルにおけるプログラム合成（場所対象物を用いるアルゴリズムはここでは述べない）、などの場面で適用される．プログラムの文とそれによる状態の変化というHoare のプログラム論理の考え方は、関数型命令型の区別なく使用できる．

（２）　その成功した問題解決による分析結果は、変換組み立てフェーズにより、逐次化されたプログラム（のモデル）に、まとめ上げられる．ここでは処理構造のある問題の扱いは省略する．

３．２　問題解決の原理（分析フェーズ）

サブゴールへの分解と個々のサブゴールの関数や操作の適用による入力仕様への還元が試行錯誤的に行なわれる．以下にその要点を述べる．

（１）　ゴールは、データと述語よりなる．

１）　データによる分類

入出力共通にあるデータか、出力（途中の状態を含む）だけにあるデータかを区別する．前者はそのまま入力方向へ伝える．後者は関数やスキーマを適用して還元を計る．サブゴールを解決しようとして適用の結果新たに発生したサブゴールのデータは、属性文法における合成属性および継承属性のパラメータに相当する．

２）　述語の種類による分類

①入力（構造）要素と出力（構造）要素の対応を表わすもの．

②入力あるいは出力内の構造を表わす関係．

③入力あるいは出力内におけるデータの大小関係、あるいは、sortプログラムにおけるord のような集合的データの属性などの内部拘束条件．

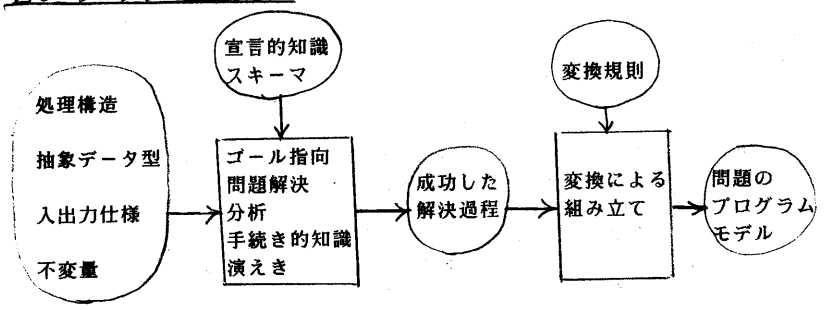
（２）　サブゴールの還元を行なうための候補の取り出し．

出力仕様の（部分）整合するものが候補の関数あるいは手続きとして取り出される．整合にはデータ内構造関係の整合が第一に使われる．

（３）作業領域への展開

操作プログラムや関数の定義の写しをそのまま作業領域に作る．そして展開されたものの出力仕様部分と問題出力仕様部分、展開されたものどうしの出力仕様部分と入力仕様部分、あるいは、展開されたものの入力仕様部

図８　プログラム合成の過程



分と問題入力仕様部分との整合を付ける（その対応する束縛をそれ用の特別な部分空間に保持する）．なお図では複雑になるので、図９では簡単化したものを示した．これにより、データ内構造による展開は扱える．

（４）　問題入力仕様との整合、入力の仮定の充足検証．

展開毎に、その展開の入力側が問題入力と整合するかどうかのチェックは必ず行なう．これにより無駄な探索を防ぐことができる．状態が完全に問題入力仕様と一致すれば問題解決は終了する．整合が完全にとれない限りは、中間状態であると考える．

（５）　データ間関係の伝播、集合属性の伝播、

元の状態における入出力間関係は、新たに展開された状態における要素と問題入力要素との対応として付け直す．データ内の大小関係、集合データ属性などの拘束条件は新に展開された状態においてそれが関連する要素があれば伝播させることができる．

（６）レベルとフェーズの扱い

データの構造の途中のノードから根へ枝を辿るとき、途中出会う関係membあるいはe の回数をそのノードのレベルという．これによりそのノードを処理する算譜構成要素が何重のループの中にいるかが分かる．サブゴールの管理は、今処理しているレベルのもの、もう一つ先の状態で、同レベルのもの、一つ下のレベルのもの（後始末は、一般にそれより下の部分木の処理し、場合によってはさらにその結果を上方に伝播するためのもので、属性文法の合成属性に相当するデータフローを扱う）、それより上のレベルのもの（属性文法の相続属性のように、上方からのデータフローをもたらす）、と分けて管理される．レベルとフェーズ（異なった処理単位の導入が必要な場合に新しいフェーズになる）は複数のサブゴールがあるとき揃えて進行していく．

再帰を含む変換プログラムの例題）　Insertion Sortプログラム（図９）

ここでは、まず、関数sortの入出力がnonnull の場合を述べる．出力のリストを一つの要素と残りの部分に分ける操作を適用する．出力リストのord 属性はその部分リストにもあてはまるので、それを操作の適用前の状態１にも伝播させることができる．また出力における各データ要素の入力の要素との対応関係は、状態１におけるデータ要素と入力の要素との対応関係に書き直される（図では複雑になるので省略した）．これにより操作はその入出力仕様が決まり、副問題insertとなる．状態１では、求める関数sortの出力仕様が再度適用可能で、その結果は状態２になる．この状態に、関数headおよび tail を適用すると問題入力に還元がなさる．関数insertの問題解決においては、両方の入力がnonnull の場合には、二つの入力の要素間の大小関係により、二通りの場合分けが生ずる．

複合データ構造の例題（電子メール問題）（図１０）

外部ユーザ関数の入出力仕様が与えられた場合、その求める出力データから内部で使用するデータ構造の欄の意味的繋がりを経て（たとえば、この問題の場合、mail-drop はregistryのデータであり、同時に表の集合でもある）問題入力迄辿ることにより、各データ構造において必要とする操作を見いだす．この操作の行なう内容を与えるプログラムは、そのデータ構造の主鍵の指定や関数従属などの拘束条件の記述を知識ベースにあるスキーマにあてはめることにより、作り出すことができる．

３．３　変換組み立ての原理

以下に述べる手続きにより、成功した問題解決により得られた展開結果の変換組み立てを行なって、完全なプログラムの木を作り出す．

（ａ）　変換問題の場合ーーーループの融合

同じ入力木の繰り返しデータに対応する繰り返し文型の算譜構成要素は一つにまとめる．再帰関数の場合も、共通の構文は一つにまとめる．その他の算譜構成要素については、再帰関数あるいは繰り返し構文のプログラムのいずれの場合でも、入力木のデータの前始末、中始末、後始末の位置に算譜構成要素を位置付ける．半順序の関係にある算譜構成要素どうしは、トポロジカルソートにより逐次化する．

出力仕様の順序に注意して編集する．

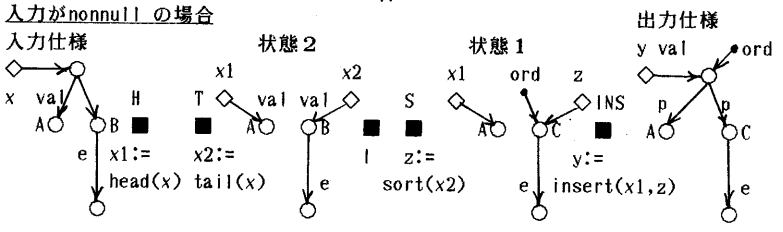
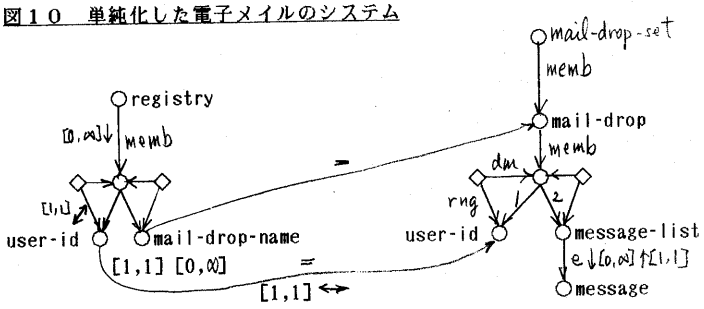


図９　Insertion Sortの合成

（ｂ）　フィルタリングと更新問題の場合－－－ループの融合および判定文の重畳による組み立て

（ａ）とほぼ同様であるが、フィルタリングのプログラムにおける未決定の判定文型の算譜構成要素を、更新のプログラムにおける既定の判定文型の算譜構成要素と同一のものとする．二つの判定文のスコープ下の算譜構成要素は一緒にする．これによりループ不変量は不変に保たれる．

図１０　単純化した電子メイルのシステム



（ｃ）　直積結合の複合データ構造の場合－－－データ構造どうしの欄間の意味的結合（保全性）を表わす関係は、登録などのシステム操作には不変量として働く．

３．４　実現変換と目的言語プログラムの生成

今まで述べて来た合成方式により得られたプログラムのモデルは、抽象レベルのそれであって、実際に計算機で走るプログラム言語のプログラムに落とすためには、実現変換してやることが必要である．これによりその言語が用意している構文やデータ構造操作のプログラム（のモデル）のレベルにまでその表現を直すことができる．目的プログラミング言語ごとに用意されたプログラムコード生成ルーチンで走査することにより、実現変換によって得られた具体レベルのプログラムモデルから目的言語プログラムが生成される．紙面の都合上ここでは述べないが、命令型プログラム言語のプログラムに実現変換するときは、副作用を利用した問題解決（ポインタの回転(rotation)およびスライド(slide)など到達可能性を利用したものを含む）が必要となる．

４．　終わりに

対象物と関係を主に用いて情報をモデル化して表現する意味モデルの基本理念を述べた．またその構造表現機能を活用して、仕様を問題解決により分析し、その結果を変換してプログラムを組み立てるプログラム合成の基本原理について述べた．

本文で述べた合成方式は、フレーム構造をフルに活用するアルゴリズミックなデータ処理手続きを主体としている．システムに内蔵した基本的な部品プログラムに問題向きに手を加えることにより、このシステムプログラム自体を作り出す研究は、挑戦に値する課題であると思われる．そのとき、本論文で述べたグラフ処理や複合データ構造処理プログラムの合成法のアプローチが土台として使えるものと予想される．

最後に、日頃お世話頂く棟上ソフトウエア部長、言語処理研究室および情報システム研究室の方々に感謝致します．

参考文献

[1] 間野暢興：”仕様とプログラムのモデルを用いたプログラム合成方式”、情報処理学会夏のシンポジウム「プログラムの合成、変換、再利用」報告集、pp.11-20（1985）．

[2] 間野暢興：”抽象データ型を用いたプログラムの合成における知識表現と問題解決”、情報処理学会ソフトウエア工学研究会資料46-9　（1986）．

[3] 間野暢興：”仕様とプログラムのモデルに基づく抽象プログラムの実現変換”、電子通信学会オートマトンと言語研究会資料AL85-69（1986）．

[4] 間野暢興：”処理構造を持つ抽象プログラムの合成について”、電子通信学会オートマトンと言語研究会資料AL85-88（1986）．

盛光印刷所