ソフトウェア基礎論 34－1
（1990. 3. 9）

対象物－関係指向知的プログラミング環境における自己記述からのシステム生成

間野 暢興
電子技術総合研究所

意味モデル操作システムは、対象物－関係のモデル（意味モデルと呼ぶ）により表わされた対象問題の全ての情報を操作することにより、ファイル処理およびデータ構造操作を行なう問題のプログラム生成および再利用を計る知的プログラミングシステムである．このような知識に基づくシステムは、対象問題領域の拡張とともにその処理機能の拡張をも必要とする．本論文では、意味モデル操作システムにおける、知識の構造、各種関係の定義、構造体の定義などから、意味モデル管理ルーチン群およびそれらを駆使した各種ツール（直接実行機、記号実行機、書き換え規則機、問題解決機、など）を生成する方式について考察する．

SYSTEM PROGRAM GENERATION BASED ON SELF-DESCRIPTION IN AN OBJECT-RELATIONSHIP-ORIENTED INTELLIGENT PROGRAMMING ENVIRONMENT

Nobuoki Mano

ElectroTechnical Laboratory

1-1-4, Umezono, Tsukuba-shi, Ibaraki-ken 305, Japan

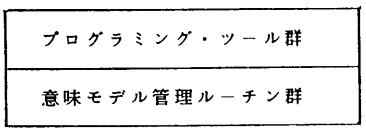
S-model Manipulating System is an intelligent programming system for program-generation and reuse in the field file processing and data structure manipulating software. It manipulates S-model-information on object problems represented in an uniform object-relationship formalism. It is desired that knowledge-based systems such as S-model Manipulating System are extensible. This paper describes some methods which generates S-model manipulating routines and various programming tools based on them (i.e., direct executor, symbolic executer, rewriting rulemachine and problem-solver) from the knowledge about structure of knowledge, definitions of composite objects and relationships, and so on.

1. はじめに

意味モデル操作システム[1]は、意味モデル[2]により表現された対象問題の全ての情報－－要求、仕様、設計、知識、コードーーを操作することによりプログラムの生成再利用を行なう知的プログラム開発環境である．現在のところ、ファイル処理[2][3]およびデータ構造操作プログラムをその対象問題領域としている．意味モデルでは、構文論的にはミクロからマクロまでの凡ゆるレベルの情報が対象物－関係の一様な表現形式で表わされ、意味論的には、対象物や関係のラベルの意味付けは、知識構造（下位の概念は上位のそれを継承する）、データ型の操作の手続き的定義、集合論的観点からの制約条件、などで与えられる．従来のオブジェクト指向システムと比べて、対象物だけでなく関係とそれに関する知識の存在も含んでおり、あらゆる点で表現の構造化と集合論的・論理学的扱いが可能となっている．

意味モデル操作システムは、その中核をなす意味モデル管理ルーチン群と、それらを駆使したツール群（直接実行機、記号実行機、書き換え規則機、問題解決機など）からなる（図１）．このような知識に基づくシステムは、対象問題領域の拡張とともにその処理機能の拡張をも必要とする．そのためには、システムが、そのシステム自体を対象問題とした記述と自身の各種設計機能の適用とから、システムのプログラムを（半）自動的に生成できる機能を持つことが望まれる．

図１　意味モデル操作システムの構成



本論文の内容は次ぎの通りである．第２章では、意味モデルの表現および意味モデル操作システムについて、その概要を紹介する．第３章および第４章では、知識の構造、各種関係の定義、インスタンスの構造の定義などを用いて、意味モデル管理システムルーチン群、および、抽象構文木として表現されたプログラムモデルの直接実行機および記号評価機、を各々生成する方式について述べる．これらの生成は、

入力データの構造　　⟶　プログラムの制御構造
入力データ要素の位置⟶　対応する処理の位置
データ要素の辿り　　⟶　アクセス関数：前始末
出力構造　　⟶　出力組み立て：後始末
データフロー　　⟶　属性文法の観点

という、「データの構造に基づくプログラミング」の概念に則りその生成が行なわれる．第５章では、前向き推論（状態空間探索）および逆向き推論（還元法）のスキーマ・アルゴリズムをカスタマイズすることにより、特定問題領域向きの書き換え規則機および問題解決機を生成する方式を説明する．

意味モデルと関連する研究には、データ意味論[4]と意味ネットワークの手続き的意味論[5]がある．また、ソフトウエア開発における形式的な方法の重要性およびそこで使用されるツール群について述べている論文には文献[6]がある．しかし、これらの研究はその思想の枠組みの説明に留っており、プログラムの生成再利用に関しては触れていない．

2. 意味モデルとその操作システム

2．1 意味モデルの表現形式

現在プログラミングや設計の分野で注目されているオブジェクト（対象物）指向プログラミングは、

① データとその属性および操作（メソッド）を一体化したオブジェクト（クラスとインスタンス）を基本記述単位とする、

② 上位下位関係に基づく知識の構造化とそれに基づく下位概念における上位概念の属性とメソッドの継承、

③ 対象物間のメッセージの交換とプログラムの駆動、

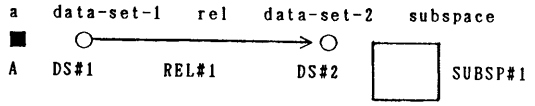
などの概念から成り立っている[7]．

オブジェクト指向プログラミングは、システムの基本構成法として非常に重要な概念であることは確かであるが、システムを定義、設計、記述、生成、管理する立場から考えると不十分で、対象物(object)と関係(relationship)の両方を重視することが必要であると思われる．筆者は、プログラミングの分野でこれらの目的を達成するために、対象問題（あるいはシステム）の要求定義、仕様、設計、プログラム、知識のほとんど全ての情報を、「意味モデル」の表現形式で記述することを提唱してきた[1][2][3]（なお、本論文では、③の機能を除いたものを考える）．意味モデルは、述語論理、集合論、抽象構文、抽象データ型、知識構造、フロー構造などの表現法を統一した一様な対象物－関係の形式で表現するものである．意味モデルの図式表現は、図２（ａ）に示すように、（部分空間(subspace)を含む）対象物と関係のラベル付き有向グラフの形式で表現される．ラベルの小文字は対象物と関係の型、大文字は固有名を表わす．それぞれは、オブジェクト指向的には、クラスとインスタンス名に対応す

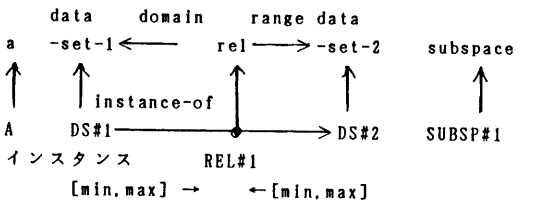
る（同図（ｂ））．なお記号の説明は次節参照．

図２　意味モデルの二つの表現

（ａ）　図式表現



（ｂ）　そのクラスとインスタンスによる表現



２．２　意味モデル操作システムの構成

２．２．１　意味モデル管理システム

知識構造

意味モデル操作システムの知識構造は、抽象データ型、対象物、関係、操作の上位下位構造からなる．下位の概念は上位の概念のメソッドやスロットを継承する．意味モデルにおける対象物と関係の手続き的意味[5]は、それらの型を抽象データ型としたスロットとメソッドにより定義される．

（１）　対象物

データ処理の問題記述のための対象物の主な基本型としては、データ集合（記号○）、識別子（記号◇）、算譜構成要素（記号■）、部分空間（内部に部分モデルを持つ長方形を記号として用いる）などがある．これらの型は、さらにいろいろな型に分類される．データ集合は、メッセージ（成功／失敗／中断）、アクティベーション・レコードなどシステムで用いる型を含む．算譜構成要素の上位下位構造（の一部）は下記のようになる．

```
program component ---
  iteration-statement-- while, repeat-until,
  test-statement--- if-statement, case-statement
  statement-list-statement,
  call-stament ---    ,
  no-operation-statement
```

対象物インスタンスのスロットには、入射インスタンス関係リスト、および、出射インスタンス関係リストがある．

対象物クラスのメソッドとしては、指定された部分空間において、与えられたクラス名のインスタンスを作る、インスタンスのスロット名に対応する値を取り出す、－－、などがある．

（２）　関係

二項関係には、そのエッジの矢先と矢元の対象物の型により定義される様々な型があり、その型により持つ属性が異なる（推移律など）．データ集合間の関係は、図２で示すように矢印つきの[min,max]（一方のデータ集合の一つの要素に対応する他方のデータ集合の要素数の最小値と最大値）で表わされる対応情報[4]を属性として持つ．これは、仕様記述において、集合の和、積、差、基数(cardinality)、１対１、多対１、全体関数、部分関数、の上への写像、の中への写像、鍵、関数従属、同値類などを表わすのに用いられる．　次ぎに算譜構成要素間関係の型と定義を記す．ここで、[ ] で囲まれている関係は省略可能、そのままの関係は使用が必須、であることを表わす．また複数個の関係の並んでいる順は辿る場合の順序を示している．

- root ---- procedure-headから program component への関係、
- body ---- iteration-statement から program component への関係、
- [bf] --- program componentからその前に行なわれる program componentへの関係、
- [af] --- program componentからその後に行なわれる program componentへの関係、
- then, else --- if-statement から program component への関係、
- s-1, s-2, s-n --- statement-list-statement から program componentへの関係、
- c-1, c-2, c-n --- case-statement から program component への関係．

関係インスタンスのスロットには、対応情報（一部の関係）、順序番号、矢元対象物名，矢先対象物名、関係の真理値、などがある．

関係クラスのスロットには、domain、range 、対応情報（一部の関係）がある．

関係クラスのメソッドには、指定された部分空間において、両端点とクラス名を与えてインスタンスを作る、端点の内一方を取り出す、などがある．

（３）　部分空間

部分空間は対象物の一つの型であるが、部分グラフ表現を内部に持つ．プログラムの状態、メソッドの事前条件および事後条件、算譜構成要素の判定文および繰り返し文における条件式、書き換え規則のＬＨＳおよびＲＨＳ、などを表わすのに用いられる．さらに、コンテキストベース（２．２．２参照）の構成要素と

してのローカル知識ベースおよびローカル・データベースを表わすのに用いられる．

部分空間インスタンスのスロットには、その部分空間内に含まれる対象物インスタンスのリスト、その部分空間内に含まれる関係インスタンスのリスト、などがある．

部分空間クラスのメソッドには、それに属する全対象物インスタンスを取り出す、それに属する全関係インスタンスを取り出す、整合、展開、などがある．

2．2．2　各種ツール

意味モデル操作言語は、手続き型およびパターン（ルール）指向型（前向き推論、逆向き推論）のマルチパラダイム言語である．意味モデル管理ルーチンを用いたツールとして、直接実行機、記号実行機、書き換え規則機、各種問題解決機がある．これらは全て、事前条件、事後条件、ＬＨＳ、ＲＨＳなどの部分空間のパターン整合をとり展開を行なう、アクテベーション・レコード・スタック（実は木）、および、アクティベーション・レコードとローカルデータベースの繋がりを通して組織化されたローカル・データベースの集合体であるコンテキスト・データベース[8] をデータ構造として用いながら動作する点で共通性がある．コンテキストデータベースは、コンテキスト木のあるノードからは、そのノードから木の根への路上にあるローカルベース中の全ての情報が可視である．また、あるローカルベースにある関係の否定を置くことにより、それより根に近いローカルベースにあるその関係の存在を消し去ることができる．コンテキストデータベースは、仮定の設定、状態の変化などを表わすに用いられる．

・dynamic --- 子アクティベーション・レコードから親　　アクティベーション・レコードへの関係．

・ldb ---アクティベーション・レコードからそれに附属するローカル・データベースへの関係．

2．3　意味モデル管理ルーチンのＬＩＳＰ上の実現

上に述べた意味モデルの表現は、モデルだけの世界での操作により扱われるものと仮定している．しかし、実際には（計算機上では）、モデルの対象物および関係はある基底言語（ここではＬＩＳＰを用いる）のグローバルデータ構造上のフレーム構造として表わされている．フレーム構造（実はレコード構造の特殊化）の登録、除去、検索などの基本操作は、それらに対応して定義された下記のようなＬＩＳＰ関数を評価することにより実行される．

・(create frame-type) －－
　　指定された型のフレームの生成．

・(add frame-id, subspace-id, slot-name, slot-value)－－
特定のフレームのあるスロットにスロット値を登録．

・(ret-list object-id　subspace-id 'in-arc/ 'out-arc)－－－－
　　指定された部分空間内のある対象物インスタンスへの入射あるいは出射関係を全て検索する．

・(ret-1 relationship-id subspace-id 'from-node /'to-node)－－－
　　指定された部分空間内のある関係インスタンスの矢元あるいは矢先の対象物名を検索する．

・(ret-all subspace-id 'within-arc) －－－
指定された部分空間の下に属する関係を全て取り出す．

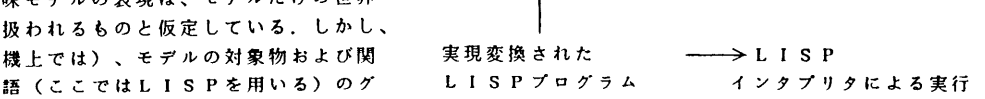
・(ret-sup class-name subspace-id)－－－
　　指定されたクラスの指定された部分空間の下での上位概念を検索する．

・(remove relationship-id subspace-id)－－
　　指定された関係インスタンスを指定された部分空間から取り除く．

プログラムの抽象構文木からのＬＩＳＰプログラムへの変換

意味モデルで表わされたプログラムは、図３に示すように、そのまま直接実行機（４章参照）で解釈実行するのと、ＬＩＳＰコードに変換してＬＩＳＰインタプリタで解釈実行するの、２通りの実行形態がある．プログラムの抽象構文木からＬＩＳＰコードを生成するには、正式には実現変換の手続きを踏む必要があるが、両者の構造間の準同型性を仮定して、算譜構成要素の各型に、対応するＳ式のテンプレートを用意しておき、プログラム抽象構文木を辿りながら、各ノードでそのテンプレートから具体的なＳ式を作り、それらを組み立ててコードを作り出す．

図３　意味モデルのプログラムの２通りの実行形態



3．　意味モデル管理ルーチンプログラムの生成

3．1　概要

意味モデル操作システムの知識構造には、下記の情報がその構造の骨格をなすものとして存在する．

①　対象物、関係、操作における上位下位の分類木．

②　関係のクラスにおけるそのdomainと rangeの定義、

③　構造体インスタンスの内部スロットの構造．
④　フレーム構造における対象物－関係－部分空間の相互間の不変量関係（図４）．

本章では、これらのメタ知識から意味モデル管理ルーチンのプログラムを生成する方式について述べる．

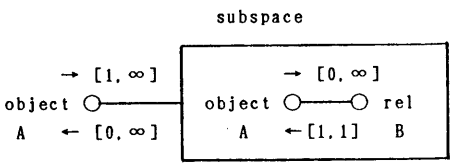
３．２　対象物－関係－部分空間間の不変量関係に基づく意味モデル管理ルーチンの設計

図４は、対象物および関係とそれを含む部分空間間に成り立つ不変量の関連とその対応情報を示している．（ａ）は、対象物を中心とした場合の部分空間と関係との結びつきを、（ｂ）は、関係を中心とした場合の部分空間と対象物との結びつきを、（ｃ）は、上位関係sup に対象物どうしの結び付きを、それぞれ表わしたものである．

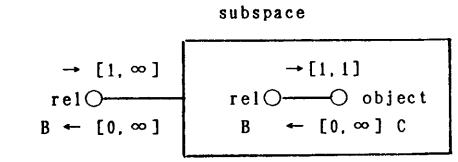
登録の操作においては、追加先の対象物の基数が１より大のときは、集合への追加操作となる．追加先の対象物の基数が１であるときは、スロットへの値の書き込みで済む．　検索の操作においては、複数の要素からの単一要素の検索か、スロットの属性値からの検索か、を判定し、それぞれに応じたプログラムを用いる．

図４　対象物－関係－部分空間の相互関連

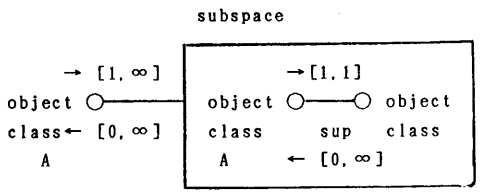
（ａ）　object フレーム　（cardinality(A)=1）



（ｂ）　relフレーム　（cardinality(B)=1）



（ｃ）　object class フレーム



複数のデータ構造間の意味保全性は、操作ごとに、上に述べた（複合）データ構造のデータ構造不変量を同時に満たすことにより、保たれる．そのためには、管理ルーチンの登録(create-object, create-rel) および除去(delete)の操作を、２．３で述べた原始操作(add, ret-, remove) を用いたプログラムモデルとして定義する．管理ルーチンの登録操作あるいは除去操作の引数から繋がりを辿り、図４の拘束条件を考慮することで、対象物インスタンスあるいは関係インスタンスについての操作かそれともそれらのスロットについての操作かを同定する．不変量を満たす操作の複数データ構造への伝播により、組み立てられたプログラムは以下に記すようになる．

登録
・対象物(create-object class-name subspace-id)
　－－部分空間に登録、クラスに登録．
　（部分空間も同じ）
・関係(create-rel class-name
　　from-node-id to-node-id subspace-id)
　－－－部分空間に登録（ただし、これ以外に二つの部分空間にまたがるものあり）．
　　クラスに登録、その両端点の対象物に登録．

除去(delete object/relationship-id subspace-id)
・対象物の場合－－部分空間に登録のもの消去、
　　クラス中のインスタンス消去．
　　出入関係を全て消去、さらにそれらの関係の
　　反対側の端点対象物に登録されたその関係消去．
・部分空間の場合－－その下に属する、対象物、
　　および関係の全て消去．
　　その下に属する部分空間を再帰的に消去．
　　部分空間どうしの関係消去、
　　部分空間を含む部分空間から消去．
　　クラス中のインスタンス消去．
・関係の場合－－－両端点の対象物から消去．
　　部分空間に登録のもの消去、
　　クラス中のインスタンス消去．

３．３　推移的関係を扱うプログラムの設計

３．３．１　データの構造記述に対する検証

意味モデルで用いる関係の内で推移律の成り立つ関係としては、上位(sup) －下位関係(sub) 、アクティベーション・レコード間関係(dynamic) などがある．また、それらにより形成される構造は、木構造であると仮定している．上位下位構造あるいはアクティベーション・レコード・スタックの内容が本当に木構造となっているかどうかの検証を、それらの実データの構造を辿りながら、与えられた意味モデルの木構造記述と対応を付けることにより行なうことができる．なお、直接実行機（４．１参照）における評価対象プログラ

ム中の操作の事前条件と与えられたデータとの整合操作は、これと同じである．

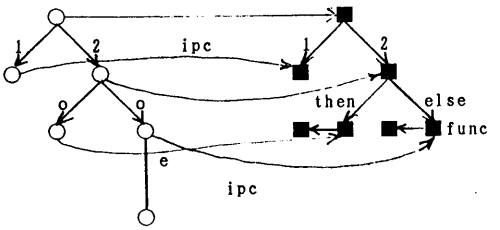
データの構造の検証：
　　実データ　＜－＞　データの抽象構文木表現．
直接実行機：
Ａ１：実データ　＜－＞　Ａ２：評価対象プログラム中の操作の事前条件．
＜－＞：整合（高次整合も含む）．

実データとデータ木モデル（木、線形リストなど）を、出発点として指定された対象物から指定された型の関係の方向へ、対象物あるいは関係どうしの比較を行ないながら辿り、実データがデータ木モデルの一つの実現例であることを検証する．ノードに印をつけて、二度辿ることのないようにする．違いがあればその検出も行なう．

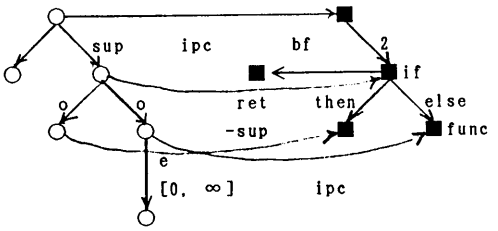
３．３．２　構造を辿る管理プログラムの生成（標準プログラムのカスタマイゼーション）

図５　構造の同型性

（ａ）線形リストの単一解検索プログラムのモデル．



（ｂ）　継承属性を検索するプログラムモデル．



意味モデルの知識ベースにある標準データ型をカスタマイズすることにより、意味モデルの構造を辿る管理ルーチンの生成を計ることを考える．

例）　線形リスト検索プログラムからの上位概念の継承属性を検出するプログラムの生成．

上位下位ハイエラーキ（木）構造における葉から根への検索は、sup: [1, 1]であるので、レコードを単位セルとする線形リスト（非空）の単一解検索とみなすことができる．また各ノードで行なう検索手続きは、そのデータ位置に行き、あるいは、戻りにつけることができる．　図５において、対応がついた場合に、（ａ）の一般プログラムをカスタマイズしたものが（ｂ）である．
ただし図中の関係の意味は次ぎの通りである．
・ｏ --- データの直和を表わす関係．
・ｅ --- データ列からその繰り返し要素への関係．
・ipc --- 入力木の要素と対応するプログラム木の要素　　を結ぶ関係．

４．　ツールとその生成－１
－直接実行機と記号評価機－

４．１　各評価機の動作

意味モデル操作システムにおいては、抽象構文木形式で表現された手続き型プログラムモデル（Ａ１）を意味モデルの表現形式で与えられたデータ（Ａ２）に対して評価するツールとして、直接実行機と記号評価機がある．

直接実行機　直接実行機では、Ａ２のデータは意味モデルで表現された実データの形式で与えられ、Ａ１のプログラムを構成する算譜構成要素を通常の逐次型における実行順に取り出し、その時点での状態を操作の事前条件を含意するのであれば、その操作を適用しその事前条件と事後条件の差に基づく変化を状態に加える．そして、プログラムの出口まで達したときは、成功として、評価結果を返す．直接実行機は、意味モデル操作システムのシステムプログラムや意味モデル操作システムにより生成されたプログラムのラピッドプロトタイピングに有用である．

記号評価機　記号評価機では、Ａ２のデータは、Ａ１の入力仕様である．判定文は評価不能であるので、各々の分岐に、アクティベーションレコードを現在の親アクティベーションレコードに付けることによりスタックを木として扱い、その条件式の真偽を仮定として各アクティベーションレコードに附属するローカルデータベースに展開し、コンテキスト木を作り出す．繰り返し文に関しては、一回だけ展開を行なう．　記号評価機を利用する場面を下に記す．

①　プログラムコードを１操作ずつデータに適用することにより、データの部分構造への操作の適用結果を意味モデルの図式表現によりプログラムの動作を理解しながら進めることができる．

②　人間プログラマは、１ステップごとのデータの状態変化を頭の中で思い浮かべながら、プログラムコードを書いていると思われる．上のことは、記号評価機

の助けを借りれば、１ステップずつインクリメンタルにそのステップがプログラマの意図に合っていることを確認しながらプログラミングを行なうこともできることを意味する．

③　実現変換においては、記号評価機を抽象プログラムに適用することにより、途中の各ステップにおける状態記述を見いだす．それらに表現関数を逆に適用して、対応する具体レベル状態記述の列を得る．そして個々の隣接した二つの状態を繋ぐ具体レベルのコードを問題解決機により見いだし、それらを繋いで具体プログラムが得られる．

④　正当性の検証における使用．

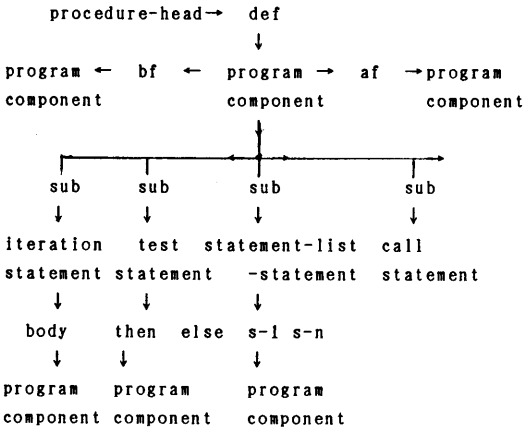
４．２　各評価機の仕様

これらのツールが評価の対象とする意味モデル算譜構成要素のハイエラーキー（およびそれらの間を繋ぐ関係）は意味モデル操作システムの知識構造の中に表現されているので、その分類（部分）木をもとに、これらのツールの一方の入力の構造を定めることができる．

出力仕様は、算譜構成要素に応じたアクティベーション・レコード・リスト（１対１：その関係）およびローカル・データベースを副作用とし、評価がプログラム終了の地点に無事到達した場合を成功、途中の状態が次ぎの算譜構成要素の事前条件を含意できない（すなわち評価を行なえない）場合を失敗とし、そのメッセージを評価機の出力とする．

４．３　各評価機の生成

図６　算譜構成要素に関する対象物と関係を知識構造中で辿って得られる展開



直接実行機

２．２．１の知識構造におけるprogram componentを根として、算譜構成要素間関係を通して、その下位に属する算譜構成要素を（program component が再び現われるまで）辿ると図６のような展開図が得られる．sub は分類を表わすので、それらをデータ木および対応するプログラム木で場合分けするcase文の算譜構成要素を作る．図６を基に、対象物－＞関係－＞対象物と辿るときのアクセス関数を各パスごとに見いだす．二度目のprogram component には、関数direct execute の再帰呼び出しを対応させて割りつける．それから後は、各場合分けごとに検討する．

直接実行機のアルゴリズムとしては、一方のモデル表現された入力をアクテベーションスタックの先頭のローカルベースに置く．

①bfの処理あれば行なう．、

②新たな算譜構成要素に出会うごとに、アクティベーション・レコード・スタックに対応するアクティベーション・レコードをプッシュ、

③各算譜構成要素を型別に処理、

- call statement－－
  変数をアクテベーションスタックの環境で評価．呼び出し文の事前条件と現在のコンテキストとの整合を取れたならば、その束縛をアクティベーション・レコードに登録し、それを基に事前条件と事後条件の差に基づく変化を新たなコンテキストに展開する．
- 意味モデルサブルーチンモジュール呼び出し－－、変数間の束縛対応を環境に登録．
- 実行可能な算譜構成要素（ＬＩＳＰコードあり））
  －－Ｓ式を組み立て、ＬＩＳＰインタプリタで評価する．
- test statement－－
  条件式を現在の状態で解釈評価
- iteration statement －－
  繰り返しごとに条件式を現在の状態で解釈評価、その真偽により終了か否かを判定．
- statement-list-statement－－
  繰り返し文で処理、空で終了．
- no-operation-statement－－何もしない．

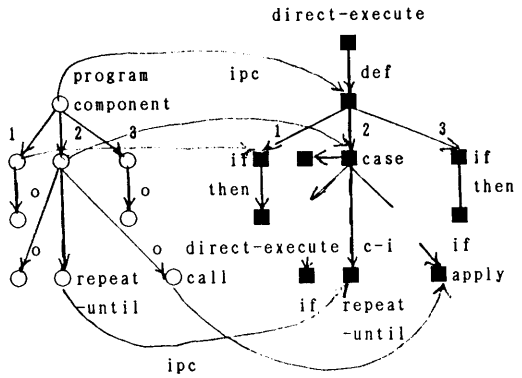
④その算譜構成要素処理終了－－アクティベーション・レコード・スタックからポップ、状態復旧を計る．子からの成功失敗のメッセージの処理と親への伝達．

⑤afの処理あれば行なう．

これらを整理すると、図７に示すような関数direct executeの抽象構文木が得られる．

記号評価機　　省略.

図7　プログラム木をデータとみなした場合の入力構造



5.　ツールとその生成－II
－書き換え規則機と問題解決機－

5．1　概要

もともとは人工知能の分野のもので意味モデル操作システムでも用いるツールとして、書き換え規則機と問題解決機がある．これらは、前向き推論（人工知能における状態空間探索）および逆向き推論（人工知能におけるand/or-graph還元法）[9] を行なうツールであり、書き換え規則機は前向き推論を、問題解決機は逆向き推論を主とし前向き推論を従として、行なう．これらのアルゴリズムは、抽象操作を使用することにより問題領域によらないスキーマ表現をとることができる．問題領域により使用するデータ構造が決まると、操作が具体的に定まる．ここでは、後戻り法による縦型探索アルゴリズムを成功失敗だけでなく、コルーチンを用いた中断状態も許すように拡張したアルゴリズムについて述べる．

5．2　種々の問題領域におけるプログラム合成のための問題解決の解析

本章でその一部を述べる問題解決のためのツールは、以下に記す問題領域のプログラムを合成する研究から得られたものである．

・副作用のない関数プログラム、
・回転やスライド操作のような配列・ポインタの副作用を利用する命令型プログラム、
・ファイル処理プログラム、
・関係データベースの計算を含む検索プログラム．

大抵の問題は、以下の節に示すような基本的な問題解決ツールを幾つか組み合わせて用いることにより、解決が可能となる．そのためには、これらのツールは、次ぎに述べるような柔軟な制御が可能であることを必要とする．

・問題解決における中断状態の存在、
・ツールの部分的問題解決への適用と、全体的問題解決のための複数ツール間のデリケートな制御、
・ツールの用いるデータ構造（アクティベーション・レコード木やコンテキスト・データベース）の部分的変形や融合、
・問題解決の軌跡（and/or-graph）において、新しく生成された状態と同一内容を持つ既存状態の検出．

本論文で述べるツールは、意味モデル管理ルーチンの操作を用いて記述されたプログラムであり、意味モデル操作システム上に実現されたアクティベーション・レコード木やコンテキスト・データベースを操作することができる．そして、成功した問題解決の軌跡から問題のプログラムの抽象構文木モデルが生成される．

5．3　状態空間探索（前向き推論）機

状態空間探索（前向き推論）を行なうツールとしては、書き換え規則機（プロダクション生成機）、および、操作（および公理）を前向きの方向（入力仕様から出力仕様の方向）に適用する問題解決機が挙げられる．これらのツールは共通のスキーマアルゴリズムにより表現できる．以下、8クイーン問題[10]のアルゴリズムを参考に状態空間探索のアルゴリズムを考える．このアルゴリズムの出力として次ぎのものがある．

① 問題解決のメッセージ（成功、失敗、中断）：関数の値とする．
② 副作用として作られるデータ構造（アクティベーション・レコード木、コンテキスト・データベース）．
③ 適用結果（あたらしく作られる状態内容）．

状態空間探索の場合は、状態対象物をそのノードとし適用操作（あるいは規則）をエッジとした探索木の意味モデルで表現されることとなる．適用操作（あるいは規則）が状態遷移をひきおこす入力と考えられる．状態空間探索のアルゴリズムは、後戻り型の場合、木を辿る順序と一致しているので、これに対応を取ることにより、プログラム木が表現できる（図8）．

①現在の状態について拘束条件の充足性テストを行なう．
②現在の状態についてゴール到達性のテストを行なう．
③さらに現在の状態に適用可能な候補全てについて繰り返し演算を行なう．候補を一つ取り出しそれを現在の状態に適用後、このアルゴリズムの再帰呼び出しを行なう．失敗で戻ってきた場合、状態の復旧を行なう

ことが必要である．そして次ぎの候補について同様な操作を繰り返す．繰り返しのループを抜けるのは、それ以上候補がない場合失敗終了として、あるいは、候補の適用結果が一つでも成功、の場合である．

アルゴリズムは、メッセージとして、成功／失敗／中断のいずれであるかの情報と中断の場合アクティベーション・レコードのノード名を返す．また成功／中断の場合は、途中副作用として作られたアクティベーション・レコード木とそれらが指すローカル・データベースを残しておく．

5．4　問題解決機

基本的な問題解決機としては、3種類の逆向き推論による問題解決機が用いられる．ここでは紙面の関係上、一番簡単なものだけ述べる．

逆向き推論問題解決－and/or-graph還元法

問題の出力仕様を入力仕様に還元する問題解決法は、人工知能の分野では、還元法とよばれ、その探索グラフはand/or-graphで表わされることが知られている．図9に示すように、問題の出力仕様をゴールとみなすと、それは分割された複数のサブゴールのａｎｄに書き換えられる．これは探索グラフにおいてａｎｄノードを用いて表わされる．またサブゴールを入力仕様に還元する際には多くの候補の操作があり、探索グラフにおいてはｏｒノードを用いて表わされる．探索の軌跡が木でなくグラフになるのは、分割あるいは操作の適用により、新しく生成された状態と同一内容を持つ状態がすでにノードとして存在している場合があり、その場合は新しいノードを作らず既存のノードで代用するので、探索の軌跡は木でなくなるためである．

図は示さないが、還元アルゴリズムは、成功／失敗／中断の状態を許すもので、縦型探索と横型探索の両方を混合した戦略のアルゴリズムとなっている．ａｎｄ処理に関しては、そのすべての子が成功であるか、一つでも失敗したならば、ループを飛び出す．

5．5　プログラム・スキーマのカスタマイズ

スキーマでは、上位抽象操作で指定しておく．データ構造の指定により対応する下位操作が用いられる．スキーマにおいて、問題分野の意味を生かしたカスタマイズの行なわれるセグメントを以下に枚挙する．

・部分空間どうしの整合操作．
・現在の状態の問題探索ゴールとの一致判定．
・現在の状態の拘束条件充足性の判定．
・パターン整合により候補の部分空間を取り出すデータ構造の指定、
・整合結果の束縛を用いた部分空間の展開．

6．　おわりに

意味モデル操作システムを対象として、知識ベースにある自己記述からの、意味モデル管理システム、直接実行機などの評価機、および、人工知能向きツールなどの生成方式について述べた．これらにおいて、知識構造内の知識データを辿ることの様々な利用法や、データの構造がプログラムを書く際のベースになるということ、抽象化の利用法などについて述べた．

参考文献

[1]　間野暢興：”対象物－関係モデリングをベースとした知的プログラミング環境”、電子通信学会ソフトウエアサイエンス研究会資料SS88-39 (1989).

[2] Mano, N.:"Modeling of Data-processing Software for Generating and Reusingtheir Programs", Proc. of 10th International Conference on Software Engineering, pp.231-240 (1988).

[3] Mano, N.:"Semantic Model of File-processing Software and its Application to Automate Program Generation", Proc. of 11th Annual International Computer Software and Applications Conference (COMPSAC '87), pp.195- 204 (1987).

[4] Abrial, J.R. : "Data Semantics", in "Data Base Management", Klimble, J.W. and Koffeman, K.L. (eds.), North-Holland (1974).

[5] Levesque, H. and Mylopoulos, J.: "A Procedural Semantics for Semantic Networks", in "Associative Networks", Findler, N.V. (ed.), Academic Press, pp.93-120 (1979).

[6] Bjorner, D.: "On The Use of Formal Methods in Software Development", Proc.of 9-th ICSE, pp. 17-29 (1987).

[7]　大特集”オブジェクト指向プログラミング”、情報処理学会誌、Vol.29, No.4 (1988).

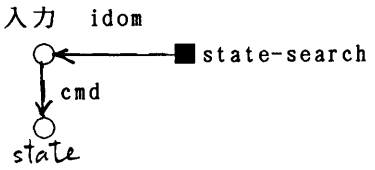
[8] McDermott, D. & Sussman, G.J.:"The CONNIVER Reference Manual", MIT AI-MemoNO.259a (1974).

[9]　白井良明、辻井潤一：”人工知能”、岩波情報科学講座 (1982).

[10] Filman, R.E., and Friedman, D. : "Coordinated Computing: Tools and Techniques for Distributed Software", McGraw-Hill (1984) （雨宮真人、尾内理紀夫、高橋直久（訳）：”協調型計算システム－分散型ソフトウエアの技法と道具立て”、マグロウヒル (1986)）.

図８　後戻り型の状態空間探索アルゴリズムのスキーマ（抽象構文木）

（ａ）　入出力仕様



（ｂ）　入力およびプログラムの抽象構文木

（・memb－－－集合の集合からその要素の集合への関係）

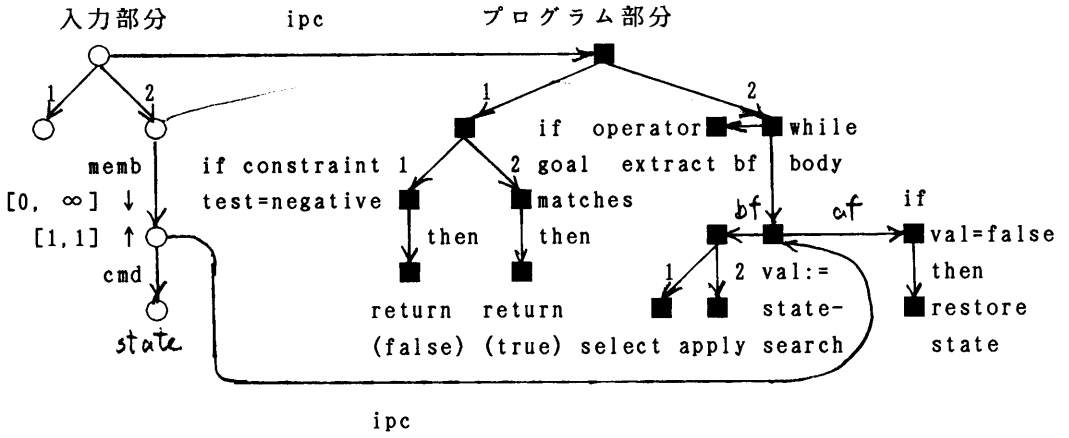


図９　還元法におけるａｎｄ／ｏｒ－ｇｒａｐｈ