ソフトウェア工学 46－8
（１９８６．２．６）

ＭＥＮＤＥＬにおける並列プログラムの部品結合

内平直志　関俊文　粕谷利明　本位田真一

（株式会社 東芝 システム・ソフトウェア技術推進部）

１．まえがき

ソフトウェアの生産性は、まだまだ非常に悪いと実感する。我々の身近なところでも、値段の高さのわりに、使ってみると不満な点だらけというオーダーメイドのソフトウェアがまかり通っている。完璧で満足のいく仕様を書くことは、どだい無理な話であるから、できあがるソフトウェアの質の良し悪しは、プログラマの能力というより気配りに依存することになる。

値段が高くて不満の多いソフトウェアは容認しない，という意識の改革が重要である。そして、納得のいく値段で不満の少ないソフトウェアを作成するための方法論が必要である。その方法論として最近注目されているのが、再利用とプロトタイピングである。

ソースプログラムの再利用を促進する鍵は、部品（あるいは言語）が再利用しやすい構造であること、部品の検索と結合（入出力の整合性を取ること）を機械が支援することの２点である。

再利用しやすい構造としては、Ada のパッケージやオブジェクト指向言語がある。とくにSmalltalk-80は検索・結合に関する支援は弱いけれど、再利用に適した言語構造のためプログラマを、できるだけ再利用しようという気にさせる言語である。

検索・結合の双方に対しては数多くの研究が成されている。そのほとんどは、逐次プログラムにおけるものであるが、上手に運用すれば実用に役立つレベルまできている。しかし並列プログラムに関して、特にその結合に関しては、まだまだわからない面が多い。本報告では、Prologベースの並列オブジェクト指向言語ＭＥＮＤＥＬ［本位田８６］におけるプログラム部品結合という観点から、

１．意味ネットを利用した部品検索・結合

２．並列プログラムにおける部品結合

について述べる。

またプロトタイピングは、効率は悪くてもよいから早期に作成し、核となるアルゴリズムやマン・マシンインターフェイスを検証・確認しようというものである。ＭＥＮＤＥＬは、リアルタイムシステムのプロトタイプ記述を目的とする言語であり、我々は再利用をプロトタイピングの重要な手段として位置づけている。［本位田８５］

２．逐次プログラムにおける部品結合

プログラム部品には、Ada のジェネリックのように内部コードには直接触れずパラメタやそのタイプの設定により外部から調整する部品（ブラックボックス型部品）と内部コードを直接修正して再利用する部品（ホワイトボックス型部品）がある。パラメタの調整も内部コードの修正も“部品結合”フェイズの作業と考えられる。ここでは、ブラックボックス型部品における部品検索・結合について考える。

部品検索は、部品が蓄積されているデータベースから、欲しい部品を検索する方法である。会話形式で欲しい部品を探索していくエキスパートシステム［Mikame85］のような方法以外は、欲しい部品の仕様とデータベースの部品仕様とのパターン・マッチにより検索する方法が一般的である。この場合、仕様をどのような形式で表現するかがポイントとなる。最も単純な方法はキーワードで仕様を表現するものであり、実用レベルでよく使われているが、これはあくまでも欲しい部品のだいたいの候補を挙げる程度

のものである。

これに対して、キーワード方式を発展させて、部品の扱う操作名や操作する対象名の組で部品を表現するもの［石川８５］［岡村８５］や、入出力のデータ定義式とソート関係式［鯵坂８５］、格ラベルを持つ手順的表現および入出力関係による述語的表現［西田８４］、一階述語論理式［吉田８５］等で部品の仕様を表現する方法が提案されている。論理式で仕様を完全に表現するならば、より厳密な検索ができるだろう。しかし現段階では、仕様を論理式で記述することは、プログラムを作成する以上の手間がかかる。我々は、入出力仕様だけに限定しても有効な部品検索・結合が可能であると考え、ＭＥＮＤＥＬによる部品結合では、入出力仕様による部品の仕様表現を採用した。

また、与えた仕様を満たす単一の部品は存在しないが、複数の部品の組み合せでなら仕様を満たせるといった場合がある。このような複数個の部品の組み合せを検索するときには単一の部品の検索方法の単なる拡張では、原理的に難しい場合がある。しかし、ＭＥＮＤＥＬのように入出力仕様で部品を表現する方法ならば、複数部品の組み合せの検索に対してもアプローチが比較的容易である。

自動的（半自動的）部品結合のついても多くの研究がなされている。それらには、部品検索と連携しているもの［西田８４］と、部品は人間が用意するもの［Sueda ８５］［坂上８５］がある。いずれにしろ用意された部品と部品、あるいは部品とそれを埋め込む親プログラムとのインターフェイスを調整するのが部品結合である。インターフェイスの調整は、単一化（unification)と同一化（identification）に分類できる。

単一化は、必要としている仕様より部品の仕様のほうが汎用性を持っている場合（検索される部品は、たいがいそうである）におこなわれる。つまり部品の汎用的な部分を必要としている仕様と単一化することによって、部品を特定化する。

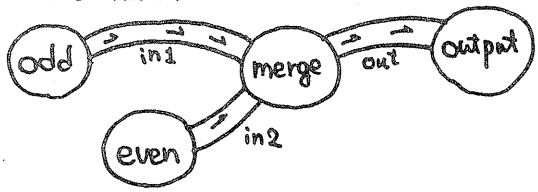
同一化は、部品間のパラメタ間を関連ずけることである。パラメタの関連づけおよびその整合性のチェックは、辞書やフレーム表現による“知識”を参照することによって機械的におこなわれる。［Sueda ８５］［坂上８５］

ＭＥＮＤＥＬは、入出力属性の同一化による部品結合をおこなう。この同一化のために参照する知識として、入出力属性を意味ネットワークで表現した属性ネットワークを提案する。［内平８５］

３．並行プログラムにおける部品結合

ここで考える部品とは、タスクやプロセスのような並行に動きうる単位である。ここではプロセス間の結合について考察してみる。メッセージ伝達に基づいて通信し同期をとるようなプロセスにおいて、プロセス間の結合とは、まず『メッセージの送り手・受け手を決定すること』である。この場合、部品（プロセス）内部における送り手・受け手の指定は通信チャンネルのような間接指定であるほうが扱いやすい。またすべてのプロセスは最初に生成され、メッセージの送り手は静的（固定的）に決定される場合を考える。（Smalltalk のようにインスタンスを動的に生成したり、メッセージの送り先を動的に決定できる言語では部品結合を考えにくい）

［尾内８４］にあるＯｃｃａｍの例で考えてみよう。（図１）




```
CHAN in1.in2.out :
 PAR
  odd(in1)      奇数を生成するプロセス
  even(in2)     偶数を生成するプロセス
  merge(in1,in2,out)
       奇数と偶数をマージするプロセス
  output(out) 端末に表示するプロセス
```

図１　Ｏｃｃａｍのプログラム

この例では、４つのプロセスodd,even,merge,output が部品であり、チャンネルin1,in2,out を各プロセスに割り当てることが『部品結合』に相当している。

ＭＥＮＤＥＬは、各オブジェクトが並行に動き、メッセージの送り手・受け手は非同期式の通信パイプにより静的に決定される。そして、各オブジェクトの入出力属性をその通信パイプで自動的に結合しようというのがＭＥＮＤＥＬの部品結合である。

Ｏｃｃａｍのマージの例は、２章の逐次プログラムの部品結合におけるパラメタ間の関連づけと同じ技術で自動的結合が可能である。なぜならば、この例ではデータは滝のように流れるだけで、デッドロック等の問題が生じえないからである。哲学者の食事のような問題で、仕様がデッドロックフリーという条件を含むような場合、部品結合は単にメッセージ送り手・受け手を決定するだけでなく、そのメッセージを送ったり受けとったりする順序とかタイミングをも、その与えられた仕様を満たすように決定する（生成する）必要があろう。ＭＥＮＤＥＬにおけるメッセージ授受のタイミングの生成に関しては現在研究途上である。

４．Prologベースの並列オブジェクト指向言語 ＭＥＮＤＥＬ

4.1 オブジェクト

MENDELのオブジェクトは、部品に適した言語構造をしている。すなわち、他のオブジェクトとの独立性がきわめて高い。具体的には、MENDELではオブジェクトの内部と外部は、パイプ栓を通してしか情報交換できない。すなわち内部からは外のオブジェクトを直接指定することはできない。パイプ栓には属性名がついており、オブジェクト内部では、この属性名で入出力メッセージを参照する。この属性名は単なるパイプ栓のＩＤではなく、出入するメッセージの属性（仕様）を規定したものである。オブジェクト間の関係づけは、図２に示すようにパイプ栓とパイプ栓を通信パイプで配管することによって成される。この通信パイプは非同期式の１対１一方向通信路である。

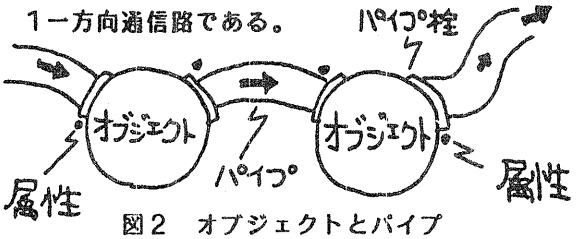


図２　オブジェクトとパイプ

4.2 オブジェクト間の自動配管

MENDELでは、パイプ栓とパイプ栓との配管が自動的に行うことができる。すなわち、オブジェクト群に対して、あるゴール（入出力属性）を与えると配管支配人が、そのゴールを遂行するために必要なオブジェクトを推論によって選択し、通信パイプを配管する。この自動配管は各オブジェクトの入出力仕様に関する知識（メタ知識）と配管支配人の持つ配管ルールにより行なわれる。具体的な配管方法は５章でのべる。

4.3 MENDEL の文法概説

(1) オブジェクト

オブジェクトは、メタ知識と対象知識から構成される。

```
<object name >
{
    spec : {
            <specification part>
                …メタ知識を記述     }
    body : {
            <body part >
                …対象知識を記述     }
}
```

メタ知識とは、オブジェクトの外部仕様、特にインターフェイス関係の情報である。対象知識は、オブジェクトのメソッドの集まりであり外部からの入力メッセージに対する処理内容が記述されている。

(2) 対象知識

対象知識はいくつかのメソッドと共通知識とから構成されている。各パイプ栓の属性名とパイプから入力する値（属性値）をまとめて属性テーブルと呼ぶ。オブジェクトは、この属性テーブルにメッセージを受信すると適当なメソッドを起動する。メソッドは次のように表記する。

```
method{ (<属性>? <論理変数名>)。。。
        (<属性>! <論理変数名>)。。。
    <－Prolog　の節のボデイ.
      Prolog節（メソッドに固有の知識）
}
```

全ての <属性> ？の後の論理変数の値が属性テーブルから受信可能ならば、そのメソッドが起動される。そして、メソッドが終了したとき <属性> ！の後の論理変数が定数とユニファイされていれば、その定数値が属性テーブルを通して通信パイプに掃き出される。

(3) メタ知識

メタ知識には、①メソッドの入出力の記述、②スーパークラスの定義、③内部状態変数の定義の３つがある。

オブジェクトの全てのメソッドの入出力仕様をシステム述語methodを用いて明記する。

```
method(<相手先指定子>: <属性> ？。。。
       <相手先指定子>: <属性> ！。。。)
```

ここで <相手先指定子> は、メッセージの宛先を指定するものであり、種類として

```
< オブジェクト名> …直接オブジェクト名
                    を指定する
  METAPHOR    …自動配管ルールによ
                    って決定される
  ｓｅｌｆ    …自分自身と配管する
```

があり、デフォルトはMETAPHORである。配管支配人は、このメタ知識を用いて配管する。最も基本的な自動配管ルールには、属性名が同じもの同士をパイプでつなぐというものがある。

内部状態変数の定義は、

```
variables(< 変数名> !<初期値> )
```

とする。

(4) オブジェクト間交信の設定

```
                ‥‥communicate 文
```

システム述語communicate は、拡張したモジュール呼び出しといえる。すなわち、与えることができる属性と欲しい属性を指定すると適当なオブジェクトを選択し配管し、各オブジェクトを並列に動かすことによって欲しい属性値を得る。オブジェクトの選択・配管は自動配管に任せても、プログラマが記述してもよい。自動配管の場合は次のように表記する。

```
communicate [
  METAPHOR( < 属性>! <論理変数名> 。。。
            < 属性>? <論理変数名> 。。。)
]
```

このcommunicate 文は、オブジェクトの<body part> のProlog節の述語として書けるだけでなく、MENDELのプログラムを開始させるためのゴール節としても使われる。

4.4 MENDELにおけるメッソッドの起動

オブジェクトは、並列に動く他のオブジェクトから非同期にメッセージを受信し、そのメッセージの組み合せによってメソッドを選択する。たとえば

```
method(a?,b?,c?,d!)
```

において、a?,b?,c?が異なったオブジェクトから送信される場合には、a?,b?,c?がそろうまでこのmethodは選択されない。従って、異なるオブジェクトから送信される複数のメッセージが全てそろって初めてメソッドは選択される。

MENDELにおけるメソッド選択は、Dijkstraのガード付きコマンドとみなせる。すなわち、宣言されたメソッドのうち、ある複合and 条件が真となったとき（すべてのメッセージがそろい、かつすべてのメッセージの内容が条件を満している時）に始めて、そのガード以降の本体の実行が許される。オブジェクトのメソッドが選択され実行中の場合に、そのオブジェクトに送ら

れてきた他のメッセージは処理待ちの状態になる。複数のメソッドが同時に実行されることはない（相互排除）。

また、属性テーブルをオブジェクト内で有効なワーキング・メモリとみなせば、各メソッドはプロダクション・ルールとなる。すなわち、<属性>?による入力待ちが条件部で、<属性>!による出力が実行部である。

5．属性ネットワークによるMENDELの部品結合

5.1　オブジェクト＝部品

MENDELにおいてオブジェクトをプログラム部品とみなすと、配管支配人によるオブジェクトの選択および配管は、一種の部品検索・結合と考えることができる。部品の仕様は、<specification part>の入出力属性（method文）に表れている。この属性による部品の自動検索・結合がMENDELの自動配管である。配管とは、出力属性と入力属性の同一化にほかならない。

5.2　属性ネットワークの導入

属性を“意味的（semantic）”に構造化することにより、より適切な自動配管（属性の同一化）が可能になると我々は考えた。そこで属性というものを分析してみよう。

例えば、“体温”という属性について考えてみると、属性“体温”が取りうる値は“36.5”とか“37.8”とかの“実数”である。また“体温”は“温度”の下位概念と考えることができる。また、“人間の体温”という属性は、“体温”に“人間の”という１つの限定を加えた形になっている。

そこで”体温”とか“人間”とか“実数”というもの（クラスと呼ぼう）を節点（node）とする意味ネット（semantic network）で構造化する。枝（link）の関係としては、とりあえずis-a（上位概念・下位概念の関係）とhas-a（構成要素の関係）の２つだけとする。これを属性ネットワークと呼ぶ。上の体温の例は属性ネットワークにより図３のように表現される。

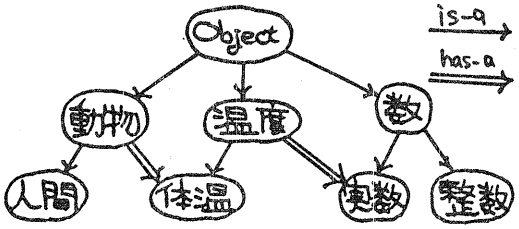


図３　属性ネットワーク

このとき、is-a関係に関しては、single inheritanceとし、全てのクラスのis-a関係の上位クラスとしてObjectを設定する。これは、Smalltalk-80における階層構造とほぼ等価である。すなわち、is-aがclass と subclass の関係、has-a が classとそのinstance variablesの関係に相当する。

5.3　属性の定義

この属性ネットワークに基づいて、属性の記述形式を定義する。

〈属性名〉::= 〈クラス名〉
　　| 〈属性名〉of〈クラス名〉
〈クラス名〉::= 〈Ｐｒｏｌｏｇのアトム名〉

'of' はhas-a 関係を表現する関係であり、“体温”の例では、“実数 of 体温 of 人間”のような属性記述ができる。この意味は、『人間の体温の値（それは実数）』である。このとき属性のいちばん左端のクラスは、integer・string・signal（信号、値はなし）といった物理的なデータ型でなければならない。

5.4　新しい配管方式

属性ネットワークの導入により、“意味的”に配管（同一化）可能な属性を検索できる。

○配管の方向

配管支配人は出力属性にとって、配管可能で最も似ている入力属性を検索する。すなわち出力属性側から入力属性を探す。

○配管可能（同一化可能）

ある出力属性にとって配管可能である入力属性を定義しよう。その原則は、『ある出力属性にとって、意味的に自分を包含する（限定度が弱い）入力属性は、配管可能である』ということである。例えば、①“実数 of 体温 of 人間”と②“実数 of 体温 of 動物”は、動物の下位クラスが人間であるので、出力属性①から入力属性②には配管可能だが、その逆は受け手（入力属性）のほうが限定度が強いので配管できない。また、①“実数of：体温”と②“実数of：体温of：人間”は“of：人間”の分だけ限定度が強いので出力属性②から入力属性①への配管が可能である。これを厳密に定義すると次のようになる。

（定義１）配管可能な集合

出力属性A（$\exists a_1$ of a_2 of …… of a_m）に対して、配管可能な入力属性の集合をU(A)とすると、

$$B(\exists b_1 \text{ of } b_2 \text{ of } \cdots \text{ of } b_n) \in U(A)$$

$$\overset{\text{def}}{\Longleftrightarrow} \begin{pmatrix} m \geq n \\ \text{かつ} \\ b_i \text{ は } a_i \text{ の上位または同じクラス} \\ \text{for all } i \in \{1,2,\cdots\cdots,n\} \end{pmatrix}$$

○類似に関する全順序

配管可能な属性の集合の中で最も似ている属性を配管の第１候補とするわけだが、その“最も似ている”ものを決めるために類似に関する全順序を配管可能な属性の集合に導入する。

（定義２）類似に関する順序

出力属性A（$\exists a_1$ of a_2 of…of a_l）に対して配管可能な入力属性の集合をU(A)とする。さて、$B(\exists b_1 \text{ of } b_2 \text{ of}\cdots\text{of } b_m) \in U(A)$、$C(\exists c_1 \text{ of } c_2 \text{ of}\cdots\text{of } c_n) \in U(A)$としたとき、AにとってCよりBの方が似ていることを、$d_A(B) < d_A(C)$と書くとすると、この定義は次のようになる。

$$d_A(B) < d_A(C) \overset{\text{def}}{\Longleftrightarrow}$$

$$\left(\begin{array}{l} \exists i \in \{1,2,3\cdots\cdots, \min(m,n)\} \text{ s.t.} \\ \left(\begin{array}{l} b_1 = c_1, b_2 = c_2, \cdots b_{i-1} = c_{i-1} \\ c_i \text{ の下位クラスが } b_i \text{ で、} b_i \text{ の} \\ \text{下位または同じクラスが } a_i \text{ である。} \end{array}\right) \\ \text{または} \\ \left(\begin{array}{l} m > n \text{かつ } b_i = c_i \\ \text{for all } i \in \{1,2,\cdots\cdots,n\} \end{array}\right) \end{array}\right)$$

すなわち、属性のより左側（物理的データ型に近い方）にあるクラスがより似てるという観点から属性の類似に関する順序を入れたわけである。そして、この順序は全順序になっている。例えば、A＝実数 of 体温 of 人間とすると、

B＝実数 of 体温

C＝実数 of 体温 of 動物

D＝実数 of 温度 of 人間

の順序関係は次のようになる。

$$d_A(C) < d_A(B) < d_A(D)$$

5.5　インプリメント

MENDELの配管部分はPrologで書かれている。属性ネットワークは、Prologの節として表現した［小山８５］。この表現によればPrologインタプリタ自身が属性ネットワーク上の演繹操作を行なうので、わざわざ演繹用のエンジンを作る必要がない。

６．　例題

6.1　奇数と偶数のマージの例（Ｏｃｃａｍの例と比較して）

２章のOccam の例をMENDELで記述すると図４のようになる。数を次から次へと生成するようなプロセスを作る場合、MENDELではmethodが繰り返し呼ばれる構造にしなければならない。そのために、methodの出力がそれ自身の入力となる再帰的methodをつくる。Odd オブジェクトの場合は図５のようにプログラムされる。

6.2　機械の機能を検索する例（関係データベースと比較して）

分野名（reizohko）の機械の名前(youtousei

) から、その機械の部品の機能を求める問題を、RDB(Relational Data Base)を用いて解く場合［西田８１］と比較して考えてみる。この問題をRDBを用いて解く場合、次のような関係表が与えられているとする。

COMPONENT 表

(Machine DOMain, Machine NAME,
Machine COMPonent)

FUNCTION 表

(DOMain,COMPonent, FUNCTion)

この時、上記問題を検索するための関係代数的演算は、

(((COMPONENT[MNAME=youtousei])
[MCOMP=COMP,MDOM=DOM]
(FUNCTION[DOM=reizohko]))[FUNCT])

となる。ここではMCOMP とCOMP、MDOMとDOM を同一化して２つの表を結合（join）することにより、機械の機能を求めている。これに対してMENDELでは、関係表は各オブジェクトとその入出力属性として表現され、関係代数的演算はcommunicate 文により表現される（図６）。このとき、RDBにおいては人間が指定した結合演算が、MENDELの自動配管機能により自動的に行なわれる。すなわちMENDELは、RDBにおける結合操作を属性間の配管に置き換えたものとも考えることができる。

このとき、属性ネットワークは図７に示すようになり、Prolog表現すると図８になる。ここで、machine-domainとdomainはis-aの関係により配管され、component of machineとcomponent はhas-a の関係により配管されたものである。

６．まとめ

属性ネットワークを利用した部品検索・結合

入出力属性を意味ネットで構造化したことにより次の効果があった。

- 配管に柔軟性がでる（意味ネットの効果）
- 属性名に秩序ができる（属性名の形式化による効果，例：機械名－＞名前 of 機械）

MENDEL に限らず部品のパラメタの同一化のために意味ネットを利用することは十分有効であろう。今後、理想の部品自動合成を達成するためには、オブジェクトの仕様をどれだけ入出力属性で表現できるかがポイントになる。この点でMENDELは、まだまだ改良の余地がある。たとえば、入力と出力の論理的なつながりが記述で

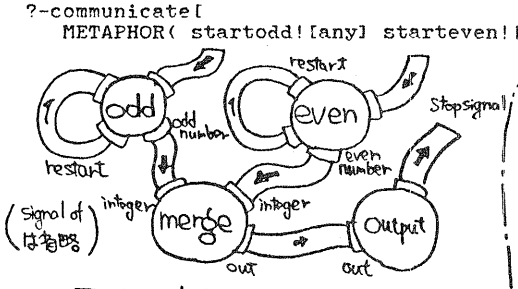


図４ 奇数と偶数のマージの例

```
Odd
(
 spec: ( method( startodd? self:restart! oddnumber! ).
         method( self:restart? oddnumber! self:restart! ).
         variables( in!1 ).
 body: (
   method( startodd?any in?Pre
            in!Post self:restart!Post oddnumber!Post
            <-  Post is Pre + 2.
   method( self:restart?any in?Pre
            in!Post self:restart!Post oddnumber!Post
            <-  Post is Pre + 2.
```

図５ MENDELのプログラム例

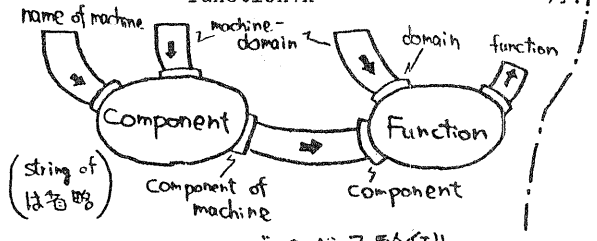


図６ 関係データベース的例

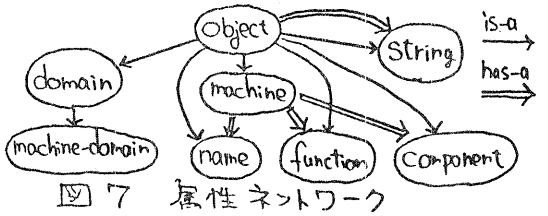


図７ 属性ネットワーク

```
sem(Z.is_a:Z.0).
sem(machine.PROPERTY.X)
        :-sem(object.PROPERTY.Y).X is Y + 1.
sem(object.has_a:string.0).
sem(machine.has_a:name.0).
```

図８ 属性ネットワークのＰｒｏｌｏｇ表現

きないといった点である。

並列プログラムの部品結合

通信パイプの配管という観点から、並列プログラム言語MENDELの部品検索・結合について考えた。MENDELの言語構造は、次の２点で並列プログラムの部品結合に適している。

・パイプ経由の通信ゆえ外界のオブジェクト名をオブジェクト内部で直接記述しなくてよい。
・各メソッドは、属性テーブルから入力することによって始まり、属性テーブルに出力することによって終る。メソッドの途中では外部に一切影響を及ぼさない。

謝辞　本研究は、第五世代コンピュータ・プロジェクトの一環として行なわれた。本研究の機会を与えて下さった新世代コンピュータ技術開発機構の各位に深く感謝する。

参考文献

［鯵坂　８５］
鯵坂，阿草，大野：関数スキーマベースを用いたソフトウェア設計自動化，情報処理学会論文誌，Vol.26，No.2 (1985).

［石川　８５］
石川，他：ソフトウェア部品検索の一手法，情報処理学会第３０回全国大会　4U-12 (1985).

［内平　８５］
内平，粕谷，関，本位田：MENDELS ZONEの構想ー知的プログラミング環境への序曲ー，情報処理学会第３１回全国大会予稿集5M-3　(1985).

［岡村　８５］
岡村，垂水，阿草，大野：ＭＯＭＯＣＬＩにおけるクラス再利用のための知的検索，日本ソフトウェア科学会第２回大会論文集4A-2 (1985).

［尾内　８４］
尾内：並行プロセス記述言語Occam (2)，bit Vol.16 No.12 (1984).

［小山　８５］
小山，田中：Definite Clause Knowledge Representaton,Proc. of The Logic Programming Conference'85(1985).

［坂上　８５］
坂上，田村：処理モジュールの構造的知識を利用した画像処理プログラムの自動生成システム，情報処理学会論文誌，Vol.26，No.4 (1985).

［西田　８１］
西田：言語情報処理，pp110-pp116，コロナ社（1981）.

［西田　８４］
西田，藤田：ライブラリモジュールを用いたプログラムの半自動詳細化，情報処理学会論文誌，Vol.25，No.5 (1984).

［原田　８５］
原田，篠原：部品合成によるプログラム自動生成システム　ＡＲＩＥＳ／Ｉ，情報処理学会研究報告　85-SW-42，Vol.85，No.16 (1985).

［Ｍｉｋａｍｅ　８５］
K.Mikame et.al.:Knowledge Engineering Application In Image Processing,Proc. of Graphics Interface'85 (1985).

［本位田　８５］
本位田，内平，粕谷：推論型システム記述言語ＭＥＮＤＥＬとソフトウェアプロトタイピング，Proc of The Logic Programming Conference '85 (1985).

［本位田　８６］
本位田，内平，大須賀，粕谷：推論型システム記述言語ＭＥＮＤＥＬ，情報処理学会論文誌 Vol.27，No.2 (1986).

［吉田　８５］
吉田，加藤，杉本：一階述語論理式を用いたソフトウェアモジュールの機能検索，Proc. of The Logic Programming Conference'85 (1985).

［Ｓｕｅｄａ　８５］
N.Sueda, S.Hon-iden et.al.：Prolog Application in Software Components Reuse,The Role of Language in Problem Solving Ⅰ，pp.209-220，North-Holland (1985)