ソフトウェア基礎論 24—5
(1988. 2. 12)

部品合成による自動プログラミング・システムの実現方法について

Automatic Programming by Fabrication of Reusable Program Components and its Realization Method

古宮　誠一
Seiichi　Komiya

情報処理振興事業協会(IPA)　技術センター

SOFTWARE TECHNOLOGY CENTER
INFOMATION-TECHNOLOGY PROMOTION AGENCY, JAPAN (I.P.A.)

あらまし　プログラム自動合成技術の中で、実用的で、かつ、手続き型言語によるプログラムを自動的に生成できるのは部品合成による方法だけである。一方、実用の世界では、自動プログラミングシステムよりもソフトウェアの部品化と再利用を支援するツールを構築するほうが容易であり、実用的であると考えている人が多い。ところが、部品合成による自動プログラミングシステムを実際に作成してみると、部品合成による自動プログラミングシステムのほうが実用的なシステムの構築が容易であることが判った。本稿では、その理由を明らかにするとともに、部品合成による自動プログラミングシステムの構築方法を明らかにする。

1．はじめに

人間向きの高度な仕様記述から、そこに記述されたユーザの要求を満たすプログラムを自動的に作り出す技術を自動プログラミング(automatic programming)という。類似の言葉にプログラム自動生成(automatic program generation)とプログラム自動合成(automatic program synthesis)があり、これらの言葉は若干異なった響きを持って使われている。自動生成という言葉は、主にソフトウェア工学の分野で使われ、なんらかの仕掛があり、所望のプログラムが、人間の指導性を生かして半自動的に作り出されるというイメージがある。これに対して、自動合成という言葉には、何も仕掛のない（と思われる）ところから所望のプログラムが自動的に作り出されるというイメージがあり、主に人工知能の分野で使われる。

自動合成などによって得られるプログラムを目標プログラム(target program)といい、目標プログラムの記述言語を目標言語(target language)という。目標プログラムの満たすべき要件や作り方の指示をまとめたものを要求仕様(requirements specification)といい、要求仕様を与えることを要求定義(requirements defi-nition)という。ここで重要なことは、要求定義のためにユーザが与える情報は、原則として「プログラムの機能をいかに実現するか(= how)」ではなく、「プログラムに何をさせるか(= what)」でなければならないということである。なぜなら、要求定義のためにhowの情報を与えることは、ユーザが自分でプログラミングするのと本質的には等価だからである。このため、要求定義のためにhowの情報を与えるシステムのことを特に「自動化環境でのプログラミング」と呼んで、自動合成システムと区別することもある。

これに対して、自動プログラミングという言葉は、プログラム自動合成という言葉よりも意味が広く、非手続き型言語や抽象データ型言語によるプログラミングも含まれる。そして、ときには自動化環境でのプログラミングをも含めて考えることもある。

2．プログラム合成技術の分類と実用性からの評価

現段階でのコンピュータ技術から見て、大規模なソフトウェアでも実用に耐えられるのは、手続き型言語によるプログラムだけである。また、実用の世界で使われているプログラムは、その殆どが手続き型言語によるものである。従って、手続き型言語による実用的な規模のプログラムを自動的に作り出す技術を開発することは意義が大きい。

ところで、プログラム自動合成技術は、①プログラム変換による方法　②論理による方法　③帰納的推論による方法　④部品合成による方法　の４つに分類で

きる[4]。

ここで、プログラム変換とは、予め用意した変換規則に従って、１つのプログラムを同じ意味を持つ別の形式に機械的に置き換えることである。従って、プログラム変換によるプログラム合成とは、実行不可能ではあるが要求仕様の記述も１種のプログラムであると考えて、これを段階的に変換して行くことにより、実行可能なプログラムを作り出すことである。プログラムの記述言語が変換の前後で変わらないものを狭義のプログラム変換と呼び、変わるものを広義のプログラム変換と呼ぶ。①のプログラム合成過程が狭義のプログラム変換に相当し、②のそれが広義のプログラム変換に相当する。プログラム変換は、関数型言語（例えば、純LISP）や論理型言語（例えば、PROLOG）で記述されるプログラムの合成に向いている。これは、これらの言語で記述されたプログラムが持つ「参照の透過性(= referential transparency)」と呼ばれる性質によるところが大きい。１回の実行中に同じ変数に異なった値を代入することが禁じられているとき、そのプログラムは参照の透過性があるといい、プログラム変換の際に、変数の値による場合分けをすることなく、字面だけで変換の正当性が検証(= verification)できるからである。

帰納的推論とは、事実の例から、それを含む普遍的な事実（＝　理論）を導くことである。従って、帰納的推論による方法は、要求仕様を例題で与え、これからプログラムを合成する方法であるが、４つの方法の中では実用化が最も困難である。これは帰納的推論の意味からも判るように、断片的な事実の集合から広い範囲で成立する事実を導くことに困難があるからである。事実、数学的理論も帰納的推論のためには殆ど成果を残していない（成果があるのは、僅かに数学的帰納法と呼ばれる部分のみである）。

①－③の方法で合成されるプログラムの殆どはtoy program（＝　おもちゃのような単純なプログラム）の域を出でいない[4]。言い替えれば、単純な問題にしか適用できないということである。

一方、部品合成による方法は、図１に示すように、

1. 与えられた要求仕様を理解して、単純な要求仕様の集まりとなるように分解する過程
2. 分解後における個々の要求仕様に近い部品の有無により、次のいずれかを行う過程
   - 該当する部品のある場合には、部品を検索して個々の要求仕様に合うようにカスタマイズする
   - 該当する部品のない場合には、部品の生成関数などを用いて、個々の仕様に合うような部品を生成する
3. 2のようにしてできた部品同士を与えられた要求仕様に合うように組み合わせて行く過程

の３つの過程からなり、これらを自動化することにより実現されている。従って、複雑な問題を単純な単位に分解して捉え、後でこれらを組み合わせて行くのが部品合成による方法だと考えることができる。ここで、分解後における個々の要求仕様に合わせてプログラムを生成する過程が、①－③の各方法におけるプログラム生成の全過程に相当すると考えれば、部品合成による方法だけが実用的であるというのも首肯けよう。そして、部品合成による方法は、参照の透過性を必要としないので、手続き型言語のように、この性質を持たない言語によるプログラムの生成にも適している。

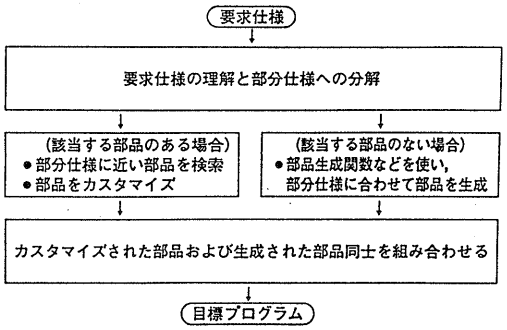

図１　部品合成による自動プログラミングシステムにおけるプログラムの生成過程

３．部品の取扱い方に関する戦略について

（１）領域内での部品の網羅性について

自動プログラミングであるからには、要求仕様は自然言語などの人間向きの高度な仕様記述によって与えられる。それが、部品合成による方法を用いているのであれば、与えられた仕様のプログラムを合成するには、どのような部品が必要かを人間向きの高度な仕様だけから決定しなければならない。このことは、与えられた要求仕様を理解して、プログラム合成に必要な部品の仕様(= 部分仕様)の集合に分解する機能が必要であることを意味している。次に、部分仕様に適合する部品を自動的に検索して、得られた部品同士を自動的に結合すれば、求める仕様のプログラムが得られる筈である。この一連の処理を示せば次のようになる。

①与えられた要求仕様を理解して、部分仕様(= 予め用意された部品と同レベルの仕様)へと分解する過程

②部分仕様ごとにこれを満足する部品を自動的に検索する過程
③与えられた要求仕様に合うように、検索された部品同士を自動的に組み合わせて結合する過程

しかし、部品合成による自動プログラミングが、①～③だけでいつもうまく行くわけではない。①～③だけでうまく行くためには、一定の領域内ならば、ユーザのどのような要求にも応えられるように、その領域内で必要となる部品がすべて用意されていなければならない。言い替えれば、その領域内において、必要な部品が網羅されていなければならないということである。

このとき、ユーザの似て非なる要求に合わせて部品を用意したとすれば、たくさんの類似部品ができることになる。しかし、このような考えで多くの部品を用意したとしても、部品が用意されていないような要求をいつでも作り出すことができるから、この方法ではユーザのすべての要求に応えることはできない。

（２）求める仕様の部品を生成する方法

或る領域内ならば、ユーザの要求するどのような部品でも得られるようにする方法は、次の２つである。

①ソースコードの塊として部品を用意するのではなく、ユーザの要求に合わせて必要な部品のソースコードを生成できるように、部品生成のためのサブルーチンや関数を用意する。
②類似部品を作らず標準部品を網羅して、ユーザの要求に合わせて標準部品のソースコードを自動的に修正（カスタマイズ）する。

①または②の方法で、想定する領域内で必要となる部品を網羅できるというのは、類似した要求に対して、それらの共通部分からなる仕様の部品(= 標準部品)用意し、変分に対してはカスタマイズによって吸収できるようにしているからである。そして、標準部品を必要な数だけ用意すれば、領域を上手に絞ることにより部品を網羅できるからである。①の方法は、規模の大きな部品を用意するのに適しているが、ソースコードの形で用意するので部品の検索処理が必要となる。一方、②の方法は、規模の小さな部品を用意するのに適しており、部品の検索処理が不要である。

①の方式を採用しているのが、杉山(富士通研)らによるKIPS[7]と原田(電力中研)らによるSPACE[2]である。一方、②を採用しているのが、原田(電力中研)らによるARIES/I[1]と今中ら(阪大)によるAPSS[3]である。また、①と②を相補的に採用しているのが、古宮(IPA)らによるPAPS[5]である。PAPSの方式を表したものが図１である。

（３）類似部品群の中から１つを同定する問題

ここで、類似部品について考えてみよう。類似部品は、その扱い方から次の４つに分類できる。

①設計仕様書は同じだが、ソースコードが異なる部品
②機能仕様は同じだが、設計仕様の異なる部品
③機能仕様のレベルで、一方が他方の機能の一部または全体を含んでいる部品
④機能仕様は異なるが、プログラム構造が類似している部品

部品登録において複数個の類似部品が生じることを許した場合、最大の課題は、類似部品群の中から最適な部品を１つ同定することであろう。類似部品群の中でも仕様の差異を定性的に表現できないものは、部品検索時において同定不可能である。そのような部品はタイプ①のみである。従って、タイプ①の部品については、そのような類似部品が複数個できることのないように、部品登録時にチェックすべきである。現在、部品合成による自動プログラミングを実現しているシステムは、すべてこの原則に従っている。

タイプ②の類似部品で識別が必要なのは、部品における設計思想の相違であろう。アプリケーションによっては、例えば、「メモリ・サイズを小さくすることに重点を置いた設計」、「処理速度を速くすることに重点を置いた設計」、「計算精度を良くすることに重点を置いた設計」、「扱うデータのサイズを大きくすることに重点を置いた設計」などを識別して用いる必要があろう。この問題を扱っているシステムはPAPSである。PAPSでは、部品を設計仕様のレベルでモデル化しているので、部品における設計仕様の相違が識別できるようになっており、これにより部品合成の際に設計思想の同じ部品同士での結合ができるようになっている。

類似部品群の中から１つを同定するという問題を積極的に扱う必要があるのは、タイプ③の類似部品であろう。タイプ③の類似部品が複数個できることを許し、この問題を積極的に扱っているシステムがMENDEL[8]である。MENDELは、Prologベースの並列オブジェクト指向言語で、そこで使われるオブジェクトをプログラム部品と捉え、部品の検索・結合を自動的に行うことにより、求めるプログラムを得ようとするものである。一般に、オブジェクト指向言語では、オブジェクト同士で交わされるメッセージでしか、互いに結合できないように制限されている。MENDELでは、各オブジェクトにおけるメッセージの入出力属性を規定する機構を

持っていて、一方のオブジェクトの入力属性が他方の出力属性に一致するときに、２つのオブジェクトは互いに結合できるとしている。そして、各オブジェクトの入出力属性をIS-Aの関係とHAS-Aの関係に基づく意味ネットワークで表現するとともに、ここでのHAS-A関係をPrologの節で表現し直したものからなる(類似度に関する)全順序関係を定義して、一方のオブジェクトの出力属性を満足する部品群(= 類似部品群)の中から、最適な部品１つを選出するのにこれを利用している。

しかし、部品の検索・合成の過程で、類似部品群の中から最適なものを１つ同定するのは、処理時間の点から考えても得策ではない。実用から考えれば、どのような類似部品も複数個できることのないように、部品の登録時にチェックすべきである。このような思想を貫いているのがPAPSである。

タイプ④の部品は、通常の感覚では類似部品ではなく別部品であろう。従って、類似部品郡の中から１つを同定する問題の対象とはならない。寧ろ、プログラム構造が類似していることから、必要な部品を作り出すのに利用される。このような例としてはAPSSがある。APSSでは、求める仕様の部品がないとき、機能仕様は異なるがプログラム構造の類似しているものを類似部品と見なし、類推によってこれを検索して代用するという方式を採っている。しかし、求める部品を類似部品からどのように作り出すかのガイダンスを出力するのみで、ソースコードを自動修正する機能までは持っていない。

(４)部品を利用したプログラム開発環境の戦略比較

ソースコードを部品化する場合、部品の切り出し方には２つの戦略がある。１つは、部品化とソフトウェアの再利用を支援するシステムを構築する場合における戦略である。この場合には、不特定多数のユーザによる部品の作成・登録を想定するので、誰にでも部品が切り出せるように、部品間のインタフェースを簡易にするというものであり、類似部品があっても良く、対象とする領域を部品が網羅するための積極的な措置は採っていない。従って、必要な部品が用意されているという保証もなく、あったとしても類似部品の中からユーザの要求に最も近い部品を要求仕様のみから自動検索するのは殆ど不可能である。検索された部品をユーザの要求に合うように修正（＝　カスタマイズ）するのは手作業によらなければならない。また、部品の用意されていない部分については、どのような仕様のソースコードが必要なのかが判らないので、その部分のソースコードを自動生成することは不可能である。

もう１つは、部品合成による自動プログラミングシステムを構築する場合における戦略である。この場合には、部品の作成・登録がツール作成者なので、部品同士での組合せが容易になるように部品間のインタフェースを設定するというものであり、類似部品がないように、かつ、対象領域をできるだけ網羅するように部品を選び、ユーザの要求に合わせて各部品を自動的にカスタマイズするようになっている。さらに、部品が用意されていない部分については、その部分の要求仕様に合わせて必要なソースコードを自動生成するようになっている。ユーザの要求する部品が出揃ったところで、これらの部品を組み合わせることにより、与えられた仕様のプログラムを得ることができる。この一連の処理を表したものが図１である。

このようなことができるのは、ツール設計者が対象領域のプログラムを良く分析し、そこで必要となる部品を精選するとともに、どのようにすればユーザの要求する部品を生成でき、かつ、組み合わせることができるかを分析しているからである。以上の理由により、部品合成による自動プログラミングシステムが、再利用支援システムよりも構築し易いことが首肯けよう。しかも、対象領域とそこで使われる部品を上手に選べば、実用的な自動プログラミングシステムの構築が可能である。自動プログラミングシステムは自動化率が極めて高いので需要も多く、用途ごとにシステムを用意すれば、様々な要求に対処できるようになる。

なお、ライフサイクル一貫支援システムもソフトウェア開発に部品を利用するが、部品切り出しにおける戦略は、自動プログラミングシステムと再利用システムの中間にあると考えられる。部品の利用方法から見たこれらの比較を表１に示す。

４．部品合成による自動プログラミングシステムの構築方法の比較

部品合成による自動プログラミングシステムの構築方法は、図２に示すように２つある。１つは、部品をソースコードの塊としては持たず、部品生成のための関数やサブルーチンを持ち、これによって細分化された要求仕様の部品を生成するという方式で、部品同士を組み合わせて行くためのプログラムモデルを持っている。このようなシステムの例としては、富士通研究所のKIPSや電力中研のSPACEなどがある。一方、ソースコードの塊としての部品を持ち、これを検索して、ユーザの要求に合うように部品を自動的にカスタマイズする方式である。この種のシステムの例としては、電

表1　部品を利用したプログラム開発支援ツールの比較

比較項目	部品合成による自動プログラミング・システム	ライフサイクル一貫支援システム	部品化によるソフトウェアの再利用支援システム
部品の作成者	ツールの作成者（特定の技術者）	ツールの作成者＋ツールの利用者	不特定多数の部品作成者
部品の選定	類似部品を作らないように，また，想定する領域をできるだけ網羅するように部品を選定する。	ツールの作成者が作る部品には類似部品はないが，利用者の追加により類似部品ができることがある。	類似部品があっても良く，想定する領域を部品が網羅しなくても良い。
部品の切り出しのための戦略	部品同士の組合せが容易となるように部品を切り出す。	どちらかと言えば，左に同じ。しかし，明確には意識されていない。	不特定多数による部品の切り出しが容易となるように部品を切り出す。
要求仕様を理解する機能	要求仕様を理解し，用意された部品の単位にこれを分解する機能を持つ。	右に同じ。	要求仕様を理解する機能を持たない。
部品の検索方法	要求仕様の情報だけから必要な部品を自動的に選定し，自動的に検索する。	右に同じ。	ツールの利用者がキーワードを与えることにより検索する。検索機能を持たないツールもある。
部品をカスタマイズする方法	分解された要求仕様な合わせて部品の仕様を自動的にカスタマイズする。	左記のような機能を持つものもあるが，エディタを使って利用者がオウンコーデイングするものが多い。	利用者のカスタマイズ作業をエディタで支援するものが多い。
部品が用意されていない部分のソースコード生成法	サブルーチンや関数などを使い，分解された要求仕様に合わせてソースコードを生成する。	右に同じ。	エディタを使って利用者がオウンコーデイングするものが多い。
部品同士の組合せ方法	プログラム・モデルまたは部品表現モデルに従って，部品同士を組み合わせて行く。	骨組み部品の中に必要な部分部品を埋め込む。この作業をエディタ画面を利用したツールにより支援する。	ライブラリ複写機能を持ったエディタを使って，部品をソースプログラム中の必要な位置に複写する。
部品および生成されたプログラムの品質保証	部品の品質が保証され，生成されたプログラムの品質も保証される。	ツール作成者が作った部品の品質は保証されるが，生成されたプログラムのデバグが必要。	部品の品質は保証されない。生成されたプログラムのデバグが必要。

表2　部品合成による自動プログラミングシステムの比較

主なシステム	KIPS	SPACE	ARIES／I	PAPS
主な開発者	㈱富通研究所の 杉山　健司	㈱電力中央研究所の 原田　実	㈱電力中央研究所の 原田実，篠原靖志	情報処理振興事業協会の 古宮　誠一
要求仕様の与え方	制限された自然言語（日本語）	CDフロー図，階層構造図，データ構造テーブル，ファイル・テーブル，機能テーブル，ロジック・テーブル，データ移送テーブルの2図5表を使って与える。	制限された自然言語（日本語）	現在稼働中のものは 階層化されたメニューとオウンコーディング しかし，次のものを計画中 階層化されたメニューと誘導型デシジョンテーブル
対象領域	事務処理分野（バッチ処理）	事務処理分野（バッチ処理）	事務処理分野（バッチ処理）	事務処理分野（バッチ処理）
要求仕様を理解し,分解する機能	両方とも持っている。	両方とも持っている。	両方とも持っている。	両方とも持っている。
部品としてソースコードの塊を持つか	持っていない（部品をスライスしたものは持っている）。	持っていない（部品をスライスしたものは持っている）。	部品化されたソースコードのかたまりを持っている。	部品化されたソースコードのかたまりを持っている。
部品の自動検索機能	部品（＝ソースコードの塊）を持っていないので必要なし。	部品（＝ソースコードの塊）を持っていないので必要なし。	部品の仕様を表すフレームの中に見出し情報を持っている。	フレームを使って検索条件を絞り，ルールによって検索。
部品をカスタマイズする方法	部品（＝ソースコードの塊）を持っていないので必要なし。	部品（＝ソースコードの塊）を持っていないので必要なし。	部品の仕様を表すフレームの中に適合化ルールの形で持つ。	カスタマイズのための知識をフレームの形で持っている。
部品が用意されていない部分のソースコード生成法	部品生成知識をオブジェクトのメソッドとして持っている（部品をスライスしたものを獲得し定数として利用する）。	部品生成知識をC言語による関数の形で持っている。 （部品をスライスしたものを獲得し定数として利用する）。	必要だが，部品を使わずにソースコードを生成する機能は持っていない。	オウンコーディングを知的にガイドする方式を採用。 誘導型のデシジョンテーブルにより生成する方式を検討中。
部品同士を組み合わせて行く方法	Precondition, Postcondition の概念に基づくプログラムモデルに従って組み合わせて行く。	生成された部品をプログラムモデルに従って組み合わせて行く。	要求フレーム＋データ導出モデルに従って部品を組み合わせて行く。	part-of と has-parts の概念に基づいて，部分部品を骨組み部品の中に埋め込む。 precondition, postcondition の概念に基づいて骨組み部品同士を組み合わせて行く。
採用しているモデル	意味モデル（要求仕様の理解） プログラムモデル（部品組立） コードモデル（コード生成） 3つともオブジェクト指向＋データ指向からなる対象世界のモデルを採用。	プログラムモデルを持っているがその詳細は不明。	フレームに各種のスロットを追加した対象世界のモデルを採用。	部品表現のモデル（part-of や precondition, postcondition などの概念をフレームに追加した対象世界モデル）を採用。 さらに，part-of の概念に基づくプログラムモデルも持っている。

力中研のARIES/IやIPAのPAPSなどがある。この方式では、部品が用意されていない部分については、その部分の要求仕様に合わせてソースコードを生成する機能が必要である。

これらのシステムについて、その構築方法の比較を表２に示す。

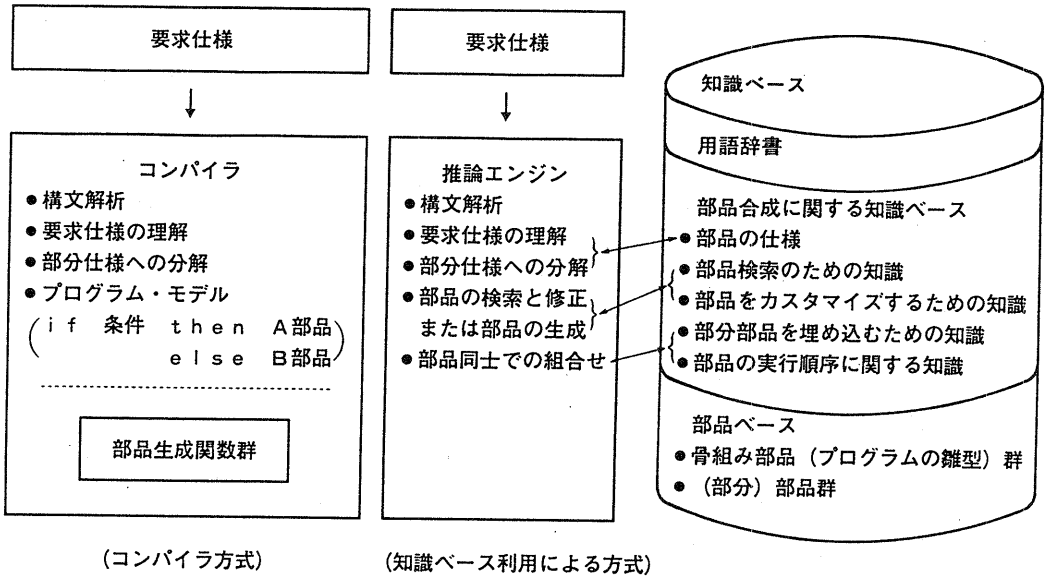

図２　部品合成による自動プログラミング・システムにおける２つの構築方式

５．おわりに

プログラム自動合成技術の中で最も実用的なのは、部品合成による方法であることをその理由とともに示した。部品を利用してソフトウェア開発するツールとして、自動プログラミングシステム、部品化・再利用支援システム、ライフサイクル一貫支援システムの３者を比較して、最も構築し易いのは部品合成による自動プログラミングシステムであり、実用性も最も高いことを示した。最後に、部品合成による自動プログラミングシステムの構築方法を示し、代表的なシステムの構築方法を比較した。なお、プログラム自動合成システムの理論と研究動向については、文献[3]に詳しい解説がある。部品合成による自動プログラミングの構築方法と研究動向については、文献[5]に詳しい解説がある。

［参考文献］

[1]原田 実,篠原靖志：　プログラム自動生成システムARIES/Iの開発,情報処理研究,No.13,電力中央研究所,pp.59-88(1985).

[2]Harada,M.: "Specification Compiler: SPACE", Proc. of IEEE COMPSAC'87,Tokyo,pp.171-180 (October,1987).

[3]今中武,上原邦明,豊田順一：類似したプログラムの再利用による自動プログラム合成,ロジック・プログラミング・コンファレンス,pp.47-56(1987).

[4]古宮誠一：「プログラム・シンセシス法の分析と実用化への方向付け」,技術センター第４回発表会論文集,情報処理振興事業協会,pp.17-25(1985).

[5]古宮誠一：「部品合成によるプログラム自動合成システムPAPS ～知識工学的アプローチを用いたその実現方式～,情報処理学会研究報告,87-SF-21,Vol.87, No.40,pp.5-12(1987).

[6]古宮誠一,原田 実：解説「部品合成による自動プログラミング」,情報処理,Vol.28,No.10,pp.1329-1345 (1987).

[7]杉山健司,秋山幸司,亀田雅之,牧之内顕文：対話型自然言語プログラミングシステムの試作,電子通信学会論文誌,Vol.J67-D,No.3,pp.297-304(1978).

[8]内平直志,関 俊文,柏谷利明,本位田真一：MENDELにおける並列プログラムの部品合成,情報処理学会研究報告,86-SW-46,Vol.86,No.7,pp.57-64(1986).