ICOT Technical Memorandum: TM-0757

TM-0757

知的プログラミングシステム

中村英夫，本位田真一，
内平直志（東芝）

July, 1989

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan
(03) 456-3191～5
Telex ICOT J32964

Institute for New Generation Computer Technology

知的プログラミングシステム

Intelligent Programming Systems

中村英夫(1)　本位田真一(1)　内平直志(1)

Hideo Nakamura,Eng.D.　Shinichi Honiden,Eng.D.　Naoshi Uchihira

プログラムの信頼性と生産性を向上させることは、ソフトウェア工学の主目的である。しかし、同時にこの２点を満たすことは困難であった。一方、人間の知的行為を研究するＡＩ技術が発展しいくつかの成果が出てきている。これを高度な知的行為であるプログラミングに適用した知的プログラミングシステムを開発した。　本システムは定理証明手法及び部品再利用手法の長所を融合させており、プログラミングの信頼性、生産性を同時に向上させている。
さらに、大規模なシステムを制御するリアルタイムシステムを対象とする場合は、仕様を定める段階から可能な限り厳密な取扱が必要である。そこでこういった場合の有効な手段であるプロトタイピング手法も組合せている。

One of the major objectives of Software Engineering is to improve the reliability and productivity of software development. However, it has proved difficult to satisfy this point at the same time. In this paper, we present an intelligent programming suppot system, to which the recent achivement of AI technology has very effectively been applied. In this system, named MENDELS ZONE, we tried to combine the advantages of theorem proving technique and software reuse method. Prototyping method is also added to this combination to get high consistency, which is strongly wanted to the development of large scale real-time control systems.

［１］まえがき

ＡＩ技術とプログラミングを関連づけた「知的プログラミング」という言葉がある。しかし、この言葉の意味は必ずしも確定したものではない。たとえば、

(1) ＡＩ（人工知能）を研究するために、人間の代表的知的行為としてのプログラミング

(2) ＡＩ言語（たとえば、ＥＳＰ、ＯＰＳなど）によるプログラミングとその方法

(3) ＡＩ技術の成果を利用した支援環境を用いたプログラミング

などような、いくつかの考え方がある。本稿では、(3) に沿った知的プログラミングシステムについて論ずる。さらに、大規模な制御システム向きに開発した知的プログラミングシステムを紹介し、その技術的特長について述べる。

［２］知的プログラミングシステムのフレームワーク

プログラミングの支援を行うためひとつの有力な方法は「自動プログラミング」である。ここでいう自動の意味は、与えられた環境のもとで計算機が人間にかわって人間には実行困難な作業を代行するというものである。たとえば

・すべての場合をチェックする

・すべての可能性を列挙する

などである。計算機はこういった作業を、与えられた知識と

(1) システム・ソフトウェア技術研究所

Systems and Software Engineering Laboratory

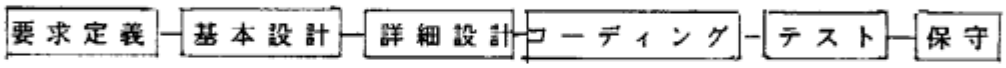
おり正確かつ高速に実行するという性質を持つ。従来のプログラミング言語のコンパイルは、この性質を利用してアルゴリズミックに整理された知識を用い、構文を解析しコードを生成する自動プログラム変換メカニズムである。この意味でコンパイルは自動プログラミングである。しかし、現状のように、プログラミングという概念が単なるコーディングだけでなく仕様記述にまで拡大して来ると、従来方式で可能な自動化技術の範囲以上のプログラミング支援技術が必要になって来る。むしろ解くべき問題及びその解決方法の理解がプログラミング支援の課題になって来ていると考えられる(4)。

こういった流れに対応するために、ソフトウェアの開発方法も従来のウォータフォール型よりもプロトタイプ型が採用されることが多くなってきている。（図１参照）しかし、単にこの方式を導入しただけでは、仕様のあいまいさや不完全さが残り、コーディング以降でないと仕様を確認することが困難である。この方式でのプロトタイプはごく部分的なものでしかないからである。その結果、やはり保守作業などはコードレベルでなされることになり、仕様とプログラムの乖離はなくならず、保守作業の困難さは解決できない。

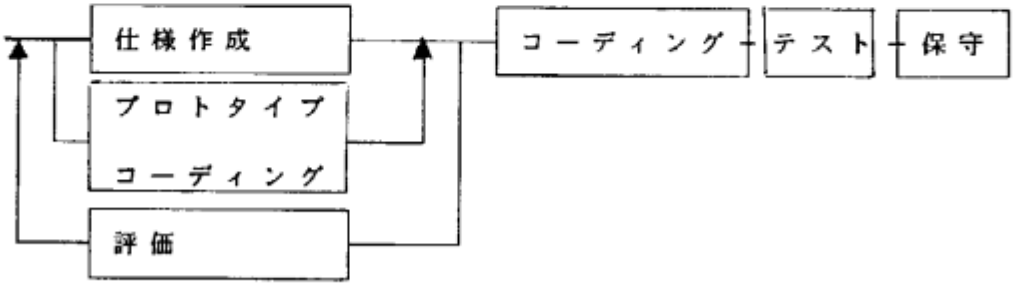
この問題に対し、形式仕様型の開発方法の研究が行われている。今までの代表的な研究に、Green らのＣＨＩ(1) やBalzerらのＳＡＦＥ(2) がある。それぞれ、Ｖ及びＧＩＳＴと呼ばれる高水準仕様記述言語を用いて仕様を記述し、その仕様記述の範囲で評価・保守を行おうとしたものである。このアプローチでは、プロトタイプの評価を行うために、何らかの形でプロトタイプが実行される必要がある。従って、この

方式での高水準仕様記述言語は、そのための計算モデルを持ち、かつ、その操作的意味が明確なものでなくてはならない。

・ウォータフォール型（非形式的仕様記述）



・プロトタイプ型（非形式的仕様記述）



・形式仕様型（形式的仕様記述）

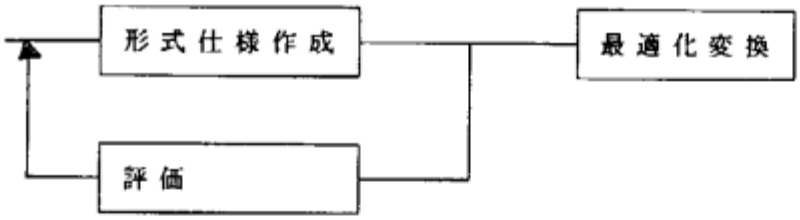


図１　ソフトウェア開発方法のモデル

この種の言語とその処理系には、定形型と非定形型の２つのタイプがある。

定形型は、問題の種類を限定することにより仕様記述方法を限定し、それに従った処理系を作成するいわゆるＰＯＬ(Problem Oriented Language) 型のものである。問題種類の限定方法によっては、比較的汎用な処理系にすることも可能である。たとえば、第４世代言語と言われているものの一部にこれに相当するものがある。

非定形型は、汎用のプログラム表現を持つもので、ききほ

どのＶ言語やＧＩＳＴ言語はこれにあたる。処理系には、問題解決のヒューリステックスやプログラム変換知識、定理証明手法などを導入し、処理の高水準化をはかっている。

このように仕様の記述範囲の拡張や、それに伴う一貫性の維持、プログラムの自動生成によるプログラムの正当性向上などの言語体系や処理方法に関する研究がなされてきた。これに加えて「知的プログラミング」では、プログラマとの対話を通じて開発しているプログラムや問題を理解することにも重点がおかれている。すなわち、プログラムの自動生成手法がたとえ完成されたとしても、それには仕様を与える必要があり、この仕様記述自体が直接プログラムを記述する場合より容易でなければならないからである。また、プログラマが解決しなければならない問題は、多くの場合技術的に自動生成可能なこと以外にある。知的プログラミングシステムとしては、こういったプログラマが解決すべき問題点をプログラマに明らかにする仕様獲得支援システムであることが望まれる。ＭＩＴのプログラマの弟子(Programmer's Apprentice

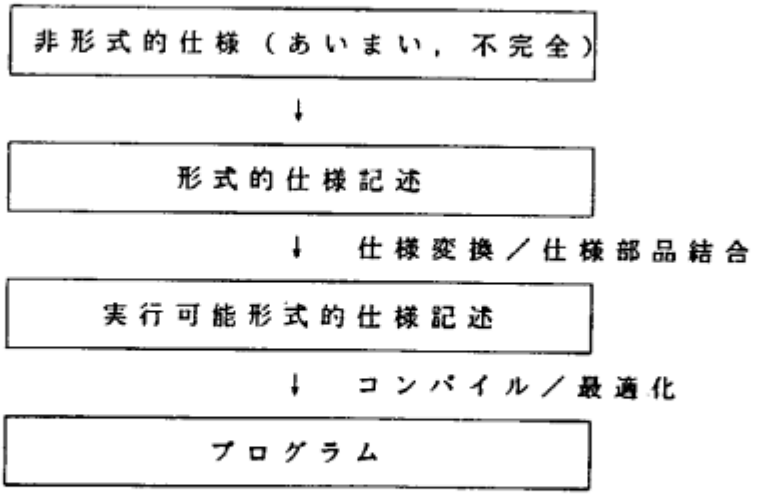


図２　知的プログラミングのプログラム開発モデル

(3)) などはこの流れを組むものである。

我々の知的プログラミングシステム研究は、これらを組み合せたアプローチである。すなわち、形式仕様型の開発方法モデルおよび、あいまいな仕様から実行可能な仕様記述を得るというプログラミングパラダイムを用いている。さらに形式仕様型のプログラム開発モデルを図２に示す開発モデルに詳細化している。そして、このプログラム開発モデルを実現する知的プログラミングシテムを、論理型並行オブジェクト指向言語を基本としたフレームワークで構成している[6]。（図３参照）不完全さやあいまいさを含んだ仕様記述から、妥当な部分を形式仕様化し、それを変換し実行させてみせる。このことにより、プログラマと対話しながらプログラムの正当性を確保し、かつドキュメントをも得る。

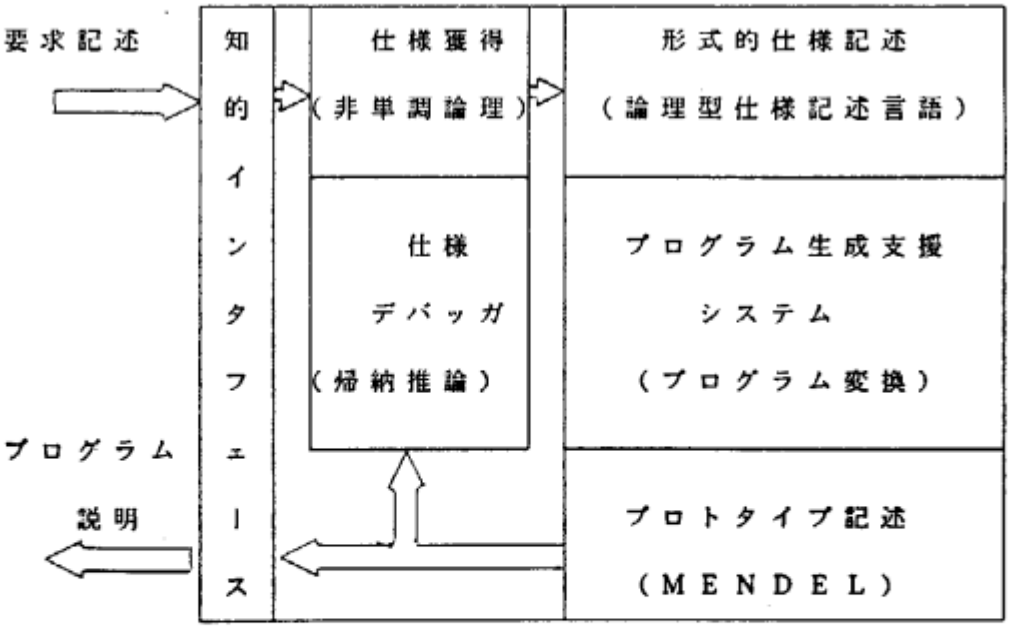


図３　知的プログラミングシステムのフレームワーク

このフレームワークのもとに、発電所や鉄鋼製造ラインな

どの制御システムに見られる並行処理プログラミングを支援するプロトタイピングシステムを開発した。このシステムは、一つのモデル的な知的プログラムミングシステムであり、論理型並行オブジェクト指向言語ＭＥＮＤＥＬ(Meta Inferential System Description Language)[5] を基本としたプログラミング支援環境という意味でＭＥＮＤＥＬＳ　ＺＯＮＥと呼んでいる。以下開発したシステムについて述べる。

[３] 並行処理プログラミングシ支援システム

ソフトウェアの信頼性・生産性の向上をはかるのがソフトウェア工学の目的である。前述した様にこの２つの目的はある意味でトレードオフの関係にあり同時に満たすことは困難である。一つの解決方法としてプロトタイピングと部品再利用手法が行なわれている。しかし、これまでのプロトタイピングは制御システムのような並行処理ではなく、主に逐次処理を対象にしていた。また、部品再利用手法についても、部品結合の概念に制限が多かったり、抽象度が高すぎて実際的でないなどあいまいさの残る手法であった。一方、あいまいさを排除する手法として定理証明手法があり、完全性を主張している。しかし、この手法は計算可能性に関する困難さがあり、実現例としてはトイプログラムの範囲を越えていない。これらの長所・短所をまとめると図４のようになる。本システムは両者を組合せたものであり、部品再利用手法を用いて生産性を向上させ、定理証明手法を部品間のメッセージ制御の手法として用いることで、生産性と正当性の両方を満たすことが可能となった。従来より、並行処理プログラムでは、

	生産性	正当性
部品再利用手法	○	△
定理証明手法	×	○

図４　ソフトウェアの生産性／正当性と手法の関係

同期制御が一つの重要な課題である。多数の並行して動作するプロセスがデッドロックを起こしたりすることは、制御対象の重要な事故につながるからである。この検証といった観点から、同期制御に関して時制論理(Temporal Logic)を用いた研究なされてきた。さらに、正しいプログラムを生成するといった観点からは、時制命題論理(Propositional Temporal Logic：ＰＴＬ）を用いた研究がある。これはＰＴＬには有限回での決定手続が存在するという性質があり、それを利用して、ＰＴＬで記述された同期仕様から状態遷移図を生成する手法である[7]。本システムは、この手法の利点を部品再利用方式と組合せて実現した点に特長がある。

［４］　プロトタイプ記述言語ＭＥＮＤＥＬ

ＭＥＮＤＥＬは、Prologベースの並行オブジェクト指向言語である。ＭＥＮＤＥＬの並行処理単位はオブジェクトであり、プログラム生成における再利用単位もオブジェクトである。各オブジェクトはパイプを通じて互に結合される。オブジェクトとパイプの間にはパイプキャップがあり結合の可否を定める属性を持つ。各オブジェクトはメッセージがパイプを通して或はシグナルゲートを通して到着することで起動される。パイプを通過するメッセージは同時には１個でありパ

イプ上のゲートによって制御される。全体としてはデータストリーム型のプログラムとなる。これらの各構成要素間の関係を図５に示す。以下、各要素について簡単に述べる。

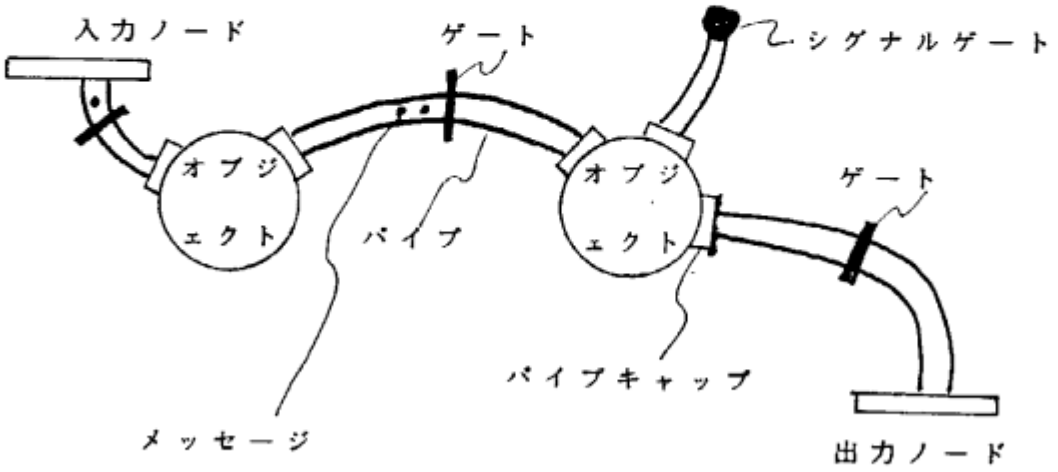


図５　ＭＥＮＤＥＬのプログラム要素

１）オブジェクト

オブジェクトは、宣言部、メソッド部、ジャンク部の３つの部分より構成されている。宣言部では、それぞれ、入力パイプキャップ属性、出力シグナル属性、内部状態変数を定義する。メソッド部では、コミットメントオペレータを用いたガード付メソッドをPrologのゴールとして記述する。

例．method(hour?H.minute!M) ←

H>0 | M is H*60.write(M) nl.

各メソッドは、入力項（この場合hour? の後のＨ）がユニファイされ、かつガード（この場合Ｈ＞＝０）が満たされた時のみ選択され実行される。

２）　メインプログラム

ＭＥＮＤＥＬのプログラムは唯１つのメインプログラムを持つ。メインプログラムの役割は、各オブジェクトを関連づ

け、その各々にイニシャルメッセージを送ること、および入力ノード／出力ノードのデータ入出力を行なうことである。本システムでは、メインプログラムは入出力属性の仕様記述からプラン生成手法を用いて自動生成される。

3）　ゲート及びゲートコントローラ

オブジェクト間を結合するパイプにはそれぞれ唯一のゲートがあり、このゲートでメッセージの流れを制御する。ゲートは1度に1回だけ開けることができ、その時にメッセージが一つだけ流れる。ゲートコントローラは、すべてのゲートの開閉制御を行う。この制御方式にもとづいた同期制御プログラムは時制論理を用いた仕様記述から自動生成される。

4）　シグナルゲート

メッセージを送らずにオブジェクトに起動をかけるために用いる。

[5] 部品検索と部品結合

今回開発したシステムは、生産性向上のために部品再利用手法を用いている。これは対象システムの入出力属性が与えられると、ライブラリに登録してあるＭＥＮＤＥＬオブジェクトをプラン生成手法を用いて検索・結合する方法である。出力属性側からプラン生成を開始し、属性のマッチしたオブジェクトを検索・結合して行くが、必ずしも適切なオブジェクトがすべて用意されているとは限らない。そこで、本システムのラピッドプロトタイピング方式を活かすために次善解をシステムが提示するようになっている。これには、部品ライブラリの中にある属性の意味ネットワークを用いる。本シ

ステムの場合属性間の関係としては、is-a及びhas-a を用いている。

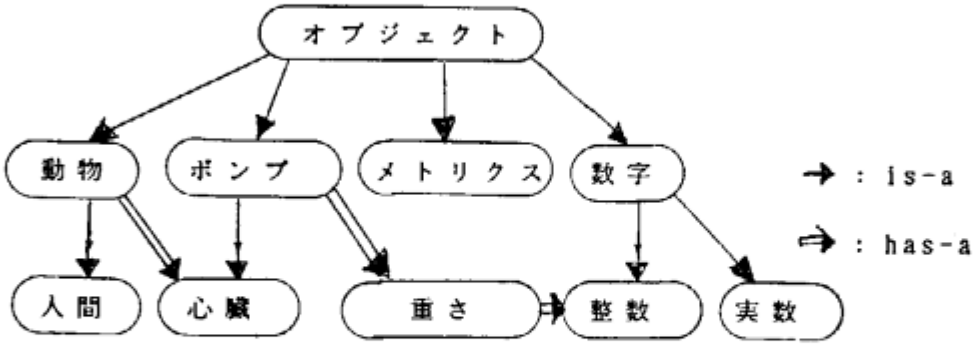


図６　属性ネットワークの例

この属性ネットワークをたどることにより、次善の部品結合案を示すことができる。次善の部品結合は出力側の属性が意味的に入力側の属性に含まれている時に結合可能であるという方式で行なわれる。たとえば、次の２つの属性は、

１）　整数　of　重さ　of　心臓　of　人間

２）　整数　of　重さ　of　心臓　of　動物

入力側が２）で出力側が１）の時のみ結合可能である。なお、この方式による次善案の列は全順序になる(8)。

[６] 同期部生成

並行処理プログラムにおいて、同期の問題は避けられない。かつ、人間にとってあらゆる場合の同期処理を作成することは容易ではない。そこで本システムではシステムの信頼性，正当性向上のために、同期仕様をＰＴＬで記述し、定理証明手法を用いて仕様を完全に満たす手順を作成している。ＰＴＬは、従来の命題論理に次の時相オペレータを追加したものである。□：常に，◇：いつかは，○：次は，Ｕ～：～が成立するまでは、の４種類である。これらのオペレータを用い

て、対象システムの同期仕様を与えることができる。たとえば、◇Pと記述するだけで、プロセスPはいつかは動という仕様を与えられ、Pに関する単純なデッドロックなら防ぐことができる。このように比較的容易に同期仕様を与えることができる。ここでのポイントは、PTLが有限回で決定可能な論理体系である点である。ここで決定可能とは、与えられた論理式を満たす手順（モデル）の有無を定められることである、このPTLをMENDELの言語仕様に対応づけて同期に関する手順を生成している。MENDELでは図7に示すようにゲートの開放とゲートに留っているメッセージの通過をg-´真´で示している。この例は、ゲートg1,g2 が交互に開閉するシステムである。

□（g1→○g2）

□（g2→○g1）

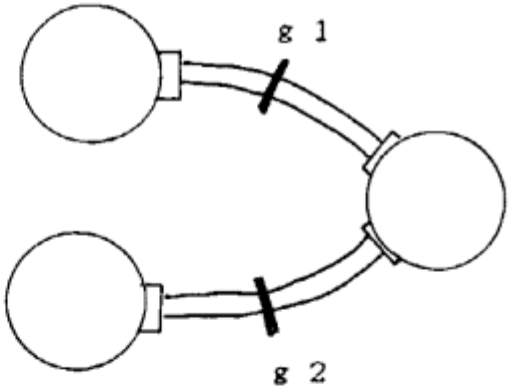


図7　PTLによる仕様記述とMENDEL

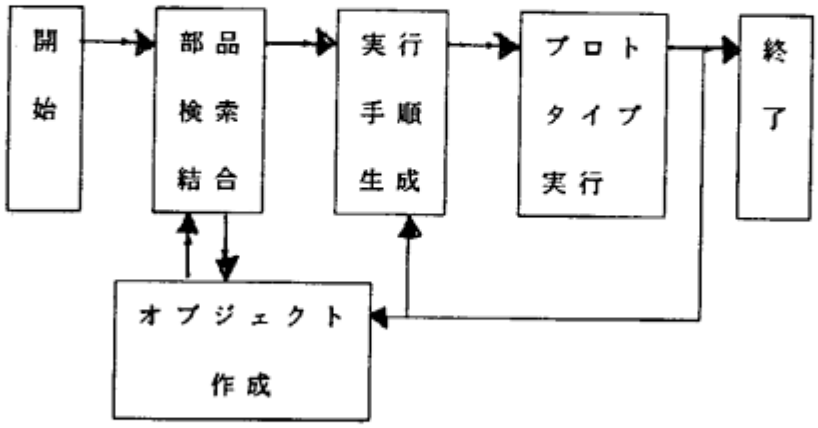


図8　プロトタイプ開発の手順

図８に，本システムを使用してプロトタイプを開発する時の流れを示す。部品検索・結合およびプロトタイプ実行によって各仕様を確認し、不具合があれば実行手順、オブジェクト仕様などを修正して新しいプロトタイプを得る。本システムは、こういったプロトタイピングサイクルを支援するために、豊富なマン・マシンインタフェースを備えている。

[7] プログラム生成例

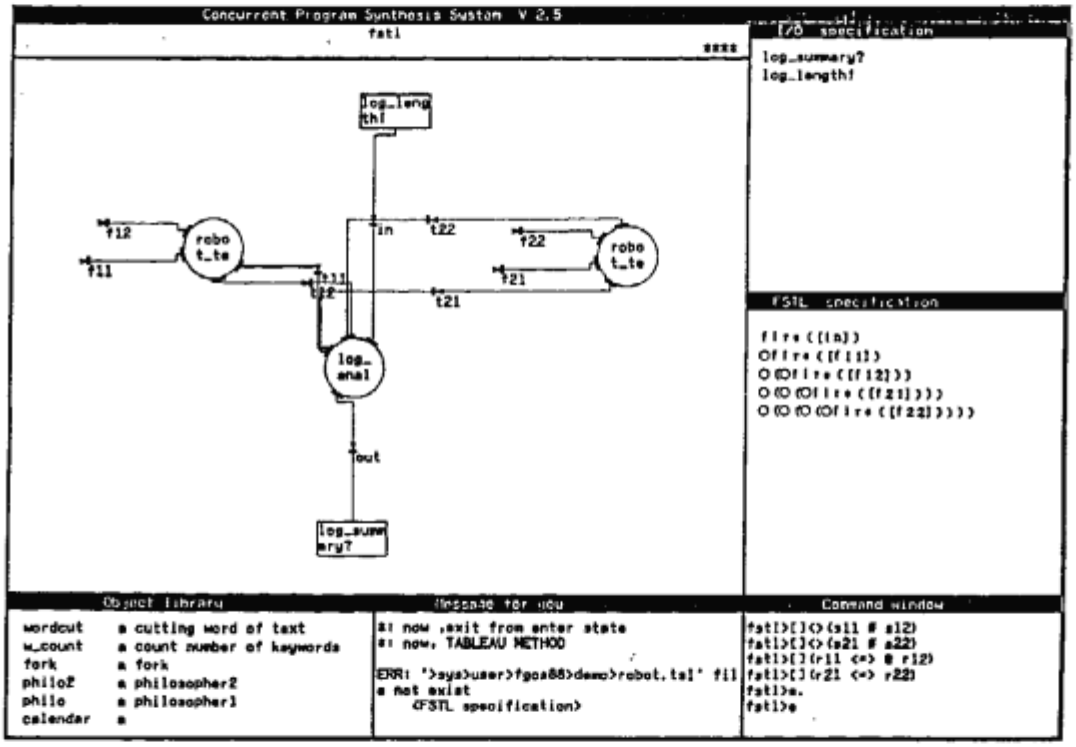


図９　ＭＥＮＤＥＬＳ　ＥＯＮＥの画面例

図９の例題は、左右２つのロボットが組立作業を行い、その作業状況が常時中央のログ分析に送られているシステムである。四角で囲まれた属性log-lengthを入力としlog-summaryを出力するという仕様を与えた時の部品結合例である。この入出力属性を画面右上に表示している。左下は部品群の表示である。右中央は、ＰＴＬを改良した仕様記述言語ＦＳＴ

Ｌによる同期仕様である。右下は会話用のウィンドウである。

図１０の右半分のウィンドウはＦＳＴＬで与えられた同期仕様から自動生成された状態遷移図（モデルグラフ）である。システムは、このモデルグラフを解釈実行する。実行結果が、ユーザの期待と異なれば、仕様や部品を調整する。

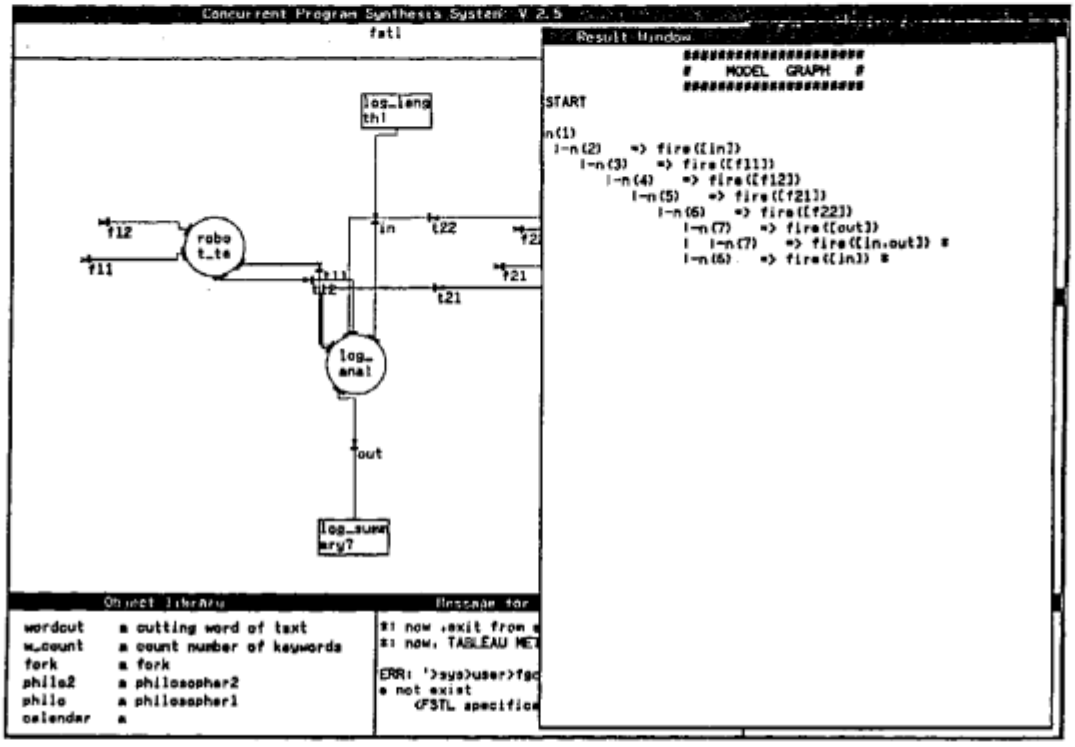


図１０　ＭＥＮＤＥＬＳ　ＺＯＮＥ画面例（モデルグラフ）

［８］あとがき

本稿では、知的プログラミングシステムを一般的に論じ、知的プログラミングシステムのフレームワークを示した。さらに、このフレームワークに従って研究開発した、制御用並行プログラミング支援システムＭＥＮＤＥＬＳ　ＺＯＮＥを紹介し、その技術的特長を述べた。

なお、本システムの一部は（財）新世代コンピュータ技術開発機構（ＩＣＯＴ）の再委託を受けて研究開発したもので

あり、ここに担当研究室の長谷川隆三第一研究室長始め関係各位の方々に深く感謝致します。

文　　献

(1) Smith.D.R.,et.al.:Research on Knowledge-based Software Environment at Kestrel Institute.,IEEE Trans. on SE,Vol. SE-11,No.11,pp.1278-1295(1985)

(2) Balzer,R.:A 15 Year Perspective on Automatic Programming,IEEE Trans. on SE,Vol.SE-11,No.11,pp.1257-1267(1985)

(3) Rich,C.,et.al.:Initial Report on a LISP Programmer's Apprentice,IEEE Trans. on SE,Vol.SE-4,No.6,pp.456-467(1978)

(4) 上野晴樹：知的プログラミング環境－プログラム理解を中心に－，情報処理，Vol.28,No.10,pp.1280-1296(1987)

(5) 本位田，内平，大須賀，柏谷：推論型システム記述言語ＭＥＮＤＥＬ，情報処理学会論文誌,Vol.27,No.8(1986)

(6) Uchihira,N.,et.al.:MENDELS:Concurrent Program Synthesis System Using Temporal Logic,Springer-Verlag,LN315,pp.50-69(1988)

(7) Wolper,P.:Synthesis of communicating process from temporal logic specification,STAN-CS-82-925,Stanford Univ.,(1982)

(8) Uchihira,N.,et.al.:Concurrent Program Synthesis with re-usable Components Using Tempol Logic,COMPSAC,pp.455-464(1987)