54—R014

第５世代の電子計算機に関する調査研究報告書

—— 基礎理論研究分科会 ——

昭 和 55 年 3 月

JIPDEC

財団法人 日本情報処理開発協会

この報告書は，日本自転車振興会から競輪収益の一部である機械工業振興資金の補助を受けて昭和５４年度に実施した「第５世代の電子計算機に関する調査研究」の成果をとりまとめたものであります。

序

わが国における社会経済は、資源、エネルギー問題を始めとして、国際的な変動と、不確実性の流れのなかにある。同時に、的確な情報の加工利用が重要視される情報化社会の形成が指向されている。

コンピュータは、われわれの情報活用においてすでに不可欠なツールとなっているが、今後１０年間には多くの諸問題を解決するため更に高度な技術が要求され、新たな理論、技術にもとづくコンピュータ・システムの実現が望まれる。

このため当協会では「第５世代コンピュータ調査研究委員会」を設置し、１９９０年代に実用化されるべきコンピュータ・システム（第５世代コンピュータ）はどのようなものになるか、またその開発プロジェクトはどのように進めていくべきかについての調査研究を昭和５４年度から２か年の予定で開始した。

本年度は第１年目として、委員会の下部機関として社会環境、基礎理論、アーキテクチャの３分科会、ワーキンググループを編成したほか、大学・研究所への研究委託、米国からの外人講師招聘等により１９９０年の社会シナリオ、技術シナリオを作成するとともに第５世代コンピュータに必要とされる理論と技術について調査研究を行った。

次年度は、具体的な目標設定と開発計画、及び体制について研究を行う予定である。

最後に、調査研究にご協力を頂いた第５世代コンピュータ調査研究委員会委員を始め関係各位に厚く御礼申し上げる次第である。

昭和５５年３月

財団法人 日本情報処理開発協会

会 長 上 野 幸 七

基礎理論研究分科会

（順不同）

	委員名	所属
主査	淵　一博	電子技術総合研究所パターン情報部長
委員	伊藤貴康	東北大学工学部通信工学科教授
〃	大須賀節雄	東京大学宇宙航空研究所助教授
〃	長尾　真	京都大学工学部電気工学第2教室教授
〃	広瀬　健	早稲田大学理工学部数学科教授
〃	古川康一	電子技術総合研究所ソフトウェア部 情報システム研究室主任研究官
〃	米沢明憲	東京工業大学理学部情報科学科助手

目　　次

1．はじめに

1. はじめに — 問題意識

基礎理論研究分科会の役割は、第5世代コンピュータおよびそれをベースとする新世代の情報処理技術体系のイメージづくりについて、基礎理論的な研究のサーベイを通して寄与することであろう。社会環境条件研究分科会、アーキテクチャ研究分科会との相対関係からいって、ここで「基礎理論」というのは、数学的な、といった狭い意味でなく、もっと広く、ソフトウェア基礎論の研究、人工知能の研究、パターン情報処理の研究を包含したものを考えるべきであろう。

それらの分野の研究のサーベイから、第5世代のコンピュータについて、確定した像を導き出せるわけではないであろうが、それに寄与できる段階には来ているように思われる。第5世代コンピュータについて前もっての定義はない。ただ、現行のコンピュータの漸進的改良という線上でなく、その先に飛躍した段階を想定したいという問題意識がある。その第一義的な意識の出てくる由縁は、現行のコンピュータに対する不満であり、それは漠然とした表現では「使いにくい」ということであろう。

そのことは、最外縁のユーザ大衆にとって現状の技術についてあてはまる。その不満は現在の技術の未熟さかて来ているわけであるが、それは現行の技術体系をだんだんと改良していけば解消するものであろうか。あるいは現行のコンピュータの基本的な造りの悪さに起因するものであろうか。

同様の不満は、コンピュータ技術の内部からも出てきている。それは大規模なソフトウェアの作成の困難さということに集約されよう。それは、ソフトウェア工学の未熟さによるのだろうか。それもある。一方、コンピュータの造り自身への反省も出てきている。

そこで第1の問題意識は、

◎ 使いやすいコンピュータ（システム）

ということになろう。使いやすさにはいろんな側面がある。機能の面からもいく

つかの把え方があるだろう。ユーザ側からのニーズとしては、社会環境条件研究分科会から抽出・整理されてくるであろうが、その中に「日本語マシン」の問題意識がある。現状は、日本語の自由な使用という理想にほど遠い。最近ようやく、漢字かな入出力やそれに伴う処理技術が進歩しはじめたが、それはまだ「字」のレベルにすぎない。それが第一歩であるのは確かであるが、それを日本語として、データベースの問合せやプログラミング、質問応答や助言として用いるのは、研究面で試みが始まった段階にすぎない。自由な使用には「意味」の取扱いが可能になってこなくてはならないが、そのためにはまだ多くの研究が必要である。これは、日本語処理ということにも、もっと高機能化が必要であるということである。

日本語は、コンピュータとの自然な会話形態であるが、自然な会話（便いやすさ）ということからは、図形的会話ということも挙げておかなければならない。またコトバの入出力機能としては音声入出力も望まれる。これらの実現には高い機能を必要とする。そこで、

◎ 高機能化・多機能化・高性能化

が使いやすさの一つの条件であり、また問題意識として考えられるべきであろう。

コンピュータの高性能化ということで一番分りやすい例は、超高速数値計算であろう。これはいまや特殊応用ということかもしれないが強いニーズがある。その高性能化からは現行のコンピュータの造りから離れた「非ノイマン型」アーキテクチャへの試みが始っている。

一方、高機能化や多機能化という観点からは、数値計算だけでなく、もっと広く「記号処理」の高度化と高能率化が必要になってくる。それはコトバの処理、ソフトウェア作製などにとって必要なことである。

高機能化にとって今後重要になる観点は、

◎ 知識情報処理

である。現在のコンピュータ技術では、問題解決のほとんどの所を「プログラム」

にしてやらなければならない。これがコンピュータの使い難さにも通じている。プログラムの自動作成は研究テーマの一つであるが、視野を広くとらなければ進歩は望めまい。一つには、ソフトウェア基礎論的検討がもっと必要である。第二には「知識」の組込みが必要になってくる。問題の対象領域についての情報、さらにはそこでの法則性の情報を組み込む方向で、コンピュータの問題解決能力を一段向上させる必要がある。このような情報は「知識」と呼ばれるものに相当する。これはいいかえれば、コンピュータシステムの中に世界のモデルを持たせることである。

これは、人間とコンピュータの相互了解の範囲を拡大することに相当し、コンピュータの問題解決能力を向上させるだけでなく、自然な会話にも必須の能力であり、「使いやすさ」を実現させる基礎である。

言語や知識を介して高機能を実現するということは、７０年代において「人工知能」研究の主テーマであった。これにはまだまだ数多くの研究テーマが残されているが、第５世代コンピュータを考えるときの素材がここに存在しているといえる。

この方向の高機能化の観点からも、新しいコンピュータのつくり（アーキテクチャ）への要望やそれへのヒントが出てきている。

そこで、

◎ 新しいコンピュータ・アーキテクチャ

の問題意識が生じる。これは、

◎ 伝統的な体系からの脱却

という問題意識につながるといえよう。そこで、

◎ 新しい情報処理技術体系

ということが、使いやすさから出発した、総合的なゴールといえよう。

こういった問題意識について、

◎ 基礎理論による裏づけ

が可能かということになる。

コンピュータの誕生には、チューリング機械の理論という（いかにも基礎理論らしい）数理論理学の成果が巧みに生かされた。また生理学でのマカロック・ピッツのモデルは論理素子の概念に結びついている。これらがノイマンらによって巧妙に統合化され、現在のコンピュータの構成原理（ノイマン型アーキテクチャ）が立てられたのである。

しかし、その後のコンピュータは、むしろ工学的に自律的に発展してきた。オートマトン理論、形式言語理論などがあったが、それらがコンピュータの進化を直接導いたとはいいがたい。逆に、コンピュータ工学自体が、内容的には、既存の論理学より豊富なものをつくり出したといえる。

その内容の形式化（理論化）が本格的になったのは７０年代に入ってからである。それは「ソフトウェア基礎論」の分野である。これは現在発展中で、まだ多くの研究テーマを今後に残している。しかし、一方では、新しいプログラミングのスタイルの提案や新しいアーキテクチャへの意識がそれに関連して生じている。

前述した「人工知能」の分野では、人工知能用プログラミング言語という問題意識から新言語の提案があったが、それもまた、新しいマシンへの意識をはらむものである。

注目すべきは、両者の動きが一応独立した展開にありながら、最近ではそれらのベクトルの向きが合ってきつつあるということである。

そういう現象（これについては第２章でやや詳しく触れる）からすると、広い意味での基礎理論が、新しい情報処理技術体系、ひいては新しいアーキテクチャ（第５世代コンピュータ）へ寄与するということは、８０年代において大きな可能性があると思われる。

第５世代コンピュータへ向けての問題意識を整理すれば上のようなレベル設定になるであろう。そういった観点で行われた検討結果からやや大胆にまとめれば、第５世代コンピュータに関して、

知識情報処理システム

その中核としての知識情報プロセッサ

というイメージが浮び上ってくる。後者は、第3章で論じるように、推論マシン（推論エンジン）と考えてよい。

さらに、こういったイメージを実現するために、その過程において、

◎ システムを作りやすい高度の環境

が不可欠であるという問題意識が生じる。これについても第3章で論じる。

これはゴール実現のための手段であると同時に、単なる支援システムでなく、進化するシステムとしての知識情報処理システムの要素として重要視されなくてはならない。それ自身、知識情報処理のプロトタイプの役割も果すのである。これはまた、人材育成（我国では、先進的な研究者、技術者の数が米国等に比してあまりにも少ない）の環境として、層の厚い技術として知識情報処理を育てる土壌となろう。

第3章で論じることの全体像の見とり図として図1－1を作成してみた。これは「仮設」であるが、現段階でのイメージとして、今回の提案としておきたい。

このような構図は先進的にすぎるように見えるかもしれない。しかし、従来技術の漸進的改良の線上に難問の解決方策が見えず、それが新世代への悲観論を生んでもいるが、それは現行技術のある種の成熟現象と見ることができ、むしろ、新世代への機が潜在的に熟していると読むこともできよう。一方これは、これまでの先進的研究の延長上にあり情報処理技術の必然的な方向と見ることもできる。タイミングの問題は別にして、要は立ち止まるか、前進するかにあると思われる。

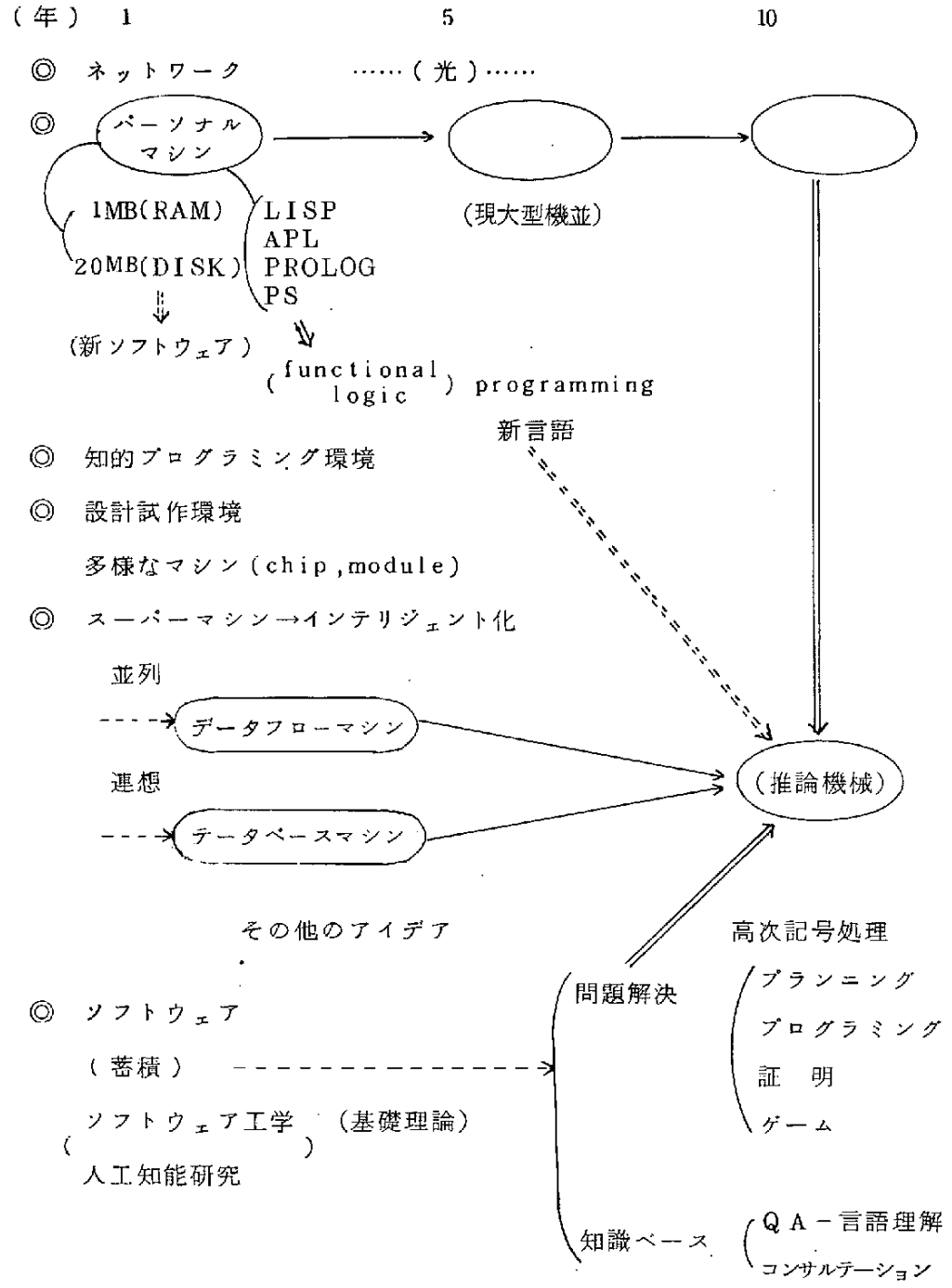

（年） 1
5
10
◎ ネットワーク
……（光）……
◎ パーソナルマシン
（現大型機並）
1MB(RAM)
20MB(DISK)
LISP
APL
PROLOG
PS
（新ソフトウェア）
（functional logic）programming
新言語
◎ 知的プログラミング環境
◎ 設計試作環境
多様なマシン（chip,module）
◎ スーパーマシン→インテリジェント化
並列
データフローマシン
連想
データベースマシン
（推論機械）
その他のアイデア
高次記号処理
◎ ソフトウェア
（蓄積）
ソフトウェア工学
（基礎理論）
人工知能研究
問題解決
プランニング
プログラミング
証 明
ゲーム
知識ベース
Q A－言語理解
コンサルテーション

図 1 － 1　開発の構図（案）

2．基礎的諸研究の相互関連

2. 基礎的諸研究の相互関連

７０年代は、コンピュータ関連技術がそれぞれに興味深い展開をした時期と見ることができる。そしてその後期においては相互の関連が段々見えてくるようになった。このことから、８０年代の総合的な展開と９０年代へ向っての結実の準備が進んできたとみることができ、新世代の情報処理技術、それの核となる新しいコンピュータ像を予感させるものがある。

特に、ＬＳＩ技術の進展は、技術上は革命的とはいわないまでも飛躍的であり、そのインパクトは革命的と予想され、新しい時代への土壌を提供してくれている。新時代の責務は、それをいかにして真に活用するかということであろう。

ここでは、７０年代に展開された基礎的な研究の進展状況とその相互関連を探ってみたい。それぞれの項目の詳細は、ワーキング・グループの報告に集成されている。本章では、その概況と相互関連の可能性に重点を置く。

2.1 新しいコンピュータ・アーキテクチャの試み

コンピュータ・システム全体についていえば、分散処理とかネットワーク化が新しい方向として、７０年代に進展してきた。一方、コンピュータ自身の内部構成の基本原理はノイマンの発明以来あまり変化していないといわれる。その上、諸々の事情からＩＢＭ Ｓ／３７０などの「標準アーキテクチャ」への固定化が進行してきた。分散処理のようなシステム構成のフィロソフィーの変化は、コンピュータ・アーキテクチャ自身に影響する可能性はあるが未だ顕在化していない。

コンピュータ・アーキテクチャ自身の研究は古くから続けられている。連想処理、並列処理、可変構造などの提案は昔からある。スタック方式、タグ方式、ハッシュ方式など、手法上の提案も数多い。また、高水準言語にのっとったアーキテクチャという考えも以前から提唱されている。

しかし一部に実用化されている手法（スタックなど）があるとは言え、現時点

まででは必ずしも実を結ぶ段階には達していない。ボトムアップ的な発想による試作品はプログラミングの困難などに直面して、成功という評価が得られなかった。一方、高水準言語機械の考えも実現法がコンベンショナルではとり立てての有利性が実証されず、また採用する言語の構成自体、ノイマン型を前提にするものでは循環論法的である。

これまでの所の実績はむしろノイマン方式の強固さを再確認してきたように見える。ここで、現在の実用的汎用コンピュータのすべての基本原理であるノイマン方式について若干考察しておこう。

ノイマン方式の原理的な構成法は、一様な構造の記憶装置と単純な構成の演算制御装置の組合せにある。複雑な高度の機能の実現は「プログラム」にまかされる。プログラムは単純な構成の命令群からなるが、そのプログラム自身、一様構造の記憶装置に収納され、その実行法はシリアルであり単純である。

ノイマン方式の本質は、汎用マシンを単純な構成で実現するところにある。それは当時の、また最近までの電子技術の水準に合ったものであった。たとえばラムダ計算などを直接実行するマシンから始めたとすれば、汎用コンピュータの実現は大巾に遅れたであろう。その後は、この単純な構成の上で各種の拡張が試みられてきた。

ノイマン方式の良さも、一方での不満もこの構成法に起因する。

最近の不満でもっとも大きいのは、プログラムが複雑で困難になることである。大規模なソフトウェア（プログラム）でその困難がとくに大きくなる。ノイマン方式は、各種の改良、拡張があったとしても高機能の実現は原理的にほとんどをソフトウェアにしわよせする。高能力化のための階層的な記憶装置や複数の入出力装置の存在は、アーキテクチャを複雑にしている最大の原因だが、その複雑化は同時にソフトウェアの複雑化を招いている。

分散処理等の考えはこの状況を変える要因にはなりうる。しかし、プログラムをごく原始的な命令から組上げなければならないという構成原理が宿命的なのである。

プログラミングについて高水準言語の採用はその解決策として従来進められてきた。それにしても、コンパイラ等の言語処理系が必要でソフトウェアの肥大化を招く。

一方、高性能化、高機能化のために、並列処理や連想機能が考えられてきた。ところがそれらの考えをベースにした試作品もまた、もっとひどいプログラミングの困難さを招いた。アレイ構成やマトリックス構成のコンピュータが超高速数値計算用に試作された。それを専用機と位置づけても、プログラミングの困難さは致命的であった。連想機能については、連想記憶装置を実現する技術的困難も関係している。

この状況は最近変化しようとしている。一つにはＬＳＩの長足の進歩があり、複雑なハードウェアの実現が可能になってきている。それとともにＬＳＩの利点を活かすのにノイマン方式がネックと感じられるようになってきている。また、構成法に関して新しいアイデアの出現がある。一方、プログラミングについても、そのスタイルについての反省が進められてきた。

並列コンピュータの構成法について、データフロー型アーキテクチャが最近注目されるようになってきた。データフローの概念は計算過程を制御中心でなく、データの流れに沿って記述するものである。並列な演算をデータの流れに沿って素直に記述し、それをハードウェア的に実現しようとする。しかし、そのままでは専用プロセッサになってしまう。可変的な構造を実現するために、デニスはメッセージ交換の考えに基づいた構成法を示した。ここに、並列処理と可変構造の考えの融合が見られる。それは、構成法からすれば、ノイマン型とは全く違った方式になる。

興味深いのは、最近になってこれにソフトウェア工学側の発想との融合が見られるようになってきたことである。データフローは関数の概念をベースにする。一方後述するように、ソフトウェア工学の中で、プログラミングの困難から抜けだすために、新しいプログラミング・スタイルが提案されている。その中にバッカスの関数的プログラミングがある。これがデータフローマシンの考えと結びつ

くのである。

これは従来の、並列マシンのプログラミングの困難さという固定観念からすると、画期的な状況変化である。バッカスの提案自体は、非ノイマン型のプログラミング方式の提案として、ハードウェアの存在は必ずしも前提としていない。しかし、それがハードウェア構成法としてのデータフローマシンと結びつくのである。

実際には、関数型プログラミング言語について、それに自然であるような中間言語（単一割当て言語、Single Assignment Language, ＳＡＬ）が存在し、このＳＡＬがデータフロー言語と自然に結びつく。このＳＡＬ自身、データフローマシンとは全く別の動機で、ソフトウェア工学で、検証容易な言語として提案されたものとも合致するのである。

さらに、このＳＡＬは、推論システムの研究（結合グラフ法）から導びかれるものと類似している。デニスらのデータフローマシンはさしあたり数値計算用と想定されているが、ここにおいてさらに大きな可能性が見出される。第３章で論じるように、データフローマシンを、推論マシンへと拡張していく可能性が見出されるのである。

これは、後述するような、論理的プログラミングの考えや形式的仕様言語をカバーし、また関係データベース（それに基づく、データベースマシン）の話題を包含し、さらには自然言語の意味論にも接続する可能性を示す。これが実証されるかは今後の課題であるが、このストーリーは非常に魅惑的なものである。このような機械が実現すれば、多くの話題がマシンレベルまで含めて有機的に統一されることになる。

2.2　新しいプログラミング・スタイル

コンピュータ技術においてこの所ますます大きな問題になっているのは、ソフトウェア（プログラム）の生産性の問題である。コンピュータシステムのコストの中でソフトウェアの占める割合が大きくなり、その大半を占める状況になってき

ている。それは単にコーディングの量が増えるからだけではない。プログラムの品質を保証するためにますます多くの労力が必要になってきているのである。

そこでのポイントの一つは、プログラムの「正しさ」を保証すること、また、正しいプログラムを作成する手法を開発することである。プログラムを要求仕様に合わせて（正しく）作る方式を確立することがソフトウェア工学の提唱の大きな問題意識になっている。

「仕様」を明確に提示するには、仕様の記述法（仕様用言語）の問題がある。そのためにはプログラミング言語より高水準の言語が必要である。

いま仕様が明確に与えられたとして、それをもとに正しく効率のよいプログラムが自動的に合成できれば、それは理想であろう。しかしその理想の実現はまだ先である。その前に、たとえば、正しさを保証しつつ段階的にプログラムを作り上げていく手法を確立することが必要であろう。それはプログラムの作り上げ方（スタイル）の問題といえる。

ダイクストラの「構造的プログラミング」の提唱以来、プログラミング・スタイルの反省が行われるようになってきた。

このような問題意識に立つとき、現在常用のプログラミング言語は適当であろうか。スタイルの反省は、プログラミング言語のあり方の反省を伴う。この流れの中から最近「論理的プログラミング」と呼ぶべき提唱が出てきた。一つは、述語論理プログラミングの提唱であり、もう一つは関数型プログラミングの提唱である。言語として、前者は、述語論理形式をとり、後者は関数型論理をベースにする。そういう論理形式自身をプログラミング言語として用いようという提案である。

これは、形式的仕様言語の二つの流れ（関数型と述語論理型）に対応するとともに、仕様言語とプログラミング言語のあり方について反省をうながすものである。形式的仕様は広義の（抽象度の高い）プログラムと見なすことができる。この抽象プログラムは、システムの条件に合わせて具体化されていく。その変換が同一言語（論理）の上でなされうるならば、多くの利点が生じるであろう。この

線に沿った研究が始められつつある。このような考えは、ソフトウェア工学の大きな流れの一つでもある「データ抽象化」の考えとも今後結びついていく可能性がある。

「プログラムの変換」は正しさ（意味の同一性）を保証しつつ行われる必要がある。「検証」（システム）は、ソフトウェア工学の大きな課題の一つでもある。しかし、一方のプログラミング言語に、慣行の言語を選ぶと、種々の困難が生じる。このため、検証容易なプログラミング言語の提案がある。たとえばアシュクロフトらのＬＵＣＩＤがある。これは前述のＳＡＬの一種である。

関数的プログラミング言語と述語論理的プログラミングは、スタイルにおいてやや異なるが、実はその間に自然な対応関係がある。その対応の中間言語として出てくるものは、ＬＵＣＩＤのようなＳＡＬ（の拡張）になっている。

ＳＡＬがデータフロー・マシンの考えと結びつくことは前述のとおりである。このことからデータフロー・マシンを拡張した「推論マシン」が構想されるのである。

2.3　プログラミング言語の意味論

プログラムの正当性の検証ということは、「意味」が同一であることを確認することだといえる。このためには、プログラミング言語の「意味論」の確立が必要である。

プログラミング言語のような「人工」言語では、その意味規定も単純であろうと思われるかもしれない。しかし、それは歴史的にいっても結構難航したのである。人工言語の構文法は「人工的」である（単純である）といえるが、その「意味」はどうも「人工」ではないようである。それは、自然言語の意味論の難航と並行していた。

６０年代から７０年代にかけて、いくつかの意味論の試みがなされてきた。一つは、操作的意味論である。それは、抽象的機械を想定し、そこでのインタープリータの記述によって、言語の意味を与えようとする。これは６０年代後半、ウ

ィーン・メソッドとして展開された。しかし、これを検証に用いるには、大きな困難があった。

７０年代に入って、ホーアらによって、公理的意味論が始められた。これは、６０年代後半のフローチャートに対するフロイドの理論を発展させたものである。この場合、公理化される言語の選択が問題になる。それとともに、公理系および推論規則の「正しさ」も理論的には問題になる。

これに並行して、スコットらによって指示意味論が展開された。これは、モデル理論に基づいて、言語の意味論を数学的に厳密に設定しようというものである。スコットの理論は、公理的意味論や操作的意味論を裏づけるものとも考えられる。それをベースに、「ソフトウェア基礎論」が最近展開されはじめている。

これらの試みから判明してきたことの一つは、慣行のプログラミング言語は、意味論的試みから見て難物だということである。それは、意味論の立て方が悪いのであろうか。興味深いことは、そのような理論的アプローチとは独立に直観的になされた、プログラミング・スタイルへの反省事項が、意味論上での難点と合致するということである。

たとえば、構造的プログラミングで、ＧＯ　ＴＯの使用が好ましくないといわれた。一方、この部分の意味論的構成も難しい。

これらのことは、新しいプログラミング・スタイルとそれを支える新しいプログラミング言語という方向を裏づける。従来型の言語の難点は、バッカスの指摘では、ノイマン型マシンを前提にしていることにある。新しい方向は、非ノイマン型言語への転換ということであり、それは非ノイマン型コンピュータへの期待につながるものである。

前述した関数的言語、述語論理的言語ではその意味論はむしろスッキリしたものになる。そして、それは、データフロー・マシンのような非ノイマン・アーキテクチャに結びつくのである。

2.4　関係データベース

データベース・システムの展開は、７０年代のコンピュータ技術の発展の中核的要素の一つである。大量のデータをどう整理しておき、どう利用するか。経験のつみ重ねで進んできた。それとともにそれを理論的に整理する試みも進んできた。

コッドの関係データベースの提案は７０年代初頭のものであるが、データベース構成法として大きな流れになろうとしている。これは「関係」の概念をベースにしている。その検索用言語としては、述語形式（関係計算）と関数形式（関係代数）が提案されている。両者は相互に変換可能である。これは、ある特殊な論理系と見なすことができ、７０年代を通して、理論的な研究が盛んに進められた。

前述したように、プログラミングの立場から、論理形式の提唱が行われている。関係データベースの考えはこれと密接に結びつく。両者の関連と融合化は最近興味深い研究テーマになりつつある。

関係データベースをもとにした「データベースマシン」の研究も７０年代に盛んになされた。しかし、実用レベルに達したものはまだない、といえる。関係データベースは理論的にスッキリしたものであるが、試作されたマシンの構成法は経験的でアドホックなアプローチによっていたように思われる。

しかし、データベース・マシンは、新しいアーキテクチャのマシンへの芽として期待する向きが多い。ただ、８０年代でのデータベース・マシンの研究は従来の試みから一段と飛躍する必要があろう。それへ向けての注目すべき研究成果が最近現われ始めている。

すでに指摘した論理的プログラミングとの関連から、それは、データフロー・マシンと関連することが予想される。実際、データフロー型データベース・マシン（関係代数マシン）の提案が最近現われている。これはまだ「専用」マシンの段階であるが、汎用化へと進む可能性を持っている。データベース上での高次の検索は、まさに「推論」そのものである。データベース・マシンの立場からも、推論マシンへの可能性が導きだされるといえる。

現在、データベース用言語とプログラミング言語は、別体系になっているのが普通である。これは望ましいことではない。それらを一体化することは、今後の課題であるが、これまで論じたように、それは十分可能であると考えられる。それは新しいソフトウェア体系の出発点になるであろう。それとともに、マシンのレベルを含めて統一的な観点に立つ有機化が期待されるのである。

2.5 言語学から

プログラミングやデータベース問合せに自然言語を使いたいという要望が高まりつつある。しかし、それはコンピュータとの会話形態を高度化したいという気持ちであり、自然言語の理論（言語学）とコンピュータ・システムが内面的に関連してくると考える人は現在あまりないように思われる。それはどうであろうか。

言語学としては、言語の実質を究めるのが本務であろう。それとともに、言語モデルをたてる理論言語学がある。ここでは、理論言語学での動きを、その枠組の面から眺めてみたい。

６０年代はチョムスキー理論をベースにした言語学の大展開があった。それは「生成変形文法」理論と呼ばれている。深層構造を設定し、変形規則によって、言語の表層構造を導くという枠組である。それは、言語の統語的現象を説明しようとする所に主眼があった。その枠組は７０年代に入って大きな変化を受けはじめた。それは、「意味」への関心が前面に出てくる段階に来たことと関連がある。生成変形理論は、生成意味論と解釈意味論の二大流派に分れることになる。この状況は、ワーキング・グループ報告書に記述されている。ここでは、理論の枠組みに種々の変形が生じていることを指摘するにとどめる。

一方、７０年代の初頭に、モンテギュによる新しい言語理論の出現があった。これは、言語哲学の流派から出てきたもので、意味論の構成法を直接に意識するものである。それは言語哲学の中の形式意味論を発展させ、英語の一部分について適用したもので、統語論と意味論をはじめて一体化したモデルということができる。この問題は古来から難問視されてきたものである。

主流的な言語学では、この問題はむしろ避けて通っていた。意味の扱いは、現象的な分析との関連で部分的に提示されるだけで、意味部門全体の構成法を明示的に提示することはなかったのである。

モンテギュ理論について、言語学プロパーからの評価はいろいろに論じられている。否定的な意見もある。しかし一方では、交流も始まっている。生成意味論では、そのモデルの類似性からそれが盛んである。一方、チョムスキーの属する解釈意味論も、それに並列して独自に、論理学との結びつきを探めている。

モンテギュの理論は、論理学的な言語理論である。そこでは、ベースの論理として、意味論の明確な「内包論理」が導入され、自然言語の文をその内包論理に変換する手続きが与えられる。この枠組で、それまで自然言語への適用が疑問視されて来た形式意味論の適用法が示されたのである。

内包論理は、様相論理を「関数形式」化したものである。様相論理の意味論（モデル理論）はクリプケによって導入された。これは、「可能世界」をベースに論理式の意味を規定しようとするものである。

このような動きをコンピュータ技術の側から眺めるとどうなるのであろうか。

関数型の論理はコンピュータの世界ではなじみの深いものである。ＬＩＳＰという言語はそれをベースにしている。関数型言語は、関数的プログラミングの提唱により再浮上してきている。内包論理は、それの一種の拡張に相当しているのである。

また、モンテギュ理論の構成法自体、全く別個の意識で作られたＬＩＮＧＯＬという言語解析システムと酷似している。このように、モンテギュ理論は、コンピュータ技術の中から生れたものと（偶然にも）大きく重なり合うのである。

前述したプログラミング言語の意味論も源流は形式意味論にある。興味深いことの一つは、両者におけるそれぞれのブレークスルーの時期が（これまた偶然にか）ほぼ同じ時期にあるということである。

さらに、内包論理のモデル理論（クリプケの可能世界モデル）は、データベース的な観点から見ると直観的によく了解されるものでもある。

コンピュータとの関連で要望される自然言語の範囲は、文学の世界と異って、論理的に把握可能な部分に限られるであろう。それが不自然な制限であれば、自然言語という趣旨に反するであろうが、その範囲は十分に広いと考えられる。この観点からは、論理学的な言語学の発展は、コンピュータ側からも大いに期待される。それとともに、種々の重なり現象が観察されることから、そのような言語学とコンピュータ技術は、内面的にも非常に密接な関係にあることがうかがわれる。この関係を現実のものにすることは容易とは言えないであろうが、しかし、将来に向けてその可能性は大いに期待すべきであると思われる。

2.6 人工知能研究の流れ

人工知能研究の歴史は古い。その中にもいくつかの流れがある。７０年代に入って、人工知能研究の流れは大きく転回したように思われる。そこでの特徴的なことは、「言語と知識」という問題意識である。このような問題意識は、現場のコンピュータ技術の側からは高踏的だと思われたようである。ソフトウェア工学、データベース論などには、それに対抗した問題意識があったように思われる。しかし、最近になって、それらの動きは大きく重なりはじめ、融合していく傾向が現われてきている。

７０年代の人工知能の幕あけは、ウィノグラードの質問応答システムの出現に象徴される。これは、限定された積木の世界を対象にしたものであるが、背後に知識のモデルをもち、自然言語によって質問応答するものであった。

それ以降、自然言語の理解のモデルとそれに関連した知識の表現法、利用法ということが人工知能研究の中心的な問題意識になってきた。それを通して、言語学や心理学との交流が盛んになってきた。それとともに、実現方式の研究は、コンピュータの利用形態の高度化の試みとも重なり合うようになってきた。

後者の流れの中に、人工知能用プログラミング言語の問題がある。ＰＬＡＮＮＥＲにはじまるこの種の提案は、プログラミング言語自体の拡張方向であるとも把えられる。

ＰＬＡＮＮＥＲは推論システムのプログラミング言語化として試みられた。それには、パターン照会機能、非決定処理、多重データベースの着想があった。一方、証明システム（推論システム）の研究のその後の進展によって、PLANNERのような機能を直接推論システムとして把えなおせることが判ってきた。前述した述語論理的プログラミングの提唱はそれにあたる。その考えも基づくＰＲＯＬＯＧなどは、ＰＬＡＮＮＥＲを論理的に再整理したものということができる。このことは、（プログラム言語の）意味論という観点からも有意義である。ＰＬＡＮＮＥＲの明確な意味論は判然とせず、それが定着しなかった一因でもあろう。

ＰＬＡＮＮＥＲ自身はその後、アクタ・モデル（ＰＬＡＳＭＡ）に転進している。これは、メッセージ交換の考えに基づいて計算過程をモデル化するものである。このスキームは、データフローの考えの拡張形態であると把えることが可能である。両者は、別々の動機に発したものである。７０年代の研究を振りかえってみると、独立した動機からの研究が相互に関連し合うようになるという現象がいくつも見られ、これもその一例である。

知識表現の研究は、人工知能研究の大きなテーマであるが、これについても、データベース研究との関連が強くなってきている。データベースの高度化の問題意識とそれが結びつくようになってきている。従来はこれらは別々の研究グループによって進められてきたが、両者を統合して把握しようという試みが見られるようになってきた。データベースの高次な構成法の問題は、前述した内包論理と可能世界モデルとも深く関連すると考えられる。

人工知能研究には、自然知能の解明という問題意識があり、それは現在、認知科学の提唱となっている。それと同時に、人工知能研究は、コンピュータ技術の高度化を先どりする役割をも担ってきた。

７０年代を通じてみると、人工知能研究を含めて、コンピュータ技術の研究はいくつかの流派に分れて、対抗しつつ進展してきたと言える。それとともに、その後期にはそれらの相互関連が生じ互いに融合する傾向が生じてきた。これは80年代へ向けての貴重な芽であり遺産であるということができよう。

３．開発の構図(案)

3. 開発の構図（案）

本章では、１９９０年代の先進的な情報処理システムを開発していくための道程を明らかにする。はじめに、３．１で最終目標である知識情報処理システムのイメージを与える。３．２では、当面必要となる研究環境について述べる。３．３では、知識情報処理システムの核となる知識ベースと推論機構からなる推論システムの開発過程について述べる。３．４では、そのような推論システムを実現するのに適したプロセッサについて述べる。推論システムが実用に供するためには、現存するプロセッサに比して、機能的にも速度的にも飛躍的な向上が必要である。そして、そのためには、全く新しいアーキテクチャに基づくプロセッサが要求されている。本報告書では、データフロー型アーキテクチャの有する高度並列機能に焦点を当てて、主として、その線に沿ったハードウェアの開発スケジュールを示す。３．５では、知識情報処理システムの応用として、知的プログラミング環境の実現のための開発過程を提案する。３．６では、同じく本システムの応用として、知的ハードウェア作成環境の実現ステップについて述べる。これらの２つは、知識情報処理システムの開発のための道具ともなるもので、２重の意味で、重要なテーマである。

最後に、他の応用システムについて、概説する。

3.1 知識情報処理システムのイメージ

3.1.1 機能イメージ

知識情報処理システムをブラック・ボックスと考えたときの機能を、Man－Machine Interface および問題解決能力の２点について、列挙してみよう。

(1) Man－Machine Interface 機能

Man－Machine Interface については、第１に、自然言語、図形などを

介して計算機と対語できること、があげられる。自然言語の入力手段としては、タイプ入力は勿論のこと、文字の読み取り、さらには、音声入力も可能であるとする。図形の入力手段としては、図面や写真、地図などの読み取り、自由手書きの実時間入力などが可能であるとする。このような対話機能を実現するためには、知識情報処理システムは、自然言語や図形などを介して伝えられるメッセージの意味を理解できなければならない。

第2に、メッセージの意味の理解に止まらず、システムとのよりよいマッチングを図るために、質問の誘導、助言などの能力を必要とする。あるいは、質問の意図を察して、たとえば簡潔に要約のみを答えるなどの答え方を考えることも必要である。これらの能力は、知的な対話能力ということができよう。

(2) 問題解決機能

問題解決は、将来は人間と計算機の協調システムがより大きなウエイトを占めるであろう。それは、ＣＡＤに代表されるような計算機の使い方である。その場合、(1)で述べたMan−Machine Interfaceの改善が非常に大切となる。

ここで、問題に対する人間と計算機の相互理解の問題が重要となってくる。人間が問題解決過程に自由に干渉できるためには、システムは、人間の理解に合わせた形で問題を理解していなくてはならない。これは、問題が形式化されている問題領域に関する巾広い知識をシステムが持っていることに他ならない。人間−機械系による問題解決方式では、人間の役割は、問題解決の途中で必要となる戦略の決定である。すなわち、ある場面で、取り得る方法が複数個あったとき、どの手を選ぶかである。これは、仮説の設定と考えられ、設定された仮説は、機械系によって検定される。勿論、仮説の設定自身を、機械系によって行わせられる場合もある。この、仮説を設定して検証する機構は、将来の知識情報処理システムが持たねばならない大切な機構の１つである。この機構は、論理的には、不完全な知識の扱いに関連しており、非常に困難な課題として知られている。

問題領域についての新たな知識を付加する機能も重要である。その中には、いくつかの異なる問題が存在する。その第1は、与えられた新たな知識を既存体系に組み入れる問題である。そのとき、矛盾が生じるかも知れず、そのような場合を検出することと、矛盾を取除くことが大きな問題となる。第2は、生データの集合から法則性をつかみ出すような、機能的推論あるいは学習機能である。第3は、問題領域についての人間の有する知識をシステムに注入する問題である。人間は必ずしも、きちんと整理した形で物事を理解しているとは限らない。この問題は、ソフトウェア工学における要求分析の考えと似ている。すなわち、知識情報処理システム利用のための支援システム自身を、(別の)知識情報処理システムとして実現することが期待される。

3.1.2 システム・イメージ

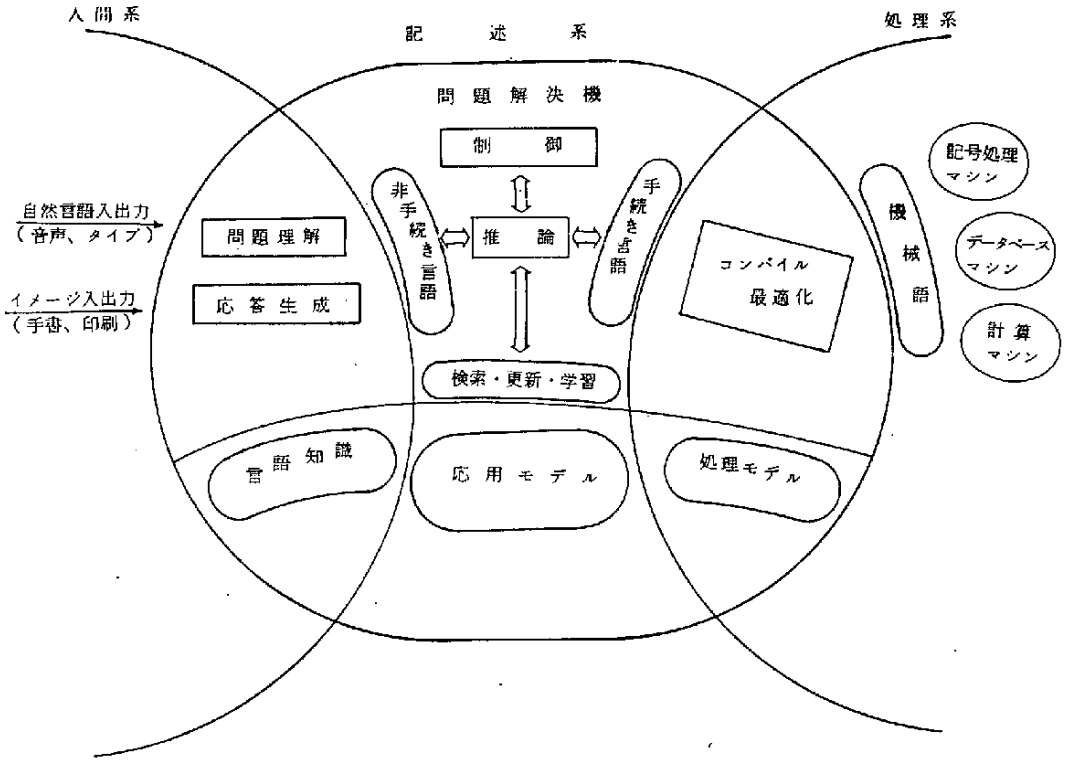
前項で述べた機能を実現するためのシステムのイメージを、図3－1に示す。

本項では、とくに、このようなシステムを実現するための、ソフトウェアおよびハードウェアの満たすべき要件を列挙する。

(1) ソフトウェアの要件

知識情報処理システムのソフトウェアは膨大なものとなり、また、学習機能に見られるように柔軟な拡張機能が要求される。処理の内容は、高度な非数値処理が要求される。問題解決システムの記述には、非決定的なアルゴリズムが本質的である。しかも、これらの機能を高効率で実行させたい。実行速度はハードウェアに負う所が大きいが、ソフトウェアの構造、品質にも大きく依存する。これらの諸要求は、既存のソフトウェア技術では、満足させることは困難である。

新しいソフトウェア技術の核となると考えられる第1のアイデアはデータ抽象化法と、それに基づく階層的プログラミング技法である。これは、問題の複雑性を $n_1 \times \cdots \times n_m$ から $n_1 + \cdots + n_m$ に減少させる、画期的な考えである。第2のアイデアは、ルールに基づくプログラム変換技法あるいは、記号レベルで



Key technology

音声理解
画像理解
自然言語理解
知的対話処理
知識表現
メタ知識
推論根拠管理
計算の意味論
データ構造論
メッセージ・フロー計算機構
VLSI
ハードウェア設計

記号実行
PROLOG
関係代数

ソフトウェア設計

人工知能

システム設計

データ抽象化

図３－１．知識情報処理システムの概念図と実現のためのKey technology

のプログラム実行法である。これにより、抽象具体変換が自動的に行える。さらに、変換過程がルールにより表現されるので、汎用性が生じる。そして、実は、これら2つのアイデアは、知識情報処理システムによって容易に実現される。この再帰的な性格こそ、情報処理技術のもつ大きな利点なのである。

(2) ハードウェアの要件

問題解決を行う推論システムは、すでに述べた通り、高度な非数値処理を必要とする。そのための専用プロセッサ、すなわち、記号処理マシンの開発が望まれる。記号処理マシンの持つべき機能は、記号列のパターン照合機能、非決定的処理のための自動後戻り制御機能、それに伴うガーベッジ・コレクションなどの記憶管理機能などである。非決定的処理をより高速に行うための並列処理機能も重要である。この方向の研究に、ＬＩＳＰマシン、ＰＲＯＬＯＧマシンなどがある。

第2の重要なコンポーネントは、データベース・マシンである。データベースとしては、推論システムとの整合性などから関係データベースが最も有望であり、そのマシンとしては、関係代数演算を直接実行するマシン（関係代数マシン）が望ましい。

ハードウェア化、あるいは（超）ＬＳＩ化のための技術、すなわちハードウェア設計技術の高度化も合わせて必要となる。そして、ソフトウェア開発システムの場合と同様に、高度なハードウェア設計システムは、それ自体、知識情報処理システムによって、実現され得るものである。

3.2 ＬＩＳＰマシンとローカル・ネットワーク

全体像を図３－２に示す。全国の主要大学、研究機関をネットワークで結び、研究の共同化、研究成果の共有化の場を作る(a)。各所には、(b)のようなLISPマシンとローカル・ネットワークによる情報処理研究用のラボラトリ・オートメーションの設備が設けられる。実験ステーションの機能、性能は次の通りである。

まず、LISPマシンについて述べる。その言語仕様であるが、蓄積されたプロ

グラムとの互換性を重視して、現在広く使われているLISPのままにするか、今後の使い良さを重視して、新しい言語仕様とするかが、大きな選択肢である。新しい言語仕様としては、LISPにClu、PROLOG等の機能を無理のない形で融合させたものを想定することができる。いずれにしろ、システム記述としての言語仕様も利用者に開放し、システムの拡張、変更等を自由に研究の一環として行なえるようになっていなければならない。インタプリタ、コンパイラの他に、画面編集の機能をもつエディタ、記号実行の機能を持つデバッガ、実行状態を逐次図表示できるトレイサ、ドキュメント作成・管理システム等が用意される。

ハードウェア構成としては、LISPマシン本体は、1M以上の自由セルを確保できるだけの主記憶を持ち、現在の大型機同等以上の処理能力を持っているものとする。

10Mbyte以上の、ローカルなファイル記憶を持ち、漢字カナまじり文、簡単な図形の表示機能を持つ、高性能なディスプレイ装置が利用者とのインタフェースを務める。＋α部分であるが、日本語処理実験ステーションに対しては、使い勝手の良い、日本語入力装置や、高速な辞書検索を可能にする、ローカルなデータベース・マシンなどがつけられる。ハードウェア研究用実験ステーションに対しては、3．6節を参照されたい。

このようなLISPマシンとローカル・ネットワークのシステムを出来るだけ早く、（4～5年内に）実用化し、全国に配置することが急務である。なぜ、LISPマシンとローカル・ネットワークが必要であるかを、技術の流れの中で位置つけたのが、図3－3である。そして、その流れの中の主要技術の日米比較が図3－4である。日本の現状が、いかにアンバランスで不健全であるかが明瞭に示されている。この不健全さは、ここ5～6年の日本の情報処理研究、新技術開発の遅れとし、歴然とした日米ギャップとなって表われている。このギャップを埋める第一歩が、国家プロジェクトによる、LISPマシンとローカル・ネットワークの全国配備である。表3－1の日米比較は、その差の大きさを示してはいるが、希望的な目でみれば、きちんとした方向づけさえ与えてやれば、さほど困難なく

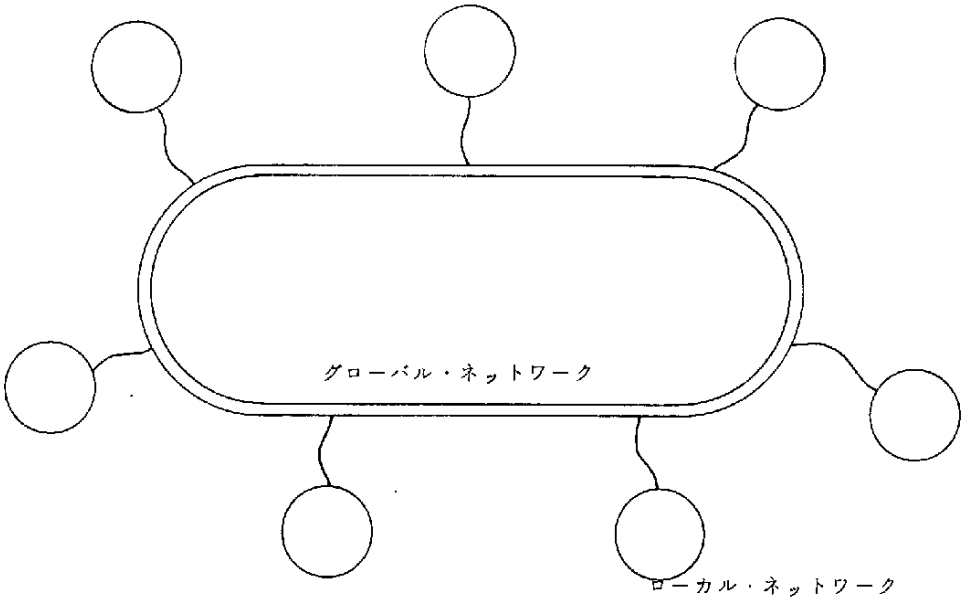
追いついていけるだけの技術的な蓄積をしているとも見ることができる。大局的な視点の欠除が最大の日米ギャップの要因であるともいえる。

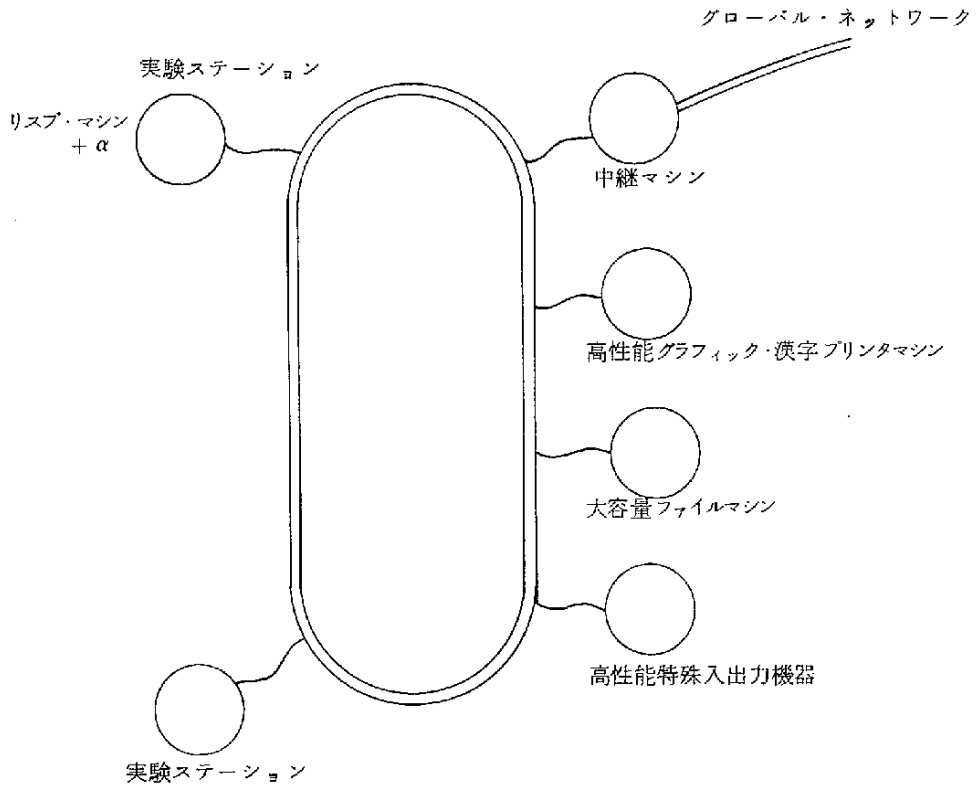
3.3 推論システム

汎用の推論システムは、知識ベースと推論機構とから成る問題解決機と考えられる。問題解決の難易は、対象とする問題の性質によって大きく異なる。難易を決定する要因としては、論理的なものに、知識体系の完全性の成否、不変性の成否があり、物理的には、扱うべき知識の量の問題がある。

3.3.1 データベース

データベースの問題は、論理的な構造が不変で、かつ知識は各時点で完全であるが、データ量が膨大な場合である。そのデータは、与えられた論理的構造を満足するモデルと考えられ、そのモデル自身は時間と共に変化する。この種の知識の扱いには、データベースの技術を生かしてデータを組織化することにより、検索、管理などが能率良く行える。論理的には、「閉じた世界の仮定」に基づく推論方法により、推論を関係代数上の演算に帰着できる。そのようなデータの扱いは、関係代数プロセッサによって、高速に実行できる。関係代数プロセッサの提案は、いくつかあるが、最も有望なものは、パイプライン・モードの並列性を有する、データフロー型のプロセッサ〔Tanaka,et al. 1980〕であろう。この面でのアプローチは、商用データベースの高度化にも直接つながり、知識情報プロセッサとしての能力は低いが、前期5年のターゲットとしては、恰好なものと思われる。関係データベースの欠点として、未知情報の扱いに難があること、階層構造の有するアクセス・パス方式の高効率性の達成が困難であること、関係がつねに一つの処理単位となり、並列同時アクセスの環境でのロックの範囲を限定できないことなどがあげられているが、それらの問題は、いずれも解決可能である〔古川、1979〕。





(b) ローカル・ネットワーク

図3－2　研究用ネットワーク

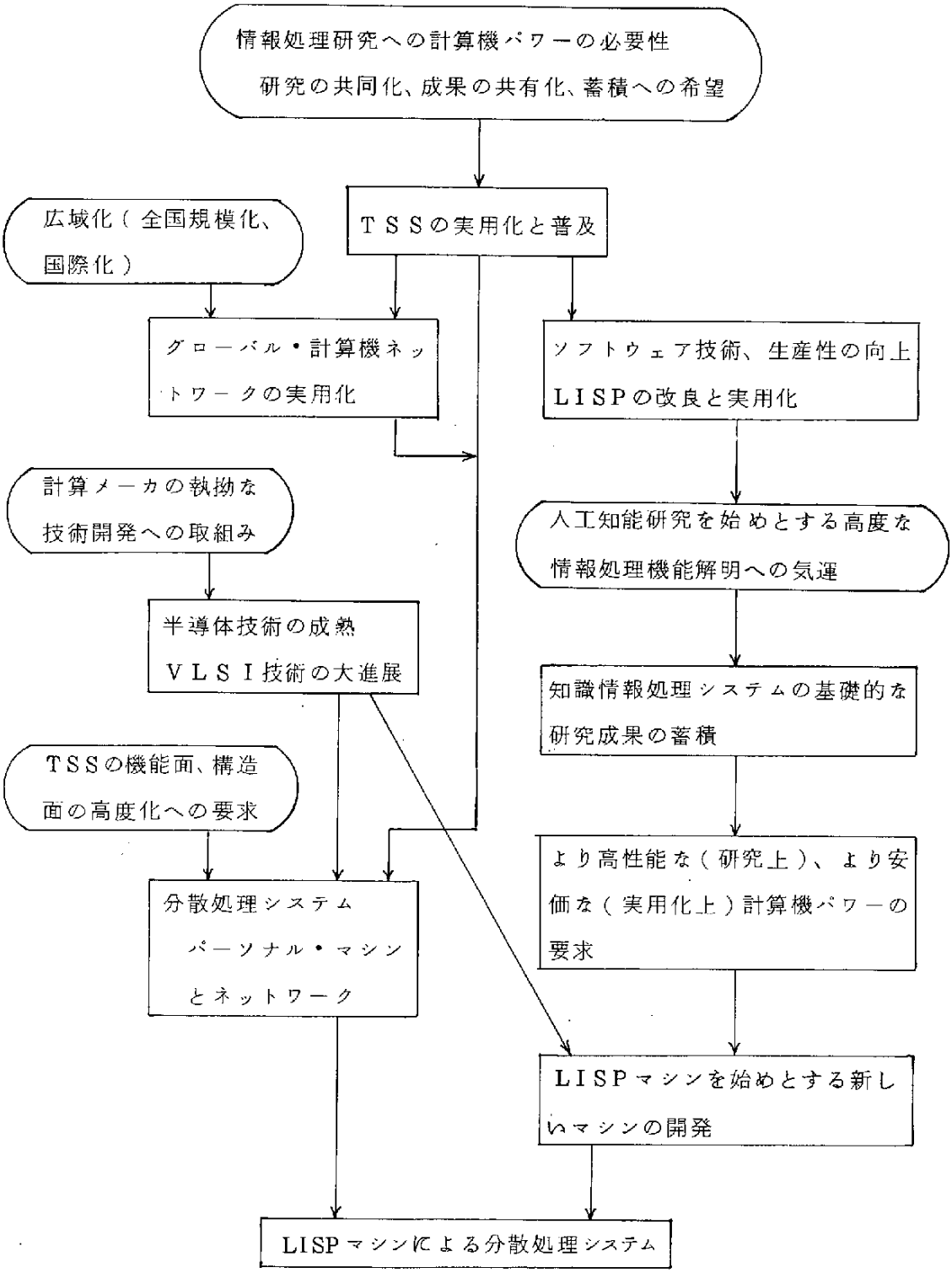

情報処理研究への計算機パワーの必要性
研究の共同化、成果の共有化、蓄積への希望
広域化（全国規模化、国際化）
ＴＳＳの実用化と普及
グローバル・計算機ネットワークの実用化
ソフトウェア技術、生産性の向上
ＬＩＳＰの改良と実用化
計算メーカの執拗な技術開発への取組み
人工知能研究を始めとする高度な情報処理機能解明への気運
半導体技術の成熟
ＶＬＳＩ技術の大進展
知識情報処理システムの基礎的な研究成果の蓄積
ＴＳＳの機能面、構造面の高度化への要求
より高性能な（研究上）、より安価な（実用化上）計算機パワーの要求
分散処理システム
パーソナル・マシンとネットワーク
ＬＩＳＰマシンを始めとする新しいマシンの開発
ＬＩＳＰマシンによる分散処理システム

図３－３　技術の流れ

表３－１　日米比較

	アメリカ	日本
ＴＳＳ	１０年前より、大学研究機関に広く普及し、標準的な研究設備として最大限に活用されている。会話型の利用技術が多数開発されている。	どうにかアメリカ並みの利用形態をとっているのは極くわずかである。会話型として自由に使えるのは、ミニコンの場合がほとんどである。これが、ソフトウェアの蓄積、普及を極端に悪くしている。
グローバル.ネットワーク	１０年前よりＡＲＰＡＮＥＴが研究機関に提供されている。各サイトに研究成果が蓄積されるにつれ、ここ数年本来の効用を発揮するにいたっている。無料であることも大きい。	今年あたりから、電々公社のパケット通信網が利用可能になる。
ＬＩＳＰ	各ＴＳＳには、必ず製作され、研究用の言語としては最も利用価値の高い言語として活用されている。製作技術の改良が続けられ、適用範囲を広げつつあり、会話型 プログラミングの最大の成果	本格的に利用できる所はごくわずかである。ようやくその必要性が認められ、普及への気運が生じつつある。この場合も、ＴＳＳの普及の遅れが大きな障害となっている。
ＶＬＳＩ　技術	大学や研究機関に、ＶＬＳＩ技術を駆使しうる研究者や技術者が育っている。またかなりの試作設備を持った所も現われ、ＶＬＳＩを有効に利用する技術の研究.開発を大きく展開しようとしている。	メーカにおける生産技術としては、アメリカを凌駕しつつある。しかし、大学や研究機関での利用技術への取組みは非常に遅れている。
ＬＩＳＰマシン	各所で研究．開発が行われ、数ケ所で利用者に提供できる段階になった。商用になるのは、もう少し時間を要する。	研究は、かなり活発である。しかし、本格的実用を目指したもの、利用の現場に結びついているものという点では大きく遅れている。
ローカル.ネットワーク	４、５年前、ＸｅｒｏｘでＥｔｈｅｒｎｅｔというほぼ決定版が実用化されて以来，各所で同様のものが開発され、一般的な利用形態として定着しつつある。	研究はかなり活発であった。しかし、利用の実状に合い、有効に利用されているという点では、遅れが目立つ。

3.3.2 述語論理

知識体系が不完全で、かつスタティックである場合は、通常の時間変数を含まない一階述語論理系の上で表現できる。一階述語論理系の演繹法として、resolution原理が良く知られている。resolution原理は、推論規則が非常に単純であり、よく研究されている。近年の成果としては、命題部分の扱いと変数部分の扱いを分離させた〔Sickel１９７５〕等の結合グラフ法が知られている。述語論理によるアプローチの問題点としては、知識の使い方に関する知識（メタ知識）を扱えない点があげられる。すなわち、そのような知識は、一般には推論過程自身を制御する情報を与えるものであり、そのような情報の利用によって、無駄な探索を避けることが可能となる。近年、一階述語論理の部分集合であるHorn論理を扱う言語ＰＲＯＬＯＧに関連して、メタ知識を組み込む試みが行われている〔Gallaire, et al. １９７９〕、〔Bundy, et al. １９７９〕。このような方向の研究は、述語論理アプローチの有効性を追求するものとして、価値がある。

ＰＲＯＬＯＧに関して言えば、それは、非手続き型のプログラミング言語であり、さらに、非決定的アルゴリズムの記述ならびに実行が可能である。すなわち、高度なプログラミング言語として、その有用性が期待される。ＰＲＯＬＯＧは、ＰＬＡＮＮＥＲとほとんど同じ機能を有している。それは、ＰＬＡＮＮＥＲの論理版と考えられる。ＰＲＯＬＯＧの利点は、その内部構造に基づく、自動後戻り制御の高速化である。それは、構造共有法と呼ばれている方法である。

一階述語論理とデータベース処理の融合問題は、〔Reiter, １９７７〕の形式化がよく整理されている。Reiterは、有限の対象上の論理操作を、等号の公理と領域限定公理（Domain Closure Axiom）を用いないで行う方法を導入した。存在限定子のみを含む質問の処理は、個々のデータに当って条件を満たすものを求めればよいので、上の２つの公理を必要としない。ところが、（∀ｘ）の形の全称限定子を含む質問は、可能なすべてのものの真偽を当たることが必要となり、

上記の２つの公理を使うことになる。Reiterの導入した方法は、関係代数の除算によって、（∀ｘ）の処理を代数的に行う方法である。さらに、Reiterは、データベース中の事実（公理）を用いた演繹過程を、その他の演繹過程から切り離し、一階述語論理の証明過程から、データベースに対する検索手続きを抽出することに成功した。この方式によって、「閉じた世界の仮定」が成り立たないような複雑な世界の表現が可能となり、データベースは機能的に一層高度化し得ることが明らかとなった。

3.3.3　推論根拠の管理

知識体系が不完全でかつダイナミックである場合、仮説に基づく推論と、新たな知識に基づく仮説の修正などの機能を実現することが必要となる。これは、図３－５のような、推論系と推論根拠管理系の２つの部分系から成る問題解決機によって実現できることが、〔Doyle，　１９７８〕等によった示された。推論根拠管理系は、推論系に対して、メタな系になっており、推論系の制御を行うのに利用できる。実際、Doyle は、彼のシステム（Truth Maintenance System, ＴＭＳ）を用いて、論理的に最も能率のよい後戻り制御方式（Dependency Directed Backtracking）を開発した。それは、矛盾が発見された時、自動後戻り制御のように、直前の選択点に戻って次の選択子を選ぶ方式をとらず、その矛盾を引き起こした原因となる選択点に直接戻ってしまう。また、推論系と推論根拠管理系が分離しているので、これまでに行った推論自身は、誤った部分を除いて、すべてそれ以後の推論で生かされる。

DoyleのＴＭＳは、日常的な推論で問題となる推論方式である Default Reasoning に対する論理的に整理された方法を与えている。すなわち、Default Reasoning によって、「鳥は一般に飛べる」ので「ペンギンも飛べる」と結論づけたときに、あとでその誤りに気がついたとき、その矛盾を検出して、何がいけなかったかを検知し、それを正す（「ペンギンは飛べる」は誤り）ことができる。

ＴＭＳは、その名前が示すように、データベースの更新時に、常にデータベース全体を矛盾のないように管理することにも利用できる。このことにより、知識の段階的学習が可能となる。

これまで述べてきたことから明らかなように、ＴＭＳを持った問題解決機は、３．１．１で与えた知識情報処理システムの機能イメージの実現にとって十分な機能を備えている。

ＴＭＳ自身、未だ完成されたシステムではなく、その理論も展開中である。この方向での研究の推進が期待されている。

3.4 データフロー・マシンとデータベース・マシン

高度な知識情報処理システムには、高機能データベース（知識ベース）と強力な推論マシンが、核となるハードウェアである。

データベースの操作が、計算記述のより大きな部分を担うことになる。大容量データベースの検索もさることながら、動的な小さなデータベースを作り操作するということが、高次の推論には必須になる。そこで基幹をなす、アーキテクチャ技術は、現在宣伝されている言葉を用いれば、データフロー・マシンとデータベース・マシンである。データフロー・マシンは、Dennisらの作業[注]を整理し、より一段と発展させたもの、データベース・マシンは、関係データベースにおける作業を土台にし、より一段と発展させたものというところが直感的なイメージである。高機能の推論機能を高速に実行するのが、データフロー・マシンの役割であり、大容量のデータを蓄え、検索・管理するのが、データベース・マシンの役割である。

具体的には、知識情報処理システムにおいては、知識ベースと推論マシンは、融合したものとして存在し、不可分である。むしろ、一つのものを、データの側

注) Dennis 等の現状を全面的に認めるという意味ではない。Hewitt の Actor 理論の方が、よりすぐれた性質を持っているし、Dennis 等も、その欠点を改善しつつある。
ここでは、データフロー・マシンを普通名詞として用いる。

面から見た場合が知識ベースで、プログラムの側面から見たのが推論マシンである。ここでは、とり合えず2つのものが存在するとして、それぞれに期待される機能とその機能が2つのアーキテクチャ技術にどのように関連しているかを以下に列挙する。

（1）推論マシン ⟷ データフロー・マシン

データベース・マシン

(a) 高度な記号処理機能

リスト処理を最も低次のものとして、各種の高度な記号処理をしなければならない。記号としては、集合、バグ、対、数式、論理式等々が想定されるが、いずれも、リスト処理技術をベースにして実現される。リスト処理は、LISPの上で、副作用なしに、高度なデータ構造の表現と操作を初めて、可能にし、実用化した。したがって、記号処理はデータフロー・マシン上で高能率に処理しうる。またリスト処理における記憶領域の動的な配分とゴミ集めの技術は、データフロー・マシンの実現のための基本的技術でもある。

(b) パターン照合、連想機能

論理式におけるUnification，引数受渡しの一般化としてのパターン照合、集合等における、等号や包含関係の判定をはじめとする各種記号データ構造の合成、分解等々。低次のものから高次のものまで、各種のパターン照合や連想機能が多用される。この機能は、データベース・マシンの基本機能であるとともに、データフロー・マシンの基本実現技術の一つともなっている　また、複雑な記号データ構造の照合等の操作に対しては、専用のデータフロー演算器が用意されることとなろう。

(c) 非決定的演算、大データ演算処理→並列処理

計算は、〝この内のどれか〟という非決定的な表現が多用される。適切なものを選択していく機能が推論機能ともいえる。もちろん低次のものから高次のものまで同意されるであろうが、利用者は、この推論機能を前提としてプログラムを書くようになる。また、多数のデータ要素を一度に、変換、写像するよ

うな、大データに対する演算子も多く用いられるようになる。現在の商用計算機（フォン・ノイマン型）に対しては、非決定的演算は通常は後戻り制御によって、大データ演算はくり返し制御によって実現されている。しかし、いずれも、単純なものを除いては、非常に効率の悪いものになっている。一方、微細な並列処理を効率良く行なうデータフロー・マシンに対しては、これらの演算子を並列制御によって実現することができる。ある意味では、これらの演算は、データフロー・マシンの出現によって、はじめて、効率の良い有用な演算機能となりうるということもできる。勿論すべての非形定的処理がすべて、並列処理によるという意味ではない。データベースのTruth Maintenance機能における、ルール群の適用の場合などの非決定的処理には、やはり後戻り制御が用いられるであろう。この場合の後戻り制御は、実現技術というより推論機能としての選択の結果とみる方がふさわしい。

(d) 対象指向（object oriented）

プログラムや知識表現におけるモジュラリティを高め、構造化を進める方法として、対象指向の記述形式が色々な方向から提案されてきた。プログラム言語における抽象データ型、仕様記述における抽象データ構造、並列処理記述におけるモニタと順路式、知識表現におけるフレームの考え方等である。この必要なものを一ケ所にまとめ、他とのインタフェースを明確にする記述法は、データフロー・マシンにおける、交信の局所性や他との交信法の明確化を保証し、データベース・マシンにおいては、参照や検索の局所性、他との関係の明確化の保証を与えてくれる有用な考え方である。また、副作用を考慮した対象指向の記述によって、データフロー・マシンに、基本的な計算機構をくずさずに、経過依存性を導入することができる。

（Ⅱ）知識ベース ⟷ データベース・マシン

データフロー・マシン

(a) データのプログラム化、プログラムのデータ化

データとプログラムの双対性は、受け継ぐべき重要な性質である。この考え

方は、データへの参照とプログラムの呼び出しの同一視に進む。 Micro-PLANNER に代表される人工知能向言語におけるパターンによる呼び出し機構としてまず具体化され、さらにＰＲＯＬＯＧに代表される論理プログラミング言語において、一段と整理された。すなわち、プログラム自身も述語論理式として、データと同一の形式にまとめられる。これは、データのプログラム（手続き）化であり、プログラム（手続）のデータ（宣言）化である。この機能は、知識ベースの最も基本となる土台である。知識ベースには事実としてのデータに加え、その事実を支配する一般的な記述（プログラム、ルール、定理）も蓄えられ、推論マシンの処理を受けることにより、高度な演繹を行なうことができる。プログラムのデータ化は、プログラムに関する知識の表現と処理を可能にし、より高階な知識ベースを作ることができる。データのプログラム化は、データをミクロな意味での対象指向としてとらえるもので、データベースへの参照をデータフロー概念に統一する重要な考え方である。

(b) 高度な連想機能

大容量のデータベースの検索のためには、高度なハッシング技術を駆使することになる。また並列検索の機構も必要となる。これらはすべて特殊目的のデータフロー・マシンとして設計される。

(c) 高度な代数演算，構造化

関係代数的な演算には、推論マシンの大データ演算と同等の機能が必要になる。また、大規模になるにつれ、蓄積されたデータ群の構造化が要求されるようになる。重要な関係（Ｉｓ－ａ，Ｐａｒｔ－ｏｆ等）を軸にした構造、フレーム的（対象指向的）な考え方の導入等々である。これにより、検索範囲の意味のある限定や、並列検索を高度化することができる。このためには、推論マシンの全機能を駆使することになるし、さらにこの考え方をすすめれば、データベース・マシンのデータフロー化に結びつく。例えば、意味ネットワークのハードウェア化は、その兆候の一つである。

3.5 知的プログラミング環境

3.5.1 はじめに

知的プログラミング環境の達成は、知識情報処理技術の追求にとって、2つの面で重要な課題である。第1は、知識型情報システムの開発のための道具として欠かすことができないものである。知識型情報システムは、大規模ソフトウェアとなり、その機能は高度で、複雑なものとなる。また、その開発は、漸進的に行われる。そのようなソフトウェア・システムを実現するためには、高度なソフトウェア開発支援システムを必要とする。このようなシステムを、ここでは知的プログラミング環境と名づける。

第2は、知的プログラミング環境自身が、知識型情報システムとして実現され、したがって、知識情報処理技術の応用システムと考えられる。

本節では、このような知的プログラミング環境を達成するためのステップを、従来のアプローチの延長上に位置づけられるものと、新たな方向を目指すものの2つに合けて論ずる。そして最後に、人間ー機械インタフェース機能をも考慮した、プログラム開発のための諸要素を統合したプログラミング開発支援システムのイメージを与える。

3.5.2 従来型アプローチ

ハードウェア環境としては、従来型計算機のネットワーク、A4版グラフィック・ディスプレイよりなる高機能パーソナル・コンピュータを想定する。これは、Xerox社のPARC（Palo Alto Research Center）で開発、使用されているAlto maclinc と、それによるコンピュータ・ネットワークEther Net と類似なものである。

ソフトウェア環境としては、基本的なものとして、グラフィック・ディスプレイ機能を十分に生かしたエディタがある。

つぎにソフトウェア開発方式について述べる。現在、ソフトウェアの開発方式は、ＩＢＭ始め、各計算機メーカが、必死になって追求している。それらの特徴は、つぎの通りである。

(1) ＴＳＳ base

端末入力をベースにした会話型システム。

(2) 構造化プログラミング

ＩＢＭのＨＩＰＯに見られる実用的なものから、Ｄijkstra によって提唱されている厳密な方法論まで、数多くのシステムが開発されている。

(3) システム記述言語

高水準のプログラミング言語を拡張して作られた言語で、ソフトウェア生産性の飛躍的向上に役立つ。

(4) 図式言語

フロー・チャートに見られる図式表現の長所と、構造化プログラミング法を組み合わせた方法で、構造化が図によって明示され、ソフトウェアの設計、ドキュメント、保守性にすぐれている。図を直接グラフィック端末から入力し、編集・管理するシステムの開発が、２～３年のうちに実現するであろう。

(5) データ抽象化と階層プログラム用言語

Pascal,Simulaの延長上の言語として、データ抽象化による階層プログラミングを行うための言語（ＣＬＵ，ＭＥＳＳＡ等）が開発されている。

(6) Friendly Interface

ＩＮＴＥＲ－ＬＩＳＰ に見られるような、ユーザとの対話機能を重視したシステムも開発されている。とくに、ＤＷＩＭ（Do What I Mean）は、入力のつづりのわずかな誤りの検出に始まり、省略記法、さらにはユーザの思い違いに基づくエラーを検出し、自動的に修正する機能をも有する。

(7) 要求仕様、仕様記述法

ソフトウェアの設計の段階の生産性を向上するための、いろいろな実際的な手法が開発されている。

これらのうち、とくに図式言語による構造化プログラミングと、データ抽象化に基づくプログラミング技法は、今後とも注目に値する技術である。とくにデータ抽象化技法は、要求仕様から始まって、プログラム設計、コード化、ハードウェア設計に至る全システム設計に共通に使える手法と考えられ、一層の研究と、システムの開発が期待される。データ抽象化技法は、各レベルに合った言語の設計と、その言語による問題の記述を可能にする技術である。また、あるレベルで、n種類の応用プログラムがあり、データ表現がm通りあるとすると、従来の方法ではn×m通りのすべての組合せについて、それぞれ別個のプログラムを必要としたが、データ抽象化に基づく階層プログラミングによれば、抽象データ型の演算によって記述されたn種類の応用プログラムと、その抽象データ型を実現するm通りの別個のプログラムを用意すればよい。これによって、システムの複雑度をnをn×mからn＋mに減少させることが可能となる。

3.5.3 新方式に基づくアプローチ

ソフトウェア技術の新しい考え方は、プログラムをより厳密に、数学的に扱うことを基本にしている。そのねらいは、ソフトウェアの高品質化、プログラム言語の高水準化、ソフトウェア生産性の飛躍的向上などである。このアプローチの核となる考えは、問題自身の非手続き的な記述である。これは、問題の仕様と呼ばれ、プログラムが問題解決の方法（Ｈｏｗ）を記述しているのに比して、解決すべき問題自身（What）の記述となっている。

仕様は人間が記述するものであるが、その仕様を満足するプログラムをどのようにして作るかによって、３つの異なるアプローチが存在する。その第１は、仕様、プログラム共人間が与えるアプローチで、そこでは、そのプログラムが果して本当に仕様を満足しているかどうかを判定する検証問題が中心的課題となる。この方式を検証方式と名付ける。

第２のアプローチは、与えられた仕様を満足するプログラムを自動的に作り出す方法で、自動プログラミングとして知られている。

第3のアプローチは、プログラム言語の水準を上げて仕様言語との同一化を図るアプローチである。すなわち、人間が記述した仕様を、計算機はプログラムとして実行する方式である。この方式を仕様実行方式と呼ぶことにする。

新しいソフトウェア技術の核となるアイデアに、データ抽象化法に基づく階層的プログラミング技法と、ルールに基づくプログラム変換技法があることは、すでに述べた。本項では、上記3つのアプローチの他に、上述の分類に属さない第4のアプローチとして、このルールに基づくプログラム変換方式を採り上げる。

(1) 検証方式

検証方式の詳細なサーベイは、ワーキング・グループの報告書「ソフトウェア工学」に譲るが、本報告書では、その知的プログラミング環境における役割を中心に論ずる。

検証方式の最大の目的は、正しいプログラムの生産技術を確立することである。これまでにも多くの努力が注がれてきたが、実用性を考えると必ずしも、その努力は実っていないのが現状である。それは、正しいプログラムを作る技術が、必ずしも正しくないかも知れないが、それでも結構役に立つプログラムの作成技術に比して著しく困難であったからである。この点を解決できる見込みのある技術が、データ抽象化による階層化法に他ならない。この方法は、それ自身、大規模ソフトウェア開発のための強力な指導原理を与えるのみならず、検証にとっても適していることが、多くの研究によって明らかになってきた。すなわち、プログラムを階層化することによって、検証問題を、階層内の検証問題と階層間の検証問題に分解することが可能となった。

検証技術自身には、2通りの代表的なアプローチがある。その第1は、Hoare Logicに基づく方法で、プログラムは、通常のALGOL型言語を対象としている。第2は、式の等価変換、あるいは記号レベルのプログラムの実行に基づく方法で、対象となるプログラムは、いわゆる副作用のない、関数形のプログラムである。

そして、抽象データ型を扱える言語も、いくつか設計され、検証自身の機械

化を目指しているシステムも、2、3報告されている。それらの試みは、いまだ実用に供せる程にはいたっていない。今後の課題の第1は、この新しい枠組での技術の蓄積である。原理的には、プログラムの生産性が飛躍的に向上すると予想されるが、新たな問題として、仕様記述が必ずしも容易でないことが分かってきた。すなわち、この方向を推進するための、現在最も必要としている技術は、仕様記述の方法論である。また、これに関連して、一般のプログラマへの普及が困難であることも問題である。これは、教育問題とも、密接につながっている。第2の課題は、理論的な問題である。現在、並行プログラムの検証問題は、未だ解決されていない。また、副作用を伴うプログラムの扱いも、完全ではない。第3の課題は、検証の機械化、高速化である。第4に、プログラミング言語の使い易さの追求がある。現在提案されているシステムは、いずれも構文規則が複雑で、プログラムが書きにくく、しかも、可読性も悪い。この問題は、将来は自然言語によるプログラミングが実現された暁には解決するであろうが、より実際的な解を探ることも必要であろう。

これらの研究を積み重ねることによって、知的プログラミング環境の実現に役立つ技術が確立されるであろう。さらに検証方式の研究開発は、(2)で述べる自動プログラミングの基礎を与える。なぜならば、プログラムは、検証から一歩進んだ、プログラムのConstructive な存在証明（ある仕様を満足するプログラムが存在することの証明）から自動的に組立てられるからである。

(2) 自動プログラミング方式

仕様からプログラムへの変換は、WhatからHowへの変換であり、その質的な差は大きい。その変換は、一般には、高度な知的思考活動を必要とし、その自動化は困難な課題である。その中でも、比較的簡単なアプローチは、すでに解法が知られている問題の解法を辞書化し、問題が与えられたときは、その辞書を参照して解法を知る方法で、このような辞書はアルゴリズム・バンクと呼ばれている。その場合でも、実際に必要とするプログラムと辞書中にあるアルゴリズムの記述には、大きな隔たりがある。さらに、すべての問題の解法を辞書

化するわけでなく、プログラムの部品の単位での辞書化が必要である。そのとき、各部品の結合の問題が生じてくる。これらの問題に対する適切な解を求めることが、この方法の成否を決定づけるであろう。

第2の、より困難な課題は、解法自身を発見することである。これは、人工知能の分野で、問題解決の1つとして研究されてきた領域である。問題が単純で、基本演算を高々十数個組み合わせればプログラムが作れる場合には、すべての組合せを調べるような強引な方法によっても解を得ることが可能である。また、通常は、発見的な方法によって、場合の数を大巾に減らすことも可能である。しかしながら、問題が多少複雑になると、そのような方法では手に負えない。問題の複雑さを本質的に減少させる工夫が必要である。その基本的な考えは、問題をより単純ないくつかの問題（サブ・ゴール）に還元する方法で、問題解決の分野ではProblem Reduction Methodとして知られている。先に述べたデータ抽象化による階層化は、このProblem Reductionを行う一つの方法であると考えられる。ここで、2つのアプローチが考えられる。その1つは、抽象データ型を人間が与えてやるアプローチである。問題解決を人間と機械の協調により行うという考え方に立てば、この方式は有望である。他のアプローチは、サブ・ゴールの設定自身を機械に行わせる方法である。このような流れを追求しているいくつかの研究もみられる。たとえば、〔間野,1979〕のデータフロー解析に基づくプログラム・スキーマ（プログラムの大まかな構造）の選定によるProblem Reductionなどである。

自動プログラミング方式の問題点の最大のものは、検証方式の第1の問題点と同様、仕様記述自身の困難性である。アルゴリズム・バンク方式の問題点は、出来たプログラムの質が保証されない点と、プログラムの理解が困難な点である。プログラムの質を保証するには、効率に関する仕様も考えなければならないが、現状ではそこまで行っていない。解法生成方式の問題点は、解法が未知の問題を解くのに、ほんとうに有効かどうかが不明である点である。さらに、自動プログラミング方式が、(3)および(4)で述べる仕様実行方式あるいは

プログラム変換方式に比して、どれほど有効であるかも、十分に吟味してみる必要がある。

(3) 仕様実行方式

プログラミング言語自身を高水準化することによって、仕様言語の水準にまで持ち上げるのが本方式である。現在、２つの流れがある。第１の流れは、非決定性手続きの直接実行を行う言語で、ＰＬＡＮＮＥＲ，ＰＲＯＬＯＧなどがこれに属する。その基本的な考えは、問題解決の基本的機構である Problem Reduction の自然な記述を可能にすることである。サブ・ゴールの設定がいくつか考えられるとき、非決定的な計算機構により、適当なサブ・ゴールの選択と、それが誤りであったときの制御が自動的になされ、ユーザのプログラムでそれらの制御を行う必要がない。この方式の利点は、検証方式で必要であった仕様とプログラム間の検証が不要な点である。また、計算と検索が同一表現で行われるという長所もある。欠点は、問題解決過程がシステムによって制御されているので、きめの細かい能率のよいプログラムを作れない点である。たとえば、サブ・ゴールの選択を誤ったときには、自動的に後戻り制御が行われるが、その戻り方は常に最近の選択点とするのが通常である。（この点に関しては、最近、Truth Maintenance Systemの研究から、もっと効率良い方法が提案されている） 制御構造の問題は、本質的には並行プログラムの枠組の中で論ずるべきであり、そこでの理論の整備が望まれる。

第２の流れは、関数型プログラミング、あるいは、代数に基づくものである。この方法は、抽象データ型による階層プログラミングと密接に結びついている。上位の階層の関数は、下位の階層の関数の組合せによって表わされる。この関係は、数学的には等式であるが、手続きの実行時には、上位の表現が下位の表現によって置きかえられて実行される。これは、３．１で述べたルールに基づくプログラム変換技法に当たる。また、プログラムの変換は、ＰＲＯＬＯＧやＰＬＡＮＮＥＲでのサブ・ゴールの設定に相当している。その相違は、代数流のプログラムの変換は等価変換であるが、サブ・ゴールの設定は、必ずしも等

価変換でなく、論理の用語で言えば、含意（ＡならばＢ）を基にしている。すなわち、サブ・ゴールが満たされれば必ずゴールが満たされるが、その逆は成り立たない。言い換えれば、ゴールを満たすものは、他にもあるかも知れない。この事実は、ＰＲＯＬＯＧ等の有する非決定性に関連する。

仕様実行方式の最大の問題点は、言語としての表現力が弱い点である。通常のALGOL型の言語が持っている固定した記憶場所を示す変数の概念がないので、割り当て文が書けない。本質的には、共通の記憶場所の書き替えを利用したアルゴリズム、すなわち副作用を必要とするアルゴリズムの記述ができない。その典形的な例は、並行して走る複数のプログラムが、同一のファイルをそれぞれ独立に読み書きするときの、制御の問題である。

Hoare は、このような共有資源を、抽象データ型と同様の形式で扱うためのMonitor と呼ばれる機構を導入した。この形式化は、代数プログラム系に、副作用操作機能を埋め込む手がかりとなるかも知れない。

仕様実行方式は、プログラムの可読性が良く、漸進的なプログラムの開発も容易である。プログラムは、断片的な論理命題あるいは等式の集合として実現されるので、その機能の拡張あるいは修正は、新たな論理命題あるいは等式の追加あるいは更新によって行われる。このような長所は、知的プログラミング環境の実現に大きく寄与すると思われ、この方式での汎用性の追求が強く望まれる。

(4) プログラム変換技法

ルールに基づくプログラムの変換は、これまで述べてきたように、代数的システムでの検証、実行などを行う基本的な機構である。ここでは、この方法の最適化への応用を中心に述べる。

等価変換規則を用いて、プログラムをより効率の良い、しかも等価なプログラムに変換することが可能である。これは、元のより効率の悪いプログラムを仕様と見立てれば、自動プログラミングの一種とも考えられる。再帰プログラムの繰り返しプログラムへの変換、関係代数プログラムの階層データベース・

プログラムへの変換などがその例である。

この種の変換は、自動プログラミングの問題としては、そのギャップが少さいので、比較的簡単な問題に属し、しかも効率の悪い関数型プログラムを、より効率の良いALGOL型プログラムに変換できるので、その実用的価値は大きい。

3.5.4 プログラム開発支援システム

知的プログラミング環境は、これまで述べた諸機能を合わせ持つようなプログラム開発支援システムを作り出すことによって、達成される。そのような支援システムの重要な機能のうちで、これまでに触れていなかった側面に人間－機械インタフェースがある。仕様言語は、抽象度は高いが、依然として形式言語である。知識情報処理システムとして、プログラム開発支援システムを実現する場合、自然言語や手書き図形などを仕様言語の代わりに使えることが望ましい。

さらに、プログラムの理解や虫取り（プログラムの誤りの除去）などを計算機支援によって実現することもシステムとしての有用性を高めるのに役立つであろう。

図３－４に、システム全体の概念図を示す。

3.6 知的ハードウェア試作環境

ＶＬＳＩ技術が、情報処理研究、技術全般に大きな影響を及ぼしつつあり、今後一層大きくなるであろうとは、誰もが予感している。しかし、ＶＬＳＩそのものの開発は軌道に乗ったものの、ＶＬＳＩを何にどう使うか、特に使いこなすための技術の研究・開発は極端に遅れている。特に日本では、使いこなす技術は、研究の場には、遠く手の届かぬものになっている。しかも、この状況は程度の差こそあれ、ハードウェア技術全般にわたって言える。約２０年間にわたって多くの改善への独立が投じられたソフトウェアに対する作成環境の整備状況に比べ極端に遅れている。研究者が、自由に大規模なハードウェアを開発し、さらにＶＬ

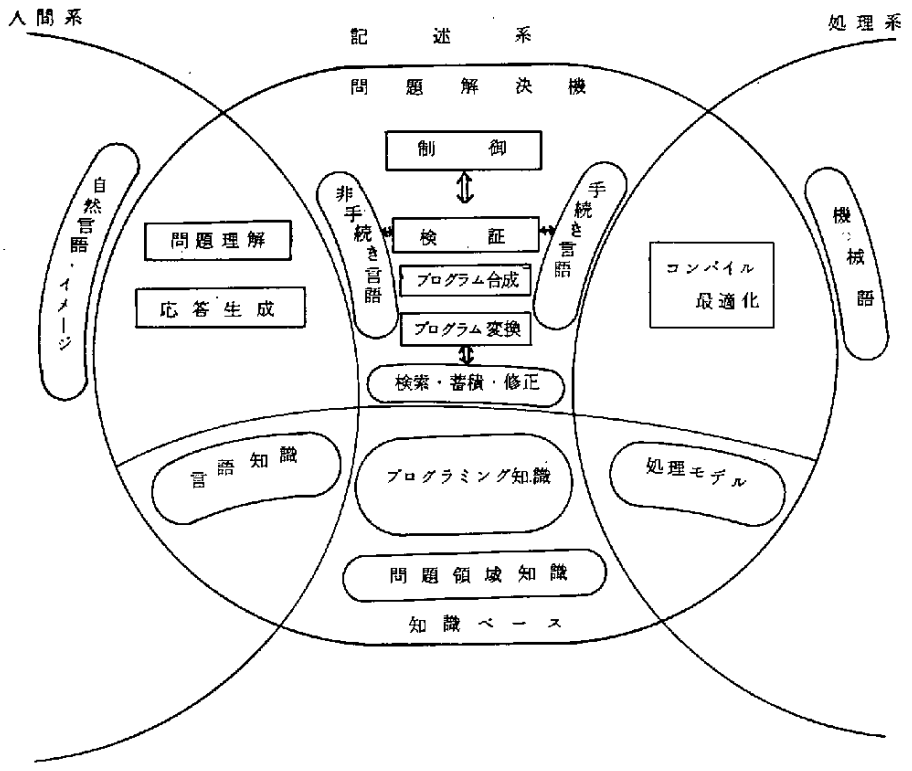


図３－４　知的プログラミング環境の概念図

ＳＩ技術を自由に使いこなせるようにするため、試作環境の整備を行なう。

例えば、図３－５に示すような既存技術の組合せで可能な試作環境は早急に整備すべきである。このような設備を使い、ＶＬＳＩ技術への大筋をつかみとりながら、本格的なＶＬＳＩマスク設計支援システムの研究開発を進める。これ自身が大きな研究テーマであり、知識情報処理システムの一つの応用となる。具体的には、次のような研究テーマになる。

(1)　チップ記述言語とプログラム技法、階層的設計技法の研究・開発

(2)　シミュレーション、デバッグ技法の研究・開発

(3)　チップのテスト技法の研究・開発

(4)　ＶＬＳＩ向アルゴリズム論の研究

(5)　知的ＣＡＤシステムの開発

自動レイアウト機能、前例との比較・学習機能、支援機能、機能的等価性

の立証機能、テスト及び正当性証明機能、デザイン・ルールのチェック機能、設計変更に対するチェック機能。

自動合成機能。

その他、ソフトウェアの開発技法がすべて適用できる。

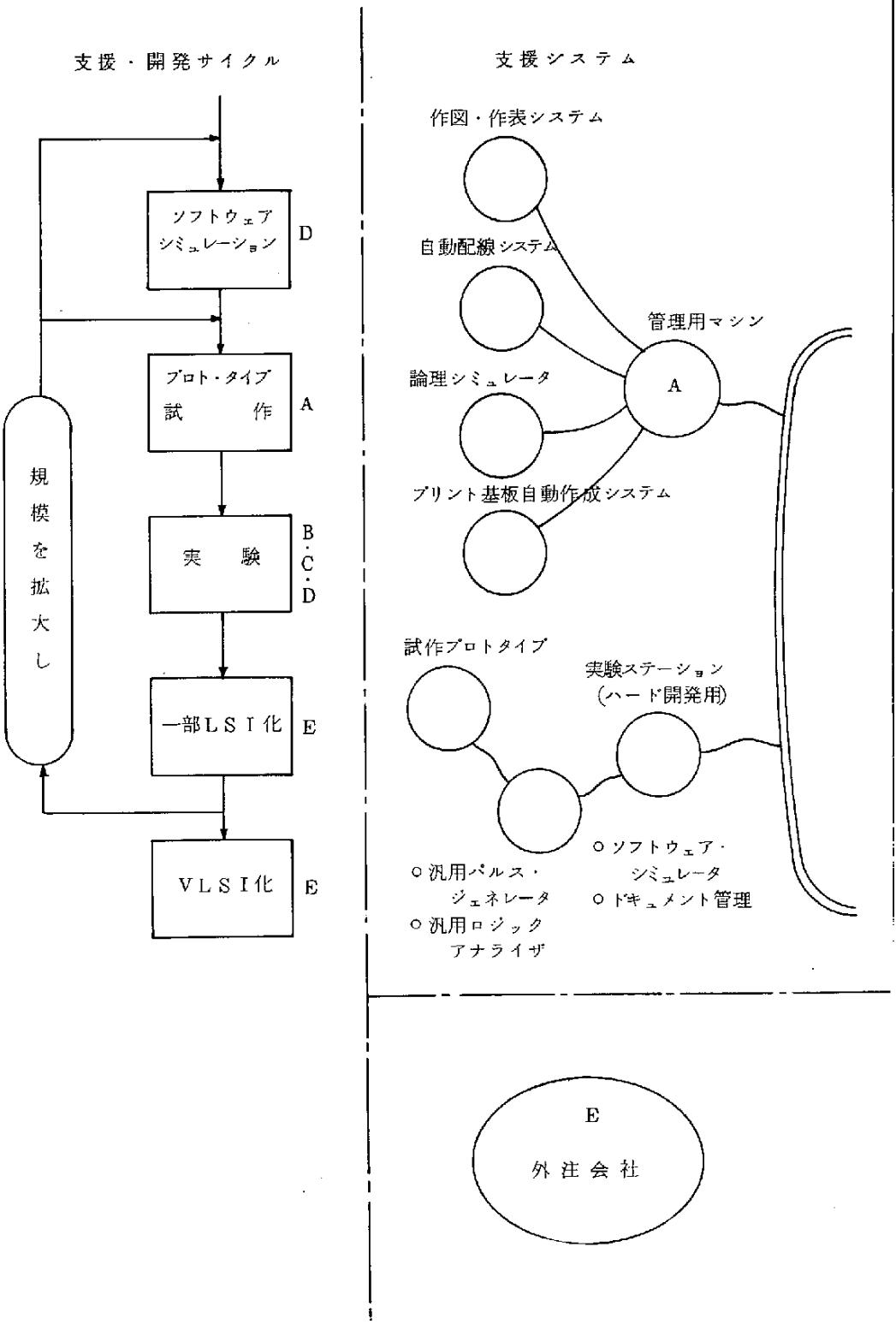


図３－５　大規模ハードウェア研究・開発支援システム

参考文献 （第3章）

〔Tanaka, et al. 1980〕 Tanaka, Y., et al.
〝Pipeline Searching and Sorting Modules as Components of a Data Flow Database Computer〞, working paper, 1980.

〔古川, 1979〕古川, 〝データベースの知的アクセスに関する研究〞, 電総研研究報告 №804, 1979.

〔Gallaire, et al. 1979〕 Gallaire, H. et al. 〝Controlling Knowledge Deduction in a Declarative Approach〞 Sixth IJCAI, 1979.

〔Bundy, et al., 1979〕 Bundy, A. et al. 〝Solving Mechanics Problems using Meta-level Inference〞, Sixth IJCAI, 1979.

〔Reiter, 1977〕 Reiter, R. 〝An Approach to Deductive Question-Answering〞, BBN Report №3649, 1977.

〔Doyle, 1978〕 Doyle, J. 〝Truth Maintenance Systems for Problem Solving〞, MIT AI-TR-419, 1978.

〔間野, 1979〕 〝データフローの概念に基づくプログラム合成方式〞, 信学研技報 AL 79-70, 1979.

4．各　　論

4. 各　　論

本章では、基礎理論研究分科会の下に作られた4つのワーキング・グループ(WG)、すなわち、自然言語処理WG、知識情報処理WG、マシンの理論WGおよび情報基礎論WGのまとめた調査報告、ならびに調査委託された、仕様記述、並列プロセスの理論（早稲田大学理工学部　広瀬研究室委託）、および機械翻訳（京都大学工学部　長尾研究室委託）に関する調査報告の概要を示す。

4.1　自然言語処理

4.1.1　はじめに

計算機の高度化には、i）計算機速度の向上　ii）知的機能の向上 が考えられるだろう。計算機のもつ知的レベルを向上させることは、究極的には計算機のもつ知的機能を人間並みにしたいということであろう。自然言語を計算機で理解するという試みは、特に後者の ii）に関係している。

現在の計算機の理解できる言語は、マイクロ命令にはじまり、機械語、アセンブリ言語を経て高級プログラミング言語に至るまで、さまざまなレベルがある。しかし、これらは全て人工言語とよばれる範疇の言語である。これに対し我々人間は、日常生活の場において、自然言語、音声、イメージ等を媒介にして相互の意志疎通を行っている。ところがこの、自然言語や音声、イメージ等の理解は、現在の計算機が最も不得意とするものの一つである。

しかし、計算機にこれらを理解させようとする（ある意味で野心的ともいえる）試みは、１９７０年代に入って、解決に向けて一定の前進がみられた、というのも事実である。このような自然言語と計算機をめぐる研究開発の輪は、計算機科学者のみならず、言語学者、心理学者、哲学者、論理学者をも巻き込み、拡大の一途をたどり、学際的な研究交流の場さえ生れようとしている。これは、そ

もそも自然言語そのものに内在する特質から生れた当然の帰結であるとみなされようが、極めて健全な動きであるといえる。

本ワーキング・グループでは、大きく二つにテーマを分けて調査を行った。一つは、計算機によって自然言語処理をするための技術の現状がどのようなものであるかを知るための調査である。もう一つは、現代の言語学の潮流を探るための調査である。これには、電総研の外部から、山梨正明　大阪大学講師、寺津典子富山大学講師の協力をあおいだ。本報告書の資料としての自然言語処理ワーキング・グループの報告書は、各担当者が実際に作成した原稿のおよそ5分の1にあたる要約である。完全な原稿は、いずれ機会をみて、印刷公表したいと考えている。

4.1.2　第Ⅰ部　自然言語処理システム

ワーキング・グループ報告書の第Ⅰ部は、自然言語処理システムの研究の現状の調査と分析で、8章から構成されている。しかし、本調査が2年にわたることを想定して、第3章と第4章は、今回の報告書に含めなかった。また、調査を進める段階で、本年の報告書では、ある特定のテーマの重要性が認識され、それを集中的に調査することにした。第Ⅰ部の構成と担当者は次のとうり。

第1章　はじめに（田中穂積・電総研）
第2章　構文解析（佐藤泰介・電総研）
第3章　知識表現
第4章　文の生成
第5章　問題解決と推論 ― Default Reasoning
（松本裕治・電総研）
第6章　談話理解（内田ユリ子・電総研）
第7章　音声理解（板橋秀一，三国一郎・電総研）
第8章　応用システム
8．1　機械翻訳（池田尚志・電総研）
8．2　質問応答システム（田中穂積・電総研）

第２章では、構文解析技術を述べる。自然言語処理技術のなかで、形態素処理、構文解析については、ほぼ問題点が明らかになり、しかも技術的な進歩がみられる。問題点の多くは、構文解析が、意味処理や、文脈レベルの処理と完全に分離できないことに起因していると思われる。それらを除けば、技術的に確立されてきていると考えてよいだろう。構文解析用のアルゴリズムとシステムについては、人工言語（形式言語）の構文解析手法を自然言語に応用するために、さまざまな拡張がなされている。これは、文脈自由文法を骨子として、各規則に何らかの手続き（プログラム）を付加し強化することによって、柔軟性をもたせ、文脈依存的な処理を行なわせようとするものである。我が国での研究状況と日本語の特質が、構文解析にどのような影響を及ぼすかについても調査した。

第５章では、最近、不完全な知識の下で行われる推論の研究が進んでいるため、それに焦点を当てて調査した。これはdefault reasoning とよばれ、質問応答システムの問題解決を行う場面で、ますます重要になってきている技術である。調査によればこれは、(i)変数割当てにおけるdefault、(ii)閉じた世界におけるdefault、(iii)階層表現におけるdefault、(iv)Frame問題、などに分類整理される。(i)は変数に値が陽に割当てられていない時、変数にあらかじめセットされている常識値を用いる推論である。(ii)の閉じた世界では、ある事実の成立が証明不可能な時には、その事実の否定が成立するというものである。(iii)は、階層構造における同レベルの概念間にdisjoint性が成立するというdefault reasoningである。(iv)は、箱の移動により、箱の位置は変化するが、箱の色は変化しないこと等をどのように扱うかという問題である。以上をどのようにして形式化するかが、default reasoning の最近の研究により、部分的に明らかになってきている。これらをも説明する。

第６章の談話理解では、主として、Halliday等の最近の研究を調査することにした。今回は、Hallidayの用語の説明とcohesion（結合関係）に関連する基本的な考え方を説明する。談話理解の問題は、今後増々重要になると思われるが、明確な理論とモデルが模索されている段階にある。しかも語用論に属す事柄

が極めて多いのが特徴である。現実の言語現象に幅広くあたり、分析する必要があると思われる。

第７章の音声理解の項では、特に７０年代の米国における音声理解プロジェクトの結果を中心に調査した。７０年前半の自然言語理解システムの研究の成果は、音声認識の研究者に希望を与えたように思われる。音声認識に言語情報が必要であるとの認識は基本的には正しいといえるが、米国における音声理解プロジェクトの結果は、言語情報の利用が十分有効であるという段階にまだないということを改めて多くの研究者に認識させる結果になった。そして、むしろ、もう一度音響、音声学的レベルの処理を見直す方向が示唆されていることが明らかにされた。

第８章では、自然言語の応用システムについての調査を行った。機械翻訳システム開発の歴史と現在の研究動向が概観される。質問応答システムと同様に、対象分野を限定することにより、実用化数歩のシステムがすでに実現可能な時期にきている。８０年代は、そうした方向の研究が、進展すると思われる。しかし、機械翻訳に関していえば、かな漢字変換と同様に、当面、人の介入による post editing をどのようにスマートに行うかが、短期的なレベルでは重要になると思われる。

4.1.3　第Ⅱ部　現代言語学の潮流

第Ⅱ部では、現代言語学の潮流を調査した。構成は、つぎのようである。

第１章　生成文法理論の展開（寺津典子・富山大）

第２章　生成文法理論における意味論の展開（山梨正明・大阪大）

第３章　ヨーロッパにおける言語学の展開（横山晶一・電総研）

第４章　モンタギュー文法（大谷木重夫・電総研）

第１章では、チョムスキー文法が、初期の標準理論から拡大標準理論を経て最近の痕跡理論の導入に伴なう修正拡大標準理論へ進展していく過程を追い、チョムスキー文法が、どのように理論的に進化したかを調査した。これらの理論的な進化が、どのような理由によりなされたかを歴史に沿って説明した。特に修正拡

大標準理論の基本的な考え方を詳しく説明し、痕跡が導入された意義と動機をまとめている。修正拡大標準理論の大略は下図のようである。

（意味解釈規則）

(1)文文法 ： $\overset{B}{\rightarrow}$ 基底構造 $\overset{T}{\rightarrow}$ 表層構造 $\overset{SI-1}{\rightarrow}$ 論理形式

意味解釈規則（ＳＩ－２）
他の認知体系
} ： 論理形式 → ‘意味’

第２章では、生成文法理論における意味論の展開を調査した。第１章は、主として統語論が中心であった。生成文法理論は１９７０年に入って、意味が重視され、それは、解釈意味論、生成意味論、格文法理論等の諸派に分岐してきている。チョムスキー文法の進展も、統語的側面のみならず、‘意味のとらえ方の進展’という視点でみなおすことができる。

第１章でも触れた解釈意味論に対置する形で生成意味論が生まれた動機、理由についても調査した。解釈意味論では、意味解釈規則の入力となる深層レベルを次第に表層レベルに近づける。生成意味論では、深層レベルを一層深化させ、意味表示レベルと同一視して行く方向をとる。両者の利害得失は一概にはいえないが、後者は、最近では語用論、発話行為論とも接点をもちつつある。いずれにしても、両者の意味論に、従来軽視されてきた語用論の問題をどう取り込むかが今後の課題であろう。

第３章では、ヨーロッパにおける言語学の展開を調査した。ともすれば、日本では、チョムスキー文法の隆盛の陰に隠れて、ヨーロッパの言語学の現状を知る機会が少ない。しかし、テニエールらの結合価の理論は、動詞を出発点とした文の構造分析として、注目すべきものであろう。またプラーグ学派の先導的な役割も無視することができないだろう。今回は、その概略を説明する。

第４章では、最近注目をあびているモンタギュー文法について調査した。モンタギュー文法は、統語論と意味論を融合する道を与え、入力文を極めて体系的な方法で内包論理式に翻訳する枠組みを与えた。内包論理は、様相論理の一種であるが、モンタギューは、trivial でない程度の英語文について、これまで言語

哲学の分野で問題とされていた、Frege の関数原則、すなわち全体の意味は部分の意味から計算可能であるという原則が整合性をもって成立する方式を提案した。今回の調査では、その概要を報告するにとどめたが、このモンタギュー文法の枠組みは、生成意味論の学者により注目され、それとの等価性が議論されるに及び、非常に活発に研究が進められている。また、計算機科学者の側でも、モンタギュー文法の考え方を取り入れたシステム作りが試みられるなど、今後の研究課題として、実り豊かな土壌を与えてくれたと評価されている。

以上で、自然言語処理ワーキング・グループ報告の大略説明を終えるが、自然言語処理の研究は長期のものであることを認識するとともに、第5世代の計算機に対しても少なからぬインパクトを与えうる可能性をもっている、ということを指摘したい。したがって、全体として難問解決への努力をおこたってはならない。しかし、それとともに研究開発の輪が、我が国でもっと広がることを期待したい。そうした輪の拡がりの中で、実用化を目指す試みも行われなければならないと思われる。

4.2 知識ベース・システム

知識ベースは、人工知能の応用システムを開発していく過程で生れた概念で、データベースと同様に情報をためておく基地であるが、その情報の質は、単なる個別的事実の集合にとどまらず、一般的な知識（概念）や、知識の利用に関する知識などを含み、人間の有する知識に近いものである。

情報システムを、人間と計算機の2つの成分から成るシステムと考えるとき、計算機サブシステムの機能をより高度化することが、情報科学の1つの大きな目的となっている。知識ベースを有する計算機サブシステムを考えると、それは、実世界のモデルの表現が可能で、実世界のモデルを含まない従来の計算機サブシステムからの大きな飛躍が期待される。実世界のモデルを有する計算機サブシステムの特長を列挙すると、つぎのようである。

1. 処理プログラムと人間サブシステム間のインタフェースとして、実世界の

モデルが利用でき、人間－機械インタフェースが改善される。

2. 柔軟な情報システムが実現される。データベースにおけるデータ独立性、プログラミングにおける機械独立性、移殖性などが容易に実現できる。

3. モデルに照らした処理プログラムの検証、プログラムの改良、合成などが可能となる。

4. 仕様のわずかな変更に伴う処理プログラムの変更が容易になる。

5. プログラムの蓄積性がよくなり、大規模なソフトウェア・システムの漸進的な開発が可能となる。

このような計算機サブシステムを作成する際に最も大きな問題となるのは、個々の問題の処理プログラムの記述ではなく、ある問題領域での知識、すなわち実世界のモデルの記述である。これまでに、各種の知識表現システムが開発されてきたが、それらは、その能力に一長一短があり、それらの利用法も確立されているわけではない。

知識ベース・システムのサーベイでは、はじめに、これまでに開発されてきた各種の知識表現システムについて概観している。それらは、セマンティック・ネットワーク、プロダクション・システム、ＫＲＬなどである。つぎに、システムの漸進的な開発にとって必須となる新しい知識の獲得と、その既存の知識体系への組み込み、すなわち機能的推論および学習の問題について述べる。学習系を含む閉じていない知識ベースは、不完全な知識の扱いを含む。とくに、そこでは、単調でない推論が問題となる。それは、常識による推論のもつ１つの側面であるが、人間が通常行っているように、不完全な知識でも、システムは常識的に考えて妥当な答を出すことができる場合はよくある。たとえば、通常、「鳥は飛べる」と考えついて大きな間違いはないが、だ鳥やペンギンは飛べないので、「鳥は飛べる」というのは正しくない。データベースに、例外的な場合を全部列挙しておけば知識は完全となるが一般には、常識的に正しい事実と２、３の例外を並べるだけである。このように、知識が不完全な状態で、「ペンギンは飛べるか」と聞かれると、もし、「ペンギンは飛べない」ことが例外事項としてデータベースに

記載されていなければ、「ペンギンは飛べる」という結論を、とり合えず出しておく。ここで大切な点は、常識によるこのような推論が誤りであると気がついた時、その誤りを適確に訂正できる機能である。最近のこの分野での大きな成果である、このような機能を有するTruth Maintenance Systemについての概説が、つぎに述べられている。このシステムは高度な知識ベースを実現する際に最も重要な機能の1つである仮説の設定と、それに基づく推論のメカニズムを容易に実現できる点が注目される。

知識ベースを含む計算機サブシステムは、知識工学と呼ばれる分野で、多くの実験システムの開発がなされ、そこで開発の方法論も合わせて研究が進められている。それらのシステムの特徴は高度な専門家の知識を計算機サブシステムの中で知識ベースとして実現している点である。現在、医療、科学、工学などの分野で、各種の応用システム開発が試みられている。それらについての詳細は、各論にゆずるが、ここでは、計算機科学の分野での応用が、これから一層大切になることを指摘したい。そのうち、データベースの高度化について、とくに詳しく述べられている。その他に、プログラミングの分野での応用については、本報告書3．5節およびソフトウェア工学についてのワーキング・グループの報告書を参照されたい。また、今後は、超ＬＳＩの利用技術の開発が、これからの大きな研究課題となることが予想されるが、そこでのＣＡＤシステムも知識ベース・システムとして実現されるであろう。さらに機械翻訳システムも同様である。要するに、今後開発が期待される大規模システムのほとんどのものは、それが高度な機能を必要とすればするほど、知識ベース・システムとして実現される可能性が高い。

知識ベースとデータベースは、全く異なる分野で研究され、それぞれ発展してきたが、それらの情報システムにおける役割は類似しており、同一の理論の上で論じられることが必要である。関係データベースは、述語論理の枠組で形式化されており、それを基にして統合が可能である。データベースの研究の成果は、大量データの組織化法と、そこからのデータの検索法で、これを知識ベースに取り

込むことが必要となってくる。本WGの報告書では、データベースの高度化について、このような視点からも論じている。

最後に、知識ベースあるいは高度なデータベースのためのハードウェアについての考察を行っている。知識ベースにおける主要な処理は問題解決に伴う演繹操作であるが、その基本機能は、記号のパターン照合を基にした記号処理、および非決定手続き処理である。さらに、大量データベースに対する高速の検索処理も重要な機能である。前者には、ＬＩＳＰマシン、ＰＲＯＬＯＧマシンなどがあり、後者には、データベース・マシンがある。前者については、マシンの理論のWG報告書に、そのサーベイがある。後者については、本ワーキング・グループの報告書で、とくに関係代数演算を直接実行するマシンについて述べられている。

4.3　ソフトウェア工学

ハードウェアの進歩に比べて、ソフトウェア開発技術が大きく遅れをとっていることは周知の事実である。このことが、今後、超ＬＳＩの出現を迎えるにあたり、一層深刻な問題となることは、目に見えている。この大きなボトル・ネックを取り除くことが、今後の情報処理研究の最大の課題であることは、論を待たない。現在のソフトウェア技術のかかえている最も大きな問題は、ソフトウェアのハード化である。すなわち、ＯＳ、言語処理プロセッサ、データベースのような大規模ソフトウェアは、その維持、管理が大変なうえ、仕様の些細な変更に伴うシステムの改訂や、虫取り、あるいは他の計算機システムへの移行などが非常に困難である。この問題は、データベースのように、情報の蓄積が本質的であるシステムにとっては、まさに死命を制する問題である。

この問題を解決するには、つぎの２つの面からの追求が必要である。

(1)　システムを、必要以上に複雑にしないための技術。

(2)　より複雑な問題を解決し得るソフトウェア技術。

この２つの要求は、一見矛盾しているように見えるが、(1)は、問題が有している複雑さ以上にシステムが複雑になることを防ぐことを、(2)は、現在のソフトウ

ェア・システムによって実現し得る機能を凌ぐ、より高度な機能が実現されるべきことを、各々要求している。この2つの目標を達成するための、新しいソフトウェア工学の追求が必要である。その中心的な役割を果す可能性のある技術の第1は、データ抽象化に基づく階層プログラミング技法であり、第2は、ルールに基づくプログラムの変換技法である。この第2の方法は、記号レベルでのプログラムの実行方式（記号実行）と言い換えることもできる。

これらの2つの技術は、知識ベースを基にした知識情報処理システムにより、容易に実現される。その場合、ここでの知識ベースを構成する知識は問題領域に関する知識およびプログラミング言語に関する知識である。問題領域に関する知識は、実世界のモデルとなっており、プログラミング言語に関する知識は処理のモデルを与えていると言える。このようなシステムでは、ソフトウェアの開発を、それらのモデルの開発の問題に置き替えることが可能である。それらのモデルの記述言語は、人間系に近い程非手続き性が要求され、処理系に近い程、手続き性が要求される。この2つの側面を満足する言語として、関数型言語、論理プログラミング言語などが、最近話題になっている。関数型言語は、表現能力には劣るが、モデルの記述に適し、モジュラリティも良い。このため、複雑なプログラムを、各モジュールの単純な結合によって、容易に開発し得る。先に述べたデータ抽象化に基づく階層プログラミング法を採り入れることにより、プログラムの構造が非常に単純になり、開発が一層容易になる。このアプローチでの最大の問題は、関数型あるいは論理型言語が、履歴に依存する計算の記述に向いていない点である。この問題を避けるには、プログラミング言語を表現力豊かな、ALGOL流の言語にすることが必要である。その場合に、そのプログラムを、人間が書く方式と、システムが作り出す方式とが考えられる。前者のアプローチでは、仕様とプログラム間の検証問題が重要となり、多くの方法が開発されている。これらの各方法については、ソフトウェア工学のワーキング・グループの報告書で詳しく述べられている。後者の、自動プログラミング方式は、前者に比べて、より一層困難な課題である。その点については、本報告書の3．2節、3．5節を参照

されたい。

知識ベースに基づくシステムにおいて、プログラミング言語に関する知識が必要であるが、この知識は、プログラミング言語の数学的な意味の扱いが必要となる。この点については、ソフトウェア工学のワーキング・グループの報告書において、第3章基礎理論の項で述べられている。その項では、ソフトウェア工学の、より本質的な研究テーマである、アルゴリズムについて、複雑性、並列性などの観点から論じている。

4.4 マシンの理論

いかなるプログラム言語に適したオブジェクト・コードを生成するのか、あるいは直接実行するのかによって、その計算機のアーキテクチャは決定される。ＩＢＭ360、370、303X等、現在の汎用計算機の標準版であるアーキテクチャの流れも、ＦＯＲＴＲＡＮ、ＣＯＢＯＬ、ＰＬ／Ｉという言語を対象とした流れと見ることができる。ＡＬＧＯＬを対象としたBurroughsの計算機、最近のＬＩＳＰマシン、ＡＰＬマシンに至るにつれ、その傾向は顕著になりつつある。いわゆる高級言語マシン、その程度の差こそあれ、すべての汎用計算機は、汎用プログラム言語マシンとして設定されてきたと断言できる。

アーキテクチャが対象とする言語によって決定されるとしたら、その言語の骨組み、言語のアーキテクチャともいえるものは、何によって定められるのであろうか。それを定めるものが、マシンの理論ワーキング・グループが調査・研究の対象とした計算の機構と呼ぶものである。

新しい計算の機構を論じ、新しい計算機アーキテクチャを考案しょうという試みは、決して新しいものではない。むしろ、いささか飽き飽きした議論でもある。チューリング・マシン、ポストのプロダクション・システム、λ—計算、帰納関数理論等々、計算機構の理論的支柱は、電子計算機の出現以前に列挙された。そして、チューリング・マシンに基づいた計算機アーキテクチャの確立と後にＩＢＭ型計算機として不動の地位を獲得していく流れの初めから、別の計算機構に基

づくよりすぐれたアーキテクチャをという試みが行われた。しかし、いずれも芳しい成果を得ずに終る。Burroughsの孤独な抵抗であるＡＬＧＯＬマシンも、売るためには、その上にＦＯＲＴＲＡＮコンパイラを乗せねばならなかった。

それでは、何故計算機構を、あえて今、議論するのであろうか。第五世代計算機のアーキテクチャを、あえて、その大本の計算機構から論じようとするのは、今までとは違った新しい情勢が生じつつある今、情報処理技術の流れに新しい変曲点を生じつつあるという確信からである。

まず第一に、新しい計算機構、新しいプログラム言語への動きである。1970年頃から再び復興した人工知能研究は、着実な努力が続けられ、高度な利用形態を可能にしつつある。その過程で、人間の持つ各種の知識を表現しようという研究は、新しい記述系を提案し、新しい計算の機構を確立するまでになった。同じく、１９７０年頃から、プログラムの意味を厳密に定め、正しさの証明を理論的に行おうという研究が本格化した。まず、既存のプログラム（言語）に適用された。結果は、それ程芳しいものではなかった。理論の未熟さを克服する努力とともに、既存言語に対する強い反省が起った。さらに進んで、新しい計算機構、言語を提案するまでになった。

第二は、素子技術の進歩である。その進歩は、極端な低廉化と均一化である。主記憶、処理装置が、それぞれ別々の素子技術で作られ、どれもが非常に高価であった時代は、この事実を陽に認めた計算機構が優位を占める。チューリング・マシンの理論とＦＯＲＴＲＡＮ、ＣＯＢＯＬである。ＶＬＳＩ技術の進歩は、この事実を急激な勢いで遠い過去のものとしつつある。さらに均一化は、処理装置と主記憶装置を、数の上でも機能的にも同質化し、従来のハードウェア技術では考えられなかったような、高度な並列処理アーキテクチャを可能にしつつある。

新しい計算機構、新しい計算の記述系への提案がなされてきたということと、その提案を素直にアーキテクチャ上に実現しうるような素子技術が成熟しつつあるという２点が、マシンの理論グループの調査・研究の指針である。

マシンの理論ＷＧ報告書の目次と担当執事者は以下の通りである。

8. 駆動型プログラミング　（内田俊一・樋口哲野）

8．2　並列処理マシンを構築する動き

8．3　ソフトウェア工学との関連性

8．4　知識表現をめざす言語との関連性

9. 並行プログラミング　（塚本享治）

9．2　歴史的概観

9．3　適用分野

9．4　プログラミングの概念

9．5　言語の設計要計

付録　代表的並行プログラム言語

10. 対象指向型プログラミング　（横井・二木）

10．1　Class etc

10．2　Closure etc

10．3　Monitor etc

10．4　仕様記述、知識表現

11. 新計算機の骨組　（横井・内田・元吉・三国）

12. 新プログラム言語の構造　（横井・二木・佐藤）

13. む　す　び

各執筆者の所属は以下の通りである。

パターン情報部

推論機構研究室　横井俊夫

佐藤泰介

元吉文男

音声認識研究室　三国一郎

ソフトウェア部

情報システム研究室　内田俊一

言語処理研究室　二木厚吉

制　御　部

論理システム研究室　　大谷木　重　夫

情報制御研究室　　塚　本　享　治

電子計算機部

計算機方式研究室　　山　口　喜　教

（慶応大学）　　樋　口　哲　野

その他、島田俊夫（方式研）、長谷川洋（情シス研）、横山晶一（推論研）、松本裕治（推論研）、松田利夫（東京理科大）、久野巧（人間機械研）等の諸君が議論に加わった。

4.5　仕様記述と並列プロセスの理論

第5世代のコンピュータを考えるとき、多重処理系は1つの必然的な方向であろうと思われる。

また、〝ソフトウェア学〟の立場からみたソフトウェアの方法論、様々な状況の形式的取り扱い、プログラム言語の行方、あるいは支援系など、いずれを勘考しても、さらに、多重処理系が招来するであろう状況を考察しても、仕様記述の研究は第5世代コンピュータに関わる最も重要な課題の1つといえよう。

委託報告書（広瀬研究室）では、まず

「2.　協同型逐次プロセスと同期命令」において、米国国防省で企画・発表されている統一言語〝Ada〟での「待ち合せ（Concept of rendezvous）」の概念に至る経過を、主として同期基本命令（Synchronization primitives）に焦点を絞って調べる。さらに、最近、第5世代のコンピュータあるいは言語の候補者として大きな期待がよせられている「データフロー計算機」あるいは「関数的プログラミング」について

「3.　データフロー計算機／関数的プログラミングへのコメント」

で言及する。

以上のような現在の状況をふまえた上で、次には

「 4. 並列および非決定性システムの仕様記述 ― その局所的アプローチと大局的アプローチ ― 」

で、仕様記述の現状を概観する。

さて、現在の仕様記述では、1階の述語論理を用いて記述するものが多いように思われる。これは歴史的にみて自然な成り行きと思うが、これが適当か否かは疑問である。

プログラムの意味は、本来、動的なものであろうが、その仕様記述は静的に表現されることが望ましい。数学的処理も、それによって容易になる。しかし、従来の1階述語論理のみによっては、動的な部分が多く残り勝ちになろう。

どのような論理が、どのような場合に仕様記述とよく適合するかを考えなくてはならない。

「 5. 強制論理(Forcing logic) 」

では、このような論理についての1つの試みを述べる。

さらに

「 6. プロセス・データ表現(Process−Data Representation−PDR−)」

は、強制論理を用いて、プロセス側とデータ側双方の状況を記述する仕様記述の1つの試みである。

〝 複雑な状況をわかりやすくする 〟ことは仕様記述の大きな目的の1つである。したがって仕様記述には、様々なレベルとそのレベルでの記述方法が考えられる。PDRは、このような立場での、状況を分りやすくすること、と可読性を配慮した仕様記述として試みたものである。

5. の Forcing Logicと 6. のPDRは現在進行中の研究・試行であり、今後いろいろに変化することも考えられる。

この他、広瀬研究室では、比較不可能な真理値をもつ多値論理体系 ― 束の構造をもった真理値の集合上での述語論理 ― で完全性定理が成立することが確認され、その仕様記述への応用が研究されつつあるが、これについては後日にゆずる。

また、並列アルゴリズムのための形式化を行われているが、議論をより豊富にしてから提示しようと思う。各章は、各々独立に読めるよう配慮し、各々に文献表を付けた。

また、以上の報告書の内容は大勢の方々の参加と討論によるものであるが、本報告書をまとめるにあたって、

「2. 協同型逐次プロセスと同期命令」

については　土居範久氏が、

「3. データ・フロー計算機／関数的プログラミングへのコメント」

と

「4. 並列および非決定法システムの仕様記述」

については　米沢明憲氏が、

「5. 強制論理」

については　広瀬健氏が、

「6. プロセス・データ表現」

については　斎藤信男氏が、それぞれまとめたものであることを付記する。

4.6 言語の機械翻訳とその関連技術

言語の計算機処理の中でも、言語の機械翻訳の研究は、一種の結合技術としてみるべきものである。そこには計算機のハードウェア技術、ソフトウェア技術と、言語情報そのものの工学的観点からの整理と活用のシステムという3つの要素が存在する。機械翻訳のためのハードウェア技術としては、大量の辞書情報やテキスト情報のデータベース、検索のためのアーキテクチャが必要で、これからの開発にまたねばならない。さらに機械翻訳は人間が種々のレベルで計算機に介入するといったシステム構成をとることが必要であって、そのための高度な知的端末装置の開発が急務である。

委託報告書（長尾研究室）では、このハードウェアの面は省略し、主としてソフトウェア面と、言語情報の面についてとりあげた。

まず第1章の機械翻訳の現状では、世界の機械翻訳の歴史と現状、機械翻訳の

種々の方式とその得失について述べた。そして特に最近ヨーロッパ共同体で真剣にとりあげられている多言語間機械翻訳についてふれた。

ヨーロッパ共同体では加入６ケ国語（近く９ケ国語に拡大の予定）に文書を翻訳しなければならず、ぼう大な人数と金を使って行っているが、とても処理しきれず、１９７６年に機械翻訳システムＳＹＳＴＲＡＮを導入し英仏翻訳に利用している。しかし６ケ国語、９ケ国語になるとこのような単一言語対単一言語の翻訳システムでは間に合わず、またＳＹＳＴＲＡＮの翻訳の質にもまだまだ満足できないため、現在の最新の技術を用いて多言語間自動翻訳システムを、これから４～５年計画で作成し実用に供するという計画で、完成すれば大きな効果を与えるものと考えられる。

第２章では機械翻訳システムの構造についてくわしく説明した。現在の機械による翻訳はMachine－aided Human Translation、即ち人間の翻訳作業に機械がいかに便利さを提供し、人間を助けるかという観点からのシステムであるが、これからはHaman－aided Machine Translation System を作ってゆくことになる。このようなシステムでの各種の開発要素について詳しく説明した。

第３章は京都大学で研究を行った技術論文（特に電気工学関係）の論文表題の英文和訳の研究について報告した。論文表題のもつ構文構造は、文の構造ほどの多様性はない。副詞・形容詞などを無視すれば、構造の基本的要素は名詞と動詞（ing形）、前置詞およびＡＮＤなどの接続詞である。科学技術情報センターの文献速報電気工学編に現れた約１０,０００論文の表題について、このような基本的文型をしらべたところ約1,０００であった。そこでこれら1,０００の文型に対して名詞句の係り受け関係にあいまいさの生ずるところのみを、簡単な意味要素のチェックでしらべることによって、その係り受け関係を明らかにし、その他の部分はそのままで１つの構造パターンとして採用した。この構造パターンに対応する日本語の構造を用意し、日本語の論文表題を合成するものである。このようにすることによって解析の方法は比較的簡単となり、日本語の対応も楽に行うことができた。現在８０％の翻訳成功率である。いくつかの点で改良を行い、また適当に人間が介

入することを行えば、実用の見通しも暗くない。

第4章は京都大学で研究を行った日英機械翻訳システムに関する報告である。計算機ソフトウェアのマニュアルの文章を解析して英語に翻訳するもので、日本語の形態素解析、構文解析、意味解析、日本語深層構造から英語の深層構造への移行、英語の構文合成、形態素合成とからなる。現在システムが出来上がったばかりで、その評価と改良はこれからである。システムはＬＩＳＰによって書かれている。

第5章は機械翻訳に用いるソフトウェアについて解説している。機械翻訳においては、言語の文法・辞書情報と、それらをどのように表現し計算機に入れ、これを活用して解析と合成を行うかという2つの部分は、明確に分ける必要があるとされている。2の場合のソフトウェアとしては種々の言語の特殊性をも受け入れられねばならない。現在の代表的なソフトウェア・システムの構造について説明した。

第6章は機械翻訳における辞書の役割について述べている。特に、京都大学で研究した日常用語の通常の辞書を計算機で活用しようとするときの、諸問題についての研究結果を説明した。

昭和55年3月発行

発行所　財団法人　日本情報処理開発協会
東京都港区芝公園3丁目5番8号
機械振興会館内
TEL（434）8211（代表）

印刷所　株式会社　昌文社
東京都港区芝5丁目26番30号
（全専売ビル）
TEL（452）4931

54-R014