Library and Information Science No. 9 1971

自然言語の処理を目的とするＩＲシステムのプログラム的考察

A Survey of IR Programming Systems for Natural Language Processing

斉　藤　孝
Takashi Saito

Résumé

As conversation between man and computer becomes more intense, the problems associated with communication between them become more apparent. This resulted in the development of artificial languages in the field of computer programming and systems. Natural language processing for the computer-based information storage and retrieval system can be defined as manipulation of symbols and files with artificial language.

This survey covers current developments and design principles of programming languages for symbol manipulation and file management for application to natural language processing. The significant features of these languages for IR oriented data processing systems are discussed principally in terms of examples drawn from existing and proposed systems in Japan.

はじめに

コンピュータの発達にともないソフトウエアシステムの開発も著しいものがある。そのなかでＩＲシステムをコンピュータにより設計しようとする試みも，便利でしかも記述の簡単なプログラム言語が準備されることによって一段と容易になっている。そこでこれまでコンピュータに直接触れることのなかったＩＲの実務家に身近なものになりつつある。

初期の試みは，コンピュータそのもののハードウエアシステムの貧弱さにも実現化の失敗はあるが，むしろソフトウエアシステムの不完全さや，複雑さなどによりア

斉　藤　孝：東京芝浦電気株式会社電算機システム技術部
Takashi Saito, Computer System Engineering Dept., Tokyo Shibaura Electric Co.

プリケーションシステムの実務家が簡単には接近できなかったことが原因し，初歩的な開発に終っていることが多い。

プログラム言語の開発は，コンピュータ処理の目的が科学計算を主体としたものから事務計算に移行していくにしたがって高度なものとなっている。プログラム言語は自然言語に対応する１種の人工言語である。そこで言語とするからには，自然言語と同様に文字記号や，文法とか意味といったものがある。ＩＲシステムが機械翻訳と同じようにデータとして自然言語を取扱うとすれば，直接翻訳 (assemble) とか編集翻訳 (compile) といったプログラム言語の取扱いと同じことをコンピュータは考えなければならない。この両者には，プログラム技術として共通したものが存在することになる。

本稿は以上の観点に基づいて，自然言語を取扱うシステムの設計と開発において検討すべきプログラム技術を考察したものである。

I. ＩＲシステムの性質

A. システムの構成

ＩＲは正式には Information Storage and Retrieval が示すように情報の蓄積を前提としたシステムである。蓄積とは索引づけ (indexing) による情報ファイルの作成のことであり，検索は探索 (searching) によるファイル中のレコードや項目 (item) の検索をすることに対応する。索引づけは文献などの内容を抽出するといった主題分析によってシステムの公式語 (formal language) に変換することである。公式語としては，文献の記述に使用する自然言語に対してシステム内でのみ有効な規則をもった情報の媒体であるキーワードとその集合であるシソーラスなどが考えられる。

B. システムのモデル

ＩＲシステムは情報のレベルによって事柄検索 (Fact Retrieval) と文献検索 (Document Retrieval) の各システムに分けられる。ＦＲシステムは与えられた質問に対して直接回答をしようとするもので数値データの検索に見られるような二者択一的な情報を単位とする。たとえば，「絶対零度とは？」といった質問に対して「摂氏マイナス 273.15 度」といった回答をする。これに対してＤＲシステムは，情報の所在を指示して「理化学辞典の745ページを参照」といった回答をする。したがって，ＦＲシステムは確定的な検索を，ＤＲシステムは確率的な検索のモデルが成立する。

1. システムの機能

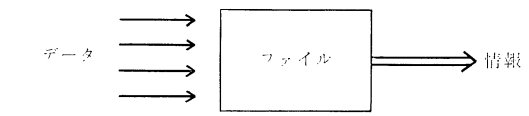
システムの基本的な機能は，大量のデータをファイルしておいて必要なときに情報を検索して取出す図１-Ａのようなものである。

2. システムの手順

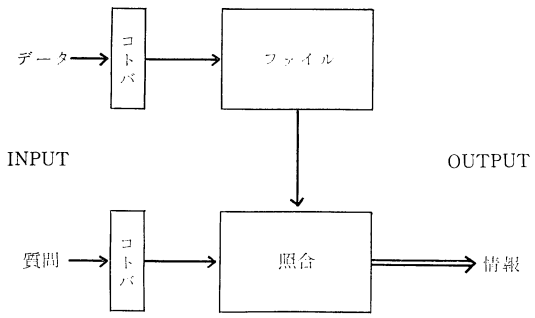
これは人間の記憶の方法と記憶の取出し照合における思考判断の演算，つまり知識活動に似せて図１-Ｂのようにする。

3. システムとコンピュータ

コンピュータを中心とするシステムは，処理内容から見れば記号処理とファイル処理を主体にし図２のような関係となる。



A. システムの機能



B. システムの手順

図１．ＩＲシステムの機能と手順

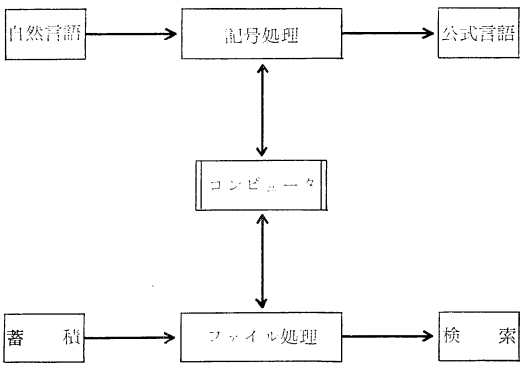


図２．ＩＲシステムとコンピュータ

C. システムの技術

ＩＲシステムの設計に利用される手法は，大きく次の４つの段階に分けて考えられる。

1. 情報の整理

二次情報の作成を目的とするものであって抄録法，索引法さらに機械翻訳といったものがある。

2. 情報ファイルの作成

ＩＲに利用できるファイルの種類は，次のようなものがある。

(i) 直接ファイル (Direct file)
(ii) 逐次ファイル (Sequential file)
(iii) インデックス逐次ファイル (Indexed sequential file)
(iv) 木構造ファイル (Tree structure file)
(v) インバーテッドファイル (Inverted file)

3. 質問の分析

これは論理演算や自然言語の構文 (syntax)，意味 (semantics) の分析のことである。

4. ファイルの検索

レコードの項目などの探索法に次のようなものがある。

(i) 逐次探索法 (Sequential access)
(ii) ２分探索法 (Binary search)
(iii) 索引探索法 (Index term search)
(iv) 自然言語探索法 (Natural larguage search)

II. ＩＲシステムのプログラム開発

A. ハードウエアシステムとソフトウエアシステム

ＩＲなど一般にアプリケーションプログラムと称するシステムを開発するにあたり問題となることは，先ず使用するコンピュータシステムについての考慮であろう。

コンピュータシステムとは汎用のものでは演算装置を中心に入出力装置システムといったハードウエアシステムと，論理的実体とみなされるソフトウエアシステムから構成される。ハードウエアシステムがプログラム開発に影響を与えるのは，概して演算スピードとか記憶容量の大きさとか，外部記憶装置の種類とかさらにアキュムレーションの方法とかいったことが考えられる。しかし実際はこれらの制約は，準備される基本ソフトウエアシステムによってほとんど意識されないようになっている。ソフトウエアシステムはハードウエアシステムに対応して呼ばれるが，その基本的な目的はコンピュータにある手順を記憶させたりそのための連絡をするインターフェースといった役割をするものであり，コンピュータプログラムと同義と言える。ただし，コンピュータプログラミングといった場合は，論理的な実体に限らないで回路上の準備などによる物理的な実体も含めて指すことがあるので注意したい。

ソフトウエアシステムは初期のコンピュータにあったようなハードウエアシステムのコントロールを中心とした思想のものからプログラムのためのプログラムシステムといった人間の利用に適するプログラム言語という新しい概念を生みだし，非常に高度なソフトウエアシステムの体系が構成されつつある。

B. ソフトウエアシステムの体系

ＩＲの処理もコンピュータを中心にシステム化を考えると，特定の問題を処理するプログラミングシステムが成立する。それは，具体的には索引を作成したり，探索をしたりする作業をシステム的に分析しさらに体系化し論理的手順を抽出するアルゴリズム作成の動きになる。この種のプログラミングシステムのことをアプリケーションプログラムと定義したい。

アプリケーションプログラムの作成には，先ずソフトウエアシステムの体系を理解しなければならない。今日の汎用コンピュータのソフトウエアシステムは次のようなサブシステムによって構成される。

(i) プロセスサブシステム： プログラムの実行の単位をジョブと呼ぶがこれを複数個に集めてプロセッサに持ってくる機能を果す。
(ii) 記憶サブシステム： 記憶装置の領域の割当てや保護を行なう。
(iii) コンソールサブシステム：時間割りシステム(TSS)などのように端末機器の利用者がコンソールとの入出力管理を目的とする。
(iv) ファイルサブシステム： 磁気テープ，ドラム，ディスクといった２次記憶装置を通常ファイル装置と呼ぶが，これらの管理を目的とする。
(v) 言語プロセッササブシステム： これはアプリケーションプログラムの作成に解放されている唯一のソフトウエアシステムの領域である。これまでのサブシステムをオペレーティングシステム（ＯＳ）と呼ぶのに対し，このサブシステムはしばしば基本ソフトウエアシステムと呼ばれる。基本ソフトウエアシステムは，アセンブラ，コンパイラとい

ったプログラム言語の形態をしている。

C．自然語とプログラム言語

ＯＳを含めて基本ソフトウエアシステムの発展は，ハードウエアシステムのそれと同じように第1，2，3，世代と呼ぶ区分がある。第1世代は機械語とか記号言語(Symbolic language)といったものによる絶対番地を指定する絶対コーディングの (Absolute coding) プログラム記述であった。次に大きなプログラムの単位をいくつかの部分に分け，その部分の中で最初から何番目の番地といった具合に指定する相対コーディング (Relative coding) が出来るようになった。この発展がアセンブラー言語の開発となった。

同時に，日常人間が使用している数学上の表現とか，あるいは処理の手順といったものを英語に近い言語，つまり自然言語や疑似自然言語を使用してプログラムを記述しようとすることとなった。これが FORTRAN, COBOL といったコンパイラー言語である。コンパイラーはひとつの数式などを機械語やアセンブラーなどではいくつかのステップにより記述しなければならないのに対して，約束された文法によるステートメントから沢山のプログラムを生成するといった自動コーディング (Automatic coding) を行う。

コンパイラー言語の出現により，コンピュータに使用できる言語は，自然言語を含めて図3のように言語のレベルによって定義づけられる。すなわち，自然言語を頂点としてその下に問題向き言語 (Problem oriented language) と手続き向き言語 (Procedure oriented language) と2つのタイプの言語に分けられ，その下にアセンブラー言語と機械語とが続く。

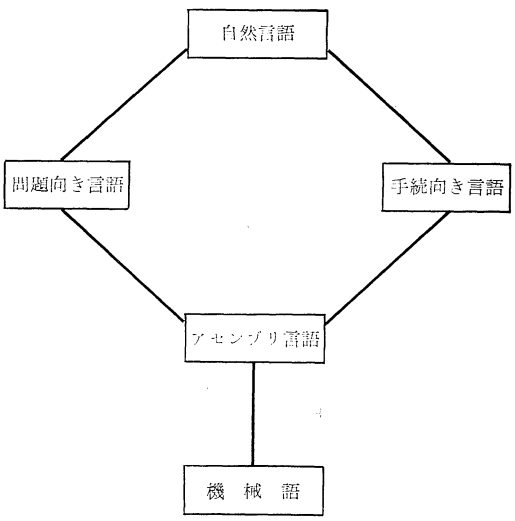


図3．自然言語とプログラム言語

自然言語とプログラム言語を対応づけて特色を考えると，人間はシステム化により問題を限定しその解決の手順を出来るだけ自然言語によって表現することを試みる。自然言語は問題の記述に適する言語 (object language) である。しかし，自然言語はあいまいさが多く，論理性にしばしば不足する。そこで論理的思考に適する言語 (metalanguage) を考えそれをプログラム言語とする。

D．プログラム言語の分類

ここで対象とするプログラム言語は，コンパイラーレベルの言語である。普通プログラム言語は次のように分類される。

(i) 手続き向き言語 (Procedual Language) COBOL, FORTRAN, ALGOL, PL/1

(ii) 非手続き向き言語 (Non-Procedual Language) RPG, DETAB

(iii) リスト処理言語 (List Processing Language) IPL-V, LISP, L 6

(iv) 記号処理言語 (Symbol Manipulation Language) COMIT, SNOBOL

(v) 数式処理言語 (Formula Manipulation Language) FORMAC, MATHLAB

(vi) 問題向き言語 (Problem Oriented Language) GPSS, APT, COGO

(vii) ファイル処理言語 (File Management Language) IDS, GIS

この分類の中で (i) 手続き向き言語は，別名汎用プログラム言語と称されるので本稿では，必要なこと以外は考察しない。

ＩＲシステムのプログラム設計と開発に関係するものは，言語の仕様に，またコンパイラー技術の面には (i) 手続き向き言語の設計思想が，プログラムの構成としては，(ii) 非手続き向き言語や (vi) 問題向き言語が，さらに最も重要なアルゴリズムは，リスト処理，記号処理言語などがある。同時に，外部入出力装置のコントロールに関する技術は，ファイル処理言語に学ぶものが多い。本稿は (iii) と (iv) と (vii) を中心に考察する。その前に，手続き向き言語を簡単に知っておきたい。

その代表的な FORTRAN, COBOL は，あまりにも

有名なので省き,ALGOLについて言及しよう。ALGOLは1959年に提出された計算手順の記述言語 (metalanguage) である。したがってその記述法もBNF (Backus Naur Form) 文法[1]に基づいて整然としている。ただし，実際にコンピュータ用には，データの入出力の部分に関する取り決めがないことなどにより互換性がないのが欠点である。ALGOL の大きな特徴は，回帰的 (recursive) な取扱いや変数の取扱いといったことがはっきり言語的に定義の出来る方法があることである。それによって，FORTRAN と違って言語の構文にしたがってプログラムを生成する構文向きコンパイラー (Syntax Direct Compiler) と言われる[2]。

ALGOL は計算のアルゴリズムを明記することを主体に設計されているのでＩＲ的には，どこまで有効かは多くの疑問がある。

PL/1 は FORTRAN などの欠点であった文字と英数字の取扱いがうまくできないことを改良してある。すなわち COBOL や FORTRAN などが使用されるシステム言語としてそのまま使用できるように設計されている。さらにリアルタイムシステム，ＯＳ，コンパイラーの記述といったことに利用出来る言語である。PL/1 は COBOL の拡張言語とともにＩＲシステムの開発にとって今後有力なシステム言語となりそうである。

非手続き向き言語とは，FORTRAN, COBOL に代表される言語により記述されるプログラムが，概して一次元的であり，処理手続きにより始めから１つずつ実行していくタイプなのに対して，処理の手続きが多次元的に定義できるものを言う。例として RPG (Report Program Generator) などは決められた枠の中に必要パラメータを記入し，必要なプログラムを生成していく。

現在開発されているＩＲアプリケーションプログラムは，この種のタイプに属するものが多い。

E．プログラム言語とＩＲ

現在，ハードウエアシステムを中心に第３世代と称されるが，この時期において高水準の言語の普及や，本格的なＯＳの開発などソフトウエアシステムの著しい発展が見られる。

特にＯＳの制御プログラムは，バッチのモニターシステムとリアルタイムシステムの制御プログラムを統合し，データベースの管理機能を拡大したり，マルチプログラミング，遠隔入出力，マルチプロセッシング，TSS などを導入した極めて高度なものとなっている。

アプリケーションプログラムも映像表示装置を利用したもの，ソフトウエアの性能自体を分析するソフトウエアといったものから，もっと個々の問題の処理に適するソフトウエアシステムといったぐあいに開発されている。

ＩＲシステム用のソフトウエアも，ＩＲのシステム分析が進むにしたがって，必要とする機能が体系化され独自のアルゴリズムとその演算が形成されようとしている。

III. システムのアルゴリズムと演算

A．記号処理のアルゴリズムと演算

自然言語のように文字列 (string) などをデータとするＩＲの問題には，日本語と英語の語順の違いを操作する機械翻訳とか，あるステートメントから目的のプログラム記号を生成するコンパイラーの編成操作と共通する文字列処理 (String Manipulation) やリスト処理 (List Processing) といった所謂，記号処理がある。

そこで，記号処理のアルゴリズムと演算を整理してみる必要がある。

一般に計算のアルゴリズムを見ると，計算という演算過程，すなわち整数とか実数で表現された数値の量の変化に関心があることは明らかである。これを記号操作において構造として見た場合，その中においては値の間に加減乗除して他の変数の量といるという，きわめて弱い関係しか成立しない。

しかし，記号処理に重点を置くと記号と呼ばれるそれ以上には分割できない基本単位を考えて，その配置，構造,関係などの変形に注目することになる。このことは，構造的な変化に興味があり結合とか連結といったアルゴリズムを意味する。

B．プログラミングの基本概念

非数値データの記号を操作するプログラム言語を見ると，プログラムの技術上欠くことのできない基本概念がある。

1．データの表現

記号処理を理解するのに必要な概念として，リスト (List)，リスト構造 (List Structure)，文字列 (String)，二進木 (Binary Tree) といったデータの表現に関する定義がある。

通常のコンピュータは連続するデータを取扱うように作られていて，記憶装置にたとえば KEYWORD といった文字列は１文字１番地とすれば，図４のように連続

番地	内容
(1000)	K
(1001)	E
(1002)	Y
(1003)	W
(1004)	O
(1005)	R
(1006)	D

A．文字の挿入

番地	内容
(1000)	K
(1001)	E
(1002)	Y
(1003)	—
(1004)	W
(1005)	O
(1006)	R
(1007)	D

B．前文字の挿入後

図４．文字列の連続番地の記憶

番地に割付けられている。したがって順番に入っている単純なデータを操作するには都合がよい。しかし，たとえば KEY-WORD というように分離記号を入れようとするとW文字以下の番地をひとつづつ下にさげたりして空の場所を作り入れなおさなければならない。つまり構造というデータの性質を表現しようとするとデータの操作が非常に難しくなる。このような複雑なデータをいかにコンピュータの記憶装置内で表現し，それを能率よく処理するかといった目的でリスト構造のアルゴリズムとの演算が考えられている。

2．リスト構造

リスト構造ではデータを連続して記憶させる代わりに，レコードの中にデータだけでなくデータと論理的に続くデータの入っている番地を含むようにする。これをポインター又はリンク（図 5）と呼ぶ。

3．リスト処理

リスト構造の例でも明らかなように次のようなアルゴリズムが抽出される。すなわちデータの追加の場合には，空いた場所にデータを入れてそのひとつ前のデータのリンクを呼び出し，そのリンクには現在のデータの入った場所を記憶させる。一方データを削除するには目的のデータを取除いてリンクの部分をひとつ前のデータのリンクに入れておく。

番地	内容	ポインター
(1500)	K	1502
(1501)	Y	1503
(1502)	E	1501
(1503)	—	1505
(1504)	O	1506
(1505)	W	1504
(1506)	R	1507
(1507)	D	終り

図５．リスト構造の記憶

このような処理の演算に次のような機能が必要となる。

(ｉ) データの定義に関するもの

(ii) データの追加と削除

(iii) データの呼び出しと入れかえ

(iv) データの書き写しと追跡

4．文字列とその処理

ある記号の集合を文字列（string）と呼び，その中で指定した部分集合をパターンと呼ぶ。そこで文字列の処理とは，文字列のパターンを調べたり別のパターンに変換する演算を言う。

5．リスト処理と文字列処理

文字列処理では，文字の集合からパターンを追加したり削除したりする作業を行なうので当然のこととして文字列をリスト構造にしておきリスト処理をするほうがよくなる。したがって通常の文字列処理は，リスト処理の演算も含むものとなる。

C．記号処理アルゴリズムの道具

リスト，リスト構造，文字列，二進木といったデータ表現の処理のアルゴリズムを演算するには，連想記憶装置（Associative Memory）や押込み記憶装置（Push-Down Stack）といった道具を論理的に準備する。

1．連想記憶装置の模倣

ハードウエア的に連想記憶装置とは，本来これまでの記憶装置と逆の制御を行なう図６のような記憶装置を指す。たとえば，記憶装置に対してキーワードを与えてそれと一致した内容を格納している番地を知るといった機能をもてる。この種の記憶装置の開発は，現在磁性素子では困難なため実用化されていない。したがってソフトウエア的に模倣しようとする。リスト構造とその演算もひとつの試みである。

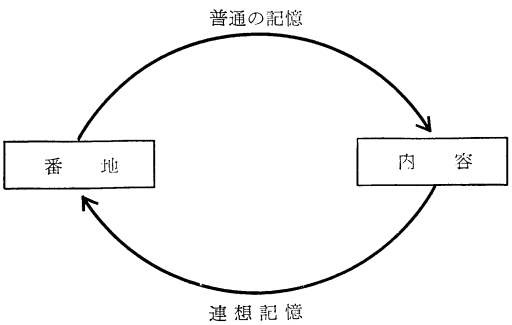


図６．連想記憶装置

2. 押込み記憶装置の模倣

これはＰＤスタックとか棚と称するものを記憶装置内に作成することにより実現する。

このスタックの利用は記憶装置内に演算データを一時的に格納し，先に書込まれたデータほど後に読出すといった LIFO (Last-In-First-Out) の原理を与えるアルゴリズムの道具とし利用される。

数式とその演算の取扱いに便利な記法である逆ポーランド記法は，図７のような格納状態となる。逆ポーランド記法は演算子後置の形に文字列を変形したものである。この形に表現したものは，その演算処理手順がスタックによって演算処理中は他の番地の必要もなく中間結果を格納する場所を求める必要もない。

3. 回帰的操作

一般に手続き中に自分自身を用いること，具体的にはプログラムＰがＰ自身を呼出し用いることを回帰的操作 (recursive procedure) と呼ぶ。この操作は有限個の規則により問題の解法というアルゴリズムの定義の本質である。回帰的にプログラム言語が記述できるということは，記憶容量を少なくすることと，行なう動作を単純にすることができ有限の法則で無限の表現（図 8）ができることを意味する。

操　作	格　納　状　態
X	X
Y	Y，X
＋	(X＋Y)
U	U，(X＋Y)
V	V，U，(X＋Y)
W	W，V，U，(X＋Y)
÷	V/W，U，(X Y)
＋	(U＋V/W)，(X＋Y)
X	(X＋Y)・(U＋V/W)
T	T，(X＋Y)・(U＋V/W)
＋	(X＋Y)・(U＋V/W)＋T

図７．押込み記憶装置の動作

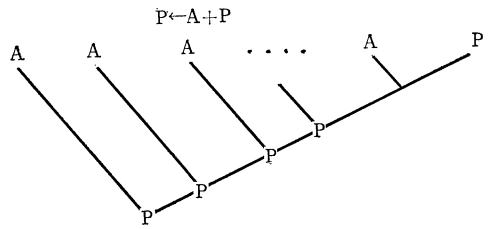


図８．回帰的操作の表現

D．記号処理のプログラミング機能

記号処理のプログラミングは，非数値的計算の場に見られる。非数値的計算とはこれまでの事務処理などに見られるソート，マージや編集 (editing) といったものや，COBOL, FORTRAN といったプログラム言語を編成翻訳 (compile) することなどである。ＩＲにおける問題の処理は，典型的な非数値的計算の分野に入る。

これらは自然言語により記述された不確定なモデルである意味情報や図形などのデータを取扱うからである。

記号処理のプログラミングは，一般に次の４つの基本的な演算単位により満足できる。

(ｉ) 文字列の登録
(ii) 文字列の照合
(iii) 文字列の連結
(iv) 文字列の置換

IV. 記号処理のプログラム言語

A．機械翻訳のプログラム言語

ＩＲに比較して自然言語のコンピュータによる翻訳，所謂機械翻訳の歴史は古く1946年頃に始まる。無論，この頃にはコンパイラーレベルのプログラム言語は開発されてはいなかった。その後1954年にＭＩＴの Yngve が中心になり言語学的な考え方や用語法を取入れた文字列処理のプロセッサーが開発され COMIT[3] と名づけた。COMIT 言語の記述法は，一定の順序に従って実行される多くのプログラム規則の集合による。この規則により操作されるものをデータと呼び，データは記憶内部に設定した作業領域 (work space) において文字列を保持する。たとえば，文字列は文字を要素としそれを＋記号を用いて次のように分解表示する。

Ｉ＋Ｎ＋Ｆ＋Ｏ＋Ｒ＋Ｍ＋Ａ＋Ｔ＋Ｉ＋Ｏ＋Ｎ

COMIT の演算とはこのようなデータに対しプログラム規則により記述した変換手順を適用していくことを言う。

1. 置換の規則

この規則は文字列の置換 (replacement) を目的とする。

データ	INFORMATION STORAGE
規　則	*STORAGE=RETRIEVAL*
実行後	INFORMATION RETRIEVAL

2. 順序の変更の規則

この規則は作業領域内のデータ要素の順序を変更 (re-arrangement) する。

データ	ＫＥＹＷＯＲＤ
規　則	*Ｒ＋Ｏ＝２＋１*
実行後	ＫＥＹＷＲＯＤ

この規則の意味は，等号の右辺の２と１という数によって，左辺の２番目と１番目のデータの要素が交換されることを示す。

3. 挿入の規則

この規則によって文字別に指定したパターンを任意の場所に挿入 (insertion) できる。

データ	INFORMATION RETRIEVAL
規　則	*INFORMATION≡1+STORAGE*
実行後	INFORMATION STORAGE RETRIEVAL

この他の規則としては削除 (deletion) などがある。COMIT は機械翻訳向きプログラム言語であることに特色がある。

B．文字列操作のプログラム言語

文字列を操作するアルゴリズムと演算をより発展させ開発したものに SNOBOL[4] がある。

SNOBOL は 1964年に D. J. Farber などによる発表により有名になったものである。現在は SNOBOL-4 という拡張された本格的な言語も開発されている[5]。

SNOBOL が先の COMIT と違う点は，COMIT が規則と称する一種のサブルーティンの単純な集合によって記述するのに対し FORTRAN などにある言語仕様を含んでいることである。

その違いを整理してみると次のようになる。

(ｉ) COMIT は文字列そのものを取扱っていたのに対し，SNOBOL は文字列変数と呼ぶ名前づけができる。したがって文字列の登録など取扱いが非常に容易である。

(ii) 判断命令がある。したがって文字列の照合などにおいて合致しない場合など次の手順に進むことができる。

(iii) サブルーティン (function) の定義ができる。

(iv) レコードをひとつにまとめることができる。

SNOBOL は COMIT などと比較すると，一段と汎用性のある文字列処理が可能な言語といえよう。

1. SNOBOL の文字列

文字列とは，文字を適当につなぎ合せたものを言う。TOSBAC-SNOBOL[6] において取扱い可能な文字は，次の48種類である。

(ｉ) 数字：０，１，２，３，４，５，……８，９　　10種類

(ii) 英字：Ａ，Ｂ，Ｃ，Ｄ，……Ｙ，Ｚ　　26種類

(iii) 特殊文字：＋，－，＝，(，＊，▼，$，ゟ など12種類)

したがってカナ文字は対象にはならない。

これらの文字により文字列とは，次のようなものを言う。

▼ABCDEF▼　　▼360▼　　▼INFORMATION▼

2. 文字列の登録

この操作に当り文字列の取扱いが容易なように文字列変数といった名前づけをする。

STRNG 1=▼INFORMATION RETRIEVAL▼

これは STRNG 1 と名前づけをした文字列変数に右辺の文字列を登録することを意味する。

3. 文字列の連結

この演算は文字列を２つ以上結びつける。

STRNG 1=▼INFORMATION STORAGE▼	
STRNG 2=STRING 1 ▼AND RETRIEVAL▼	
実行後	INFORMATION STORAGE AND RETRIEVAL

連結 (concatenation) は自然言語のように単語，語句，文章といった可変長の文字列を利用し文を編集しようとするテキスト編集 (texteditor) にとって非常に都合がよい。テキスト編集のプログラムをアセンブラーなどで作成しようとすると意外に手間取る。

4. 文字列の照合

キーワードなど可変長の文字列の比較，照合などの操

作は，文字列の連結同様にＩＲでは必ず利用する。SNOBOL において照合の演算とは，次のようなパラメータにより文字列を調べることを言う。

(ｉ) 同じ文字列である
(ii) ある文字列の１部である
(iii) まったく違う文字列である

```
STRNG 1=▼COMPUTER BASED INFORMA-
        TION▼
KYWRD =▼COMPUTER▼
照 合 実 行    STRNG 1    KYWRD
```

この場合 STRNG 1 を参照文字列 (reference string) と呼び，KYWRD をパターン文字列 (pattern string) と呼ぶ。

照合の演算は前者の中に後者が含まれているかどうかといった判定を行なうことである。

5. 文字列の置換

自然言語に単語を自由に与えることにより違った文字列を生成するといったことは，ＤＲシステムにおいてわずか KWIC 索引の編集にしか見られない。しかし，FR システムにおいて推論と変換の手続きは文字列である規則集の置換ばかりである。SNOBOL の文字列の置換とは，照合と登録を組合わせて，文字列の１部または全部を他の文字列にすることをいう。

```
STRNG 1=▼DOCUMENT RETRIEVAL SYSTEM▼
STRNG 1  ▼DOCUMENT▼=▼FACT▼
実 行 後  ▼FACT RETRIEVAL SYSTEM▼
```

6. 基本的パラメータ

SNOBOL は基本的には，これまで説明した４つの演算機能の組合せにより問題の手続きを記述しようとする。そのプログラム構造は文字列をリスト構造に展開しておき，次に示すようなパラメータにより動作 (図 9) するサブルーティンから構成される。

(ｉ) ｘをｙの直後に入れる。
(ii) ｘをｚの直前に入れる
(iii) ｘを相続くｙとｚの間に入れる
(iv) ｘを文字列の終りに入れる
(ｖ) ｘを文字列の頭からｎ番目に入れる

C．手続き向き言語による記号処理

COMIT, SNOBOL は記号処理だけを目的とした特殊言語である。したがってソフトウエアシステムとしてしばしば準備されていない場合がある。また機能的に内部演算のみを主体とするために外部記憶の演算をするのに都合が悪い。特に大量の情報を磁気テープなどに蓄積しようとするＤＲシステムにとって，このことは大きな欠点となる。こんな理由で汎用言語と言われる COBOL とか PL/I などにも記号処理の演算機能を加えることが提

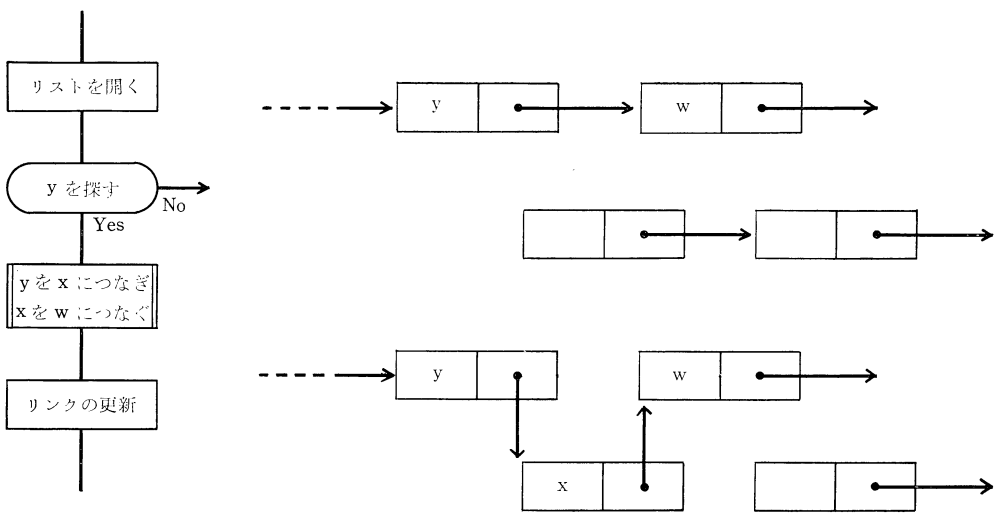


図９．リ ス ト 操 作 の 基 本 動 作

案されている[7]。

COBOL は最近の傾向としてコンピュータシステムの周辺機器の発達にともなって，初期の COBOL に見られなかったソート，マージ，表引き (table search)，乱呼出し (random access)，乱処理などとさらに遠隔地の端末からのデータ処理に通信機能が加えられつつある。

1．COBOL の記号処理

元来，COBOL のデータとはカードの何桁目から何桁といったように固定位置 (fixed format) による入力方法を採用している。これは操作できるデータの最小単位の文字の取扱いを都合よくするためである。しかし通信端末から入力されるデータの性質は，固定長固定位置といったものでなく自然言語の性質と同様に可変長のものである。したがって，出来るだけ冗長性を少なくすることが要求され余分な空白を除いたり適当な分離記号を入れることによって情報を効率よく伝達しようとする。この処理は1種の記号処理にほかならない。COBOL の記号処理は，INSPECT，STRING，UNSTRING といった動詞による文で記述できる。

2．文字列の置換と照合

文字列の中に含まれる適当な文字の数を数えたり，連続する文字列のパターンを数えたりする。

```
データ　STRNG 1≡INFORMATION
INSPECT　STRNG 1　TALLYING　COUNT 1
　　FOR ALL “I”
実行後　　COUNT 1=2
```

この例は STRNG 1 に登録された INFORMATION という単語中に含まれるＩという文字を調べその個数を COUNT 1 に入れる，という記述である。又，特定の文字列を別の文字列に置換するという操作は次のようにする。

```
データ　STRNG 1=COMPUTERS
INSPECT　STRNG 1　REPLACING ALL “E”
　　BY “I” “R” BY “N” “S” BY “G”.
実行後　STRNG 1=COMPUTING
```

3．パターンの抽出

パターンとは適当な文字列のことであるから単語，語句といったものがパターンになる。そこで文章などから単語や語句を抽出するといったパターンの抽出は次のようにする。

```
データ　STRNG 1=COMPUTER INFORMA-
　　TION, SYSTEM
UNSTRING STRNG 1 DELIMITED BY “,” OR
SPACE INTO WRD 1, WRD 2, WRD 3.
実行後　WRD 1=COMPUTER　WRD 2=IN-
　　FORMATION　WRD 3=SYSTEM
```

4．文の生成

文の生成すなわち連結は，UNSTRING 文の逆の機能を果す。これによりいくつかの文字列を連結しひとつの新しい文字列を生成できる。

```
STRING WRD 1 DELIMITED BY SPACE WRD
2 DELIMITED BY “,” WRD 3 DELIMITED BY
SPACE INTO STRNG 1.
```

この例では実行後，先のパターンの抽出と逆の結果が得られる。

COBOL の記号処理機能の追加は，今後のアプリケーションの開発をより容易にすると考えられる。特に，SNOBOL などの特殊言語にない豊富な入出力の機能があるからである。

ただし回帰的な関係の定義が許されないのはひとつの欠点として残る。

D．構造を取扱うプログラム言語

記号処理には，文字列で示されたデータの性質である構造を操作しようとするリスト処理も含まれる。リストとは，本来我々の周囲に簡単に見ることのできる表，帳簿，名簿といった形式を抽象化したにすぎない。文字の系列もリストの1種と理解できる。また，項目やその集合のレコードさらにファイルといったものもリストである。

1．リストの演算

一般にリスト処理の演算機能を考えると，次のような基本操作が抽出される。

(i) あるデータを示す文字列を探しだす。リスト処理では文字列をブロックと称する。

(ii) ブロック中のデータを読み出し，データについて演算し新しいデータを書き込む。

(iii) 新しいブロックをリストに挿入する。

(iv) あるブロックを取除く。
(v) 2つのリストを1つに連結する。
(vi) 1つのリストを2つに分離する。

これらの演算は本質的に文字列のそれと同じである。

2．リスト構造の本質

リスト処理の特徴は，構造という概念をいかに表現し操作しようとするかである。リスト構造はそこで，単に階層的分類による文，節，句，単語，文字といった見方ができるように「リストを要素として許すリスト」といった関係づけである。いま一度，このことを BNF 方式で定義づけるならば，次のようになる。

〈リスト構造〉::=〈データの配列〉|〈データ〉〈〈リスト構造〉の配列〉

この記法で明らかなように，右辺に定義したものが左辺にも登場するといった回帰的な関係づけにリスト構造の本質がある。

3．リスト構造の表現

リスト構造は図示することにより樹木のようになるので別名木構造と称する。木構造の性質は自然言語の構文や算術式などに見られる演算順序の関係（図10）などが引合にだされる。算術式において最小単位のリストとしては，節と呼ぶ演算記号と2つの変数からなっている。図10の木構造は逆ポーランド記法による演算順序を示す。

4．プログラム言語

リスト構造を取扱う言語として，すでに開発されたものに LISP[8], L 6[9] などが知られている。LISP についてはこれまで多くの紹介[10]があるので割愛する。本稿において L 6 について，TOSBAC 用に開発された T-L 6[11] を説明する。L 6 は Loboratories, Low-Level Linked List Language の略称であり，本来はコンピュータによる動画 (animation) の作成を目的としている。しかしアメリカにおいて，しばしば言語学的研究システムやFRシステムなどに実験的に利用されている。T-L 6 の文法，記述法などは TOSBAC のマニュアルを参照することにして，ここでは演算機能を考察する。

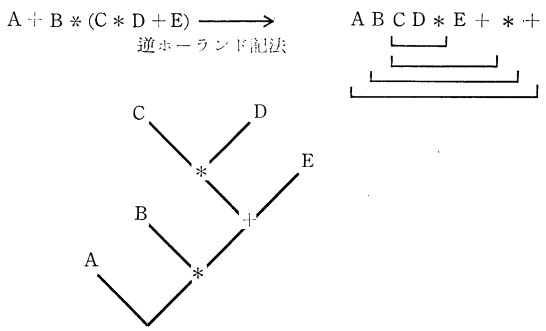


図10. 演算順序のリスト構造

5．データの構造

プログラミング上の概念として，ブロックと区分とポインターというものがある。ブロックはデータの集合のことであり記憶装置内部において論理的な語をひとつの単位とするものから構成される。ブロックを構成するデータが多種多様になってもよいので，データの性質にしたがってその長さも一定にはならない。そこで区分によって可変長のデータの長さを定義する。図11 のようにポインターはブロックの間につながりをつけて関係を与えたり，順序づけたりする。

6．L 6 の演算

L6 の演算は次のような命令文により与える。

(i) GET 命 令 文

(W, GT, 2)

この命令文によって，2語の長さのブロックをひとつとして記憶内部に確保され，ベースレジスターの中にそのブロックへのポインターが入れられる。

(ii) SET EQUAL 命 令 文

(STRNG 1, EH, COMPUTER)

これにより STRNG 1 という変数名の場所に右の文字列が登録される。

(iii) POINT 命 令 文

(WPQ, P, W)

今，仮に区分にポインターが入っていて他の区分にそれを入れるには，Pという命令によってWの指示するブロックとWPの指示するブロックとの指示するブロックの間に相互的な連結が可能になる。

L 6はこの他 IF～THEN といった条件文の組合せにより問題を記述できる。例として普通の算術式を逆ポーランド記法に変換するには，次のように記述すればよい。

データ（A＊((B＊C)＋(D＊E)))→A B C＊D E＊＋＊

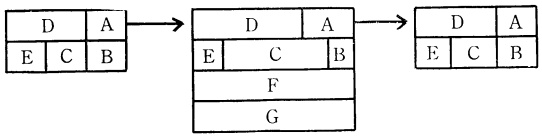


図11. L 6 の デ ー タ 構 造

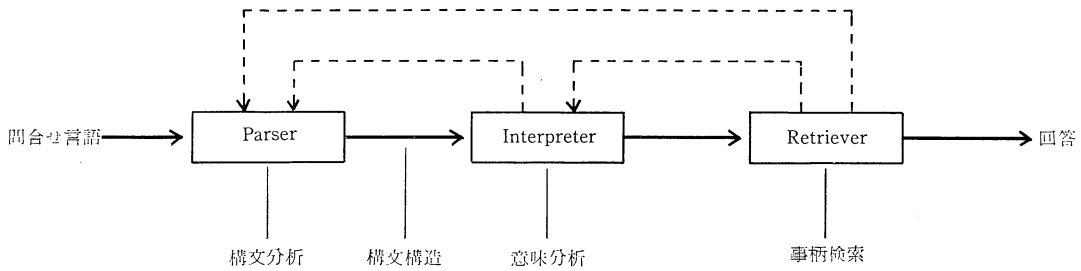


図12. Fact Retrieval システムの構成

```
INPUT (I, E, O) (I, IN, 1)
  IF (I, EH, '( ) INPUT
  IF (I, EH, ' ) NEXT
IF ANY (I, EH, +) (I, EH, −) (I, EH, ＊)
  (I, EH, / )
  THEN (O, GT, OP) (OS, E, I) INPUT
IF (I, EH ') ) THEN (I, PR, OS)
  (O, FR, OS) INPUT
  (I, PR, I) INPUT
```

この例で (I, IN, 1) の命令文は，カードより１字単位に文字列を読取りＩという区分に入れることを意味する。

Ｅ．その他の記号処理プログラム言語

記号処理プログラム言語は，特殊言語として SNOBOL, COMIT, LISP, L6 などが知られているが，そのほかに IPL-V[12] などがある。一方汎用言語といわれる COBOL に仕様段階であるが記号処理に適する文がある。また FORTRAN, ALGOL といった手続き向き言語にもリスト操作を目的として FLPL (FORTRAN List Processing Language), SLIP (Symmetric List Processing) などと称するプロセッサーが開発されている。さらに PL/1 などは，今後有力な記号処理の言語となるであろう。

Ｆ．記号処理プログラム言語の応用

記号処理の技術は，ＩＲシステムのプログラム技術，特に内部演算に応用される。

ＤＲシステム，ＦＲシステムなどにおいて質問文を自然言語そのものや，それに似たキーワードの組合せにより与えるシステムには，しばしば質問文の構文分析 (parsing) といった前処理がある。

１．構文分析

構文分析の技術は，プログラム言語そのものも編成翻訳などの段階で必要とする。最近では，N. Chomsky などにより発達された句構造文法 (phrase structure grammar)[13] の導入による研究が多い。

ＦＲシステムの多くは，質問文をシステムが準備する問合せ言語 (Query language) といった疑似自然言語により作文する。先ず，その文が文法に合っているかどうかを診断するプロセッサー (parser) を図 12 のように設定する[14]。

Condition :
IF formula { IS [NOT] GREATER THAN / IS [NOT] LESS THAN / IS [NOT] EAUAL TO / EQUALS / EXCEEDS } formula.

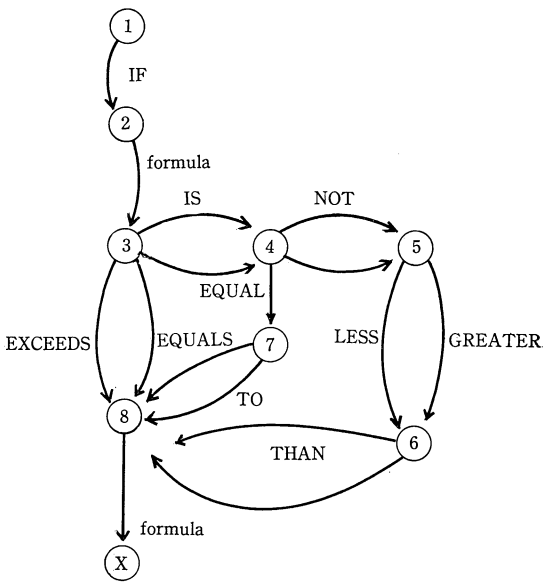


図13. 推移図式の例

2．プログラム言語の構文

自然言語より数段と形式化した人工言語であるプログラム言語は，COBOL などの構文分析のプロセッサー (parser) として図13に示すような推移図式 (transition diagram)[15] に描くことの出来る文法を作成しておく。構文分析とはこのような図式をたどっていくことである。

3．問合せ言語

問合せ言語の文法は，句構造になっているのが都合がよい。句構造とは文脈独立文法 (context free grammar) とも呼ばれ，次のように4組による定義による。

$G=(V_N, V_T, P, S)$

V_N: 非終端語彙

V_T: 終端語彙

P : 書変え規則

S : 初期文字列

この規則により質問文の生成過程は 図14 のような木構造になる。そこで，句構造文法は典型的な記号処理の演算が応用されることが解る。この種の問合せ言語は，QUERY[16] や ELIZA[17] BASEBALL[18] などが知られている。これを評価した最近の論文として R. F. Simmons[10] のものがよくまとまっている。

4．論理条件のアルゴリズム

問合せ文を構文分析した後，検索の本質である質問条件と内容の論理的な判断をする。

質問条件とは 図15 に示したような基本条件グラフのことである。この手順のアルゴリズムは，決定表 (Decision table) と呼び，判断とその結果行なうべき過程を2次元的な表の形で書いたもの[20]として利用される。例としてプログラム言語では次のように記述できる。

IF (AGE NOT LESS THAN 25 AND LESS THAN 35) AND HEALTH="GOOD" AND FEMALE, THEN COMPUTE RATE=1.57, COMPUTE PAYMENT=20000.

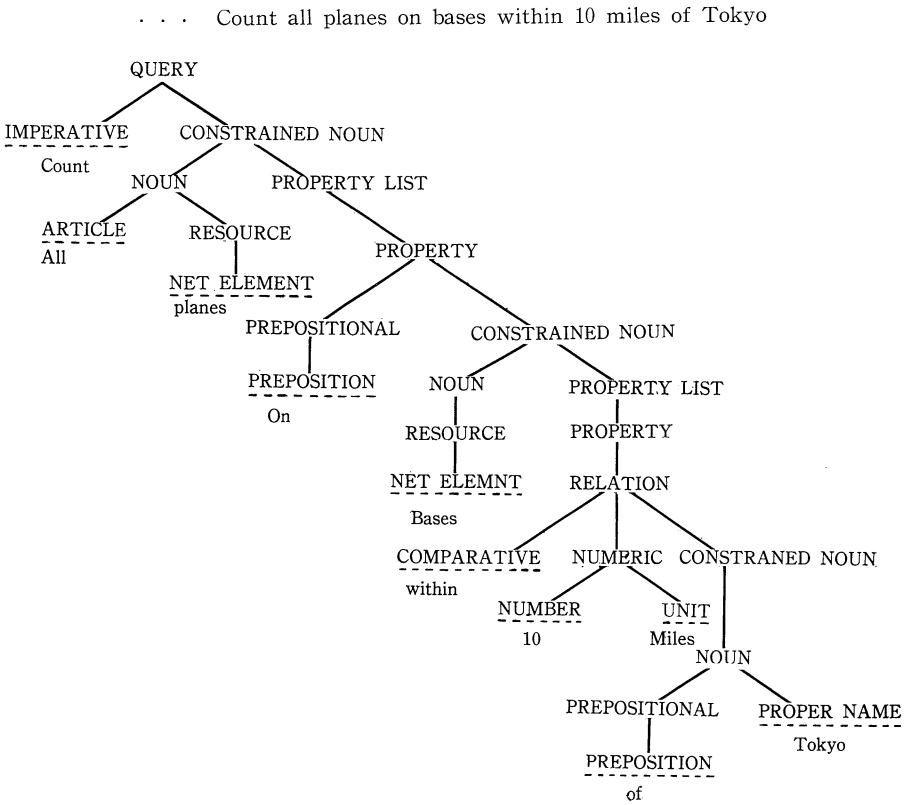


図14. 問 合 せ 言 語 の 構 造

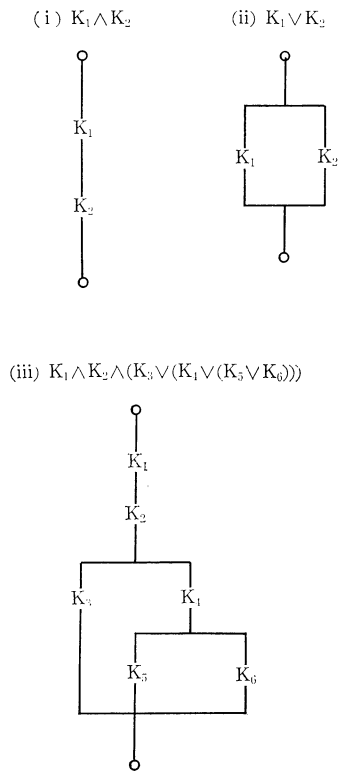


図15. 基本条件グラフ

（i） 論理句と値の規則

(0, 0, ∨) → 0
(0, 1, ∨) → 1
(1, 0, ∨) → 1
(1, 1, ∨) → 1
(0, 0, ∧) → 0
(0, 1, ∧) → 0
(1, 0, ∧) → 0
(1, 1, ∧) → 1
(0, —) → 1
(1, —) → 0

（ii） 逆ポーランド記法による論理演算

P ≡(1, —, 1, ∨, 1, 0, ∧, 1, ∨, ∧)
P1≡(0, 1, ∨, 1, 0, ∧, 1, ∨, ∧)
P2=(1, 1, 0, ∧, 1, ∨, ∧)
P3=(1, 0, 1, ∨, ∧)
P4=(1, 1, ∧)
P5=1

図16. 論理演算の手順

この種の記述に問合せ言語では出来るだけ簡単な記述法で，しかも複雑な条件を満足できる記述が必要となる。

5．内容検索

ＤＲシステムはＦＲシステムのような問合せ言語を考えなくてもよい。したがって複雑な構文を持った質問条件がないので高等な構文分析プロセッサー（parser）は必要としない。

一般にパラメータを固定の位置に記入したりキーワードとその論理的関係づけをするオペレータにより質問文を作文する。

ＤＲシステムの問合せ言語とは，次のように定義できる。

〈問合せ言語〉::=〈単位条件〉|〈問合せ言語〉|〈問合せ言

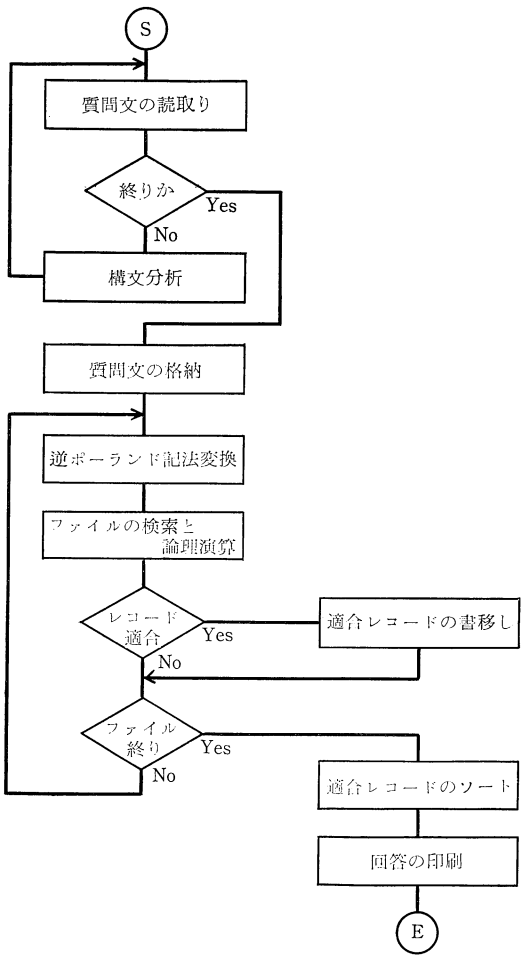


図17. 検索サブシステムの構成手順

語〉∧〈問合せ言語〉|〈問合せ言語〉∨〈問合せ言語〉
〈単位条件〉::=〈演算形式〉〈論理オペレータ〉〈演算形式〉
〈演算形式〉::=〈項目名〉|〈常数〉|〈変数名〉|〈演算形式〉
〈論理オペレータ〉::≡＊|＋|－

そこで，ＤＲシステムの質問文は，次のように書くことができる。

$((K_1+K_2+K_3) * (K_4+K_5))+K_6$

この構文はカッコ (parenthesis) を用いていくらでも複雑にすることができる。この解決として逆ポーランド記法へ変換した文字列をＰＤスタックを利用し 図16 のような論理演算をする。

6．ＤＲシステムの検索サブシステム

これまでのプログラム技術の応用により検索サブシステムは，図17 のように構成される。

(ｉ) 問合せ言語による質問文の読取り
(ii) 質問文の構文分析
(iii) 質問文の逆ポーランド記法への変換
(iv) ファイルの検索と内容の検索
(ｖ) 項目の値と質問条件の値の演算
(vi) 適合レコードの外部記憶装置への書移し
(vii) 適合レコードの受付順のソート
(viii) 回答の印刷

V. ファイル処理のプログラム言語

A．ファイル管理システム

記号処理のプログラム言語が，内部演算を中心にした機能から構成するに対し，外部記憶装置の管理とか大量のデータの操作を目的とするファイル処理のプログラム言語をファイル管理システム (File management system) またはデータベース管理システム (Data base management system) と呼ぶ。ファイル処理の対象は，次の４つとされる。[21]

(ｉ) ファイルの生成 (file creation)
(ii) ファイルの更新 (file maintenance)
(iii) ファイルの検索 (file search)
(iv) レポートの作成 (report generation)

ファイル処理の言語として，これまでにＣＤＣ社の INFOL[22] などが磁気テープを蓄積媒体とし，ＩＢＭ社の GIS[23] や GE社の IDS[24] やさらに ADM, MARK-IV[25] などが磁気ディスクを媒体とし開発され知られている。

B．プログラムの機能

ファイル処理のプログラムは，主に外部記憶装置上に蓄積したファイルのレコード項目の照合演算を中心機能とする。項目はデータとして記憶装置に格納されたりする。

1．データの格納

主記憶装置にデータを入れることを格納と呼び，外部記憶装置にレコードを書込むことを蓄積と呼ぶ。一般にデータを格納するには，次の２つの方法が考えられる。

(ｉ) 固定長記憶 (fixed length) と可変長記憶 (variable length) 格納するデータがすべて同じ大きさの記憶場所を必要とするか，またはそれぞれのデータが異なった大きさの記憶場所を必要とするか，またはそれぞれのデータが異なった大きさの記憶場所を必要とするか，という方法。
(ii) 最大長による可変長記憶 格納するデータがそれぞれ異なった大きさの記憶場所を必要とするので全て可変長である場合は，データの最大の大きさの記憶場所を準備する方法。

2．データの探索

データの主記憶装置上における格納形式により探索の方法は異ってくる。データの格納方法により探索の方法は，次のようになる。

(ｉ) 固定長データの探索 各データの記憶場所の大きさは一定であるから表引き (table search) が利用される。
(ii) 可変長データの探索 各データの記憶場所の大きさが異なるので表引きは利用できない。そこで，各データに分離記号やデータのサイズを入れておくことやポインターを用いる。

C．ファイルの蓄積

ファイルの蓄積とは，情報ファイルを磁気テープや磁気ドラム，磁気ディスク装置などの外部記憶装置に入れておくことを言う。

蓄積において重要なことは，ファイルの構造に関する技術である。ファイルは項目の値をある一定順序に配列したレコードにより構成される。項目の値とは，文献などでその属性となる著者名，出版社，出版年，著名といったデータのことである。

ファイルの構造は磁気テープへの蓄積のように物理的な位置によって順序関係の決まる線型構造と，磁気ディスクや磁気ドラムなどの乱処理記憶装置などにより実現する木構造とかチェーン構造などがある。

1．線型構造のファイル

この構造をもつファイルは一般に逐次ファイルと呼ばれて，逐次探索を目的とする。

線型構造をもつファイルの特色は，レコードをグループ化して蓄積できることが挙げられる。これをブロッキング技法と呼び，グループ化したレコードを物理レコードとも呼ぶ。ブロッキング技法により一度に主記憶装置内部に呼べる単位が自由に決められ逐次処理が早い。しかし，反面ファイルの更新において追加，削除といった場合に全てのファイルの書変が必要となる。この欠点を除くためにポインターの利用による改善が考えられる。この方法はファイルの更新などで追加があった場合は，適当な空き場所を見つけてそこに蓄積しておいて，もとのレコードのポインターの内容だけを修正してやる方法である。この欠点はブロッキング技法が利用できないことである。

2．木構造のファイル

ポインターによって次のレコードの蓄積場所を知るといった方法を利用すれば，物理的な位置に関係なく順序関係の表現ができる。

木構造またはリスト構造さらにその応用であるチェーン構造（図18）のファイルは磁気ディスク，ドラムなど乱呼出し処理装置に展開する。

D．ファイルの探索

ファイルの探索は，質問により作成した検索条件に従って，まずファイルを選び出す。

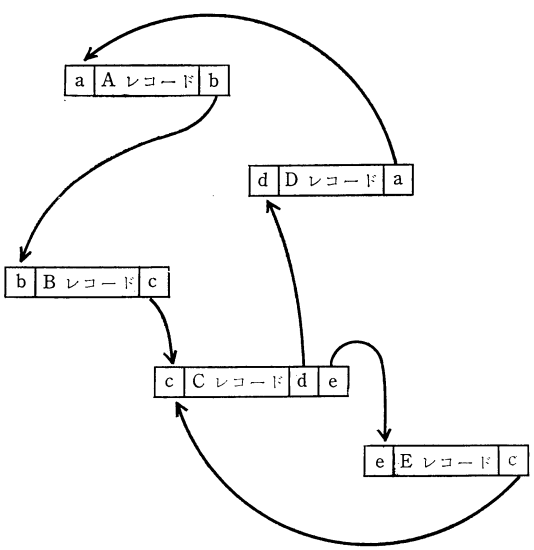


図18．チ　ェ　ー　ン　構　造

次にそのファイルの全部のレコードや質問に関連するレコードをひとつひとつ指定の項目についてその内容を調べ，該当するレコードを取出す。

1．ファイルの検索条件

ファイルの検索条件の内容としては，次のような指定が必要とされる。

(i)　どのファイルを検索するか，といったファイルの蓄積構造の指定

(ii)　各レコードとどの項目を照合するかの指定

(iii)　その項目の内容の指定

(iv)　照合の対象となった項目が照合した時に回答として取出すべき他の項目の指定

2．構造検索のパラメータ

ファイルの蓄積構造に対応して決るものであって，どのファイルを検索するかといった指定の処理である。構造検索は記憶装置における線型構造，木構造，チェーン構造などの蓄積構造の定義により一意的に決定される。検索パラメータとして 図19 に示すようなものを準備する。

E．ファイル処理言語の構成

ファイル処理の言語は，最近ではオンラインを目的とする汎用システムとして COBOL や PL/1 に近い高水準の機能をもったものとして開発されている。概して，この言語のシステム構成は必要とする基本サブルーティンやファイルの入出力管理をするサブシステムなどにより構成される。

プログラム言語的には，次のような区分から構成される。

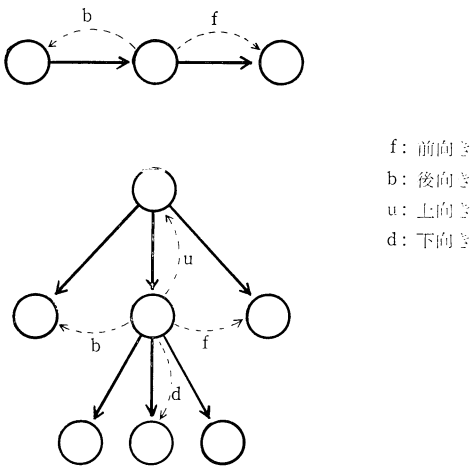


図19．検索パラメータの動き

(i) ファイルの定義 (file description)
(ii) レコードの定義 (record definition)
(iii) 項目の定義 (item definition)
(iv) 構造の定義 (structure definition)
(v) 検索条件の指定 (retrieval criteria)
(vi) 出力の編集指定 (report description)

F．チェーン構造ファイルの言語

ファイル処理言語の中で自由にファイル構造を記述できるもので優れているのは，IDS (Integrated Data Store) である[26]。IDS の特徴は，情報主題の関連づけ，蓄積，消込，検索などを磁気ディスク上のファイルに展開することである。言語的には，IDS は COBOL を拡張したものであるので COBOL の利用者のシステム設計に従って簡単に記述できる。

IDS の主な機能を整理してみると，次のようになる。

(i) データを主題別，索引別などと有機的に結びつけて蓄積でき，検索にあたっては関連あるデータを芋づる方式に探索をする。これはリスト検索である。
(ii) データの重複を避けるようにして磁気ディスクの蓄積効率をよくする。
(iii) 乱呼出し検索や逐次検索のどちらも利用できる。
(iv) 磁気ディスク上のデータは，ページという思想で蓄積と検索がされるので主記憶装置との転送回数が著しく減少する。
(v) IDS 言語は，まったく新しいものでなくて COBOL をホスト言語としている。

1．レコードとチェーン

IDS のレコードは他のレコードを結びつけるために前後にチェーンフィールドと称するポインターを持つ。

2．ページ

IDS は情報ファイルの蓄積に磁気ディスクを利用する。そこで主記憶装置のやりとりは，簡単なマクロ命令によりページ方式により遂行される。

G．ＩＤＳの言語

IDS は COBOL をホスト言語とする。そこで COBOL と同様に次の部門から構成される。

IDENTIFICATION DIVISION
ENVIRONMENT DIVISION
DATA DIVISION
PROCEDURE DIVISION

この中で特に DATA DIVISION の記述に IDS SECTION という特別の記述が加えられる。

IDS SECTION は IDS ファイルの定義するものである。

1．ＩＤＳファイルの定義

磁気ディスク上のファイルのサイズを指定するには，次のように記述する。

```
MD IDSFILE; PAGE CONTAINS
                    1000 CHARACTERS
          ; FILE CONTAINS 5000 PAGES.
```

これは IDSFILE という名前の IDS ファイルの1ページの大きさを1000文字として，ファイルは5000ページの大きさとする意味である。

2．ＩＤＳレコードの定義

レコードの定義は各種の自由な形式にファイルを構成する時にもっとも重要な指定である。

```
; RETRIEVAL VIA { フィールド名 FIELD
              { チェーン名   CHAIN
              { CALC CHAIN
```

これはレコードの検索する方法を示す。

3．ＩＤＳのフィールドの定義

フィールドとは，項目のことでありその定義は，レコード中の各項目の内容を記述する。

```
02  AUT-1   PICTURE   ×(20).
02  TITL-1  PICTURE   ×(180).
02  BIB-1   PICTURE   ×(16).
02  IND-1   PICTURE   ×(8).
02  FILLER  PICTURE   9999.
```

この例は，それぞれの項目が20，180･･･文字の長さである文献レコードの記述である。

4．ＩＤＳのチェーンの定義

チェーンの定義はファイルへのレコードの挿入方法を指定するもので，チェーン中のレコードの親子関係つまり階層関係を表現する。

```
98  KEYWORD-1  CHAIN  MASTER
  ; CHAIN—ORDER IS SORTED
  ; LINKED TO PRIOR.
```

これは KEYWORD-1 という名前のチェーンの親レ

コードでありポインターが前後にあることを意味する。

5．ＩＤＳのプログラミング

IDS のプログラミングは図20のようなファイル構造の想定図にしたがって行なう。例として次のようなレコードがあるとする。

(i) カテゴリレコード：分類番号，分類名

(ii) 文献レコード：標題著者名，日付

(iii) 所属組織レコード：所属コード

(iv) 索引レコード：キーワード

これらのレコードを 図20 にしたがって関係づけるには，次のように記述する。

```
01  CATEGORY-REC;  TYPE IS 10
        RETRIEVAL VIA CALC CHAIN.
    02  UDC PICTURE 9(6).
    02  CLSS PICTURE ×(22).
    98  CALC CHAIN DETAIL RANDOMIZE ON
        UDC.
    98  AB  CHAIN  MASTER,  CHAIN—ORDER
        SORTED.

01  CORPORATE—RED;  TYPE IS 20
        RETRIEVAL VIA CALC CHAIN.
    02  COR-NO PICTURE  9(4).
    02  COR-NAME PICTURE  ×(30).
    98  CALC CHAIN DETAIL RANDOMIZE ON
        COR-NO.
    98  CB  CHAIN  MASTER.  CHAIN-ORDER
        SORTED.

01  DOCUMENT-REC;  TYPE IS 30
        RETRIEVIL VIA AB CHAIN.
```

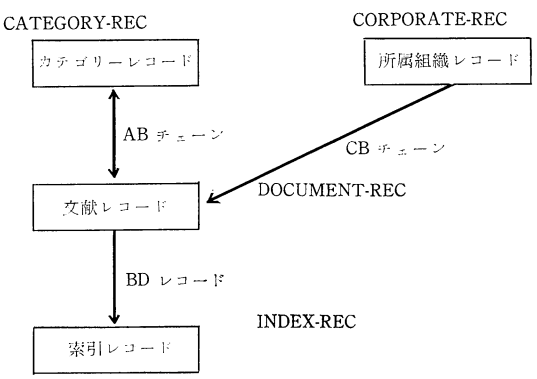


図 20．文献ファイルのチェーン関係図

```
    02  TITL-1   PICTURE   ×(60)
    02  AUT-1    PICTURE   ×(28)
    02  DAT-1    PICTURE   9(6).
    02  DOC-NO   PICTURE   9(6).
    98  AB  CHAIN  DETAIL,  SELECT  UNIQUE
        MASTER,  MATCH-KEY  UDC,  LINKED
        TO MASTER.
    98  CB  CHAIN  DETAIL,  SELECT  UNIQUE
        MASTER, MATCH-KEY COR-NO.
    98  BD  CHAIN  MASTER,  CHAIN-ORDER
        LAST.

01  INDEX-REC;  TYPE IS 40
        RETRIEVAL VIA BD CHAIN.
    02  KEY-1  PICTURE   ×(20).
    98  BD  CHAIN  DETAIL,  SELECT  CURRENT
        MASTER.
```

ファイルの定義に対し，処理手順の記述は，例として図21 のようなシソーラスファイルの 検索を 次のように記述する。

```
MOVE INQ-WORD TO T-KEYWORD.
ENTER IDS;  RETRIEVE WORD-REC,
            IF ERROR GO TO ER 2.
RN.
ENTER IDS;  RETRIEVE NEXT TO A-CHAIN,
            IF WORD-REC GO TO R-RTN,
            IF ERROR GO TO ER 3;
            MOVE  TO  WOR  KING-STO-
              RAGE,
            IF ERROR GO TO ER 4.
IF W-CLASS NOT EQUAL TO "SYN" GO
  TO RN.
```

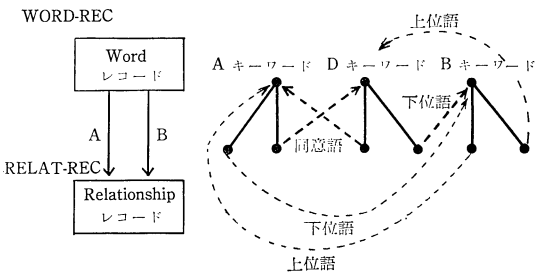


図 21．シソーラスファイルのチェーンとキーワードの関係

```
ENTER IDS; RETRIEVE MASTRE OF B-
           CHAIN,
             IF ERROR GO TO ER 5;
             MOVE TO WORKING-STORAGE,
             IF ERROR GO TO ER 4.
MOVE T-KEYWORD TO KEYW (×)
ADD 1 TO ×.
GO TO RN.
R-RTN. MOVE 1 TO ×
          ⋮
```

ファイル処理の言語は，IDS のように COBOL を基礎とした拡張言語以外に GIS のように専用言語といったものが使われている。残念ながら著者は GIS を使う経験がなかったので本稿において触れなかった。

結　　論

本稿は，自然言語を処理する人工言語といった関係であるプログラム言語について，ＩＲシステムの設計と開発に影響を与える技術的側面を中心に考察した。結論として挙げられることは，ＩＲアプリケーションの観点において抽出される技術とは，内部演算として記号処理のアルゴリズムとその演算があり，外部記憶の入出力の制御やファイルの管理といったファイル処理がある。

現在，プログラム言語として，記号処理とファイル処理の両方を十分に満足できるように設計されたものはない。ある程度の機能が準備されたものに PL/1 や SNOBOL-4 などがあるが，まだまだ多くのコンピュータシステムに実現していないので当分の間日常語にはなりそうもない。

ＤＲシステムに限って言えば，これまで比較的にファイルやレコードの構造も固定しており検索の論理や事後処理といった要求もプログラム的には単純なものが多い。

MEDLARS や CAS などの大規模なＤＲシステムも専用プログラムシステムであって，プログラム言語とは見なされない。しかし専用であるが故にシステムの運営が効率よく，しかも容易に使用できるという事実は見逃せない。

ただし，ＩＲシステムがハードウエアシステムとソフトウエアシステムの両方の面でつねに流動的に発展していくという観点で見れば，汎用性のある言語としてないことは拡張性などの要求を満すのに不完全なシステムとなる。

今後，ＩＲシステムはソフトウエア的には統合化されたファイルを自由に処理できる高水準のプログラム言語システムの開発と，各種の必要とするアルゴリズムを体系化したりさらに標準化することによるプログラミングの容易性の研究が必要と思われる。

引 用 文 献

1) Naur, P. "Revised report on the the algorithmic language ALGOL 60," *Comm. CAM*, vol. 6, 1963, p. 1-17.
2) Cheatham, T. E. & Sattley, K. "Syntax directed compiling," *Proc. AFIPS*, 1964.
3) Yngve, V. *COMIT; Programmer's reference manual*. MIT Press, 1961.
4) Farber, B. J. "SNOBOL," *J. ACM*. 1961, p. 21-27.
5) Griswold, R. E. & Poage, J. F. *The SNOBOL programming language*. Prentice-Hall, 1968.
6) 東京芝浦電気 TOSBAC-3400 SNOBOL 解説書 TOSBAC アプリケーションライブラリー No. 34AP425C 1970.
7) 大駒誠一. "記号処理言語 COBOL" *Bit*, vol. 3, 1970, p. 73-78.
8) McCarthy, J., et al. *LISP 1.5; programmers manual*. MIT Press, 1962.
9) Knowlton, K. C. "A programmer's description of L 6," *Comm. ACM*, vol. 9, p. 616-625.
10) 藤野精一. リスト処理と言語解析. 東京, 朝倉書店, 1970.
11) 東京芝浦電気. TOSBAC-3400 T-L6 解説書 TOSBAC アプリケーションライブラリー No. 34AP426C 1970.
12) Newell, A. & Tonge, F. "An introduction to information processing language V," *Comm. ACM*, vol. 3, 1960, p. 205-210.
13) Chomsky, N. *Syntactic structure*. The Hague, 1957.
14) Salton, G. Automated language processing 〈Schecter, G. ed. Information retrieval: A critical view〉 Washington, Thompson, 1967.
15) Conway, M. E. "Design of a separable transition diagram Compiler," *Comm. ACM*, vol. 6, 1967, p. 396-408.
16) Cheatham, T. E. "Translation of retrieval requests couched in a semiformal English-like language," *Comm. ACM*, vol. 3, 1962, p. 34-39.
17) Weizenbaum, J. "ELIZA — a computer program for the study of natural languag com-

munications between man and machine," *Comm. ACM*, vol. 9, 1966, p. 36-45.
18) Kellogg, C. H. Designing artificial languages for information storage and retrieval 〈Schecter, G. ed., Information retrieval: a critical view. Washington, Thompson, 1967〉
19) Simmons, R. F. "Natural language question-answering systems: 1969," *Comm. ACM*, vol. 13, 1970, p. 15-30.
20) Sable, J. D. *Data management system study: final report*. Auerbach Corp., 1968.
21) 小林功武等. データベースの管理/CODASYL REPORT を中心に企画センター, 1970.
22) Control Data Corporation. *3600/3800 INFOL reference manual*. No. 60170300 1966
23) IBM system 360 — Generalized information system application description manual. 1968.
24) 東京芝浦電気 TOSBAC-3400 IDS 解説書 TOSBAC アプリケーションライブラリ・ No. 34AP410C 1970.
25) Reinawald, L. T. and Soland, R. M. "Conversion of limited-entry decision tables to optimal computer programs II: mininum storage requirement," *J. ACM*, vol. 14, 1967, p. 742-755.
26) 東京芝浦電気 IDS と ALISS を中心にした情報の蓄積と検索システム TOSBAC アプリケーションライブラリー No. ＊＊AP418C 1970.

参 考 文 献

1) Bobrow, D. "A comparison of list processing languages," *Comm. ACM*, vol. 1964, p. 37-48.
2) Floyd, R. W. "A descriptive language for symbol manipulation," *J. ACM* vol. 8, 1961, p. 579-582.
3) Garvin, P. L. ed., *Natural language and the computer*. New York, McGraw-Hill, 1963.
4) 日本情報処理開発センター. 新しい検索技術と検索システムに関する基礎理論の体系化. 1969.
5) Rosen, S. ed., *Programming systems and languages*. New York, McGraw-Hill, 1967.
6) 斉藤 孝. 電算機による情報の蓄積と検索システムの設計 — ALISSの開発と応用 — 慶応義塾大学修士論文, 1969.
7) 斉藤 孝 ファイル作成・検索プログラム ドクメンテーションコンピュータコース テキスト 1970.
8) Schwarcz, R. M. "A deductive question answer for natural language inference," *Comm. ACM*, vol. 13, 1970, p. 167-183.
9) 竹下 章. 〝汎用ファイル処理システムの性格とその発展,〟 **情報処理**, vol. 10, 1969, p. 84-93.
10) 和田 英一. 〝プログラム言語の発展,〟 **情報処理**, vol. 11, 1970, p. 313-322.