LR(k) アナライザ および マイクロプログラム記述言語 に関する研究

1975 年 2月

菊野 亨

内　容　梗　概

本論文は筆者が大阪大学大学院基礎工学研究科（物理系専攻）の学生として，當研究室において行なった情報処理に関する研究のうち，LR(K)アナライザおよびマイクロプログラム記述言語に関する研究を二編にまとめたものである．

緒論および各章の第1節では研究の現状，工学上の意義，本研究の目的について概説している．各章の最後の節および結論では，本研究で得られたおもな結果と残された問題についてまとめてある．

第1編第1章では，コンパイラの語い解析部におけるトークン間の区切りの問題について考察している．原始プログラムのトークンの順はLR(K)文法で記述され，各トークンを構成する文字列は一般に正規集合であるとし，トークンの長さでん個の先読みで区切りがつくかどうかはんを指定すれば判定できるが，んを与えなければそれは判定不能であることを示している．さらに，区切りの可能性を一時記憶するレジスタ数最小のスキャナの構成法を示し，決定性言語との関係などについても論じている．

第1編第2章では，コンパイラの構文解析部のモデルとして，陽に文法に依存しない，先読みをもつアナライザを導入した．2.3節では，与えられたアナライザと等価で，先読み最小のアナライザを求める方法を示している．また，2.4節では，アナライザの簡単化の方法について述べている．いわゆるパーザで考えるより，本論文のアナライザで考えるほうが簡単化に関しては一般に有利である．

第2編では，マイクロプログラム方式の小型電子計算機UXを対象として設計したフローチャート型のマイクロプログラム記述言語FMLの仕様とそのトランスレータの作成法について述べている．

LR(K)アナライザおよびマイクロプログラム記述言語に関する研究

目　次

緒　論

コンパイラにおける語い解析部のおもな目的は，与えられた原始プログラム（数字や英字，特殊文字などの"文字"の系列）を読み，それを整数，変数名などといった基本的なカテゴリに属するように区切り，このように区切られた実際の文字系列をシンボルテーブルなどに格納して，原始プログラムをトークン（カテゴリの名前）とパラメータ（シンボルテーブルへのポインタなど）の対の系列に変換することである．構文解析部では，語い解析を終えたプログラムを読み，トークンの系列にもとづいてその構文（句構造）を決めつつ，わたすべきパラメータを定めて，必要なときにシンボルテーブルの処理や内部コードを発生するためのセマンチックルーチンを呼ぶ．

本論文の第1編第1章は，発表論文(1)～(4)において公表したものをまとめたもので，文脈を利用した語い解析の方法について述べている．

よく引かれる例であるが，フォートランの　DO　250　I＝…は文脈（すなわち，トークンの並び方の情報）を利用しなければ，トークン名とその区切りの位置を決定できない．＝のつぎがたとえば13，…であれば，はじめのDOがdo文を示すキーワードであるが，13・…ならDO250Iが実数形の変数名である．フォートランでは空白は特別な場合以外は意味がなく，普通，文中では無視される．

そこで本論文では，このフォートランの例を含むように，つぎに来うる長さkのトークン名の系列を構文解析部（KnuthのLR(k)パーザを考える．）から教えてもらい，それを利用して文字列をトークンごとに区切るという方法について考察している．kを長くすれば，あいまいさなく区切れる可能性がふえるが，一般には，k＋1個先のトークンまで一致してもあいまいさがおこりうる．そのため，k＋1個先のトークンまで見るならば，その一番目のトークンの区切りに関するあいまいさがなくなるという条件をつけ，この条件がなりたつときk－区分可能であるという．

本論文では，kを指定すればk－区分可能かどうかは判定できるが，

kを与えなければそれは判定不能であることを示し，さらに，区切りの可能性を一時記憶するレジスタの数が最小のスキャナの構成法，このスキャナとLR(k)パーザのモデルで受理される言語が決定性言語になることそのほかについて述べている．

第1編第2章では，発表論文(5)～(7)において公表したアナライザの等価性と簡単化に関する研究を述べている．

通常，パーザは，特定の文法Gに対し，与えられた入力系列がそのGで導出されるかどうか判定し，もし導出されるならその導出木を出力する．ところが，内部コード発生などの意味づけの処理をするとき，固定したもとの文法Gにおける導出木そのものは必ずしも必要でない．とくに，右辺がただ1つの非終端記号からなるようなルール（たとえば，〈term〉→〈factor〉など）に対しては意味づけの処理を行なわないことが多い．

そこで，本論文では，構文解析部の本質的な機能を，わたすべきパラメータを定めてセマンチックルーチンを正しく呼ぶことであるとし，この機能を果すものをアナライザと呼ぶ．アナライザはその内部で特定の文法は用いていない．アナライザのモデルとして，一定量までの先読みを許した決定性プッシュダウンオートマトンを考える．アナライザはパラメータと制御記号を格納するためのプッシュダウンスタック(pds)をもつ．アナライザは動作を決定するために必要ならトークンを先読みし，入力のトークンに付随したパラメータをpdsにスタックしたり，pdsの頭からn(nは1以上の整数)コマの内容をとり出し，それらのパラメータ部分をパラメータとしてあるセマンチックルーチンpを呼ぶ．呼ばれたルーチンpはわたされたn個のパラメータに依存して新しいパラメータ(中間結果を格納する場所を指し示すポインタなど)を1つ求め，これらを用いて内部コードを発生するとともに，新しいパラメータをpdsの頭のコマに返す．

二つのアナライザは同じ入力系列に対して，同じパラメータで同じセマンチックルーチンをつぎつぎと呼ぶとき，等価と定義する．本論文では，任意のアナライザが与えられたとき，それと等価なアナライザのう

ちで，先読みの長さがつねに最小のアナライザを求める方法，さらに，先読みが最小のものの中でもっとも簡単な（状態数が最小の）アナライザを求める方法などについて述べている．いわゆるパーザで考えるより，アナライザで考えるほうが簡単化に関しては一般に有利である．

Wilkes は“電子計算機の制御部（ハードウェア）を組織的に設計する手段”としてマイクロプログラミングの概念を導入した．最近では，書き換え可能な制御記憶が使えるようになり，オペレーティングシステムやコンパイラの一部のマイクロプログラム化（いわゆるファームウェア化）や，高級言語で書かれたステートメントを直接実行する計算機（高級言語計算機）に対する関心が急速に高まっている．それとともに，ユーザにマイクロプログラムを開放することも試みられてきた．しかし，マイクロプログラム記述言語として提供されているものは，通常，ハードウェアにかなり依存しており，マイクロプログラムを書くには，ハードウェアの細部に関するかなりの知識を必要とする．したがって，マイクロプログラム記述用の高級言語の開発が望まれている．

本論文第2編では，発表論文 (8) ～ (10) として公表したマイクロプログラム記述言語とそのトランスレータに関する研究を述べている．本学情報工学科にある小型電子計算機 UX は16ビット長の垂直型のマイクロ命令を備え，書き換え可能な4K語の制御記憶をもったマイクロプログラム方式の計算機である．この UX を対象として，フローチャート型のマイクロプログラム記述言語 FML を設計し，そのトランスレータをコーディング中である．本論文では，FML の仕様とそのトランスレータの作成法について述べ，さらに，そこでのいくつかの問題についても論じている．この言語のおもな特長は「ハードウェアに固有の制限事項をなるべく隠し，ユーザが覚えやすく，使いやすい」ことである．マイクロプログラムの効率は重要であるので，トランスレータでは種々の最適化を試みている．トランスレータは PL/I で書かれており，中型計算機 FACOM 230-45S で実行される．

関連発表論文

(1) 近藤，菊野，谷口，嵩
"語い解析と構文解析について"，電子通信学会オートマトンと言語研究会資料（昭和47年7月）．

(2) 菊野，谷口，嵩
"語い解析に関する一考察——一方向有限オートマトンによる正規集合間の区切りについて——"，電気関係学会関西支部連合大会資料（昭和47年10月）．

(3) 谷口，菊野，嵩
"文脈を利用した語い解析"，電子通信学会オートマトンと言語研究会資料（昭和48年6月）．

(4) 谷口，嵩，菊野
"文脈を利用した語い解析"，電子通信学会論文誌，57－D巻4号（昭和49年4月）．

(5) 谷口，菊野，嵩
"構文解析の簡単化について"，電子通信学会オートマトン研究会資料（昭和47年1月）．

(6) 菊野
"ANALYZERの簡単化"，京都大学数理解析研究所「情報科学の数学的理論」研究会資料（昭和47年2月）．

(7) 谷口，嵩，菊野
"アナライザの等価性と簡単化"，電子通信学会論文誌，56－D巻7号（昭和48年7月）．

(8) 菊野，杉山，安積，谷口，嵩
"あるマイクロプログラム記述言語FMLの設計"，電子通信学会全国大会資料（昭和49年7月）．

(9) 杉山，安積，菊野，谷口，嵩
"FMLトランスレータにおけるマイクロ命令の最適化について"，電子通信学会全国大会資料（昭和49年7月）．

(10) 菊野，杉山，安積，谷口
"あるマイクロプログラム記述言語 FML とそのトランスレータについて"，情報処理学会論文誌（投稿中）．

第1編　LR(K)アナライザ

第1章　文脈を利用した語い解析

§1.1　序言

コンパイラにおける語い解析部の目的は，構文解析その他あとの処理がしやすいように，与えられた原始プログラム（数字や英字，特殊文字などの"文字"の系列）を読み，それを整数，変数名などといった基本的なカテゴリに属するように区切り，このように区切られた実際の文字系列をしかるべきテーブルなどに格納して，原始プログラムをトークン（token，カテゴリの名前）とパラメータ（テーブルへのポインタなど）の対の系列に変換することである．構文解析部では，語い解析を終えたプログラムを読み，トークンの系列にもとづいてその構文を決めつつ，必要なときに内部コードを発生するためのセマンティックルーチンを呼ぶ．

本論文では，語い解析における各トークン間の区切りの問題を論じる．テーブルに関する処理，たとえば二重定義の有無の判定や，登録などは各トークンに区切ったあと別のルーチンが行うものとし，ここでは問題にしない．

現実のコンパイラでは，整数や変数名などを表わす文字系列の長さには制限がつけられているのが普通であるが，本論文では各トークンを表わす文字列あるいはその集合は一般に無限の正規集合であると考える．たとえば，〈変数〉は英字ではじまる英数字の系列全体とする．この方が言語として柔軟性があるし，また，有界性を仮定して議論してもその異なる要素の数は一般に大きく，アルゴリズムなどの手数はぼう大となる可能性がある．

正しい原始プログラムにおけるトークンの現われる順はいわゆる文法で記述される．ここでは，線形時間で解析できるようにLR(K)文法[1]を考える．文字の集合をΣ，トークンの集合をV_T，トークンの並び方の順を定めるLR(k)文法を$G=(V_N, V_T, P, S)$とする．fを

V_T の各元に Σ 上の一つの正規集合を対応させる写像とする．G の各終端記号がトークンで，これはまた正規集合の名前でもある．

文字系列をトークンごとに区切ろうとするとき，原始プログラムにおけるトークンの並び方の情報をまったく利用せず（利用した場合の $L(G) = V_T^*$ のときに相当），任意に与えられた $w \in \Sigma^*$ を一方向有限オートマトンでスキャンし，文字数で有限の遅れを許し（いいかえれば，有限個の文字を先読みし），つぎつぎと区切りの位置とそのトークン名を決定していくモデルについては，文献(5)で発表した．このモデルでは，語い解析の処理を構文解析から切り離して独立に行えるが，あいまいさなく区切れるための条件が非常に厳しくなる．

よく引かれる例であるが，フォートランの DO 250 I = … はトークンの並び方の情報 ── すなわち，文脈 ── を利用しなければ，トークン名とその区切りの位置を決定できない．= のつぎがたとえば 13，… であれば，はじめの DO が do 文を示すキーワードであるが，13．… なら DO 250 I が実数形の変数名である．フォートランでは空白は特別な場合以外は意味がなく，普通，文中では無視される．

処理中の各時点について，すべてのトークンではなく G の正しいセンテンスとしてつぎに来うるトークンについてだけ考えるならば，あいまいさなく区切りがつく可能性がふえる．そこで，つぎに来うるトークン名（の系列）をパーザから教えてもらい，それを利用して文字列のトークンを区切る方法が考えられる．文献(4)では，このように，つぎに来うるトークン名の情報を利用し，有限オートマトンにより文字数で有限の遅れ（ℓ 個まで）を許し，区切りとトークン名があいまいさなく一意に決められるための条件や，それを満たすときのスキャナとパーザの構成法を述べている．しかし，上のフォートランの例 DO 250 I = 13，… については，整数や変数名にいくらでも長い系列を許すなら 250 I = 13，の部分が長くなり，文字数が有限の遅れでは DO で ⟨do⟩ であることを見出すことはできない．

そこで本論文では，上のフォートランの例を含むように，"遅れ"を"文字列の長さで ℓ 個'ではなく，"トークンの系列で長さん個"とし，

n+1個先のトークンまで一致すればその一番目のトークンを取り出すという方法を考える.

たとえば, n=5とし上の例で, 簡単のため, (イ) <do><整数><整数形変数><等号><整数><コンマ> と (ロ) <実数形変数><等号><整数><小数点> のニとおりの可能性しかないとすると, DO 250 I = 13, … が与えられて, 文字Dを見た時点で <do> の途中と <実数形変数> またはその途中の可能性があり, DOまで見た時点ではDOで <do> と <実数形変数> またはその途中の可能性がある (Dのみで <実数形変数> というのはなくなる). この例では, <do> か <実数形変数> かを決めるのに, 途中におけるあいまいさはつねに2以下であり, コンマを見たときそこまでが(イ)と一致し, また(ロ)でないことがわかるので, DOで <do> と決定される.

nを長くとれば, あいまいさなく区切れる可能性がふえるが, 一般には, n+1個のトークンの系列に一致してもあいまいさがおこりうる. そこで, 本論文では, n+1個先のトークンまで見るならば, その一番目のトークンの区切りに関するあいまいさがなくなるという条件をつける. この条件をみたすとき, Gとfはn-区分可能であるということにする.

本論文の1.3節では, パーザとスキャナのモデルについて詳述し, 1.4.1でn-区分可能性の定義を正確に述べる. スキャナはパーザからつぎに来うるトークンとして長さn+1のトークン系列の集合をパラメータとして受けとり, この情報を利用してつぎのはじめのトークン名とその区切りを決定する. スキャナは最初のトークンの区切りの可能性があるごとに, レジスタにそのトークン名と区切りの位置をセットし, 可能性がなくなればそれをクリアする. 1.4.2で, G, f がn-区分可能なとき, そのようなレジスタの個数が最小ですむようなスキャナの構成法を示している.

nが指定されれば, G, fがn-区分可能かどうか判定できるが, nが指定されなければその問題は決定不能であることを1.4.3で示す.

1.5節では, n-区分可能なG, fで生成される言語は, 実は決定性

言語になることを述べるが，これを，同じ文字系列の部分を繰返してスキャンしないような一方向のスキャナを構成して示している．

1.6節では，スキャナの等価性，および簡単化に関連して，残された問題などについてふれている．

§1.2 諸定義

文脈自由文法（CFG）を $G=(V_N, V_T, P, S)$ で表わす．V_N，V_T は，それぞれ，G の非終端記号，終端記号の集合，P は書換え規則（ルール）の集合，S は始記号である．$V=V_N \cup V_T$ とおく．一般性を失わず，ルールの右辺は ε（空系列）でないとし，また，無だな非終端記号やルールを含まないとする．G で生成される言語を $L(G)$ で表わす．

文字の有限集合 Σ 上のすべての正規集合の族を $\mathcal{R}_\Sigma$ とし，f を V_T から $\mathcal{R}_\Sigma$ の中への写像とする．ただし，各 $A \in V_T$ に対し，$f(A)$ は空系列 ε を含まない正規集合とする．ここで，f を，通常の言語理論における"代入"になるように，つぎのように拡張する．$f(\varepsilon)=\varepsilon$，$x \in V_T^*$，$A \in V_T$ に対し $f(xA)=f(x)f(A)$，$L \subseteq V_T^*$ に対し $f(L)=\{f(x) \mid x \in L\}$ とする．系列 γ の長さを $|\gamma|$ で表わす．系列 γ，非負整数 i に対し $FIRST_i(\gamma)$ を，$|\gamma| \geq i$ なら γ の長さ i の初期部分系列，$|\gamma|<i$ なら γ とする．系列の集合 L に対し $FIRST_i(L)=\{FIRST_i(\gamma) \mid \gamma \in L\}$ とする．γ の初期部分系列の全体（γ 自身とも含める）を $PREFIX(\gamma)$ で表わし，$PREFIX(L)=\{PREFIX(\gamma) \mid \gamma \in L\}$ とおく．とくに（γ 自身を除いて）γ の真の初期部分系列を $PROPERPREFIX(\gamma)$ と書き，L に対しても同様に定義する．系列 γ に対し，$\gamma=\gamma_1\gamma_2$ のとき γ_2 を $\gamma_1 \backslash \gamma$ で表わす．ただし ω が γ の初期部分系列でないとき，$\omega \backslash \gamma$ は定義されない．$\omega \backslash L=\{\omega \backslash \gamma \mid \gamma \in L\}$ とおく．

§1.3 スキャナのモデル

図1.1のようなパーザとスキャナを考える。パーザはいわゆるGに対するLR(k)パーザ(1)で，コントロール部，プッシュダウンスタック，長さkの先読みトークン系列を保持するレジスタLAからなる。図1.1に示すようにすでに決定されているトークンの系列 $A_1 \cdots A_{i-1} A_i A_{i+1} \cdots A_{i+k-1}$ に対し，長さkのトークンの系列 $A_i A_{i+1} \cdots A_{i+k-1}$ を"先読み"として（レジスタLAに入っている），$A_1 \cdots A_{i-1}$ までの構文（部分木）が求まっていたとする。つぎの A_i をスタックしたのち，A_i を含め $A_1 \cdots A_{i-1} A_i$ 上の部分木を決定するのに，つぎのk個の先読み $A_{i+1} \cdots A_{i+k-1} A_{i+k}$ が必要であるとする。すなわち，つぎのトークン A_{i+k} があらたに要ることになる。このときパーザは，Gのセンテンスで $A_1 \cdots A_{i+k-1}$ を初期部分系列としてもつすべてのセンテンスのこれ

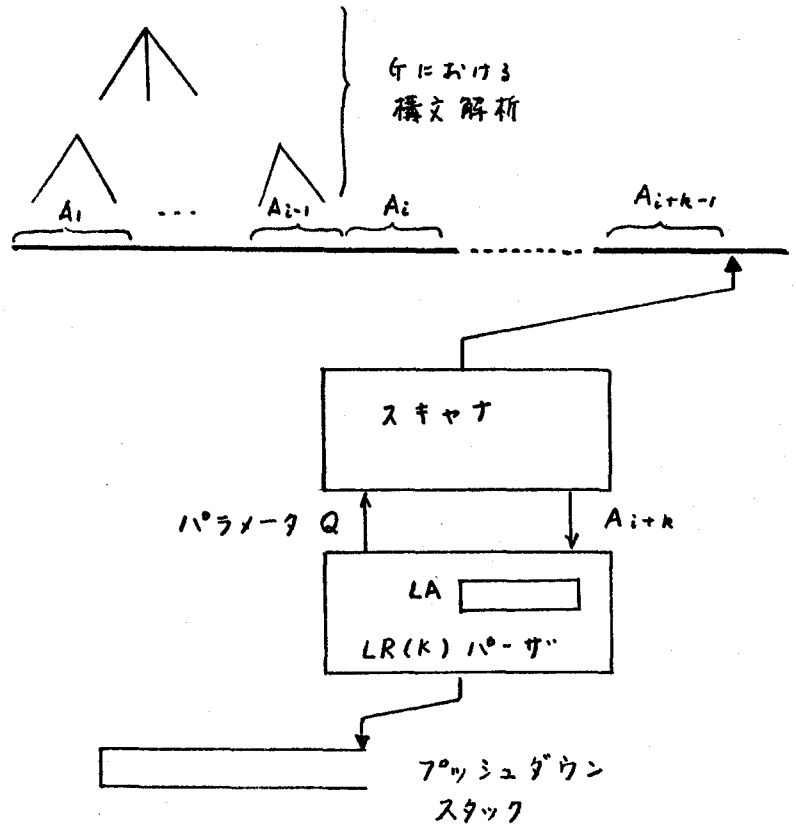


図1.1 スキャナとパーザ

に続く長さ $n+1$（nは非負整数）のトークン系列のすべて（$A_{i+k}\cdots A_{i+k+n}$として来うるものの全体，すなわち $FIRST_{k+1}(A_1\cdots A_{i+k-1}\backslash L(G))$，いまこれを$Q$と表わす）を求め（状態から知る），これをパラメータとしてスキャナにわたす．パーザの構成法とパラメータの決め方を1.7節に示す．

スキャナは今までに決定済みの最後のトークン A_{i+k-1} に対する文字系列のつぎからスキャンしはじめ，Q内の一つのトークン系列にはじめて一致するまで（はじめて $f(Q)$に属するまで）スキャンし，その時点で最初のトークン A_{i+k} とその区切りの位置を決定する．一般には，あいまいさが残り，一意に決まるとは限らないが，本論文では，そこまでスキャンすればあいまいさがなくなるという条件を考え，この条件をみたすG，fをn-区分可能であるという（定義は後述）．

n-区分可能であるとき，スキャナは入力文字系列をはじめて $f(Q)$に属するまでスキャンし，その時点で一意に定まるつぎのトークンA_{i+k}とその区切りの位置をきめ，A_{i+k} をパーザに返す（もし，スキャンの途中で，以後どんな文字系列が来ても $f(Q)$に属さないとわかったとき，入力系列を拒否して停止する）．つぎはA_{i+k} に対する文字系列のつぎの文字からスキャンする．

文字系列の同じ部分のスキャンはたかだか$n+1$回であるので，もち3ろん線形時間で動作する（区切りとトークン名の決定およびトークン系列の構文解析）．

つぎのようなスキャナのモデルを考える．スキャナはQの最初のトークン名と区切りの位置の可能性を記憶するために一般にN個のレジスタ $M_1, M_2, \cdots, M_N$ をもつものとする．このレジスタ群を総称してMレジスタという．説明の便宜上，ほかに，スキャン開始位置を示すポインタ BP_0，読み取る入力文字を指し示すポインタ BP をもつものとする（スキャン開始時 $BP = BP_0$）．スキャナのコントロール部は，状態遷移図で表わされる．各状態では，そこでとる動作が指定されている．

(1) セット：空いているMレジスタに，Qの最初のトークンの可能性として，トークン名とその区切りの位置（現在の BP の値）を格納する．

(0) クリア：あるMレジスタの内容をクリアする．

これらは必要に応じ，適当な状態に至ったとき実行する．各状態で，必要なら上のクリア，セットを行なったのち，BPを一つ進め，つぎの入力文字を読み，つぎの状態に遷移する．初期状態（スキャン開始時の状態）では，すべてのMレジスタは空，最終状態では，（必要ならクリア，セットののち）ただ一つのMレジスタがセットされていて，この内容がトークン名と区切りの位置になっている．

§1.4 k-区分可能性の判定

1.4.1 k-区分可能性の定義

説明の便宜上，エンドマーク⊣を考え，そのトークン名を⊣とする（$f(\dashv)=\{\dashv\}$，$V_T \cup \{\dashv\}$，$\Sigma \cup \{\dashv\}$ をあらためて，それぞれ，V_T，Σ とし，⊣，⊣についてはいちいちことわらない）．

以下（1.4.3以外），G（$LR(k)$文法），$f: V_T \to \mathcal{R}_\Sigma$，$k \geq 0$ を固定して考える．$Q = \{ FIRST_{k+1}(w \backslash (L(G)\dashv^k)) \mid w \in PROPER\text{-}PREFIX(L(G)) \}$ とする．$Q \in Q$，$\xi \in \Sigma^*$ について

$$TOKEN_Q(\xi) = \{ [A, t] \mid A \in FIRST_1(Q),\ 1 \leq t \leq |\xi|,\ FIRST_t(\xi) \in f(A),\ FIRST_t(\xi) \backslash \xi \in PREFIX(f(A \backslash Q)) \}$$

とする．ここで，A はトークン名に，t は ξ 中における区切りの位置に対応しており，$TOKEN_Q(\xi)$ は ξ までスキャンしたときの Q の一番目のトークンの可能性の全体を表わしている．つぎの条件をみたすとき，G，f は k-区分可能であるという．

条件：各 $Q \in Q$ について，$\xi \in f(Q)$，$PROPERPREFIX(\xi) \cap f(Q) = \phi$ なる任意の ξ に対し，$TOKEN_Q(\xi)$ はただ1個の要素 $[A, t]$ のみを含み，かつ，$\xi \notin PROPERPREFIX(f(FIRST_1(Q)))$．

これは，前述のように，ξ を左からスキャンしていくとき，はじめて $f(Q)$ に属した時点で，最初のトークンに関する区切りにあいまいさがないことを要求している．$f(Q)$ に属する時点に至る途中では $TOKEN_Q(\xi)$ はいくらでも多くの要素を含むかもしれない．しかし，G，f が k-区分可能であるとき†，1.3節で述べたモデルのスキャナで，Mレジ

---

† k-区分可能でなければ，1.3節のモデルのスキャナではトークン名とその区切りを決定できない．

スタの個数Nが有界（GとfとんできまるG）のものが存在する．ここでは，さらに，Mレジスタの個数が最小のものが構成できることをつぎに示す．

1.4.2 スキャナの構成法

［定理 1.1］ G，fがん－区分可能なら，Mレジスタの個数Nが有界のスキャナが存在し，さらに，Nが最小のスキャナが構成できる．

（証明） Nが最小のスキャナの構成法を示す（そのNが有界であることは以下の構成法からあきらか）．Nが最小であるためには，スキャナの動作の各時点で，そのときセットされているすべてのMレジスタがそれぞれ今後適当な入力文字系列が来ればそれが正しい区切りとして使われるようになっていればよい（すなわち，今後いかなる文字系列が来ても，正しい区切りとして使われる可能性のないものは，それがわかった時点ですぐにクリアすること）．

説明の便宜上，各$Q \in \mathcal{Q}$ごとに，スキャナのコントロール部S_Qをつくると考える．$\mathcal{Q}$の全体$\mathcal{Q}$は1.7節の方法で求まる．$FIRST_1(Q) = \{B_1, \cdots, B_r\}$とする（各$B_j$は互いに異なるもの）．正規集合$f(B_j)$，$f(B_j \setminus Q)$を受理する状態数最小の決定性有限オートマトンを，それぞれ，F_j，F_j'とする．F_jの状態名を$s_{j1}, \cdots, s_{jm_j}$，F_j'の状態名を$s'_{j1}, \cdots, s'_{jn_j}$とする．$s_{j1}$，$s'_{j1}$はそれぞれの初期状態とする．$m_1 + n_1 + \cdots + m_r + n_r$次元の，各要素が0または1または2のベクトル全体を考え，各ベクトルをS_Qの状態名にする．ここで，F_j，F_j'の状態名をF_1，F_1'，$\cdots$，F_r，F_r'の順に横に一列に並べ，上のベクトルの各成分を対応するF_j，F_j'などの状態名で参照する（図1.2）．

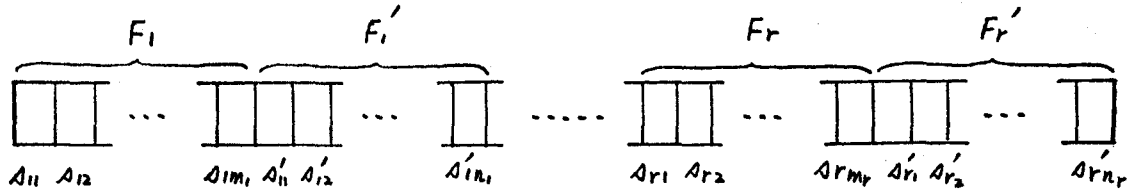


図1.2 S_Qの状態の表現

S_a の初期状態として，成分 a_{11}，a_{21}，…，a_{r1} が1で，ほかは0のベクトルをとる．1がただ一つで他は0のベクトルのうち，1のある場所が F_1'，…，F_r' の最終状態（の一つ）に対応する成分であるようなベクトルをすべて，S_a の最終状態とする．遷移はつぎのようにきめる†（最終状態からの遷移は定義しない）．F_j' において，文字 a を読んで状態 a_{jl} に遷移するような（もとの）状態の集合を $P_j'(a, a_{jl})$ であらわす．S_a の二つの状態 v，v' 間につぎの関係があるとき，v から文字 a による v' への遷移を定義する．

(イ) v における F_j 中の状態 a_{jl} に対応する成分が1で，かつ a_{jl} から文字 a による遷移が定義されているならば，v' におけるその遷移先の状態に対応する成分は1，そうでないところは0．

(ロ) v' の F_j' 中の各状態 a_{jl}（$l \neq 1$）に対応する成分の値は v における $P_j'(a, a_{jl})$ 内の成分の要素の和になっている．ただし，2を越えるときは2とし，また，$P_j'(a, a_{jl})$ が空のときは0とする．

(ハ) v' の各 a_{j1} の値は $P_j'(a, a_{j1})$ 内の各成分の要素の和に，さらにつぎの値 d_j を加えたものである．ここで d_j は v の F_j 中の最終状態に対応する成分が1のとき1，そうでないときは0とする．ただし上の値が2を越えるときは2とする．

v における動作（Mレジスタのセット，クリア）はつぎのように定める．まず空のMレジスタをセットするのは，上の(ハ)において，$d_j = 1$ でかつ a_{j1} の値が1（$P_j'(a, a_{j1})$ が空かまたはその各成分の要素がすべて0）のときであり，セットする値はトークン名 B_j と現在のBPの値の対である．このMレジスタ名は a_{j1} に対応させ，以後，この"1"が遷移とともにほかの成分へ移動するとき，このレジスタ名もその成分へ移動させて対応づける．セットされているMレジスタをクリアするのはつぎのときである．v 中の状態 a_{jl} について，(a) a_{jl} の値が2で，かつ，$P_j'(a, a_{jl})$ 内の成分で1のものがあるとき，その1をもつ成分

---

† すべてのベクトルのうち，S_a の状態として意味があるのは，$F_1, F_2, \cdots F_r$ 中にはそれぞれ1が高々一つで残りは0のものに限られる．

に対応したMレジスタをクリアする．または，(b) q'_{j_1} の値が1であっても，ν' 中の $F'_1, \cdots, F'_r$ の成分で0以外（すなわち1または2）の値をもつもの（ただし q'_{j_1} を除く）を $q'_{i_1}, \cdots, q'_{i_u}$ とすると，もし $L(q'_{j_1}) \subseteq \{L(q'_{i_1}) \cup \cdots \cup L(q'_{i_u})\}\Sigma^*$ ならば†，q'_{j_1} に対応したMレジスタをクリアする．ここで，$L(q'_{j_1})$ などは，状態 q'_{j_1} から（F'_k の）最終状態へ至るような入力系列の集合を表わす．

上の構成法からわかるように，(a) または (b) でクリアされたものは，以後どんな入力系列が来ても，最初の正しい区切りとして採用されないものであり，クリアされていないものは以後適当な入力系列が来ればそれが区切りとして採用されるものである．（証明終り）

なお，スキャン開始の位置から最終的に決定したトークンの区切りの位置に至るまでの途中で，区切りの可能性として何回でもセットされ，またはクリアされるので，たとえば（区切りの位置などをセットせずに）一度スキャンしておいて二度目のスキャンで正しいトークンと区切りを見つけるといったことはできない．

1.4.3 k-区分可能性の判定

[P1] G, f, k を与えて，G, f が k-区分可能かどうかは決定可能である．

これは，1.4.2で構成した各スキャナ Sa において，初期状態から，$F'_1, \cdots, F'_r$ の少くとも一つの F'_j の最終状態の成分が1以上で，かつ，その全成分の和が2以上であるような状態へ至る道が存在しないかどうかを調べることによって判定できる．

ところが，k を指定しなければこの問題は決定不能である．つぎにこれを示す．

---

† この判定はスキャナ自身がこの時点で行うものではない．ν' に対して，こうなっているかどうかがきまっている．

[定理 1.2] G, f を与えて，k を指定しないとき，G, f が k-区分可能となる k が存在するかどうかは決定不能である．

証明の準備として，つぎの概念を定義する．

系列の集合 $A=\{\alpha_1, \alpha_2, \cdots, \alpha_n\}$, $B=\{\beta_1, \beta_2, \cdots \beta_n\}$ (各 α_i, $\beta_i \in \Sigma^*-\{\varepsilon\}$) に対し，二つの系列 $\alpha_{i_1}\alpha_{i_2}\cdots\alpha_{i_m}$ と $\beta_{i_1}\beta_{i_2}\cdots\beta_{i_m}$ において一方が他方の初期部分系列になっているような添字の系列 $i_1 i_2 \cdots i_m$ ($1 \le i_j \le n$, $j=1, \cdots, m$) でいくらでも長いものが存在するとき，A, B は両立するという．

[P2] 上記の系列の集合 A, B が任意に与えられたとき，この A, B が両立するかどうかは一般に決定不能である．

Post の対応問題をチューリング機械 M の停止問題に帰着させ，後者の決定不能性を用いて前者の解の存在が決定不能であることを証明する方法（たとえば (2)）において，チューリング機械 M から A と B を構成したとき，「M が停止しないときかつそのときのみ A, B が両立する」ということがなりたつ．これより証明されるが詳細は省く．

任意の $A=\{\alpha_1, \alpha_2, \cdots, \alpha_n\}$, $B=\{\beta_1, \beta_2, \cdots, \beta_n\}$ に対し，CFG $G_{A,B}$ と写像 $f_{A,B}$ をつぎのようにつくる．$G_{A,B}=(\{S, S', S''\}, \{\phi, i, i', i'' \mid 1 \le i \le n\}, P, S)$，ここで $P=\{S \to iS'i', S \to iS''i'', S' \to iS'i', S' \to \phi, S'' \to iS''i', S'' \to \phi \mid 1 \le i \le n\}$ とする．$f_{A,B}$ は各 i について $f(i)=\{i\}$, $f(i')=\{\alpha_i\}$, $f(i'')=\{\beta_i\}$, $f(\phi)=\{\phi\}$ と定義する．$G_{A,B}$ は LR(1) 文法である．

[P3] A, B が両立する必要十分条件は，この A, B に対し上記のようにつくった文法 $G_{A,B}$ と写像 $f_{A,B}$ が k-区分可能となるような k が存在しないことである．

(P3 の証明) [十分性] A, B が両立しないとすると，ある $m \ge 1$ が存在して，どのような添字の系列 $i_1 i_2, \cdots i_m$ に対しても二つの系列 $\alpha_{i_1}\alpha_{i_2}\cdots\alpha_{i_m}$, $\beta_{i_1}\beta_{i_2}\cdots\beta_{i_m}$ は一方が他方の初期部分系列となることはない．このとき $G_{A,B}$, $f_{A,B}$ は $k=m-1$ に対し k-区分可能であることをつぎに示す．系列 $\omega \in PROPER\ PREFIX(L(G_{A,B}))$ としてつぎの四

つの場合がある．(a) $w = i_1 i_2 \cdots i_j$，(b) $w = i_1 i_2 \cdots i_t \$$，(c) $w = i_1 i_2 \cdots i_t \$ i'_t \cdots i'_{t-j}$，(d) $w = i_1 i_2 \cdots i_t \$ i''_t \cdots i''_{t-j}$．(a) の w に対し，Q は $i_{j+1} \cdots$ と $\$ \cdots$ の形の系列を含むが，$f(i_{j+1}) = \{i_{j+1}\}$，$f(\$) = \{\$\}$ ゆえ，最初のトークン $TOKEN_Q(\xi)$ はそれぞれ $[i_{j+1}, 1]$ または $[\$, 1]$ と一意にきまる．(b) の w に対し，$Q = \{i'_t i'_{t-1} \cdots i'_{t-h}, i''_t i''_{t-1} \cdots i''_{t-h}\}$ である．$f(i'_t \cdots i'_{t-h}) = \alpha_{i_t} \cdots \alpha_{i_{t-h}}$，$f(i''_t \cdots i''_{t-h}) = \beta_{i_t} \cdots \beta_{i_{t-h}}$ であるが，$\alpha_{i_t} \cdots \alpha_{i_{t-h}}$ と $\beta_{i_t} \cdots \beta_{i_{t-h}}$ は一方が他方の初期部分系列ではないので，スキャンした系列 ξ が $\xi = \alpha_{i_t} \cdots \alpha_{i_{t-h}}$ となったとき，最初のトークン $TOKEN_Q(\xi)$ は $[i'_t, |\alpha_{i_t}|]$ と一意にきめられる（β の方である可能性はない）．$\xi = \beta_{i_t} \cdots \beta_{i_{t-h}}$ となった場合も同様（形式的には $t-h<1$ のときエンドマーク $\dashv$，そのトークン $\dashv$ を入れて議論する必要があるが，$\alpha_{i_t} \cdots \alpha_{i_{t-j}} = \beta_{i_t} \cdots \beta_{i_{t-j}}$，$j \leq h$ となることはないので，文の終りを見た時点ではトークンの区切りは一意にきまる）．(c)，(d) の w に対し，Q はそれぞれ $\{i'_{t-j-1} \cdots i'_{t-j-h-1}\}$，$\{i''_{t-j-1} \cdots i''_{t-j-h-1}\}$ で，この場合もそれぞれ $[i'_{t-j-1}, |\alpha_{i_{t-j-1}}|]$，$[i''_{t-j-1}, |\beta_{i_{t-j-1}}|]$ と一意にきまる．

［必要性］ $G_{A,B}$，$f_{A,B}$ が h-区分可能であるとする．系列 $w = i_1 i_2 \cdots i_t \$ \in PROPERPREFIX(L(G_{A,B}))$ を考えると $Q = \{i'_t \cdots i'_{t-h}, i''_t \cdots i''_{t-h}\}$ であるが，このとき $\alpha_{i_t} \cdots \alpha_{i_{t-h}}$，$\beta_{i_t} \cdots \beta_{i_{t-h}}$ は一方が他方の初期部分系列にはならない（そうでなければ，区切りがあいまいになり h-区分可能性に反する）．添字の系列 $i_t \cdots i_{t-h}$ として長さ $h+1$ の任意のものをとってよいから，A と B は両立しない．（証明終り）

P2，P3 により，定理 1.2 が証明された．

§1.5 DLとの関係

k-区分可能なクラスについて，まず，

[P4] 各正規集合を生成する文法を左線形文法でつくり，この各文法とGとをあわせた全体の文法をG'とする．G（LR(k)文法），fがk-区分可能であるが，G'が任意に大きいk'に対しLR(k')文法でないようなG，fが存在する．

もとの文法Gの部分を固定していることに注意．題意の例としては，たとえば，$G=(\{S,X\},\{A,B,C\},P,S)$，$P=\{S\to AB, S\to XC, X\to A\}$とし，$f(A)=\{a\}$，$f(B)=\{d^nb \mid n\geq 1\}$，$f(C)=\{d^nc \mid n\geq 1\}$をとればよい．

しかしながら，生成される言語$f(L(G))$のクラスは決定性文脈自由言語（DL）のクラスをはみ出ない．（定義よりDLを含むのはあきらか．ゆえに，DLのクラスと一致する．なお，一般にDLに正規集合を代入すればDLのクラスをはみ出る．たとえば$L=\{wcw^R \mid w\in\{a,b\}^*\}$，$f(a)=\{0\}$，$f(b)=\{1\}$，$f(c)=\{00,11\}$とすれば，LはDLであるが，$f(L)=\{\xi\xi^R \mid \xi\in\{0,1\}^*-\{\varepsilon\}\}$はDLではない．）

[P5] G，fがk-区分可能なら，$f(L(G))$はDLである．

$f(L(G))$を受理する決定性プッシュダウンオートマトン（DPDA）を構成するかわりに，ここではさらに，Gに対するLR(k)パーザとfを区切るスキャナを，スキャナにおける再スキャン（同じ文字系列の部分を2度以上スキャンすること）をなくして構成できることを示す．$f(L(G))$を"受理"するには，トークン間の区切りの位置は必要ないので，スキャナにおける区切りの位置（BPの値）を覚えることを取り除けばよい．そうすれば，パーザとスキャナをあわせて有限のコントロール部となり，しかも入力は一方向であるので，DPDAになっている．

1.4.2で構成したパーザを修正し，スキャナにわたすパラメータはつぎに来うる長さ$2k+1$のトークン系列の集合であるようにする．これを一般に$R\subseteq V_T^{2k+1}$で表わす．スキャナ部は，1.4.2で構成したスキ

ャナ S_q を各 $q \in Q$（Qは長さ $n+1$ のトークン系列のある集合）ごと $1+N+\cdots+N^n$ 個ずつ用意する（Nは1.4.2で構成した必要なMレジスタの個数の最小値）。さらに，Mレジスタはそれぞれの S_q に用意する．この S_q をここではとくに部分スキャナと呼ぶ．スキャナ部には，これらの部分スキャナをつぎつぎと起動させ，また停止させるようなつぎのような有限コントロール部Cをもつ．動いている部分スキャナは一般にいくつかあるが，Cはこれらが起動された順序関係を樹状のグラフTで覚えている（一つの節点から出る枝の数はN以下，高さはn以下）．節点のラベルは $q \in Q$ で，それに対応して一つの部分スキャナ S_q が動いている．その節点から出る枝にはその S_q の現在セットされているMレジスタの番号とそれに格納されているトークン名Aがつけられる．

まず，わかりやすいように，最初の段階を考え，Cがまだ空のTしかもっていないとし，パラメータRをもらったとする．Cは部分スキャナ S_{q_0}，$q_0 = FIRST_{n+1}(R)$ を起動させ，スキャン開始位置よりスキャンを始める．S_{q_0} がそのMレジスタをセットするとき（レジスタ番号をi，トークン名をAとする），樹の根（ラベル q_0）よりラベルi，Aの枝を出して新たに設けた節点（ラベルは $FIRST_{n+1}(A \backslash R)$，これを q_1 とおく）へ結び（これがこの時点のT），S_{q_0} 以外にこの時点から部分スキャナ S_{q_1} をその初期状態から動かす．

一般に，Tのある節点 v（根から v へ至る道上のラベルのトークン名の系列を $A_1 \cdots A_r$ とする）に対応している部分スキャナがそのMレジスタをセットするとき（そのMレジスタの番号をi，トークン名を A_{r+1} とする），節点 v よりラベル i, A_{r+1} の枝を出し，新たに設けた節点（ラベルは $FIRST_{n+1}(A_1 \cdots A_r A_{r+1} \backslash R)$，これを q' とおく）へ結び，この時点から部分スキャナ $S_{q'}$ を起動さす．Tのある節点 v に対応している部分スキャナがそのMレジスタ i をクリアするとき，v から出ている枝でラベルに i をもつ枝も含め，それ以降の部分木をTから取り去り，その部分木中の節点に対応していた部分スキャナをすべて停止さす．

スキャナ部の出力（パーか部への返事）はTの根に対応した部分スキャナ S_q が決定する．その S_q が入力系列を拒否すれば，拒否．S_q が

その最終状態に入ったとき，スキャナ部は停止し，S_Q のセットされているただ一つのMレジスタ内のトークン名をパーザに返す．ここで，T は根からそのMレジスタ番号をラベルとする枝のみが一つ出ているが，根とその枝を除き，残ったものをつぎの T とする．

パーザでつぎのトークンが必要になったとき，あらたに $R' \leq V_T^{2k+1}$ を求め，それをパラメータとしてスキャナ部にわたす．スキャナ部は前の T をひきつぎ（すなわち，この T の節点に対応した部分スキャナはそれぞれある状態まで動いている），前のつづきをスキャンする．以後起動する部分スキャナは R' を用い，上と同様に決定する．

各部分スキャナはMレジスタを N 個以下しか持たず，また T の根に対応した部分スキャナが最終状態に入るのはそれが起動されスキャンを始めたところからちょうど $k+1$ 個のトークン系列にはじめて一致したときである．したがって，T の節点から出る枝の数は N 以下で，高さは k 以下であり，コントロール部 C は有限である（もし，今後いかなる入力系列が来てもトークンとして採用され得ないものを（Mレジスタに）残しておくならば，それから起動されるものはいくらでも増えるかもしれない）．

§1.6 結言

ここでは，まず，スキャナの等価性について少しふれる．1.4.2のスキャナ S_Q は，$n+1$ 個先のトークンまで完全に一致するところまでスキャンせずとも，その途中で，最初のトークン名とその区切りにあいまいさがなくなればその時点で停止し，パーザにそのトークンを知らせればよい．すなわち，決定したトークン名とその区切りの位置さえ同じなら，どこまでスキャンするかは等価性ということから見ればあまり意味がない．そこで，二つのスキャナは，パーザから同じパラメータを受けとってスキャンを開始するとき，決定したつぎのトークン名とその区切りの位置が互いに同じなら等価ということにする（ただし一方が拒否すれば他方も拒否すること）．モデルは1.3節のトークンの可能性をセットするためのあるきまった個数以下のMレジスタをもつものを考える．

この等価性は判定可能である．二つのスキャナを並列に動かすいわゆる"直積マシン"をつくる．このとき，トークン名，区切りの位置ともに同じものがセットされたMレジスタは対応づけて覚えておく．一方が停止してあるMレジスタの値を出力したとき，等価なら，他方は今後来るいかなる入力系列に対してもそれと同じ値をセットしているMレジスタ（これがなければ等価ではない）を一つ残して停止すべきである．Mレジスタのセット，クリアの状況も"状態"にとりいれて考えるなら，上の判定は有限オートマトンの問題となるので，決定できる．

簡単化を一般に議論するのは困難と思われる．それに関連して，つぎのような問題が残されている．

(1) 拒否をパーザでも行うことにし，n-区分可能性の条件をそこなわない範囲でパラメータ Q を Q'（$Q' \geq Q$）で適当におきかえ，Q' の個数を全体として減らすこと．

(2) （上のように予測を修正し）つぎつぎの予測 Q の系列がある正規集合になっているならば，スキャナは（パーザから情報をもらわない形で）パーザから切りはなして独立に構成できる．パーザとそのパラメータの求め方をきめると，パラメータの系列はDLゆえ，これが正規集合

かどうかは判定できる[3]．

(3) パーザから与えられるパラメータを固定して，スキャナの部分を簡単化する問題がある．パラメータは必要なとき何回でも見ることができるとし，それを見るコストを適当に導入し，スキャナを全体として最適化する問題は興味深い．パラメータは最初に一度しか見ない（これは各パラメータごとにスキャナのコントロール部を別々につくるやり方）か，または，入力文字とともに毎回見る（パラメータを見るコストを零と考えている）ようなモデルなら，有限オートマトンの簡単化と同じ問題になる．

なお，PL/I は予約語などの制限をできるだけ除いて柔軟性に富んでいる言語であるが，PL/I のたとえば DECLARE (A1, A2, …, An)… は，n が任意に大きくてもよいとすると，n を固定した本論文のモデルでは解析できない．

§1.7 補足

ここでは，パーサの構成法とパラメータの決め方を示す．

$\gamma \in V^*$ に対し，$L(\gamma) = \{ w \mid G$ で $\gamma \overset{*}{\Longrightarrow} w,\ w \in V_T^* \}$ とおく．文献(1)と同じように，状態 q の集合 Q をつぎのように求める．Gのルールに1から π（Gのルールの個数）までの番号をつけ，あらたに出発ルール $S_0 \to S \dashv^{k+h}$（番号0をつける）を考える．p 番目のルールを $X_p \to X_{p1} X_{p2} \cdots X_{pn_p}$ とする．まず $q_0 = \{ [0, 0, \dashv^{k+h}] \}$，$Q = \{ q_0 \}$ とする．任意の $q \in Q$，$Y \in V$ について $q_Y = \{ [p, j+1, \alpha] \mid [p, j, \alpha] \in q$ かつ $X_{pj+1} = Y \}$ を求め，空でなければ Q へ入れる．ここで q' は

$$q' = q \cup \{ [\ell, 0, \beta] \mid X_{pj+1} = X_\ell,\ \beta \in FIRST_{k+h}(L(X_{pj+2} \cdots X_{pn_p} \alpha)),\ j < n_p,\ [p, j, \alpha] \in q' \}$$

をみたす最小の集合である．状態 q にあるとき，LA にある長さ k の系列が $Z(q, p) = \{ FIRST_k(\alpha) \mid [p, n_p, \alpha] \in q \}$ に属していれば，ルール p を用いた簡約動作，$Z(q, stack) = \{ FIRST_k(\beta) \mid j < n_p,\ \beta \in L(X_{pj+1} \cdots X_{pn_p} \alpha)$ なる $[p, j, \alpha] \in q'$ が存在する $\}$ に属していればスタック動作（LA の左端のトークンをスタックする）を行う．スキャナを働かせる時点はスタック動作直後であり†，そのときの Q はつぎのように決める．スタック後の状態を q とし，LA の残りの長さ $k-1$ の系列を $A_{i_1} \cdots A_{i_{k-1}}$ とすると††，$\delta(q, A_{i_1} \cdots A_{i_{k-1}}) = A_{i_1} \cdots A_{i_{k-1}} \setminus \{ FIRST_{k+h}(\beta) \mid \beta \in L(X_{pj+1} \cdots X_{pn_p} \alpha)$ なる $[p, j, \alpha] \in q'$ が存在する $\}$ を Q とし，スキャナにわたす．

---

† $k = 0$ のときは，つぎのトークンをスタックしたい時点でスキャナを働かせると考える．

†† 動作開始時の過渡期は $A_{i_1} \cdots A_{i_{k-1}}$ は短かいが，この場合も同様に考える．

第2章　アナライザの等価性と簡単化

§2.1　序言

文法Gに対するパーザとは，与えられた入力系列がそのGで導出される文かどうか判定し，もし文ならその導出木を与えるものをいう．コンパイラにおけるいわゆる構文解析部のおもな目的は，語い解析を終えたプログラムを読み，シンボルテーブルの処理や内部コードを発生するセマンティックルーチンをつぎつぎと正しく呼ぶことである．たとえば，いま，アルゴルのコンパイラを考え，語い解析を終えたプログラムの一部として，<identifier>:=<identifier><multiplying operator><identifier><adding operator><identifier> … (*) が与えられたとしよう．コンパイラはアルゴルの文法におけるこの系列の導出木を忠実に求め，かつ，そこに現われるすべての書き換えルールに一対一に対応したセマンティックルーチンを呼ぶとは限らない．とくに，右辺がただ一つの非終端記号からなるようなルール（たとえば，<term> → <factor> など）に対しては意味づけの処理を行なわないことが多い[16]．すなわち，内部コード発生などの意味づけの処理をするとき，固定したもとの文法における導出木そのものが必ずしも必要でない．

そこで，本論文では，セマンティックルーチンを正しく呼ぶという本質的な機能だけに着目し，その機能を果すものをアナライザと呼ぶ．アナライザはその内部で特定の文法は用いていない．アナライザのモデルとして，一定量までの先読みを許した決定性プッシュダウンオートマトンを考える．アナライザが呼ぶセマンティックルーチンは複数個あり，それぞれに名前がついている．アナライザに与えられる入力記号列は，たとえば前述の（*）のようなものであるが，各トークン（<identifier>など）にはパラメータ（シンボルテーブルへのポインタなど）が付随している．アナライザの動作はトークンの系列によって決まる．アナライザはパラメータと制御記号を格納するためのプッシュダウンスタック

(pds)をもつ．アナライザは動作を決定するために必要なら トークンを先読みし，入力のトークンに付随したパラメータをpdsにスタックしたり，あるいは，pdsの頭からn(nは1以上の整数)コマの内容をとり出し，それらのパラメータ部分をパラメータとしてあるセマンチックルーチンpを呼ぶ．前者をスタック動作，後者を簡約動作という．呼ばれたルーチンpはわたされたn個のパラメータに依存して新しいパラメータを一つ求め，これらを用いて内部コードを発生するとともに，新しいパラメータをpdsの頭のコマに返す(n個のパラメータが1個のパラメータに置き変わり，pdsはn-1コマ短くなる)．たとえば，前述の例(*)において，<identifier><multiplying operator><identifier>に付随したパラメータ(テーブルへのポインタなど)を左からq_1，q_2，q_3とすると，q_3までスタックしたのちつぎのトークンを見ると<adding operator>であり，乗除算を先に実行しなければならないので，この時点で簡約 RED(3) CALL P_i を行なう．ここで呼ばれたルーチンP_iは，中間結果を格納する場所を定めたのち，q_1，q_3で示される内容を乗算または除算(q_2で決まる)して結果を新しいロケーションに格納するという内部コードを発生するとともに，中間結果を格納した場所を指し示すポインタなどを新しいパラメータとして返す．このアナライザは，先読み，スタック，簡約動作をくり返しながら，与えられた入力系列を受理または拒否する．

2.2節で，アナライザのモデルについて述べ，アナライザの等価性を定義する．アナライザとパーザの違いについても説明する．2.3節では，任意のアナライザAが与えられたとき，Aと等価なアナライザのうちで，先読みの長さがつねに最小のアナライザBを一つ求める方法を述べ，2.4節では，さらに，先読みが最小のものの中でもっとも簡単な(フローチャートの節点数が最小の)アナライザB_mを一つ求める方法を述べる．

§2.2 アナライザの等価性

2.2.1 アナライザのモデル

図2.1のようなモデルを考える。入力テープにはトークンとそれに付随するパラメータの対の系列が与えられる。プッシュダウンスタック（pds）は制御記号を書込むCトラックとパラメータを書込むSトラックにわかれている。ヘッドNHはつぎにスタックすべきパラメータのあるコマを指し，LHは先読みするコマを指す。先読みは1コマずつ行ない，一度先読みしたコマはあとで決して見ない。トークン，制御記号，アナライザが呼ぶセマンチックルーチン名の集合を，それぞれ，Σ，Γ，Pとする。動作開始時，ヘッドNHとLHはともに入力テープの左端にあり，SHはpdsの底のコマを指している。動作中，LHはNHのコマよりたがだかkコマ右にあるとする。制御部は計算機のプログラムに対応してフローチャートで書く。$\dashv$をエンドマークとし，$\Sigma \cup \{\dashv\} = \{a_1, a_2, \cdots, a_m\}$とする。各節点では，以下のいずれか一つを行なう。

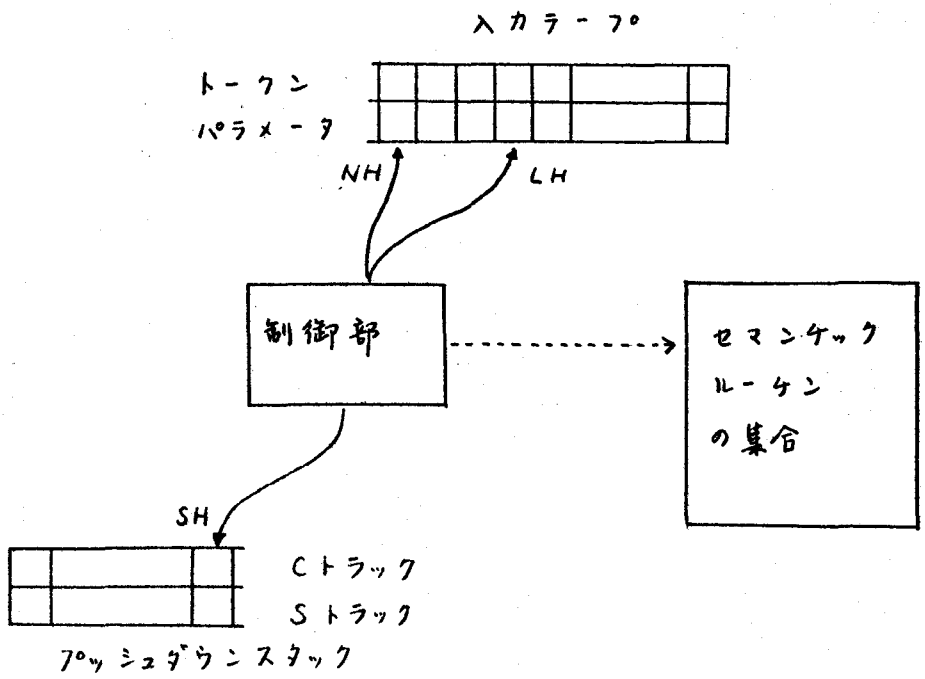


図2.1　アナライザ

(1) 先読み動作 LOOK. LHのコマのトークンを見てつぎの動作を決める. 図2.2 (a) の場合, そのトークンが a_i ならつぎは節点 L_i を実行する. LHを1コマ右へ動かしておく (エンドマークを見たとき以外).

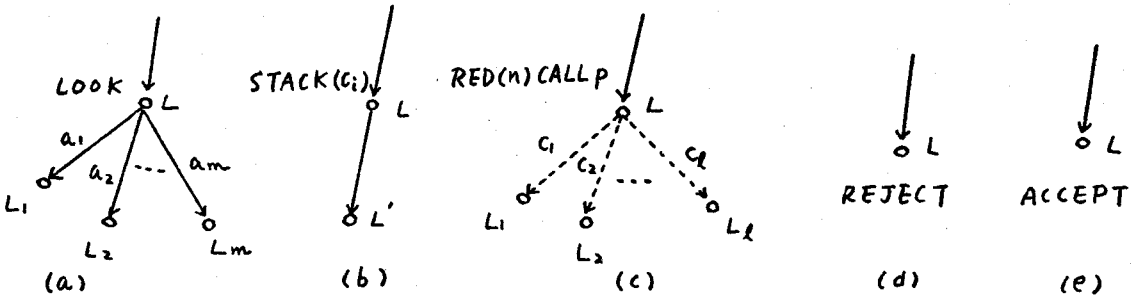


図 2.2 アナライザの基本動作

(2) スタック動作 STACK(C_i). SHの現在あるコマのCトラックに制御記号 C_i を書込み, SHを1コマ右へ動かして, そのSトラックにNHのコマのパラメータ記号を書込む. NHを1コマ右へ動かし, つぎは節点 L' へ (図2.2 (b), 図2.3 (a))

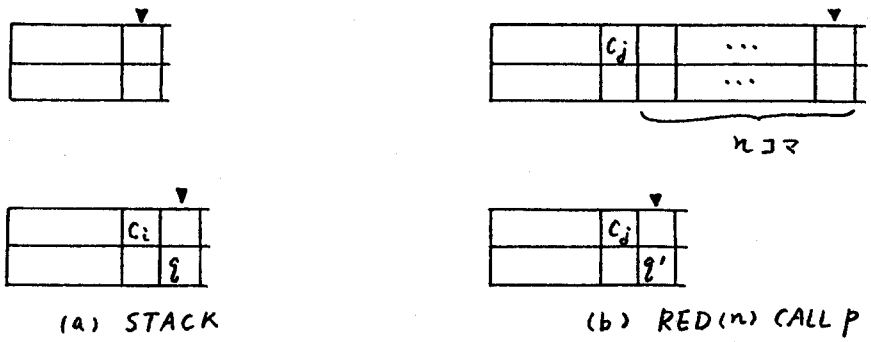


図 2.3 プッシュダウンスタックの変化

(3) 簡約動作. RED(n) CALL p. Sトラックの頭からnコマの内容をパラメータとして, ルーチンpを呼ぶ. pdsをn-1コマ短くし, 頭のコマのSトラックにルーチンpが返した新しいパラメータ ℓ' を書込む (図2.3 (b)). 一つ左のコマの制御記号を見てつぎの動作を決める. 図2.2 (c) の場合, それが C_j ならつぎは節点 L_j へ.

(4) 拒否動作 REJECT. 入力系列を拒否して, 停止する.

(5) 受理動作 ACCEPT. 入力系列を受理して, 停止する. この

とき SH はスタックの底から2番目のコマを指していて†，NH，LH はエンドマークのコマを指しているものとする．

アナライザ自身の動作は入力として与えられるトークンの系列で決まる．アナライザ A が受理するトークンの系列の集合を L_A とする．トークンの系列 w に対する A の全動作系列とは，w に対してとった受理または拒否に至るまでの A の先読み動作，スタック動作，簡約動作（パラメータの数とルーチン名 p も含める．以後も同様）の系列をいう．

［定義］ アナライザ A_1，A_2 は $L_{A_1} = L_{A_2}$ であり，かつ，受理する任意のトークンの系列に対する A_1 と A_2 の全動作系列から先読み動作を除去した残りのスタック動作と簡約動作の系列が互いに同じとき，等価であるという．

先読み動作や制御記号に関する動作は等価性に直接関係はない．また，L_A に属さない拒否する系列に対しては，拒否さえすれば拒否の時点の遅れやそれまでの動作系列は違ってもよい．入力系列の拒否はセマンチックルーチンでなく，アナライザが行なうものとしているので，$L_{A_1} = L_{A_2}$ を要求している．上の意味で等価であれば，トークンに付随するパラメータまで含めて全く同じ入力系列を与えたとき，A_1 と A_2 は同じパラメータで同じセマンチックルーチンをつぎつぎと呼ぶことになり，したがって同じ内部コード系列が発生される．また逆に，各セマンチックルーチンの処理のやり方にかかわらず同じ内部コード系列を発生するといういわゆるプログラム理論でいう強等価性を考えると，それは上で定義した等価性と一致する．

上述の等価性の意味が直観的にわかりやすいようにつぎのような樹状グラフを導入する．アナライザ A で受理されるトークンの系列 w に対する A のスタック動作，簡約動作の系列から決まる w の"句構造"を樹状グラフで表わし，各節点のラベルにはその句にまとめたときに呼んだルーチン名をつけて得られる木を $\tau_A(w)$ とする．たとえば，$w = a_1 \cdots$

---

† これは入力系列が最終的には一つの句にまとめられることを要求している．

a_5 に対し，先読みと制御記号に関する動作を除いた動作系列が，STACK，STACK，RED (2) CALL p_1，STACK，STACK，STACK，RED (3) CALL p_2，RED (2) CALL p_3 のとき，$\tau_A(w)$ は図 2.4 のようになる．アナライザ A に対し

$$T_A = \{ \tau_A(w) \mid w \in L_A \}$$

とする．アナライザはこの木をいわゆる"左から右へ，下から上へ"つくっていく．スタック動作と簡約動作の系列が同じということは，この木が同じということである．ゆえに，上の定義はつぎのようにも述べられる．

[定義] アナライザ A_1，A_2 は $T_{A_1} = T_{A_2}$ のとき，かつそのときのみ，等価であるという．

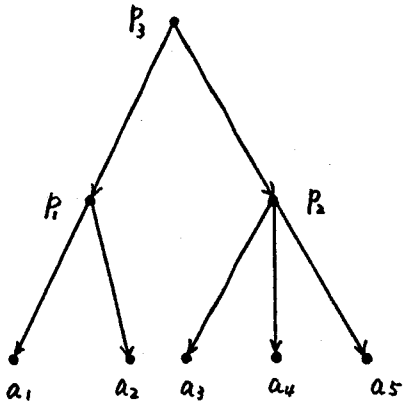


図 2.4　木 $\tau_A(w)$ の一例

2.2.2　アナライザとパーザの関係

ここではアナライザとパーザの関係について考察してみる．文脈自由文法（CFG）を（V_N，V_T，R，V_S）で表わす．V_S は始記号の集合で，便宜上，複数個の始記号を考える．CFG G と，G の非終端記号の集合 V_N からある記号の集合への写像 h に対して，G の文の任意の導出木 t に対し端点でない各節点のラベル（非終端記号）X をそれぞれ $h(X)$ につりかえて得られる木を簡単に $h(t)$ と書く．G のすべての文のすべての導出木に対する $h(t)$ の全体を $T(G, h)$ とする．

[補題 2.1] 任意のアナライザ A が与えられたとき，CFG G と非終端記号の集合から P への写像 h を，$T(G, h) = T_A$ がなりたつように構成できる．

構成法を 2.2.3 に示す．非決定性プッシュダウンオートマトンによって空スタックで受理される入力系列の集合を生成するような CFG のつくり方はよく知られているが，G のつくり方においてその通常の方法と違う点は，先読み系列を考慮すること，pds へのスタックは1コマ，ポップアップは一般に n コマであること，ポップアップ後の行き先（状態）はまとめた句の一つ左の句に対応するコマの制御記号に依存して決まることなどである．

二つのアナライザ A_1, A_2 が等価かどうかは，A_1, A_2 を同時に模倣する決定性プッシュダウンオートマトンを構成することによって判定することもできるが，二つのアナライザからそれぞれ上述の CFG と写像を構成し，$T(G_1, h_1) = T(G_2, h_2)$ かどうかを調べる（CFG の句構造的等価性の判定[14]と同様な方法で決定できる．ラベルがついているところだけ違う）ことによっても判定できる．また，この文法 G は，いわゆるもとの言語の意味を文法の上で定義している場合（Knuth の方法[15]），構成したアナライザが目的にかなった正しいものであるかどうかの検証にも役立つ．

本論文では，LR(k) パーザ[1]のモデルも上述のアナライザと同じように，先読み，スタック，簡約の動作を基本動作と考える．アナライザと違うところは，簡約のとき，そこで使う文法 G のルール名を出力することである†．

一般に，$T(G, h) = T_A$ であるような任意の G について，G に対する LR(k) パーザが存在するなら，そのパーザ乙において，簡約のとき，そこで使う G のルール名の代わりに左辺の非終端記号名 X で決まる

---

† G のルール名を出力する代わりに，そのルールの右辺の長さと左辺の非終端記号名を出力すればよいなら，一般に，より簡単なパーザが得られる．

セマンチックルーチン $h(X)$ を呼ぶことにすると，A と等価なアナライザが得られる．このアナライザを $a(Z)$ とする．もし h が一対一なら，アナライザとパーザは本質的に違わない．しかしながら一般に，与えられた A に対して h が多対一であるような G と h しか存在しない場合もあり†，この場合には，最簡形のアナライザはどのように G を選んでも，それに対する最簡形のパーザ乙に対する $a(Z)$ として求めることができるとは限らない．以下にその例を示す．図 2.5 のような T_A をもつアナライザ A を考えよう．受理するトークンの系列は，$aa'cc$ と $bb'cc$ の

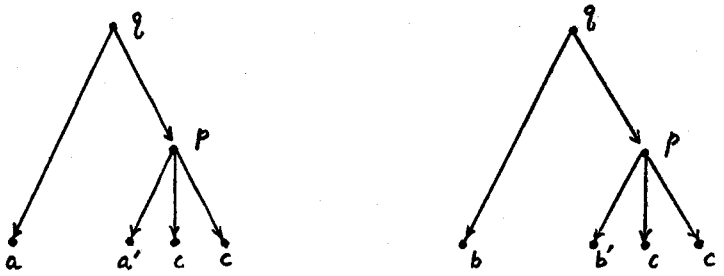


図 2.5　T_A

2個だけで，まとめられる句のトークンの系列が違っても同じルーチンを呼んでいる（パラメータに依存して異なる内部コードが発生される）．ところで，$T(G, h) = T_A$ なる G は $G = (\{S, X, Y\}, \{a, a', b, b', c\}, \{S \to aX, S \to bY, X \to a'cc, Y \to b'cc\}, \{S\})$ 以外にない（X と Y を同じものにすれば，A が拒否すべき系列も生成してしまう）．アナライザが A と G に対する LR(0) パーザのフローチャートの節点数はあきらかにアナライザの方が少なくてよい．アナライザでは aa'，bb' に付随するパラメータをスタックしたあと同じ節点に行ってもよいが，パーザでは以後の簡約に使うルール名が異なるため，違う節点に遷移されて区別しておく必要があり，さらに $aa'c$，$bb'c$ や $aa'cc$，$bb'cc$ まで読込んだときも，それぞれ区別しておく必要がある．

---

† たとえば，簡約のときセマンチックルーチンを呼ばないような $X \to Y$ の形のルールが存在しないように文法をつくりかえれば，非終端記号が増えて多対一つの G と h になり，この G，h に対し，h' が一対一の $T(G, h) = T(G', h')$ なる G' は一般に存在しない．

2.2.3 文法Gの構成法

与えられたアナライザAから，$T(G, \ell) = T_A$ なるGとℓを構成する方法を述べる。

アナライザの動作の任意の時点で考えて，NHのコマからLHの左となりのコマまでに書かれているトークンの系列をその時点における先読み系列という。一般に，ある一つの節点の動作を実行しようとするとき，過去の動作の履歴によって異なる先読み系列をもっている。そこで，まず，各節点ごとに先読み系列が一意に対応するようにAのフローチャートを変換する。Aのフローチャートの各節点L，長さk以下の$\Sigma^* \cup \Sigma^{*+}$の任意の元$\alpha$に対し$(L, \alpha)$をつくってこれを新しい節点名にする。$(L, \alpha)$における動作はLにおける動作と同じにし，遷移先は先読み系列の変化を考慮して決める（たとえば$(L, a\alpha)$でスタック動作なら行き先は(L', α)，(L, α)で先読みしそれがトークンaなら行き先は$(L', \alpha a)$などとする。L'はもとのLからの遷移先）。これから，むだな節点を除く（必ずしも必要でない）。以下，このフローチャートを対象にする。先読み動作は等価性に直接意味をもたない。そこで，Aにおける実行開始点，または簡約動作，スタック動作直後の節点から，つぎの簡約動作，スタック動作，受理動作に至るまでの先読み動作の系列をひとまとめに考え，この節点の系列をm系列と呼ぶ。Aのm系列の全体をMとし，$\xi \in M$の終りの節点の動作を$a(\xi)$（スタック動作，簡約動作，受理動作），$a(\xi)$がスタック動作のときそこで書込む制御記号を$c(\xi)$，ξの終りの節点における先読み系列の左端の記号（節点によって一意に決まっている）を$s(\xi)$で表わす。

さて，Gをつくる前に，まず，AからCFG $G' = (V_N', \Sigma, R', V_S')$をつぎのようにつくる。

(1) $V_N' = \{ (\xi, X, \xi') \mid \xi, \xi' \in M,\ a(\xi)$がスタック動作，$X \in \Sigma \cup P \}$

(2) ルールR'はつぎのようにつくる。

(i) $a(\xi)$がスタック動作でそこからξ'の始点へ枝があるようなす

べての $\xi, \xi' \in M$ に対し，$(\xi, X, \xi') \to \lambda$ を R' のルールとする．ただし $\lambda = s(\xi)$．

(ii) 以下の条件 (a), (b) がなりたつような m 系列 $\xi_1, \cdots, \xi_n, \xi_n', \xi_0'$ に対し，すべての $X_1, \cdots, X_n \in \Sigma \cup P$ について $(\xi_0, p, \xi_0') \to (\xi_1, X_1, \xi_1')(\xi_2, X_2, \xi_2') \cdots (\xi_n, X_n, \xi_n')$ をつくり R' のルールとする．ここで，$\xi_0 = \xi_1$, $\xi_1' = \xi_2$, $\cdots$, $\xi_{n-1}' = \xi_n$ である．

(a) $1 \leq i \leq n$ なる各 i について，$a(\xi_i)$ はスタック動作

(b) ξ_n' の終りの節点の動作は簡約 RED (n) CALL p で，そこから ξ_0' の始点へラベルが $c(\xi_1)$ の破線がある．

(3) Aの動作開始点から始まる m 系列 ξ，$a(\xi')$ が受理動作である n 系列 ξ' をもつ $(\xi, X, \xi') \in V_N'$ をすべて始記号とする．

むだな非終端記号やそれを含むルールを除く．この G' のルールで $(\xi, X, \xi') \to X$ の形のものを除き，他のルールの (ξ, X, ξ'), $X \in \Sigma$ を単に X で置き換えて得られるルールをもつ CFG を G とする．写像 h を $h((\xi, X, \xi')) = X$ と定義すると，この G と h について $T(G, h) = T_A$ がなりたつが，証明は長くなるので省く．

§2.3 先読み最小のアナライザ

2.3.1 諸性質

任意のアナライザAが与えられたとき，それと等価なアナライザの中で，先読みの長さ（ヘッドNHとLH間の距離）が最小のアナライザBを求める方法を述べる．ここで，先読みの長さが最小という意味は，Aと等価な任意のアナライザA'の受理する入力系列wに対するはじめからt回目のスタックまたは簡約動作時における先読みの長さを$\ell(A', w, t)$と書くことにすれば，入力系列wのどのt回目のスタックまたは簡約動作時を考えても，$\ell(B, w, t) = \min\{\ell(A', w, t) \mid A'$はAと等価な任意のアナライザ$\}$がなりたつことである（このようなアナライザが存在することは，後述の議論からわかる）．L_Aに属さない拒否する入力系列に対する動作は拒否以外全く自由である．パーザZに対しても，アナライザと同様，先読みの長さを$\ell(Z, w, t)$で表わす．以下で示すように，このようなアナライザBは，文法を適当に選べば，そのパーザの出力をつけかえて得られる．つまり，先読みの長さについてはアナライザも（文法を適当に選んだときの）パーザも同じである．

［定理］ 与えられたアナライザAと等価で，先読みの長さが最小のアナライザBは，つぎの条件(1)，(2)をみたす文法$\bar{G}$に対するパーザで，パーザとして先読み最小のものZから，出力をつけかえて得られるアナライザ$a(Z)$として求められる．

(1) $T(\bar{G}, \bar{h}) = T_A$なる$\bar{h}$がある．

(2) $\bar{G}$に右辺の同じルール$U \to \alpha$，$V \to \alpha$があれば，$\bar{h}(U) = \bar{h}(V)$である．

（証明） $a(Z)$が先読み最小でないとしたら，あるwとtについて$\ell(A', w, t) < \ell(a(Z), w, t)$なるアナライザA'が存在する．ところが，つぎの補題に示すように，A'と同じ先読みの長さをもつ$\bar{G}$に対するパーザ$z(A')$がつくれる．Zと$a(Z)$は同じ先読みゆえ$\ell(z(A'), w, t) < \ell(Z, w, t)$となり，パーザZが先読み最小という仮定に反する．

（証明終り）

[補題 2.2] A と等価な任意のアナライザ A' に対し，A' を修正して $\bar{G}$ に対するパーザ $z(A')$ がつくれる．ただし，A' と $z(A')$ は各 w, t について $l(A', w, t) = l(z(A'), w, t)$ である．

(証明) パーザ $z(A')$ の構成法を示す．まず，A' を 2.2.3 の前半で述べるように，各節点に対して先読み系列が一意に対応するように修正しておく．A' の pds の S トラックをなくし，あらたに V トラックを設けて，そのコマに $\bar{G}$ の（非終端および終端）記号を書く†．制御部の状態数をふやし，$z(A')$ はその状態でもとの A' の状態（節点名 L）と V トラックの内容の先頭の N（$\bar{G}$ のルールの右辺の長さの最大値）コマの中味 β を覚えているとする††．すなわち，状態は (L, β) の形とする．また，制御記号をふやし，制御記号を見ればそのコマを含めそれより左の $N-1$ コマ分の V トラックの中味がわかるようにしておく．$z(A')$ はスタック動作のとき，先読みで覚えているつぎの入力記号を V トラックにスタックし†††，それも状態で覚える．また，制御記号として，A' の書く制御記号 c とそのコマから左へ $N-1$ コマ分の V トラックの中味 β とをあわせて (c, β) と書いておく．$z(A)$ の先読み，スタック，簡約動作の時点は A' のそれと同じにする．A' の動作が簡約で，RED (n) CALL p とすると，$z(A')$ は状態から V トラックの先頭の n コマ分の中味 $Y_1 \cdots Y_n$ がわかる．$\bar{G}$ の条件 (2) より $\bar{r}(Y) = p$ なるルール $Y \to Y_1 \cdots Y_n$ は一意に定まる．そのルールの名前を r とすると $z(A')$ はルール r による簡約を行なう．スタックを $n-1$ コマ短くする．左のコマの制御記号が (c, β) なら，遷移先は (L', β') とする．ここで L' は c による A' の状態の遷移先であり，β' は βY（Y はルール r の左辺）の左端の記号をおとしたものである．この $z(A')$ が $\bar{G}$ に対するパーザであることは $\bar{G}$ の条件 (1) からあきらかで，また，$z(A')$ の先読み，スタ

---

† V トラックはなくてもよいが，説明の便宜上用いる．

†† スタックの長さが N より短ければ，その長さだけ．

††† A' の各節点には一つの先読み系列が対応しているので，つぎの入力記号が一つ定まる．

ック，簡約動作の時点はA'のそれと同じであるので，$z(A')$の先読みの長さもA'のそれに等しい．（証明終り）

2.3.2 構成法

上述の$\bar{G}$は，補題2.1より$T(G, k) = T_A$を満たすGとkを一つ構成し，このGから，Gと句構造的に等価な逆方向決定性文法の求め方[13]においてkで同じ値になる非終端記号だけをまとめることによって得られる．$\bar{G}$の非終端記号$\bar{X}$に対する$\bar{k}$の値は$\bar{X}$に属するGの非終端記号に対するkの値とする．補題2.2からわかるように，アナライザAと同じ先読みで$\bar{G}$に対するパーザがつくれるので，$\bar{G}$はLR(k)文法である．この$\bar{G}$に対する先読み最小のパーザZを，本論文のモデルを用いて構成し，出力をつけ加えてアナライザ$a(Z)$に修正すれば，定理より，それが求めるBになっている．先読み最小のパーザの構成法をつぎに示す．

先読み最小のパーザの構成法

$\bar{G} = (\bar{V}_N, \Sigma, \bar{R}, \bar{V}_S)$，$\bar{V}_S = \{S_1, S_2, \cdots, S_I\}$とする．$\bar{G}$に対し，$\hat{G} = (\hat{V}_N, \hat{\Sigma}, \hat{R}, \hat{V}_S)$を$\hat{V}_N = \bar{V}_N \cup \{S_0\}$，$\hat{\Sigma} = \Sigma \cup \{\dashv\}$，$\hat{R} = \bar{R} \cup \{S_0 \to S_j \dashv^k \mid 1 \le j \le I\}$，$\hat{V}_S = \{S_0\}$のようにつくる．$S_0$，$\dashv$は$\bar{V}_N \cup \Sigma$に属さない新しい記号とする．$\hat{V} = \hat{V}_N \cup \hat{\Sigma}$とおく．$\hat{R}$の元の個数を$\pi$とする．各ルールを1から$\pi$までの数で順序づけ，とくに$S_0 \to S_j \dashv^k$（$1 \le j \le I$）には$j$を割当てる．$r$番目のルールを$X_r \to X_{r1} X_{r2} \cdots X_{rn_r}$で表わす．よく知られている通常のLR(k)パーザの構成と同じ方法で，状態†qの集合と，各状態qにおける動作を決めるための長さkの先読み系列の集合$Z(q, r)$，$r = 1, \cdots, \pi$，$Z(q, ST)$をそれぞれ求める．この先読み系列の集合は，状態がqにあるとき，$Z(q, r)$に属する系列を先読みすればルールrによる簡約，$Z(q, ST)$に属する系列を先

---

† 一つの状態qは$[r, j, \alpha]$（rはルール番号，jは$0 \le j \le n_r$の整数，αは$\hat{\Sigma}$上の長さkの系列）の形の元のある集合．

読みすればつぎの入力記号の読込みを行なうように定める．状態$\mathcal{S}$の記号$Y \in \hat{V}$による遷移先の状態を$\mathcal{S}_Y$で表わす．初期状態は$\mathcal{S}_0 = \{[j, 0, \dagger^k] \mid 1 \leq j \leq I\}$である．

パーカのフローチャートはつぎのようにつくる．$\mathcal{L}$を$\Sigma^* \cup \Sigma^* \dagger$内の長さk以下の系列全体の集合とする．各状態$\mathcal{S}$と各$\alpha \in \mathcal{L}$について対$(\mathcal{S}, \alpha)$をつくり，フローチャートの節点とする．節点$(\mathcal{S}_0, \varepsilon)$を動作開始点とする．各節点$(\mathcal{S}, \alpha)$における動作はつぎのように定める．

(I) αが$Z(\mathcal{S}, 1) \cup \cdots \cup Z(\mathcal{S}, \pi) \cup Z(\mathcal{S}, ST)$に属するどの系列の初期部分系列でもないとき，拒否．

(II) (i) αをその初期部分系列としてもつ系列が$Z(\mathcal{S}, ST)$にあり，他の$Z(\mathcal{S}, \gamma)$，$1 \leq \gamma \leq \pi$には存在しないとき，STACK(C)．ここでCは節点$(\mathcal{S}, \alpha)$を識別できる記号（すなわち節点$(\mathcal{S}, \alpha)$の名前）とする．つぎは節点$(\mathcal{S}', \alpha')$へ移る．ただし，αの左端の記号をaとすると$\mathcal{S}' = \mathcal{S}_a$，$\alpha'$は$\alpha$から左端の$a$を除いた残りの系列．

(ii) αをその初期部分系列としてもつ系列が$Z(\mathcal{S}, \gamma)$にあり，他の$Z(\mathcal{S}, \gamma')$，$\gamma' \neq \gamma$，$Z(\mathcal{S}, ST)$には存在しないとき，ルールγによる簡約．簡約後に見る制御記号をCとし，Cが節点$(\mathcal{S}', \alpha')$を表わしているとすると，つぎは節点$(\mathcal{S}'', \alpha'')$へ移る．ただし，$\mathcal{S}'' = \mathcal{S}'_{X_\gamma}$，$\alpha'' = \alpha$．

(III) αを初期部分系列としてもつ系列が$Z(\mathcal{S}, 1)$，…，$Z(\mathcal{S}, \pi)$，$Z(\mathcal{S}, ST)$のうちの相異なる二つ以上の集合に同時に存在するならば，先読み．入力記号aを見れば，つぎは節点$(\mathcal{S}, \alpha a)$へ移る．

(IV) $\{[j, 1, \dagger^k] \mid 1 \leq j \leq I\}$の部分集合であるような$\mathcal{S}$について，節点$(\mathcal{S}, \dagger)$における動作は受理．

このフローチャートは一般に，動作開始点から到達できないようなむだな節点を含んでいるので，それを除去する．上のつくり方において，先読みは必要なだけしかしていない．すなわち，それより短い先読みでは正しい動作が決まらないような入力系列が存在するので，kに対するパーカのうちで，先読みが最小のものになっている．

§2.4 アナライザの簡単化

与えられたアナライザAと等価で，かつ，先読み最小のものの中で，フローチャートの節点数最小のものをBmとする．

［補題2.3］ Bmと2.3節で構成したBは，拒否するトークンの系列も含めてすべてのトークンの系列に対し，先読み，スタック，簡約，受理または拒否の全動作系列が互いに等しい．

（証明） もし，受理する系列に対する先読みが異なっていれば，BmまたはBの先読み最小性に反するので，先読みも含めて動作系列が同じでなければならない．Bの拒否の時点以前では，それまでに見た系列のあとに適当な系列をつければ受理に至るものが存在するので，Bよりはやく拒否すれば等価でなくなり，Bの拒否の時点以前で，スタック動作，簡約動作が異なれば等価でなくなる．また，Bの拒否の時点以前で先読みがBと異なれば，ある受理する系列に対して先読みが違うことになり，BmまたはBの先読み最小性に反する．拒否がBの時点より遅ければ，以後どんな系列でもいずれ拒否しなければならないような節点が存在することになり，節点数最小に反する．（証明終り）

したがって，Bとすべての入力に対して同じ動作をする†ものの中で節点数最小のものを見つければ，それがBmになっている．Bからこのようなを求めることは原理的には可能であるが，一般には複雑である．そこで，本論文では，簡約後の遷移先に関して上のBがつぎの条件Cを満たしている場合について考え，条件Cを満たすものの中でフローチャートの節点数最小のものを求める問題を考える．

［条件C］ 各制御記号に節点名が一意に対応していて††，かつ，簡約後の遷移先は，簡約後に見る制御記号とそのときの出力（数nとルーケン名p）の対で一意に決する．

---

† 制御記号は違ってない．

†† スタック動作のときその節点名を制御記号として書込むことにすれば満足される．

なお，一般の場合は，簡約後に見る制御記号と簡約の節点名の対で遷移先を定めている．いわゆる CALL，RETURN を用いて書けるプログラムは上述の条件を満たしている（制御記号のみで決まる）．

上の条件をいいなおせば，簡約 RED (n) CALL p の節点からラベル c の破線が節点 L へ出ていたら，その破線を除き，かわりに，c に対応する節点からラベルが (n, p) の枝を節点 L へ結んで得られる"遷移図"が決定性（すなわち，一つの節点から同じ (n, p) のラベルの枝が二つ以上の異なる節点へ結ばれない）であるということである（一般には非決定性になってしまう）．条件 C を満たしているなら，2.2 節で述べたフローチャートのかわりに，上述のような決定性の遷移図で，アナライザの動作を記述できる．通常，パーザはこの形で動作を記述し[8]，上の (n, p) のかわりに，そのとき簡約に用いたルールの左辺の非終端記号名で遷移先を決めている．

Pager は LR (k) パーザの状態数最小化の問題はこの遷移図を決定性有限オートマトンとみて，有限オートマトンの簡単化の問題に帰着されることを示した[8]．われわれの場合も Pager と類似の方法で遷移図を簡単化できる．Pager のパーザでは簡約のときの出力は簡約に使う G のルール名であるが，アナライザでは数 n とルーケン名 p の対 (n, p) である．また，われわれのモデルでは先読みは一コマずつ行なうようになっているが，"先読み最小の中で"という条件のため先読み動作も入力に対して固定しているので，先読み動作も有限オートマトンの出力と考えればよい．本質的には Pager のと同じであるので，詳細は省く．

なお，条件 C を考えない一般の場合は，非決定性の遷移図を簡単化しても，必ずしもアナライザの簡単化になっていない．アナライザをこの条件 C の範囲内で簡単化しても，$T(G, n) = T_A$ なるすべての G に対する最簡形のパーザ Z_G（簡約のときの出力をルール名でなく，ルールの右辺の長さと左辺の非終端記号名にした場合も含めて）から得られるアナライザ $a(Z_G)$ うどれよりも簡単なアナライザが得られる場合がある．2.2 節で述べた例がそうである．

§2.5 結言

コンパイラの構文解析部の本質的な機能を抽出し，陽に文法に依存しないアナライザのモデルを導入した．与えられたアナライザAを表現した文脈自由文法Gと写像hを求め，それよりAと等価で先読み最小のアナライザを構成出来ること，そのアナライザの簡単化の方法などを示した．

第2編　マイクロプログラム記述言語FMLとそのトランスレータ

§1　序言

本学情報工学科にある小型電子計算機UXは垂直型のマイクロ命令を備え，書き換え可能な制御用メモリをもったマイクロプログラム方式の計算機で，機械語命令を何ステップかのマイクロプログラム（マイクロルーチンとよぶ）として実行する．マイクロプログラム記述言語としてマイクロアセンブリ言語がメーカから提供されているが，この言語は2，3の命令を除いて，マイクロ命令と1対1に対応している．したがってマイクロプログラムをこのアセンブリ言語で書くには，ハードウエアの細部に関するかなりの知識を必要とする．

そこで，新しくフローチャート型のマイクロプログラム記述言語FML（Flowchart type Microprogramming Language）を設計し，そのトランスレータをコーディング中である．この言語のおもな特長は「ハードウエアに固有の制限事項をなるべく隠し，ユーザが覚えやすく，使いやすい」ことである．マイクロプログラムの効率化（長さと実行時間を短くすること）は重要であるので，トランスレータでは種々の最適化を試みている．トランスレータはPL/Iで書かれ，中型計算機で実行される．

マイクロプログラム記述言語としてはPL/Iに似た高級言語MPL[19]やサポートシステムCAS[20]，MPGS[21]で使われた言語が知られている．これらは汎用性を目指した言語である．特にCASはハードウエア設計のサポートシステムとしても利用される．これに対し，FMLはUXを対象とする比較的小規模な言語である．

§2 ハードウェアとマイクロ命令

ここでは，メーカから提供されているマニュアル[27]にもとづいて，UXのハードウェアおよびマイクロ命令について概説する．

2.1 ハードウェア構成

アセンブリ言語（機械語）でプログラミングするユーザからみたUXのハードウェア構成は図3.1の斜線を入れた部分である．

STR（status register）にはいわゆるPSW（program status word）を保持している．IRC（interruption code register）はUXの割込み表示レジスタである．汎用レジスタはGR0，GR1，…，GR7の8個あり，特にGR7は命令カウンタとして使われる．

マイクロ命令でプログラミングするユーザからみたマイクロプログラムプロセッサのハードウェア構成を図3.1に示す．

マイクロプログラムは通常制御用メモリCS（control storage）に格納される．CSの最大容量は4Kワード†で読み書きが可能である．マイクロ命令は1ワードで構成されており，CSへのアクセスも1ワード単位で行なわれる．ローカルメモリは図3.1に示すように4つの領域に分かれている．EX領域とSNI領域にはマイクロ命令を，LP領域には演算に使用する定数を格納している．SSA領域にはマイクロルーチンの開始番地を機械語命令ごとに用意している．各領域の容量は64ワードである．

MAR（microprogram address register）はマイクロ命令の命令カウンタでつぎに実行する命令の番地を示している．SCR（sequence control register）とDCR（data path control register）はともに命令レジスタで，CSやローカルメモリから読み出されたマイクロ命令が入る．AB（address buffer register），MB（memory buffer register）は主メモ

---

† 1ワード＝2バイト＝16ビット

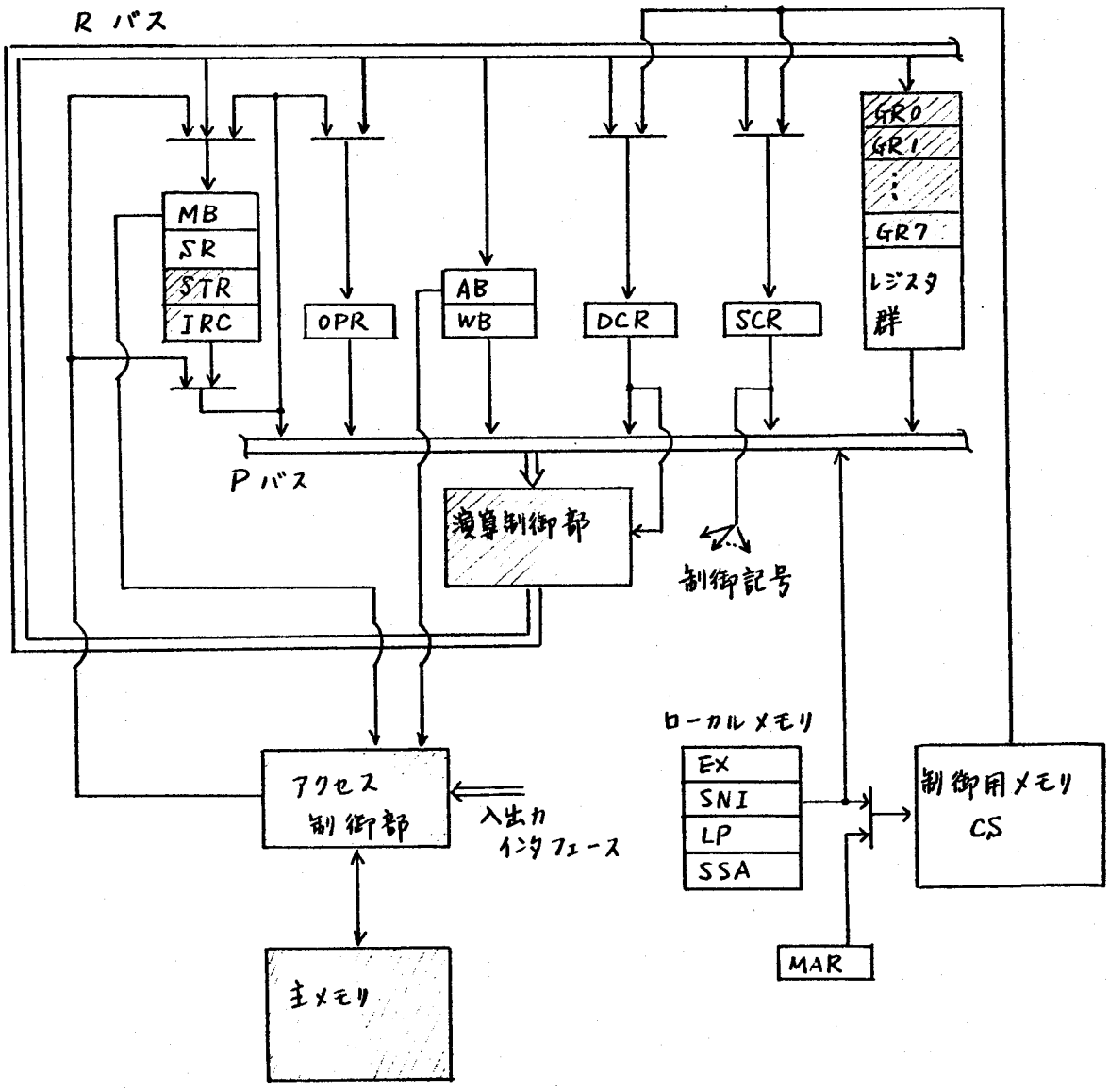


図 3.1　マイクロプログラムプロセッサ

りと入出力インタフェースにアクセスするためのレジスタで，それぞれ番地と読み取りまたは書き込みデータを格納する．OPR（operation regioter）はUXの機械語命令を保持するレジスタであり，WB（work buffer register）は一時記憶に用いられるレジスタである．AB，MB，OPR，WBは汎用レジスタとして使うこともできる．SR（system status register）でマイクロプログラムプロセッサの状態を保持している．他にレジスタ群として1ワードの定数レジスタC0000，C0001，C0002，C0004，C0007，C00F0，C2000，CFFFFなどがある．

2.2 マイクロ命令

UXのマイクロ命令はSCIとDCIの2種にわけられ，前者はマイクロ命令の実行順序の制御やメモリへのアクセス（読み取り，書き込み）のタイミング制御を，後者は各種の演算を行なう命令である．ここではマイクロアセンブリ言語の表現（mnemonic）をかりて各マイクロ命令の機能を概説する．

(1) SCI

分岐命令としてB（branch），BAL（branch and link），BC（branch on condition）がある．BALでは戻り番地がWBで保持される．BCではブロック（256ワード単位の領域）外へは分岐できない．

SNI（select next instruction）は条件部で指定する条件が成立していないときはこの命令のつぎのマイクロ命令を，成立しているときはその代わりにSNI領域の指定された番地のマイクロ命令を実行する．

コントロール命令にはつぎに実行するマイクロルーチンの開始番地をSSA領域から読み出すSSA（set start address），主メモリに読み取り要求を出すAREQ（access request），主メモリに書き込み要求を出すWRITE，読み取りや書き込み要求命令の完了の確認を行なうWAIT，CSへの書き込みを行なうWCS（write control storage），STRへの書き込みを行なうSCC（set condition code）などがある．

(2) DCI

転送命令 MV (move), LP (load pattern), 算術演算命令 A (add), AMA (add and move to AB), AC (add carry), S (subtract), C (compare), 論理演算命令 AND, OR, EOR (exclusive or), シフト演算命令 SR (shift right), SL (shift left), SRD (shift right double), SLD (shift left double) と特殊命令 EX (execute) がある. DCR と MAR を除くすべてのレジスタがオペランドとして許されている. LP はオペランド部で指定する番地の LP 領域から1ワードの定数を読み取る. AMA は主メモリへアクセスするときの番地計算に便利な命令で, 加算結果をオペランドのレジスタと AB に同時に格納する. SRD と SLD はともにオペランド部で指定する2つのレジスタを連結してシフト演算を行なう. EX は OPR 内の機械語命令コードで指定される EX 領域からマイクロ命令を読み出し実行する.

2.3 マイクロ命令の実行

マイクロプログラムプロセッサのサイクルタイムを T とおくと, マイクロ命令の実行時間は SCI が 2T, DCI が 2T あるいは 4T である.

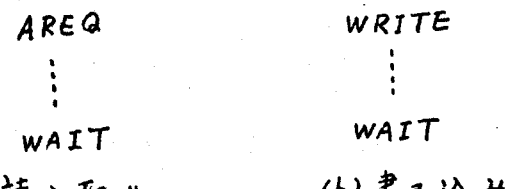
ここで注意すべき点は, 4T の DCI のつぎの番地に SCI が置かれた場合, この2つの命令は並列に実行される. また, 命令の実行順序を変更する命令 (たとえば分岐命令) を実行する場合, その命令の次の番地にある命令を実行したあとで, 分岐先の命令に制御を移す.

2.4 主な制限

UX のハードウェアに置かれている種々の制約はマイクロアセンブリ言語ではたとえばオペランドの使用制限となって現われて, アセンブリ言語を使いにくいものとしている. FML ではこうしたハードウェアに固有の制限をユーザから隠しているが, 詳しくは後述する. ここでは, アセンブリ言語上の主な制限事項を以下に示す.

C1. AB, OPR, WB, MB, SR, STR, IRC をオペランドとして用いる DCI のつぎの番地に SCI の AREQ または WRITE を置いてはいけない.

C2. 主メモリからの読み取り（書き込み）は図3.2に示すように AREQ（WRITE）と WAIT の対で行なわれる.



(a) 読み取り　　(b) 書き込み

図3.2 主メモリへのアクセス

C3. 主メモリからの読み取りおよび書き込みでは，主メモリの番地，データを入れるのにそれぞれ特定のレジスタ AB, MB を使用しなければならない.

C4. AREQ または WRITE の実行から WAIT の実行までの期間では MB, SR, STR, IRC をオペランドとして使用する DCI の実行は禁じられている.

C5. CS への書き込み命令 WCS では C3. と同様，番地とデータをそれぞれ特定のレジスタ AB と MB に格納することが必要である.

§3 FML

3.1 FML設計の基本方針

言語設計にあたって，プログラムの生産性を高め，しかも発生されたマイクロプログラムの効率を悪くしないように注意した．そこで設計の目標を

(1) ユーザが覚えやすく，使いやすく，しかもコーディングが見やすい．

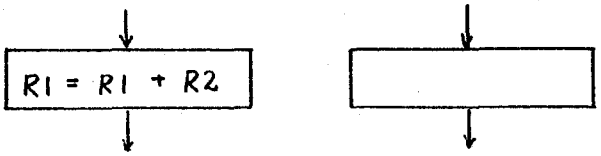
こととした．さらにマイクロプログラムでは特に効率も重要なので

(2) 汎用性を与えるためハードウェアに固有の制限事項をユーザから隠すが，便利な機能は（かなりハードウェアに依存していても）積極的に残す．

という立場をとった．(1) と (2) は一般に矛盾する要求なので，かねあいが問題になる．

3.2 FML の文

FML ではプログラムはいくつかの文から構成され，文はフローチャート記号の中に特定の文字系列（表現とよぶ）を書いたものとする．フローチャート記号は文の種類を表わし，表現はその機能を詳細に記述している（図 3.3 ）．フローチャート記号の一覧表を表 3.1 に示す．



(a) 代入文　　(b) フローチャート記号　　(c) 表現

図 3.3　代入文

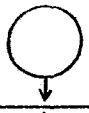
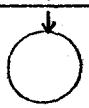
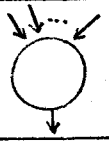
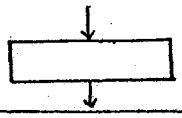
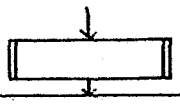
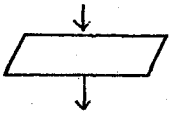
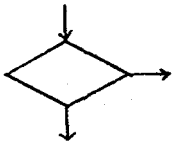
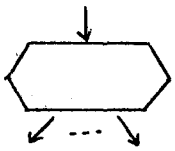
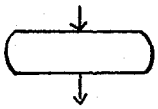
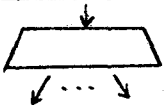
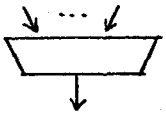
フローチャート記号	名前	機能
	メイン	主プログラムの入口を示す.
	入口	サブルーチンやマクロの入口を示す.
	出口	プログラムの出口を示す.
	コネクタ	分岐の入口，合流および無条件分岐を示す.
	代入	代入が行なわれることを示す.
	コール	サブルーチンコールやマクロコールが行なわれることを示す.
	入出力	メモリへのアクセスが行なわれることを示す.
	判定	判定結果にもとづいた2方向分岐が行なわれることを示す.
	スイッチ	多方向分岐が行なわれることを示す.
	宣言	パラメータを宣言することを示す.
	フォーク	並列文の開始を示す.
	ジョイン	並列文の終了を示す.

表3.1　フローチャート記号の一覧表

ここではプログラムを形式的に定義することはやめ，FML 特有の文についてのみ説明する．FML ではオペランドとして

① タイプ1レジスタ … マイクロ命令で使えるレジスタ（AB, MB, OPR, C0000, … ）をすべて含む．

② タイプ2レジスタ … 仮想的な16ビットレジスタで R0, R1, …, R9, RA, RB, RC のいずれかとする．

③ 連結レジスタ … 16ビットレジスタを連結したもので，R と R′ を16ビットレジスタとするとき R′R′ と書く．

④ 部分レジスタ … 16ビットレジスタの連続する4ビット以下の部分で，レジスタRのiビットからjビットまでの部分レジスタを R(i−j) と書く．

⑤ 定数 … 任意の16ビットの定数を Hxxxx（xは16進数とする）の形で書ける．

⑥ ビット列 … {0, 1, *}（* は don't care を意味する）の上の系列で，その長さは4以下である．

を許している．

(1) 代入文（図3.3）

代入文で許す表現はマイクロ命令 DCI とほぼ1対1に対応している．以下で opnd, opnd1, opnd2 はいずれも16ビットレジスタとする．

代入文で許す表現	対応するマイクロ命令
opnd1 = opnd2	MV
opnd1 = SR(opnd2)†	SR, SRD
opnd1 = SL(opnd2)	SL, SLD
opnd1 = opnd1 & opnd2	AND
opnd1 = opnd1 | opnd2	OR
opnd1 = opnd1 # opnd2	EOR

† 対応するマイクロ命令が SRD（SLD）の場合は，opnd1 と opnd2 は同一の連結レジスタを表わす．

```
    opnd1 = opnd1 + opnd2          A
    opnd1 = opnd1 + opnd2 @        AC
AB, opnd1 = opnd1 + opnd2          AMA
    opnd1 = opnd1 - opnd2          S
```

(2) 入出力文 (図3.4)

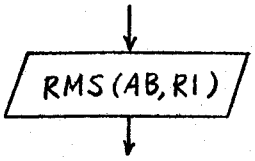


図3.4 入出力文

入出力文ではメモリへのアクセスを1つの命令で書けるようにし前述の制限C2. を解決している. C3., C5. についても, 番地はABに入れる必要があるが, データの置かれるレジスタは16ビットレジスタであれば注意でよい. 主メモリについては特に1バイト単位のアクセスも可能である.

入出力文で許す表現	機能
RMS (AB, opnd)	主メモリからの読み取り
WMS (AB, opnd)	主メモリへの書き込み
RBMS (AB, opnd)	1バイト単位の読み取り
WBMS (AB, opnd)	1バイト単位の書き込み
WCS (AB, opnd)	CSへの書き込み
WCC	STRへの書き込み

こうしてAREQ, WRITE, WAITをユーザから隠すことによって, FMLではC1. とC4. で述べたオペランドの使用制限を除いている.

(3) 判定文 (図3.5)

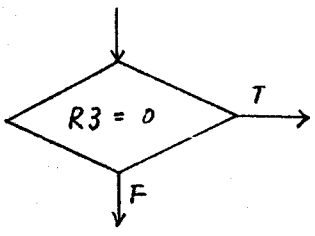


図3.5 判定文

判定文では2項の論理式を直接書ける．16ビットレジスタ，フリップフロップの内容の判定以外に，部分レジスタの内容があるビット列に等しいかどうかの判定もできる．

判定文で許す表現	機能
opnd1 θ opnd2 †	レジスタの内容の比較
CFF = 1	フリップフロップの判定
ZFF = 1	フリップフロップの判定
R(i-j) = w	部分レジスタの内容とビット列wの比較

ここでθは比較演算子とする．プログラムの中では通常図3.6(a)の一般形で使うが，マイクロ命令SNIに対応した同図(b)のSNI形も許す．LBL1，LBL2はラベルとする．

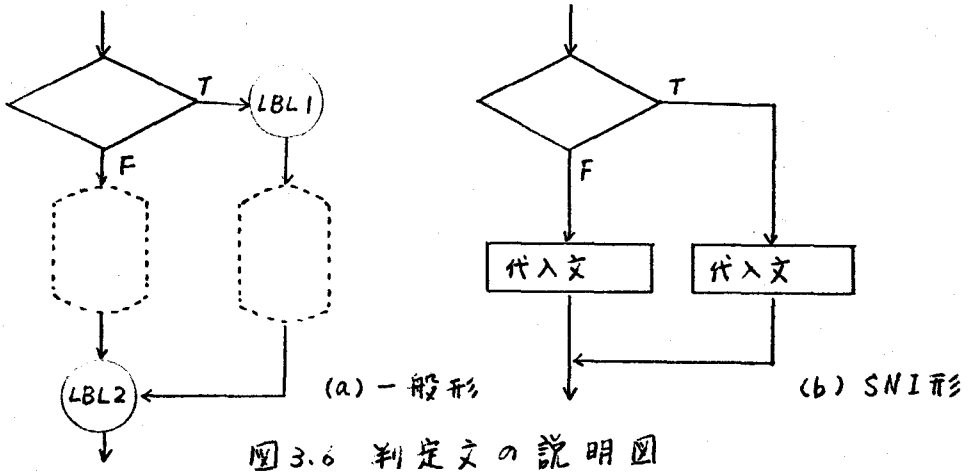


図3.6 判定文の説明図

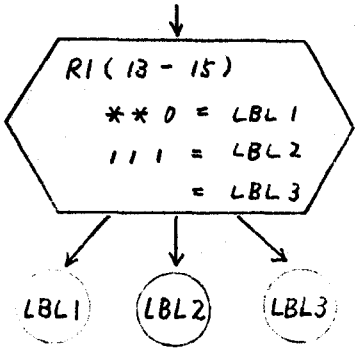


図3.7 スイッチ文

---

† opnd2 が定数レジスタ C0000 となることが多いので，見やすさのため，その場合特に表現 opnd1 θ 0 を許している．

(4) スイッチ文（図3.7）

部分レジスタの内容にもとづいて多方向分岐する．ビット列を w_i，ラベルを LBL i と書くとき，スイッチ文で許す表現は

R(i-j)
w_1 = LBL 1
w_2 = LBL 2
⋮
w_n = LBL n

である．部分レジスタ R(i-j) の内容が，w_i 中の don't care のビット位置を除いて，w_i と一致すれば LBL i へ分岐する．各ビット列は互いに排他的に指定しなければならないが，ラベルはその必要がない．特に w_n だけ空系列を許し，レジスタの内容が w_1, w_2, …, w_{n-1} のいずれとも一致しないとき LBL n へ分岐する．

(5) フォーク文，ジョイン文（図3.8）

フォーク文，ジョイン文ともに表現としてラベル LBL を許す．同じラベルをもつフォーク文とジョイン文，さらにその間の部分（並列1，並列2，…，並列n）をまとめて並列文とよぶ．実用上の制限として並列文のネストを禁じ，並列も2つ（並列1と並列2）にした†．

3.3 プログラム例

FML プログラム例として機械語命令 MULT に対するマイクロルーチンを図3.9に示す．MULT はそのオペランド部で指定される汎用レジスタ GRD の内容を被乗数 P（最上位ビットは符号）とし，インデックス修飾の計算をして求まる実効番地から読み込まれる1ワードデータを乗数 Q として，符号付2進数乗算を実行する．積は2ワードデータで

† 入出力文では番地を入れるため AB が必要となるので，2つ以上の並列が入出力文を含むことは許されない．

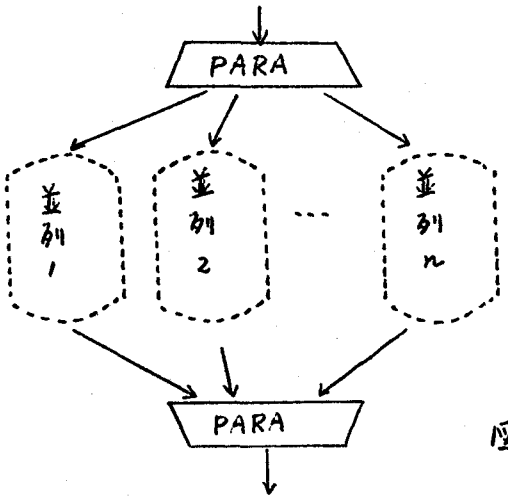
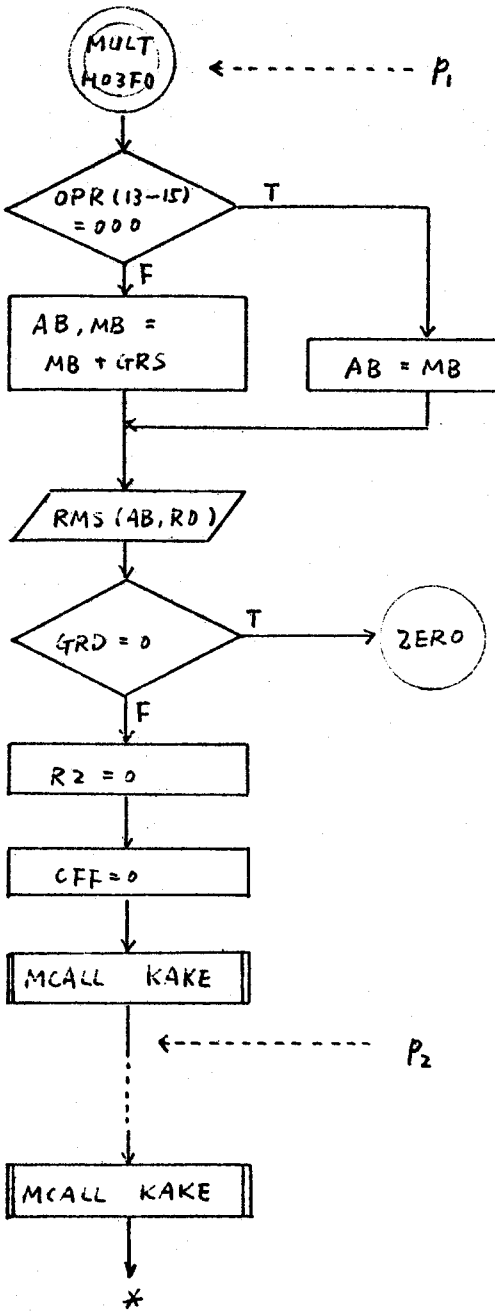


図 3.8　フォーク文とジョイン文

で求められ，GRD と GRDO に格納される．図3.9は紙面の都合で Q > 0 を前提とした場合のプログラムとなっている．

図3.10には図3.9上で p_1 と p_2 の間のプログラムを，マイクロアセンブリ言語で書いたコーディング例を示している．2つのプログラム例からもわかるように，FML はマイクロアセンブリ言語に比べ，かなり使いやすく，しかも見やすくなっている．



H03F0 はマイクロルーチン MULT の開始番地を指定する.

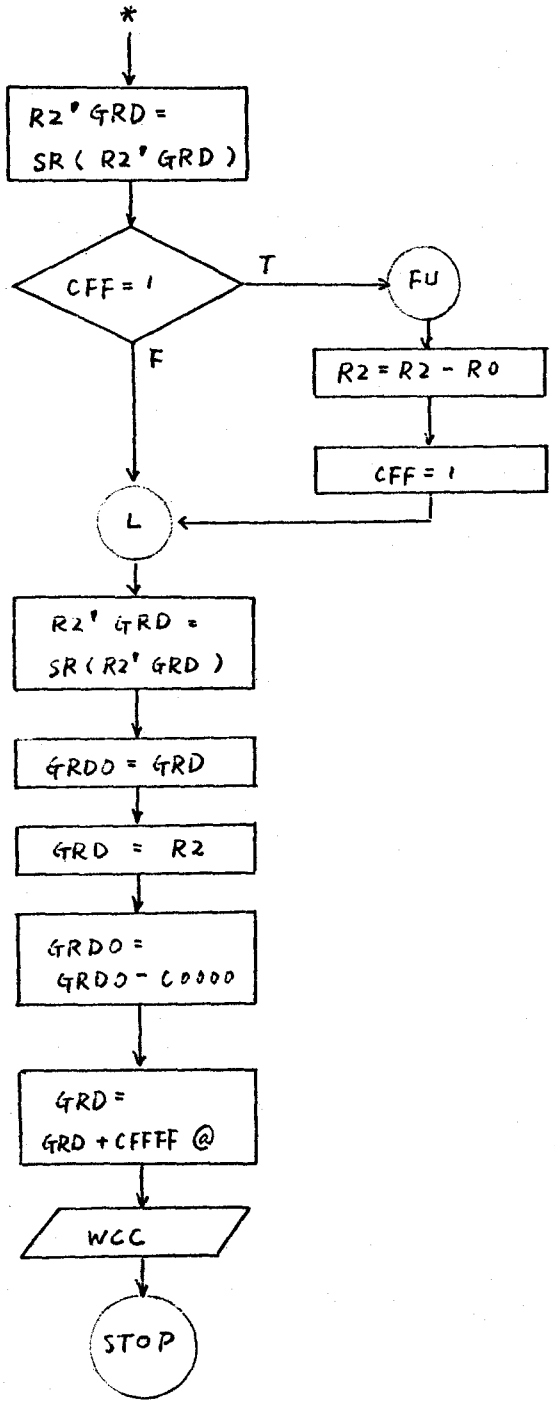
OPR の 13～15 ビットが 000 なら MB の内容を，そうでなければ MB と GRS の和をそれぞれ実効番地として AB に格納する.

乗数 Q を RD に読み込む.

GRD に入っている被乗数 P が零かどうかの判定

クリアしておく

中間結果を 1 ビットシフトしながら，乗数 Q を加える操作を，マクロコールの形で，15 回実行する.

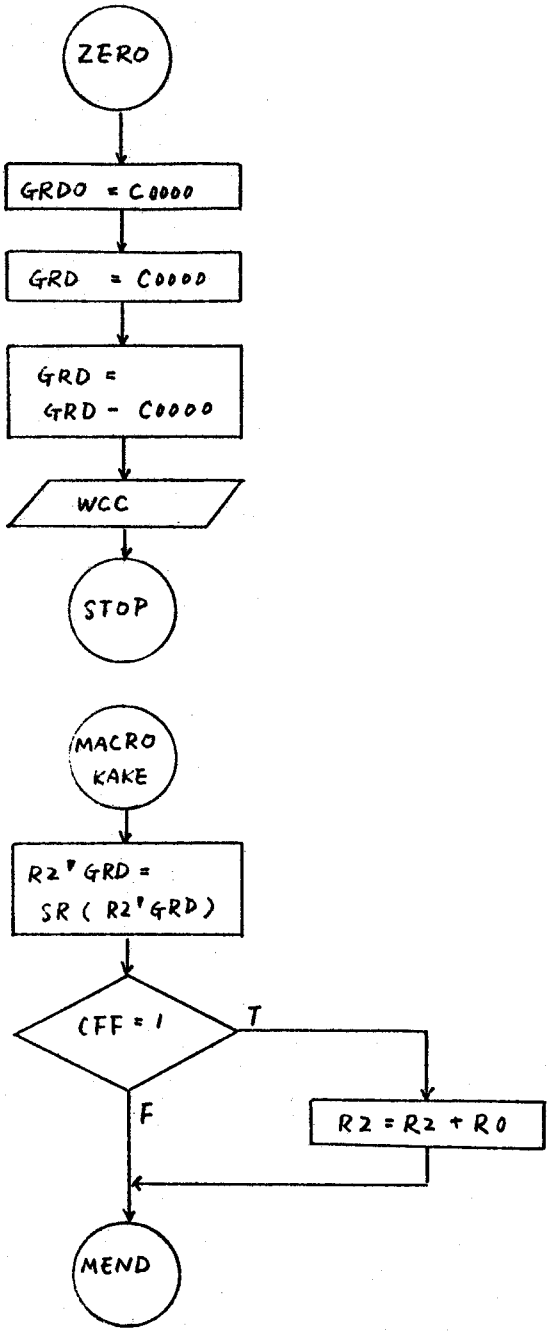


被乗数Pの符号をCFFにとり出す。

その符号ビットが1(負)なら、求まっている積をその補数表現に変換する。
0(正)ならそのまま。

積をGRDOとGRDに格納する。

積の符号をコンディションコードに書き込む。



被乗数 P が零の場合，積も零となるので，GRDO，GRD に C0000 を転送する．

積が零であることをコンディションコードに書き込む．

KAKE という名前のマクロの定義．

中間結果を 1 ビットシフトしながら，乗数 Q を加える操作．

図 3.9　FML プログラム例

```
MULT    ORG     00 111111 0000
        MV      OPR , AB
        AND     C0007 , AB
        SNI , Z 000000
        AMA     GRS , MB
        NOP
        AREQ
        C       C0000 , GRD
        BC , Z  ZERO
        WAIT
        MV      C0000 , AB
        SL      C0000 , CT
        SRD     AB , GRD
        SNI , C 001010
        NOP
```

OPR の内容を保存しておく必要があるので，OPR の内容を AB に転送し，AB 上で 13～15 ビットの判定を行なっている．

制限 C1. より NOP が必要．

この 2 命令で，GRD の内容が零かどうかの判定を行なう．

上の AREQ と この WAIT の対により (C2.)，乗数を MB に読み込む．

CFF を 0 にクリアするための命令．

SNI 領域の 001010 番地には マイクロ命令（A MB，AB）が格納されている．

図 3.10 マイクロアセンブリ言語で書いたコーディング例

§4 FMLトランスレータ

4.1 トランスレータの基本構造

FML プログラムの処理過程を図3.11に示す．トランスレータはフローアナライザ，オプティマイザとジェネレータから構成される．今回の機構化では，FML プログラムは一定の規則に従ってカードに穿孔しなければならない．原則的に1つの文（フローチャート記号）が1枚のカードに対応し，図3.12のような形で穿孔する．

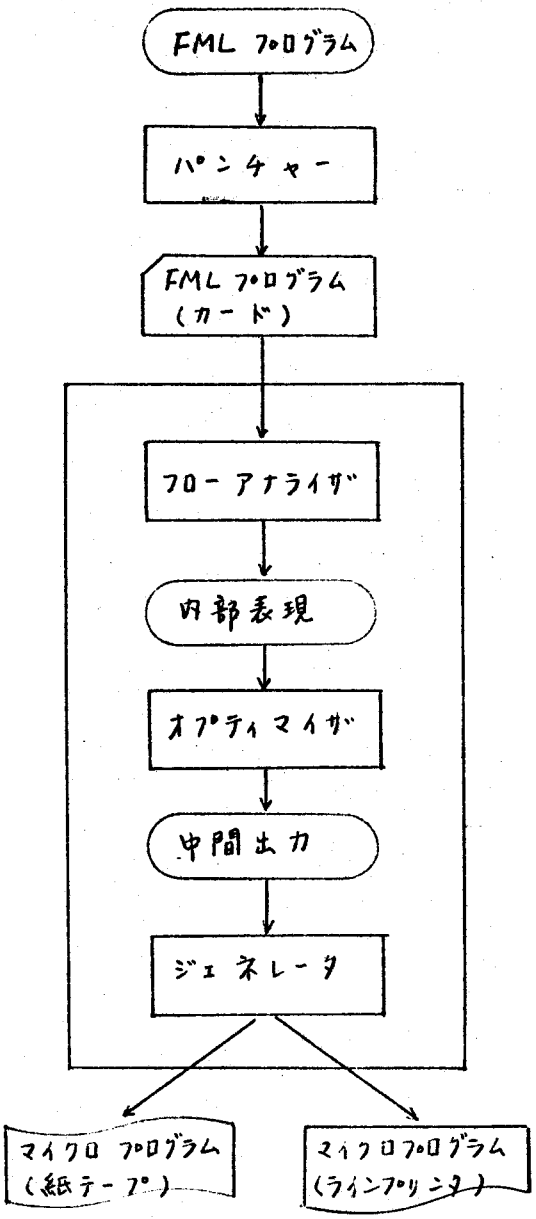


図3.11 FMLトランスレータ

制御フィールド	ラベルフィールド	タイプフィールド	ステートメント フィールド
*	FML	PROGRAM	
	MULT	MAIN	H03F0
		IF	OPR (13 - 15) = 000
		T	AB = MB
		F	AB, MB = MB + GRS
			RMS (AB , R0)
		IF	GRD = 0 , T = ZERO
			R2 = 0
			CFF = 0
		MCALL	KAKE
		⋮	⋮

図 3.12 ソースリスト

4.2 フローアナライザ

フローアナライザでは語い解析，構文解析，マクロ処理，（プログラムコントロール）フローの解析を行なう．フローの解析ではオプティマイザが必要とする情報，たとえばデータの依存関係や制御の流れなどを集めている．先ずプログラムを基本ブロックに分割しプログラムコントロールフローグラフを求めたあと，いわゆるインタバル解析を行なっている[22]．

4.3 オプティマイザ

オプティマイザは内部表現されたFMLプログラムをマイクロアセンブリ言語のプログラムに変換するとともに，物理的な番地付けとは独立に行なえる最適化を試みている．実行順に述べると

① 結果が使われない計算の除去．
② 重複した計算の除去．
③ ループに独立な計算をループ外に出すこと．
④ スイッチ文の展開．
⑤ 分岐命令の直後のNOPを減らすこと．
⑥ タイプCレジスタに対するレジスタ割付け．
⑦ AREQ（またはWRITE）とWAITの位置決め．
⑧ 並列文の処理．

となるが，ここではFMLに固有な問題④～⑧についてだけその概要を述べる．

④では，指定されているビット列の発生や比較も含めプログラムが短くなるよう判定順序を決め変換する．UXの8個の定数レジスタとLP領域にすでにある57種類の定数を利用すると，任意のビット列（任意の位置の連続した4ビット以下の列で，残りの部分はすべて0）が3命令以下でレジスタに発生できる．判定方法としては，1ビットごとの比較による判定（A_1），ビット列ごとの比較による判定（A_2），指定されたレジスタの内容を分岐命令のインデックスとして直接分岐する方法（A_3）が考えられる．今回の最適化では，A_3は絶対番地を必要とするので考慮外とし，明らかにA_1が最適となる場合を除いてA_2を採用している．一般にはA_1は「決定表」の問題[25]に帰着される．A_2は，以下のように考えれば，「巡回セールスマン」の問題[26]に帰着される．

たとえば，発生したいビット列がw_1，w_2，w_3（図3.13(a)）であるとき，まず，同図(b)のようなグラフを作る．ここでw_0はLP領域にある57種類の定数を表わし，各節点間の枝のラベルc_{0j}はLP領域にある定数のみを利用して，c_{ij}（$\neq 0$）は（現在レジスタ中にある）ビ

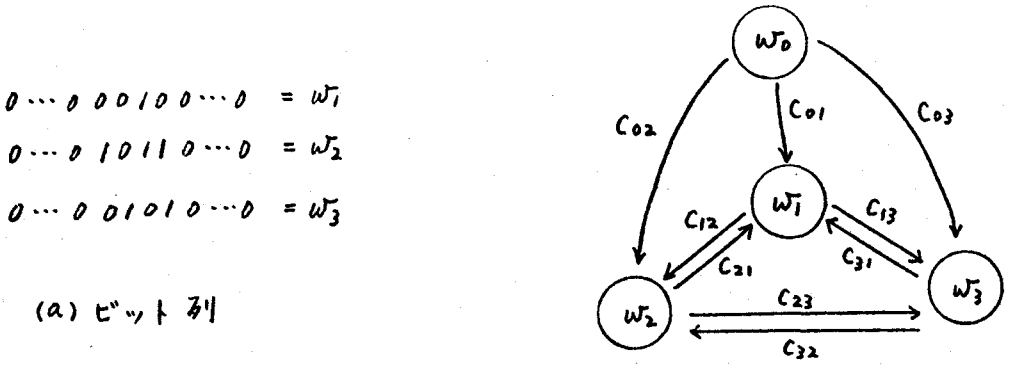


図3.13　スイッチ文の展開の説明図

ット列 w_2 も利用できるとして，w_3 を生成するのに要る命令数の最小値を表わしている．すると A_2 の問題は，w_0 から出発して，すべての節点を一度ずつたどる道の中で，その道に沿って求めた c_{ij} の総和が最小である一つの道を見つける問題となる．現在のところ，この解を能率よく求めることは困難である(24)．

UX の分岐命令はつぎの番地の命令を実行してから分岐するので，通常は，分岐命令の直後に NOP を置く．しかし条件付分岐で分岐する（あるいは分岐しない）ときのみ実行する命令 I を条件付分岐命令の前で実行しても結果に影響がないなら，NOP を I で置き換え，もとの I を省略できる．無条件分岐命令についても同様にして NOP の有効利用を行なう．

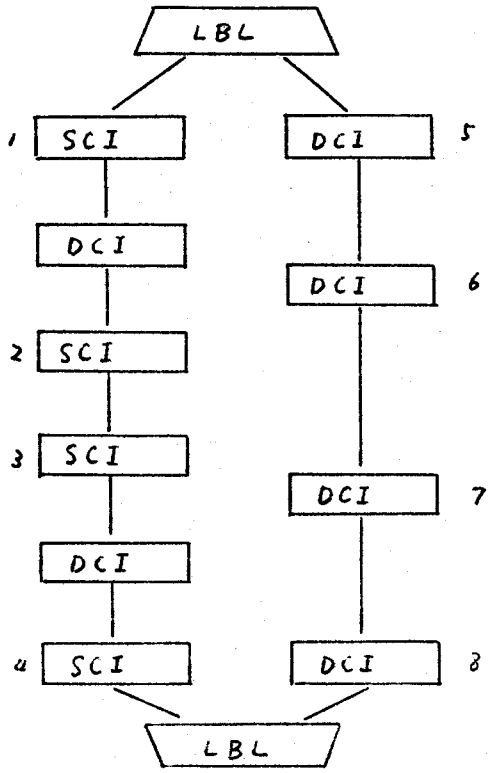
⑥では，13個の仮想レジスタとスイッチ文の展開などで作業用として必要な3個のレジスタに対し，UX に備わっているレジスタ† を割付ける．レジスタが不足するときは LP領域の空いた部分を一時記憶域として利用する．"退避のためのメモリへのアクセス回数の総数を最小にするよう" 試みているが，今回機構化したのは局所的な範囲内での最小化である．

---

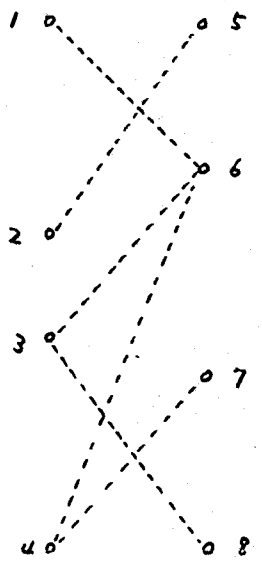
† AB，MB，OPR，WB 以外に9個のレジスタがある．

⑦は入出力文をアセンブリ命令に展開するときの問題である．入力文の場合，通常は AREQ の直後に WAIT を置いた形に展開するが，そのとき一般に WAIT で"待ち"を生じる．そこで AREQ の前あるいは WAIT の後で実行する命令で AREQ と WAIT の間で実行可能なものがあれば，それを AREQ と WAIT の間に移し"待ち時間をできるだけなくするよう"する．出力文の場合も同様にして，WRITE と WAIT の位置を決定する．

⑧では，並列文に含まれる命令の実行順序を決める際，"並列1と並列2の間で，並列に実行される命令の総数が最大となるよう"試みている．たとえば，並列文が図3.14(a)の形で与えられ，並列1と2の命令対で並列実行可能なものを破線で結んで同図(b)が得られたとする．⑧での問題は同図(b)で，マッチングを示す線が互いに交わらない「最大マッチング」を求める問題に帰着されるが，現在のところ，この解を能率よく求めるアルゴリズムは知られていない．



(a) 並列文



(b) 並列実行可能な命令対

図3.14 並列文の処理の説明図

4.4 ジェネレータ

ジェネレータではプログラムに絶対番地を割付け，条件付分岐命令でブロック外へ飛びたい場合の分岐命令のうめこみを行なう．出力はビットパターンで紙テープに，マイクロアセンブリ言語でラインプリタにそれぞれ出す．

§5 結言

今回設計したFMLでは一応，プログラム例からわかるように，所期の目標を達成していると思われる．さらに，言語の仕様や最適化の方法など十分検討しているので，マイクロプログラムの効率についてもかなり満足のいくものが期待される．残されている問題としては，4.3で述べた最適化のアルゴリズムの改善，拡張がある．特に，マイクロプログラムプロセッサではメモリへのアクセス回数の総数を最小にすることは重要であるので，レジスタ割付けの問題を全域的な範囲で解くことが要求される．さらに，ディスプレイを用いてFMLプログラムを直接入力できるようにすることも今後の問題の一つである．

結　論

本研究によって得られたおもな結果および今後に残された問題を，簡単にまとめると次のようになる．

第1編第1章では，文脈に関する情報（つぎに来うるトークン名の系列）をパーサから教えてもらい，それを利用して文字系列をトークンに区切る方法を提案し，それに関する諸性質を述べた．簡単化に関連して，つぎのような問題が残されている．(1) 入力系列の拒否をパーサでも行なうことにして，k-区分可能性の条件をそこなわない範囲で，パラメータQをQ'（Q'⊇Q）で適当におきかえて，全体としてパラメータの個数を減らすこと．(2) パーサからスキャナにわたされるパラメータを固定して，スキャナの部分を簡単化すること．

第1編第2章では，構文解析部の本質的な機能だけに着目し，そのモデルとして先読みをもつアナライザを導入した．アナライザの等価性，簡単化の方法などについて述べ，さらに，LR(k)パーザで考えるより本論文のアナライザで考えるほうが簡単化に関しては一般に有利であることを示した．残されている問題としては，入力系列の拒否をセマンチックルーチンの中でも行なうとした場合のアナライザの等価性および簡単化などがある．

第2編では，マイクロプログラム記述言語FMLの仕様とそのトランスレータの機構化について述べた．マイクロプログラムは，その性質上，プログラム自体の生産性よりも効率を重視するのが普通である．したがって，FMLトランスレータで行なっている種々の局所的な最適化を全域的なものにすることが残されている大きな問題である．さらに，FMLをより便い勝手の良いものにするためには，ディスプレイ装置を用いたデバッギングシステムの開発が今後の課題である．

謝　辞

本研究に関して，直接理解ある御指導を賜わり，つねに励ましていただいた嵩 忠雄教授に心から深謝します．

大学院修士および博士課程において，御指導いただいた情報工学科藤沢俊男教授，田中幸吉教授，木沢 誠教授，制御工学科桜井良文教授，坂和愛幸教授，辻 三郎教授に心から感謝します．

大学院を通じて御教示，御指導いただいた都倉信樹助教授，的場 進助教授，志村正道助教授，豊田順一助教授に心から感謝します．

本研究の全過程を通じて終始，適切な御指導と御助言をいただいた谷口健一助手に心から感謝します．

種々の面で御助言，御鞭撻いただいた藤井 護講師，細見輝政助手，吉岡信夫助手，神戸商船大学中村圭二郎助教授，電子技術総合研究所鳥居宏次博士に厚くお礼を申し上げます．

いろいろ御助言，御援助いただいた富士通新海卓夫氏，神田泰典氏，佐藤清澄氏に厚くお礼を申し上げます．

また，修士および博士課程を通じて，ともに研究し励ましあってきた学友安積三朗氏，奥井 順氏らの有益な御助言，御討論に感謝します．

さらに，筆者の在学中，御討論いただいた嵩研究室の方々，とくに第2編においてプログラミングに御協力いただいた杉山裕二氏，坪倉 孝氏，現日本航空井上哲次氏に感謝します．

文　献

(1) D.E. Knuth : "On the translation of languages from left to right", Inform. Control, 8, 5, P. 607 (Oct. 1965).

(2) J.E. Hopcroft and J.D. Ullman : "Formal languages and their relation to automata", Addison-Wesley (1969).

(3) R.E. Stearns : "A regularity test for pushdown machines", Inform. Control, 11, 3, P. 323 (Sept. 1967).

(4) 近藤，菊野，谷口，嵩："語い解析と構文解析について"，信学会オートマトンと言語研資（1972－07）.

(5) 菊野，谷口，嵩："語い解析に関する一考察 ── 一方向有限オートマトンによる正規集合間の区切りについて ──"，昭47電気関係学会関西支部連大，G9－21.

(6) 谷口，菊野，嵩："文脈を利用した語い解析"，信学会オートマトンと言語研資（1973－06）.

(7) 谷口，嵩，菊野："文脈を利用した語い解析"，信学論（D），57－D，4，P. 228（昭49－04）.

(8) D. Pager : "A solution to an open problem by Knuth", Inform. Control, 12, 5, P. 462 (Dec. 1970).

(9) F.L. DeRemer : "Simple LR(k) grammars", Comm. ACM, 14, 7, P. 453 (July 1971).

(10) 林："CFG－PL変換について"，情報処理，12，3，P. 145（1971－03）.

(11) 雨宮："LR(k) parserについて"，昭46情報処理学会大会，115.

(12) 関本："LR(1)文法の諸性質とそれに基づくparserの構成法"，情報処理，13，3，P. 170（1972－03）.

(13) R. McNaughton : "Parenthesis grammars", J. ACM, 14, 3, P. 490 (July 1967).

(14) たとえば，藤井，嵩："単純句構造文法の構造的等価性および句構造を保存する変換"，信学論（C），52－C，1，P. 56（昭44－01）.

(15) D. E. Knuth : "Semantics of context-free languages", Math. Systems Theory, 2, 2, P. 127 (June 1968).

(16) A. V. Aho and J. D. Ullman : "A technique for speeding up LR(k) parsers", 4th Ann. ACM Symp. on Theory of Computing, P. 251 (May 1972).

(17) 谷口, 菊野, 嵩 : "構文解析の簡単化について", 信学会オートマトン研資 (1972-01).

(18) 谷口, 嵩, 菊野 : "アナライザの等価性と簡単化", 信学論(D), 56-D, 7, P. 424 (昭48-07).

(19) R. H. Eckhouse, Jr : "A high-level microprogramming language (MPL)", SJCC, 38, P. 169 (1971).

(20) S. S. Husson : "Microprogramming : Principles and Practices", Prentice-Hall (1970).

(21) M. Hattori, M. Yano & K. Fujino : "MPGS : A high-level language for microprogram generating system", Proc. ACM, P. 572 (Aug. 1972).

(22) K. Kennedy : "A global flow analysis algorithm", Intern. J. Computer Math., 3, P. 5 (1971).

(23) 菊野, 杉山, 安積, 谷口, 嵩 : "あるマイクロプログラム記述言語 FML の設計", 昭49信学全大, 1509.

(24) 杉山, 安積, 菊野, 谷口, 嵩 : "FML トランスレータにおけるマイクロ命令の最適化について", 昭49信学全大, 1510.

(25) P. J. H. King : "Conversion of decision tables to computer programs by rule mask techniques", Comm. ACM, 9, 11, P. 796 (Nov. 1966).

(26) F. Harary : "Graph theory", Addison-Wesley (1969).

(27) 富士通 : "FACOM U300 マイクロプログラムプロセッサ解説書".

(28) 菊野, 杉山, 安積, 谷口 : "あるマイクロプログラム記述言語 FML とそのトランスレータについて", 情報処理(投稿中).