ソフトウェア基礎論　27－2
プログラミング言語　19－2
（1988. 12. 9）

並列リダクション戦略の実現を目的とするデータフローアーキテクチュアについて

A Dataflow Machine Architecture for Parallel Reduction

国持　良行　　関本　彰次

Yoshiyuki KUNIMOCHI　Shouji SEKIMOTO

静岡大学工学部

Faculty of Engineering, Shizuoka University

あらまし　一定の並列リダクション戦略よる式の評価をデータフローマシンの上で実現するためのデータフローアーキテクチュアについての一提案を行なう。通常、グラフリダクションマシンにより式の評価をする場合、式の遅延評価、式の共有、関数の効率の良い再帰呼出し、及び記憶域の再利用等は比較的容易に実現されるが、データフローアーキテクチュアを用いてそれらを実現する方法についてはあまり知られていない。本研究では、グラフリダクションマシンで実現されるそれらの機能を効果的にデータフローマシンで実現することを試みる。

Abstruct　We propose an architecture for the dataflow machine which evaluates λ-expressions by using a parallel reduction strategy. The dataflow machine realizes effective recursive function call, and the techniques, usually implemented in graph reduction machines, such as lazy evaluation, shared expressions and garbage collection.

1．まえがき

関数型言語の並列処理系の一形式として(並列)グラフリダクションマシン(ＧＲマシン)がある。所で、ＧＲマシンでは、変数への引数の配分、変換可能な式(リデックス)を特定の戦略に基づき選出するのに必要な式の走査、更には上記の処理及びリデックスの変換時におけるリスト処理に伴なう記憶域管理等に通常多くのオーバヘッドを費やす難点が見られ、その解決が必要とされている。本研究では、上記の問題点の改善を計かるための一つの方法として、特定のリダクション戦略[1]をとるＧＲマシンに着目し、そのマシンが式の評価時にとる各動作をデータフローマシン(ＤＦマシン)を用いて実現することを試みる。即ち、関数型言語による式の評価を[1]のリダクション戦略に基づき実行するＤＦマシンの一つの構成法を提案し、ＧＲマシンがもつ上記オーバヘッドをＤＦマシンの採用により改善することを試みる。

関数型言語の処理系をＤＦマシンで実現しようとする場合に検討が必要とされる幾つかの問題点が存在する。その主な点としては、(i)関数の(再帰的)呼出しに要するオーバヘッド、(ii)遅延評価、特に利用者定義関数の引数のそれに関する機能の実現、(iii)高階関数の取扱い、及び(iv)リダクション戦略の組込み等が挙げられる。

本稿で示すＤＦマシンでは、上記(i)については、命令(アクタ)を演算子と結果の行き先からなる命令コード部と引数列等からなるデータコード部に分解表現し、かつ相対番地方式を用いたプログラム構造を採用することにより、関数呼出しによるその本体の複写を不要にし、そのためのオーバヘッドを避ける方式をとっている。また、特殊な記憶参照機能を追加することにより、相互参照の制御と記憶域管理の簡単化を計っている。(ii)については制御トークンの利用の他に利用者定義関数の引数の遅延評価のための特殊アクタを採用することによって解決している。(iii)については[1]のリダクション戦略の拡張と合せて検討する必要があり、今回のＤＦマシンでは一般的な取扱いとしては組込まれていない。(iv)について

は上記(i)と(ii)に関して設けられた機能を用いることにより、関数型言語によるプログラムに関する[1]のリダクション戦略を組込んだ目的コードが本稿でのＤＦマシンに対して実現することを明らかにする。

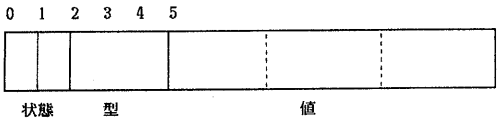
２．命令体系とそのプログラム構造

２－１　語の構造

本システムの記憶装置Ｍは複数の演算装置から共通に参照される。Ｍ内での参照単位である語はアクタの構成要素として使用される場合に域(フィールド)とも呼ぶこともある。また、語は状態部、型部と値部から構成される(図１)。状態部は語の参照に関する状態を、型部は保持する値の型を表わす。番地型と番地並び型の語の値部は、論理的にベース部、Ｂ変位部、Ｃ変位部の三領域に区分され、それらの領域中の値であるベースβ、Ｂ変位δとＣ変位ｉの三重対（β,δ,i)を用いて表現され、Ｍ中の番地を表す。βは本システムでは、後述するＤ(又はＩ)ブロックの先頭番地として利用される。δはブロック内で後述するデータ(又は命令)セルと呼ばれる構造までの変位として利用され、ｉはセル内における語までの変位として利用される。これらの番地を表わす語は、そのβが未定義の場合には相対番地を、定義されている場合には絶対番地を意味する。番地（β,δ,i）はベースβと相対番地（⊥,δ,i）の組合せと考えて、（⊥,δ,i）をαと書いて、番地を <β、α>と略記する。このとき、α、<⊥,α>、（⊥,δ,i)は形式は異るが、同じ(相対)番地を意味するから、等号(=)で結んでこの関係を表す。一方、他の型の語の値部には単に各対象を表現する数値が入る。

２－２　アクタの形式とプログラム構造

帰納的表現による関数を計算する場合に、関数呼出しによって生じるオーバヘッドは効率上特に大きな影響を与える。そこで、著者らは、関数呼出しに際して関数本体の不要な複写を極力避け、かつ、実行に必要なアクタのみを取出し得るアクタ形式及びプログラム構造の検討



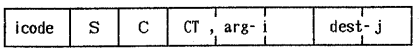
第０ビット：この語が定義されていれば１、未定義ならば０
第１ビット：この語が参照不可ならば１、可ならば０
第２～第４ビット：語の型を表し、以下の型がある。

００１：数値（整数）
０１０：論理
０１１：番地
１００：関数名
１０１：番地並び（番地型の語の並びへの指示子）
１１１：消去
０００：その他

図１　語の構造

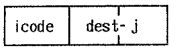
（１）概念的なアクタ形式

アクタ　Ａ＝［Ｉ，Ｄ］：



（２）本システムでのアクタ形式

命令セル　Ｉ：



データセルＤ：

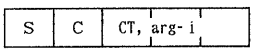


図２　アクタの形式

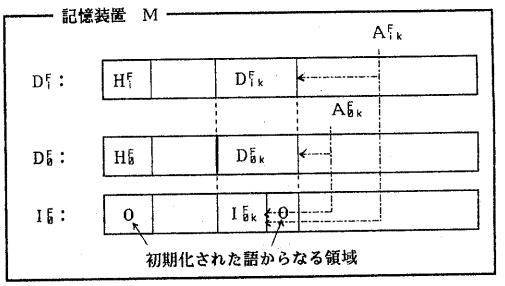


図３　関数のプログラム構造

を行った。本稿での個々のアクタの詳細な仕様は次節に述べるとして、ここでは一般的なアクタの形式とプログラム構造について述ることにする。

通常、使用されるアクタＡは図２の様に演算子icode、引数トークンの数 S(演算に必要なトークンの数)、カウントC(発火までに必要な未到着トークンの数－１)、Ａが制御トークン(CT)を必要とする時は CT 、第ｉ引数 arg - i (i = 0,...,n)、１又は２つの行き先 dest- j （j = 0,...,m)を１つのレコード形式として表される。これらに対して、本システムのアクタの形式では、図２のＡを、命令セルと呼ばれるicode、dest- j から成る部分とデータセルＤと呼ばれるS、C、もしあれば CT、arg- i から成る部分に分割して表現する。そして、１つのアクタＡを、ＩとＤによりＡ＝［Ｉ、Ｄ］のように、ＩとＤの第１番目の語を、各々Ｉ［１］とＤ［１］のように表わすことにする。

本システムのプログラム構造、すなわち、関数Ｆの定義のプログラム構造は、 Ｉブロック(I_0^f)と呼ぶ命令セルの列 $\{I_{0k}^f\}_{k=1}^{N_1}$ と原始Ｄブロック(D_0^f)と呼ぶブロックから成る(図３)。そして、D_0^fは後述するヘッダ(H_0^f)と呼ばれる１つのレコードと継続するデータセルの列 $\{D_{0k}^f\}_{k=1}^{N}$ から成る。なお、I_0^fは実行時に関数Ｆの各呼出しに対して共有されることになる。Ｄブロックについては、Ｆの各呼出しに対して、 D_0^fと同じ大きさの領域$D_i^f(i > 0)$が確保される。 その際にD_i^fにはD_0^fから H_0^Fのみが複写される。

D_i^f中の他の領域は、この呼出しでＦの定義から生じた

FP	BO	CPs	EVs	Ta	Ra	Bi	Size

FP　：この関数がとる引数の個数
BO　：この関数の原始Dブロックのベース
CPs：評価された引数の値を関数本体へ配分するCPのarg-0域の相対番地を要素に持つ並びへの指示子
EVs：遅延引数に評価要求を出すEVのarg-0域の相対番地を要素に持つ並びへの指示子
Ta　：関数本体で最初に制御トークンを必要とするアクタ群にCT "T"を出力するTGのCT域の相対番地
Ra　：RETaのarg-1域の相対番地
Bi　：この関数のIブロックの相対番地
Size：原始Dブロックの大きさ

図4　ヘッダの形式

演算機能	アクタの形式 命令セル	アクタの形式 データセル	属性φ V	N	T	F
(1)演算	i d0 d1	S C CT a0 .. an	0	0	0	0
(2) AP	i d0	S C CT a0 .. an	0	0		
(3) EV	i d0	S C CT a0		0		
(4)INIT	i d0 d1	S C a0	0			
(5)戻り	i d0	S C a0 a1 a2	0			
(6)解放	i d0	S C a0 .. an	0			

（注）iはicodeの、diはdest-iの、aiはarg-iのそれぞれ略記である。
＊1　属性φがT又はF型のときに限って存在する。
＊2　属性φがN，T又はF型のときに限って存在する。
＊3　GMアクタのときに限って存在する。

図5　個々のアクタの形式

アクタA^{f}_{k}=[I^{f}_{k},D^{f}_{k}]（但し、$I^{f}_{k}=I^{f}_{0k}$）の各引数（D^{f}_{k}[I]に書込まれるトークン）を生成するアクタ（生産者）$A^{f'}_{k'}$の実行によって書換えられる。　また、上記A^{f}_{k}と$A^{f'}_{k'}$とにおいて、A^{f}_{k}のことを$A^{f'}_{k'}$の演算結果の消費者と呼ぶことにする。なお、任意のk（k = 1, ...,N）について、I^{f}_{0k}とD^{f}_{k}（i≧0）の先頭位置は、各々I^{f}_{0}とD^{f}_{i}（i≧0）の先頭位置から同じだけ変位している。このため命令セル間に初期化された語が存在する場合がある。ここに、初期化とは、語の各ビットを0にすることをいう。以後、初期化された語の状態を未定義であると呼ぶことにする。

ヘッダHは図4のような8語のレコードである。その各語にはDブロックの確保、起動や解放に必要な情報が保持されている。尚、ヘッダHの第1番目の語を H[1]で表すことにする。

2－3　アクタの仕様

アクタAは演算機能の種類、と制御トークン（ CT ）に関する属性φにより分類される。但し、 CT は論理型トークン（ "T" 又は "F"）であるが、唯一例外として、後述するアクタ EVと INIT間では CT "T"を番地型トークンで代用する。属性φには、V、N、T、とFの4つの型がある。V型のアクタは CT 域をもたないが、他の3つの場合はもつ。N型のアクタの CT 域の存在は、単に発火の一条件として CT の到着を待つためのものである。T（又はF）型のアクタは二種類の選択的な行き先（通常の引数トークンのそれと消去トークンのそれ）をもっている。

そして、発火時にCTの値が "T"（又は "F" ）である場合、アクタAの演算機能の種類に対応する結果を生成して、通常の行き先へ転送し、"F"（又は "T" ）である場合、Aの現在入力されている各トークンを消去して、結果とは別の行き先へ消去トークンを転送する。

一方、アクタは演算機能別に六種類あり、各々の形式を図5に示す。以下に各アクタの機能を説明する。システム中の計算要素Π_jは、発火可能アクタA^{f}_{k}=[I^{f}_{k}, D^{f}_{k}]（但し、$I^{f}_{k}=I^{f}_{0k}$）のD^{f}_{k}の先頭番地 $<\beta^{f}_{i},\alpha^{f}_{0k}>$ をCスケジューラを通じて、システム中の分配要素 Δ_k から受取る。なお、この番地 $<\beta^{f}_{i},\alpha^{f}_{0k}>$ のことをアクタAの発火情報と呼ぶことにする。ここでβ^{f}_{i}はD^{f}_{k}の属するDブロックD^{f}_{i}のベース（現在ベースと呼ぶ）であり、α^{f}_{0k}はβ^{f}_{i}からD^{f}_{k}の先頭位置までの変位（相対番地）を示す。この時、Π_jはD^{f}_{0}のヘッダの要素H^{f}_{0}[6]からI^{f}_{k}（$=I^{f}_{0k}$）の属するIブロックI^{f}_{i}（$=I^{f}_{0}$）の先頭番地 γ^{f}_{0} を取出し、　命令セル I^{f}_{k}（$=I^{f}_{0k}$）の番地 $<\gamma^{f}_{0},\alpha^{f}_{0k}>$ を生成する。そして、これをもとにI^{f}_{k}（$=I^{f}_{0k}$）の各語の値を読取り、更に $<\beta^{f}_{i},\alpha^{f}_{0k}>$ を基にデータセルD^{f}_{k}の各語の値を読取り、上記の属性φの型別動作と以下で述べる演算機能別動作を行なう。なおアクタ EV のデータセルの第二語は通常の記憶読取り参照命令を用いるが、それ以外のデータセルの各語は読取ると同時にその語を初期化する記憶参照命令ld2を用いる。しかるのち、演算結果rと行き先dとベースβからなる結果パケットPR <<r,β,d>> を作り、Dスケジューラを通じて、Δ_kに転送する。

（1）（組込み）演算アクタ

算術、論理、複写（ CP ）等の組込みの演算子に対応する組込関数（その各引数はストリクト）の適用を実行するアクタである（if のような遅延引数を含む組込み関数は本システムではアクタを組合せて実現される）。 その動作は以下のようになる。A^{f}_{k}の引数トークンを演算子に適用し、得られた適用結果r、この演算アクタの発火情報中の現在ベースβ^{f}_{i}と dest-0域の内容d_0（語の内容を表す時には＊印を識別名の前に置き、＊dest-0のように表すこともある）からPR <<r,β^{f}_{i},d_0>> を作成し、Δ_kに転送する。なお、複写アクタ CP のみがT又はF型をとりうる。そして、このとき、その CP には通常の行き先の他に、トークンの消去を通知するための消去トークンの行き先 dest-1 をもつ。また、arg-0 域にあらかじめ　"T" をもつ CP をTG アクタと定義する。

（2）　APアクタ

利用者定義関数の適用を行うアクタである。その動作は以下のようになる。

(i) arg-0域の内容である関数名Gを基に、関数名表を走査し、Gに対応する原始DブロックD$_0$のベースβ^{0}_{0}を

取出す。Π_jはD_β^αの大きさ($H_\beta^\alpha[7]$に保持される)と同じ大きさの空き領域の確保要求を監視用演算装置P_θに出して、その結果確保された領域D_β^αのベースβ_β^αを受取る。そして、D_β^αにD_β^αのヘッダH_β^αを複写して、これをH_β^αと記す。

(ii) 各j($1 \leqq j \leqq n$)について次の動作を行う。arg-j域の内容a_j(Gの第j引数)を、呼出したGのDブロックD_β^αに渡すためにPR $<<\overline{a}_j, \beta_\beta^\alpha, e_j>>$ を作り、Δ_kに転送する。但し、$\overline{a}_j$はa_jが番地型の時は、a_jにβ_β^αを加え込んだ番地であり、そうでなければ、$\overline{a}_j = a_j$である。e_jは、a_jが番地型か否かにしたがって、$*H_\beta^\alpha[3]$か$*H_\beta^\alpha[2]$のいずれかの内容が指す(相対)番地並びのj番目の要素である。

(iii)Gの関数本体で最初にCTを必要とするアクタ群にに起動をかけるため、$H_\beta^\alpha[4]$の内容ξ(それらのアクタに"T"を出力するTGのCT域を指す)等からPR $<<$"T", β_β^α, $\xi>>$を作り、Δ_kに転送する。

(iv)戻り先を設定するために、$H_\beta^\alpha[5]$の内容η(D_β^α中のRETeのarg-1域の相対番地)、このAPの発火情報中の現在ベースβ_β^αと dest-0の内容d_θからPR $<<\beta_\beta^\alpha, \beta_\beta^\alpha, \eta>>$と$<<d_\theta, \beta_\beta^\alpha, \eta \oplus 1>>$を作り、$\Delta_k$に転送する。但し、$\eta \oplus 1$は相対番地 η=(⊥, δ, i) の次の相対番地(⊥, δ, i+1)を表すものとする。また、β_β^αは戻り先のベースを表し、d_θは結果の行き先を表す。

(3) EVアクタ

EV アクタと次に述べる INIT アクタは対をなして利用者定義関数の遅延引数である式Eに、Eの評価要求を伝える。EV アクタは呼出された関数本体中に出現し、INIT アクタはその関数本体を呼出した AP アクタを含む関数本体中に出現する。これらのアクタは、GRマシンで用いられる式の共有の技法をDFアーキテクチュアで実現するためのアクタである。遅延引数を考慮しなければ、プログラム中のアクタは、実行時にそのデータセルへ1つでもトークンが書込まれたら、そのアクタは将来発火することが保証され、従って、アクタ内にあるトークンは常に消費される。しかし、EV においては、遅延引数を取扱うことから上記の保証は両方ともされない(遅延引数Eは、関数内で評価要求が出されないこともあり、また、複数箇所からの評価要求がくることによる)。このため、EV のデータセルの各語は、RETa の動作の一部として初期化される。EV の動作は以下のようになる。

(i) EV の dest-0の内容 d_θはあるアクタAの特定の語を指す相対番地α'であり、この d_θ とこの EVアクタの発火情報 $<\beta_\beta^\alpha, \alpha_\beta^\alpha>$中の現在ベース$\beta_\beta^\alpha$から $\overline{d}_\theta = <\beta_\beta^\alpha, d_\theta>$ を作る。そして arg-0の内容 $<\beta, \alpha>$(これは、前述のように、 AP アクタの動作により生成された絶対番地で、通常、この番地は、対応する遅延引数に起動をかける INIT アクタの arg-0域を指す)とともにPR $<<\overline{d}_\theta, \beta, \alpha>>$を作って、$\Delta_k$に転送する。複数の箇所から評価要求がきても、2回目以降は発火しない機構については、トークンの書込み動作(2-4節) で述べる。また、EV, INIT と RETe の使用法は4章で述べる。

(4) INITアクタ

遅延引数である式Eの評価要求を EV アクタから受理し、Eの評価のための起動をかけ、更にEの評価結果の戻り先を設定する。INIT アクタの arg-0域の内容 a_θには、(i) $a_\theta = <\beta', \alpha'>$(これは、 EV アクタから送られたもので、先の EV アクタの所で示した$\overline{d}_\theta$がこれに該当する。通常、$\overline{d}_\theta$はEの評価結果を使用する各アクタへその値を配分する CPアクタの所在を示す絶対番地である)の場合と(ii) a_θ = "F" ("F"は適用された関数内でEが使用されない場合に、 その関数のRETa より送られる)との場合がある。

(i) の場合は、INITの dest-0の内容 d_θ (これは相対番地を要素とする並びへの指示子であり、現在、その各要素は、後述するEのゲートの各 CP 又は TG アクタの CT 域を指す。なお、これらアクタの属性ΦはT型である)とこの INIT アクタの発火情報中の現在ベースβ_β^αとからPR$<<$"T", $\beta_\beta^\alpha, d_\theta>>$を作り、$\Delta_k$に転送する。更にこのINITの arg-0の内容$<\beta', \alpha'>$を2つの番地β'とα'に分解し、上記現在ベース$\beta_\beta^\alpha$と dest-1の内容 d_1 (これは相対番地であり、通常、Eに対応する RETe の arg-1域を指す)とから、各々、PR $<<\beta', \beta_\beta^\alpha, d_1>>$と $<<\alpha', \beta_\beta^\alpha, d_1 \oplus 1>>$を作り、 Δ_kに転送する。ここに、$d_1 \oplus 1$は相対番地d_1の次の番地である。β'は戻り先のベースを表わし、α'は結果の行き先を表す。

(ii) の場合は、PR $<<$"F", $\beta_\beta^\alpha, d_\theta>>$を作り、$\Delta_k$に転送する。

(5) 戻りアクタ

戻りアクタには、呼出された関数本体からそれを呼出した関数本体に戻る RETa とその他に遅延引数から評価要求を出した所に戻る RETe の二種類がある。両方に共通する動作としては、arg-0の内容a_θ(これは関数本体又は遅延引数の評価結果)、arg-1の内容a_1(戻り先のベース; この値は先に述べた様に AP 又は INIT アクタにより設定される)とarg-2の内容a_2(結果の行き先; a_1同様、APか INITにより設定される)とから、PR $<<a_\theta, a_\beta^\alpha, a_2>>$を作り、$\Delta_k$に転送する。但し、$a_\beta^\alpha$は arg-1が未定義ならば、この戻りアクタの発火情報中の現在ベースβ_β^α、そうでなければ、a_1である。次に、消去トークン "er"、上記現在ベースβ_β^α、dest-0の内容d_θからPR $<<$"er", $\beta_\beta^\alpha, d_\theta>>$を作り、RETeの場合は直ちに$\Delta_k$に転送する。また、RETa の場合は次に述べる" EV に関する残留トークンの消去" 動作を行った後に、先に作ったPRをΔ_kに転送する。

" EV に関する残留トークンの消去" 動作: ヘッダ要素 $*H_\beta^\alpha[3]$で指される並びの各要素 (これらは、EV の CT 域の相対番地)が指す EV について、次の (i),(ii)の動作を行う。(i) CT 域を初期化する。(ii) arg-0域に番地型トークン $<\beta, \alpha>$(これは AP により設定され、対

応する INIT アクタの arg-0 域の絶対番地)が残留していたならば、(ii)-1 PR <<"F", β, α>> を Δ_k に転送し(これにより、INIT アクタは、後述するゲートの各 CP 又は TG アクタへ "F" を送り、それらへ到着する引数トークンを消去させる)、更に、(ii)-2 この EV の dest-0 の内容 $d_\emptyset^0$ と消去トークン"er"を用いて、PR <<"er", β^F_i, $d_\emptyset^0$>> を作り、Δ_k に転送する(後述する残留 CT を消去するため)。

(6) 解放アクタ

解放アクタは、Dブロックの再利用を目的として設けられ、後述するBL式に対して1個付加される。解放アクタには、その BL式に対応するプログラム中に後述の残留 CT の不在を通知する GM アクタとDブロックの解放する FR アクタの2種類がある。

GM アクタはPR <<"er", β^F_i, $d_\emptyset$>>を作り、Δ_kへ転送する。ここで、"er"は消去トークンであり、β^F_i はこの GM アクタの発火情報中の現在ベースであり、$d_\emptyset$ はGMの dest-0の内容である。なお、$d_\emptyset$ はこの GM アクタよりも(後に定義される)1段上と呼ばれるBL式の解放アクタを指す相対番地である。

FR アクタは、以下の(i),(ii)の動作を行うが、結果パケットは生成しない。(i)ヘッダ H^F_i の各語を初期化する。(ii)発火情報中の現在ベース β^F_i と D^F_i の大きさ(H^F_i[7]の内容)で領域解放要求パケット P_F を作り、監視用演算装置 $P_\emptyset$ に渡す。なお、D^F_i のデータセルとして利用された領域は各アクタの演算時に初期化されている。

2-4 結果トークンの書込み動作

2-4-1 記憶装置の特殊参照命令

本システムの記憶装置Mは、時刻 t における一定個の演算装置 P_j ($j \geq 0$; $P_\emptyset$ は監視用演算装置であり、その他の P_j は分配要素 Δ_k 又は計算要素 Π_i である)からの記憶参照要求 R_j のうち、t と j によって定まる適当な優先順位に基づいて1つの $R_{j\emptyset}$ を選択するものとする。また、記憶参照要求 R_j の種類には、通常の書込み要求(st) と読取り要求 (ld) とその他に次に述べる特殊書込み要求 (st1 と st2)及び読み取り要求(ld1 と ld2)があり、M は以上の指令に対応する動作を制御実行する機能をもつものとする。また、各 P_j にはレジスタ Ω があり、指令動作が終了後、参照直前の語の第0と第1ビットが、各々 Ω の第0と第1ビットに転写されるものとする。以下に各指令に対応する動作を述べる。

st1: 参照する語が未定義(その第0ビットが0)ならば、書込みを行い、その語の第0ビットを1にする。定義されてるならば、無操作。

st2: 無条件に書込みを行い、その語の第0ビットを1、第1ビットを0にする。

ld1:参照する語が定義され、かつ参照可能(その第1ビット0)ならば、その語を読取る。そして、その語の第0と第1ビットを共に1にする。そうでなければ、無操作。

ld2: 無条件に参照する語を読取り、同時にその語を初期化する。

2-4-2 結果トークンの書込み動作

以下に分配要素 Δ_k による結果トークンの書込み動作を述べる。まず、上記 2-4-1 で述べたように Δ_k には参照直前の語の状態を保持するレジスタ Ω があり、その第0と第1ビットの値を、各々 $b_\emptyset$ と b_1 とする。Δ_k は次の動作Δを繰返す。動作Δ:

(i) Δ_k はPR <<r, β, d>>をDスケジューラから受取る。但し、d は相対番地 ($\perp$, δ, i) とする(なお、d が相対番地並びへの指示子のときは、その各要素について動作Δ'((ii)から(vi)までの動作)を繰返し、d が未定義ならば動作を終了する)。動作Δ':

(ii) 絶対番地 <β, α> = (β, δ, i) へ結果トークン r を記憶参照命令 st1 を使って書込む。

(iii) $b_\emptyset$ = 1 (<β, α>の語が定義されていた)ならば、直ちにこの動作Δ' を終了する。そうでないならば、以下の発火判定に移る。

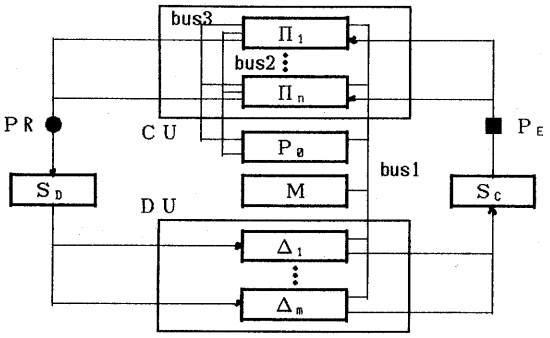
(iv) r を書込んだ語を引数とするアクタAの C域(その番地は(β, δ, 1)である)を記憶参照命令 ld1 で読取る。

(v) $b_\emptyset$ と b_1 が共に1 ならば、(iv)へ戻る。$b_\emptyset$ と b_1 が共に0ならば、β の次の番地の内容 $\beta_\emptyset$ を用いて番地 ($\beta_\emptyset$, δ, 1)(原始Dブロックの対応する C域)を通常に ld 命令で読取る。これら以外の時、無操作。

(vi) (iv) 又は(v)で読取られた(C 域の)値 c が0の場合には、Aが発火可能であることを意味する。このとき、Aの C 域を初期化し、Aのデータセル中で未定義の S, arg-i 域に原始Dブロックより定数を転写し、Aのデータセルの先頭番地 (β, δ, 0) で発火情報 P_E を作り、転送要求を出し、Cスケジューラに送る。一方、c > 0の場合には、値 (c-1) をAの C 域に st2 命令で書込む。

3 システムの構成と動作

本システムは、図6のような機能分散された循環パイプラインアーキテクチュア[5] を基に実現される。計算ユニットCUは、互いに独立で均質な計算要素 Π_i からなる。Π_i は発火可能なアクタの発火情報 P_E をCスケジューラ S_C より受理して、2-3 節に述べたアクタの演算を行う。そして、結果パケット P_R を生成し、Dスケジューラ S_D へ転送する。分配ユニット DU も、 CU 同様、一定個の分配要素 Δ_k からなる。各 Δ_k は、2-4 節で述べたように、P_R を S_D より受理して、結果を書込み、その際に、この結果を書込んだアクタの発火の可否を判定する。発火可能であることが検出されたならば、S_C へ P_E を転送する。S_C と S_D の両スケジューラはデッドロックを起こさないように、記憶装置Mに匹敵する大きさの環状バッファを用いた非同期通信を行う。両スケジューラの構成と動作は 3-1 節で述べる。監視用演算装置 $P_\emptyset$ は記憶域管理等を行ない、動作は3-2節で述べる。また、Mと各演算装



CU： 計算ユニット　　Πi (i=1,･･･,n)： 計算要素
DU： 分配ユニット　　Δk (k=1,･･･,m)： 分配要素
Pø ： 監視用演算装置　　M ： 記憶装置
Sc ： Cスケジューラ　　SD： Dスケジューラ
PE ： 発火情報　　PR： 結果パケット

図6　システム構成図

置Pjとは図6中の bus-1 で接続され、Pøと各Πiは図6中の bus-2 と bus-3で示される2本のバスで接続される。

3－1　Cスケジューラ

Cスケジューラ Sc には書込み系Wと読取り系Rと呼ばれる2つの演算装置とPEを蓄積する環状バッファQ（ Nc 個のスロット(slot) をもつ）からなる。Wの動作は以下の繰返しである。(i) ある時刻tにおいて、機械的に検出されるWの書込み位置 Tw とRの読取り位置Trの関係 Tw+1 ≡ Tr(mod Nc) が不成立であり、 かつ1個以上のΔkからのPEの転送要求が出ているならば、これらのΔkから、tとkとに依存する優先順位に基づいて1つのΔkøを選択する。 そうでなければ、(i)に戻る。(ii) そのΔkøから Tw が指すスロットにPEを書込んで、Tw を次スロットに設定する。一方、Rの動作は以下の繰返しである。(1) 関係Tw = Trが不成立であり、かつ空き計算要素Πiが幾つかあるならば、これらのΠiから適当なΠiøを選ぶ。そうでなければ、(1)に戻る。(2) Tr の示すQのスロットからPEを読取って、それをΠiøに転送し、Tr を次スロットに設定する。なお、Dスケジューラも同様の構成と動作である。

3－2　監視用演算装置 Pø

Pøは、動作系Oと解放要求パケットPF(解放する領域の先頭番地と大きさからなる)の書込み系Wの2個の演算装置とその他にPFを蓄積する環状バッファQ (Np 個のスロットをもつ)からなる。Oは以下の動作の繰返しである。(i)ある時刻tにおいて、いくつかのΠiからの領域確保要求が出ていれば、以下の確保動作を行い、確保要求がなくかつWの書込み位置 Tw とOの読取り位置 Tr が等しくないならば、以下の解放動作を行う。何れでもなければ(i)へ戻る。

確保動作：確保要求を出しているΠiのうち、tとiとに依存する優先順位に基づいて1つのΠiøを選択する。OはΠiøとの同期通信を開始し、Πiøから確保する領域の大きさを受取り、記憶資源の管理表を参照して領域を確保し、 その先頭番地をΠiø へ返した後に通信を終える。

解放動作：Tr が示すQのスロットからPFを読取って解放領域を管理表に登録し、Trを次スロットに設定する。

また、Pøの書込み系Wは、Scの書込み系Wと同様の動作をとる。

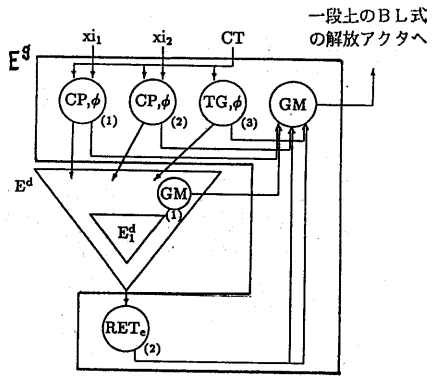
4　関数型プログラムから データフロープログラムへの変換

文献[1]のリダクション戦略の実行制御をDFアーキテクチュアを用いて実現するためには、これまでのDFアーキテクチュアでは取扱えないプログラム構造、即ち、(i)遅延引数の評価と(ii)(効率のよい)関数呼出しの機構を実現しなければならない。

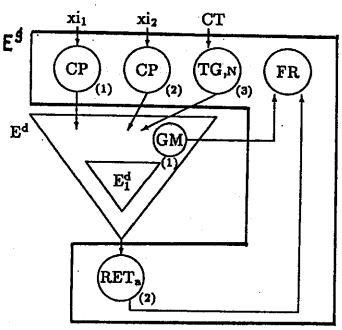
本マシンでは、(i)については、組込関数のそれの場合は要求駆動処理の計算要求に対応する制御トークン(CT)を特定のアクタに到着させることで実行制御する(4-1節)。利用者定義関数のそれの場合には、CTだけでは実行制御が行えないので、固有のアクタ(EV, INIT, RETe)を定義して関数呼出しの系列間でトークンを授受する機構を設けることにより解決している。 更に、これらの固有のアクタは、GRマシンで実現される同一式の重複評価を避ける技法(式の共有)も実現させる。

(ii)については、第一に、アクタを演算子と結果の行き先からなる命令コード部、引数列等からなるデータコード部に分割表現し、かつ相対番地方式を用いたプログラム構造を採用することにより関数呼出し時のその本体の複写を不要にしている(2章)。第二に、本章で述べる変換により得られるプログラム構造では、遅延引数Eに対してゲートと呼ばれるプログラム部分を設けている。これにより、遅延引数部分へ不要なトークンの書込みを防止し、ゲート中の解放アクタが遅延引数部分に残留するトークンがないことを保証する機構を形成するので、特別な記憶管理を必要としなくても、関数の実行と並行して領域の再利用を行うことができる。

なお、本章では変換とは、関数型プログラムの構成要素である(関数)定義式を、Iブロックと原始Dブロックとを結合したDFプログラムの(関数)定義体に変換することをいう。ここに、関数定義式は$F(x_1,\ldots,x_n) \leftarrow E$の形式をもち、Fを関数名、$(x_1,\ldots,x_n)$を変数並びといい、関数本体Eには $x_1,\ldots,x_n$以外の自由変数はないとする。また、式Eを変換したDFプログラムをE^dと記し、DFプログラムはグラフ表現で記述するものとする。



（ｉ） ＥがＬ式のときのゲート(但し、φはＴまたはＦ)



（ｉｉ)ＥがＢ式のときのゲート

図７ （ゲート） ＢＬ式ＥのゲートE^g； ()の内の数はゲート要素に割り当てられた番号である。Ｎ型以外のアクタに限り、その節点内には演算名の他に属性の型が示される。

４－１ 組込関数における遅延引数の評価

４－１－１ ゲート

原始プログラム中において、関数本体のことをＢ式と呼び、遅延引数である式のことをＬ式と呼ぶ。また、ＢＬ式とはＢ式又はＬ式のことをさす。ＢＬ式E_1とE_2について、E_1がE_2の一段上(又はE_2がE_1の一段下)にあるとは、次の関係Ｕが成立する時に限っていう。

Ｕ：E_2はE_1中に出現する真部分式（$E_2 \subsetneqq E_1$で表わす)、かつ、$E_2 \subsetneqq E \subsetneqq E_1$なるＢＬ式Ｅが$E_1$中に出現しない。■

ＢＬ式Ｅを変換したＤＦプログラムE^dには、ＥのゲートE^gと呼ばれるアクタの集合が付加される(図７)。ゲートE^gの要素は、(i)Ｅ中に出現する変数の個数(≧０)と同数の CP アクタ(それらはE^dの外部から引数を受取り、E^d の内部の変数等に引数トークンを配分する役割をもち、いずれも同一の型をもつ)、(ii)高々１個の TGアクタ(TGの出力する制御トークン(CT)によってのみ発火可能なアクタ(例１のifの第３引数である(＊３４)がこれに当たる)がE^d 中に出現する場合、又はＥを評価するためにはその値が必ず必要となる遅延引数(例１のifの第２引数中のｙがこれにあたる)がＥ中に出現する場合に限ってこの TG が必要となる。この TG はそれらのアクタ又は遅延引数に対応する EV アクタを発火させるために CT を送り込む働きをもつ)、（iii)１個のRETa(Ｂ式の時)又は RETe(Ｌ式の時)アクタ(これらは、E^d の評価結果を、それを必要とする所に出す)、及び(iv)１個の FR(Ｂ式の時)又は GM(Ｌ式の時)アクタ(これらはE^d中に後述の残留 CT がないことを１段上のＢＬ式のゲート中の解放アクタに消去トークンを出して通知する）から構成される。なお、ゲートE^g はＢＬ式Ｅに対して一意に対応するので、E^g中の要素 CP、TG、戻り、解放アクタ等のことを、単にＥの CP、TG、戻り、解放アクタ等と呼ぶこともある。

また、E^g中の CP の個数l_Cと TG の個数l_Tの和l_Kは０となることはない。更に、E^g中の各 CPと TG に次のように番号を付ける。まず、 CP には対応する変数が関数定義式の変数並び中に左にあるものから順に番号を１からl_Cまで付ける。そして、$l_C = l_K$ ならばこの番号付けを終了する。$l_C < l_K$ならば TG に番号l_Kを付けて、この番号付けを終了する。一方、E^g 中の戻りアクタとＥより一段下のＢＬ式の GM アクタにも番号を付ける。即ち、Ｅの一段下の各ＢＬ式(それらの個数を$l_G - 1$とする）には、Ｅ中で部分式として左に出現するものから順に番号を１から$l_G - 1$ まで付ける。そして、番号ｌ($l = 1, .., l_G - 1$)が付けられたＢＬ式E_lのGM アクタにも同じ番号ｌを付ける。 次いで、戻りアクタに番号l_Gを付ける。これから、ゲート中のアクタの結合関係について述べる。 以下では、$m = \max(l_K, l_G)$とおく。

(i) ここで、ＥをＬ式とする。この時、E^g中の各CP又は TG アクタはいずれも同一の型(Ｔ 又は Ｆ) をもち、かつ２つの選択的な行き先をもつ。そして、上記手続でこのアクタに付された番号をｉとする。(i)-1 その一つ(dest-０)は、外部から与えられる引数トークン(但し、TG の場合は変換時に書込まれた制御トークン)をE^d中の対応する消費者アクタ(但し、TG の場合はこれらの他に幾つかの EV アクタを含むことがある)へ渡すための相対番地(又は相対番地並び)である。 (i)-2 もう一つ(dest-１)は消去トークンを渡す行き先であり、次のように定められる。即ち、もし、$i < l_K$ ならばそれはE^g中のGM アクタの arg-(ｉ－１)域の相対番地であり、$i = l_K$ならばそれは$l_K - 1 \leqq j \leqq m - 1$なるｊをもつ$E^g$中のGM アクタの各arg-ｊの相対番地を要素とする並びへの指示子である。

また、E^g中の GM アクタの dest-０は、上記番号付けでこのアクタに付された番号をｎとする時に、(i)-3 一段上のＢＬ式の解放アクタの arg-(ｎ－１)域の相対番地である。

E^g中の RETe アクタには、E^dの評価結果の出力がその arg-０に接続される。そして、(i)-4そのdest-０はE^g中

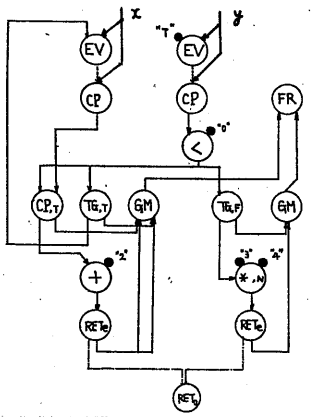
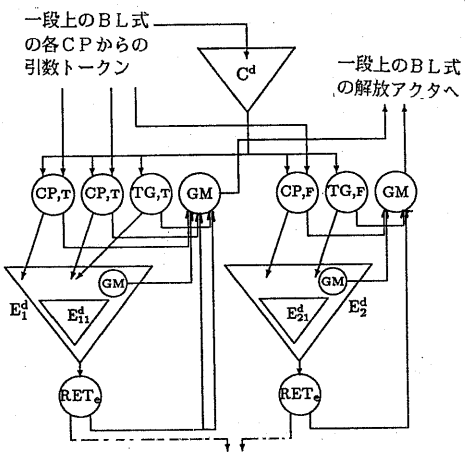


図８　式(1)　G(x,y) ← if(< y 0)(+ x 2)(* 3 4)を変換して得られるＤＦプログラム。但し、これ以降の図では、波線はアクタのdest-ｊ域の内容が結果パケット中の結果として利用されることを表す。鎖線はアクタの引数が結果パケット中の行き先として利用されることを表す。



arg-１とarg-２から決まる同一の絶対番地

図９(if Ｃ E_1 E_2)を変換して得られるＤＦプログラム

の GM アクタの $l_G-1 \leq j \leq m-1$ なるｊをもつarg-ｊ域の相対番地を要素にもつ並びへの指示子である。

従って、E^s中のGMアクタの各arg-ｊについては、(i)-2,(i)-3と(i)-4の手続きから、２つの選択的な入力が送り込まれる。即ち、GM アクタのarg-ｉ域($i = 0,\ldots,m-1$)へは、番号 $\min(i, l_K-1)+1$ をもつE^s中の CP 又は TG アクタからの入力と、番号 $\min(i, l_G-1)+1$ をもつE^s中の戻りアクタ又はＥの一段下のＢＬ式の GM アクタからの入力が接続される。

(ii) 他方、ＥをＢ式とする。このとき、CP はＴ型、TG はＮ型であり、それらの dest-０は外部から与られる引数トークン(但し、TG の場合は、変換時に書込まれた制御トークン)を対応する消費者アクタへ渡すための相対番地(又は相対番地並び)である。

また、E^s中の RETa アクタはE^dの評価結果の出力がそのarg-０に接続されている。そして、その dest-０は、E^s中の FR アクタの arg-(l_G-1)域の相対番地を指す。

【 例　１】　図８に関数定義式：

G(x,y) ← if (< y 0) (+ x 2) (* 3 4)　・・・(1)

を変換して得られるＤＦプログラムを示す。利用者定義関数本体、及びその本体より一段下の２つのＬ式(遅延引数)であるifの第２と第３引数に対応する式に対してゲートと呼ばれる構造が付加される(4-1-2､4-2-2参照)。なお、ifの第２引数のTGからは、ｘの値を要求するためのCTの行き先がｘに対応するEVへ接続され、第３引数のTGからはCT "T" を必要とする(* 3 4)を変換したアクタに接続される。■

４－１－２　組込関数における遅延引数の評価

遅延引数をもつ組込み関数の例として、関数 if を用て説明する。式(if Ｃ E_1 E_2)を変換したＤＦプログラム構造を図９に示す。条件部C^dが出力する制御トークン(CT)の行き先は、E_1とE_2のゲート中の各 CP と TG の CT 域に接続される。なお、$E_1{}^s$中の各 CP と TG アクタはＴ型、$E_2{}^s$中のそれらはＦ型である。 C^dはE_1やE_2より一段上のＢＬ式の各 CP や TG からのトークンの到着により実行が開始される。$E_1{}^d$ と $E_2{}^d$中の変数に向う各トークンは、各々$E_1{}^s$と$E_2{}^s$中の対応する CP 又は TG アクタのarg-０域へ書込まれて、C^dの出力する CT ｘ (="T" or "F")が到着するまで、出力されずに待機する。ｘが出力されると、ｘと型の一致する各 CPと TGアクタは、ｘと引数トークンがそろい次第、引数トークンを遅延引数中の消費者(EVアクタを含むこともある)に出力する。一致しない各 CP と TG アクタは、ｘと引数トークンがそろい次第、両者を消去し合い、消去トークンを同一ゲート中の GM アクタに出力する。よって、C^dの評価結果ｘが確定したとき、ｘが "T" か "F" かに従って、$E_1{}^d$又は$E_2{}^d$のいずれか一方だけが実行され、その評価結果を、式(if Ｃ E_1 E_2)の結果を入力としてとる所へ出力する。他方は、決して実行されず、ゲート中の各 CP と TG アクタが、GM アクタに消去トークンを GMアクタに出力する。なお、各GM が発火する時にはＤブロック中の各遅延引数に対応するこの領域は初期化されている。

４－２　利用者定義関数とその遅延引数

４－２－１　関数定義のＤＦプログラム構造

図10(ii)は関数定義式 $G(x_1,\ldots,x_n) \leftarrow E$ を変換したＤＦプログラム構造である。EV アクタは、このアクタに対応する遅延引数の値の評価要求が起った時に、対応する遅延引数の INIT アクタに対して評価要求を出す。k($k = 1,\ldots,n$)番目の EVアクタ はｋ番目の変数 x_k と一意に対応し、そのdest-０は、x_k に対応する CP が

E^s中に存在すれば、その CP のarg-0域の相対番地であり、存在していなければ、未定義である。また、x_k が遅延引数に対応するならば、k番目の EV の CT 域は変換時に未定義であり、評価要求がきた時に定義される。この時、並列処理を行っているので、初回のみ受理し、2回目以降は無視する。x_k が遅延引数に対応しないならば、変換時に CT を、その EV の CT 域に書込んでおき、実行時には引数トークン(番地型)の到着により直ちに対応するINIT アクタに評価要求を出すことが可能である。RETa は、E^dの評価結果を受理し、この関数(= G)を呼出した関数本体(arg-1とarg-2から決まる絶対番地で指される所)に返し、そして、"EV に関する残留トークンの消去" 動作を行い、更に、評価の終了をE^s中の FR アクタに伝える。なお、各 EV アクタ及び E^s中の要素は、ヘッダ(第2から第5語)情報により参照可能な構造になっている。

4－2－2　関数適用の変換

利用者定義関数の適用($GE_1 \cdots E_n$)は図10(i)のように AP アクタに変換される。このとき、G は変数又は関数名のいずれかであるが、前者の場合は、そのarg-0域は未定義であり、そのすべての引数は遅延引数(L式)とみなされる。次に、各引数を変換する時に、(i)L式でない E_j の場合と(ii)L式E_iである場合で変換方式が異なる。

(i) の場合は、E_j が定数ならば、対応する定数トークンを APのarg-j域に書込み、そうでなければ、E_jの演算結果がarg-j域へ入力トークンとなるように変換する。

(ii) の場合は、この AP アクタと同じ本体中に図10(i)のような、INIT アクタとゲートE_i^sが付加したプログラム構造に変換される。遅延引数E_i^dに対応するE_i^s中の各 CPと TGアクタはいずれもT型である。E_i^s中のRETe アクタは、E_i^d の評価結果を(arg-1とarg-2で指される)呼出された関数G本体のゲートE^s中のi番目変数に対応する CP アクタに渡し、消去トークンを (dest-0が指す)E_i^s中の GM アクタに転送する役割をもつ。また、INIT アクタのdest-0はゲートE_i^s中の各 CP 又は TG のCT 域の相対番地の並びを指し、dest-1はE^s中の RETeアクタをarg-1域の相対番地である。そして、このINITアクタのarg-0域の相対番地(<⊥、α>)が、AP の arg-i域に定数として書込まれる。なお、この arg-i は関数Gの第i引数に対応する。　以下に、4-2-1と4-2-2の変換を施したDFプログラムの実行について述べる。

4－2－3　利用者定義関数とその遅延引数の評価

原始プログラム上の関数適用($GE_1 \cdots E_n$) (E_iはL式, E_jは L式でない)に 4-2-2 の変換を施して得られた AP アクタのarg-i域の内容を <⊥, α>とする。実行時には、この AP アクタのデータセルが特定のDブロック D_{β^f}(そのベースをβ^fとする)中に生成されている。その AP アクタが発火すると、AP アクタの動作(2-3 節)に従って、ここで新たに関数Gに係わるDブロック$D_{\beta^{f'}}$(そのベースを$\beta^{f'}$とする)が確保され、 Gの原始Dブロック D_βのヘッダが$D_{\beta^{f'}}$に複写される。そして、各引数トークンが番地型(遅延引数)か否かに従って、各々Gの本体の対応する EV か CP アクタに渡される。E_iとE_jを例にとると、Gのi番目の EV アクタのarg-0域にはベースが追加された番地型トークン<β^f, α>が渡され(図10中の①で示す)、x_jに対応する CP アクタの arg-0域には、E_jの評価結果が渡される(図中②)。引続いて、TG アク

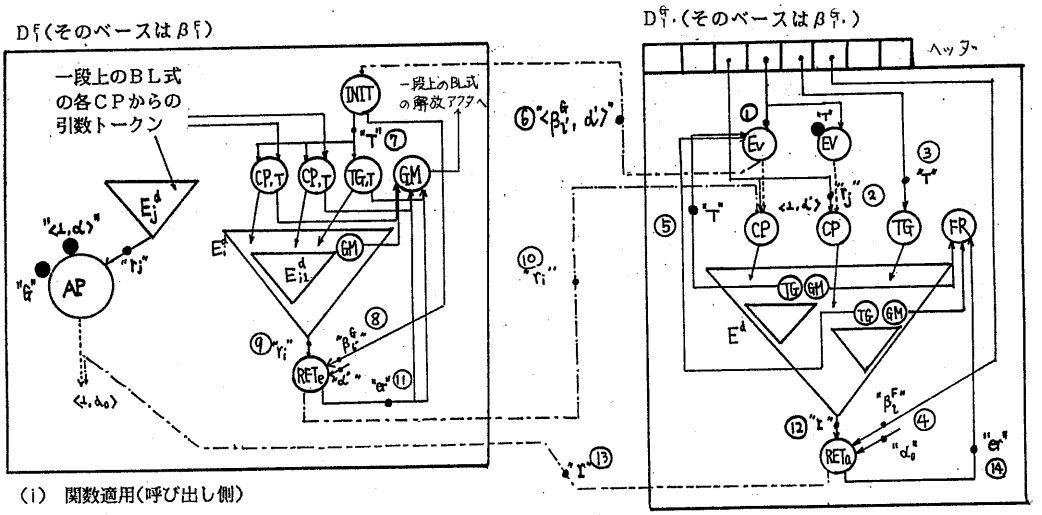


図10　関数適用($GE_1 \cdots E_n$)の実行と遅延引数E_iの評価
但し、E_iは遅延引数に対応し、E_jは遅延引数に対応しない。
この例では、i＝1、j＝n＝2.

タの CT 域へ CT "T" を転送し(図中③)、RETaアクタのarg-1とarg-2 域に戻り先を転送し (図中④)、G本体 E^dの実行が始まる。G本体の評価中に、E_i^dの値を評価するのに必ず使用するBL式の評価が開始される時、そのBL式に対応するゲートの TG アクタが、評価要求を意味する CT "T" を、i番目の EV アクタの CT 域に転送する(図中⑤)で示す)。 "T" を受理した EV アクタが発火すると、 E_i^dに対応する INIT アクタ(そのarg-0域の番地は、Gのi番目の EV アクタの arg-0域に書き込まれた$<\beta\dot{\gamma}, \alpha>$)に番地型トークン $<\beta\dot{\gamma}, \alpha'>$(これは、G本体Eのゲート$E^s$ 中のx_i に対応する CP アクタのarg-0域の絶対番地)が渡される(図中⑥)。$<\beta\dot{\gamma}, \alpha'>$を受理した INIT アクタは直ちに発火して、$E_i^s$の各 CP とTG アクタへ CT "T" を送り(図中⑦)、引数トークンの出力を可能にし、更に、RETe の arg-1と arg-2域へ、各々、戻り先ベース$\beta\dot{\gamma}$と結果の行き先α' を設定する(図中⑧)。このようにして、E_i^d の評価を開始させる。そして、E_i^d の評価結果r_iが RETe に出力されると(図中⑨)、RETe はr_iを呼出されたG本体のゲートE^sの第i引数に対応するCP (これは、このRETeのarg-1とarg-2の内容から決まる絶対番地で指される)へ渡し(図中⑩)、E^d中の第i変数等に配分される。また、RETeは消去トークン"er"をE_i^s中の GMアクタに渡す(図中⑪)。上記の動作はGRマシンにおける式の共有を実現している。

その後、E^dの評価結果rが RETa の arg-0域に出力されると(図中⑫)、RETe はrを、Gを呼出した本体に返し(図中⑬)、以下の " EVに関する残留トークンの消去" 動作を行い、更に、消去トークン "er"を転送し、E^dの評価の終了をE^s中の FR アクタに通知する(図中⑭)。

EV 関する残留トークンの消去:

Gの各k番目の EVアクタに対して、以下の(i)、(ii)の動作を行う。(i) EVのデータセルの各語を初期化する。(ii) このk番目に対応する遅延引数E_k^dが評価されていないならば、(ii)-1Gのk番目の EVのarg-0 の内容が指すINIT アクタへ CT "F" を送ることで(Gを呼出した側の)E_k^dのゲート中の各CP 又は TG アクタに、到着する引数トークンを消去させる。なお、このような場合、即ち、一般にL式のゲートにおいて、CPへCTが出力される場合、その中で遅延引数に対応する CP では引数トークンが到着しないことがある。この CP には CT が残留してしまう。このような CT を残留 CT と呼ぶが、これらの CT の消去は次の動作で行う。(ii)-2 この時点で、まだ、Gの本体中に存在する残留 CT を消去するために、k番目の EV の dest-0域の内容が指すGの第k引数に対応する CP へ消去トークン "er" を送る。

その後、FR アクタが発火する時には、このGのDブロック$D\dot{\gamma}$中に残留するトークンがないことが保証されるので、FR アクタは、$D\dot{\gamma}$の解放を直ちに(P_βに)要求することができる。

以上で、本節において、関数呼出しとその遅延引数の評価の機構がこれまでに定義したアクタを用てDFプログラム構造で構成されることを示した。そこでは、最左リダクションの遅延評価性を保証し、できる限り並列に引数を評価し、かつデッドロックを起こさない。しかも、引数の評価は一度だけであり、従って、文献[1]の並列リダクション戦略を実現しているといえる。

5 むすび

本稿では、GRマシンの実行時のオーバヘッドを避けるために、一定個の演算装置の下での[1]の並列リダクション戦略を実現するDFアーキテクチュアの一提案を行った。その戦略の組込みについては、遅延引数の評価は制御トークン、EV, INIT, 戻りアクタ等の特殊アクタを定義し、それらを用いたDFプログラム構造により実現し、その他の並列に評価可能な引数の評価は単なるトークンの到着により評価を行なうことで自然に実現することができる。これらのことは最左リダクションによる遅延評価性を保証しながら、並列評価可能な式をできる限り並列に実行し、かつデッドロックを起さない評価戦略の実現を可能にする。また、本文で述べたアクタの形式及びプログラム構造の採用により関数呼出し毎の不要な複写を避けることができ、かつ解放アクタの機能により領域の管理及び再利用を容易にすることができる。

なお、今回のDFマシンでは、一般の高階関数を取扱っていない。これは、文献[1]の戦略でそれらを取扱っていないことによる。今後は、この戦略を拡張し、一般の高階関数の取扱いを可能とするとともに、その戦略をDFマシンでも実現し得るよう更に検討を進めるつもりである。

参考文献

[1] 保坂、広川、関本 :" De Brujn の表現によるλ式のリダクションについて ", 信学技報、COMP 87-4 (1987)

[2] 堀、広川、関本 :" 結合子による並列リダクション ", 信学論(D)、Vol.J70-D、No.8、pp1498-1507 (1987)

[3] Barendregt, H. P. :" The lambda calculas - Its Syntax and Semantics", North-Holland, vol.103 (1981)

[4] Clack, C. and Peyton Jones, S. L.:" Strictness analysis - a practical approach", Springer LNCS, pp35-47 (1986)

[5] Dennis, J. B. and Misaunas, D. P. :"A preliminary architectuce for a basic data flow processor", Proc. of the 2nd Annual Symp. on Computer architecture, pp126-132 (1975)

[6] 曽和将容 :" データフローマシンと言語 ", 照光堂、(1986)