

(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ャについて

A Dataflow Machine Architecture for
Parallel Reduction

国持 良行 関本 彰次

Yoshiyuki KUNIMUCHI 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に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ャ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ャ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。

Abstract 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う記憶域管理等に通常多くのオーバー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ーヘッドを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ーヘッド、(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、そのためのオーバー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単純化を計っている。(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。

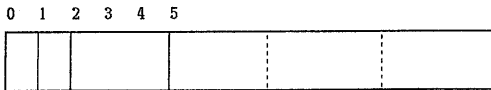
2. 命令体系とそのプログラム構造

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) の組合せと考えると、 (δ, i) を α と書いて、番地を $\langle \beta, \alpha \rangle$ と略記する。このとき、 α 、 $\langle \delta, i \rangle$ 、 (δ, i) は形式は異なるが、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ー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



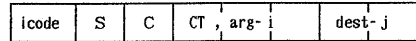
第0ビット：この語が定義されていれば1、未定義ならば0
 第1ビット：この語が参照不可ならば1、可ならば0
 第2～第4ビット：語の型を表し、以下の型がある。

- 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 :

| | |
|-------|--------|
| icode | dest-j |
|-------|--------|

データセル D :

| | | |
|---|---|-----------|
| S | C | CT, arg-i |
|---|---|-----------|

図2 アクタの形式

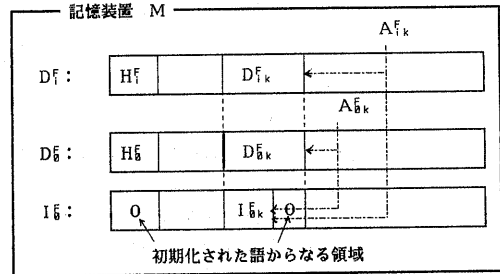


図3 関数のプログラム構造

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べることにする。

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)、カウンタ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($i = 0, \dots, n$)、1又は2つの行き先dest-j($j = 0, \dots, m$)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 \bar{k})と呼ぶ命令セルの列{I \bar{k} } $_{k \in N_1}$ と原始Dブロック(D \bar{k})と呼ぶブロックから成る(図3)。そして、D \bar{k} は後述するヘッダ(H \bar{k})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 \bar{k} } $_{k \in N_1}$ から成る。なお、I \bar{k} 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 \bar{k} と同じ大きさの領域D \bar{k} ($i > 0$)が確保される。その際にD \bar{k} にはD \bar{k} からH \bar{k} のみが複写される。

D \bar{k} 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 ハッタの形式

アクタ $A_{ik} = [I_{ik}, D_{ik}]$ (但し、 $I_{ik} = I_{ik}$) の各引数 ($D_{ik}[i]$ に書込まれるトークン) を生成するアクタ (生産者) A_{ik} の実行によって書換えられる。また、上記 A_{ik} と A_{ik} とにおいて、 A_{ik} のことを A_{ik} の演算結果の消費者と呼ぶことにする。なお、任意の k ($k = 1, \dots, N$) について、 I_{ik} と D_{ik} ($i \geq 0$) の先頭位置は、各々 I_{ik} と D_{ik} ($i \geq 0$) 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可能性がある。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。

ハッタ 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、ハッタ 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の到着を待たないものである。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_k は、発火可能アクタ $A_{ik} = [I_{ik}, D_{ik}]$ (但し、 $I_{ik} = I_{ik}$) の D_{ik} の先頭番地 $\langle \beta_{ik}, \alpha_{ik} \rangle$ 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 $\langle \beta_{ik}, \alpha_{ik} \rangle$ のことをアク

| 演算機能 | アクタの形式 | | 属性 Φ | | | |
|----------|-------------------------|--|-----------|---|---|---|
| | 命令セル | データセル | V | N | T | F |
| (1) 演算 | $i \quad d_0 \quad d_1$ | $S \quad C \quad CT \quad a_0 \quad \dots \quad a_n$ | 0 | 0 | 0 | 0 |
| (2) AP | $i \quad d_0$ | $S \quad C \quad CT \quad a_0 \quad \dots \quad a_n$ | 0 | 0 | | |
| (3) EV | $i \quad d_0$ | $S \quad C \quad CT \quad a_0$ | | 0 | | |
| (4) INIT | $i \quad d_0 \quad d_1$ | $S \quad C \quad a_0$ | 0 | | | |
| (5) 戻り | $i \quad d_0$ | $S \quad C \quad a_0 \quad a_1 \quad a_2$ | 0 | | | |
| (6) 解放 | $i \quad d_0$ | $S \quad C \quad a_0 \quad \dots \quad a_n$ | 0 | | | |

- (注) i はicodeの、 d_i はdest- i の、 a_i 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

タ A の発火情報と呼ぶことにする。ここで β_{ik} は D_{ik} の属するDブロック D_{ik} のベース (現在ベースと呼ぶ) であり、 α_{ik} は β_{ik} から D_{ik} の先頭位置までの変位 (相対番地) を示す。この時、 Π_k は D_{ik} のハッタの要素 $H_{ik}[6]$ から I_{ik} ($= I_{ik}$) の属するIブロック I_{ik} ($= I_{ik}$) の先頭番地 γ_{ik} を取出し、命令セル I_{ik} ($= I_{ik}$) の番地 $\langle \gamma_{ik}, \alpha_{ik} \rangle$ を生成する。そして、これをもとに I_{ik} ($= I_{ik}$) の各語の値を読取り、更に $\langle \beta_{ik}, \alpha_{ik} \rangle$ を基にデータセル D_{ik} 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 からなる結果バケット $PR \langle \langle r, \beta, d \rangle \rangle$ 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)。その動作は以下ようになる。 A_{ik} 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_{ik} 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 $PR \langle \langle r, \beta_{ik}, d_0 \rangle \rangle$ 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 型をとらう。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_{ik} のベース β_{ik} を

取出す。 Π_j は D_{β} の大きさ($H_{\beta}[7]$ 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_{β} に出して、その結果確保された領域 D_{β} のベース β_{β} を受取る。そして、 D_{β} に D_{β} のヘッダ H_{β} を複写して、これを H_{β} と記す。

(ii) 各 j ($1 \leq j \leq n$)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} に渡すために $PR \langle \langle \bar{a}_j, \beta_{\beta}, e_j \rangle \rangle$ を作り、 Δ_k に転送する。但し、 \bar{a}_j は a_j が番地型の時は、 a_j に β_{β} を加え込んだ番地であり、そうでなければ、 $\bar{a}_j = a_j$ である。 e_j は、 a_j が番地型か否かにしたがって、 $*H_{\beta}[3]$ か $*H_{\beta}[2]$ 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起動をかけるため、 $H_{\beta}[4]$ 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 $PR \langle \langle "T", \beta_{\beta}, \xi \rangle \rangle$ を作り、 Δ_k に転送する。

(iv) 戻り先を設定するために、 $H_{\beta}[5]$ の内容 η (D_{β} 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} と $dest - 0$ の内容 d_0 から $PR \langle \langle \beta_{\beta}, \beta_{\beta}, \eta \rangle \rangle$ と $\langle \langle d_0, \beta_{\beta}, \eta \oplus 1 \rangle \rangle$ を作り、 Δ_k に転送する。但し、 $\eta \oplus 1$ は相対番地 $\eta = (\perp, \delta, i)$ の次の相対番地 $(\perp, \delta, i+1)$ を表すものとする。また、 β_{β} は戻り先のベースを表し、 d_0 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アーキテクチャ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、 $RETe$ の動作の一部として初期化される。EVの動作は以下ようになる。

(i) EVの $dest - 0$ の内容 d_0 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_0 とこのEVアクタの発火情報 $\langle \beta_{\beta}, \alpha'_{\beta} \rangle$ 中の現在ベース β_{β} から $\bar{d}_0 = \langle \beta_{\beta}, d_0 \rangle$ を作る。そして $\arg - 0$ の内容 $\langle \beta, \alpha \rangle$ (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、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 $PR \langle \langle \bar{d}_0, \beta, \alpha \rangle \rangle$ を作って、 Δ_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_0 には、(i) $a_0 = \langle \beta', \alpha' \rangle$ (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 \bar{d}_0 がこれに該当する。通常、 \bar{d}_0 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) $a_0 = "F"$ ("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 $RETe$ 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_0 (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} とから $PR \langle \langle "T", \beta_{\beta}, d_0 \rangle \rangle$ を作り、 Δ_k に転送する。更にこのINITの $\arg - 0$ の内容 $\langle \beta', \alpha' \rangle$ を2つの番地 β' と α' に分解し、上記現在ベース β_{β} 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、 $PR \langle \langle \beta', \beta_{\beta}, d_1 \rangle \rangle$ と $\langle \langle \alpha', \beta_{\beta}, d_1 \oplus 1 \rangle \rangle$ 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、 $PR \langle \langle "F", \beta_{\beta}, d_0 \rangle \rangle$ を作り、 Δ_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 $RETe$ 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_0 (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、 $PR \langle \langle a_0, a_1, a_2 \rangle \rangle$ を作り、 Δ_k に転送する。但し、 a_1 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} 、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} 、 $dest - 0$ の内容 d_0 から $PR \langle \langle "er", \beta_{\beta}, d_0 \rangle \rangle$ を作り、 $RETe$ の場合は直ちに Δ_k に転送する。また、 $RETe$ 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 $*H_{\beta}[3]$ 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 $\langle \beta, \alpha \rangle$ (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 <<"r", 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 "r" 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_0 と消去トークン "er" を用いて、PR <<"er", β, d_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, d_0 >> 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 はこの GM アクタの発火情報中の現在ベースであり、 d_0 は GM の dest-0 の内容である。なお、 d_0 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、結果バケットは生成しない。(i)ヘッダ H_F の各語を初期化する。(ii)発火情報中の現在ベース β と D_F の大きさ($H_F[7]$ の内容)で領域解放要求バケット P_F を作り、監視用演算装置 P_0 に渡す。なお、 D_F 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_0 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_0} 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_0 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 $\langle \beta, \alpha \rangle = (\beta, \delta, i)$ へ結果トークン r を記憶参照命令 st1 を使って書込む。

(iii) $b_0 = 1$ ($\langle \beta, \alpha \rangle$ 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。

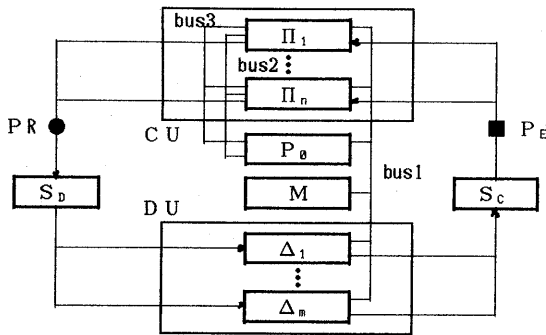
(iv) r を書込んだ語を引数とするアクタ A の C 域(その番地は $\langle \beta, \delta, 1 \rangle$ である)を記憶参照命令 ld1 で読取る。

(v) b_0 と b_1 が共に 1 ならば、(iv)へ戻る。 b_0 と b_1 が共に 0 ならば、 β の次の番地の内容 β_0 を用いて番地 $\langle \beta_0, \delta, 1 \rangle$ (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 $\langle \beta, \delta, 0 \rangle$ 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ャ[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、結果バケット 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_0 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 Π_i ($i=1, \dots, n$): 計算要素
 DU: 分配ユニット Δ_k ($k=1, \dots, m$): 分配要素
 P_θ : 監視用演算装置 M: 記憶装置
 S_c : Cスケジューラ S_d : Dスケジューラ
 P_E : 発火情報 P_R : 結果バケット

図6 システム構成図

置 P_j とは図6中の bus-1 で接続され、 P_θ と各 Π_i 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 S_c は書き込み系 W と読取り系 R と呼ばれる2つの演算装置と P_E を蓄積する環状バッファ Q (N_c 個の 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 の書き込み位置 T_w と R の読取り位置 T_r の関係 $T_{w+1} \equiv T_r \pmod{N_c}$ が不成立であり、かつ1個以上の Δ_k から P_E の転送要求が出ているならば、これらの Δ_k から、 t と k とに依存する優先順位に基づいて1つの Δ_{k_0} を選択する。そうでなければ、(i) に戻る。(ii) その Δ_{k_0} から T_w が指す slot に P_E を書き込んで、 T_w を次 slot に設定する。一方、 R の動作は以下の繰返しである。(1) 関係 $T_w = T_r$ が不成立であり、かつ空き計算要素 Π_i が幾つかあるならば、これらの Π_i から適当な Π_{i_0} を選ぶ。そうでなければ、(1) に戻る。(2) T_r の示す Q の slot から P_E を読取って、それを Π_{i_0} に転送し、 T_r を次 slot 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 P_θ

P_θ は、動作系 O と解放要求バケット P_F (解放する領域の先頭番地と大きさからなる) の書き込み系 W の2個の演算装置とその他に P_F を蓄積する環状バッファ Q (N_p 個の slot 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 Π_i からの領域確保要求が出ているならば、以下の確保動作を行い、確保要求がなくかつ W の書き込み位置 T_w と O の読取り位置 T_r

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 Π_i のうち、 t と i とに依存する優先順位に基づいて1つの Π_{i_0} を選択する。 O は Π_{i_0} との同期通信を開始し、 Π_{i_0} 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 Π_{i_0} へ返した後に通信を終える。

解放動作: T_r が示す Q の slot から P_F を読取って解放領域を管理表に登録し、 T_r を次 slot に設定する。

また、 P_θ の書き込み系 W は、 S_c の書き込み系 W と同様の動作をとる。

4 関数型プログラムから

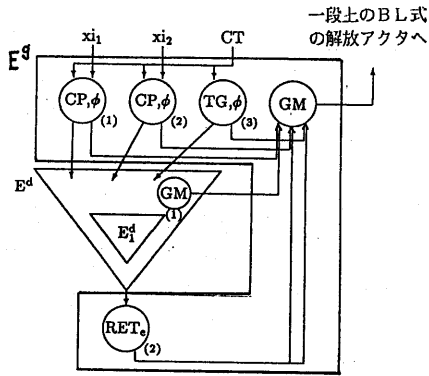
データフロープログラムへの変換

文献[1]のリダクション戦略の実行制御をDFアーキテクチャを用いて実現するためには、これまでのDFアーキテクチャ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。

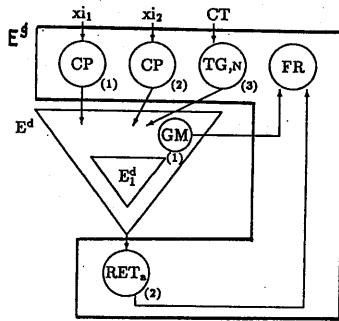
本マシンでは、(i)については、組込関数のその場合は要求駆動処理の計算要求に対応する制御トークン(CT)を特定のアクタに到着させることで実行制御する(4-1節)。利用者定義関数のその場合には、CTだけでは実行制御が行えないので、固有のアクタ(EV, INIT, RETe)を定義して関数呼出しの系列間でトークンを授受する機構を設けることにより解決している。更に、これらの固有のアクタは、GRマシンで実現される同一式の重複評価を避ける技法(式の共有)も実現させる。

(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き込みを防止し、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 $F(x_1, \dots, x_n) \leftarrow E$ の形式をもち、 F を関数名、 (x_1, \dots, x_n) を変数並びといい、関数本体 E には x_1, \dots, 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式ときのゲート(但し、 ϕ はTまたはF)



(ii) EがB式ときのゲート

図7 (ゲート) BL式EのゲートE^s; ()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指数の評価

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、遅延指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E₁とE₂について、E₁がE₂の一段上(又はE₂がE₁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: E₂はE₁中出现する真部分式(E₂ ⊆ E₁で表わす)、かつ、E₂ ⊆ E ⊆ E₁なるBL式EがE₁中出现しない。■

BL式Eを変換したDFプログラムE^dには、EのゲートE^sと呼ばれるアクタの集合が付加される(図7)。ゲートE^sの要素は、(i)E中出现する変数の個数(≧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中出现する場合、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指数(例1のifの第2引数中のyがこれにあたる)がE中出现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指数に対応する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RETe(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がないことを一段上の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^sはBL式Eに対して一意に対応するので、E^s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^s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^s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^s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1(1=1, ..., l_g-1)が付けられたBL式E_iのGMアクタにも同じ番号1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^s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^s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 ≤ j ≤ m-1なるjをもつE^s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^s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^s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^s中

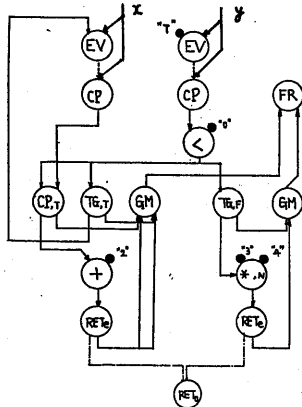


図8 式(1) $G(x,y) \leftarrow \text{if}(\langle y 0) (+ x 2) (* 3 4)$ を交換して得られるDFプログラム。但し、これ以降の図では、波線はアクタのdest-j域の内容が結果バケット中の結果として利用されることを表す。鎖線はアクタの引数が結果バケット中の行き先として利用されることを表す。

のGMアクタの $l_0 - 1 \leq j \leq m - 1$ なる j をもつ $\text{arg-}j$ 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 $\text{arg-}j$ 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 $\text{arg-}i$ 域 ($i = 0, \dots, m - 1$)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 $\min(i, l_0 - 1) + 1$ 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 は外部から与えられる引数トークン(但し、TGの場合は、交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 $\text{arg-}(l_0 - 1)$ 域の相対番地を指す。

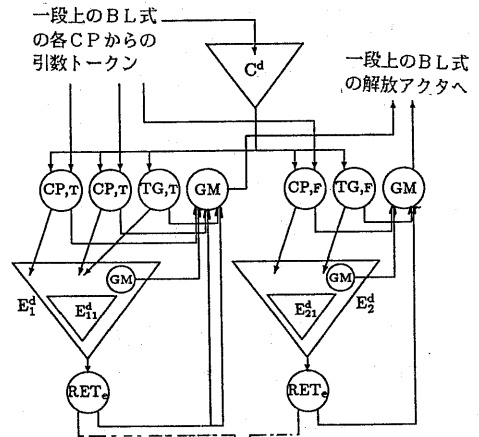
【例1】 図8に関数定義式：

$$G(x,y) \leftarrow \text{if}(\langle y 0) (+ x 2) (* 3 4) \dots (1)$$

を交換して得られる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)を交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 を用いて説明する。式($\text{if } C E_1 E_2$)を交換したDFプログラム



arg-1とarg-2から決まる同一の絶対番地

図9($\text{if } C E_1 E_2$)を交換して得られる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 と型的一致する各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($\text{if } C E_1 E_2$)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 $G(x_1, \dots, x_n) \leftarrow E$ を交換した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 ($k = 1, \dots, n$)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 域に書き込んでおき、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。RETe 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、ハッグ(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, \dots, 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_j 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_j^s が付加したプロ

グラム構造に変換される。遅延引数 E_j^d に対応する E_j^s 中の各 CP と TG アクタはいずれも T 型である。 E_j^s 中の RETe アクタは、 E_j^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_j^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_j^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($\langle \perp, \alpha \rangle$)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, \dots, 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 $\langle \perp, \alpha \rangle$ 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_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_f (そのベースを β_f とする)が確保され、G の原始 D ブロック D_g のハッグが D_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)か否かによって、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 $\langle \beta_f, \alpha \rangle$ 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

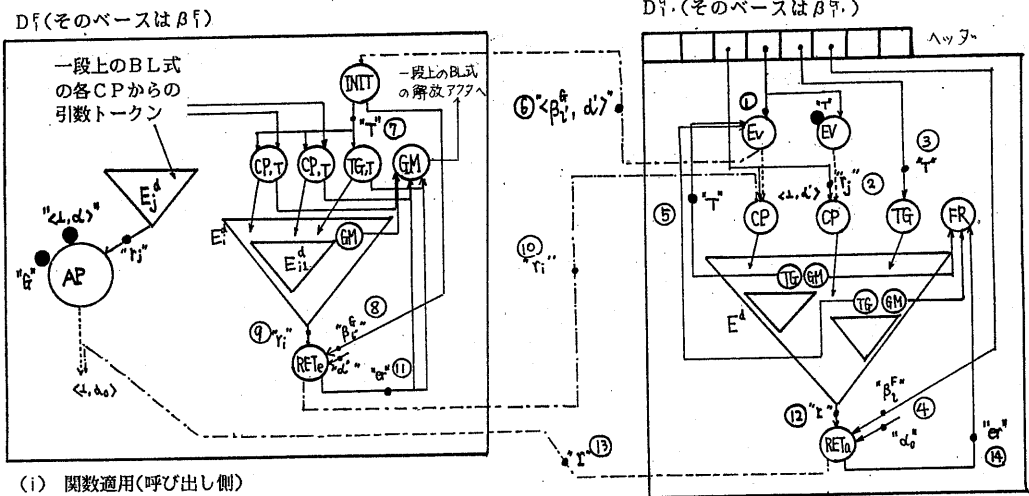


図10 関数適用(GE_1, \dots, E_n)の実行と遅延引数 E_j^d の評価
但し、 E_i は遅延引数に対応し、 E_j は遅延引数に対応しない。
この例では、 $i = 1, j = n = 2$ 。

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 B L 式の評価が開始される時、その B L 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 $\langle \beta_i^f, \alpha \rangle$)に番地型トークン $\langle \beta_i^f, \alpha \rangle$ (これは、G 本体 E のゲート E^f 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)。 $\langle \beta_i^f, \alpha \rangle$ を受理した INIT アクタは直ちに発火して、 E_i^f 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 β_i^f 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^f 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^f 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 を呼出した本体に戻し(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^f 中の FR アクタに通知する(図中⑭)。

EV に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 プロック D_i^f 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_i^f の解放を直ちに(P_i 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いて DF プログラム構造で構成されることを示した。ここでは、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実行し、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ー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 アーキテクチャの一提案を行った。その戦略の細込みについては、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こ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 Bruijn の表現による入式的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 calculus - 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 Misra, D. P. : "A preliminary architecture for a basic data flow processor", Proc. of the 2nd Annual Symposium on Computer architecture, pp126-132 (1975)
- [6] 曾和将容 : "データフローマシンと言語"、照光堂、(1986)