関数型プログラムの効率的実行機構について（各個研究メモ）

関数型プログラムの効率的実行機構について

久保正敏

一、はじめに

コンピュータシステムの規模増大につれてハードウェア・コスト低下の一方でソフトウェア・コストが増大し、現在はソフトウェア危機の時代であると言われている。ソフトウェア開発を容易にするための一つの有力な手法として、関数型プログラムが最近注目を集めている。本稿では、筆者が提案する、関数型プログラムの効率的実行機構について、簡単に紹介したい。

二、ソフトウェア危機と関数型プログラム

コンピュータのハードウェアは、真空管からＶＬＳＩ[1]へと大きな発展を遂げてきたが、この間、コンピュータ・アーキテクチャにもプログラミング言語にも大きな変化は起らなかった。このため、ＶＬＳＩの能力は益々強大になっているが、それを充分に生かしきるプログラミング体系が生まれていないというアンバランスが生じている。既存のコンピュータ・アーキテクチャ上で稼動するソフトウェアの開発・保守費用は増加する一方であり、今や図1に示す様に、コンピュータシステムのコストの殆どはソフトウェアに費されている。

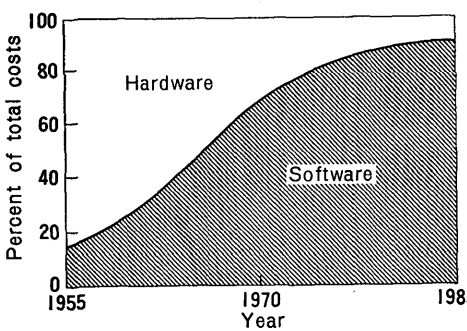


図1　ソフトウェアコストの増加傾向

この様な「ソフトウェア危機」と呼ばれる事態を打開するために、現在二つの方向が模索されている。その一つは、ソフトウェアの開発・保守を一貫して組織的に管理しようというものであり、誤りを発見しやすい形でプログラムを作成す

る技法や、ソフトウェア開発プロジェクトチームの組織法などが種々提案・開発され、「ソフトウェア工学」という新しい分野を形成しつつある。もう一つの方向は、既存のコンピュータ・アーキテクチャにとらわれず、人間が考えやすく記述しやすいプログラミング体系とその実行機構を考えようという方向である。

従来のコンピュータは、メモリ内に置かれたデータをCPU[2]が一度に一つ取り出し、計算し、結果を再びメモリに収めるという動作を繰り返す。ユーザは、このコンピュータの構造・動作を常に意識して、メモリからメモリへの写像を中心に計算手続をプログラムしなければならない。

これに対し、関数型プログラムにおいては、ユーザは、メモリの仲介なしに、数、数列、記号列等、計算の対象物（オブジェクト）を中心にプログラムを書くことができる。これは、オブジェクト指向記述と呼ばれている。プログラムは、元になるオブジェクトの定義、オブジェクトから別のオブジェクトを生成する関数の定義、および、結果を得るためにオブジェクトに次々と作用させていく関数の系列、等によって構成される。

この様な関数型プログラムは、コンピュータの内部動作、構造を意識しなければならない従来のメモリ中心記述に比べ、人間の思考に沿って計算手順を純粋に記述できるため、プログラムの誤り発見、訂正が容易であり、ソフトウェア生産性が向上すると期待されている。

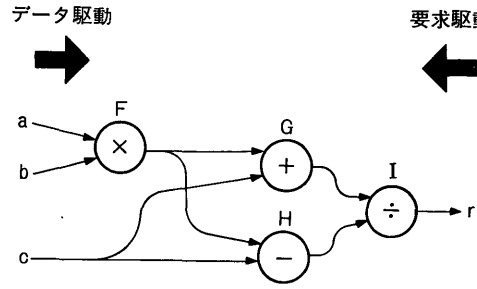
三、関数型プログラムの実行機構

前述した様に、関数型プログラムは、ソフトウェア作成という人間の負担を軽減することを目的としている。従って、関数型プログラムを実行する計算機構は、従来のコンピュータとは異なった種々の工夫、機構を導入して、今まで人間に課せられていた負荷を担わねばならない。即ち、ソフトウェアの負荷が軽くなった分、ハードウェアの負担が大きくなる訳であるが、現在のハードウェア技術は十分にこれに耐えることができるであろう。

この様に、ソフトウェア開発を容易にしたいというニーズと、ハードウェア技術の発展というシーズとを結びつけようとして、現在までに、関数型プログラム実行機構に関して、数多くの提案、開発が行なわれているが、それらは大きく、データ駆動型と要求駆動型の二つに分類できる。図2に示す様に、前者は、オブジェクト（データ）の到着に従って式の内側から（内側のカッコから）計算を進める方式であり、後者は、式の値を求めたいという要求が式の外側から与えられ、これが順々にカッコの内側へ伝播しながら計算が進められる方式である。

図2の様に、演算子又は関数を表すのにノードを、オブジェクトやデータの流れを表すのにアークを、それぞれ用いてプログラムを表現したものをデータフローグラフと呼ぶ。また、この図からわかる様に、関数GとHは並行して計算できる。即ち、関数型プログラムは並行処理可能性を秘めている点に注意したい。

さて、ここで筆者が提案したい計算機構とは、簡単に言えば前記二つの駆動方式を結合することにより、並行処理性を更に高めようとするものである。



$F(x,y)=x\times y$
$G(x,y)=x+y$
$H(x,y)=x-y$
$I(x,y)=x\div y$

$r=I(G(F(a,b),c),H(F(a,b),c))$

図2 $r=\frac{a\times b+c}{a\times b-c}$を求めるデータフローグラフ

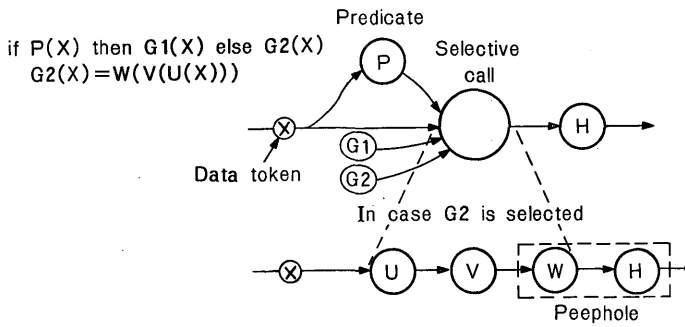


図3 データ駆動による関数の展開と最適化のぞき穴の設定

データ駆動方式は、データが到着すれば関数を起動してやるのであるが、この駆動方式による実行と並行して、データの流れの少し前方に「のぞき穴(peephole)」を設置し、その内部の関数系列を調べ、もし可能ならばこれをより効率の良い系列に最適化してやろうと言うのが、提案する方式であり、筆者はこれを最適化駆動と名付けている。

この方式は、図3に示す様に、データXが到着し、条件節が起動されて関数G2が展開された時点で、G2の最後尾に最適化のぞき穴を置き、G2の部分関数Wとその次の関数Hを結合してみて、可能ならば等価な効率の良い関数に変換するもので、いわば、データが関数の系列という道路を進行する際に、データの前方を走る露払いが関数を最適化して道路を通りやすくしてやる訳である。

ただし、データの進行速度、即ちデータ駆動による関数の実行速度に比べて最適化処理が遅く、露払いが却ってデータの進行を妨げるようでは意味がない。また、最適化という努力を払うに値する関数の等価変換規則が存在するかどうかも問題である。

これらを勘案して、プログラミング言語の一つAPLの持つ演算子を基本に関数型言語を考えることにする。APLの演算子は、ベクトルやアレイを対象とするため、その実行速度はさほど大きくはなく、最適化処理を行なって

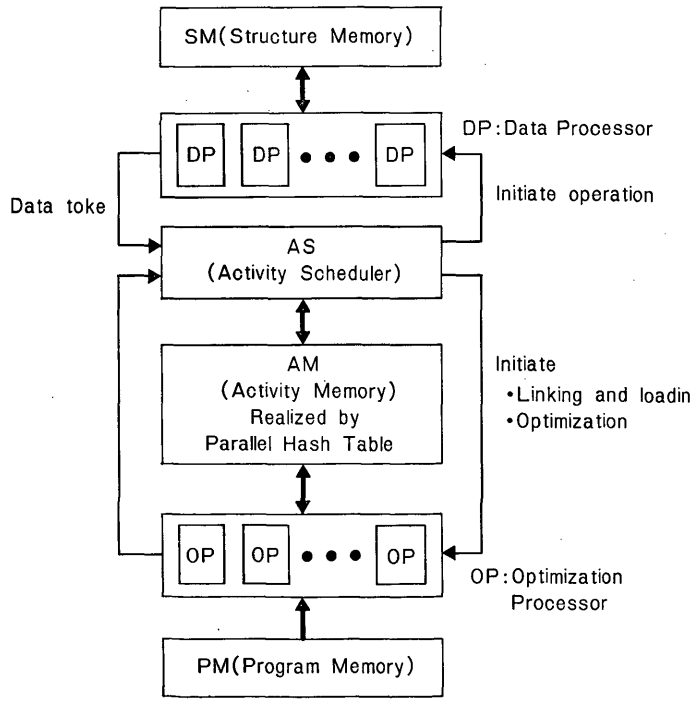


図4　データ駆動と最適化駆動を行なう関数型プログラム実行マシン

もデータ駆動実行の進行を妨げないであろう。筆者らは、ＡＰＬ演算子に関していくつかの等価変換規則を見出すことができた。ここでは詳細な説明を避けるが、これら最適化変換規則は、不要な中間結果の生成を避ける事を基本原則としている。

データ駆動による実行と最適化処理とを同時並行して行なう機構は、当然、両者を実行する二種類のプロセッサ群を必要とする。前者を担当するプロセッサをＤＰ（Data Processor）、後者を行なうものをＯＰ（Optimization Processor）と呼ぶことにする。データフローグラフの形で表現されしかもＤＰが即、実行可能な形に展開された関数型プログラムを保持するメモリＡＭを中心に、ＤＰ群とＯＰ群を結合した図４のブロック図が、関数型プログラムの効率的実行機構のモデル構造である。図中のＡＳは二種類のプロセッサ群に仕事を割り当てるスケジューラ、ＳＭはベクトルやアレイ等の構造化されたデータを保持するメモリ、ＰＭは関数の定義を含むユーザの書いた関数型プログラムを保持するメモリである。

四、むすび

書き易く読み易いプログラミング体系を求める動きの中で、関数型プログラムは最も革新的なものであり、それだけにその実行機構の研究は各所で始められたばかりで、決定打が出るのは数年から10年先になるであろう。筆者も、関数の性質を利用した効率的実行機構について提案したのであるが、その実現可能性についての検討はまだ不十分である。今後も、ハードウェア技術の発展を利用して、プログラミングという人間の負担を軽減できる計算機構の具体化に向けて検討を進めたい。その過程で、人文科学や民族学の研究に有用で使い易いコンピュータ・システムのあり方を模索したいと考えている。

註

(1) ＶＬＳＩ（Very Large Scale Integrated circuit）超大規模集積回路

(2) ＣＰＵ（Central Processing Unit）中央処理装置

（第五研究部）