抽象データ型支援アーキテクチャの一提案

4N-8

神田(こうだ)陽治　田中英彦　元岡達
東京大学　工学部

1° はじめに

コンピュータアーキテクチャは、プログラムの高速処理機能とともに、プログラミングの生産性を上げうる設計にすべきである[1]。抽象データ型を採り入れた高級言語マシンを対象とし、「抽象化」を支援できる設計を試みる。

2° 抽象化のアーキテクチャ支援

アーキテクチャの立場からみれば、プログラミングの作業の際に最も望まれる機能はすべてのプログラムが高速に走行し無駄な待ち時間をユーザに押し付けないことであり、加えて自由で高度な開発環境を整備するためにも「プログラムに書かれたユーザの意図を走行中のプログラムでも利用できる」構造を持つことだと考える。この構造の実現を、ユーザ定義した抽象オブジェクトを素直にアーキテクチャ上に展開すること、即ち抽象化支援アーキテクチャでおこなう。

3° アーキテクチャ構成上の基本理念 ― すべての事柄に名前を与える。

システム内に存在するすべての実体に一意名を割り付け、さらにシステム内で区別すべき関係にも名前を与える。†要は区別すべき事柄には名前を付け、その名前の提出によって確認がとられる。いわゆるタグマシンも名前を使うが、それは主にシステム定義のデータ型に与えることに止まっていた。

4° アーキテクチャの全体構成

4層に区分する。3層はこのアーキテクチャで重要な役割を持つ名前の管理をする。0層はマルチプロセッサ構造を決める[1]。以下で1,2層の基本概念の一部を紹介する。

3 type management:software
2 instruction set:firmware
1 addressing rule:firmware
0 multi-processor:hardware

5° アドレッシングルール

名前が指す実体へのアクセスは、systemDirectory を使うロック付のケーパビリティ機構と、名前の別名を記録している systemAliasSelector でおこなう。

systemDirectoryEntry は、ロック名と実体へのポインタからなる。そして、systemAliasSelectorEntry は、キー名前対と値名前対であり systemAliasSelector にはサーチ機能と既約化機能（2つの Entry x,y に対して if x.valuePair = y.keyPair then x.valuePair := y.valuePair;）が定義されている。

実体へのアクセスには、実体名と、ロックに合うキー名からなる名前対を提示する。ただしアクセス環境（コンテキスト名で区別）単位に名前に別の名前対（別名と呼ぶ）が連想的に付けられていることがあり、この提示した名前対の実体名に別名があれば、この連想された別名を提出された名前対のかわりとしてアクセスをおこなう。この別名を覚えることは、systemAliasSelector に、実体名とコンテキスト名をキー名前対とし、別名（これは名前対である）を値名前対とする Entry を登録することで実装する。

[1] 情報処理学会第25回全国大会　1F-7

† 関係とは例えば（6°で説明する）実引数と仮引数の引数型が一致すべきこと

6° マシン命令セット層　抽象化支援用命令について

抽象データ型の命令は2つに大別されるが、今回はモジュール支援用の命令について解説をする。[ii]

抽象データ型はアーキテクチャ上ではモジュールに展開される。モジュールはいくつかの手続きを含み，新しいデータ型を定義する。モジュール呼び出しの引数結合は **pushM** 命令により一個ずつ順におこなわれていき、発火条件を最初に満たした手続きが選択され自動的に起動される。結果が利用できるように名前に別名を付ける操作は、**popM** 命令と **assignM** 命令でおこなう。[iii]

- **pushM** 命令の動作：　**pushM** *aName* **as** *aType* **to** *aModule*　(**in** *outerContext*)

outerContext の環境で *aModule* なるモジュールへ *aName* を *aType* を引数型として、結合する。モジュールはそれが許す引数型を表にした **typeTemplates** を持ち，*aType* が正しい引数型か否かの検索の後で，*aName* に別名があればその別名を，なければ *aName* と *aType* とを、結合結果の名前対として **typeTemplates** へ覚え込む。また、発火条件を表にした **runTemplates** も持ち、**pushM** 命令実行ごとに結合済パタンと発火パタンの一致を調べ．満たすパタンがあれば対応する手続きを起動する。[iii]その結果，新しいコンテキストが作られて、結合された名前対を小さな数で参照できるアクセス環境が生成される。コンテキスト名 *outerContext* は、スタックされ *innerContext* = *aModule* が新しいコンテキスト名となる。

- **assignM** 命令の動作：　**assignM** *firstShortInt secondShortInt*　(**in** *innerContext*)
- **popM** 命令の動作：　**popM** *aResultName* **as** *aType* **from** *aModule*　(**in** *outerContext*)

assignM 命令の2つの引数は、2つの名前対 *DontCareName*，*aType* と *aName*，*aKey* を指定するものとする。**assignM** 命令は **systemAliasSelector** に *innerContext* と *aType* をキー対，*aName* と *aKey* を値対とする **Entry** を置く。**popM** 命令は、**systemAliasSelector** に，*aResultName* と *outerContext* をキー対，*aModule* と *aType* を値対とする **Entry** を置く。この2つの **Entry** が既約化されて、*aResultName* に（*outerContext* でのみ通用する）別名，*aName* と *aKey* からなる名前対が付く。

- タイプマネージャ機能の実装

このモジュールが管理する抽象データ型名を *NewType* とする。この型の実体（名が *anInstance*）の生成時に **systemDirectory** へロック名が *NewType* の **Entry** を登録し、**systemAliasSelector** に、*anInstance* と *aModule* をキー対，*anInstance* と *NewType* を値対とする **Entry** を登録すれば、このモジュールのみが実体へアクセスできる。*anInstance* をコンテキスト生成時ごとに異なる名前にすれば局所的な実体になる。

7° まとめ

考察中のアーキテクチャの一部を紹介した。アーキテクチャの問題点を見つけ完全なものとするために、現在 Pascal 言語で論理動作のシュミレータを、作成している。

ii　他の一群はイベント支援用の命令である。

iii　呼び出し時の負荷の分散が狙い。並列性がありデータフロー制御を仮定する。

iiii　モジュール起動後，次回の発火に備えて typeTemplates は初期化される。