3R-6

並列オブジェクト指向言語 DinnerBell ——その設計

河野真治・金子誠治・神田陽治　・田中英彦・元岡達
東京大学工学部

〔0〕 DinnerBellの目標

DinnerBell〔1〕はソフトウェア生産性の向上をアーキテクチャとソフトウェアの両面から達成することを目指す **Oraga** 計画のプログラミング言語である。**Oraga** とはObject Oriented Architecture to Govern Abstractions の略称である。**Oraga** は4つのサブシステム，即ち並列処理向きオブジェクト指向言語であるDinnerBell, NameMaster, ObjectPeeper, そしてシステムの実行機構であるOragaltself より成る。

DinnerBellには言語の表現能力としてデータ及び処理フローの抽象化，並列処理の記述の能力と共に，**Oraga** 計画の他の3つの部分との協調性が要求される。

〔1〕 DinnerBellの特徴

オブジェクト指向と並列処理を融合させるためにDinnerBellでは単一代入規則を採用する。さらに2つのオブジェクト間で複数のメッセージを直接に通信する場合はその順序を保証する。間接に通信するオブジェクト間の順序化はSecretary と呼ばれる特別なオブジェクトを導入する。Secretary は特定のメッセージを制御する事により順序化する。順序化の方法としてメッセージの応答待ちを行う事が考えられる〔4〕しかし，この方法は順序化に必要とされる以上に並列性を犠牲にしてしまう。

〔2〕 DinnerBellの構文

DinnerBellの構文はビットマップディスプレイを考慮した視覚的に優れた表現を持っている。 図1は，DinnerBellで記述したDining Philosopher問題の”fork”の部分である。

DinnerBell のオブジェクトはitBlock と呼ばれ，3層の構造（itBlock,method,statement）を持つ。それらはそれぞれ別々の四角い枠により囲まれる（図2）。変数のスコープはこの3層で決まり，それぞれitBlock variable,method variable,statement variable と呼ぶ。影付けされている部分はcriticalRegieon を表す。DinnerBellのメッセージはkey のついた引数とevent である。key には”：”が，event には”！”が識別子の後に付く。代入もメッセージにより行うので代入される変数もkey を持っている必要がある。これによりベクトル代入も可能となった。受取りパターンは下線を引くので容易に識別できる。メッセージパターンとその実行式の間にはNeckと呼ばれる小さな四角（□）が置かれる。

DinnerBell にはオブジェクトと変数の名前の明示的な宣言はない。システム全体で有効な名前はオブジェクトの名前だけでありNameMasterにより管理される。Smalltalk のクラスとインスタンス，ブロック文とオブジェクトの区別はDinnerBellでは重要なものではない。

変数のスコープは3 種しかないので '～' を一つ又は二つつけることにより表す。 '～' の多い方がスコープが広い。またオブジェクト間の協調動作を可能にする為の単一代入はitBlock,method,statement各々のレベルで行う。

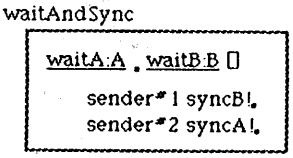


図3

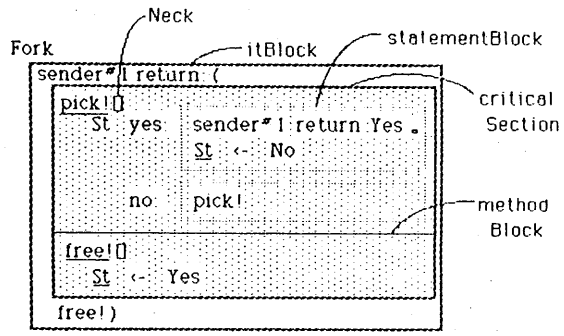


図1

Block structures of DinnerBell

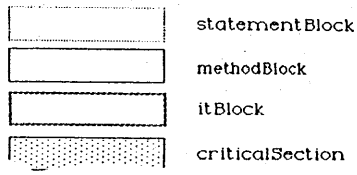


図2

〔3〕 DinnerBellの実行

DinnerBell はメッセージの起動を繰り返すことにより実行される。メッセージ送信は全て並列に実行されるが非決定性の軽減の為にいくつかの規則がある。

ひとつは直接のオブジェクト同志の通信は順序を保存する事である。これによConcurrent Prolog 〔3〕のようなリストによる繁雑なストリームの配管を簡単化することができると期待される。

次にメッセージの待ち合せにおいて同じオブジェクトからのメッセージのみ待ち合うか，異なるオブジェクトからのメッセージも待ち合うかを制御することができる。〔図3〕複数の異なるオブジェクトよりメッセージを同期して受け取る場合はメッセージパターンを','で区切ればよい。 どのオブジェクトから来たのかを判定するためにsender#n と言う擬似変数が用意されている。nはパターン上での出現順に番号である。

排他制御の為にはcriticalRegionが用意されているが，これはオブジェクトにロックを掛けるのではなく，そのコードに複数のコンテクストがはいるのを防ぐ為にある。従って，データベース等のためにロックを必要とする為にはロックを掛けるオブジェクトが必ず通るオブジェクトをつくりそのオブジェクトをcriticalRegionに設定する。

Smalltalk のBlock とオブジェクトを統一する際に問題となるのは生成と発火である。itBlock ではメッセージを送られた時には発火のみ行い生成はしない。生成はif文等で引数として渡された時にその環境で必要な擬似変数（自分自身を表すitSelf等）を束縛しながら行われる。

itBlock は再帰呼出しが可能であるがその際，自分を生成したい場合としたくない場合がある。itSelf#1 とかくと自動的に自分のコピーをつくる。Block 内でReceiverを省略した場合は，コピーをつくらないgoto statementに近いものとなる。

(4) DinnerBell Interpreter

DinnerBellの作成は現在2段階で行っている。 一つは文法に近いレベルのインタプリタであり，もう一つはハードウェアシミュレータである。ここではインタプリタの構造を解説する。オブジェクト指向言語ではプログラムの実行はメッセージ送信により行われる。 =>によりメッセージ送信を表し，Mをメッセージ，Dを受信オブジェクトとする。

M=>D

これを事件という。 通常，ひとつの事件は複数の事件を引き起す。

M=>D --> M' =>D' , M"=>D"

-->をcaused event relation という。どのような事件が引き起されるかはDとMにより決まる。return valueがある場合はactor model (2) では次の様になる。

M=>D --> Answer=>Reply-to
(Actor)

しかしReply-toをつねにMにいれておかなければならない。そこでAnswerを必要とする事件はAnswer=>Reply-toを先に起動する事にし，その実行をsuspend する機構としてconcurrent Prolog のread-only-variableと同種の考えを導入した。即ちinstantiation の為の記号 {} を導入して次のように記述する。

M=>D, Answer=>Reply-to
-->{Answer} (DinnerBell)

MとDとAnswerの3つの組だけが M=>D事件に関係する。 本来，Reply-toオブジェクトはDとは独立なのでこの点からみてこちらの図式のほうが良いと考えられる。Answerのinstantiate は他のsuspend された事件を起動するのでM, D, Suspend-list のように考えることもできる。 この3つ組を context とよぶ。Smalltalk の context に代る概念であるが スタックで構成することはできない。これをつぎの様にかく。

context (M, D, A)

DinnerBellの実行はたとえばつぎのようなcontext の変換の連続と考えられる。

context (M, D, A) -->
context (M', D', A'),
context (M", D", A"), . . .

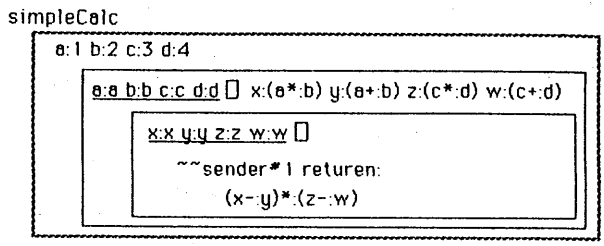
Aは他のcontext のM, Dとして現われることができ，その場合，そのcontext はAがinstantiate されるまでsuspend する。

context (M, D, A)
|
context (A , D' , A')
|
context (M, A' , A")

実際に展開した例を図4に示す。

(5) おわりに

context による実行はPrologによるシミュレータが動作しており，DinnerBellのインタプリタはCで現在製作中である。




```
time 1:context([_129,_180,_231,_282],
        [_108,_109,_110,_111]=>
              (_108-_109)*(_110-_111),_28),
        context([*,1,2],primitive,_129),
        context([+,1,2],primitive,_180),
        context([*,3,4],primitive,_231),
        context([+,3,4],primitive,_282)
time 2:context([*,_434,_473],primitive,_28),
        context([-,2,3],primitive,_434),
        context([-,12,7],primitive,_473)
time 3:context([*,-1,5],primitive,_28)
-5  --- answer
```

図4

参考文献

(1) 神田陽治
並列オブジェクト指向言語DinnerBellの概要
情報処理学会ソフトウェア基礎論研究会11-3

(2) Akinori Yonezawa
An Object Oriented Approach for Concurrent Programming
Department of Information Science
Tokyo Institiute of Technology
Research Report C-63 Novemver,1984

(3) Ehud Y.Shapiro
A subset of Concurrent Prolog and its Interpreter
Institute for New Generation Computer Technology
Technical Repot TR-003

(4) 横手靖彦
Concurrent Smalltalk
日本ソフトウェア科学会第1会大会 2E-2