5T-5

"DinnerBell + NameMaster + ObjectPeeper" による統合プログラミング環境
— ユーザインターフェース

阿部雅彦　神田陽治　田中英彦　元岡達
(東京大学　工学部)

1. はじめに

DinnerBell, NameMaster, ObjectPeeper は、われわれの研究室で研究中のオブジェクト指向計算機のプログラミング環境を構成するものである。プログラミング環境としては、エディタ、コンパイラ、デバッガ、ライブラリ等があるが、並列処理下で特に問題となるのはデバッガであろうと考えられるので、本稿ではデバッグを中心に述べる。

2. ObjectPeeper

ObjectPeeper は DinnerBell のエディット、デバッグの目的でオブジェクトの表示を行うツールである。これまでの研究では[1]、テキストの表示についてのみ考えられていたが、オブジェクトを対象とした場合次のような情報を表示しなければならない。

① オブジェクトが属するクラス、およびそのスーパークラス、サブクラス
② メッセージ・キューの内容。メッセージ送信オブジェクトも含めて表示する。
③ ステートメントの実行状態
④ 変数名と値

これらのうち、③、④が問題である。

(1) 実行状態の表示

DinnerBell は並列型の言語であるから、一時に複数のステートメントの実行状態を考えなければならない。各ステートメントは、実行待、実行中、実行完了の三つの状態をとりうるが、これをわかりやすく表示しなければならない。

(2) 変数と値の表示

DinnerBell の変数の値はオブジェクト・ポインタであり、何らかのオブジェクトを指している。指されるオブジェクトを acquaintance という。例えば図1の変数 anArray は objectB を指し、objectB はさらに番号付けられたオブジェクトを指している。一般に変数はオブジェクトから成る構造を指している。構造の末端は、整数やシンボルの様な特別な種類のオブジェクトである。ObjectPeeper はこのような構造を適切に表示することを要求される。

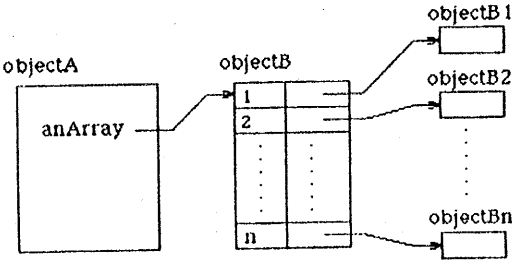

図1　acquaintance構造の例

おおまかには次のようにして表示を行う。図1の例でいうと、objecA は objectB に例えば show のようなメッセージを送る。objectB はさらに objectB1 から objectBn に対して show を送る。すると末端のオブジェクトは自分自身を表示形式にして返す。objectB はその表示形式を受け取ると、それらを組み合わせて自分の表示形式を作り、それを objectA に返す。表示形式作成の際にはパレットと NameMaster の情報を参考にする。パレットはオブジェクト表示のためのテンプレートで、クラスごとに作られる[1]

"クラス ObjectPeeper のメソッド"
show: anObject in: window
　・　・　・
sizeTree ← anObject sizeOf: aPart.
displayObject ← show: aPart in: area arrange: sizeTree.
window open: displayObject.
　・　・　・
図2　Smalltalk 流 Syntax で示したアルゴリズムの概略 (1)

以上のアルゴリズムの概略を図2，3に示

す。(第一版はC言語で記述し、実験済) ObjectPeeperはウィンドウのサイズに応じた表示を行うため、処理は2passになる。pass1でオブジェクトのサイズを計算し、pass2でウィンドウ・サイズの割りふりと割りふられたサイズ内に表示するための処理を行う。

"各クラスのインスタンス・メソッド"

```
sizeOf : aPart
    | tree subTree |
    (palette = nil) ifTrue: [palette ← self class palette].
    (自オブジェクトがリーフ)
        ifTrue: [tree ← palette の指示をもとにサイズを計算].
        ifFalse : [tree ← acquaintance 以外についてサイズを計算.
                   subTree ← aPart sizeOf : Acquaintance.
                   tree addSubTree : subTree]
    ↑tree
show : aPart in : area arrange : tree
    | displayObject sub acquaintance aPart' subArea subTree
    [palette と tree を見てサイズを割りふる]
    sub ← acquaintance show : aPart' in : subArea arrange : subTree.
    displayObject ← [acquaintance 以外について表示形式に変換]
    ↑displayObject addSub : sub
```

図3 Smalltalk流 Syntaxで示した表示アルゴリズムの概略 (2)

3. 並列処理下のデバッグ

(1) 同期の検証

並列言語におけるデバッグは、排他制御および順序化が正しく行われているかを調べることが重要である[2]。オブジェクト指向言語での同期はメッセージによって行われるから、デバッグ時にはメッセージがどのオブジェクトからどのオブジェクトへいかなる順番で送られたかを調べることになる。一般にメッセージは送信された順に受信されるという保証はないから、送信側と受信側の両方で順序を調べる必要がある。本システムでは次のような機構を組み込む。

① 送信側：特定のメッセージ送信が行われたら時間とともに記録する。

② 受信側：特定のメッセージを受信したら、送信元，時間とともに記録する。

以上のような機構により、一種のデータベースができる。ユーザは問合せ言語を用いて事象の順序を調べる。メッセージは送信されてから受信されるから、メッセージ送受の関係を調べることにより、離れたオブジェクト間での実行の前後関係を調べることができる。これは従来のトレース機能の拡張にあたる。

(2) オブジェクトの停止

2.で述べたように、ObjectPeeperはオブジェクトを表示して内容を調べ，デバッグするツールである。ObjectPeeperでオブジェクトを見る時、見ている内容はオブジェクトの現在の姿でなければならないから、オブジェクトを停止する必要がある。このため、オブジェクトには、特定のメッセージ送信が行われたら停止するような機能を設ける。これは従来のブレーク・ポイントに相当する。

ここで問題となるのは、停止させる範囲をどのように決めるかである。具体的にはacquaintanceのオブジェクトを止めるかどうかが問題である。acquaintanceの状態も含めてオブジェクトの状態であるという意味ではacquaintanceも止めなければならない。しかし出力デバイス等の共有されるシステム・オブジェクトは止められると、システムからの出力が得られないなどの不都合が生じる。

今後の実験では、止めるオブジェクトの範囲を変えて、その効果を確認する予定である。

4. 現状と今後

デバッグについては、組み込む機能の評価が必要である。また、3.で述べた問合せ言語についても検討を行う。ObjectPeeperとNameMasterの関係についてはさらに具体的な設計を行う。実装については、現在プログラムの中間木を直接実行するインタプリタを作成中である。ObjectPeeperはこのインタプリタからの情報を利用して動く。実装言語はどちらもC言語を用いる予定である。

5. 参考文献

[1] 阿部，他："オブジェクト指向言語に向いたディスプレイエディタ"，情報処理第29回全国大会 1984，4R-6

[2] 神田，他："並列オブジェクト指向言語 DinnerBellの概要"，ソフトウェア基礎研究会 1984年12月14日