# 操作列スライシングに基づくマクロ定義

3Q－1　杉浦 淳　古関 義幸
NEC C&C研究所

## 1. はじめに

ソフトウェアアプリケーション上で繰り返し行う操作をマクロとして登録しておくことは、作業の効率化に有効である。例示プログラミング (PBD: Programming by Demonstration) [1] はアプリケーション上でのユーザの操作からプログラムを生成する手法であり、PBDによりプログラミング技術のないユーザでもマクロを作成することが可能となる。このような特長のために、GNU Emacsなどのシステムに PBD機能が実装されてきた。しかし、多くの繰り返し操作が行われるにもかかわらず、これらのシステムではPBD機能は有効に活用されていない。

この原因の一つは、マクロ定義において操作記録の開始/終了を指定しなければならないことにある。これは、将来繰り返されるであろう操作をユーザが予め想定しなければならないことを意味する。しかし、同じような操作を2回目に繰り返す時になってはじめて、先ほどの操作をマクロ化すべきだったことに気付くのが普通である。また、仮に想定できたとしても、ユーザは例示操作の手順をプランニングし、その手順通りに正確に例示することが要求される。これはユーザにとってかなりの精神的負担となる。この問題を解決するために、本稿では操作列スライシング[2]を用いたマクロ定義手法を提案する。この手法は、筆者らが開発したPBDシステムDemoOfficeに実装されている。

## 2. 例示プログラミングシステム DemoOffice

### 2.1 概要

DemoOfficeは、電子メールおよび表(関係データベース)を統合したPBDシステムである。ユーザはこれらのツールでテキスト入力やデータのdrag&dropなどの編集作業を行うことができる。本システムでは、過去の操作が再利用可能とユーザが気付いた時点で、最小限の手間でマクロを作成できることを目的としている。

DemoOfficeでは、マクロ定義での操作記録指定を不要とするために、システムがユーザの操作を常に記録しておき、記録された操作履歴から必要な操作のみを抽出してマクロを定義するという方法をとる。DemoOfficeの特徴は、操作抽出のために操作列スライシング手法を用いていることである。

操作列スライシングは、特定のデータの作成に関連する操作のみを操作履歴から切り出す手法である。この手法により、ユーザは作業中のウィンドウで注目するデータを指定するだけでマクロ定義を行うことが可能である。具体的には、ユーザは特定のデータをウィンドウ内のマクロバー(図1b)と呼ばれる領域にdrag&dropすればよい。

システムは抽出された操作を一般化し、実行可能なマクロに変換する。

### 2.2 例題

ここでの例題は、電子メールとデータベースを用いた集計作業である。ある人の歓送パーティを行うことになり、その出欠を電子メールで確認するといった状況を考える。最初にユーザが次のメールを友人に送ったとする。

> *Hi all,*
> *I will be giving a farewell party for Mary next Sunday. Please let me know if you can come or not. Anything you might bring to drink would be appreciated.*
> *Regards, Atsushi*

ユーザの目的は返信メールの中から必要なデータだけをデータベース表に挿入することである。例示のトレースを図1に示す。

1番目の返信メールでは、ユーザはその内容から判断して出欠の回答"Yes"を表に入力し、次に電子メールアドレスおよび差入れの飲み物"red wine"をそれぞれ選択し、表へdrag&dropでコピーする (図1a)。

ユーザは2番目の返信メールを開いた時(図1b)、先ほど1番目のメールに対して行った操作の中で、表のセル(1, 1)および(1, 2)を作成した操作("Yes"を入力した操作と電子メールアドレスをコピーした操作)をマクロとして再利用できることに気付く。そこで図1bに示すように、表のセル(1, 1)および(1, 2)を選択し、メールウィンドウのマクロバーへdrag&dropする。システムは、これら2つのセルの内容に影響を与えたユーザ操作を操作履歴から抽出し、マクロバーに"Macro1"を生成する。

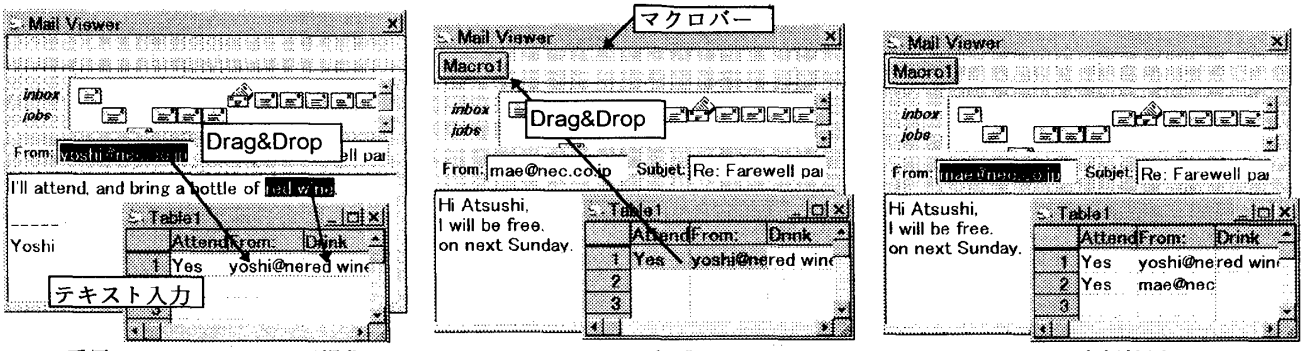
ユーザはMacro1ボタンをクリックしてマクロを実行することにより、2番目の返信メールに対しても1番目のメールと同様の処理を行うことができる (図1c)。

### 2.3操作記録

システムはユーザが行った全ての操作を記録する。DemoOfficeはデータの直接操作で編集作業を行うシステムであるため、記録される操作は、操作と操作対象の組で表現される。図2は図1aのユーザの操作記録である。

---

Simplifying Macro Definition in Programming by Demonstration
Atsushi Sugiura and Yoshiyuki Koseki
C&C Research Laboratories, NEC Corporation
4-1-1 Miyazaki, Miyamae-ku, Kawasaki 216, JAPAN




a. 1番目のメールでのユーザ操作　b. マクロの作成　c. Macro1の実行結果

図1. ユーザの例示

```
1: openTable("Table1")
2: openMail("Mail Viewer", 153)
3: cell=selectCell("Table1", 1, 1)
4: inputText("Table1", cell, "Yes")
5: text=selectText("MailViewer", CONTENT, 84, 92)
6: drag("Mail Viewer", CONTENT, text)
7: drop("Table1", 1, 2)
8: text=selectText("MailViewer", FROM, 0, 19)
9: drag("Mail Viewer", FROM, text)
10: drop("Table1", 1, 3)
```

図2. 操作履歴

### 2.4 操作列スライシング

操作列スライシングは注目するデータの作成に影響を及ぼした操作のみを操作履歴から切り出す手法である。この考え方は、プログラムスライシング[3]に類似している。プログラムスライシングは、プログラム中のある文の実行に影響を与える可能性のある文だけを残して他の文を除去する技術である。しかし、スライシングの手続きは、操作列スライシングの方が遥かに単純なものですむ。プログラムスライシングでは、ループや条件分岐などの制御構造、ポインタなどの複雑なデータ形式、複数手続き間にまたがる影響などを考慮しなければならない。これに対し操作列スライシングでは、データ間の依存関係を調べるだけでよい。以下、操作列スライシングで注目するデータをスライス基準と呼ぶ。

表やメールの内容が変更されるのは、テキストが入力されるか、データがドロップされた場合である。そこで、スライシングではそれらの操作(inputText, drop)を起点とし、操作履歴を溯って関連する操作を抽出する。例えば、図1の表Table1のセル(1, 1)がスライス基準である場合、セル(1, 1)へのテキスト入力である図2の操作4を切り出し、さらに変数cellによる依存関係から操作3を切り出す。

drop操作が基準となる場合は、変数による依存関係に加え、さらに処理が必要となる。この場合、drop操作の直前のdrag操作も関連するため、drag元のデータを新たなスライス基準に設定し、再帰的に操作抽出をする必要がある。例えば、スライス基準が表のセル(1, 2)である場合、セル(1, 2)へのdrop操作である操作7を切り出した後、直前のdrag操作6とdragするデータを選択した操作5を切り出す。さらに、dragしたのデータの作成に影響を与えた他の操作を切り出すために、"Mail Viewer"のCONTENTテキストを新たなスライス基準として操作5以前の履歴を検索する。ただし、図2の操作履歴にはメールの内容を変更する操作は行われておらず、これ以上抽出されない。

第2.2節での例題では、Table1のセル(1, 1)および(1, 2)がスライス基準となっており、結果として操作3-7が抽出される。

### 2.5 一般化

一般化はPBDシステムでの重要なプロセスの一つである。特定のデータに対して行われた操作を他のデータに適用するためには、ユーザの操作を抽象レベルで解釈する必要がある。

DemoOfficeで一般化の対象となるのはテキスト選択と表のセル選択の2種類の操作である。テキスト選択に関しては、行の先頭/末尾、テキストの先頭/末尾、選択個所の前後の単語といった情報を用いて一般化を行う。例えば、図1aでメールウィンドウで電子メールアドレスのテキストを選択した操作に対しては、「FROMフィールドの先頭から末尾までを選択」というように一般化を行う。

また、表のセル選択に関しては、最終行/列、フォーカスがあるアクティブな行/列といった一般化を行う。例えば、図1aで電子メールアドレスをdropしたセル(1, 2)を選択した操作に関しては、「表の最終行を選択」という解釈を与える。

## 3 おわりに

本稿では、操作列スライシングを用いてマクロ定義を行う手法について述べた。本手法により、ユーザは操作記録の指定不要となる。今後は、再利用性が高いと思われる一連の操作をシステムが検出してマクロを自動生成する手法を開発していく予定である。

## 参考文献

[1] Cypher, A. ed. Watch What I Do: Programming by Demonstration. MIT Press, 1993.

[2] Sugiura, A. and Koseki, Y., "Simplifying Macro Definition in Programming by Demonstration," Proc. of UIST'96, 1996 to appear.

[3] Weiser, M., "Program Slicing," IEEE Trans. on Software Engineering, Vol.SE-10, No.4, 1984, pp.352-357.