Visual BUC Shell

6G−8　ー対話履歴図形表示によるボトムアッププログラミングー

湯浅寛子　岩嵜正明
日立製作所中央研究所

1.はじめに

我々はプログラミングを行なうエンジニアや研究者を対象とした思考支援システムの研究を行なっている。この中で現在、ユーザが例示した作業からプログラムを生成するPBE(Programming By Example)機能[1]の実現を検討している。

PBE機能を応用したシステムは幾つか知られている[2],[3]が、これらは基本的に記録した作業手順をそのまま再生するものである。このため記録中の試行錯誤過程が生成されたプログラムに組み込まれるという問題がある。

我々はこの問題を解決するため、過去の対話履歴からユーザの指示に従ってプログラムを生成するBUC(Bottom Up Compose)機能[4]を提案した。今回、我々はこのBUC機能を組み込んだ対話シェル(= *Visual BUC Shell*)をLISP上に試作した。*Visual BUC Shell* は、「実験的にコマンドを実行して試行錯誤を重ね、所望の結果が得られたら、その結果を出力するプログラムをコンピュータに生成させる」というボトムアップのプログラミングスタイルを可能にする。

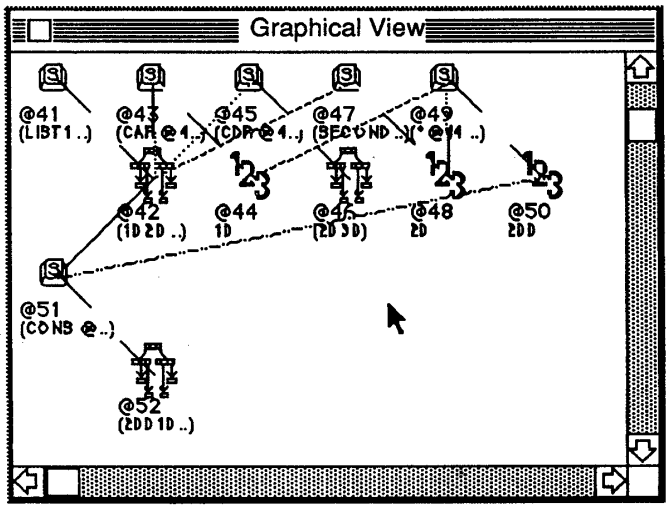
2.*Visual BUC Shell*

Visual BUC Shell の特長は、対話履歴を図形表示することによって、ユーザが対話履歴へ容易にアクセスできる点である。

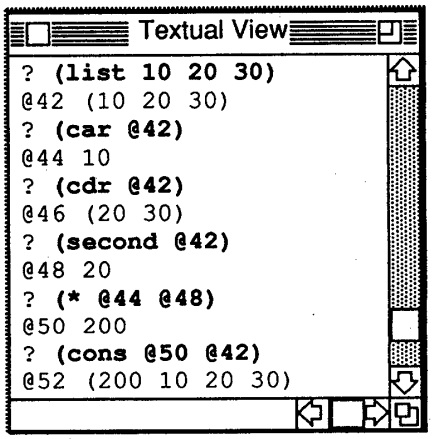
2.1 対話履歴の図形表示

図1に示す様に *Visual BUC Shell* は Textual View と Graphical View の二つのウィンドウを備える。Textual View は通常の対話シェルと同様にテキストベースでコマンドを入力したり、その結果をコンピュータが表示したりするウィンドウである。

Graphical View は対話履歴を図形として表示するウィンドウである。図1に示す様に、ユーザが Textual View にキー入力したコマンドとその実行結果をそれぞれア



・ユーザの入力をキーの形のアイコンで表わし、コンピュータの出力を実行結果の型に応じたアイコンで表わす。
・アイコン間の線分は、アイコンの論理的関係を表わす。



・「?」で始まる太字の行はユーザの入力で、「@」で始まる細字の行はコンピュータの出力である。
・「@」で始まる番号は入出力を一意に識別する識別子である。

本 *Visual BUC Shell* はApple社の Macintosh上に Allegro Common Lisp を用いて試作した。

図1　*Visual BUC Shell* の画面イメージ

Visual BUC Shell - Bottom Up Programming Approach with Graphical Interface
Hiroko YUASA, Masaaki IWASAKI
Central Research Laboratory, Hitachi, Ltd.

イコンとして表示する。また同図に示す様に、コマンドや実行結果の間の論理的な関係をアイコン間を結ぶ線分として表示する。アイコン間を結ぶ線分には生成関係を示す線分と参照関係を示す線分がある。生成関係は各実行結果がどのコマンドによって生成されたかを示す関係である。参照関係は各コマンドが先行するどの実行結果を参照しているかを示す関係である。

これらのアイコンや線分はユーザがコマンドを入力する度に逐一表示される。ユーザはGraphical View 上に表示されたアイコンをマウスでクリックすることによって、過去に入力したコマンドやその実行結果にアクセスすることができる。

2.2 プログラム生成の基本操作

次にGraphical View 上に図形として表示された対話履歴(以後これをリンクネットとよぶ)からプログラムを生成させる方法を説明する。

まず、ユーザはリンクネット上で実行結果のアイコンをマウスでクリックして選択する。選択された実行結果をターゲットと呼ぶ(一般にターゲットは、試行錯誤を伴うコマンド投入によって得られた所望の実行結果である)。

ユーザがターゲット上でマウスボタンをダブルクリックすると、*Visual BUC Shell* はターゲットと論理的に連鎖しているアイコンに図2に示すように色を付けて表示する。この色を付けた部分をパスと呼ぶ。パスはリンクネットのうちターゲットの生成に関係した部分リンクネットを表わす。

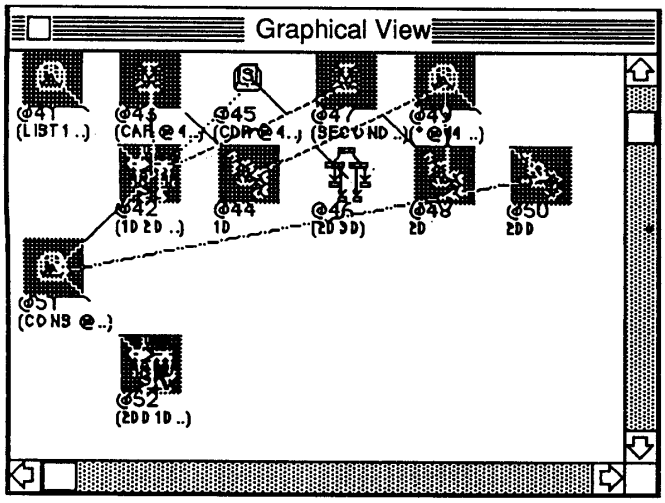


図2　パスの表示例

```
(let ((_1 (list 10 20 30)))
   (cons (* (car _1) (second _1))
         _1))
```

(「_1」は自動生成された変数)

図3　対話履歴から生成されたプログラム例

次にユーザはメニューを使ってBUCコマンドを実行して、*Visual BUC Shell* にパス上のコマンド列と等価なプログラムを生成させる。パス上に存在しない試行錯誤過程のコマンドは生成されたプログラムから自動的に除外される。図1に示したリンクネットの最後の実行結果をターゲットとしてプログラムを生成した例を図3に示す。

2.3 プログラム化範囲の指定

対話的履歴からプログラムを生成させる場合、パスの一部分を切り出してプログラムを生成したくなることがある。ここでは、切り出し範囲を指定してプログラムを生成する方法を説明する。

テキストベースのインタフェースで切り出し範囲を指定するのは、パスが枝分れしているような場合に指定が煩雑となり好ましくない。これに対し、*Visual BUC Shell* を用いると次のように容易に切り出し範囲を指定できる。ユーザはまずリンクネット上でターゲットを指定してパスを表示させる。このパスの切り出し範囲の境界点のアイコンをマウスでクリックする。パスが枝分れしている場合には、各枝で境界点を指定することができる。この操作によりターゲットから各境界点までを部分パスとして切り出す。

ユーザはこのように*Visual BUC Shell* を用いてパスの切り出し範囲を指定した後、前述のプログラム生成の基本操作と同様にメニューを用いてBUCコマンドを実行し、切り出した部分パスに対応するプログラムを生成できる。

3.おわりに

以上述べたように、*Visual BUC Shell* は、対話履歴をリンクネットとして図示する。これによりアイコンを介した直接操作が可能となり、ユーザは対話履歴に容易にアクセスできる。この結果、BUC機能の特長を生かし、コマンドを試行錯誤的に実行する過程そのものをプログラミングとみなすボトムアップ型のプログラミングスタイルが可能となる。

[参考文献]

(1) Shu, Nan C. , 'Visual Programming', Van Nostrand Reinhold Company Inc. (1988) .

(2) 'Tempo II manual', Affinity Microsystems Ltd. (1988).

(3) 'Macintosh Utilities User's Guide - MacroMaker', pp.63-81, Apple Computer Inc. (1988).

(4) 岩崎, 湯浅, 「Bottom Up Composer ：対話履歴からのプログラム生成」, ソフトウェア・ツールシンポジウム'90, (1990), pp.55～pp.65.