VPE: 論理プログラミングにおける視覚的統合プログラミング環境

後藤 文太朗† 田中 譲††

VPEは，拡張論理型プログラミング言語Vocalogに対する視覚的統合プログラミング環境である．VPEではプログラムの作成モード，デバッグモード，そしてプログラムの利用モードという三つのモードが一つの環境に混在しうるように視覚化を行っている．どのモードにおいても視覚オブジェクトの表現形態を同一にすることで，モード間でつなぎめなく視覚オブジェクトを扱える，すなわち，モード間の差異にとらわれずに視覚オブジェクトを統一的に操作することができる．VPEでは，同一オブジェクトに対してモードに応じた種々の視覚表示を与えることができる．これにより，同一オブジェクトを三つのモードで共有し，なおかつ，それぞれのモードに応じた視覚表示でみることができる．

VPE: An Integrated Visual Programming Environment in Logic Programming

FUMITARO GOTO† and YUZURU TANAKA††

VPE is an integrated visual programming environment for an extended logic programming language Vocalog. In VPE, terms and constructs are all represented as visual objects. They can be manipulated in seamlessly integrated three different modes of system operations, i.e., the programming mode, the debugging mode, and the application mode. In VPE, each visual object may change its display representation depending on its existing mode. VPE further allows simultaneous uses of three different display modes of the same object. This allows the coexistence of three different modes of system operations for a single running program.

1. はじめに

プログラミングにおける視覚化には，プログラムの作成環境の視覚化，プログラムのデバッグ環境の視覚化，そしてプログラムの利用環境の視覚化がある．プログラムの作成，デバッグ，そしてプログラムの利用はそれぞれ独立したものではなく相互に関連している．これらの環境を個々独立に視覚化したのでは，プログラム利用のための視覚オブジェクトの動きを見ながら視覚的なデバッグを行うといった，機能連携が行えない．したがって，これらの環境を一つのシステムに統合して視覚化すべきである．その際に，プログラムの作成モード，プログラムのデバッグモード，プログラムの利用モードとして個々独立に視覚化するのではなく，これらのモードが一つの環境に混在しうる統合システムにすべきである．これにより，視覚化された個々のモードを独立に利用するのみならず，モード間をわたった機能連携を行うことが可能となる．

エンドユーザーレベルで三つのモード間の自在な機能連携を行うためには，三つのモードをつなぎめなく統合することが必要となる．すなわちモード間の差異にとらわれずに視覚オブジェクトを統一的に操作することが可能となるように，どのモードにおいても視覚オブジェクトの表現形態が同一であるようにすべきである．それぞれのモード用の視覚オブジェクトの操作も含んだ見せ方を用意しておくことで，それぞれのモードに適した視覚オブジェクトの操作および表示が可能となる．これにより，三つのモード間の自在な機能連携を行うことができ，かつ各モードに特有の視覚オブジェクトの操作および表示が可能となる．

本論文では，後藤と田中により提案されている拡張論理型プログラミング言語Vocalog[8),9)]に対する視覚的統合プログラミング環境VPE (Vocalog Program-

† 北見工業大学情報処理センター
Information Processing Center, Kitami Institute of Technology
†† 北海道大学工学部電気工学科
Department of Electrical Engineering, Faculty of Engineering, Hokkaido University

ming Environment) を提案する．語彙をデータベースへの問い合わせならびに一階述語論理で記述された知識ベースへ導入することが Tanaka により行われてきた[16)~18)]．Vocalog は，それらに基づき Prolog に語彙を導入し利用できるようにしたものである．

Prolog の視覚化システムとして，Ladret と Rueher の VLP[10)]，Pau と Olason の VPP[12)]，Eisenstadt と Brayshaw の TPM[4),6)]，Numao らによる PROEDIT 2[11)]，Dewar と Cleary の DEWLAP[5)]，Polak らによるシステム[13)]などが提案されている．また，Prolog プログラムに対して，スプレッドシートをインターフェースとして用いるものとして，Emden らによるもの[7)]，Bonsignori らによる LOGIFORM[3)]，渋谷と田中による VisiLog[15)] などがある．**表 1** にあるように，これらのシステムは前述のプログラムの作成モード，デバッグモード，利用モードのうち，一つまたは二つのモードの視覚化を行っているのに対して，VPE は三つのモードが一つの環境の中に混在しうる統合システムとなるようにつなぎめなく視覚化を行っている．これにより，プログラムの利用のための視覚オブジェクトの動きを見ながら，視覚的なデバッグおよびプログラムの視覚的な作成を行っていくことなどが可能となる．

Prolog プログラム中からウィンドウなどの視覚オブジェクトを扱えるようにしたシステムとして SICStus Prolog 上の Graphics Manager[1)] や Quintus Prolog 上の ProXT や ProXL[14)] などがある．これらはプログラミング環境の視覚化を直接的には行っていない．一階述語論理に関する視覚的 CAI システムとして，Barwise と Etchemendy による Tarski's World[2)] というシステムもある．

以下本論文では，2 章で VPE の概略を，3 章で VPE の基盤アーキテクチャを述べる．4 章で実現方法を説明する．5 章で VPE の使用例をあげる．

表 1 論理プログラミングにおける視覚化システムの比較
Table 1 Comparison of visualization systems in logic programming.

1. プログラムの作成環境の視覚化
2. プログラムのデバッグ環境の視覚化
3. プログラムの利用環境の視覚化

システム	1	2	3
VLP[10)]	○	○	×
TPM[4),6)]	×	○	×
PROEDIT2[11)]	×	○	×
VPP[12)]	○	×	×
DEWLAP[5)]	×	○	×
Polak らによるシステム[13)]	×	○	×
MPL[21)]	○	○	×
Emden らによるシステム[7)]	×	×	○
LOGIFORM[3)]	×	×	○
VisiLog[15)]	○	×	○
VPE	○	○	○

2. VPE の概略

本章では Vocalog の特徴を簡単に紹介し，そのあとで VPE の特徴に関して説明する．

2.1 Vocalog の概略[8),9)]

Vocalog では，名詞と形容詞というものを定義できる．それぞれさらに，基本名詞と派生名詞，基本形容詞と派生形容詞にわけられる．

Prolog は述語表現 $p(t_1, \cdots, t_k)$ を使って関係を表現するが，Vocalog は属性名 n_i と値 t_i の対 $n_i := t_i$ をならべたリスト述語表現 $[n_1 := t_1, \cdots n_k := t_k]$ により関係を表現する．n_i のことを基本名詞と呼ぶ．基本名詞は述語の引数位置の役割名に相当し，オブジェクトの属性名とみなすことができる．例えば，

person(taro, male, "1955.07.28", sapporo)

という述語表現に対して，次のようなリスト述語表現が対応する．

```
[name := taro, sex := male,
 birthday := "1955.07.28",
 birthplace := sapporo]
```

Vocalog では，オブジェクトの属性値間に制約を付加することで，オブジェクト間の関係をジェネリックに定義でき，それを基本形容詞と呼んでいる．基本形容詞をリスト述語中で用いることにより，オブジェクト間の関係を記述できる．a を基本形容詞，n と n′ を名詞とするとき，

[n := t, a@n′ := t′]

というリスト述語は，属性 n の属性値が t であるオブジェクトと属性 n′ の属性値が t′ であるオブジェクトとの間に a という基本形容詞があらわす関係が成り立つことを表現している．例えば，

[name := 'Taro', of_the_parent@name := 'Jiro']

というリスト述語は「名前が 'Jiro' という人は名前が 'Taro' という人の親である」ことを表している．

基本形容詞 a の一般的定義は次のようになる．

```
basic_adjective(a) : :=
  [source(n_s) := X,
   destination(n_d) := Y, p(X, Y)].
```

この基本形容詞 a を使ったリスト述語の評価規則は次のようになる．

[n := t, a@n′ := t′]
⇒[n := t, n_s := X]∧p(X, Y)∧
[n_d := Y, n′ := t′].

Vocalog では，語構成子というものを使って名詞や形容詞を組み合わせることで，複雑な概念を記述できる．語構成子を用いて，名詞を組み合わせたものを名詞句，形容詞を組み合わせたものを形容詞句と呼ぶ．例えば，親であるという関係，子供であるという関係，自分自身であるという関係を表す形容詞としてそれぞれ of_the_parent, of_the_child, of_the_self があれば，兄弟であるという関係を「親の子供であって，自分自身ではない」というものとして，

(of_the child : of_the_parent)&(-of_the_self) (1)

という形容詞句で記述できる．ここで，":" と "&" と "-" が語構成子である．語構成子の種類とそれらをもちいた句を含むリスト述語の簡単な評価規則を付録に示す．詳細に関しては，文献 8)，9) を参照されたい．

名詞句や形容詞句に対して名前付けを行うことで，派生名詞と派生形容詞を定義できる．これを語彙構築機能と呼んでいる．語彙構築機能により定義される語を派生語と呼ぶ．例えば，(1)式の形容詞句に対して名前付けを行い of_the_sibling という派生形容詞を定義できる．派生名詞と派生形容詞は，基本名詞や基本形容詞と同様にリスト述語中で用いたり，語彙構築に用いることができる．

2.2 VPE の特徴

VPE は，Tanaka らにより提案されているシンセティック・メディア・システム IntelligentPad[19),20)] を用いて開発されている．IntelligentPad では，すべての視覚オブジェクトは，機能を持った紙として表現される．この紙をパッドと呼んでいる．パッドの上には複数のパッドをはることができる．パッドの上にパッドをはることで，レイアウトを行うとともに機能の合成を行うことができる．

VPE では，プログラムの作成環境，プログラムのデバッグ環境，プログラムの利用環境という三つのモードにおいてパッドという単一の表現形態を用いて視覚化を行っている．

VPE では，Vocalog における名詞や形容詞など，プログラムの部品となるものすべてをパッドとして視覚化している．それらのパッドを視覚的に組み合わせてプログラムを構成するシンセティック・プログラミングを行うことができる．形容詞 of_the_parent, of_the_child, of_the_self と語構成子 "&", "-" それぞれを視覚化したパッドを組み合わせて得られる合成パッドの例を**図 1** に示す．図 1 において四角い枠で示されるものがすべてパッドである．図 1 の合成パッドは前節の最後に示した「兄弟であるという関係」を表す形容詞句に対応する．なお，図 1 には語構成子 ":" に対応するパッドはないが，これは形容詞に対応するパッドどうしを図のように組み合わせること自体が ":" の機能を実現しているからである．

VPE では，プログラムの部品となるパッドは各モードに応じた見せ方を持つことができる．変数 X を表すパッドの各モードにおける見せ方の例を**図 2** に示す．プログラムの作成時には変数名を表示する（図 2 (a)）．プログラムのデバッグ時には，その変数が束縛されていれば束縛された値を表示し，それ以外では変数名をそのまま表示する（図 2 (b)）．プログラムの利用モードでは，変数 X に束縛された数値をパーメータで表示したり，パーメータを操作して変数 X を適当な数値に束縛することができる（図 2 (c)）．図 2 の三つのパッドは，パッドの大きさが異なったり表示の仕方が異なるが，すべて同一のパッドである．

3. 基盤アーキテクチャ

本章では，まず VPE においてパッドとして視覚化されるものを明らかにする．次に，パッドのはりあわせにより得られる合成オブジェクトについて説明す

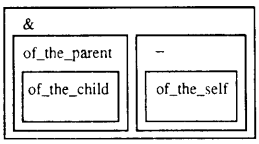


図 1 VPE における視覚的な合成例
Fig. 1 An example visual composition in VPE.

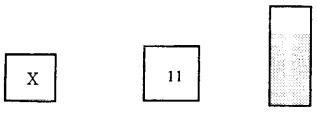


図 2 異なる表示パッドを使った同一変数の種々の見せ方
Fig. 2 Different presentations of the same variable by using different view pads.

る．そして，モードに応じたオブジェクトの見せ方について説明する．

3.1 視覚化されるオブジェクト

VPE において視覚化されるオブジェクトとそのパッドとの対応を**表2**に示す．表2に示すように，これらのパッドは VPE 基本パッドと VPE システムパッドの2種類にわけられる．VPE 基本パッドは，プログラムを構成する部品がパッドとして視覚化されたものである．VPE システムパッドは，システムの構成要素がパッドとして視覚化されたものである．ルールベースパッドは名詞や形容詞の定義と節集合を格納する機能を持つ．Vocalog パッドは Vocalog インタープリタの機能を持つ．

表 2 VPE における視覚オブジェクト
Table 2 Visual objects in VPE.

種　類	パッド名	オブジェクト
VPE 基本パッド	定数パッド	定　数
	変数パッド	変　数
	複合項パッド	複合項
	基本名詞パッド	基本名詞
	基本形容詞パッド	基本形容詞
	派生名詞パッド	派生名詞
	派生形容詞パッド	派生形容詞
	語構成子パッド	語構成子
	リスト述語パッド	リスト述語
	ヘッドパッド	節の頭部
	ボディパッド	節の本体
	節パッド	節
	基本形容詞定義パッド	基本形容詞の定義式
	派生語定義パッド	派生語の定義式
VPE システムパッド	ルールベースパッド	ルールベース
	Vocalog パッド	Vocalog インタープリタ

3.2 合成オブジェクト

VPE におけるパッドのはりあわせがどのような操作に相当するかを，主な組み合せに関して説明する．

●変数パッドへの貼付

変数パッドXの上に定数パッドcをはることで，変数 X を定数 c に単一化できる（**図3**（1））．同様に，変数パッドの上に複合項パッドや他の変数パッドをはることで，それらと単一化できる．

●複合項パッドへの貼付

複合項パッドの上に，変数パッドや定数パッド，または別の複合項パッドをはることで引数を指定できる．図3（2）では，f(c1, c2) という複合項がはりあわせにより得られている．

●名詞パッドへの貼付

基本名詞パッドと派生名詞パッドをあわせて名詞パッドと呼ぶことにする．名詞パッドの上に，定数パッドや変数パッド，または複合項パッドをはることで名詞のとる値を指定できる．図3（3）では，名詞 n のとる値として t を指定している．

●形容詞パッドへの貼付

基本形容詞パッドと派生形容詞パッドをあわせて形容詞パッドと呼ぶことにする．形容詞パッド a の上に，名詞パッド n を貼付することで a@n という句を

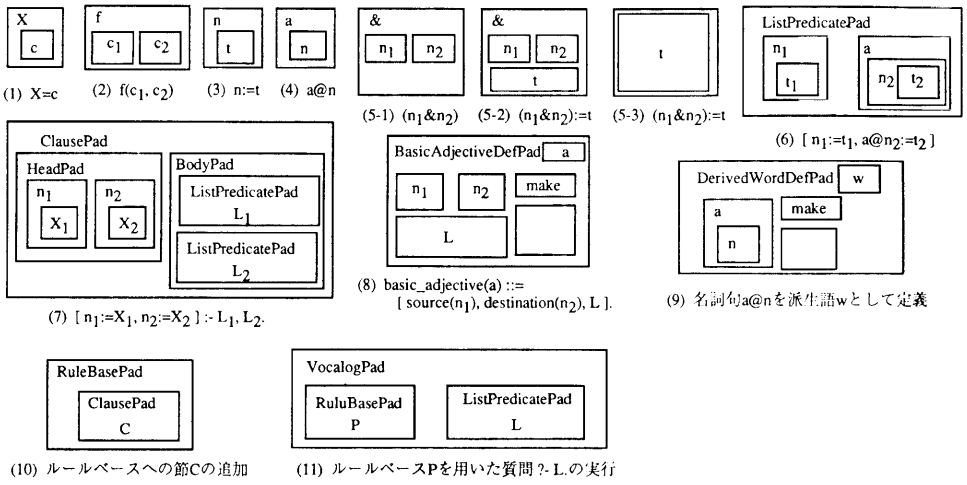


図 3 パッドの合成による合成オブジェクト
Fig. 3 Complex objects by pad composition.

表すパッドを構成できる．形容詞パッド A_1 の上に，別の形容詞パッド A_2 をはることで，合成演算子を使った形容詞句 $A_2:A_1$ を表すことができる．

●語構成子パッドへの貼付

語構成子パッドの上に名詞パッドをはることで名詞句を構成できる．さらにその語構成子パッドの上に定数パッドなどを貼付することで，その名詞句の取る値を指定できる．例えば，図3 (5-1) では，n_1&n_2 という名詞句が構成されている．図3 (5-2) では，その値として t が指定されている．また，t を表すパッドをリサイズして図3 (5-3) のようにみせることができる．形容詞句を表すパッドの構成およびその形容詞句が修飾する名詞の指定も同様に行う．

●リスト述語パッドへの貼付

リスト述語パッドの上に，名詞パッドや，名詞パッドを修飾している形容詞パッドをはることで，それらに対応する語を並べたリスト述語を構成できる．図3 (6)では，$[n_1 := t_1, a@n_2 := t_2]$ というリスト述語が構成されている．

●節パッドへの貼付

節パッドの上にヘッドパッドとボディパッドをはることで節が構成される．ヘッドパッドの上には，複数の基本名詞パッドがはられる．ボディパッドの上には，複数のリスト述語パッドがはられる．図3 (7)では，節の頭部が $[n_1 := X_1, n_2 := X_2]$ で，節の本体が L_1 と L_2 の連言である節が構成されている．

●基本形容詞定義パッドへの貼付

基本形容詞定義パッドの上に，2枚の名詞パッド n_1, n_2 とリスト述語パッド L をはることで，基本形容詞の定義を構成できる（図3 (8)）．make ボタンを押すと，make ボタンの下のパッド上に対応する基本形容詞パッドが生成される．それを drag out して利用できる．

●派生語定義パッドへの貼付

派生語定義パッドの上に，名詞句や形容詞句に対応する合成パッドをはることで，派生語の定義を構成できる（図3 (9)）．make ボタンを押すと，make ボタンの下のパッド上に対応する派生語のパッドが生成される．それを drag out して利用できる．

●ルールベースパッドへの貼付

ルールベースパッドは，節集合を格納する機能を持つ．ルールベースパッドの上に節パッドや基本形容詞定義パッドや派生語定義パッドをはると，それらに対応する定義がルールベースに追加される（図3 (10)）．ルールベースパッドの上にすでにはられているパッドをはがすと，そのパッドに対応する定義がルールベースから削除される．

●Vocalog パッドへの貼付

Vocalog パッドは質問を実行する機能を持つ．Vocalog パッドの上には，ルールベースパッドとリスト述語パッドがはられる．貼付されたルールベースパッドに格納されている節の集合が質問の実行に使われる．貼付されたリスト述語パッドに対応するリスト述語が質問として実行される（図3 (11)）．

3.3 モードに応じた見せ方

VPE には，プログラムの作成モードとデバッグモード，そしてプログラムの利用モードがある．VPE 基本パッドと VPE システムパッドは，どのモードに設定されているかというモード情報を持つ．デフォールトでは作成モードである．モード変更機能は Vocalog パッドが持つ．Vocalog パッドの上に貼付されたパッドは，Vocalog パッドで設定されたモードとなる．

VPE 基本パッドと VPE システムパッドは複数の見せ方を持つことができる．この見せ方のことを表示スタイルと呼ぶ．各モードに対して適当な表示スタイルがデフォールトで対応づけられている．モードと連動して表示スタイルが切り替わる．

例えば，変数パッドはデフォールトで次の三つの表示スタイルを持つ．

（1） 変数名を表示するスタイル
（2） 変数の束縛状況を表示するスタイル
（3） (2)の機能に加えて，変数への束縛をキーボードから与えることができるスタイル

デフォールトで(1)，(2)，(3)の表示スタイルがそれぞれ作成モード，デバッグモード，利用モードに対応づけられている．

パッドの表示スタイル変更は，**図4**に示す表示スタイル変更パッドをその上にはり，矢印のボタンを操作して表示スタイルを変更した後，表示スタイル変更

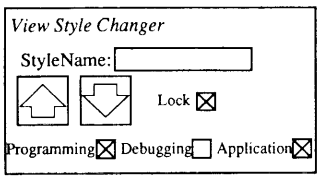


図4 表示スタイル変更パッド
Fig. 4 A view-style change pad.

パッドを剝離することで行う．モードと表示スタイルとの対応関係の変更，モードと連動した表示スタイルの変更設定解除も表示スタイル変更パッドを用いて行う．

VPE では，これらの見せ方も種々のパッドのはりあわせで実現している．デフォールトでは，各モードに対応した三つの表示スタイルが用意される．IntelligentPad システムにプリミティブとして用意されているパッドや，新たにユーザーが作ったパッドを表示スタイルとして追加できる．デフォールトの三つの表示スタイルの他に表示スタイルを追加するための特殊スタイルがある．

4. 実現手法

この章では，3章で述べた基盤アーキテクチャの実現手法を説明する．最初に，IntelligentPad の動作機構を説明する．

4.1 IntelligentPad の動作機構

IntelligentPad におけるパッドは，Smalltalk-80 の MVC 構造を単純化した構造を持つ．Mでパッドの内部状態が定義され，Vで表示の仕方が定義され，Cでマウス操作などに対する反応が定義される．Mにおけるいくつかの内部状態がスロットとして公開される．

パッド間でのデータ通信は，はりあわせ構造を介した，set，update，gimme という三つのメッセージとパッドの幾何属性を変更するジリオメトリメッセージに限られている．この様子を**図5**に示す．

パッドのはりあわせ構造において，下のパッドを親パッド，上のパッドを子パッドと呼ぶ．子パッドは自分より下のパッドのスロットのうちの一つだけに結合できる．set メッセージにより子パッドは自分が結合しているスロットに値を入れることができる．親パッドの状態が変わると子パッドに update メッセージが送られる．update メッセージが送られると，子パッドは親パッドに gimme メッセージを送り，自分が結合しているスロットの値を読み出す．set，gimme のメッセージは，親パッドが対応するスロットを持たなければ親の親に送られる．

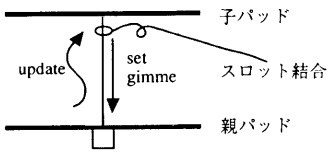


図 5 パッド間の標準インターフェース
Fig. 5 Standard message interface between pads.

パッドは共有コピーをとることができる．単一のパッドの場合，共有コピーをとると，M部が共有されて，V と C は個別に持つパッドが作られる．複数のパッドがはりあわさった合成パッドの共有コピーをとると，はりあわせ構造の一番下にあるパッドのM部が共有され，他の部分は個別に持つ合成パッドが作られる．

4.2 質問の構成と実行機構

質問を実行する際のパッドのはりあわせ構造は**図 6** のようになる．一番下に，Vocalog パッドがあり，その上に質問に相当するリスト述語パッドが貼付される．図6に，

?-[name :=taro, of_the_parent@name :=X].

という質問を実行する際のデータの流れを示す．図6における番号が以下の説明の番号に対応する．

（1） query スロットに初期化のための特殊シンボル #initialize がセットされる．

（2） Vocalog パッドから update メッセージが子パッドに送られる．子パッドは親パッドに gimme メッセージを送り，初期化フェーズであることを知る．

（3） （2）の update メッセージが親から子へと伝搬していき，Vocalog パッドの上方にあるすべてのパッドに対して，初期化フェーズであることが伝わる．

（4） はりあわせ構造の一番上にあるパッドから順順に set メッセージが親パッドに送られていき，質問に相当するリスト述語を表す文字列

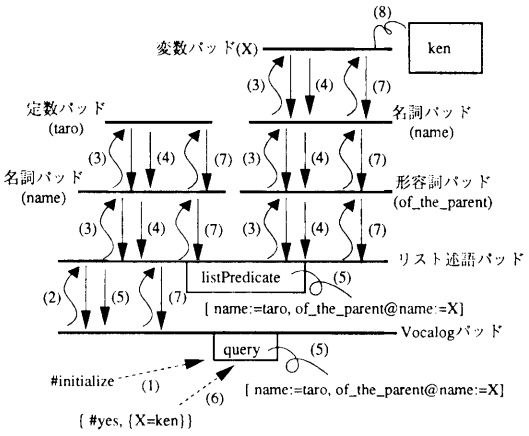


図 6 質問実行時のはりあわせ構造とデータの流れ
Fig. 6 Layout structure of pads and data flow when a query is executed.

が構成されていく．

（5） Vocalog パッド上のリスト述語パッドのスロット listPredicate に，質問に相当するリスト述語を表す文字列として，

[name := taro,
of_the_parent@name := X]

が構成され，それが Vocalog パッドの query スロットにセットされる．

（6） Vocalog パッドにおいて質問が実行される．質問の実行が成功して，特殊シンボル #yes と正解代入 {X＝ken} が Vocalog パッドの query スロットに入れられる．（質問の実行が失敗した場合は，特殊シンボル #no が Vocalog パッドの query スロットにセットされる．）

（7） Vocalog パッドから子パッドに update メッセージが送られる．（6）で得られた query スロットの内容が子パッドに伝搬する．

（8） 各パッドで表示が更新されて，変数パッドの表示が束縛された値 ken になる．

4.3 表示スタイルの切り替え機構

VPE 基本パッドと VPE システムパッドは，複数のパッドのはりあわせにより構成されている．変数パッドの例（**図 7**）を用いて表示スタイルの切り替え機構を説明する．

変数パッドは変数に関するデータ等を保持するパッド P_0 と表示スタイル切り替え機能を持つパッド P_{MF} と表示スタイル用のパッド P_1, P_2, P_3 から構成され，図 7 に示すはりあわせ構造を持つ．

パッド P_0 には，変数名Xが格納されているスロット varName，変数Xの束縛されている値 ken が格納されているスロット varValue，モードがデバッグモードであることなどのデータが格納されているスロット vocalogData がある．

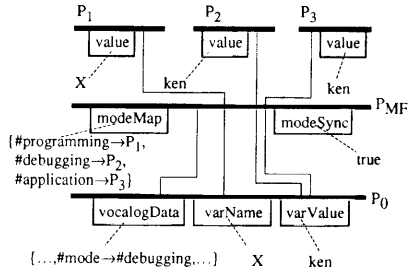


図 7 変数パッドの見せ方の切り替え機構
Fig. 7 A view-switching mechanism of a variable pad.

P_1 と P_2 はスロット value の値を表示する機能を持つ．P_1 のスロット value には，スロット結合により P_0 のスロット varName の値である変数名Xが格納される．P_2 のスロット value には，スロット結合により P_0 のスロット varValue の値 ken が格納される．これにより，P_1 と P_2 はそれぞれ変数名Xと変数Xが束縛されている値 ken を表示する．P_3 はスロット value の値を表示する機能とスロット value にキーボードから文字列をセットする機能を持つ．P_3 のスロット value には，スロット結合により P_0 のスロット varValue の値 ken が格納される．パッド P_3 を使って変数 X に対する束縛を変更できる．パッド P_3 上でキーボードから定数 taro を入力すると，スロット結合によりパッド P_0 のスロット varValue の値が taro に変わる．

パッド P_{MF} はマルチファセットパッドと呼ばれる種類のパッドである．マルチファセットパッドはその上にはられている複数のパッドのなかから一つだけを表示し，かつ自分自身の大きさを表示したパッドの大きさにあわせる．パッド P_{MF} には各モードと表示スタイル用のパッドとの対応を格納するスロット modeMap がある．図 7 では P_1, P_2, P_3 がそれぞれ作成モード，デバッグモード，利用モードと対応づけられていることを示すデータがスロット modeMap に格納されている．パッド P_{MF} はスロット modeSync の値が true のときにだけ，結合しているスロットから得られるモードと対応づけられている表示スタイル用のパッドに表示変更する．この機能によって図 7 の状態では P_2 が表示されている．

図 7 の変数パッドの共有コピーを取って，一方のパッドの P_{MF} のスロット modeSync の値を false にしてシステム全体のモード（デバッグモード）と切り離してから，利用モードの視覚表示に変更することによりデバッグモードと利用モードの両方の視覚表示をみることができる．これは変数パッドの共有コピーを取った場合，実際に共有されるのは一番下のパッド P_0 が持つ内部状態であり，その上にあるマルチファセットパッド P_{MF} が保持するデータに関しては，それぞれのパッドが個別に持つことができるからである．

5. 例

世界の都市に関するデータベースの例をあげる．図 8，図 10，図 11 がそれぞれ，プログラムの作成，デ

バッグ，利用のためのパッドである．これらのパッドを同時にみながら，プログラムの作成，デバッグを行っていくことが可能である．

データベース中には，都市の名前と緯度・経度が次のようなファクトで格納されているとする．

```
[city := tokyo,
    lattitude_deg := 35, longitude_deg := 139,
    lattitude_min := 42, longitude_min := 46,
    lattitude_ns := north, longitude_ew := east].
    ...
```

基本名詞 city は値として都市名をとる．基本名詞 lattitude_deg, lattitude_min は緯度の度と分に関する数値を値としてとる．基本名詞 lattitude_ns は北緯の場合は north を，南緯の場合は south を値としてとる．経度に関する基本名詞も同様である．

これらのファクトの作成モードにおける表示例を**図 8** に示す．図 8 において，はりあわせ構造の一番下にあるパッドは Vocalog パッドである．作成モードと対応づけられているルールベースパッドが表示されている．ルールベースパッドでは，格納されているファクトの一つを表示している．ルールベースパッド上の矢印のボタンパッドを押すことで，別のファクトを表示できる．Add と表示されているボタンパッドは，ルールベースにファクトを追加するときに使う．

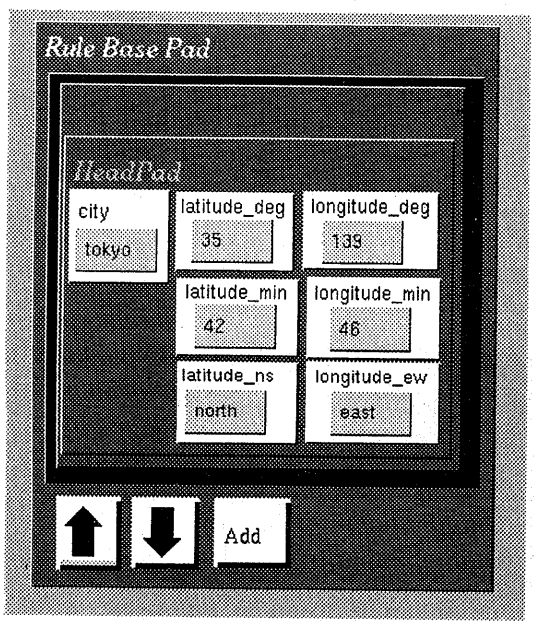


図 8 作成モードにおける表示例
Fig. 8 An example view in programming mode.

ルールベースには，さらに国名とその首都名とが基本名詞 country と capital を用いて，次のようなファクトで格納されているとする．

```
[country := japan, capital := tokyo].
    ...
```

VPE システムでは，首都の緯度情報などを得るための基本形容詞 capital's の定義：

```
basic_adjective(capital's) : :=
        [source(capital) := X,
         destination(city) := Y,
         X=Y].
```

も基本形容詞定義パッドを用いて視覚的に行える．その基本形容詞定義パッド上で生成された基本形容詞パッドを用いた例が**図 9** である．図 9 のパッドは，次のリスト述語の作成モードにおける表示例である．

```
[country := japan,
 capital's@[city := U1, lattitude_deg := V1]]
```

このリスト述語は，国 japan の首都は U1 でその緯度は V1 度であるということをあらわす．

デバッグモードにおける表示例を**図 10** に示す．図 10 に示されている三つのパッド（左上，左下，右）は同一の Vocalog パッドの共有コピーである．マルチファセットパッドを用いて表示スタイルを切り替えている．Vocalog パッドがデバッグモードにおいてデフォールト表示するのは，質問に相当するリスト述語パッド（図 10 の左上のパッド）である．図 8 と同様な基本名詞パッドが使われている．基本名詞パッドの上に変数名表示のある変数パッドが貼付されている．図 10 の左下にあるパッドは，Vocalog システムにコマンドを与えるための表示スタイルに設定されている．このパッド上の step ボタンは実行を 1 ステップ進め

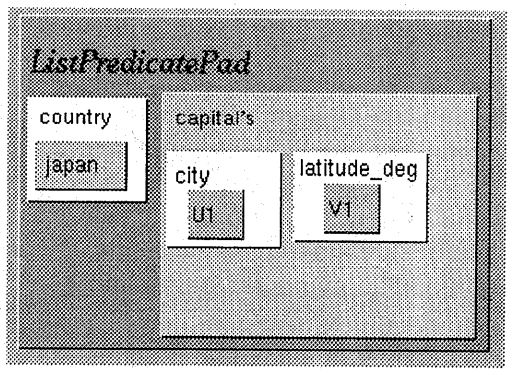


図 9 形容詞パッドの使用例
Fig. 9 An example use of an adjective pad.

るために使う．実行の結果，それで良い場合にOKボタンを，別解を求めるときにNextボタンを使う．その下の表示器は，質問が最終的に成功した場合“yes”を，失敗したときは“no”を，質問の実行中は“--”を表示する．図10の右側のパッドは，ルールベースパッドを表示するスタイルに設定されている．ルールベースパッドでは，現在選択されているファクトを表示している．ルールベースパッドの下のほうにある表示器が“?”を示しているのは，この節の選択が成功したかどうかが未定であるからである．成功すると

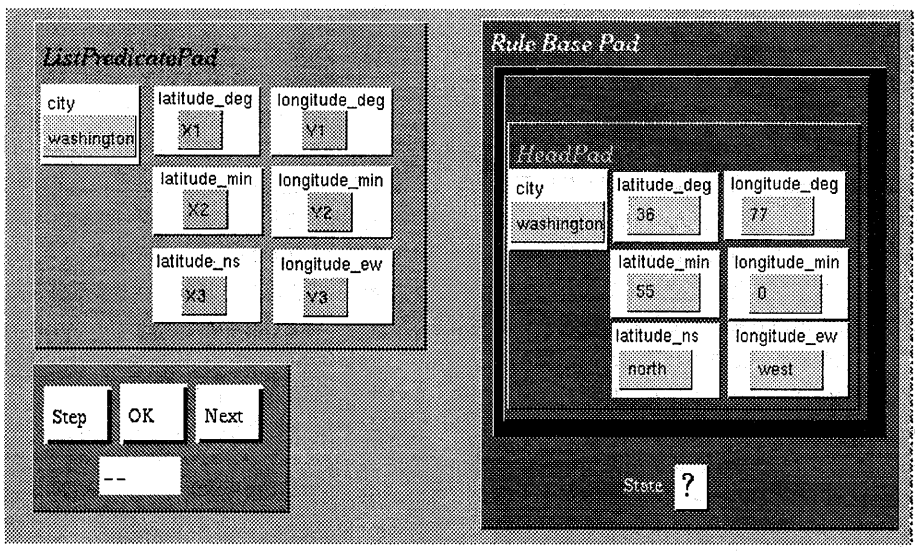


図10 デバッグモードにおける表示例
Fig. 10 An example view in debugging mode.

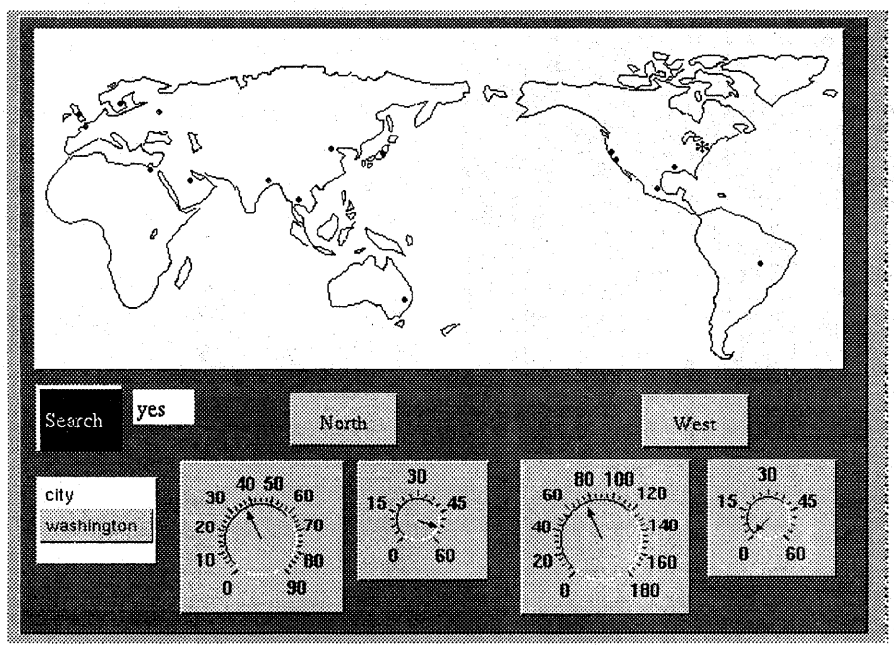


図11 利用モードにおける表示例
Fig. 11 An example view in application mode.

"○" に，失敗すると "×" に変わる．図 10 の左下のパッド上の step ボタンを押して実行が成功すると，左上のリスト述語パッド上で変数の束縛状況が示される．

図 11 のパッドでもはりあわせ構造の一番下にあるパッドは Vocalog パッドである．利用モードに対応づけられているリスト述語パッドが表示されている．このリスト述語パッドは図 10 の左上のパッドと同一のものであるが，表示スタイルが異なっている．都市名 "washington" をテキストで入力して，Search ボタンを押した後の状態が図 11 である．Search ボタンを押すことで質問が実行されている．緯度と経度がメーターを使って表示されている．地図上で "washington" の場所が "＊" 印で示されている．地図上の "＊" と "・" は同じ種類のパッドである．それらのパッドは都市名を保持しており，リスト述語パッド中の都市名に関する内部状態を参照して，それと同じ都市名のものが "＊" を表示し，他のものは "・" を表示する．Search ボタンの横の表示器はその都市が見つかれば "yes" を，見つからなければ "no" を表示する．

6. おわりに

拡張論理型プログラミング言語 Vocalog の，視覚的統合プログラミング環境 VPE を示した．VPE はシンセティック・メディア・システム IntelligentPad を用いて開発された．

VPE では，プログラムの作成モード，デバッグモード，プログラムの利用モードという三つのモードを視覚的に統合してつなぎめなく扱えるようにするために，次の二点を実現した．第一に，三つのモードすべてにおいて視覚オブジェクトをパッドという同一の表現形態で統一した．第二に，同一のオブジェクトに対して種々の見せ方（表示スタイル）を与えることができるようにした．これにより，同一オブジェクトを複数のモードで共有して，なおかつ，それぞれのモードに適した見せ方を与えることができた．

Vocalog のプログラムを構成する部品をパッドとして視覚化した．これらの部品を組み合わせることで，シンセティック・プログラミングが行えることを示した．システムを構成するものとして，節集合を格納するルールベースと質問の実行機能を持つ Vocalog インタープリタをそれぞれルールベースパッド，Vocalog パッドとして視覚化した．質問の実行に使われるルールベースの指定を，Vocalog パッドの上に指定したいルールベースパッドを貼付することで行えるようにした．質問を Vocalog インタープリタに与えることも，質問に相当するリスト述語パッドを Vocalog パッドの上に貼付することで行えるようにした．

Vocalog は Prolog に名詞や形容詞といった語彙を導入し利用できるようにしたシステムなので，VPE を Prolog の視覚的プログラミング環境として用いることも可能である．VPE では，Vocalog の名詞や形容詞を視覚オブジェクトとし，それらを組み合わせてプログラムを構成するシンセティックプログラミングと IntelligentPad のシンセティックプログラミングとの間に緊密な対応関係をとることで，従来 Prolog では不可能であったシンセティックプログラミングを論理プログラミングに導入することが可能となった．

VPE では，IntelligentPad システムの資源を使うことができるので，メディアオブジェクトとしてのパッドをより多く取り入れていくことで，視覚的なプログラミング環境をさらに充実していくことが可能である．

参考文献

1) Almgren, J., Andersson, S., Flood, L., Frisk, C., Nilsson, H. and Sundberg, J.: *SICStus Prolog Library Manual*, Swedish Institute of Computer Science (1993).
2) Barwise, J. and Etchemendy, J.: The Language of First-Order Logic, CSLI Lecture Notes Number 23, CSLI (1990).
3) Bonsignori, A., Giannotti, F., Lucchesi, L. and Turini, F.: The Logiform System, *Computers & Mathematics with Applications*, Vol. 20, No. 9/10, pp. 83-99 (1990).
4) Brayshaw, M. and Eisenstadt, M.: A Practical Graphical Tracer for Prolog, *Int. J. Man-Machine Studies*, Vol. 35, pp. 597-631 (1991).
5) Dewar, A. D. and Cleary, J. G.: Graphical Display of Complex Information within a Prolog Debugger, *Int. J. Man-Machine Studies*, Vol. 25, pp. 503-521 (1986).
6) Eisenstadt, M. and Brayshaw, M.: The Transparent Prolog Machine (TPM): An Execution Model and Graphical Debugger for Logic Programming, *J. Logic Programming*, Vol. 5, pp. 277-342 (1988).
7) van Emden, M. H., Ohki, M. and Takeuchi, A.: Spread-sheets with Incremental Queries as a User Interface for Logic Programming, *New Generation Computing*, Vol. 4, pp. 287-304 (1986).
8) Goto, F. and Tanaka, Y.: Introducing a Large

Vocabulary into Prolog, *Proc. Pacific Rim International Conference on Artificial Intelligence '90*, pp. 816-821 (1990).
9) 後藤文太朗, 田中 譲: Vocalog: 語彙を用いたジェネリックな概念記述が可能な拡張論理型プログラミング言語, 情報処理学会論文誌, Vol. 33, No. 4, pp. 512-520 (1992).
10) Ladret, D. and Rueher, M.: VLP: a Visual Logic Programming Language, *Journal of Visual Languages and Computing*, Vol. 2, pp. 163-188 (1991).
11) Numao, M., Morishita, S. and Maruyama, H.: How Should Prolog Computation Be Represented for Practical Use?, *New Generation Computing*, Vol. 8, pp. 95-112 (1990).
12) Pau, L. F. and Olason, H.: Visual Logic Programming, *Journal of Visual Languages and Computing*, Vol. 2, pp. 3-15 (1991).
13) Polak, J. and Guest, S. P.: A Graphical Representation of the Prolog Programmer's Knowledge, *Visualization in Human-Computer Interaction (7th Interdisciplinary Workshop on Informatics and Psychology, Selected Contributions)*, pp. 82-104 (1990).
14) Quintus X Window Interface, Quintus Corporation (1991).
15) 渋谷正弘, 田中 譲: スプレッド・シートを介した論理型プログラミング, 情報処理学会論文誌, Vol. 30, No. 6, pp. 709-718 (1989).
16) Tanaka, Y.: Information Space Model, *Proc. 2nd Workshop on Formal Bases for Databases*, Toulouse (1979).
17) Tanaka, Y.: Vocabulary Building for Database Queries, *Lecture Notes in Computer Science, vol. 147: RIMS Symposia on Software Science and Engineering*, pp. 215-232, Springer-Verlag (1983).
18) Tanaka, Y.: Vocabulary-Based Logic Programming, *Pcoc. InfoJapan '90*, Tokyo, Vol. 2, pp. 25-32 (1990).
19) Tanaka, Y. and Imataki, T.: IntelligentPad: A Hypermedia System allowing Functional Composition of Active Media Objects through Direct Manipulations, *Proc. of IFIP World Computer Congress '89*, San Francisco., pp. 541-546 (1989).
20) Tanaka, Y., Nagasaki, A., Akaishi, M. and Noguchi, T.: A Synthetic Media Architecture for an Object-Oriented Open Platform, *Proceedings of the IFIP 12th World Computer Congress, Madrid, Spain*, pp. 104-110 (1992).
21) Yeung, R.: MPL—A Graphical Programming Environment for Matrix Processing Based on Logic and Constraints, *1988 IEEE Workshop on Visual Languages*, pp. 137-143 (1988).

付録 語構成子を用いたリスト述語の評価規則

名詞に関する語構成子を用いたリスト述語の評価規則はそれぞれ次のようになる. ただし, n_1, n_2, n_3 は名詞, t_1, t_2, t, s は項, x, y は変数である.

- 代入 (':=')

  $[(n_1 := s) := t_1, n_2 := t_2]$

  $\Rightarrow [n_1 := t_1, n_2 := t_2] \wedge s = t_1$

- 論理演算 ('+', '&', '−')

  $[(n_1 + n_2) := t, n_3 := s]$

  $\Rightarrow [n_1 := t, n_3 := s] \vee [n_2 := t, n_3 := s]$

  $[(n_1 \ \& \ n_2) := t, n_3 := s]$

  $\Rightarrow [n_1 := t, n_3 := s] \wedge [n_2 := t, n_3 := s]$

  $[n_1 := t, (-n_2) := s]$

  $\Rightarrow [n_1 := t] \wedge \neg [n_1 := t, n_2 := s]$

- グループ・バイ演算 ('/')

  $[(n_1/n_2) := t, n_3 := s]$

  $\Rightarrow [n_2 := x, n_3 := s] \wedge$

  $t = \{y \mid [n_1 := y, n_2 := x]\}$

- 射影演算 ('≪')

  $[(n_1 \ll n_2) := t, n_3 := s]$

  $\Rightarrow [n_1 := t, n_2, n_3 := s]$

- 関数

  $[f(n_1) := t, n_2 := s]$

  $\Rightarrow [n_1 := x, n_2 := s] \wedge t = f(x)$

  (t は関数 f(x) を評価した値を取る.)

- リスト化

  $[[n_1, n_2] := t, n_3 := s]$

  $\Rightarrow [n_1 := t_1, n_2 := t_2, n_3 := s] \wedge t = [t_1, t_2]$

形容詞に関する語構成子を用いたリスト述語の評価規則はそれぞれ次のようになる. ただし, a, a_1, a_2 は形容詞, n_1, n_2 は名詞, t, s は項である.

- 逆演算 ('~')

  形容詞 a が

```
basic_adjective(a) :=
    [source(ns) := X,
     destination(nd) := Y,
     p(X, Y)].
```

と定義されているときは,

$[(a^{\sim})@n_1 := t, n_2 := s]$

$\Rightarrow [n_d := Y, n_2 := s] \wedge p(X, Y) \wedge$

$[n_1 := t, \ n_s := X]$

のように評価される. それ以外の場合は, 次のよう

に定義される．

$(a_1^{\sim})^{\sim}=a_1$

$(a_1+a_2)^{\sim}=(a_1^{\sim})+(a_2^{\sim})$

$(a_1 \ \& \ a_2)^{\sim}=(a_1^{\sim})\&(a_2^{\sim})$

$(-a_1)^{\sim}=-(a_1^{\sim})$

$(a_1 : a_2)^{\sim}=(a_2^{\sim}) : (a_1^{\sim})$

$(a_1^*)^{\sim}=(a_1^{\sim})^*$

● 論理演算 (‘+’, ‘&’, ‘−’)

$[(a_1+a_2)@n_1 := t, n_2=s]$

$\Rightarrow[a_1@n_1 := t, n_2 := s] \vee [a_2@n_1 := t, n_2 := s]$

$[(a_1 \ \& \ a_2)@n_1 := t, n_2 := s]$

$\Rightarrow[a_1@n_1 := t, n_2 := s] \wedge [a_2@n_1 := t, n_2 := s]$

$[n_1 := t, (-a)@n_2 := s]$

$\Rightarrow[n_1 := t, -(a@n_2) := s]$

● 合成演算 (‘:’)

$[(a_1 : a_2)@n_1 := t, n_2 := s]$

$\Rightarrow[a_2@(a_1@n_1) := t, n_2 := s]$

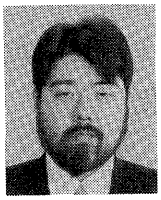
● 推移閉包演算 (‘*’)

$[(a^*)@n_1 := t, n_2 := s]$

$\Rightarrow[a@n_1 := t, n_2 := s] \vee$

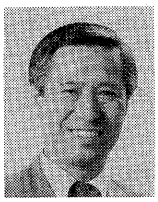
$[(a : a)@n_1 := t, n_2 := s] \vee \cdots$

(平成5年8月12日受付)
(平成6年3月17日採録)



後藤文太朗（正会員）

昭和40年生．昭和63年北海道大学工学部電気工学科卒業．平成2年同大学院工学研究科電気工学専攻修士課程修了．平成5年同大学院工学研究科電気工学専攻博士後期課程単位修得退学．現在，北見工業大学講師．論理型プログラミング，学習理論に関する研究に従事．日本ソフトウェア科学会会員．



田中　譲（正会員）

昭和25年生．昭和47年京都大学電気工学科卒業．昭和49年京都大学電子工学専攻修士課程修了．工学博士．現在，北海道大学電気工学科教授．データベースマシン，データベース理論，メディア・ベース，論理型プログラミング等の研究に従事．主たる著書，「コンピュータ・アーキテクチャ」(オーム社，共著)．IEEE，日本ソフトウェア科学会，人工知能学会各会員．