スプレッド・シートを介した論理型プログラミング†

渋谷正弘†† 田中譲†††

本論文では，論理型プログラミング言語 Prolog を可視化し，ユーザにとって使いやすいプログラミング環境を提供する VisiLog (Visual Logic) を提案している．Prolog を可視化するには，Prolog のプログラム部分，推論過程部分，実行結果の出力部分の3点をユーザにとって理解しやすく表示する必要がある．我々はスプレッド・シートに Prolog を埋め込むことにより上記の3点を表示する方法を実現した．実現された VisiLog システムは以下に示すような特徴を持つ．(1) Prolog でスプレッド・シートシステムを記述した．(2) スプレッド・シート上で Prolog プログラムが実行できる．(3) Prolog プログラムの変数とスプレッド・シートの項目の間に等値関係が定義できる．(4) スプレッド・シートのセルとセルの間に定義した計算式は代入式と解釈しないで関係式と解釈される．そのため，セルとセルにエントリした計算式の間で双方向計算が可能となる．以上の4点を特徴として持つ VisiLog を簡単な応用に適用してその有効性を考察した．

1. はじめに

論理型プログラミング言語 Prolog は 1973 年にコルメラウアにより開発された．Prolog は，一階述語論理式を制限したホーン節からなるホーン集合によって，事物と事物間の関係を記述する．Prolog のプログラムはユニフィケーションと呼ばれるホーン節とホーン節のパターンマッチングにより実行される．prolog は，言語仕様も単純でプログラムしやすく，再帰プログラミングが利用できプログラムをコンパクトにできる．また，機能を追加するなどのプログラムの変更も容易である．

Prolog を用いた実際のプログラミングでは，内部データベースと呼ばれる部分にルールやファクトをあらかじめ登録しておき，ユーザがシステムに対して質問を行う．システムは内部データベースを用いて推論を行い，解をユーザに返す．登録してあるルールやファクトを更新，表示するにはエディタを用いるか，基本述語を実行する必要がある．また質問に対する解答は，基本述語を用いて保存が可能である．

Prolog は宣言的記述力や論理性において優れている反面，以下に示すヒューマン・インタフェース機能が不備であった．

▽ルールやファクトを別々に編集すること．

▽実行結果を保存すること．

ルールやファクトを別々に編集できるようになるとプログラムの一部を変更したり，デバックを行うときに便利である．また，実行結果の保存ができると，この値を用いて新たな推論が可能となる．著者らは，Prolog のヒューマン・インタフェースを改善することを目的として VisiLog (Visual Logic) と名付けた論理型プログラミング環境を開発した．本論文では，VisiLog の機能仕様とその実現方法を述べる．本論文では，VisiLog のプロトタイプを作成し，プロトタイプ上で実現した機能について応用例を示すことで，VisiLog の有効性を明らかにする．

現在の Prolog のヒューマン・インタフェースを改善するには，プログラムの内容，実行の過程，実行結果の各々をユーザが理解しやすいように可視化する必要がある．具体的な可視化の方法として本研究では，スプレッド・シート型言語に Prolog を埋め込む方法を採用した．スプレッド・シート型言語は，セルと呼ばれる表のマス目に数値，計算式を書き込むことにより作表，計算など行う言語である．スプレッド・シートは登録したデータを加工するための豊富な関数，グラフ作成機能，マクロと呼ばれるシート上の操作手順を扱う言語を持っているため，事務処理業務における応用を中心に普及してきた言語である．これまでに "VisiCalc", "MultiPlan", "Lotus 1-2-3", "EXCEL" などのスプレッド・シート型言語が発表され，多数のユーザに使われてきた[1)-3)]．

Prolog とスプレッド・シートの各々の主な特徴を**表1**に示す．Prolog は宣言的なプログラミングやユニフィケーションによる計算などが長所であり，反面，表示や更新に関するユーザ・インタフェースの不備が短所となっている．一方，スプレッド・シートは表示，編集，自動再計算などの取扱の容易さが長所と

† Logic Programming through a Spreadsheet by MASAHIRO SHIBUYA (Department of Industrial Engineering, Faculty of Engineering, Hokkaido Institute of Technology) and YUZURU TANAKA (Department of Electrical Engineering, Faculty of Engineering, Hokkaido University).
†† 北海道工業大学工学部経営工学科
††† 北海道大学工学部電気工学科

表 1 Prolog とスプレッド・シートの特徴
Table 1 Merits and demerits of Prolog and spreadsheet.

	Prolog	スプレッド・シート
長所	• 宣言的なプログラミング • ユニフィケーションによる計算 • 推論機能 • バックトラック機能(非決定性)	• 表形式となっているので登録したデータがみやすい. • データの登録, 編集などの作業がしやすい. • 自動再計算機能
短所	• 推論の結果の保存と再表示が不便である. • ユーザがシステムに登録した述語を見つけ出しにくい.	• 算術演算以外の記号の処理が行えない. • セルとセルにエントリした計算式の間に方向性がある.

いえる. 反面, セルにエントリできる式の種類が少ないことや, 評価が方向性を持つことなどが短所となっている. Prolog とスプレッド・シートを結合することにより双方の短所を相手の長所で補うことが可能であると考えられる. Prolog の質問文や処理過程や実行結果などの保存, 表示, 更新などといった作業には, スプレッド・シートが適している. スプレッド・シートには, セルの値とセルにエントリした計算式の間に評価の方向性がある. セルの値は, セルへエントリした計算式の評価結果を代入したものである. そのためにセルの値を変更して, このセルにエントリされている計算式の中で参照している値を自動的に変更することはできない. 後述するが, この評価の方向性は Prolog のユニフィケーション機能を用いて, 両者を結び付けることにより解決することができる. この性質を利用するためには Prolog でスプレッド・シートを記述する必要がある.

また, 現在のスプレッド・シートシステムでは記号処理機能を持っていない. Prolog は記号処理機能を持っているので, セル変更を Prolog プログラムの変数として用いることができるように, スプレッド・シートと Prolog プログラムを結合できれば, この問題も解決できる. このような考察により, スプレッド・シートと Prolog の融合は意義のある試みであると考える. このような観点から開発されたシステムが VisiLog である[4)~6)]. 論理型言語とスプレッド・シートを融合したシステムは, 著者らのほかに Emden らにより提案されている[7),8)]. 彼らは, Prolog を使いやすくするために, Prolog の言語処理系を変更し, スプレッド・シート型のビジュアル・インタフェースを付加している. 一方, 我々のシステムは, Prolog の言語処理系を変更することはせずに, Prolog とスプレッド・シートの両方に幾つかの機能を追加している. その機能とは, スプレッド・シート上で "双方向計算" を可能にしたり, スプレッド・シートと Prolog 処理系を表す Prolog ウインドウの間でお互いのアクセスを可能にするなどである. これらの機能は Emden らのシステムでは, 実現されていない. また, 高級言語を用いてスプレッド・シートを記述した例として, Smalltalk-80 により記述された Smalltalk SpreadSheet がある[9)]. その大きな特徴は, セルの値に Smalltalk-80 で定義できるオブジェクトを持つことができる. そのため, セルの値として文字のほかに絵や画像やグラフなども保存できる. また, Smalltalk SpreadSheet では, Smalltalk-80 の持つ高度なインタフェースが継承されているのも特徴である. しかし, VisiLog で提案する "双方向計算" や "記号処理計算" は行えない.

VisiLog の実現法に関しては, 以下の 5 点について詳述する.

(1) 論理型言語によるスプレッド・シートの記述
(2) スプレッド・シート上での Prolog プログラムの実行
(3) Prolog プログラムの変数とスプレッド・シートの項目の間の等値関係の定義
(4) 計算評価の双方向性の実現
(5) スプレッド・シート上での自動再計算機能の実現

(4)はセル a に計算式 $b+c$ が入力されたとする. この時, 従来のスプレッド・シートで行われているのは, $a \leftarrow b+c$ という代入操作である. このため, b と a の値から c を計算するとか, b と c の値から a を計算することはできない. VisiLog では, これを $a=b+c$ という等式と解釈し, a, b, c の内の任意の 2 つの変数の値からほかの 1 つの変数の値が計算できる.

上記 5 点を実現することにより, Prolog のプログラム, 質問文, 処理過程, 結果の表示と操作を可視化することができる. スプレッド・シートの自動再計算機能を実現することにより, データの変更に伴って, Prolog の質問文の再計算を行うことができる. 試作 VisiLog は, DEC10-Prolog とシンタックスが似ている Prolog-KABA を使用し NEC PC-9801F 上に作成した[10)~12)].

以下, 本論文では, VisiLog の設計仕様 (2 章), 論理型言語によるスプレッド・シートの記述 (3 章),

スプレッド・シート上での記号処理計算（4章），双方向計算機能（5章），論理シミュレーションへの応用（6章）について述べる．

2. VisiLog の設計仕様

VisiLog は2つのモードにより構成される．2つのモードとは，基本モードとしてのスプレッド・シート，サブモードとしての Prolog や QBE である．ユーザは，必要に応じモードを切り替えて利用する．**図1**に各モードの関係を示す．モードの切り替えは，mode 機能キーにより行う．各モードでは，以下の事柄が行える．スプレッド・シートモードでは，**表2**に示す値や式の入力や編集，式の評価，ワークシート上のデータの入出力を行う．Prolog モードでは，Prolog プログラムの入出力，編集，実行を行う．QBE モードでは，データベース検索，更新，挿入，削除を行う．

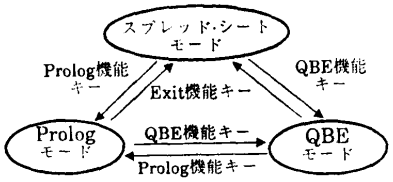


図 1 各モード間の関係
Fig. 1 Relationship among the three modes.

表 2 VisiLog の仕様
Table 2 Specification of VisiLog.

項目	種　類	内　容
値	数 値	整数，実数，指数形式
	文字列	任意の文字列
	論理値	true, false
	日 付	月/日/年
	その他	金額，etc.
式	数 式	+, −, *, /
	関 数	sum, ave, etc.
	論理式	and, or, not, xor
	Prolog	?-質問文
	QBE	_変数名
	条件式	If 条件式 then (値) [else 式(値)]
	その他	C言語とリンクする関数，etc.
機能	Move	表示画面の移動
	Form	式の入力
	Calc	計算の指示
	Float	セルの浮動指定
	View	式の画面表示
	Disp	式が参照しているセルのハイライト
	Load/Save	データの取り込み，保存
	Exit, Prolog, QBE	モード切り替え
	その他	OS の呼び出し，etc.

表 3 VisiLog で利用可能な変数や述語
Table 3 The use of variable and predicate in a VisiLog system.

種　類	内　容
セル変数	セルの値を参照するための変数
広域変数	広域的に利用可能な値を使用するための変数
ファクト	事実を記述した文字列
セル述語	セルの値の更新を行う述語
システム述語	システムで用意されている述語
ユーザ述語	ユーザが定義した述語

スプレッド・シートモードでは，VisiLog の持つ機能を利用するために表2に示す機能が用意されている．例えば，式が参照しているセルをハイライト表示する “Disp 機能” や，式中で参照されているセルの値を一時的に浮動状態にし評価する “Float 機能” などである．ユーザは，ワークシート上に値や式をエントリし，目的にあった機能キーを用いて仕事を行う．また，ユーザが Prolog や QBE のサブモードで仕事を行うためには，mode 機能キーを入力し，モードを変更する必要がある．サブモードが起動されると，ワークシートの右半分に Prolog や QBE のサブモードを利用するためのウインドウが開かれる．仕事を終えたユーザは，Exit 機能キーにより基本モードに戻ることができる．VisiLog では，Prolog とスプレッド・シートの項目の間で利用できる変数や述語が用意されている（**表 3**）．これら変数や述語は，スプレッド・シートや Prolog ウインドウ上でスプレッド・シート上のデータを利用する Prolog プログラムを実行する際に用いる．以下の章では，試作 VisiLog を簡単な応用に適用して，その有効性を考察する．

3. 論理型言語によるスプレッド・シートの記述

本システムは，スプレッド・シートに Prolog を埋め込むことにより，Prolog にスプレッド・シート型のプログラム環境を与えている．開発したシステムでは，スプレッド・シート自体も Prolog で記述されている．スプレッド・シート上に定義した値や計算式は各々 Prolog のファクトとして**図2**のように別々に登録される．

セルにエントリした値や計算式をシステムに登録する述語を**表4**に示す．

“add_value (*R*, *C*, Value)”, “remove_value (*R*, *C*, Value)” は *R* 行 *C* 列のセルの値 “Value” をシステ

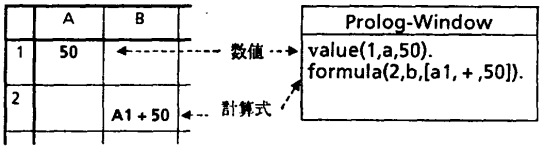


図 2 セル値の内部表現
Fig. 2 Internal representations of cell values.

表 4 セル値の値を更新するための述語
Table 4 Predicates to update cell values.

```
add_value(R, C, Value):-
  remove_value(R, C, _), assert(value(R, C, _)).
remove_value(R, C, _):-retract(value(R, C, _)) ; !.
add_formula(R, C, Formula):-
  remove_formula(R, C, _), assert(formula(R, C,
    Formula)).
remove_formula(R, C, _):-retract(formula(R, C, _)) ; !.
```

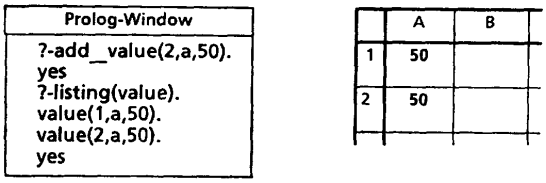


図 3 Prologプログラムの実行によるセルの値の更新
Fig. 3 An update of a cell value by evaluating a Prolog program.

ム上に追加，削除する述語である．たとえば，セル“A 2”に値 50 をエントリすると，VisiLog 内部では，“?-add_value (2, a, 50).”が起動され，その結果“value (2, a, 50).”がシステムに登録される（**図 3**）．同様に，セルの計算式は“add_formula (*R*, *C*, Formula)”によって，システム上に登録される．この時，セルにエントリされた計算式“Formula”は，四則演算式か，組み込み関数か，Prolog プログラムで定義される任意の項などの形態を採ることが可能である．この計算式“Formula”の評価は，計算時に VisiLog のインタプリタにより，どの種類のデータであるか判断され，評価される．このため VisiLog 上では，ユーザは計算式の種類を指定する必要がない．

4. スプレッド・シート上での記号処理計算

Prolog のプログラミングは，ファクトやファクトを利用するルールを定義するとともに，1つの質問文を与えることにより行われる．与えられたルールやファクトを用いて推論を行うことにより質問に対する回答を得ることがプログラムの実行を意味する．短いプログラムを作成するときは，直接 Prolog 処理系にルールやファクトを登録しながら行う．規模が大きなプログラムを作成するときは，エディタを利用する．従来の Prolog 処理系では，ルールとファクトをエディタ中でいっしょに記述したり，編集したりしている．ルールやファクトの一部を変更したい時，一度定義したルールを別のファクトで実行したい場合は，別々に扱ったほうが便利がよい．

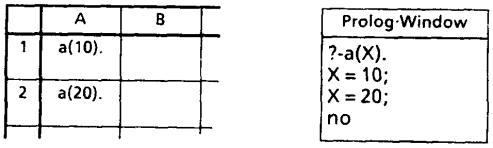


図 4 Prolog ウインドウ上での質問文の実行 (1)
Fig. 4 A query evaluation in a Prolog window (1).

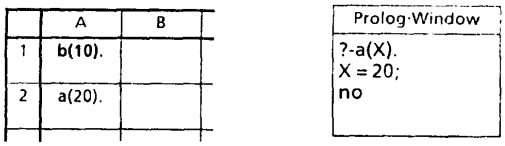


図 5 Prolog ウインドウ上での質問文の実行 (2)
Fig. 5 A query evaluation in a Prolog window (2).

VisiLog では，ルールやファクトを別々に扱えるようにするために以下の事柄を可能にした．

（1）ファクトをスプレッド・シートのセルの値として利用できるようにした．

（2）従来の Prolog 処理系と同じ働きをする Prolog ウインドウを用意した．

（3）質問文をスプレッド・シートの計算式として扱えるようにした．

（4）スプレッド・シートのセルの値を Prolog プログラムで参照できるようにした．

（1）と（2）について例を用いて説明する．**図 4** は，スプレッド・シートのセルに値としてファクトを登録し，Prolog ウインドウ上で質問文を実行し，その評価結果を表示している．スプレッド・シートのセル“A1”の値を変更し，再び同じ質問を行うと**図 5** のようになる．スプレッド・シートに登録したファクトは，セルの値となるほかに，Prolog ウインドウにてファクトをアサートした場合と同じ働きをする．このため，スプレッド・シートを Prolog のファクトのエディタとして利用できる．

VisiLog では，セルに計算式としてエントリした質問文は，以下の 4 通りに該当するときに評価される．

（a）変数を含まないもの…たとえば，“?-book.”

（b）一時変数を含むもの…たとえば，“?-book(X).”

（c）任意のセルの値を利用しているもの…たとえ

	A	B
1	abc	
2	2.5 X1	

Prolog-Window

```
?-length('A1',Y).
Y = 3
yes
?-round(X1,Y).
Y = 3
yes
```

図 6 Prolog ウインドウ上での質問文の実行 (3)
Fig. 6 A query evaluation in a Prolog window (3).

	A	B	C	D	E
1	名前	年	月	日	
2	N.Piquet	1952	8	17	
3	A.Prost	1955	2	24	
4	A.Senna	1960	3	21	

図 7 生年月日の表
Fig. 7 A birthday table.

Prolog-Window

```
zodic([H:T],Star):-zodic1(H,T,Star).
zodic1(M,D,S):- M1 is M*100, N is M1 + D, call-Star(N,S),!.
call-star(N,S) :- star(S,Min,Max), N = >Min,N = <Max.
star(牡牛座,321,420).
satr(牡羊座,421,521).
      .
      .
write-zodic:-put-data(clm,[C,D]),get-data(A),
      zodic(A,Star),write-value(Star),fail.
```

図 8 星座と生年月日を関係付ける述語 "zodic" の定義
Fig. 8 A definition of a predicate 'zodic' that relates zodiac symbols to birthdays.

ば, "?-book ('A1')."

(d) 計算式中の変数を利用しているもの…たとえば, "?-book ('X')."

このうち(a), (b)は, 従来の Prolog の質問文の形式で, (c), (d)は VisiLog により新たに追加した方法である. (c)は, シート上のデータを Prolog プログラム中で利用することを目的として追加し, (d)は, 任意の算術式中に定義した変数や広域変数などの値を Prolog プログラム中で利用するために追加した. (c), (d)を使用してシミュレーションを行った例を5章で示す.

スプレッド・シート上に, **図 6** に示すようにセルの上段にファクト, 下段に計算式がエントリされている. VisiLog には, スプレッド・シート中のセルの上段に値を, 下段に計算式を同時に示す View 機能がある. Prolog の質問文中でセルの値を利用するには, 利用したいセルの名前を記述すればよい. ただし, このセルの名前はシングルクオート (') でくくらなければならない. 図6の Prolog ウインドウ中の上側の質問文 "?-length ('A1', Y)" は, セル "A1" にエントリしている文字列の長さを求めている. この質問文を評価すると "Y=3" が得られる. これは, セル "A1" の値が3文字であることを示している. 後述するが, VisiLog では広域変数が用意されている. この変数を利用するには, 質問文中でこの変数を用いればよい. 図6のスプレッド・シートのセル "A2" には計算式として広域変数 "X1", セルの値として "2.5" がエントリされている. そのため, "X1" の値は, "2.5" となる. この時, Prolog ウインドウで質問文 "?-round (X1, Y)" を評価すると "Y=3" となる. 述語 "?-round (X, Y)" は, 実数 "X" を四捨五入した値 "Y" を求める. VisiLog のプロトタイプでは, (c), (d)に該当する質問文を実行するときは, セルの名前や広域変数をそれらが示す値に置き換えてから評価するようインプリメントしている. たとえば, "length ('A1', Y)" は "length (abc, Y)" として評価する.

(3), (4)について例を用いて説明する. **図 7** に示すようなスプレッド・シートがあり, A列に名前, B, C, D列にその人の生年月日を示す値がそれぞれ, 年, 月, 日の順で格納されている. この生年月日のデータから, 各自の生まれた星座を求めE列に格納する例を示す. この処理を実現するために, Prolog ウインドウでその人の生年月日の星座を求める述語 "zodiac" を定義し, この述語を利用して星座をセルに書き出す述語 "write_zodiac" を定義する. これらの述語を**図 8** に示す.

星座をセルに書き出す述語 "write_zodiac" の意味は左から次のようになる. "put_data (row, [clm])" はこの述語が定義された位置から, 行ごとに "[clm]" により指定された列の値をシステムに格納する. "get_data(A)" は "put_data (row, [clm])" により格納されたデータを行ごとに得る. この時のデータはリスト形式で表されており, このデータを利用し星座を求めるのが述語 "zodiac" である. 述語 "zodiac" で得られた星座 S をセルに代入するのが "write_value" である. この時, 値が代入されるセルは, "get_data" により値が取り込まれた行と "write_value" が定義されている列の交点となるセルである. この計算をすべての行について行うため "fail" を付け加える. この述語 "write_zodiac" をセル 'E2' にエントリし, 実行すると**図 9** のようになる.

以上のように, VisiLog ではスプレッド・シートの値を Prolog プログラム中で参照し, 変更したり, 追

	A	B	C	D	E
1	名前	年	月	日	星座
2	N.Piquet	1952	8	17	獅子座 write-zodic
3	A.Prost	1955	2	24	魚座
4	A.Senna	1960	3	21	牡牛座

図 9 述語 "write-zodic" の実行と，それによる生年月日の表の更新
Fig. 9 Evaluations of the predicate 'write-zodic' and the resultant updates of the birthday table.

加したりすることが行える．従来のスプレッド・シートでは，データの加工などはマクロ機能を用いたマクロプログラムにより行っていた．VisiLog では，マクロプログラムの代わりに Prolog プログラムを用いて，スプレッド・シート上のデータの加工を行える．また，Prolog の質問文を，スプレッド・シートの計算式として評価できる．そのため，従来のスプレッド・シートではできなかった記号を用いた計算が可能になった．

5. 双方向計算機能

従来のスプレッド・シートに，セル "A1" に 10，セル "B1" に 50 という値がエントリしている（**図 10** (a)）．このスプレッド・シートに，セル "A1" に計算式 "B1+60" をエントリし，評価するとセル "A1" の値は 110 に置き換えられる（図 10 (b)）．従来のスプレッド・シートでは，セルに登録されている計算式は計算結果のセルへの代入となる．このため，図 10 のセル "A1" を "10" に変更し，計算式 "B1+60" の値が "10" になるように，自動的にセル "B1" の値を "−50" に変更することができない．すなわち，あるセルに登録されている計算式の計算結果の値を変更に応じて，計算式中に現れるセルの値を自動的に変更する "双方向計算" ができない．従来のスプレッド・シートの不備な点として，以下のことが言える．

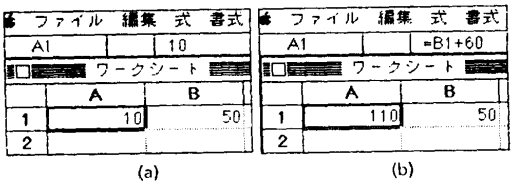


図 10 一般的なスプレッド・シートの使用例
Fig. 10 An example use of ordinary spreadsheet.

```
Prolog-Window
?-listing(add).
add(X,Y,A):- var(X),!,X is A - Y.
add(X,Y,A):- var(Y),!,Y is A - X.
add(X,Y,A):- var(A),!,A is X + Y.
```

図 11 述語 add の定義
Fig. 11 A difinition of the predicate 'add'.

セルとセルにエントリした計算式の間に方向性がある．

VisiLog のプロトタイプでは，双方向計算を実現するために，Prolog の特徴であるユニフィケーション機能を利用した．ユニフィケーション機能は，述語の引数部分を入力や出力と区別しなくても使える性質を持つ．たとえば，"$X+Y=A$" という算術式中のうちどれか 2 変数に数値を入力し，残りの変数の値を求めるとする．この問題を解くプログラムを**図 11** に示す．このプログラムは，上から X, Y, A の順に変数の値を求める．たとえば，$X=10$ で $A=5$ のとき，Y の値を求めるならば，"?-add (10, Y, 5)." という質問を行えばよい．この質問文を評価すると，2 行目の式とユニフィケーションして "$Y=-5$" を得る．このように，述語中の引数の使い分けにより双方向計算を実現した．この双方向計算を使えば，以下のことが可能となる．

(1) 計算式中で参照している値が未知のセルの値を求めること．

(2) 計算式中に存在する値が未知の広域変数の値を求めること．

(1)について例を用いて説明する．**図 12** のように某会社の 1 年間で販売された製品についての売上高がスプレッド・シート上にエントリされている(a)．図中のスプレッド・シートの下側に，現在カーソルにより選択されているセルの名前を示す Cell や，そのセルの値や式を示す Value, Formula が示されている．図 12(b) には，すべての製品の売上高の平均を求めるために組み込み関数 "ave (B2: C3)" をセル "B4)" にエントリし，評価した結果 "15" が表示されている．計算式 "ave( )" の "B2: C3" はセル "B2" とセル "C3" を対角とした領域を示す．この状態でセル "B2" の値を削除して，再びセル "D4" を評価すると，セル "B2" に値 "10" がエントリされる．VisiLog では，計算式中に現れる値の入力されていないセルの値を求めることができる．また，計算式が参照しているセルの値を一時的に浮動にして計算する

	A	B	C	D
1	製品	国内売上高	海外売上高	平均
2	ラジオ	10	20	
3	テレビ	20	10	
4				

Cell: D4
Value:
Formula: AVE(B2:C3)

(a) 実行前

	A	B	C	D
1	製品	国内売上高	海外売上高	平均
2	ラジオ	10	20	
3	テレビ	20	10	
4				15

Cell: D4
Value:15
Formula: AVE(B2:C3)

(b) 実行後

図 12 VisiLog における自動計算機能
Fig. 12 Automatic computation in a VisiLog system.

	A	B	C	D
1	製品	国内売上高	海外販売高	
2	ラジオ	10	20	
3	テレビ	20	@10	
4				25

Cell: D4
Value: 25
Formula: AVE(B2:C3)

(a) 浮動セルの再評価前

	A	B	C	D
1	製品	国内販売高	海外販売高	平均
2	ラジオ	10	20	
3	テレビ	20	50	
4				25

Cell: D4
Value: 25
Formula: AVE(B2:C3)

(b) 浮動セルの再評価後

図 13 浮動セルの指定の例
Fig. 13 An example selection of a floating cell.

	A	B	C	D
1	製品	国内売上高	海外売上高	平均
2	ラジオ	10	20	
3	テレビ	20	50	
4		90		25

Cell: B4
Value: 90
Formula: (B2 + B3)*X

(a) 実行前

	A	B	C	D
1	製品	国内売上高	海外売上高	平均
2	ラジオ	10	20	
3	テレビ	20	50	
4		90		25

Cell: B4
Value: X = 2
Formula: (B2 + B3)*X

(b) 実行後

図 14 広域変数の使用例
Fig. 14 An example use of a global variable.

こともできる．そのためには，ユーザが値を一時的に浮動にしたいセルへカーソルを持っていき Float 機能キーを入力し浮動の指定を行う必要がある．たとえば，セル“C3”を浮動セルに指定する．浮動指定を行ったセルには“@”がセルの値の前に付加される（図13(a)）．この状態でセル“D4”の値を25に変更し，計算式を再評価するとセル“C3”に値50が表示される．（図13(b)）．この状態で表示された値をセルに対して代入するか否かはユーザが決定する．

(2)について例を用いて説明する．ユーザは計算式中において任意な名前を持つ広域変数を定義し使用できる．この変数は，値が一度代入されると次の値を代入されるまで値を保持する．たとえば，テレビとラジオの国内売上高の合計が 90 になるためには，売り上げを何倍に伸ばせばよいかを求め，この値を広域変数“X”に代入する例を示す．セル“B4”に値“90”をエントリし，広域変数“X”を含む計算式“(B2+ B3)*X”をエントリする（図14(a)）．このセルの計算式を評価すると変数 X に値3が代入される（図14(b)）．

以上，VisiLog では算術計算を行うとき，計算式中に値が入力されていないセルを参照したり，参照するセルの値を一時的に浮動としたり，値を持たない変数を使用することが可能である．VisiLog のプロトタイプでは，計算式中に出現する値の不定なセルや広域変数は1つのみ許すようにインプリメントした．これは，値が不定なセルや変数が計算式中に複数個存在すると，値を一意に決定することができないためである．また，従来のスプレッド・シートで用意されている幾つかの関数“sum”，“ave”などを Prolog で定義し，VisiLog に組み込んだ．これらの関数を用いた双方向計算が可能である．

6. 論理シミュレーションへの応用

従来のスプレッド・シートでは，任意のセル上に登録した計算式が参照するセルの値が変更されたならば計算式を評価し直す，自動再計算機能を持っている．VisiLog のプロトタイプでは，個々のセルは，値の参照を受けているセルの名前を示す“参照リスト”を持っている．自動再計算機能は，セルの値が変化すると“参照リスト”に載っているセルの計算式を再計算することで実現した．この方法だと，複数の計算式がお互いに参照し合い循環してしまう場合がある．そのため，循環を防ぐために自動再計算機能を制限する必要が生じた．VisiLog では，自動再計算を禁止する関数“no-calculate (X)”を導入した．引き数“X”には，自動再計算を禁止するセルのリストを入れる．

VisiLog では，計算式として Prolog の質問文を扱えるので，質問文の再評価を簡単に行える．ただし，再評価を行うためには質問文中にセルや広域変数を示す引数を含む必要がある．具体例として，ハードウェア・シミュレーションをスプレッド・シートを用いて視覚的に行った例を示す．ボード上の入出力端子の配置に対応させ，スプレッド・シート上のセルに入出力

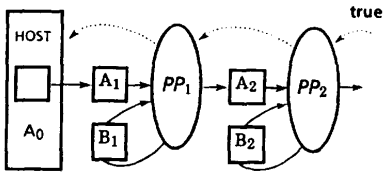


図 15 "palindrome" の回路図
Fig. 15 A diagrammatic description of a 'palindrome' circuit.

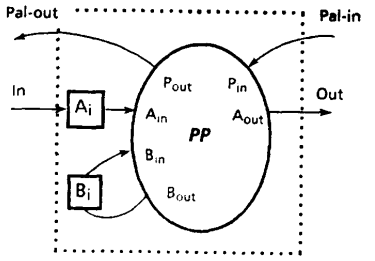
端子を割り当てる．さらに，セルは対応する端子の値を保持する．それらセルの値を表示することによりシミュレーションが視覚的に行われる．

ここでは，回文 (palindrome) を受理するマシンの例を示す[13]．回文とは，右から読んでも左から読んでも同じになる文字列のことである．このマシンを実現する処理回路を**図 15** に示す．この回路は，ホストから入力された文字列が回文であるかどうか判断し，真偽値をホストに返すように設計されている．この回路は，2種類のレジスタ A, B と組み合わせ回路 PP により構成されている．図 15 の上側にある波線の矢印は，矢印の始点にある組み合わせ回路が発する真偽値の流れを示している．ただし，右側に組み合わせ回路がない場合は，真偽値 "true" が送られる．実線の矢印は，入力された文字が移動する方向を示している．

次に，この回路の実行手順を示す．まず，HOST に入力された文字は順にレジスタ A0 を経由して右側にある A1, A2 レジスタに移動していく．ただし，Aレジスタ間に組み合わせ回路 PP が存在するときは移動の仕方が異なる．たとえば，レジスタ A1 から A2 へ入力された文字が移動する場合，この文字は B1 レジスタに値が何も代入されていないときは B1 レジスタへ移動する．B1 レジスタにすでに値が代入されているときは，この文字は A2 レジスタへ移動する．一度Bレジスタに入った文字は，移動することはできない．

この入力された文字が回文であるには，HOST に送られてきた真偽値が "true" である必要がある．真偽値が "true" になるには，組み合わせ回路 PP の左側にあるBレジスタに値が入力されていないか，AレジスタとBレジスタの値が同じで，右側の組み合わせ回路から送られてきた真偽値が "true" である必要がある．入力された文字は，右側にある組み合わせ回路から順に評価され，HOST に送られてきた真偽値が "true" のとき回文となる．

この組み合わせ回路 PP の仕様を**図 16** に示す．この



$$A_{out} \triangleq \begin{cases} nil & B_{in} = nil \\ A_{in} & B_{in} \neq nil \end{cases}$$

$$B_{out} \triangleq \begin{cases} A_{in} & A_{in} = nil \vee B_{in} = nil \\ B_{in} & A_{in} \neq nil \wedge B_{in} \neq nil \end{cases}$$

$$P_{out} \triangleq (B_{in} = nil) \vee (P_{in} \wedge (A_{in} = B_{in}))$$

図 16 組み合わせ回路 PP の仕様
Fig. 16 A specification of a combinational circuit PP.

```
palindrome(In,Out,Pal-in,Pal-out,R-Ai,R-Bi):-
   set-register-a(In,A-in),read-register-b(B-in),
   b-out(B-out,A-in,B-in),
   set-register-b(B-out,_),a-out(Out,A-in,B-in),
   p-out(Pal-out,Pal-in,A-in,B-in).

a-out(nil,_,nil).
a-out(A-in,A-in,B-in).

p-out(true,P-in,A-in,B-in):-
   ((B-in = nil);(P-in = true,A-in = B-in)).
p-out(false,_,_,_).

b-out(B-in,A-in,B-in):- A-in / = = nil,B-in / = = nil.
b-out(A-in,A-in,B-in):- ((A-in = nil);(B-in = nil)).

set-register-a(nil,nil).
set-register-a(In,nil):- e-register(1, 0 ,In).
set-register-a(In,Out):- e-register(1,Out,In).
set-register-b(nil,nil).
set-register-b(In,nil):-e-register(2, 0 ,In).
set-register-b(In,Out):-e-register(2,Out,In).
read-register-b(nil):- e-register(2 , 0, 0 ).
read-register-b(Read):-e-register(2,Read,Read).
```

図 17 述語 palindrome の定義
Fig. 17 A definition of the predicate 'palindrome'.

破線で囲まれた回路の入力を In, Pal-in, 出力を Out, Pal-out とする．この破線で囲まれた回路を Prolog の述語 "palindrome (In, Out, Pal-in, Pal-out, R-Ai, R-Bi)" で記述し，**図 17** に示す．この述語は，A, B 両レジスタの内容を示す変数 R-Ai, R-Bi を設けそれぞれの値を返すように記述している．このプログラム中の述語 "set-registor", "read-registor" はそれぞれの A, B レジスタに対し値の代入，参照を行う．

VisiLog 上で，この述語 "palindrome" を3個用い最大6文字までの文字列が回文かどうかを判断するシミュレーションを行う例を示す．スプレッド・シートのA列に HOST を B 列に組み合わせ回路 PP 1, C 列に PP 2, D列に PP 3 を示すように配置する．まず，1行目のセルを，左から A0, A1, A2, A3 レジスタを表すものとする．2行目を B 列目から B1, B

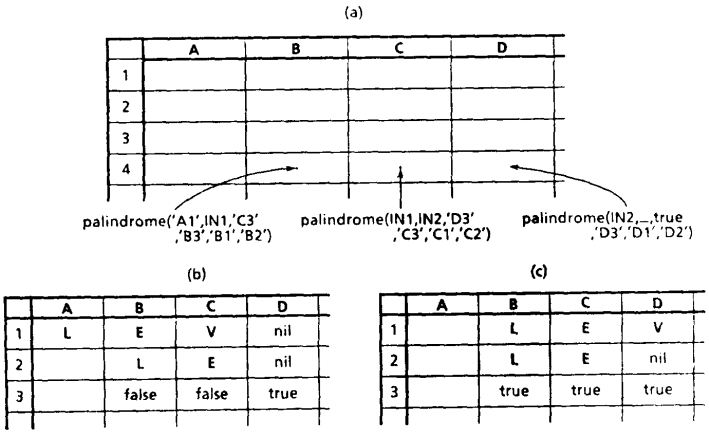


図 18 スプレッド・シートにおけるターミナルの配置とその実行例
Fig. 18 All allocations of I/O terminals to spreadsheet cells and an example execution.

2, B3 レジスタを表すセルとし，3行目は右側から送られてくる真偽値を表すセルとする．組み合わせ回路 PP を記述している述語 "palindrome" を4行目の B 列目から右へ計算式としてエントリする（図 18 (a)）．セル "A1" を文字の入出力装置として使用し，述語間のデータの引き渡しをするために広域変数 "IN 1, "IN 2" を用いる．A, B レジスタの初期値として "nil" をエントリしておく．述語 "palindrome" は，レジスタの値を格納しているセルに対し，直接値の参照や更新を行うように設計されている．この述語は，回路への入力を示す In, Pal-in に値が代入されたとき評価する．そのため，In や Pal-in を表す以外のセルの値が変化することにより自動再計算を行う必要がない．そこで，A0 レジスタを除く A, B レジスタを示すセルに対して自動再計算の禁止を行う必要がある．そのため入力を行っているセル "A1" に，自動再計算を禁止する関数 "no-calculate ([A 2, A 3, A 4, B 2, B 3, B 4])" を計算式としてエントリする．

この状態で文字列 "LEVEL" をセル "A1" に1文字ずつ入力する．図 18(b)は文字列 "LEVEL" のうち前から4文字がすでに入力されており，5文字目 'E' を入力した状態を示す．セル "A1" に値が代入されたので，セル "B4" の計算式が自動再計算される．この計算式が実行されるとセルの値を入力としているセル "C4" の計算式が評価される（図 18(c)）．最後に，セル "B3" の値が "true" となり，入力した文字列 "LEVEL" は回文であることが分かる．

以上に示したように，VisiLog では実行過程の様子をスプレッド・シートのセルを用いて表示することができる．このため，Prolog を用いたシミュレーションが簡単にスプレッド・シート上で行える．

7. むすび

本研究では，VisiLog の Prolog プロトタイプを作成し以下の事柄を実現し，その有効性を考察した．

- Prolog によるスプレッド・シートの記述
- スプレッド・シートを操作する述語の提供
- スプレッド・シートにおける方向性の削除
- スプレッド・シート上での自動再計算機能の実現
- スプレッド・シートのセルに記号を登録し，記号間のユニフィケーションによる計算

VisiLog は，従来のスプレッド・シートにはできなかった文字の計算を Prolog を計算式として実行することで可能にしている．そのため，Prolog を利用しての推論が可能となった．推論過程で生じるキーポイントとなる値をセル変数を用いて任意のセルに割り当てる．この値や，ファクト，さらには推論結果などを2次元的に表示し推論が行える．

また，VisiLog の特徴である双方向計算を利用すると事務計算が便利になる．たとえば，VisiLog 中に "販売価格＝利益＋総原価" で表せる計算式が，計算式中の項目が VisiLog 中のセルと対応されて登録されていたとする．今，販売価格を変化させそれに見合う総原価を求めたいという要求が生じたとする．この時ユーザは，販売価格を示すセルの値を変更し，次に総原価を示すセルを浮動セルとし，評価することにより心要な総原価を求めることができる．このように，VisiLog はユーザインタフェースのよいインタラクティブなシステムであり，カジュアルユーザの意思決定支援ツールとして応用することができる．

参考文献

1) LisaCalc マニュアル，アップルコンピュータ社 (1983).
2) 日本語ロータス 1-2-3 マクロブック，毎日コミュニケーションズ (1987).
3) Hergert, D., 田浦寿敏：MICROSOFT EXCEL WITH MACROS, (株)ビー・エヌ・エヌ (1987).

4) 田中 譲：推論過程の可視化によるヒューマンインターフェイスの改善に関する調査，昭和59年度第5世代コンピュータ調査研究 (1985.3).
5) 松村 修，田中 譲：推論過程の可視化による論理型言語のヒューマンインターフェイスの改善に関する研究，第30回情報処理学会全国大会論文集，3T-9 (1985).
6) 渋谷正弘，田中 譲：スプレッド・シートを用いた論理型プログラミング言語，第34回情報処理学会全国大会論文集，4U-3 (1987).
7) van Emden, M. H., Ohki, M. and Takeuchi, A.: Spreadsheets with Incremental Queries as a User Interface for Logic Programming, *ICOT Technical Report*, TR-144 (1985).
8) van Emden, M. H., Ohki, M. and Takeuchi, A.: Spreadsheets with Incremental Queries as a User Interface for Logic Programing, *New Generation Computing*, Vol. 4, pp. 287-304 (1986).
9) Smalltalk SpreadSheet ユーザーズガイド，富士ゼロックス(株) (1987).
10) Sterling, L. and Shapiro, E.: *The Art of Prolog*, pp. 303-330, The MIT Press (1986).
11) Prolog-KABA Reference Manual, 岩崎技研工業(株) (1984.2).
12) Prolog-KABA 拡張ツール WING Reference Manual, 岩崎技研工業(株) (1986.2).
13) Leisrson, C. E. and Saxe, J. B.: Optimizing Synchronous Systems, *22nd Annual Symposium on Foundation of Computer Science, IEEE*, pp. 23-36 (Oct. 1981).

(昭和63年7月13日受付)
(平成元年4月11日採録)

渋谷 正弘 (正会員)

昭和37年生．昭和59年北海道工業大学経営工学科卒業．昭和62年北海道大学大学院工学研究科修士課程修了．現在，北海道工業大学経営工学科助手．論理型プログラミング，図形を用いたユーザインタフェース等の研究に従事．

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助教授．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，ソフトウェア科学会，人工知能学会各会員．