記　号　処　理 39－1
（1986. 11. 10）

Prolog・Lisp統合化システムにおけるフレームデータの利用

田中明夫　辻村敏　金田悠紀夫　前川禎男
神戸大学　工学部　システム工学科

Prologは強力なパターンマッチング機能や自動バックトラッキングを備えているためその応用分野は広い。しかし手続き的な処理や階層的なデータの取り扱いには向いていない。

本論文では、PrologからLispを呼び出し可能にしたインタプリタシステムの設計試作とPrologからフレーム型データにアクセスできる述語の開発について述べる。これらはPrologの言語としての欠点を探りそれを補うことを目的としている。

Prolog－Lisp System and Implementation of Frame

Akio Tanaka, Satoshi Tsujimura, Yukio Kaneda, Sadao Maekawa
Department of Systems Engineering, Kobe University
1-1 Rokkodai, Nada, Kobe 657, Japan

Prolog is a language with powerful pattern matching and automatic backtracking. So it is widely used in the field of artificial intelligence. But it is unsuitable for describing procedural processing and for representing hierachical data.

In this paper, we consider Prolog interpreter system which is callable Lisp function and accessable data of frame type. This system can execute a mixed type program written in Prolog and Lisp. The purpose of this paper is to search a defect of Prolog language and to supplement it.

1. はじめに

Ｐｒｏｌｏｇは柔軟なリスト処理や非決定的な処理を得意とし推論機構も備えている。そのため、自然言語処理やエキスパートシステムなどの人工知能の応用分野に広く使われている。しかし、プログラムが第１階の述語論理式の集まりであるため手続き的な処理には向いていない。そこで、古くからこの分野で利用されている関数型言語のＬｉｓｐと融合することによってこの部分を補わせようと考え、ＰｒｏｌｏｇからＬｉｓｐ関数を実行可能にした“ＫＬＩＰＳ”（Kobe Lisp Prolog Interpreter System）を設計、試作した。

また、知識表現の形式として最も柔軟で汎用性のあるのはフレーム型である。そこで、ＫＬＩＰＳ上でＰｒｏｌｏｇからフレームシステムに直接アクセスできるようなＰｒｏｌｏｇの述語を開発した。

本研究の目的はＰｒｏｌｏｇという言語の欠点を探り、それを補う方法としてＬｉｓｐとの融合やフレームにアクセスすることなどについて考察することである。この目的に沿って本システムは次のことを基本設計方針として試作された。まずＰｒｏｌｏｇインタプリタを主体としたものにし、ＬｉｓｐはＰｒｏｌｏｇから呼び出される。またＰｒｏｌｏｇ及びＬｉｓｐの言語仕様は一般的なものとし、ＰｒｏｌｏｇとＬｉｓｐ間のインタフェイスは簡単なものにする。

2. システムの全体構成

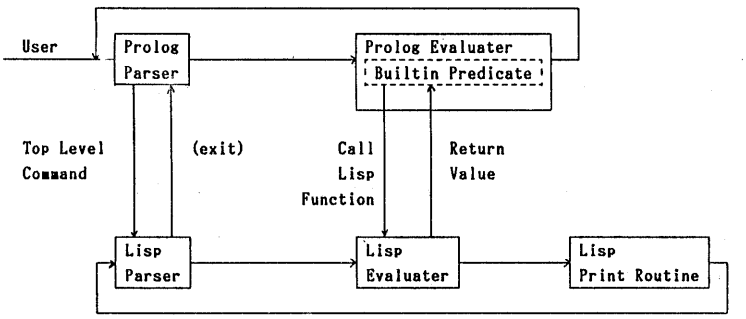


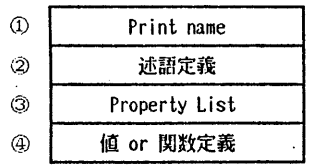
図１　システムの全体構成

システム全体の構成は図１のようになっており、Ｐｒｏｌｏｇ処理系がＬｉｓｐ処理系を包含した形をとっている。ＰｒｏｌｏｇからＬｉｓｐを呼び出すには３通りの方法がある。しかし、Ｌｉｓｐの処理系はあくまでもＰｒｏｌｏｇの処理系のサブシステムとして存在するためＰｒｏｌｏｇ処理系から呼び出さなければ動作しない。またＬｉｓｐ処理系からＰｒｏｌｏｇ処理系を呼び出すことはできない。

本システムはＵｎｉｘ上にＣ言語で記述されているため移植性が高くマイコン、ミニコンレベルで実行できる。

3. 内部形式と各インタプリタの外部仕様

（１）内部表現形式



本システムの内部データ構造は、a)リテラルアトム、b)変数、c)スモールインテジャ、d)セル、e)実行型ファンクタの５種類である。このうちa)c)d)はＰｒｏｌｏｇとＬｉｓｐで共有している。

リテラルアトムは左のような情報を持ったアトムヘッダによって表現される。このうち①②③はＰｒｏｌｏｇシステムで、①③④はＬｉｓｐシステムで使用されている。

図２　アトムヘッダ

（２）Ｐｒｏｌｏｇシステム

Ｐｒｏｌｏｇのクローズは Ｌｉｓｐとの整合性をよくするために２進リスト形式で表現されている。ただしand(“,”)に関しては実行速度を上げるためにｎ引数のオペレータとして表現されている。

Ｐｒｏｌｏｇの言語仕様はDEC-10Ｐｒｏｌｏｇに準じ、構造体の表現方法はストラクチャーシェアリング方式を採用している。また変数はグローバル変数とボイド変数の２種類とする。処理はグローバルスタック、トレイルスタック、プロセススタックを用いて行われる。

ＫＬＩＰＳを起動すると、まずＰｒｏｌｏｇの処理系に入る。Ｐｒｏｌｏｇのパーザが起動すると“Ｐ＞”というプロンプトが表示される。この状態ではＰｒｏｌｏｇのホーン節またはトップレベルコマンドが入力可能である。トップレベルコマンドはＰｒｏｌｏｇとＬｉｓｐの両システムに対して総括的な処理を行うコマンドである。

（３）Ｌｉｓｐシステム

Ｌｉｓｐのパーザが起動すると“Ｌ＞”というプロンプトが表示される。この状態ではＳ式を入力できる。Ｌｉｓｐの関数には、ユーザ定義としてEXPR関数、FEXPR関数、MACRO関数、組み込み関数としてSUBR関数、FSUBR関数の５種類がある。ユーザ関数の定義はそれぞれLAMBDA式、FLAMBDA式、MACRO式によって行われる。変数の束縛方法はShallow Binding方式を採用しており、バインドスタックを用いて処理が行われる。同一アトム名で変数としての値と関数定義との両方を同時に持たせることはできない。また、Ｌｉｓｐの言語仕様はFranz Lispに準ずる。

４．　Ｌｉｓｐの呼び出し方法

Ｐｒｏｌｏｇ処理系からＬｉｓｐ処理系を呼び出すには次の３つの方法がある。

（１）　トップレベルコマンドによる呼び出し

Ｐｒｏｌｏｇのパーザが起動している状態で“[lisp].”というトップレベルコマンドを実行することによってＰｒｏｌｏｇ処理系からＬｉｓｐ処理系へ移りＬｉｓｐのパーザが起動される。この状態ではＬｉｓｐ関数の定義やＬｉｓｐプログラムの実行ができる。(exit)関数を実行することによってＬｉｓｐ処理系から抜け出しＰｒｏｌｏｇ処理系に戻ることができる。

例

```
p> [lisp].

L> (de fact (n) (cond ((zerop n) 1)
                      (t  (* n (fact (sub1 n))))))
fact
L> (fact 5)
120
L> (exit)

p>
```

（２）　組み込み述語による呼び出し

Ｐｒｏｌｏｇの組み込み述語であるcall_lisp/3を用いることによりＬｉｓｐ関数を呼び出し実行することができる（／３は３引数という意味）。この述語の３つの引数は次のような内容である。第１引数は呼び出したい関数名、第２引数はその関数へ渡す引数リスト、第３引数はその関数からの戻り値を受け取る部分である。

この他にもＬｉｓｐシステムのデータを直接操作あるいは参照するための述語としてsetq/2,value/2,putprop/3,property/3の４つがある。

例

```
p> ?-call_lisp(fact,[5],X).

X = 120;
no
p>
```

（３）　ファンクタを関数として実行する方法

この方法はＰｒｏｌｏｇにおけるファンクタを単なる構造体としてではなくＬｉｓｐ関数として評価しようとするものである。このファンクタを実行型ファンクタと呼び、通常は普通の構造体と同様に扱われるが、未定義変数以外のものとユニフィケーションされる時、及び変数の値として出力される時には、その構造体はＬｉｓｐ関数として評価される。これはＰｒｏｌｏｇの記述性を向上させるのに役立つ。実行型ファンクタはファンクタの前に“＃”を付けることによって表現される。

例

```
p> pfact(N,#fact(N)).
p> ?-pfact(5,X).

X = 120;
no
p>
```

（４）　ＰｒｏｌｏｇとＬｉｓｐのインタフェイス

call_lisp述語によってＬｉｓｐシステムを呼び出すには第２引数の引数リストをＬｉｓｐに渡さなければならない。この時リストに含まれる全ての変数のデリファレンスを行いその構造体の完全なコピーがＬｉｓｐシステムに渡される。変数の値が未定義ならばそこをnilとしてコピーが作られる。

５．　評価

Ｐｒｏｌｏｇ処理系の実行速度は約250lipsである。これはDEC-10Ｐｒｏｌｏｇの約１／10～１／15、Ｐｒｏｌｏｇ－KABAの約１／５であるが、内部データ構造が２進リストで表されていることや最適化は行われていないなどの点を考慮するとまずまずのスピードではないかと思われる。

6.　ＫＬＩＰＳ上でのフレームの実現

フレーム型データの特徴としては、それぞれのフレームが１つの概念を記述できることや、階層型データ構造を構成することができるなどである。これらの理由によってフレーム型の知識表現が多くのエキスパートシステムや人工知能研究のモデルとして用いられている。しかしフレームシステム自身は推論機構を持たないので利用者はデモンなどによって推論機構を設計、実現する必要がある。このことは利用者にとって大きな負担になっている。一方、推論機構を持ち第１階の述語論理に基づく言語としてＰｒｏｌｏｇがある。しかしＰｒｏｌｏｇによる知識表現を考えた場合assertionによるデータベースへの登録だけでは不十分な点が多い。さらにプログラムが平坦になり、知識表現の際に必要となってくる階層化や構造化の問題に対しては不適当である。そこで、フレーム型データ構造にとっては負担になっている推論機構をＰｒｏｌｏｇにより補い、Ｐｒｏｌｏｇにとって不向きな構造化、階層化をフレームによって実現できるようにするためにＰｒｏｌｏｇからフレーム型データ構造にアクセスできるような述語の開発を行った。

また、Ｐｒｏｌｏｇとフレーム型データを結合したシステムはいくつか存在するが、それらの場合の多くはＰｒｏｌｏｇとフレームを統一的に扱う新しい言語を設計している。その時にＰｒｏｌｏｇプログラムの記述方法が特別な形に改められてしまっていることがよくある。本システムの場合はその述語の形はＫＬＩＰＳの他のＰｒｏｌｏｇの述語と全く同じである。ＫＬＩＰＳのＰｒｏｌｏｇはDEC-10Ｐｒｏｌｏｇに準じているため、一般のＰｒｏｌｏｇプログラム中からフレーム型データにアクセスできると考えてもよい。

7.　フレーム型データの構造

フレーム型データ構造はフレーム名、スロット名、ファセット名、ファセット値からなり、その４つ組で表される。

ここではフレーム型のデータは次のようなＬｉｓｐの連想リストの入れ子を用いた形式で表現されているものとする。

```
（フレーム名　（スロット名　（ファセット名　（ファセット値）
                                （ファセット値）
                                    :          ）
                            （ファセット名　（ファセット値））
                                :                    ）
                    （スロット名　（ファセット名　（ファセット値））
                                :                          ）
```

ここで各フレーム名はフレームシステムにおいてユニークな名前を持ち任意個のスロットから成っている。各スロットはスロット名といくつかのファセットを持つ。同様に各ファセ

ットはファセット名といくつかのファセット値を持っている。
ＫＬＩＰＳ上では上のようなフレーム構造のデータをＬｉｓｐシステム内にフレーム名と同じPrint Nameを持つアトムヘッダの属性リスト内に蓄えられている。

８．　フレームアクセスシステムの概略及び特徴

本システムはＰｒｏｌｏｇからフレーム型データ構造を扱えるようにするためフレーム構造のデータとマッチングするような述語を実現している。この述語は他のＰｒｏｌｏｇ述語と全く同様に使えるのでＫＬＩＰＳのＰｒｏｌｏｇプログラム内に置いてフレーム型データにアクセスすることが可能である。
このフレームアクセス述語はフレーム型データを検索するだけでフレーム内のデータを書き換えたり、削除したりは行わない。これらのフレーム操作はＬｉｓｐ関数で用意されているのでＫＬＩＰＳのＬｉｓｐシステムを通じて行う。

（１）　フレームアクセス述語ｆｉｌｌ

Ｐｒｏｌｏｇからフレーム型データにアクセスするには、ｆｉｌｌ／４という４引数の述語を用いる。４つの引数は順にフレーム名、スロット名、ファセット名、ファセット値に対応しており、この述語が実行されるとフレーム型データの中からこの４つの引数とマッチングするデータが検索される。４つの引数の任意の位置に変数を置くことができる。引数に変数を含んでいてマッチングするデータが複数個存在する場合には順次バックトラッキングを行って全てのデータとマッチングを行う。このバックトラッキングを行う際、変数が２つ以上ある場合には引数の下位の方よりalternative（次候補）を検索しマッチングを行う。下位の方よりというのはファセット値、ファセット名、スロット名、フレーム名の順である。
またＰｒｏｌｏｇプログラム中に２つ以上のｆｉｌｌ述語があり、同じフレームにアクセスした場合、それぞれのｆｉｌｌ述語はどのフレームのどのファセット値まで検索したかを覚えている。即ち、あるｆｉｌｌ述語によってあるフレームを検索し終わっても、そのフレームをまだ途中までしか検索していない別のｆｉｌｌ述語には全く影響がなく、その別のｆｉｌｌ述語にバックトラックで制御が移った場合にはそれまで検索し終わっているファセット値の次のファセット値から再び検索を続ける。

９．　ｆｉｌｌ述語によるフレームへのアクセス

ｆｉｌｌ述語の実行の方法を例を挙げながら説明する。
次のようなＬｉｓｐの連想リストで表されたフレーム型データがあるとする。

```
(class1  (boy   (value   (mike)
                         (john)
                           :     ) )
         (girl  (value   (jane)
                         (nancy)
                           :     ) ) )
```

```
(class2   (boy    (value    (tom)
                            (bob)
                              :     ) )
          (girl   (value    (susan)
                            (emily)
                              :     ) ) )
```

（１）　ファセット値が変数の場合

？－ｆｉｌｌ（class1, boy, value, Ｘ）.
というゴール節を実行すると、変数Ｘにまず“mike”が束縛される。“；”で強制バックトラックを起こさせると、次に変数Ｘは“john”に束縛される。以下同様にファセット値がなくなるまで続きファセット値がなくなると“ｎｏ”が返される。

（２）　スロット名が変数の場合

？－ｆｉｌｌ（class1, Ｘ, value, Ｙ）.
というゴール節を実行すると、変数Ｘには“boy”が、変数Ｙには“mike”が束縛される。“；”で強制バックトラックを起こさせると変数Ｙには“john”が束縛される。“boy”スロットのファセット値がなくなると変数Ｘには“girl”が変数Ｙには“jane”が束縛され、以下同様に続く。“girl”スロットのファセット値もなくなると“ｎｏ”が返される。

（３）　フレーム名が変数の場合

？－ｆｉｌｌ（Ｘ, boy, value, Ｙ）.
というゴール節を実行すると、“class1”と“class2”の“boy”スロットのファセット値が順に変数Ｙに束縛される。

（４）　２つ以上のｆｉｌｌ述語で同じフレームにアクセスした場合

このフレーム型データで男女のペアをつくることを考える。このためには次のようなゴール節を実行すればよい。
？－ｆｉｌｌ（Ｘ, boy, value, Ｍ）, ｆｉｌｌ（Ｙ, girl, value, Ｗ）.
このゴール節を実行すると、まず変数Ｘは“class1”に変数Ｍは“mike”に束縛される。次に変数Ｙも“class1”に束縛され変数Ｗは“jane”に束縛される。強制バックトラックで変数Ｗは次に“nancy”に束縛され“class1”の“girl”スロットのファセット値がなくなると変数Ｙは“class2”に束縛され変数Ｗは“class2”の“girl”スロットを順に検索する。
２つめのｆｉｌｌ述語にマッチングするデータがなくなるとバックトラックを起こして１つめのｆｉｌｌ述語に戻る。すると変数Ｍはつぎに“john”に束縛され２つめのｆｉｌｌ述語へと進む。これを繰り返し“class1”の“boy”スロットのファセット値がなくなると変数Ｘは“class2”に変数Ｍは“tom”に束縛され、２つめのｆｉｌｌ述語は“class1”“class2”の女子を順に検索する。以下変数Ｍは“bob”…と順次束縛され“class2”の“boy”スロットのファセット値も“class2”の“girl”スロットのファセット値もなくなると“ｎｏ”

が返される。

１０．　ｆｉｌｌ述語のインプリメンテーション

Ｐｒｏｌｏｇプログラム中に複数個のｆｉｌｌ述語がある場合にそれぞれのｆｉｌｌ述語が独立にフレームにアクセスできる必要がある。そのためには各ｆｉｌｌ述語がどのフレームのどのファセット値まで検索したかを覚えておかなければならない。

なおこの述語はＫＬＩＰＳ上で実行でき、バックトラックを起こしたり変数に値を束縛したりする部分はＰｒｏｌｏｇで、実際にフレーム型データにアクセスしたりどの位置まで検索したかを覚えている部分はＬｉｓｐによって書かれている。

１１．　ｆｉｌｌ述語の利点、問題点

本システムの実現による利点としては次の点が挙げられる。読解性に優れたフレーム型データ構造をそのままの形でＰｒｏｌｏｇからアクセスできることと同時にフレーム型データはＰｒｏｌｏｇのassertionの拡張と見ることができるので複雑なデータの宣言が可能である。またマッチング候補が多数ある時にはバックトラッキングしながら順次アクセスできるため一般のＰｒｏｌｏｇ述語と全く同様にプログラミングが可能である。

一方問題点としては以下のことがある。フレーム名が変数の場合には全てのフレーム名をマッチング対象にするため大規模なフレームシステムになると効率が悪くなると思われる。そのためにはマッチング対象をフレームのグループ化などによって限定することが必要であろう。

１２．　おわりに

本研究の目的はＰｒｏｌｏｇの言語としての記述性について考察しその欠点を補う方法について述べた。その方法としてＬｉｓｐとの融合やフレームにアクセスできる述語の開発を行った。ＫＬＩＰＳ上ではＰｒｏｌｏｇ、Ｌｉｓｐの混在したプログラムを実行でき、実行型ファンクタは記述性の向上に役立つと考えられる。さらにフレーム型データをＰｒｏｌｏｇから直接扱えることはＰｒｏｌｏｇのデータ表現を拡張に役立つと考えられる。

しかし、処理速度の向上や検索対象のフレームの限定化などの課題は残されている。

［参考文献］

[1] 辻村敏　“Ｌｉｓｐ関数実行可能なＰｒｏｌｏｇインタプリタに関する研究”，神戸大学工学部システム工学科昭和６０年度修士論文，1985

[2] 伊藤秀明、上野晴樹　“ＺＥＲＯ：Ｆｒａｍｅ＋Ｐｒｏｌｏｇ”，PROCEEDINGS OF THE LOGIC PROGRAMMING CONFERNCE '85,pp83-94,1985

[3] 島健一　“フレーム型知識表現における論理型推論メカニズムの検討”，情報処理学会論文誌 Vol.27 No.9 pp860-868,1986