数理解析研究所講究録
第495巻 1983年 260-280

部分評価に基づく Prologとデータベースの結合

北大工学部　堀内 謙二（Kenji Horiuchi）
田中 譲（Yuzuru Tanaka）

1. はじめに

最近，各分野で人工知能の研究がさかんであり，専門家システムの様な大量のデータを扱う分野にも応用される様になってきた。そのため，大量のデータが扱えそれらを高速に処理できる知識型システム（Knowledge Based System）の必要性が問われている．知識型システムは，膨大な知識を管理し知識の検索や更新を効率良く処理する知識ベースシステムと，知識を用いて推論を実行する推論機構によって構成されると考えられている（fig.1-1(a)）。知識はルールとファクトに分類できるが，専門知識の体系化はファクトの体系化に比べルールの体系化が遅れていることから，当面の巨大知識ベースではルールよりファクトの方が膨大になると予想される。ファクトはデータベース管理システム（DBMS）で管理できるので，fig.1-1(b)のように知識型システムの構成をDBMSと推論機構とに簡略化できる。

知識型システムは，Prolog等の論理型言語で実現するのが簡単であると思われているが，既存する論理型言語のシステムでは大容量化が難しく処理速度も満足できるものではない。そこで，知識型システムとして fig 1-1 (b) の様な構成を考え，推論機構にPrologを用いルールとファクトの一部を管理させ，残りの大部分のファクトをDBMSで管理させる試みが随所で検討され始めている。

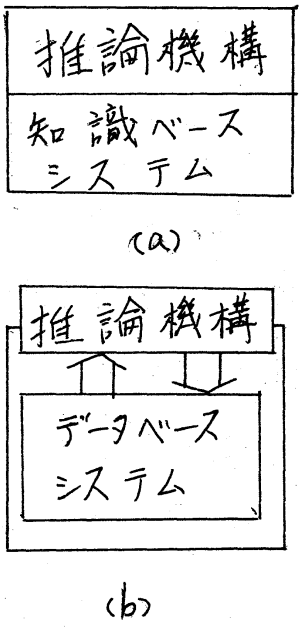

fig. 1-1

しかし，この構成方式には大きな問題点がある。Prologは与えられた質問に対してルールとファクトを用いて推論を行うが，データベース内のファクトの参照が必要になるたびにデータベースのアクセスを行わなければならない。又，そのたび毎に検索結果をPrologシステムへと転送し，Prologのデータ構造に変換しなければならない。この回数は，データベースに格納されるファクトが増えるに従い著しく多くなる。

ここに重大なボトルネックが生じる。本稿では，Prologに部分評価機構を導入し，仮想的に閉じた推論機構で部分評価された結果をDBMSへ関係代数式の形式でコマンドとして一括的に送るという方式を提案する。このことにより，ボトルネッ

ク が解消できる。

2. Prologによる知識ベースシステム

2.1 知識ベースシステム

知識ベースシステムでは「知識」が扱われ，それらはファクトとルールに大別される。ファクトとは「A君の性別は，男である。」のような個々の実体間の関係である。又，ルールとは，「ある実体が人間ならば必ず死ぬ。」という様に特定の実体ではなく，「ある実体が」という様な一般的な実体に関する関係である。

Prologの様な論理型言語では，ファクトやルールは簡単に記述でき，又，推論機構も備えているので，この様な言語は知識型システムを実現するのに適した言語であると考えられている。Prologの論理は一階述語論理内の結論部を一つに限定したHorn述語論理である。上で例として述べたファクトやルールはPrologで表現するとfig 2-1の様になる。ここで，小文字で始まる'a'，'male'，は定記号で，大文字で始まる'X'は変記号である。

```
sex(a, male).
mortal(X) :- human(X).
```

fig 2-1

この様にファクトやルールなどをHorn論理に限定した知識は，Prologで簡単に実現できる。また，これらを用いた知識ベースへの検索要求やルールによる推論

なども実現できる。

今，例として fig. 2-2 の様なファクトとルールからなる知識ベースを考える。二項関係からなるファクト Q，R と三項関係からなるファクト S，そして，述語 P, Q, R 間のルール，これらによって知識ベースが構成されている。これらを Prolog で記述すると fig. 2-3 の様になる。

Q	q.#1	q.#2
	a	1
	a	2
	b	3

R	r.#1	r.#2
	2	m
	3	n

S	s.#1	s.#2	s.#3
	a	3	ℓ
	c	4	m

$$(\forall x)(\forall y)(\forall z)\{Q(x\ z) \wedge R(z\ y) \supset P(x\ y)\}$$

fig. 2-2

```
q(a, 1).   r(2, m).
q(a, 2).   r(3, n).
q(b, 3).
s(a, 3, ℓ).
s(c, 4, m).
p(X, Y) :- q(X, Z), r(Z, Y).
```

fig. 2-3

今，fig. 2-2 の知識ベースに対して，

$$(\exists x)(\exists y)(\exists z)(\exists w)$$
$$\{P(x\ y) \wedge S(x\ z\ w)\} \quad (2.1)$$

を満足する x, y, z, w が要求されたとする。Prolog においては，この要求は，

```
?- p(X, Y), s(X, Z, W).
```
(2.2)

と表現され，fig. 2-3 のプログラムより結果として

$$X = a,\ Y = m,\ Z = 3,\ W = \ell \quad (2.3)$$

が得られる。

この様に，知識ベースへの検索要求を Prolog で実現でき，一

般的な知識ベースシステムに必要な機能はPrologシステムが持っていると考える。

しかし，Prologだけで知識ベースシステムを実現した場合には問題点があり，次にそれについて述べる。

2.2 問題点

知識ベースシステムをPrologだけで実現しようとした時，大容量化にともない次のような問題が生じると考える。

(1) 現在実現されているPrologのシステムにはメモリ階層が支援されていないので，大量のデータを扱う様なプログラムは書けない。

(2) たとえ大容量化が可能であっても，Prologの処理方式でこれらを扱うと処理速度は極端に低下すると考える。

(1)は，Prologにはデータとプログラムを区別しないという特長があるために生じる問題である。大量のデータを二次記憶に蓄えるという事は，即ち，プログラムの一部を二次記憶におとす事を意味する。又，Prologの性質上プログラムのどの部分を用いるかというのは，実行時にしかわからず，アクセスの局所性が成り立たない。このため二次記憶との結合部がボトルネックとなる。また，(2)については，たとえ大容量化が可能であっても，現在のPrologの処理方式は単一化(unification)可能な節(clause)をシーケンシャルにさがすので大量の節に対する処

理能力は満足できるものではない。

しかし，これらの問題はデータベースを利用する事により解消できると考えられ各所で検討されている。データベースは大量のデータを扱ったり，それらからキーによるデータの探索を行ったりする能力に優れている。そこで，ルールに比べてファクトが非常に多い場合，ファクトの全部又は一部をデータベースに記述する事により大容量化は可能である。しかし，この場合でも(2)の問題は解消されない。Prologではルールとファクトを用いて推論を行うが，データベース内のファクトが必要になるたびにデータベースへのアクセスを行い，又そのたび毎に検索結果をPrologシステムへと転送しなければならない。ここに重大なボトルネックが生じる。

このボトルネックを解消する方法として，

(1) データベースに関する評価を後回わしにして，まずProlog内で評価できるものを評価してしまい，最後にデータベースの評価を行うという評価機構（deferred evaluation）[3)4)]。

(2) データベースに関する評価を仮想的に行い，Prolog内で評価を行った後Prolog内の評価における履歴を解析しデータベースにアクセスするという評価機構（partial evaluation），

の二つが考えられる。(1)の評価機構は評価途中において評価の順序を入れ代えるデータベースに関するものは後に回わ

すので，Prolog特有のバックトラックの制御であるカットを扱えない。

本稿では，(2)の部分評価機構を導入することにより，カットを含むPrologの部分評価を実現し，データベースとの結合におけるボトルネックを解消できる事を示す。

3 Prologの部分評価

3.1 関係データベースとSRE

関係モデルに基づくデータベースでは，データベースを関係と呼ばれる表の集合と定義し，表の一部を抜き出したり表と表を結合したりする演算が定義されている。fig. 3-1に関係データベース用検索言語の1つであるQBE (Query By Example) による検索要求の記述例を示す。

Q	q.#1	q.#2
	a	1
	a	2
	b	3

R	r.#1	r.#2
	2	m
	3	n

query： q.#2 = r.#1 でかつ r.#2 = m であるもので q#2 ≠ 3 であるもの

QBE：

Q	q.#1	q.#2
	P.X	Y
¬	X	3

R	r.#1	r.#2
	Y	m

fig. 3-1

fig.3-1のQBEにおいて，Qの一行目の(P.X Y)はこのタプルを(X Y)としてXの値を出力しなさい

という意味で，同じ X が R の一行目の（X m）に現われているので，R を r.#2=m でセレクションした関係と q.#2=r.#1 でジョインすべき事を指示している。又，X は Q の二行目にも現われており，¬（X 3）となっていることから X の値は Q を q.#2=3 でセレクションしたものの q.#1 に含まれていてはならない。fig.3-1 の質問を関係代数式で表現すると次の様になる。

$$\{(Q - Q[q.\#2 = 3])[q.\#2 = r.\#1]\,R[r.\#2 = m]\}[q.\#1] \qquad (3.1)$$

次に，Prolog と DBMS の結合のための中間言語として本稿で用いる SRE（Structured Relational Expression）による検索要求の記述例を示す。SRE は fig 3-2 の形式で記述される。T はターゲット関係，S_i はソース関係と呼ばれる。この形式は $S_1 \times S_2 \times \cdots \times S_n$ の直積の内，Condition 部の条件を満たすタプルのみを抜き出し，T に含まれる属性へプロジェクションを行ったものに T が等しいことを意味する。SRE を用いて fig.3-1 における質問を表現すると fig.3-3 の様になる。ここでソース関係における。$R_1 \cdots R_n \bar{S} T_1 \cdots T_m$ は，$(R_1 \times \cdots \times R_n$

$$T \Rightarrow S_1\ S_2 \cdots S_n$$

Condition

fig. 3-2

Ans (%1) ⇒

Q_1(%1 %) $\bar{Q}_2$(%1 3) R(% m)

(Q_1 1) = ($\bar{Q}_2$ 1),
(Q_1 2) = (R 1),
($\bar{Q}_2$ 2) = 3, (R 2) = m

fig. 3-3

$-(R_1\times\cdots\times R_n\times S))\times T_1\times\cdots\times T_m$ を意味する。SREを中間言語とすることにより，種々の形式のDBMSに対して適したコマンド列を生成することができる。

3.2 部分評価の原理

部分評価機構とは，仮りに閉じた推論機構により部分評価を行い，その結果と残りの知識を用いて評価する機構である。

知識ベースにおける知識の集合を A とする。今，$A=A_1+A_2$ なる二つの集合に分けたとする。A_1 に対して推論を行おうとしてもこの推論は閉じていない。そこで，A_1 の推論機構を仮想的に閉じさせてそこで部分的に推論を行う。その結果得られた情報を基に A_2 上で推論を行う。ここで得られる推論結果は，A において同じ推論を行った時得られる結果と同じものとなる。この様な機構である部分評価機構を用いたPrologとデータベースの結合方式について述べる。

fig.2-3のPrologプログラムの内，q, r, s に関するファクトはデータベース上に記述してある場合を考える。Prolog 上には，ルールはそのまま記述し，データベース内に在るファクトに関しては，同じ述語名毎にアーギュメントをどの様な値とも単一化できる自由変数のままで記述する。この様なファ

```
q(X,Y).  r(X,Y).
s(X,Y,Z).
p(X,Y) :-
        q(X,Z), r(Z,Y).
```

fig. 3-4

クトを仮想ファクトと呼ぶことにする。fig. 2-3の例は，三つの仮想ファクトと一つのルールになり fig. 3-4の様たなる。この様な知識ベースに対して

?- p(X, m), s(X, Y, l). (3.2)

という質問を行うとする。この時の導出過程で，各導出節の内仮想ファクトと単一化されたリテラルをその単一化の時のアーギュメントのまま抜き出すと

q(X, Z), r(Z, m), s(X, Y, l) (3.3)

となる。一方，q, r, s がデータベース内では関係 Q, R, S で表現されている時，上の質問はQBEでは fig. 3-5の様に記述できる。これは本質的には (3.3) 式と等価な情報を含んでいる。

(3.3)式の様に部分評価によってアーギュメントに束縛の加わった仮想ファクトの集合を単一化履歴リストと呼ぶことにする。

Q	q.#1	q.#2
	P. X	Z

R	r.#1	r.#2
	Z	m

S	s.#1	s.#2	s.#3
	X	Y	l

fig. 3-5

一般に，部分評価が途中であってもそれ以前の評価によって仮想ファクトと単一化された導出節におけるリテラルの集合を単一化履歴リストと呼ぶ。(3.3)の単一化履歴リストを解析する事により次の様な関係代数式を得ることができる。

(Q[q.#2 = r.#1] R[r.#2 = m])[q.#1 = s.#1] S[s.#3 = l] (3.4)

この関係代数式をSREで表現すると次の様になる。

$$P(\%_1, m), S(\%_1, \%_2, \ell)$$

$$\Rightarrow \quad Q(\%_1, \%), R(\%, m), S(\%_1, \%_2, \ell)$$

$$(Q\ 2) = (R\ 1),\ (Q\ 1) = (S\ 1),$$

$$(R\ 2) = m,\ (S\ 3) = \ell \qquad (3.5)$$

このSREをデータベースとPrologの中間言語としてDBMSへ送り，データベース内で検索を行う，その結果として

$$P(a, m), S(a, 3, \ell) \qquad (3.6)$$

を得る事ができる。

一般のPrologシステムでは，単一化する時にファクトとルールを区別しないだけでなく，導出中にファクトとの単一化が成功するとその情報は消失する。そのため，(3.3)の様な単一化履歴リストを残こすことができない。そこで，この情報を残こすため，以下の変換規則によりファクト，ルール内に単一化履歴リストH_iを導入する。

$$p(t_1, \cdots, t_n) \Rightarrow p(t_1, \cdots, t_n, ((p, t_1, \cdots, t_n))) \qquad (3.7)$$

ここで述語pに関するファクトがデータベース内にも記述されているとすると，t_iはすべて変記号となる。

$$p_0(t_1^0, \cdots, t_{n_0}^0) :- p_1(t_1^1, \cdots, t_{n_1}^1),$$

$$\vdots$$

$$p_\ell(t_1^\ell, \cdots, t_{n_\ell}^\ell).$$

$$\Rightarrow \quad p_0(t_1^0, \cdots, t_{n_0}^0, H_0) :- p_1(t_1^1, \cdots, t_{n_1}^1, H_1),$$
$$\vdots$$
$$p_\ell(t_1^\ell, \cdots, t_{n_\ell}^\ell, H_\ell),$$
$$\text{superappend}((H_1, \cdots, H_\ell), H_0). \quad (3.8)$$

$$?- p_1(t_1^1, \cdots, t_{n_1}^1), \cdots, p_\ell(t_1^\ell, \cdots, t_{n_\ell}^\ell).$$

$$\Rightarrow \quad ?- p_1(t_1^1, \cdots, t_{n_1}^1, H_1),$$
$$\vdots$$
$$p_\ell(t_1^\ell, \cdots, t_{n_\ell}^\ell, H_\ell),$$
$$\text{superappend}((H_1, \cdots, H_\ell), H_0). \quad (3.9)$$

ここで $p_i(t_1^i, \cdots, t_{n_i}^i, H_i)$ の H_i は，リテラル $p_i(t_1^i, \cdots, t_{n_i}^i)$ を証明した時の単一化履歴リストである。また，superappend(L_1, L_2) は，L_1 にリストのリストを与えた時要素リストを前から順にappendして得られるリストをL_2とする手続き型の述語である。

fig. 3-4の例は，fig. 3-6の様に変換され(3.2)の質問は

$$?- p(X, m, H_1), s(X, Y, H_2),$$
$$\text{superappend}((H_1, H_2), H_0) \quad (3.10)$$

と変換される。(3.10)の質問を実行した結果，単一化履歴リストH_0は，

$$((q, X, Z), (r, Z, m), (s, X, Y, l))$$

に束縛される。この導出では，変記号X, Y, Zは定記号に束縛

されないで自由変数として残されたまま導出は終了する。したがって，この導出による評価方式はPrologにおける部分評価方式であると解釈することができる。

我々が試作したシステムでは，部分評価の結果としてSREでも出力する。この例に対しては，SRE (3.5) と同じものとなる。

3.3 検索モードと帰納的定義

Prologの評価は通常，答を一つだけ求めて終了する。現在実現されているPrologでは検索モードと呼ばれる評価モードを持つものが多く，このモードでは可能な答がすべて探索され出力される。今，このモードの質問を

$$?Q- L_1, L_2, \cdots, L_\ell \qquad (3.11)$$

と表わすことにする。今，fig. 3-6の例に対して

$$?Q- p(a, Z) \qquad (3.12)$$

を実行すると，$Z=n$ が得られこれ以外には解がないことが示される。

```
q(a, 1).  r(3, n).
q(a, 3).
p(X, Y) :- q(X, Z),
           r(Z, Y).
```

fig. 3-6

fig. 3-6のqに関するファクトの最後に

$$q(X, Y) \qquad (3.13)$$

を，又rに関するファクトの並びの最後に

$$r(X, Y) \qquad (3.14)$$

を付加して，(3.12)の質問する場合を考える。この知識ベース

の意味は，q, r に関するファクトは fig.3-6 にあるファクトとそれ以外にデータベース内に記述してあるものもあることを意味している。これらを変換規則に従い変換してから，

?Q- p(a, Z, H₀)　　(3.15)

の質問を行う。この質問に対する H₀は，

H₀=((q, a, 1), (r, 1, Y))　　(3.16)
H₀=((q, a, 3), (r, 3, n))　　(3.17)
H₀=((q, a, 3), (r, 3, Y))　　(3.18)
H₀=((q, a, 3), (r, 3, n))　　(3.19)
H₀=((q, a, Z), (r, Z, Y))　　(3.20)

の5種が得られる。(3.17) は Prolog 内知識のみから得られる答で，(3.20)は DBMS 内のファクトのみから得られる履歴リストで (3.13), (3.14) の仮想ファクトから得られる。他の H₀は，Prolog 内のファクトと DBMS 内のファクトの両方を用いて得られる答に対応している。(3.16)～(3.20)において，履歴リストが生成される時，仮想ファクトと単一化されたリテラルの述語には，下線が引いてある。DBMS に送るコマンドとしては，例えば (3.15) に対しては

R[r.#1 = 1][r.#1]　　(3.21)

であり (3.20) に対しては，

(Q[q.#1 = a][q.#2 = r.#1]R)[r.#1]　　(3.22)

が生成される。このようなコマンドによってデータベースか

ら得られる結果の和集合が検索モードによる質問(3.15)の答である。

このように，ここで示した部分評価機構では，ファクトの一部がPrologに残っていて質問が検索モードで発せられるような場合でも正しい結果を出力することができる。

部分評価では，DBMSへの検索は最後に一度だけ行えばよい。DBMSからの検索結果を用いて推論を再び続行する必要は生じない。

次にルールが再帰的に定義されている場合を考える。fig. 3-7はグラフにおけるpathとarcの意味を考える。ノードを"○"で，ノードXからノードYへのアークを"X○⟶○Y"で表現している。ノードXからノードYへアークを介して行けることを「パスがある」といい，"X○⇝○Y"又は"X○……○Y"で表わしている。

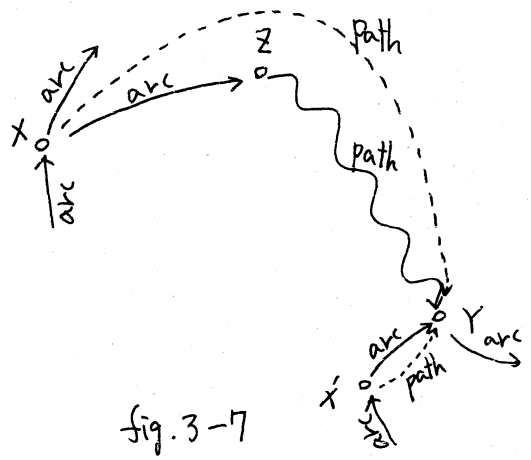

fig. 3-7

パスをPrologで定義し，アークに関するファクトはデータベース内にあるとして仮想ファクトで定義するとfig. 3-8になる。

ここで

```
?- path(X, goal)
```

(3.23)

を部分評価で処理すると，その時得られる単一化履歴リスト

は、

$H_0 = ((arc, X, goal))$

に束縛され，更に新しい答を求めると

$H_0 = ((arc, X, Y)(arc, Y, goal))$

に束縛される。順次DBMSが別の解を要求することにより、Prologはarcの推移閉包(transitive closure) arc^* を [arc.#2 = goal] でセレクションする検索要求を

arc, arc*arc, arc*arc*arc, ・・・

の系列の形でコマンド系列として与える事を意味する。DBMS側では，実際に検索を実行し新しい答が見つからなくなった時Prologへのコマンド要求を停止すればよい。

```
arc(X, Y).
path(X, Y) :- arc(X, Y).
path(X, Y) :-
    arc(X, Z), path(Z, Y).
```

fig. 3-8

4 カットの部分評価

4.1 Prologのカットとnaturalカット

Prologにはカットと呼ばれるバックトラックの制御のための記号があり、通常"!"で表現される。fig.2-3のPを

```
p(X, Y, Z) :- q(X, Y), !, r(Y, Z).
p(X, Y, Z) :- s(X, Y, Z).          (4.1)
```

と定義しなおしてみる。

(4.1)のPの動作は、推論中にPが呼ばれた時、qについて推論がなされ、成功するとrについて推論が進む。ここで、

失敗が引き起こると Q にバックトラックしないですぐ P の失敗を引き起こす。もちろん次の節 s によって定義された P に推論が及ぶ事はない。(4.1) は通常

```
p(X,Y,Z) :- if q(X,Y) then r(Y,Z)
                      else s(X,Y,Z).   (4.2)
```

と解釈されている。ところが、質問

```
?-p(X,2,Z)                             (4.3)
```

は成功し X=a や Z=m を返えすのに、

```
?-p(a,Y,Z)                             (4.4)
```

は失敗する。即ち、(4.2) の解釈は正しくない事がわかる。(4.2)に対応して "||" で表わす natural カットを導入し、P を

```
p(X,Y,Z) :- q(X,Y),||,r(Y,Z).
p(X,Y,Z) :- s(X,Y,Z).                  (4.5)
```

と表現する。not を

```
not(p) :- p,!,failure.
not(p).                                (4.6)
```

と定義すると 次の二つのルール

```
p(X,Y,Z) :- q(X,Y),||,r(Y,Z)           (4.7)
p(X,Y,Z) :- not(not(q(X,Y))),||,q(X,Y),
                      r(Y Z).          (4.8)
```

は等価である。また、first を

first(g) :- g, !. (4.9)

と定義すると,

p(X, Y, Z) :- q(X, Y), !, r(Y, Z) (4.10)

はnaturalカットとfirstを用いて

p(X, Y, Z) :- first(q(X, Y)), ||, r(Y, Z) (4.11)

と表現できる。first(g)はgを充足する最初の値の組を見つけ,gの変記号をその値に束縛する作用を持つ。このことより"!"を"first"と"||"で置き換えてもよいことになる。

4.2 naturalカットとfirstの部分評価

従来のカット"!"はnaturalカット"||"に"first"を加えた制御機構と等価であるので,naturalカットとfirstの部分評価を実現することは従来のカットの部分評価を実現するのに必要十分である。

リテラルPに対して(4.5)の定義がなされているとする。それをアーギュメントを除いた形式で次のように大文字で書く。

P :- Q, ||, R.
P :- S. (4.12)

このPに対しての質問,

?- A, P, B (4.13)

に対する履歴リストはQの成功か失敗によって変わる。しかし基本的には,

(A*, (cut, (Q1*, ···, Qn*), R*, S*), B*) (4.14)

の形を履歴リストに残すことによって部分評価は可能である。ここで，A^*はAをゴールとする導出の単一化履歴リストで，$Q_1^*, \cdots, Q_n^*$はそれぞれAの推論が終わった後Qをゴールとするすべての可能な導出における単一化履歴リストであり，R^*, S^*は，Aの充足の後，Qの充足とは関係なく各々を独立のゴールとして推論を行った場合のそれぞれの履歴リストである。B^*はAの充足の後，Q,R,Sとは無関係にBをゴールとして行った導出の履歴リストである。$A^*, B^*, Q_i^*, R^*, S^*$のSREをそれぞれ r-A, r-B, r-Qi, r-R, r-S とする。又 r-Q を $r\text{-}Q_1 \cup r\text{-}Q_2 \cup \cdots \cup r\text{-}Q_n$ としたとき (4.13) の質問に対するデータベースコマンドは

$$(r\text{-}A * r\text{-}Q * r\text{-}R \cup r\text{-}A * r\text{-}\overline{Q} * r\text{-}S) * r\text{-}B \qquad (4.15)$$

となる。ここで，r-A * r-B は r-A, r-B をデータベースに送る事により得られる関係 A', B' をリテラルA, Bの共通変記号に対応する属性でナチュラルジョインすることを意味している。又，関係 R, S に対して $R * \overline{S}$ は

$$R - (R * S)[\omega(R)] \qquad (4.16)$$

を意味する。ただし，$\omega(R)$ はRの属性集合で R がジョインの単位元でない限り

$$R * \overline{S} \neq \overline{S} * R \quad , \quad R * \overline{S} * T = R * T * \overline{S} \qquad (4.17)$$

を満足する。又，$\overline{S}$ が先頭にある場合のジョインについては，

$$\bar{S} * R = \text{if } S = \phi \text{ then } R \text{ else } \phi \qquad (4.18)$$

と解釈する。

first(Q)の部分評価は，これに対する履歴リストが(first Q^*)となるように1，Q^*に対応するSREを$r\text{-}Q$とするとき，SREとして$f(r\text{-}Q)$を出力する．DBMS側では

$$r\text{-}Q_1 * r\text{-}Q_2 * \cdots * f(r\text{-}Q_\ell) * \cdots * r\text{-}Q_n \qquad (4.19)$$

は$f(r\text{-}Q_\ell)$に先行する$r\text{-}Q_1 * r\text{-}Q_2 * \cdots * r\text{-}Q_{\ell-1}$と$r\text{-}Q_\ell$をジョインし$Q_\ell$に関するタプルIDの最少のタプルへセレクションを加えこれらを抜き出し，残りの$r\text{-}Q_i$をジョインすると解釈される。

5 まとめ

Prologのような論理型言語は，知識ベースシステムを構築するのに強力な言語であるが同時にいくつかの問題を含んでいる。これらの解決方法として関係データベースとの結合について各所で研究がなされているが，この場合でも推論機構とデータベースシステムの結合部にボトルネックが生じると考える。

我々は，このボトルネックの解消法として部分評価が有効な手段であることを示し，その機構を概説した．他にもデータベース上に記述されているファクトに関する評価を後回しにする手法も研究されているが，この方法ではカットを取り扱う

ことができない。部分評価による本方式はカットにも適用でき、現在，リカーション，カットを含むPrologのプログラムの部分評価を実行できる試作システムがProlog/KR上で実働している。

参考文献

1) Kowalski, R. [1979] 'Logic for Problem Solving', Artifitial Intelligence Series 7. NorthHolland

2) Clocksin, W.F. and Mellish, C.S. [1981] 'Programing in Prolog', Springer-verlag Berlin Heidelberg.

3) Chakravathy, U.S., Minker, J. and Tran, D. [1982] 'Interfacing Predicate Logic Languages and Relational Databases', Proc. of the 1st Int. Logic Programing Conf. 91/98.

4) Kunifuji, S. and Yokota, H. [1982] 'Prolog and Relational Data Base for Fifth Generation Computer Systems', Technical Report of ICOT Reserch Center.