ＰＩＥにおける集合演算の実装方式

2C-7

松原　健二　，　相田　仁　，　田中　英彦　，　元岡　達

（　東京大学　工学部　）

1.　はじめに

ＰＩＥは論理型言語をＯＲ並列性にもとづいて高並列に実行する。ＰＩＥの応用分野においては、知識情報処理に代表されるように、高度な推論機能が要求される。そのような分野での問題解決を記述するためには、従来の pure Prolog の範囲内では不十分であると考えられ、プログラム言語に様々な機能を持たせる必要がある。

そこで、有用であると考えられるいくつかの機能について、必要以上に並列性を制限することなく、効率良く記述できることが可能な論理型言語に関する検討を進めている［１，２］。本報告では、その中から解の寄せ集めの実装方式について述べる。

2.　解の寄せ集めの実装方式

ＰＩＥにおいて解の寄せ集めを行なう方式として、次の２つが考えられる。

１）部分解集合を求める問題に分解する方式

２）高機能を持つ構造メモリを用いる方式

2．1　部分解集合を求める問題に分解する方式

次のような定義とＧＦを考える。

```
p(X) :- a(X).
q(X) :- b(X).
?-bagof(X,p(X),S),q(S).---①
```

この例のように、与えられたＧＦと単一化可能な定義節が複数存在するときは、ｐ(Ｘ)の解の寄せ集めは、ａ(Ｘ)の解集合とｂ(Ｘ)の解集合との和集合を求める問題に帰着される。そこでＵＰによりＧＦの書き換えを行ない、あとは子供のＧＦの解が一通りの場合のＡＮＤ並列方式により処理を進める。

まずＧＦ①はＵＰにより次の３つのＧＦにＡＮＤ分割され、書き換えられる。

```
?-bagof(X,a(X),S1),
    $store(S1,&a1).---②
?-bagof(X,b(X),S2),
    $store(S2,&a2).---③
?-$bagmerge(&a1,&a2,X),
    q(X).---④
```

ここで'＄'で始まる名前の述語は構造メモリに対する操作を行なうことを示しており、&a1，&a2はＡＮによりＵＰに供給されている構造メモリの空きアドレスである。＄ store（Ｓ，&a ）は構造メモリのアドレス&a にＳの値をたくわえることを表わす。次にＧＦ②、③を実行する。このときＧＦ④の実行は保留される。この管理はＡＣにより行なわれる。ＧＦ②、③の実行により得られた解は、構造メモリ上にたくわえられる。この解は必ず一通りしかない。

そしてＧＦ②、③の実行が共に終了し、success コマンドが到着したならば、ＡＣはＧＦ④の実行を開始する。

＄bagmerge（&a1，&a2，Ｘ）の実行により、解集合Ｘが求められる。処理の様子を図１に示す。

この方式では構造メモリに必要とされる機能は少なくてすむが、ＡＮＤ並列によるオーバーヘッドが問題である。

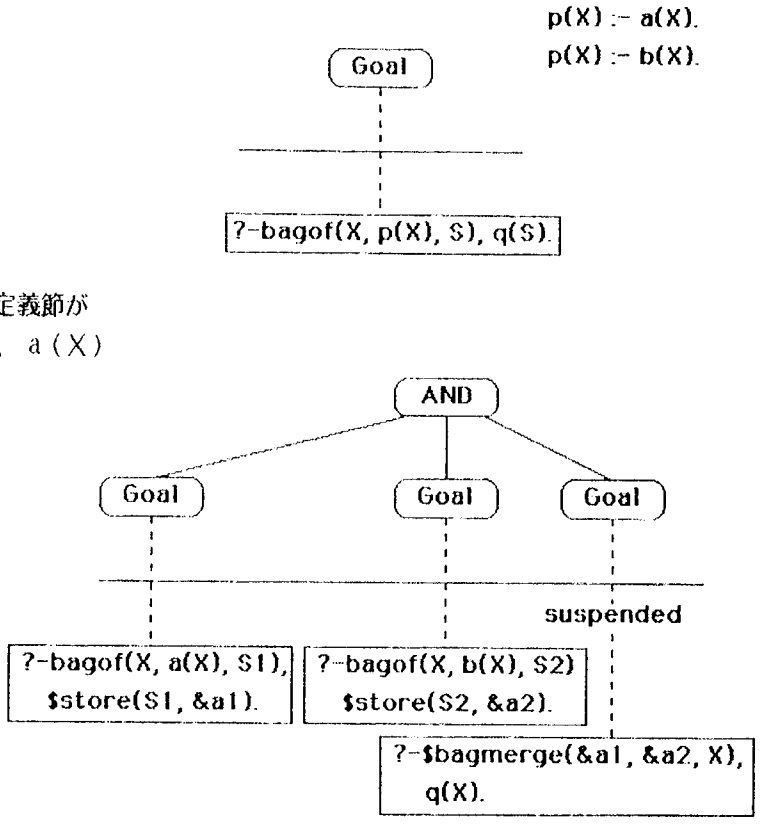

図１．　解の寄せ集めの実装方式（１）

2．2　高機能を持つ構造メモリを用いる方式

構造メモリに解の寄せ集めを行なうの操作を実装すると、構造メモリに対する負荷は増加するが、より自然な方法で解を寄せ集めることができる。

まずＧＦ①はＵＰにより次のようにＡＮＤ分割され、書き換えられる。

```
?-p(X), $app (X, &s ).---⑤
?-q(&s ).---⑥
```

$ app (X, &addr) は、構造メモリのアドレス&addrにXの値を付け加えることを示す。

次にＧＦ⑤が実行される。このとき⑥のＧＦの実行は保留される。この管理はＡＣが行なう。

ＧＦ⑤の実行により得られた解は、構造メモリのアドレス&s に付け加えられていく。ＧＦ⑤の実行が終了すると、ＧＦ⑥の実行が開始される。処理の様子を図２に示す。

１），２）のどちらの方式が適当であるかの判断は、各モジュールに対する負荷を考慮して決めなくてはならない。

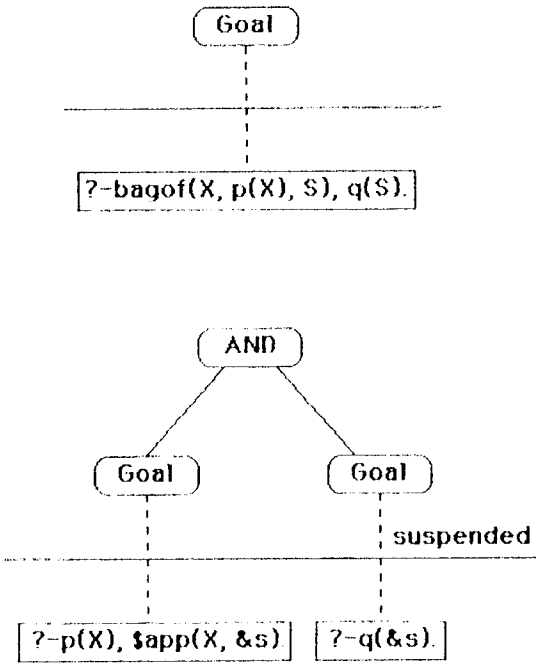

図２．　解の寄せ集めの実装方式（２）

3．　自由変数を含む解の寄せ集めの実現方法

解の寄せ集めを行なう際に、自由変数が現れる場合を考える。このとき、自由変数の値それぞれ（自由変数がいくつかあるときには、その値の直積ひとつひとつ）について、解の集合を求めなくてはならないことがある。次のような定義とＧＦを考える。

```
dick likes beer .      tom likes beer.
bill likes cider.      tom likes cider .
?-bagof (X, X likes  Y, S),
            p(Y, S).---⑦
```

ＧＦ⑦の解集合Ｓは次の２つに分けることができる。

```
Y=beer  S= [dick,tom]
Y=cider S= [bill, tom ] ---⑧
```

自由変数がいくつか現れるときには、一般にすべての自由変数について解集合を分類する必要はない。従って、ある自由変数について解集合を分けたい場合には明示的に表現したほうがプログラムは解りやすい。そこでＧＦ⑦は、⑧のようにＹについて解を分けたいときには、ここでは次のように書くことにする。

```
?-bagof (X on Y, X likes  Y, S),
            p(Y, S).---⑨
```

´on´は後に続く自由変数により、解集合を分類することを表わす。´on´が省略されているときには、自由変数による分類を行なわない。

このようにすると、ＧＦ⑦の実行により得られる解集合Ｓは、次のようになる。

```
S= [dick, tom, bill ]
```

ＰＩＥにおいて、自由変数の値により解集合を分類しようとすると、２つの方法が考えられる［３］。

１）自由変数の値それぞれについて解き直す方法

２）可能な解のすべての組み合せを求め、それを分類する方法

１）の方法では、自由変数の値を求めたあと、その値を元のＧＦに代入して再び解き直す。従って解の分類をする必要はなくなるが、ＧＦを何回も解き直すことになり、効率はよくない。

２）の方法では、すべての解の組み合せを構造メモリにたくわえる。解の分類をＵＰにより行なうか、構造メモリにより行うかにより２つの実装方式が考えられる。どちらの方式が適当であるかは、構造メモリへの解の蓄積方法および自由変数の標記方法の問題と合わせて検討を進めている

4．　おわりに

ＰＩＥにおける解の寄せ集めの実装方式と自由変数を含む場合の実現方法について述べた。

＜参考文献＞

［１］相田他，「ＰＩＥにおける集合および多重世界の実現方式」，第29回情処全大，2B-1，´84

［２］相田他，「並列処理向き論理型言語の提案」、情処ソフトウェア基研，10-5，´84

［３］丸山他，「高並列推論エンジンＰＩＥの階層的構成とそのシミュレータ」，信学技法，ＥＣ84-45 ´84