# 分散環境における ユニフィケーションの実現

7Y-4

六沢　一昭　　市吉　伸行　　近山　隆　　中島　浩

(財) 新世代コンピュータ技術開発機構　　三菱電機(株)

## 1．はじめに

ループの防止と終了の保証は分散環境でユニフィケーション処理を実現する際の主要課題である。

分散環境では他のプロセッサにあるデータへの参照ポインタを扱うが，このポインタを不用意に未定義変数へバインドするとループが生じてしまう。またループの生成を避けるために外部参照ポインタのバインドを行なわず dereference を続けると，ユニフィケーションが永久に終わらない可能性がある。

本稿ではループを防ぎ終了を保証する分散ユニフィケーション方式について述べる。この方式はマルチＰＳＩ第２版[1] 上のＫＬ１処理系に適用する予定である。

## 2．計算モデル

次のような計算モデルを仮定する。

- 局所メモリを持つ有限個のプロセッサ(ＰＥ) がネットワークで結合されている。
- 共有メモリはなく，メッセージ通信によってのみＰＥ間処理が行なわれる。
- ＰＥは自分の局所メモリへの参照ポインタを他のＰＥに輸出できる。このためＰＥは他のＰＥにあるデータへの参照ポインタ(『外部参照ポインタ』と呼ぶ) を持ちうる。

外部参照ポインタを図１に示す。外部参照のチェインを伸ばす形で外部参照ポインタを輸出する(『間接輸出』と呼ぶ．図２参照) ことがある。また輸出済み未定義変数に外部参照ポインタをバインドすることがある。このため外部参照ポインタの参照先がまた外部参照ポインタであることもある。

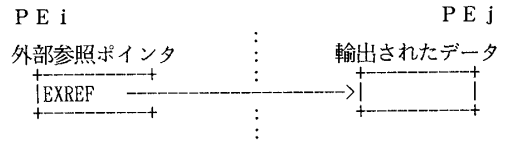
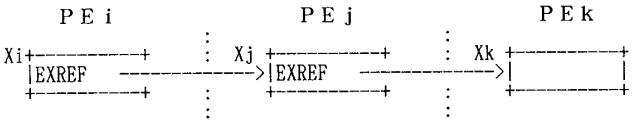



図１．外部参照ポインタ




Xi : 間接輸出によって生じた外部参照ポインタ
Xj : 間接輸出された外部参照ポインタ

図２．間接輸出

---

A Distributed Implementation of Unification

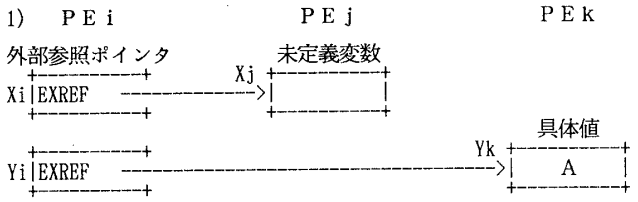
Kazuaki ROKUSAWA *　Nobuyuki ICHIYOSHI *
Takashi CHIKAYAMA *　Hiroshi NAKASHIMA **
* ICOT　** MELCO

## 3．ユニフィケーション

外部参照ポインタＸを含むユニフィケーションは以下のようにして行なう(exref(Ｙ) はＹへの外部参照ポインタを，value(Ｙ) は具体値Ｙそのものを示す)。

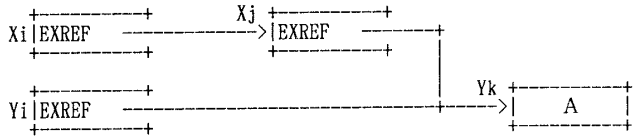
- Ｘと未定義変数Ｙ：
  ＹにＸをバインド，
  あるいは
  %unify(Ｘ,exref(Ｙ)) をＸが指すＰＥへ送信．
- Ｘと具体値Ｙ：
  %unify(Ｘ,value(Ｙ)) をＸが指すＰＥへ送信．
- Ｘと外部参照ポインタＹ：
  %unify(Ｘ,Ｙ) をＸが指すＰＥへ送信．

%unify(Ｘ,Ｙ) を受信したＰＥは，ＸとＹのユニフィケーションを行なう。図３にユニフィケーション例を示す。

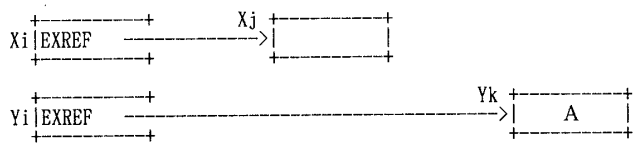



PEi : %unify(Xi,Yi) ---> PEj

2) (ＰＥｊが Xj に外部参照ポインタ Yi をバインドした場合)

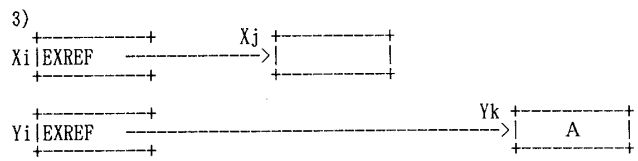



2) (ＰＥｊが %unify を送信した場合)




PEj : %unify(Yi,exref(Xj)) ---> PEk

3)




PEk : %unify(exref(Xj),value(Yk)) ---> PEj

4)

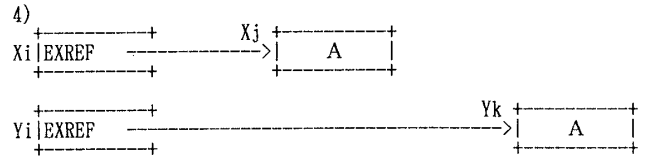



図３．ユニフィケーション(Xi=Yi) の例

4．ループ生成の防止

外部参照ポインタを単純に未定義変数へバインドするとループが生じてしまう(図4)。一方，dereference を続けるとユニフィケーションが永久に終わらなくなってしまう(図4ではＰＥｉ，ｊ間を %unify が永遠に往復する)。

以上を解決するには，バインド規則を設けそれに従って外部参照ポインタをバインドすることが必要である。

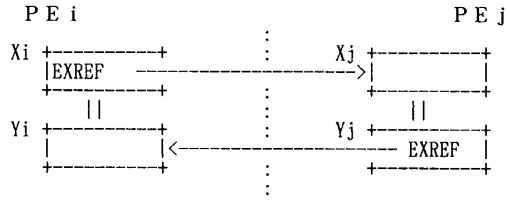



図4．ループ生成の例

4－1．ＰＥ番号によるバインド規則とその限界

間接輸出を認めない計算モデルならば，未定義変数にバインドする外部参照ポインタをＰＥ番号の大きい方から小さい方(逆でもよい) に限ることによってループの生成を防ぐことができる[2],[3]。図4でｉ＞ｊを仮定すると，ＰＥｉではバインドが行われるが，ＰＥｊでは %unify が送信されてループは生じない。

本稿で仮定している間接輸出を認める計算モデルではこの方式はうまく働かない。バインドを行なわない(ＰＥ番号小→大) 外部参照ポインタが，間接輸出によってバインド可能(ＰＥ番号大→小) の振りをしてしまうからである。

図5は，ＰＥｉの未定義変数 Xi を指す外部参照ポインタをＰＥｊがＰＥｉへ間接輸出した状況を示している。ここで Xi と Xi' のユニフィケーションを実行するとバインドが行われループが生じてしまう。

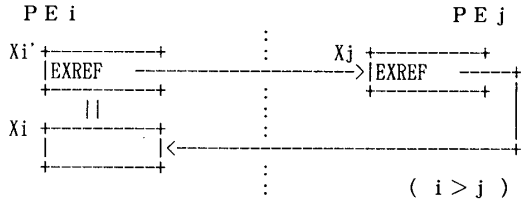



図5．間接輸出によるループ生成例

4－2．Safe/Unsafe属性を用いたバインド規則

以下に示す Safe/Unsafe属性を導入する。

・外部参照ポインタＸは以下の場合 Unsafe である。

(a) ＸのあるＰＥ番号 ＜ Ｘの指すＰＥ番号

(b) Ｘが Unsafe外部参照ポインタを指す.

・外部参照ポインタＸは Unsafe でなければ Safeである。

Safe/Unsafe を用いたバインド規則は以下のとうり。

「Safe外部参照ポインタのみバインドする。」

図5で Xj は (a) により Unsafe に，Xi' は (b) により Unsafe となるので，Xi へのバインドは行われない。このバインド規則によりループが生じないことを以下に示す。

あるループが存在したとすると，それは

(1) Safe外部参照ポインタのみからなる。

(2) Unsafe外部参照ポインタのみからなる。

(3) Safe 及び Unsafe外部参照ポインタからなる。

のいずれかになるが，いずれも成り立たない。(1) が成り立たないのは明らかである。(2) が生じるには Unsafe外部参照ポインタのバインドが必要となるのでやはり成り立たない(輸出のみによってループが生じることはない)。(3) は Unsafe外部参照ポインタを Safe外部参照ポインタが指すことになるのでこれも成り立たない。

5．終了の保証

外部参照ポインタが具体値を指す場合ユニフィケーションが終了するのは明らかなので，ここでは未定義変数を指す場合についてのみ考える。

ユニフィケーション開始時点における外部参照のチェインの長さは有限としてよいが，未定義変数には外部参照ポインタがいつでもバインドされうるので，ユニフィケーションを行なっている間にチェインが伸びてしまう可能性がある。しかしＰＥ台数は有限でありバインドされるのは Safe外部参照(小さいＰＥ番号への外部参照) ポインタのみなので，ユニフィケーションの間に起こるチェインの伸びは有限である。従ってチェインの長さが有限である外部参照を考えればよい。

外部参照ポインタ同士(ＸとＹ) のユニフィケーションでは %unify(Ｘ,Ｙ) が送信され，Ｘ側の外部参照のチェインが dereference される。

外部参照ポインタＸと未定義変数Ｚのユニフィケーションは，Ｘが Safe ならばＺにバインドして終了。Unsafe ならば %unify(Ｘ,exref(Ｚ)) が送信され，Ｘ側は dereference されるがＺ側に新しく外部参照ポインタが生まれるので dereference は差し引きゼロとなる。

このため Unsafe外部参照ポインタと未定義変数のユニフィケーションが連続すると dereference が進まずユニフィケーションは終了しないがこれは起こりえない。図6の状態(dereference が進むとこの状態に達する) でのみ外部参照ポインタと未定義変数のユニフィケーションが連続するが，未定義変数を直接指す外部参照ポインタの Safe/Unsafe はＰＥ番号の大小関係と一致するので，２つの外部参照ポインタのいずれか一方は Safe になる。従って必ず Safe外部参照ポインタと未定義変数のユニフィケーションが起こり，ユニフィケーションは終了する。

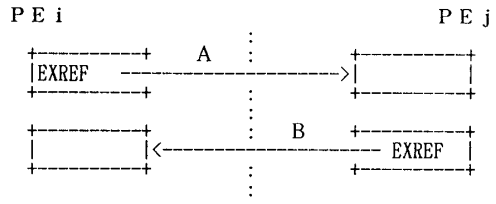



図6．Ａ，Ｂのいずれか一方は Safe である。

6．おわりに

間接輸出を認める分散環境におけるユニフィケーション方式について述べた。本方式は Safe/Unsafe属性を導入することによってループの生成を防ぎ，ユニフィケーションの終了が保証されている。

<参考文献>

[1] K.Taki, "The Parallel Software Research and Development Tool: Multi-PSI system," Technical Report TR-237, ICOT, 1987.

[2] N.Ichiyoshi et al, "A Distributed Implementation of Flat GHC on the Multi-PSI," ICLP, 1987.

[3] I.Foster, "Parlog as a System Programming Language," PhD thesis, Imperial College of Science and Technology, March 1988.