# 小型化版ＣＨＩの ＳＵＰＬＯＧコンパイラにおける ＯＲ節のコンパイル方式

新　　淳・小長谷　明彦・梅村　護
日本電気（株）Ｃ＆Ｃシステム研究所

## １．はじめに

通産省第５世代コンピュータシステム開発プロジェクトの一環として、論理型言語の高速実行を目的とするバックエンド型推論マシンCHIの試作を完了し、これを改良、小型化した小型化版CHIを開発中である[1]。本稿では、CHIのシステム記述言語SUPLOGのコンパイラにおけるOR節のコンパイル方式について述べる。

## ２．ＯＲ節のコンパイル

実用的なPrologプログラムでは、OR節の使用が不可欠である。

```
祖父(X,Y) :-
  父親(X,Z),(父親(Z,Y);母親(Z,Y)).
```

図１．　簡単なOR節の例

このようなOR節を含むPrologプログラムをコンパイルする場合、図２のようにOR節を別手続きに展開してコンパイルする方法と、別手続きにしないでコンパイルする方法[3]とが考えられる。

```
祖父(X,Y) :- 父親(X,Z),両親(Z,Y).
  両親(X,Y) :- 父親(X,Y).
  両親(X,Y) :- 母親(X,Y).
```

図２．　OR節の別手続きへの展開

別手続きに展開してコンパイルする場合、展開しない場合に比べてコンパイルは簡単になるものの、展開した述語を呼び出すオーバヘッドが増え速度向上の点からは望ましくない。またレジスタ上でアクセスできるデータを一旦メモリに退避しなくてはならないといった欠点がある。SUPLOGコンパイラはOR節を別手続きに展開せずにコンパイルするようにした。本稿ではこの方式について説明する。

## ３．変数の分類

CHIのアーキテクチャは、D.H.D. Warrenの提案した仮想Prologマシン[2]（以下WAM: Warren Abstract Machineと略）を基本としている。

WAMでは、変数が参照される期間に注目して、レジスタ上に割り当てる一時変数と、スタック上の環境フレーム中に割り当てる永久変数とに分類している。更に永久変数を安全変数と非安全変数とに分類している。

SUPLOGコンパイラではWAMの分類を拡張し、拡張ゴールの概念を導入して変数の分類を行っている。これはレジスタを破壊しない組込述語を通常の述語とまとめて１つのゴールとして扱う技法である。例えば、

```
f(X):-var(X),!,g(X).
```

のような節の場合、単純にWAMの分類に従えば、Xは永久変数に分類される。しかし、varおよび!は組込述語であるので、これらを述語呼び出しとはせずにインライン展開すれば[4]、上記節の本体部を１つのゴール（拡張ゴール）とみなせる。拡張ゴールの導入により、コストの高い環境フレームの生成を最低限に抑えることが可能となる。

Prologコンパイラでは、この変数の分類が最も重要となるが、OR節を展開しないでコンパイルする場合も変数をどのように分類するかが鍵となる。次節でOR節中に出現する変数の分類について述べる。

## ４．ＯＲ節中に出現する変数の分類

OR節を別手続きに展開しないでコンパイルする場合の変数の分類もWAMの分類を基本とする。問題となるのは、OR節の中に出現し、OR節の中でどのような実行がすすむかによって分類が異なる変数をどう分類するかである。

```
f(X) :- ( X=b,!,h(X);          ①
          a(X),b(X) ) .        ②
```

図３．　実行過程により分類の異なる変数

上の例では、①のパスに注目すると変数Xは一時変数と分類され、②のパスに注目するとXは永久変数と分類される。ここで変数分類の基準を次のように拡張する。

- 一時変数：全ての実行パスにおいて一時変数として分類される変数。
- 永久変数：一時変数以外の変数。

安全変数の分類も次のように拡張する。

- 安全変数：全ての実行パスにおいて、安全変数と分類される永久変数。
- 非安全変数：安全変数以外の永久変数。

図３の例では、Xを永久変数に分類する。

さて、OR節の中で初めて出現しOR節の後にも出現する変数のうち、OR節の１部の選択肢にのみ出現するものに関しては、次の問題を考慮しておく必要がある。

まず、次の例を考える。

```
...,( f(X);          ①
      g    ), j(X),..   ②
```

上の例においては、変数Xは永久変数と分類される。さて、②のパスにおいて、j(X)を実行する時点で変数Xが未定義状態であることが保証されなくてはならない。

---

Compiling Disjunction in SUPLOG Compiler for CHI Compact Version

Atsushi ATARASHI, Akihiko KONAGAYA, Mamoru UMEMURA
NEC Corporation

PLMコンパイラ[3]では、最初のOR節の直前でその時点で未使用の永久変数に対してput_variable命令を生成して解決しているが、コンパイルが複雑になる欠点がある。小型化版CHIではallocate命令において永久変数割り当てと初期化を同時に行うので、コンパイルが簡単に行える。

また、次の例を考える。

```
f :- ( X=a;              ①
       true ) , j(X).    ②
```

この例の場合、Xは一時変数と分類されるが、先程と同様に②のパスにおいてXが未定義状態でなくてはならない。このためには、OR節の前でこの一時変数を初期化することも考えられが、①のパスでグローバルスタックを不必要に伸ばすこととなり、望ましくない。

そこで、SUPLOGコンパイラはOR節の中に初めて出現しOR節の後でも出現する一時変数に関して、各々のOR節の選択肢の最後でまだ出現していないものを初期化する命令を生成する。上の例では、②の選択肢の最後に於てXを初期化する。

## 5．fail時の変数の復元

OR節中の選択肢を実行してfailした場合、OR節の直前の状態が復元されなくてはならない。本節では、fail時に変数の状態をOR節を実行する直前の状態に戻すように注意した点について述べる。

・一時変数

WAMでは、複数の選択肢を実行する場合、fail時に別の選択肢を実行できるように選択肢フレームを生成し、この中に現在有効なレジスタを退避する。failした時は、この選択肢フレームからレジスタを復旧する。

OR節を実行する際にも失敗時に別の選択肢を実行できるように選択肢フレームを生成するので、OR節の直前で有効な一時変数を選択肢フレームに退避するようにした。このとき、nケの一時変数が有効であった場合、これらを$A_0$レジスタから$A_{n-1}$レジスタに連続して割り当てるように注意した。選択肢フレームには0番レジスタから連続して退避するので、GCの際に問題となるごみを含んだレジスタ退避しないようにするためである。

・永久変数

WAMでは、未定義の永久変数に値が与えられた場合、永久変数の属する環境フレームが最新の選択肢フレームより以前に生成されたものであれば、failした時に変数が未定義状態に戻ることが保証される。

このことに注目して、「allocate命令は、永久変数をアクセスする命令、call命令の内、最初に出現するものの直前におく」というWAMの条件を次のように拡張した。

・allocate命令は、永久変数をアクセスする命令、call命令、OR節に対する選択肢を生成する命令の内、最初に出現する命令の直前に置く。

こうすることによって、OR節の実行時には必ず環境フレームが選択肢フレームの下にあるので、fail時に永久変数も正しく復元される。

## 7．コンパイラの出力命令列

今までに述べたことをもとに、SUPLOGコンパイラはOR節に対して次のような命令列を生成する。

```
    ..., ( <選択肢1>;
             :
           <選択肢n>),...

    allocate            % ORの前に生成
        :
    try_me_else(N,$2) % 選択肢生成、一時変数退避
    {選択肢1に対する命令}
    {ORの中と後に出現する一時変数で、
      選択肢1では出現しないものの初期化}
    Jump(Exit)
$2: retry_me_else($3)
        :
    retry_me_else($l-1)
    {選択肢iに対する命令}
    {ORの中と後に出現する一時変数で、
      選択肢iでは出現しないものの初期化}
    Jump(Exit)
        :
$n: trust_me_else_fail
    {選択肢nに対する命令}
    {ORの中と後に出現する一時変数で、
      選択肢nでは出現しないものの初期化}
Exit:
```

## 8．おわりに

小型化版CHIのSUPLOGコンパイラにおけるOR節のコンパイル方式について述べた。本稿で述べた方式を用いることにより、OR節を別手続きに展開することなくコンパイルできる。今後は、ここで述べた技法を実際の応用プログラムに対して評価する予定である。最後に、本方式の検討にあたり、種々の有益なコメントを頂きました神戸日本電気ソフトウェア（株）の阪野正子氏に感謝いたします。

## 参考文献

[1] 中嶋　他：逐次型推論マシンCHI小型化版の設計思想、情報処理学会第３３回全国大会論文集、1986

[2] Warren, D.H.D:An Abstract Prolog Instruction Set, TR309, AI Center, SRI International, 1983

[3] Roy, P.V.:A Prolog Compiler for the PLM, UCB/CSD 84/203, Berkeley, 1984

[4]新　他：小型化版CHIのSUPLOGコンパイラにおける最適化技法、情報処理学会第３４回全国大会論文集、1987