C-Prolog コンパイラの開発（２）

――― 内部構造について ―――

3E-2

○小林　茂，松本憲幸，落合正雄，本位田真一
（株）東芝

1．はじめに

スーパーミニコンをターゲットとした，C-Prologのコンパイラを開発した。C-Prologはエジンバラ大学で開発され，最も普及しているPrologシステムの１つであるが，インタプリタのみでコンパイラは持たない。ここで紹介するコンパイラはインタプリタの述語の１つとして呼出されるもので，その生成オブジェクトもインタプリタの管理下で実行される。本稿では，コンパイルされた述語もインタプリタによる実行のときと同じように呼出せ，かつそのために実行効率を落とさないようにするために，我々が用いた手法を中心に説明する。

2．インタプリタへのオブジェクト述語の組込み

インタプリタの中でコンパイルされた述語（オブジェクト述語と呼ぶ）を実行する際に使用するメモリ資源（ヒープ領域，スタック類等）は，コンパイルされない述語（ソース述語と呼ぶ）と共有とした。また，述語間での制御の移動には，

① call ……子ゴールのための述語呼出し
② continuation …成功による親ゴールへの復帰
③ fail ……失敗による親ゴールへの復帰

の３つの場合があるが，ソース述語からオブジェクト述語への制御の移動はCALL機械語命令，逆方向への移動はRETURN機械語命令で行うこととした。

3．述語の呼出し機構

3.1　制御のためのデータ構造

C-Prologインタプリタでは，プログラムの実行をフレームと呼ぶデータ構造（論理的なスタック上にとられる）により制御する。フレームは述語呼出しに対応して作られる。

オブジェクトの実行は，Warren[1] の提案する方式に従い，主に選択点（choice-point）により制御する。選択点は，代替節を持つ述語の呼出しに対応して作られる。フレーム，選択点とも，述語実行時のレジスタ類を保存するために使われる。これらのレジスタ類の中で，制御に関し，重要なものは，

①Continuation時のための継続ゴールアドレス
②Fail時のための代替節アドレス

であり，後述するソース述語・オブジェクト述語間での呼出し制御では，これらの扱いがポイントになる。

3.2　オブジェクト述語間の呼出し

本コンパイラは，ソースファイル単位でコンパイルを行う。オブジェクト述語間の呼出しは，一般に図１のように，コンパイラされたファイル毎に作られる間接参照テーブルと，述語定義のエントリ構造であるファンクタ構造体を通して行うこととした。ファンクタを通して呼出すことにより，述語が再定義された場合にも，呼出し側は最新の定義を参照できる。

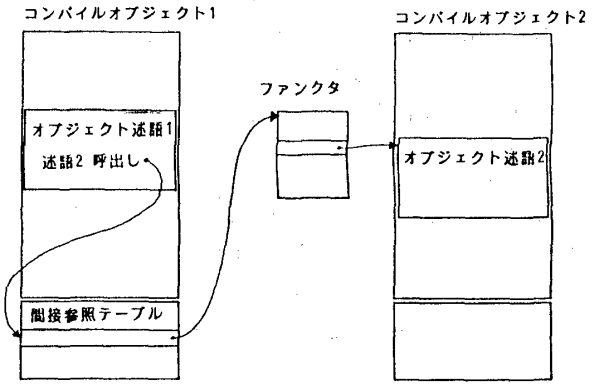

図１．　オブジェクト述語の呼出し

3.3　ソース述語・オブジェクト述語間の呼出し

C-Prologインタプリタのファンクタ構造体は，ソーイ述語を登録するためのフィールドを持っているが，これにオブジェクト述語を登録するためのフィールドを追加した。これらを以下src-def フィールド，obj-def フィールドと呼ぶことにする。これらのフィールドは，述語定義により，表１のような値を持つ。（call-srcルーチンについては後述する。）

フィールド	未定義	ソース述語	オブジェクト述語
Src-def	nil	ソース述語定義	nil
Obj-def	call-srcルーチン	call-srcルーチン	オブジェクト述語定義

表１．　述語定義のフィールド値

An Implimentation of C-Prolog Compiler（２）
S. Kobayashi, N. Matsumoto, M. Ochiai, S. Honiden
TOSHIBA CORPORATION

これらの述語定義に従って，プログラムを実行していく時，ソース述語からオブジェクト述語への制御の移動のために追加される処理の負荷は，インタプリタにとって大きな問題ではない。しかし，逆方向の移動では，移動毎にテストを行うことは，オブジェクトの実行速度やサイズの点から望ましくない。この問題を避けるために，これらの制御の移動を仲介する，次のような仮想的な述語を考えた。

```
call-obj(Obj-goal) : -Obj-goal, cont-src.
call-obj(Obj-goal) : -fail-src.

call-src(Src-goal) : -Src-goal, cont-obj.
call-src(Src-goal) : -fail-src.
```

ここで，

① call-obj …… ソース→オブジェクトcall処理を行う。
② cont-obj …… ソース→オブジェクトcontinuation処理を行う。（必ず成功する。）
③ fail-obj …… ソース→オブジェクトfail処理を行う。（必ず失敗する。）
④ call-src …… オブジェクト→ソースcall処理を行う。
⑤ cont-src …… オブジェクト→ソースcontinuation処理を行う。（必ず成功する。）
⑥ fail-src …… オブジェクト→ソースfail処理を行う。（必ず失敗する。）

このうち，①～③は，実際はインタプリタにより，呼出されるインタフェースルーチンである。④～⑥は，オブジェクト実行制御のために参照される領域，具体的には，

④ call-src …… ファンクタのobj-def フィールド
⑤ cont-src …… オブジェクト述語実行時の，接続ゴールアドレスレジスタ
⑥ fail-src …… オブジェクト述語実行時の，代替節アドレスレジスタ

にセットされるインタフェースルーチンである。これらのルーチンによる実行制御を図２に示す。

4．述語のスコープと呼出しの高速化

C-Prologインタプリタでは，すべての述語はインタプリタ内の任意の箇所から参照できる。

本コンパイラでは，オブジェクト述語の呼出しもファンクタを通して行うため，このスコープの規則は保たれている。しかし，そのために間接参照が深くなっているので，これを直接に呼出せば，オブジェクト実行の効率化が可能である。そこで，特定の述語について，その述語の呼出しを同一ファイル内からのみに限定し，高速な呼出しを行うことを宣言する "local"述語を追加した。

local宣言された述語は，ファンクタによる呼出しではないので，大規模なプログラムの開発時にソースファイル間での述語の重複を避けられるという効果もある。

なお，述語の再帰的な呼出しは，常に local述語型の呼出しとしてコンパイルされる。また，コンパイル時の最適化オプションにより，ファイル内のすべての述語について，同一ファイル内からの呼出しを local述語型とすることも可能である。

5．おわりに

C-Prologのコンパイラについて，述語の呼出し機構を中心に説明した。現在，オブジェクトの実行性能評価，及びオブジェクトの最適化・操作性の改善などを行っている。

〔１〕D. H. D. Warren : An Abstruct Prolog Instruction Set, SRI Insternational, Technical Note 309, (1983)

〔２〕松本 他 ：C-Prologコンパイラの開発（１），情報処理学会，第33回全国大会予稿集，(1986)

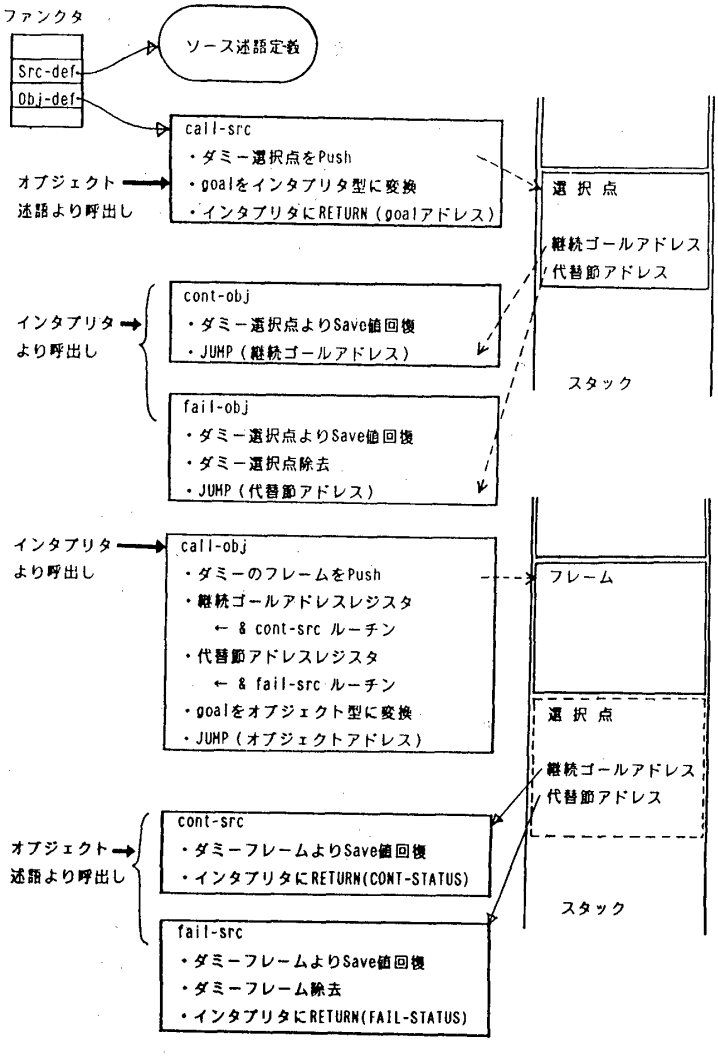

図２． ソース述語・オブジェクト述語間の呼出し処理