5J-6

Common ESP 機械語ジェネレータの概要
-リターゲッタビリティと最適化-

佐藤良治、高橋文男、田中吉広、実近憲昭
(株) AI 言語研究所

1 はじめに

AI 言語研究所 (AIR) では、ICOT で開発された言語 ESP をもとに、汎用商用機上での普及を目的として実用的 AI システム記述言語 Common ESP(CESP) を研究開発している。AIR では昨年度、エミュレータ実行方式 [Tan'89] で第一次試作を行なった。本年度は、まず性能面から機械語生成コンパイル方式を採用し、さらに移植性を考慮して機械語ジェネレータジェネレータ方式を採用した。本報告では、高い移植性と高い性能という相矛盾する目標を同時に達成するため、機械語ジェネレータでとられた基本方式について報告する。なお、CESP は、Prolog をベースとしてオブジェクト指向のモジュラープログラミング機能を融合した言語である。そのほかに制御機能、データ型などを拡張している。また開発環境として使いやすいプログラミングシステムを目指している。

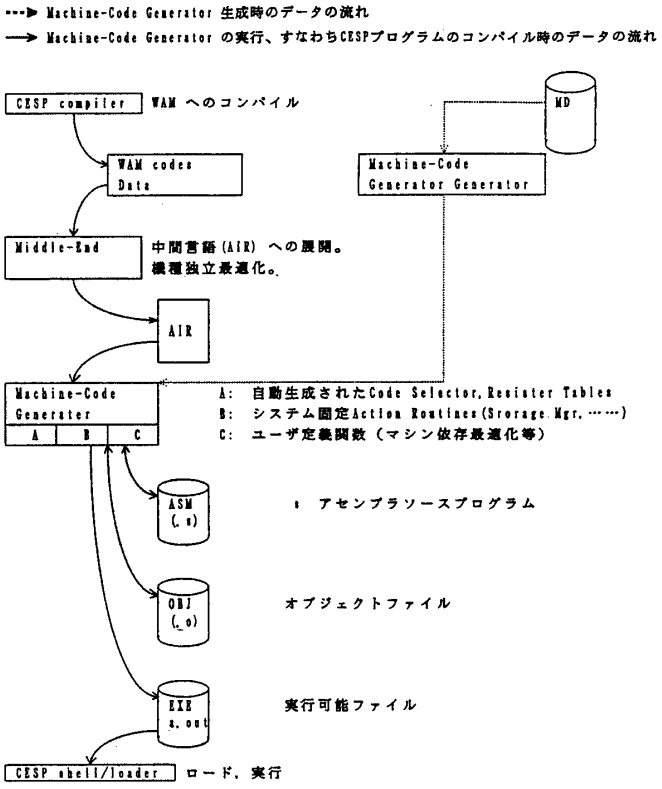
2 言語オブジェクトの実行特性

CESP の機械語生成に関するオブジェクトの特性のうち重要なものは以下のとおりである。
a. 利用する機械語およびアドレッシングモードは単純で数少ない。
b. 間接分岐が多い、動的に実行フローが決まる。
c. 基本ブロックが短い、コンテキストスイッチが頻繁である。
これらの特性から機械語生成部の処理方式を考察すると以下のことがいえる。特質 a のため、比較的単純な中間表現 (IR) やコード生成部 (CG) 構成で実現できる。また特質 b のために、通常の手続型言語で行うような従来のグローバル最適化の手法 [Aho86]、すなわち関数内の基本ブロックにまたがるフロー解析を行なう手法、をそのまま適用することの効果は小さい。一方、特質 c のために、逆に基本ブロックにまたがったなんらかのグローバルな最適化が重要である。

3 全体構成および処理概要

CESP コンパイラは、入力ソースをパーズして継承解析を行い、Prolog やオブジェクト指向の機能を 100 個ほどの CESP WAM[War83] およびテーブル群にコンパイルする。ミドルエンド (ME) が CESP WAM を入力として別の中間表現 (IR) に展開し、機種独立最適化を施す。ついで CG が IR を入力としてコード選択やストレージ束縛等を行ってアセンブラプログラムソースを生成し、アセンブリし、ロードする。CG のうちコード選択部等は、マシン記述 (MD) ファイルからコードジェネレータジェネレータ (CGG) で自動生成される。そうして生成された CG は、実行時ルーチン (RTR) の生成にも使用される。(図参照)

「機械語ジェネレータ，機械語ジェネレータジェネレータ構成図」



4 リターゲッタビリティ

機械語生成部の移植性を通常リターゲッタビリティ (異なるターゲットマシンへの移しやすさ) というが、本システムでは高いリターゲッタビリティを達成するために、以下のことを行なった。
- 機種独立な抽象的中間表現の採用
- 中間表現上で機種独立な最適化を行なう
- 表駆動コードジェネレータジェネレータ方式の採用
- レジスタ割当処理の機種独立取り扱い

我々は、機種依存部を機種独立部から分離し、機種依存部を最小化するために、WAM よりも低レベルに、機械語にマップしやすいような機種独立抽象的中間表現 (IR) を設定した。この IR は、15 の命令、および 6 ユナリオペレータ、11 バイナリオペレータ、10 関係オペレータ、7 種類のオペランドからなる。

ME では、CESP WAM を IR に展開するとともに、機種独立な最適化処理 (モード分離、冗長デレファレン

An Overview of Common ESP Machine Code Generator
- Its Retargetability and Optimization -
Yoshiharu Sato, Fumio Takahashi, Yoshihiro Tanaka, Noriaki Sanetika, AI Language Research Institute, Ltd.

ス除去、分岐チェイン除去等)、インライン化の決定を行う。我々は、最適化処理を機種独立表現の上で可能な限り行なって、移植性と性能向上を同時に達成しようというアプローチをとった[Cho83]。

CG部は、いわゆる表駆動CGG方式をとった。CG部は、コード選択部、ストレージマネージャ(SM)、その他のアクションルーチンなどからなり、CGGが、コード選択部でパターンマッチを行う決定木関数およびSMが管理するレジスタテーブル群を、MDから自動生成する。このCGG方式によってマシン依存情報を非手続き的に記述するだけで機種ごとのCGが生成できるようにした。MDファイルは、マシンレジスタの記述、アセンブラテンプレート、レジスタメモリ転記規則、IRのパターンとアセンブラ行を対応させるマッピング規則などからなる。マッピング規則は、IRのトークンの列で示すパターン、それぞれのトークンに付随する制約や転記規則適用の指定、およびアセンブラ生成等のユーザアクションの指定などからなる。

```
[Mapping rule の例]
:: stm_offset_st Oprd
  Oprd^OnReg(A)?{}:{ForceReg(A,LD)} const
  {Emit("movl $2,$3.reg@($4.val)");}
```

システムあるいはユーザ定義のトークンマクロ機能、および転記規則をマッピング規則と別に抽象化したことにより、マッピング規則の記述量を大幅に減らせた。

またレジスタ割当方式は、命令選択後にレジスタ束縛を行ういわゆる仮想レジスタ方式を採用せず、命令選択の前に可能な処理をやる[Cho83]あるいは同時並行に行なう方式を検討した。

5 最適化処理

これまでProlog系言語の最適化研究はWAMなど高水準の実行方式とメモリ管理方式の観点からのものが多いが、我々はマシン資源の有効利用をめざすという観点から最適化を検討した。我々が特に重視したものは以下のものである。

- レジスタの最適割り付け
- 実行時ルーチン呼び出し時のオーバヘッドの軽減
- インライン展開の活用

レジスタ割当(RA)に関しては、CESPが上述のような動的な特性b、cをもつため中途半端な従来のフロー解析に基づく方式の効果が疑問であるため、ある程度のコード品質が出せると期待できるOn-the-fly方式を採用した。On-the-fly手法は、コード選択と同時に必要なレジスタをその場で割り当てていくものである。この方式は基本的にはローカルな情報しか利用しないものであるが、これを、基本blockの概念の拡張、レジスタトラッキング、グローバルなコンテキスト情報を含むヒューリスティックス(経験則)、などで補った。

実行システムは、組み込み述語等のC関数とアセンブラ記述の実行時ルーチン(RTR)とを使用している。現在、オーバヘッドが最小になるようなRTRパラメータ渡し方法などを検討中である。また現在、部分的なインライン展開を行なっているが、さらにきめ細かいインライン展開方式を検討中である。

現在、プロトタイプがSUN3-60上でAppend-100で123 KLIPS、Reverse-30で104 KLIPS、8Queenで78 KLIPSほどのよい性能を出しており、最終的にこれを上回る性能値を出せる見込みである。

6 これまでの研究との比較

CG方式は、マクロ展開、インタープリタ、トリーマッチング、表駆動パーザ生成と分類できる[Gan82]が、本システムは最後のいわゆるCGG方式である。Prolog系言語のCGGはほかにあまり発表されていない。

CGG方式の研究分野ではGraham-Glanville[Gan85]の影響が大きく、最近は実用的なシステムも開発されている[Tur85,Yat88等]。しかし我々は、これらの属性付文脈自由文法等をベースにした強力なパターンマッチャではなく、以下の理由で、状態遷移オートマタベースの単純なもので十分であると判断した:上記特質aがある、文脈自由文法で書いてもルールのネストが浅い、構文的あいまい性が多く意味処理中心である、複雑な命令にde-compilationする必要のないRISCを重視する。

Prologコンパイラの機械語生成レベルのレジスタ割当を含む最適化に関する研究もほとんど出版されていない。動的特性という点で類似のLISPやSmalltalkでは、抽象マシン変数全部の静的割当、On-the-fly、グラフ色付まで種々試みられている[Kes86,Lar86,Ste89等]。本システムのレジスタ割当手法はOn-the-fly方式をヒューリスティックスで補うというアプローチを採用した。これである程度の良い性能は出せる見込みであり、CESPのような動的な言語では妥当な選択であると思う。

7 まとめと今後の方向

CESP機械語生成部は、強力な言語機能を犠牲にすることなく高い移植性と高い性能を達成することを目標としている。これらを同時に達成するために、CGG方式でCGを自動生成し、機種独立最適化、ヒューリスティックベースのOn-the-flyレジスタ割付などを行った。今後、リターゲッタビリティおよび性能の評価を行いたい。また、有効なヒューリスティックスの体系的研究を行って、CESP等Prolog系の言語に適した機械語生成の最適化の新しい手法を検討していきたい。特にinterprocedural[Wal86]な(句や述語にまたがる)処理の機械語レベル最適化、さらには従来のグローバルフロー解析では全くお手上げの、間接分岐をまたがった最適化技術も検討していきたい。

8 参考文献

[Tan89] '暫定版Common ESPシステムにおける実行方式', 田中他, 情報処理学会第38回前期全国大会,3P-6

[Aho86] 'Compilers: Principles, Techniques, and Tools', A.V.Aho et al, Addison-Wesley, '86

[Cho83] 'A Portable Machine Independent Global Optimizer - Design and Mesurements', F.C.Chow, Stanford Univ.Tech.Note#83-254, Dec'83

[Gan82] 'Retargetable Compiler Code Generator', M.Ganapathi et al, Computing survers V14,#4 Dec'82

[Gan85] 'Affix Grammar Driven Code Generation', M.Ganapathi, ACM TOPLAS V7 #4 Oct'85

[Kes86] 'EPIC - A Retargetable, Highly Optimizing Lisp Compiler', R.R.Kessler et al, ACM SIGPLAN Notices V21,#7, July 1986

[Lar86] 'Register Allocation in the SPUR Lisp Compiler', J.R.Larus et al, SIGPLAN Notices V21,#7, Jul'86

[Ste89] 'A Simple Interprocedural register Allocation Algorithm and Its Effectiveness for LISP', P.A.Steenkiste et al, ACM TOPLAS V11 #1 Jan'89

[Tur86] 'Up-Down Parsing with Prefix Grammars', P.K.Turner, SIGPLAN Notices V21,#12,Dec'86

[Wal86] 'Global register allocation at link-time', D.W.Wall, SIGPLAN Notices V21,#7, Jul.'86.

[War83] 'AN ABSTRACT PROLOG INSTRUCTION SET', D.H.D.Warren, SRI Technical Notes 309

[Yat88] 'Dynamic programming and industrial-strength instruction selection: Code generation by tiring, but not exhaustive search', J.Yates et al, SIGPLAN Notices, V23,#10,Oct'89