EFSコンパイラの基礎的研究

3U－5

野地　健太郎　　山本　章博

日本・アイ・ビー・エム　北海道大学工学部

1．はじめに

Smullyanが導入したEFS(Elementary Formal System)は，最近の研究により言語生成に有用であり，文字列を基礎項とする論理プログラミングであることが明らかになった[1]．これまで構文解析技術などに利用されてきた言語理論は，言語のクラスが大きくなるにつれ，受理機械の構造が変わっていくという欠点があった．EFSでは論理プログラムの意味論により言語の生成と受理を統一的に扱うことが可能である．しかし，EFSの処理系は今まで提案されていない．本稿ではEFSの単一化にパターン照合アルゴリズムを利用し，EFS仮想機械を構成する．

2．EFS と言語

Σを記号の有限集合，Πを述語記号の有限集合，Xを変数の可算集合とする．$(\Sigma\cup X)^+$の要素を**項**，Σ^+の要素を**基礎項**と呼ぶ．本稿では記号を$a, b, c, \ldots$，述語記号を$p, q, \ldots$，変数を $x, y, z, x_1, x_2, \ldots$，項を$\pi_1, \pi_2, \ldots, \tau_1, \tau_2, \ldots$で表わす．アトム，基礎アトム，論理式，節，空節，グランド節，確定節，ゴール節，代入などを一階述語論理と同様に定義するとき，確定節の有限集合Γを**EFS**といい，Γの要素をΓの**公理**という．EFS Γとn引数の述語記号$p\in\Pi$に対して

$$L(\Gamma, p)=\{(w_1,\ldots,w_n)\in(\Sigma^+)^n \mid \Gamma\vdash p(w_1,\ldots,w_n)\leftarrow\}$$

と定義する．言語$L\subseteq\Sigma^+$は$L=L(\Gamma,p)$となるようなΓ, pが存在するとき，**EFS定義可能**，あるいは**EFS言語**であるという．

例　$\Sigma=\{a, b, c\}$, $\Pi=\{p, q\}$とし，EFS Γを

$$\begin{aligned}\Gamma=\ & q(abc)\leftarrow\\ & q(axbycz)\leftarrow p(x, y, z)\\ & p(ax, by, cz)\leftarrow p(x, y, z)\\ & p(a, b, c)\leftarrow\end{aligned}$$

とする．このとき$L(S, q)=\{a^nb^nc^n \mid n\geqq 1\}$である．

EFSの実行は基礎ゴール節$\leftarrow A$ (Aは基礎アトム)からのSLD反駁である．公理の本体に出現する変数が必ず頭部に出現していれば，$\leftarrow A$からの反駁には基礎ゴール節しか現われない．このように変数の出現を制約してもEFSの言語の生成力に影響を与えないことが知られている[1]．さらに本研究では述語記号は全て1引数と仮定する．引数を区切るためのカンマ（，）を記号と考えれば，このように仮定しても一般性を失わない．

3．単一化

2節の2つの制約を満たすEFSを実行するときには基礎ゴール節中のアトム$p(\pi)$と公理の頭部の単一化を考えればよい．それでも，公理の頭部の引数は変数と記号の列であるから，単一化代入はただ一つではない．そこで単一化にAho-Corasick法[2]などのパターン照合アルゴリズムを用いる．そのため公理の頭部の形を次の4とおりに制限する．

1)$p(\pi_1x_1\pi_2\ldots x_{n-1}\pi_nx_n)$　2)$p(\pi_1x_1\pi_2x_3\ldots\pi_{n-1}x_{n-1}\pi_n)$
3)$p(x_0\pi_1x_1\pi_2\ldots x_{n-1}\pi_nx_n)$　4)$p(x_0\pi_1x_1\ldots\pi_{n-1}x_{n-1}\pi_n)$

ただし，$\pi_1,\ldots,\pi_n\in\Sigma^+$，$i\neq j$ならば$\pi_i\neq\pi_j$かつ$x_i\neq x_j$とする．これらの頭部と基礎アトム$p(\pi)$を単一化するときには$\pi_1,\ldots,\pi_n$を基礎項$\pi$中のキーワードとみなしてAho-Corasick法により検索し，キーワード間の文字列を切り出せばよい．以下では公理の頭部が1)の形である場合に詳細を説明するが，僅かに変更するだけで他の場合にも同様の手法が適用できる．

基礎アトム$p(\pi)$と1)の形のアトムが与えられたとき，πの先頭から走査してπ_iがj回目に出現し終わる位置を$w(i,j)$とする．二つのアトムの単一化とはn項組$(w(1,j_1),\ldots,w(n,j_n))$で

$$w(1,1)=|\pi_1|,\quad w(n,j_n)<|\pi|$$

$$w(i,j_i)+|\pi_{i+1}|<w(i+1,j_{i+1})\quad(i=1,\ldots,n-1)$$

を満たすものを全て求めることになる．そこで深さnの木Tで次の5条件を満たすものを構成する．

1) 根Rについては　$label(R)=w(1,1)=|\pi_1|$.
2) 深さi ($i\geqq 2$) のノードNに対してあるjが存在して$label(N)=w(i,j)$.
3) 深さiのノードNがMの親であれば，

A Fundamental Study of EFS Compiler

Kentaro Noji : IBM Japan, Yamato Lab.

Akihiro Yamamoto : Hokkaido Univ., Faculty of Tech.

$label(N) + |\pi_{i+1}| < label(M)$.

4) すべての葉Lの深さはnであり $label(L) < |\pi|$.

5) N, Mが兄弟でNがMより左にあれば

$label(N) < label(M)$.

このようなTが構成できなければ二つのアトムは単一化不可能である．木Tが得られた場合には，Tの根から葉にいたる経路のラベルからなる組$(l_1,\ l_2, \ldots, l_n)$が求める組である．この経路の表す単一化代入は

$$\theta = \{x_1 \leftarrow [l_1, l_2], x_2 \leftarrow \pi[l_2, l_3], \ldots, x_n \leftarrow \pi[l_n, |\pi|]\}$$

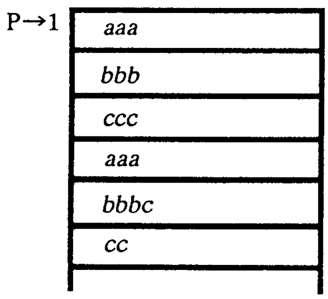
である．ここで$\pi[k,h]$は基礎項$\pi = a_1 a_2 \ldots$の部分項$a_k a_{k+1} \ldots a_h$を表している．

4. EFS 仮想機械

EFS仮想機械の基本構成はPrologの仮想機械であるWAM[3]に従うが，メモリ基本構成はCODE，HEAP，STACK，TRAILの他にUNAを用いて単一化代入の検索を行う点が異なっている．また命令セットもWAMにほぼ従うが単一化とバックトラックに関する部分が異なる．以下においてWAMとの相違点を説明する．

1）UNAの構成

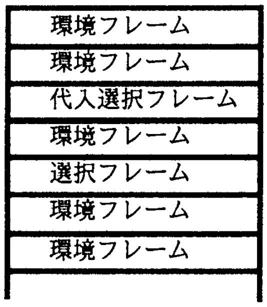
メモリUNAはキューQとして用い代入を格納する．節の頭部が3節の1）の形であれば，単一化が成功し木Tが得れれると，Tを深さ優先探索してラベルの組$(l_1,\ l_2, \ldots,\ l_n)$から代入$\theta$の各束縛の左辺$\pi[l_1,\ l_2]$, $\pi[l_2, l_3], \ldots, \pi[l_n, |\pi|]$をこの順で$Q$に挿入してゆく．UNAの概念図を下に示す．



UNA

2）バックトラックの方法

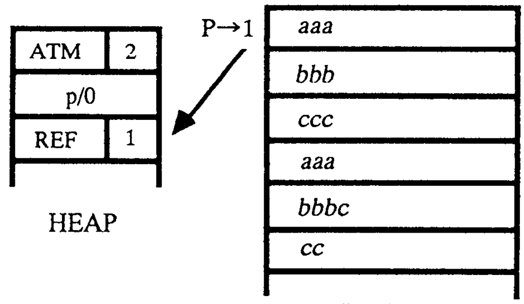
EFSのバックトラックは二種類ある．一つは確定節を選択するためのものであり，もう一つは代入を選択するためのものである．そこでSTACK領域には，WAMと同様に環境フレーム，選択フレームの他に代入選択フレームを格納する．代入選択フレームには代入する文字列のための情報が格納される点がWAMの選択フレームと異なる．STACKの状況を図示すると次のようになる．



STACK

3）単一化代入の求め方

1)で構成したUNAに積まれた代入のポインタをそのままHEAP領域に送る．バックトラックが必要になればUNAのポインタを変数の個数分だけインクリメントすることにより代入を探索する．



UNA

5．まとめ

本稿ではEFSが文字列を項とすることから単一化にパターン照合アルゴリズムを用いて仮想機械を構成する方法について基礎的な考察を行なった．今後研究を進めて行く必要がある課題としては，3節で導入した節の頭部への制約を理論的に解析することや，その制約を弱めるた場合の仮想機械の構成方法を検討することなどがあげられる．

参考文献

[1]S. Arikawa, T. Shinohara, A. Yamamoto : Learning Elementary Formal Systems, Theoretical Computer Science, 95, 97-113(1992).

[2] A.V.Aho, M.J.Corasick: Efficient String Matching, An Ade to Bibliographic Search, CACM,18,333-340 (1974).

[3] H. Ait-Kaci : Warren's Abstract Machine, A Tutorial Reconstruction, The MIT Press(1991).