ショートノート

YAPXの効率的実現法†

林 達也††

筆者は先に，論理型言語による横型下降方式の構文解析法 YAP およびその拡張系 YAPX について述べた[1),2)]．そこでは主として，機能的特長と実現のための基本的なメカニズムならびに時間・空間上の原理的能率に重点を置いて考察した．本稿では，YAPX が持っている時間的・空間的能率に関する高い原理的能力を現実のものとする効率的な実現法について述べる．本手法のポイントは，解析スタックへの格納要素を位置集合（リスト型データ）でなく核位置（アトム）とすることである．核位置とは，位置集合において，他のすべての要素を最左導出する要素のことである．これにより，大幅な速度向上と記憶容量の削減を計ることができる．またこれに伴い，ホーン節の第1引数（入力解析スタックの先頭要素）は不要となる．同時に，ホーン節を左端，中間，右端のタイプ別にグループ化する必要もなく，効率上さらに有利となる．

1. まえがき

筆者は先に，論理型言語による模型下降方式の構文解析法 YAP およびその拡張系 YAPX について述べた[1),2)]．そこでは主として，機能的特長と実現のための基本的なメカニズムならびに時間・空間上の原理的能率に重点を置いて考察した．本稿ではこれに対して，YAPX の効率的な実現法について検討する．

具体的な最適化の手法としては，以下に挙げるものを考える．

（1） カット記号，差分リストの採用

（2） 引数，条件式（補強項）の導入による解析木ないし解析パスの絞り込み

（3） 核位置なる概念を導入して，解析スタックの要素を位置集合（リスト型データ）[1),2)]でなく核位置（アトム）とする．

これらの中で(1)，(2)に関しては，YAPX[3)]，BUP[4),5)]，BUP-XG[6)]，SAX[7)] で既に導入しているので，ここでは(3)について述べることにする．その場合，省略スタックや解析木スタック等は処理の本筋には関係ないので，以下ではホーン節の引数を解析スタック（入力，出力）に限定して考えることにする．また例題としては，全体を通じて，**図1**の文法例を用いることにする．

2. 実現方法

2.1 方法のポイント

まず位置集合（文献 1)，2) 参照）において，他の要素（位置番号）をすべて最左導出* するような要素（位置番号）のことを「核位置」と呼ぶことにする．そして，核位置 n から最左導出される要素の集合を $f(n)$ で表すことにする．

本方法のポイントは，解析スタック（入力，出力）において，スタック要素である位置集合をその核位置で代表させようとする点にある．これにより，速度向上と記憶容量の削減が計れる．またこれに伴い，ホーン節における第1引数（入力解析スタックの先頭要素を抽出分離したもの）は不要となる．また，規則右辺の構文要素に対応するホーン節を左端型，中間型，右端型にグループ分けして，それぞれに全く同一の入力を引数として一々与えていたが，これを改め全体を一まとめに統合できる．

これにより，無駄な処理操作（出力が空に終ることが自明の処理）を排除することができる．

さて，解析スタックに核位置のみを格納するようにすると，第0規則の s の左位置（⊢）（図1参照）を例外として，一般に文法規則の左端要素の左位置は核位置ではないので，スタックには格納されない．そこで，以下では，アクションを左端型と中間・右端型に分けて説明する．なお，本手法は文法に ε 規則やスラ

† An Efficient Implementation Method of YAPX Parsing System by TATSUYA HAYASHI (Fujitsu Laboratories Ltd.).

†† (株)富士通研究所

* ここでは，図1のように位置番号を付与された文法規則において，右辺の記号 A（非終端）（左の位置番号を n とする）と左端の記号 B（終端または非終端）（左の位置番号を l とする）に関して，$A \overset{*}{\Rightarrow} B\gamma$（$\gamma \in V^*$）のとき，「$n$ は l を最左導出する」と言う．

```
#
0   s.  →  |- -|
           s -|

1   s   →  0  1
           np vp

2   np  →  2
           [n]

3   np  →  3   4   5
           [det] [n] rel

4   vp  →  6   7
           [vt] np

5   vp  →  8
           [vi]

6   rel →  9      10
           [relpn] cs

7   cs  →  11 12    13
           cs [conj] s

8   cs  →  14
           s
```

図1 文法例
Fig. 1 A context free grammar.

ッシュカテゴリを含む場合も，全く同様に適用することができる．

2.2 中間・右端型アクション

この場合は，構文要素の左位置が直接スタックに格納されている．そこで，構文要素を $^{n}A^{m}$ (n, m はそれぞれ左位置，右位置．A が右端型の場合は m はなし) とすると，対応するホーン節は次のようになる．

(i) $a([n|X], [[m|X]])$. … (中間型)
(ii) $a([n|X], Y) :- b(X, Y)$. … (右端型)

ここで，第1引数：入力解析スタック，
第2引数：出力解析スタック，
b：当該規則左辺の要素に対応する述語，
である．

図1の $^{4}[n]^{5}$ や ^{5}rel に対応するホーン節は次のようになる．

$n([4|X], [[5|X]])$.
$\mathrm{rel}([5|X], Y) :- np(X, Y)$.

2.3 左端型アクション

この場合は，左位置が直接スタックには格納されていない．そこで左端要素 $^{l}A^{m}$ に対しては，ホーン節を次のようにする．

$a([n_1|X], [[m, n_1|X]])$.
$a([n_2|X], [[m, n_2|X]])$.
⋮
$a([n_p|X], [[m, n_p|X]])$.

ここで，$l \in f(n_1), f(n_2), \cdots, f(n_p)$.

例えば，図1の ^{3}[det]^{4} の場合，次のようになる．

det([⊢|X], [[4, ⊢|X]]).
det([7|X], [[4, 7|X]]).
det([10|X], [[4, 10|X]]).
det([13|X], [[4, 13|X]]).

本手法ではこのように，左端型のホーン節は対応する核位置の数だけ必要となるので，一般にプログラムサイズは増加する傾向にある．ただしその反面，重要な側面として，終端記号の左端要素に対して，核位置は一種の文脈的役割を果たすので，条件式を用いて早期に解析木あるいは解析パスの絞り込みを行うことができる（例えば，動詞を読み込んだ時点での性数，テンス，ボイス，現在分詞，過去分詞，動名詞のチェック等)．

一方，速度を多少犠牲にしてもよい場合には，

det([n|X], [[4, n|X]]):-member(
n, [⊢, 7, 10, 13]).

のように一つのホーン節にまとめることもできる．

なお，同じ記号の左端要素が複数個存在し，かつ，共通の核位置を持つ場合は，その核位置に対するアクションは一つのホーン節にまとめられる．

2.4 左回帰アクション

左回帰規則の回帰要素のアクションは，左端型として独立して取り扱うことはしないで，その左位置を最左導出する位置番号を持つ同一記号の構文要素のアクションに含める．

すなわち，回帰要素を $^{l}A^{m}$，それを最左導出する要素を $^{p}A^{q}$ (左端型)，$^{q}A^{r}$ (中間・右端型) とすると，

(i) $l \in f(q)$ … (中間・右端型)
(ii) $\exists n \{l, p \in f(n)\}$ … (左端型)

である．よって，

(i) $a([q|X], [[r|X]|[m, q|X]])$.
$a([q|X], [[m, q|X]|Y]) :- b(X, Y)$.
(ii) $a([n|X], [[q, n|X]|[m, n|X]])$.

とすればよい．

例えば，図1の $^{11}cs^{12}$ の場合は，^{10}cs のホーン節に含めて次のとおりとする．

$cs([10|X], [[12, 10|X]|Y]):-\mathrm{rel}(X, Y)$.

3. あとがき

YAPX は(1) 時間的・空間的能率のよい横型下降方式の構文解析手法を採っている，(2) 左回帰規則，ε 規則を共に許す，(3) 言語の外置現象を考慮して，柔軟にスラッシュカテゴリを記述することができる，(4) 条件式の導入により解析パスあるいは解析木の

早期絞り込みを行う，（5）構文処理のみで閉じていない．すなわち論理型言語をベースとして，意味処理との有機的な融合が計られている，といった特長を有している．

本稿はこれに加えて，「核位置」なる概念を導入して，YAPX の効率的な実現を計ったものである．

現在本手法に基づく YAPX システムを開発中であるが，最後に本手法の効果について外し触れておきたい．

まず図1の文法の場合であるが，これは極めて単純なので，位置集合の要素数は1～6と小さい．したがって，効果はそれほど顕著には現れない．ただし，左端要素 *np*, [*n*], [det], [*vt*], [*vi*], [rel*pn*], *cs*, *s* に対応する核位置は，*np*～3 個，[*n*]～4 個，[det]～4 個，その他はすべて1個のみである．このためホーン節は全体で 50 となる．従来の方式だと 71 なので，この場合は逆にプログラムは減少する．

次に中規模の英文法[8]に対して現在検討中であるが，YAPX の場合，左回帰規則に加えて，ε 規則や右辺第一項へのスラッシュカテゴリの適用により，約 300 規則となる．この時，位置集合の要素数は 10^0～10^2 のオーダである．また，条件式を付加した形でのプログラムサイズは従来に比べて約 1.5 倍に増加する．これに対してデータ領域は，今のところいくつかの簡単な例文に対してであるが 1/2～1/20 であった．また，解析時間も 10～50 倍の向上が見られた．

なお，具体的な性能評価に関しては，システム開発が終了した時点で別途報告する予定である．

謝辞 YAPX の効率的な実現法についていろいろと検討を手伝ってくれた宮俊司氏ほか(株)富士通ソーシャルサイエンスラボラトリの関係者の方にお礼を申し上げる．

参考文献

1) 林　達也：論理型言語による構文解析法 YAP について，情報処理学会論文誌，Vol. 29, No. 9, pp. 835-842 (1988).
2) 林　達也：拡張 CFG とその構文解析法 YAPX について，情報処理学会論文誌，Vol. 29, No. 5, pp. 480-487 (1988).
3) 林　達也：YAPX にもとづく自然言語処理のモデル化，情報処理学会論文誌（投稿中）(1987).
4) Matsumoto, Y. et al.: BUP: A Bottom-Up Parser Embedded in PROLOG, *New Generation Computing*, Vol. 1, No. 2, pp. 145-158 (1983).
5) 田中穂積ほか：ボトムアップ構文解析システム BUP 上での英語文法開発と BUP の評価，*Proc. LPC 84*, p. 11 (1984).
6) 今野　聡，田中穂積：左外置を考慮したボトムアップ構文解析システム，コンピュータソフトウェア，Vol. 3, No. 2, pp. 19-29 (1986).
7) 松本裕治，杉村領一：論理型言語に基づく構文解析システム SAX，コンピュータソフトウェア，Vol. 3, No. 4, pp. 4-11 (1986).
8) 東工大田中研究室：中規模英文法（仮称）(1987).

（昭和63年10月24日受付）
（平成元年7月18日採録）

林　達也（正会員）

1937年生．1960年早稲田大学第一理工学部応用物理学科卒業．同年富士通(株)入社．以来，ソフトウェア工学（コンパイラ自動作成，設計支援，部品化），データベース（リレーショナル，分散，マルチメディア），自然言語処理（機械翻訳，自然言語インタフェイス，情報検索），人工知能（エキスパートシェル，知識表現）等の研究開発に従事．富士通研究所情報処理研究部長代，ソフトウェア研究部長を経て現在川崎研究所所属．元本学会編集幹事．1973年度本学会論文賞受賞．ACM，日本ソフトウェア科学会，日本モーツァルト協会各会員．AAI (Applied Artificial Intelligence) (Hemisphere) Editorial Board メンバ．