L属性文法に基づく意味解析の並列アルゴリズムの開発

6G−6

椎名広光　増山繁

shiina@toki.tutkie.tut.ac.jp, masuyama@tutkie.tut.ac.jp

豊橋技術科学大学知識情報工学系

1 はじめに

本稿では、LR構文解析の並列アルゴリズム、およびLR文法[2]にL属性を付加した属性文法[2]の意味解析の並列アルゴリズムについて述べる。本並列アルゴリズムは、並列計算の理論的なモデルであるCREW PRAM[1]上で実現している。付加したL属性とは、属性の評価が解析木を先がけ順の処理できる属性の部分クラスである。L属性は、逐次処理の$LL(k)$構文解析[2]の中で同時に処理できるが、LR構文解析では、同時に処理できない。そこで、同時に処理するためにL属性に制限をつけたLR属性[5]が提案されている。

一方、構文解析の実行中に意味解析を同時に行う並列アルゴリズムを開発しようとすると、プロセッサ数が多くなってしまう。そのため、プロセッサ数の増大を押えるために、構文解析を終了してから意味解析を行うことにした。そのことにより、プロセッサ数の削減と同時に、LR文法に付加する属性のクラスをLR属性からL属性に広げることが可能になった。

本研究で考案したLR構文解析の並列アルゴリズムは、入力文字数をn個とすると、$O(n^3)$個のプロセッサを用いて$O(\log n)$時間を要する。一方、意味解析の並列アルゴリズムは、入力文字数をn個とし、L属性の種類をm個とすると、$O(n^4m^4)$個のプロセッサを用いて$O(\log^2 n * \log m)$時間を要する。この並列アルゴリズムにより、L属性文法による意味解析がクラスNCに属することが明らかになった。

また、ここで示す並列アルゴリズムは、つぎの仮定を設けている。

1. 文法が*Chomsky*標準形[3]となっている。
2. 属性の評価式は、多項式で表される。

2 構文解析の並列アルゴリズム

構文解析の並列アルゴリズムは、次の性質[3]を用いている。

- LR構文解析は、与えられた文法に対し予め構成されているLR構文解析表を状態遷移図として用いる決定性プッシュダウンオートマトン[3]による動作と同じである。また、文法が*Chomsky*標準形であるので、LR構文解析表のうち非終端記号から生成する部分のみを考えると、正規化された決定性プッシュダウンオートマトン[1]の動作と同じである。LR構文解析表[2]の終端記号から生成する部分は、スタックを変化させないと考える。このようにすると、入力文字から作られるLR構文解析の状態遷移は、正規化された決定性プッシュダウンオートマトンの動作と類似であると考えられる。

以下では、図1の文法と入力の例を併用して説明する。図1の文法の構文解析表を表1に示す。

2.1 文の認識

前処理　文法の各終端記号を1個入力するとき、それぞれ予想される入力の前後の状態をLR構文解析表を用いて計算する。入力前のLR構文解析表の状態は、LR構文解析表のgoto部分の入力文字に対するエラーでない欄の状態である。入力後の状態は、入力前の状態から終端記号を入力して、逐次処理の構文解析における構文解析表で遷移する状態である。(ただし、以前の状態遷移が記録されていないため、構文解析は1文字入力に対する状態の還元(reduce)の回数が分からない。言い替えれば、各終端記号からでは、スタックに*Pop*する状態の回数が分からないのと同じである。一方、スタックに*Push*する状態は確定している。還元の回数が分からない状態は、特定の場合に限られる。そのため、分からない状態を、0回以上繰り返すとする。)なお、この処理は、文法が与えられたときにあらかじめ1回だけ求めておけば良いので、構文解析の計算量の計算に含めない。

Step 1　状態遷移の大まかな記録を計算する。(処理の様子を図2に示し、その結果を図3に示す。)

Step 1.1　構文解析は始めに入力の各終端記号が完全2分木の葉にあると考える。そして、各葉について予想される入力の前後の状態を計算する。ただし、最初と最後の入力記号は特別に扱う。最初の入力記号では、入力前の状態は開始状態だけである。最後の入力記号では、入力後の状態は受理状態だけである。

Step 1.2　入力文字の各文字が完全2分木の葉にあるとみなして、ボトムアップに処理する。完全2分木の各節点での処理を1つのまとまった処理とする。各節点の処理では、その節点の下にある文字に対する入力の前後の状態を計算する。完全2分木での同じ高さの節点(兄弟)は独立に処理される。各節点では、左右それぞれの子供の状態の組合せ(節点の下にある終端記号を入力して得られる予想される入力の前後の状態)に対してプロセッサを割り当てる。各プロセッサでは、ある節点の左側の子の入力後の状態と右側の子の入力前の状態の左側の子の入力後の状態と右側の子の入力前の状態が一致したとき、次の処理をする。

1. その節点の入力前の状態を左側の子の入力前の状態とする。
2. その節点の入力後の状態を右側の子の入力後の状態とする。

Step 1.3　完全2分木の根までStep 1.2を繰り返し計算する。計算回数は、log(入力数)回で、根まで到達し、そこで処理を終了する。

Step 2　未知の*Pop*する状態の回数を計算する。これは、連続するスタックに*Push*する回数から計算する。

Step 2.1　連続して*Push*する状態の回数を求める。まず、スタックを変化させる状態のみについて考える。*Push*する状態を左括弧とし*Pop*する状態を右括弧であると考える。

Step 2.1.1　*Pop*する状態に適合する状態の連続する個数を1つの左括弧とする。その左括弧の重みを連続する個数とする。この連続する個数の計算は、*String Matching*の並列アルゴリズム[1]を適用する。また、*Push*状態のところの右括弧の重みは、0以上とする。

Step 2.2　重みを付けた括弧について対応する左右の括弧を求める。

Step 2.3　対応する括弧において左括弧の重みを右括弧の重みとする。

Step 2.4　右括弧の重みをそれに対応する*Pop*の回数とする。

Step 3　状態の遷移過程が決定性プッシュダウンオートマトンによる受理に対応するかどうか調べる。

Step 3.1　*Push*する状態と対応する*Pop*する状態が適合するか調べる。すなわち、スタックに*Push*で積んだ状態を対応する*Pop*がスタックから降ろせば適合するとする。

以上のアルゴリズムは、$O(n^3)$個のプロセッサを用いて$O(\log n)$時間で実現される。

2.2 構文解析木の作成

構文解析木の作成には、状態遷移の記録を使う。状態遷移の記録は、文の認識の過程で作られる。結果は、パーステーブルに格納される。

A Parallel Algorigthm for Semantic Analysis based on L Attribute Grammar

Hiromitsu SHIINA , Shigeru MASUYAMA

Toyohashi Univ. of Tech.

Step 1　各状態対に対して、生成規則による始まりと終わりの関係があるかどうか計算する。状態対の生成規則の始まりと終わりを弧で示した図を図4に示す。

Step 2　状態の関係(節点)間の親子を示すポインターを計算する。

Step 3　親子を示すポインターから結果をパーステーブルに格納する。

以上のアルゴリズムは、$O(n^2 \log n)$ 個のプロセッサを用いて $O(\log n)$ 時間で実現される。

3　意味解析の並列化

意味解析の並列化では、構文解析木に対して木の先がけ順の訪問の並列アルゴリズム [1] を用い、各節点や枝の処理は、多項式の評価の並列アルゴリズム [1] を適用している。

Step 1　属性の評価式を割り当てる。構文解析木の枝を互いに反対の方向を示す有向枝で置き換える。この有向枝は構文解析木全体でオイラー閉路を構成する。すなわち、構文解析木の根から始めてすべての節点または葉を通過して再び根に戻る有向路がある。よって、有向枝は、1列にならんでいると考えることができる。プロセッサを各有向枝に1つずつ割り当て、各属性の評価式を有向枝に割り当てる。割り当て方を以下に示す。

Step 1.1　継承属性 [4] を割り当てる。構文解析木の根から葉の方向へ向いている各有向枝には、継承属性の値を代入する評価式を割り当てる。各節点では、そこで生成する生成規則に関する状態の遷移過程の始めと終りの状態のデータを持っている。特に、継承属性の評価の場合には、葉を除く各節点で生成する規則と有向枝の指す先が持っている状態を利用して、どの評価式を割り当てるかが一意に決められる。

Step 1.2　合成属性 [4] を割り当てる。構文解析木の葉から根の方向へ向いている各有向枝には、合成属性の値を代入する評価式を割り当てる。合成属性の評価の場合には、各節点から出ている有向枝の指す先が持っている状態を利用して、どの評価式が割り当てられるかが一意に決められる。

構文解析木と属性の評価式の割り当ての様子を図5に示す。

Step 2　属性値を評価する。

Step 2.1　付け加えた各枝にプロセッサを割り当てる。各プロセッサでは属性値を計算する。ただし、属性値の評価式の入力変数が未定の時は、計算できない。すなわち、分割統治法 [1] で実現できる。

Step 2.2　Step2.1 での有向枝はオイラー閉路をなすので、根を始めとする1本の列に並んでいると考える。それらの各有向枝を葉とみなして、完全2分木を作る。属性値の評価式の入力変数の値が分かっている節点は、属性値を評価する。分からない節点は、左右の子供の評価式を合成する。属性値の評価は、多項式の評価 [1] と同じである。

Step 2.3　Step2.2 の処理を log(有向枝の数) 回繰り返す。

以上のアルゴリズムは、$O(n^4 m^4)$ 個のプロセッサを用いて $O(\log^2 n * \log m)$ 時間で実現される。

4　まとめ

本稿で示したアルゴリズムより、LR 文法に L 属性を付加した意味解析は、クラス NC に属している問題であることが分かった。今後は、アルゴリズムの改善と属性のクラスを拡大した意味解析の並列アルゴリズムを開発したいと考えている。

参考文献

[1] A.Gibbons,W.Ryyter,Efficient Parallel Algorithms , Cambridge University Press,(1989).

[2] A.V.Aho, R.Sethi; J.D.Ullman, Compilers (Principles, Techniques, and Tools) Addison-Wesley (1986),原田 賢一訳, コンパイラ 1,2(原理・技法・ツール), サイエンス社,(1990).

[3] J. ホップクロフト J. ウルマン: オートマトン 言語理論 計算論 1,2 野崎, 高橋, 町田, 山崎 訳, サイエンス社,(1983).

[4] 佐々 政孝, プログラミング言語処理系, 岩波書店,(1989).

[5] Masataka Sassa,Harushi Ishizuka and Ikou Nakata,A Contribution to LR-attribute Grammars, Journal of Information Processing , Vol 8,No. 3.(1985).

文法　　入力文字
1:E→AA　4:A→a　bcaa
2:A→BA　5:B→b
3:A→CA　6:C→c

図1 文法と入力の例

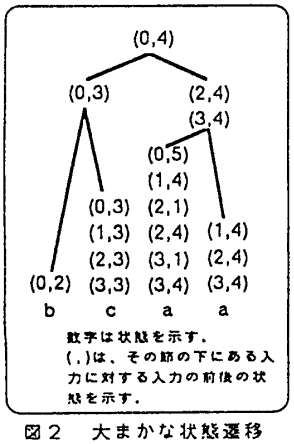

図2　大まかな状態遷移を求める様子

表1: 構文解析表

状態	action				goto			
	a	b	c	⊣	E	A	B	C
0	S5	S6	S7		4	1	2	3
1	S5	S6	S7			8	2	3
2	S5	S6	S7			9	2	3
3						10	2	3
4				ac				
5	r4	r4	r4	r4				
6	r5	r5	r5					
7	r6	r6	r6					
8				r1				
9	r2	r2	r2	r2				
10	r3	r3	r3	r3				

⊣: 入力の最後　r: 還元 (reduce)
S:Shift　ac: 受理

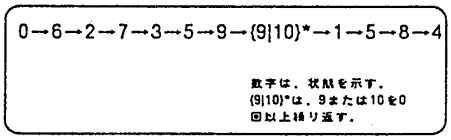

図3　大まかな状態遷移

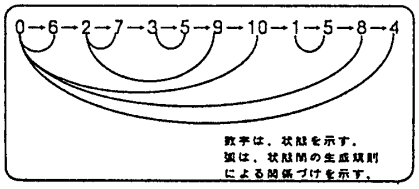

図4　生成規則と状態間の関係づけ

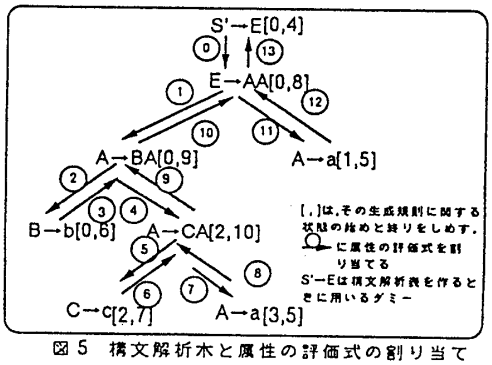

図5　構文解析木と属性の評価式の割り当て