属性文法に基づく意味解析の並列アルゴリズム

3X−2

椎名広光　増山繁
豊橋技術科学大学

1　はじめに

属性による意味解析 [4] は、コンパイラにおける意味解析の1つの手法である。属性とは意味情報のことで意味規則で表現され、文法規則に意味規則を付加したものを属性文法という。属性文法は、1968年 Knuth [1] により初めて提案され、これまでにそれを用いた意味解析の手法が幾つか考案されている。これらの手法は、属性の評価能力に制限をつけて逐次計算機におけるコンパイラ内部の意味解析部で用いられている。従って属性文法は、コンパイラの構文解析に基づきクラス分けされている。(このクラス分けの例を、2に示す。)このように属性のクラス分け [5] は、逐次アルゴリズムからみた解析によって多くなされている。ところが、並列アルゴリズムからみた属性のクラス分けや属性文法を用いた意味解析の並列アルゴリズムの解析は、従来あまりなされていない。そこで、本研究では、幾つかの属性文法に基づく意味解析の並列アルゴリズムをそれぞれ示し、それらの意味解析の問題が全てクラスNCに属することを明らかにする。

なお、本研究に於ける属性による意味解析では、次のような条件を課している。

1. 意味解析に対応している構文解析の文法は Chomsky 標準形 [3] で表現されているものとする。
2. 属性を評価する前に、構文解析木は作成されているものとする。
3. 属性の評価式は多項式で表現されるものとする。

また、本研究で扱う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 [2] (Concurrent Read Exclusive Write Parallel Random Access Machine) 上で実現している。(PRAM とは、共有メモリを持つ SIMD 型の並列計算機モデルで、並列アルゴリズムの理論的な解析によく用いられる。なお、CREW は、1つのメモリの番地に同時読み込み可かつ、書き込みは1つのプロセッサのみ可であることを示す。)

2　属性文法と依存グラフ

意味解析において、属性は、構文解析木の各記号(節点)に付随していると考える。各属性は、継承属性(inherited attribute)と合成属性(synthesized attribute)に分けられ[5]、継承属性は解析木の上から下へ、合成属性は解析木の下から上へと属性値が定まっていく属性のことを言う。

属性文法の形式的な定義を以下に示す [4]。属性文法 G は、3つ組 (G_u, A, R) として以下のように定義される。

(G_u: 文脈自由文法, A: 属性, R: 属性の評価式)

1. 文脈自由文法 $G_u = (N, T, P, S)$
   (N : 非終端記号, T : 終端記号, P : 生成規則, S : 開始記号)
2. 各文法記号 $X \in N \cup T$ に互いに素な2つの有限集合 $AS(X)$, $AI(X)$ ($AS(X) \cap AI(X) = \phi$) が付随している。$AS(X)$、$AI(X)$ はそれぞれの合成属性、継承属性の集合である。属性の集合は $A(X) = AS(X) \cup AI(X)$ とする。各記号 X の属性 a を $X.a$ で表す。
3. 各生成規則 $p \in P$ に対し、意味規則の集合 $R(p)$ が付随している。生成規則 p を
   $p : X_0 \rightarrow X_1 \ldots X_{N_p}$ とすると、$R(p)$ の各意味規則 $X_k.a = f(X_1.a_1, \ldots X_{N_p}.a_m)$ の形をしている。

また、構文解析木の節点に対応する属性の依存関係をグラフに表したものを依存グラフという。生成規則 $p : X_0 \rightarrow X_1 \ldots X_{N_p}$ に対する依存グラフ DG_p は、p の属性間の依存関係を表すグラフである。この DG_p は、次のように定義される。

$DG_p = (DV_p, DE_p)$
$DV_p = \{X_k.a | X_k.a \in A(X_k), 0 \leq k \leq N_p\}$
$DE_p = \{(X_i.a, X_j.b) | X_j.b = f(\ldots X_i.a \ldots) \in R(p)\}$

例えば、1つの生成規則
$X_0 \rightarrow X_1\ X_2$
$\{i_0 = f_0(s_0), i_1 = f_1(i_0, s_2),$
$i_2 = f_3(i_0, s_1), s_0 = f_4(i_0, s_1, s_2)\}$
(i は、継承属性、s は、合成属性を表し、非終端記号 X_0, X_1, X_2 はそれぞれ次の属性を持つ。

X_0:i_0,s_0.X_1:i_1,s_1.X_2:i_2,s_2)
に対応する依存グラフは次の図1で示される。

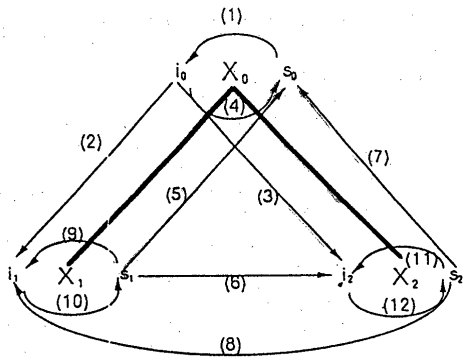

図1　依存グラフ

2.1　属性の種類

属性は、その評価能力によりクラス分けされる。本稿では、評価能力によるクラス分けのうち S[4][5]、L[4][5]、l[4][5]、SNC[4][5] を取り上げ、その性質について述べる。

1. S 属性: S 属性は、合成属性のみからなり構文解析木を葉から根へのボトムアップに処理して、属性値を評価できるクラスである。
2. L 属性: L 属性は、構文解析木上を先がけ順に処理して属性値を評価できるクラスである。
3. l 属性: l 属性は、文法の各記号(終端記号、非終端記号)について、その属性間に線形順序が存在し、解析木の各節の属性値が常にその順序に従って評価できる性質を持つ。
4. SNC 属性: 構文解析木の節点に対応する属性の依存関係をグラフに表したものを依存グラフという。SNC 属性には、依存グラフに閉路がなく、かつ依存グラフに付されている属性がトポロジカルソート [4] の順序で評価できるという性質がある。

また、上記の属性のクラスには、
S ⊂ L ⊂ l ⊂ SNC　の包含関係が成り立つ。

3　属性による意味解析の並列アルゴリズム

本節では、S、L、l、SNC 属性に対して、属性による意味解析の並列アルゴリズムの概要を示す。

3.1　属性が S 属性の場合

S は合成属性のみからなる。依存グラフは、構文解析木の各辺を木の下から上への有向辺で置きかえることで得られ、依存グラフの各節点に対して属性の評価式を割り当てる。属性値の評価の順序は、有向辺の順序関係を保っている。そのため、構文解析木の節点または葉に対して木の高さの低い順に番号を付け、同じ高さの節または葉に対しては、適当に順序をつける。

評価順序がついた後は、straight-line arithmetic program の並列アルゴリズム [2] を適用して全ての属性値を評価する。

Step 1:　構文解析木の各辺を木の下から上への有向辺で置きかえる。
Step 2:　構文解析木の節点または葉に対して木の高さの低い順に番号を付ける。同じ高さの節または葉に対しては、適当に順序をつける。
Step 3:　構文解析木の各節点に対して、属性の評価式を割り当てる。
Step 4:　順序付けられた属性の評価式に、straight-line arithmetic program の並列アルゴリズム [2] を適用して属性式を評価する。

A Parallel Algorithm for Semantic Analysis based on Attribute Grammar.
Hiromitu SHIINA, Shigeru MASUYAMA
Toyohashi Univ. of Tech.

3.2 属性がL属性の場合

はじめに、構文解析木の各辺を双方向の辺で置き換える。これによって依存グラフが得られ、この依存グラフには Euler 閉路が存在する。属性の評価式の割り当ては、依存グラフの各有向辺に対して行なう。属性の評価式の順序づけは、依存グラフを Euler 閉路に対応する巡回隣接リストで表現し、根から出発する先がけ順訪問順序を求める並列アルゴリズム (Euler Tour Technique) [2] を用いて順序づける。順序づけられた属性の評価順序に straight-line arithmetic program の並列アルゴリズム [2] を適用して属性式を評価する。

Step 1: 属性の評価式を割り当てる。構文解析木の辺を互いに反対方向の有向辺で置き換え、依存グラフを作る。この依存グラフは、構文解析木の根から出発する Euler 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辺がある。よって、有向辺は、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辺に1つずつ割り当て、各属性の評価式を以下のようにして有向辺に割り当てる:

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辺には、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辺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辺には、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辺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。

Step 2: 属性の評価式の評価順を決める。属性の評価式の順序づけは、依存グラフを Euler 閉路に対応する巡回隣接リストで表現し、巡回隣接リストに対する根から出発する先がけ順訪問順序を求める並列アルゴリズム (Euler Tour Technique) [2] を用いて順序づける。

Step 3: 順序付けられた属性の評価式に、straight-line arithmetic program の並列アルゴリズム [2] を適用して属性式を評価する。

3.3 属性がl属性の場合

はじめに構文解析木の各節において、l属性の性質より、属性の各評価式に対して、それぞれ次の評価式を並列に求め、属性の評価式のリストを作る。この属性の評価のリストはl属性の定義から一定時間で作成できる。作成したリストをもとに straight-line arithmetic program の並列アルゴリズム [2] を適用して属性値を評価する。

Step 1: 構文解析木の各節において、l属性の性質より、属性の各評価式に対して、それぞれ次の評価式を並列に求め、属性の評価式のリストを作る。

Step 2: 作成したリストをもとに根からの深さ優先探索順に属性の評価式に順序をつける。

Step 3: straight-line arithmetic program の並列アルゴリズム [2] を適用して属性値を評価する。

3.4 属性がSNC属性の場合

属性の評価順に半順序関係がつけられるので、依存グラフに対してトポロジカルソート [4] を実行した結果を評価順とみなすことができる。(依存グラフの各節点には属性の評価式が付されている。) 属性の評価順が求められれば、属性の評価式が1列に並んでいると考えられる。よって、1列に並んでいる属性の評価式に対して straight-line arithmetic program の並列アルゴリズム [2] を適用して全ての属性値が評価できる。先に、依存グラフに対するトポロジカルソートの並列アルゴリズムを示す。

Step 1: 依存グラフのうち有効辺で指されていない節点に対して、新しく1つの節点を付加する。

Step 2: その新しい節点から有効辺で指されていない節点への有効辺を新しく付加する。

Step 3: 新しく有効辺を付加した依存グラフの辺に −1 の重みを付加する。

Step 4: 重みを付加した依存グラフに対して、新しく付加した節点から各節点への Shortest Path [2] を求める。(Shortest Path を求める並列アルゴリズムは参考文献 [2] 参照)

Step 5: Step 4 の結果、同じ重さが付された節点同士は、任意に順序関係を与える。

Step 6: 重みの大きい節点を優先して順番をつける。これがトポロジカルソートで求められる結果である。

SNC属性の並列アルゴリズムを以下に示す。

Step 1: 属性の評価順に半順序関係がつけられるので、依存グラフに対してトポロジカルソート [4] を実行して属性の評価順をつける。

Step 2: straight-line arithmetic program の並列アルゴリズム [2] を適用して属性値を評価する。

4 まとめ

本稿の属性(意味)の評価には、すべて straight-line arithmetic program の並列アルゴリズムを適用している。(式の数を n、式全体の演算子の数を m、最大位数を d とすると、straight-line arithmetic program は、$O(m^3)$ 個のプロセッサ数を用いて $O(\log^2 n + \log n \log d)$ 時間で処理できることが分かっている [2]。straight-line arithmetic program の並列アルゴリズムの部分は、$O(f_p(m))$ 個のプロセッサ数を用いて、$O(f_t(n,d))$ 時間で処理できるとする。)

本稿において、いずれの意味解析でも、入力終端記号数を n、属性の種類を m、各属性の評価式における演算子の数を u、評価式全体の最大位数を d とすると、$O(f_p(nmu))$ 個のプロセッサを用いて $O(f_t(mn,d))$ 時間で実現できる。ただし、各属性のクラスは、評価式の数が異なるため、並列アルゴリズムの実行に要する時間の係数は ($C(x):x$ を属性の種類とした時のアルゴリズムの時間の係数)、

$$C(S) < C(L) < C(l) < C(SNC) \quad \text{となっている。}$$

以上、本研究では属性による意味解析の並列アルゴリズムを示した。これらの並列アルゴリズムは、効率の良い並列アルゴリズムが存在することを明らかにしており、クラス NC に属していることが分かる。今後は、すべての属性の値が一意に決定される属性のクラスである Well-Defined 属性 [4][5] による意味解析の計算量を解明したい。

参考文献

[1] D. Knuth: Semantics of Context-Free Languages, *Math. Syst. Th.*, Vol. 2, No. 2(1968).

[2] A. Gibbons, W. Rytter: *Efficient Parallel Algorithms*, Cambridge University Press(1988).

[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). *J. Inf. Process.*, Vol 8, No.3 (1985).

[5] Pierre Deransart, Martin Jourdan, Bernard Lorho: Attribute Grammars, *Lec. Notes in Comp. Sci.*, Vol. 323, Springer(1988).