拡張ＡＥＬＬ(2)型構文解析手法を用いたインタプリタのインプリメントについて

3U－2

岡田　尚
（株）東洋情報システム

１．はじめに

前回、ELL(2)型構文解析手法を使い構文文法のｶﾀﾛｸﾞ定義方式の簡易言語ｲﾝﾀﾌﾟﾘﾀを開発した[文献１]。

今回、このｲﾝﾀﾌﾟﾘﾀの改良を目的に拡張AELL(2)型構文解析手法の導入を検討した。そのおもな狙いは、「意味制約によるﾊﾟｽ絞り込み」（先読み処理の改善）、「意味制約による構文文法内容の明確化」（文法表記の改善）、「字句解析時、意味ｴﾗｰ･ﾁｪｯｸの同時実行」（解析機能の改善）、「構文木（解析木でない）の直接生成」（処理の改善）である。また、ｲﾝﾀﾌﾟﾘﾀの実行機能として、ｶﾀﾛｸﾞ上での「ｴﾗｰ対処(ﾁｪｯｸ・ﾘｶﾊﾞﾘ)指定」機能の効果を検討した。本稿では、拡張AELL(n)型構文解析手法とそのｲﾝﾀﾌﾟﾘﾀへのｲﾝﾌﾟﾘﾒﾝﾄ効果を報告する。

２．拡張AELL(n)文法とは

拡張LL(n)文法は、再帰下向き構文解析手法を行うための文法（０回以上の繰り返しを含む）である。そして、拡張ELL(n)文法は、拡張LL(n)文法内のパス分岐点でのみ先読み字句解析を行う文法である。

拡張AELL(n)文法では、構文文法において右辺の各記号（終端／非終端記号）に対し、解析前の継承属性と解析後の合成属性に基づいた比較条件式列（ＡＮＤ論理演算）を指定でき、先読み字句解析時、字句解析以外にＬ属性［文献２］の更新・生成が同時実行され、先読み字句のｼﾝﾀｯｸｽ、及び、Ｌ属性条件式の真偽判定によるパス選択が実行される（表１）。

３．拡張AELL(n)構文解析手法による解析実行機能の改善

ﾌﾟﾛｸﾞﾗﾐﾝｸﾞ言語の意味には、静的意味(static semantics)と動的意味(dynamic semantics)があり、以下ではそれを区別して述べる。

まず、Ｌ属性条件式を正規表現内に組み込むことの字句解析上の主な効果をあげる。

(1) 解析能力の改善：字句解析時、静的意味ｴﾗｰ･ﾁｪｯｸを同時実行できる。
(2) 先読み処理の改善：意味制約による構文ﾊﾟｽの絞り込みができる。
(3) ｏｒ文法表記の改善：意味区分による構文文法の分岐表現の指定ができる。
(4) 繰り返し文法表記の改善：繰り返し回数を属性値により制御できる。

a) 左回帰的文法表現の禁止（⊠：Ａ → Ａ ｕ）

b) 構文規則の右側に現れる正規表現(ｏｒ表現)「$e_1 | e_2 | \cdots | e_n$」に対して

$$\forall i,j \quad (i \neq j)$$
$$\mathrm{FIRST\_n\_terminal\&attr}(e_i) \cap \mathrm{FIRST\_n\_terminal\&attr}(e_j) = \phi$$

c) 構文規則の右側に現れる正規表現（繰り返し表現）「｛ｅ｝」に対し

$$\mathrm{FIRST\_n\_terminal\&attr}(e) \cap \mathrm{FOLLOW\_n\_terminal\&attr}(\{e\}) = \phi$$

d) 属性（合成属性・継承属性）は、Ｌ属性(解析を上から下、左から右への１回の解析で属性評価できるもの)であり、属性の依存関係に「右→左」のものが存在しない。

注）記号の説明

正規右辺文法 Ｇ＝（ＶＮ，ＶＴ，Ｓ，Ｐ）で、

$$A \in R_{IA}\ VN\ R_{SA},\ V_{AELL} = R_{IA}\ (VT \cup VN)\ R_{SA},\ u \in V^{*}_{AELL}$$

R_{IA} ：記号(終端／非終端)の継承属性条件式の論理積 $\prod \mathrm{cond\_exp}_{IA,k}$
R_{SA} ：記号(終端／非終端)の合成属性条件式の論理積 $\prod \mathrm{cond\_exp}_{SA,k}$
$\prod_j R_{IA,1,j}$ ：（ｉ－１）番目目の先読み終端記号とｉ番目の先読み終端記号との間にある非終端記号（パス分岐点を起点とする）の継承属性条件式とｉ番目の先読み終端記号の継承属性条件式との論理積
$\prod_j R_{SA,1,j}$ ：（ｉ－１）番目目の先読み終端記号とｉ番目の先読み終端記号との間にある非終端記号（パス分岐点を起点とする）の合成属性条件式とｉ番目の先読み終端記号の合成属性条件式との論理積

e_1, e_j, e ：V_{AELL}の正規表現（連結表現・ｏｒ表現・繰り返し表現の入れ子表現）
｛ｅ｝ ：正規表現で、０個以上の繰り返しを表現する．（拡張文法であるため）

FIRST_n_terminal&attr () ：空記号（ε）を含まない実終端記号の最大ｎ個とその前後の属性条件式の並び

$$\prod_j R_{IA,1,j}\ V_{T1}\ \prod_j R_{SA,1,j}\ \prod_j R_{IA,2,j}\ V_{T2}\ \prod_j R_{SA,2,j}\ \cdots$$

FOLLOW_n_terminal&attr () ：空記号（ε）を含まない実終端記号の最大ｎ＋１個の先頭の１個の実終端記号とその前の並びから、継承属性の条件式とその後ろの合成条件式を除いたもの

$$\prod_j R_{IA,2,j}\ V_{T2}\ \prod_j R_{SA,2,j}\ \prod_j R_{IA,3,j}\ V_{T3}\ \prod_j R_{SA,3,j}\ \cdots$$

表１．拡張 AELL(n) 文法の説明

その他、属性を使用することの解析後の効果をあげる。

(5) 構文木（解析木ではない）を直接生成できる。
(6) ｶﾀﾛｸﾞ文法定義方式のｲﾝﾀﾌﾟﾘﾀに利用する場合、動的意味ｴﾗｰ･ﾁｪｯｸをｶﾀﾛｸﾞに組み込むことが可能になる。

また、構文文法定義からパーザーを自動生成する場合の効果をあげる。

(7) Ｌ属性条件式による意味制約は、先読み字句数の上限ｵｰﾊﾞｰによる構文文法の手直しの発生を抑止する上、手直し作業を容易にする。

A Study of Intepreter By Attribute-directed Extended LL(2) Grammar
Takashi Okada
TOYO INFORMATION SYSTEMS CO,LTD.

4．拡張AELL(n)文法による言語表現の特徴

L属性を使って、「左－＞右」方向でｼﾝﾀｯｸｽを特定するL属性条件式、及び、字句を言語文法で指定できることが分かる。その例として、通常のﾌﾟﾛｸﾞﾗﾑ言語の終端記号の種別(関数・変数)による構文ﾊﾞｽの識別[文献４]がある．ここでは、可変長ﾃﾞｰﾀ項目と可変個のﾚｺｰﾄﾞ数を持つﾚｺｰﾄﾞ・ﾌｫｰﾏｯﾄ解析を例に、その他の特徴を示す（表２）。

【L属性が字句解析処理を制御する例】

可変長ﾃﾞｰﾀ項目指定：＝　NUMBER　↓clm_limit = ↑value　IDENT　↑clm_value = ↑value　↓clm_limit = -1

【L属性が構文内の繰り返し回数を制御する例】

可変個のﾚｺｰﾄﾞ指定：＝　NUMBER　↓rec_num = ↑value

0（（↓rec_num＞0）IDENT, ・・・）∞　↑clm_value = ↑value　↓rec_num = ↓rec_num - 1

注）「↑属性名」：合成属性，「↓属性名」：継承属性
注）終端記号「IDENT」の字句解析時、継承属性「↓clm_limit」が0以上の場合、それを字句ﾃﾞｰﾀの最大byte長として処理するものとする．
注）終端記号「NUMBER」は、整数値を字句ﾃﾞｰﾀとする終端記号を示す．

表２．レコード・フォーマットの構文文法例

5．拡張AELL(n)解析手法のｲﾝﾀﾌﾟﾘﾀへのｲﾝﾌﾟﾘﾒﾝﾄ効果

第３節から、L属性文法定義を構文文法定義に導入すると、ｲﾝﾀﾌﾟﾘﾀは「構文木の直接生成とその部分木実行」・「実行前(解析中・解析後)の意味ｴﾗｰﾁｪｯｸの組み込み」を可能にし、それらが言語の実行直前までのｲﾝﾀﾌﾟﾘﾀの処理効率・ﾒﾓﾘ効率・ｴﾗｰﾁｪｯｸ内容の改善をもたらす。

ここでは、言語文法の定義をｶﾀﾛｸﾞで記述することを特徴とするｲﾝﾀﾌﾟﾘﾀのｶﾀﾛｸﾞ定義機能を問題にする。ｶﾀﾛｸﾞ内の言語文法定義にL属性を使用することで、図１に示すようにｶﾀﾛｸﾞの定義内容を拡張できる。この拡張したｶﾀﾛｸﾞ定義では、言語の非文脈自由なｴﾗｰ［文献３］を静的意味ｴﾗｰと動的意味ｴﾗｰに区分して取り扱った（表３）。

このｶﾀﾛｸﾞ内のｴﾗｰ指定機能から、拡張AELL(n)解析手法のｶﾀﾛｸﾞ式ｲﾝﾀﾌﾟﾘﾀへのｲﾝﾌﾟﾘﾒﾝﾄ効果を考えると、ｲﾝﾀﾌﾟﾘﾀのｴﾗｰ処理の内、ｼｽﾃﾑ・ｴﾗｰ以外のすべてのｴﾗｰ対処(ﾁｪｯｸ・ﾘｶﾊﾞﾘ）の指定がｶﾀﾛｸﾞで定義できることが分かる(図２)．また、その静的/動的意味ｴﾗｰ指定が、実行処理単位ではなく構文木（or 部分木）単位に指定できるため、ｴﾗｰの指定をより共通化・部品化でき、それが意味ｴﾗｰﾁｪｯｸ機能の品質を均一化し、また、変更や拡張を容易にする。

6．まとめ

以上で、拡張AELL(n)型構文解析手法とそのｲﾝﾀﾌﾟﾘﾀへのｲﾝﾌﾟﾘﾒﾝﾄ効果、特にｴﾗｰ処理のｶﾀﾛｸﾞ定義による実現についての報告を終える。品質の高いｲﾝﾀﾌﾟﾘﾀは、処理対象のﾌﾟﾛｸﾞﾗﾑ、および、ｼｽﾃﾑで発生するほとんどの手続き的不具合を検出し処理(ﾘｶﾊﾞﾘ)できなければならない。また、この機能の実現が、ｲﾝﾀﾌﾟﾘﾀの高い実行信頼性と非手続き的言語構築を容易にする。

今後は、拡張AELL(2)型構文解析手法を導入したｲﾝﾀﾌﾟﾘﾀを実現し、属性を用いた言語構築機能とその処理系の効果的側面を追求していきたい。

No	エラー種別	説　明
1	ｼｽﾃﾑ(system)・ｴﾗｰ	翻訳できるﾌﾟﾛｸﾞﾗﾑ・ｻｲｽﾞやﾒﾓﾘ・ｻｲｽﾞのｵｰﾊﾞｰなど、処理の限界を越えたｴﾗｰ
2	構文(syntax)・ｴﾗｰ	字句・ｶｯｺ・ｶﾝﾏなど構文文法の記述ﾌｫｰﾏｯﾄに違反したｴﾗｰ
3	静的意味(static semantic)・ｴﾗｰ	ﾌﾟﾛｸﾞﾗﾑ記述内での変数・関数などの使用方法の誤りｴﾗｰ（定義有無、型・修飾の不具合、関数の戻り値や引数の名前・型の不具合..etc）
4	動的意味(dynamic semantic)・ｴﾗｰ	ｱﾄﾞﾚｽ指定ｴﾗｰ、ｱｸｾｽ・ｴﾗｰ、ﾃﾞｰﾀ処理ｴﾗｰ(数値over/underflow・文字列長の溢れ・NIL値..etc)など、実行時の物理的ｴﾗｰ（関数の実行順序ｴﾗｰなど、実行時の間接的ｴﾗｰを含む）
5	ﾕｰｻﾞ(user)・ｴﾗｰ	ユーザが意図に反した実行状況・実行結果を想定したﾕｰｻﾞ固有のｴﾗｰで、処理系ではｴﾗｰにならない

表３．ﾌﾟﾛｸﾞﾗﾑ解析実行時に発生するｴﾗｰの分類

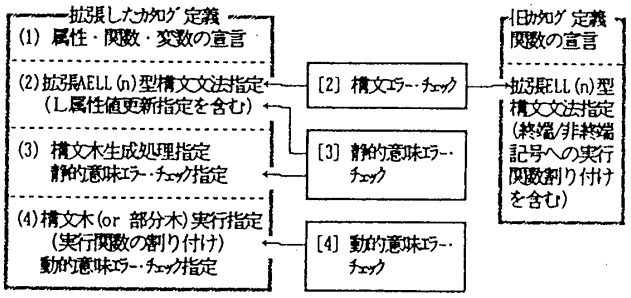

図１．ｶﾀﾛｸﾞ定義内容とｴﾗｰ・ﾁｪｯｸ機能との関係図

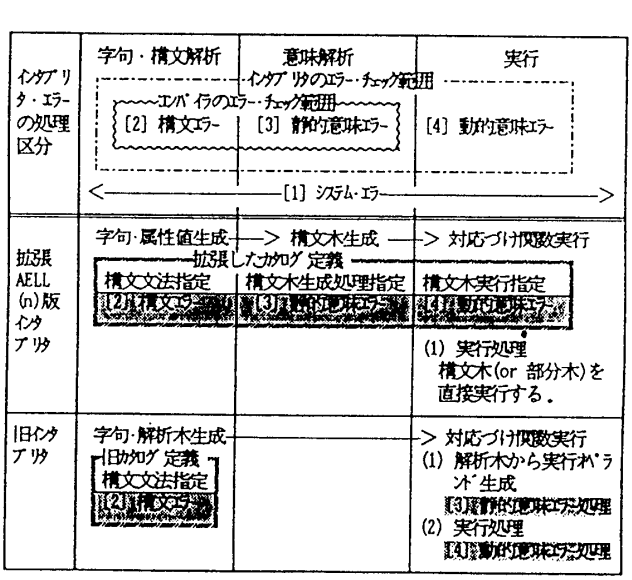

図２．ｲﾝﾀﾌﾟﾘﾀの処理過程とｴﾗｰ処理との関係図

【参考文献】

[1] 岡田　尚："「ﾛｰﾀｽ 123」・「一太郎」文書ﾌｧｲﾙの集計用4GLﾂｰﾙ(第１版)" 第48回情報処理学会全国大会論文集
[2] 佐々政孝：属性文法－ﾁｭｰﾄﾘｱﾙ，ｺﾝﾋﾟｭｰﾀｿﾌﾄｳｴｱ，Vol3 No.4 Oct.1986 P 377- P 395
[3] R．ﾊﾝﾀｰ：『ｺﾝﾊﾟｲﾗ構成論』，近代科学社，P232-P248
[4] 大木康幸、他："属性値主導型 拡張LL(1)文法の提案" 第48回情報処理学会全国大会論文集，1994.3