3F-3

プログラミング言語の静的意味解析
Pascalの型の解析を例に

武藤望，武市正人，小林光夫
電気通信大学

1．はじめに

プログラミング言語の意味解析を文脈自由構文解析に寄生させる処理モデルは数多く提唱されている（たとえば attribute grammar [2],[4], affix grammar [5], production system [6]）が，本稿では Pascal [3] を例に，①宣言・定義による構文規則の生成，②順位文法類似の属性評価規則系，を基礎とする処理のモデルを示す．

2．宣言による構文規則の生成．

現在，プログラミング言語の構文(syntax)は普通，文脈自由言語としての規定を指し，バッカス記法(BNF)などにより記述されている．そして，たとえば識別子(identifier)の使用に対する宣言の必要性は，いわゆる文脈条件として自然言語で付記されている．本稿では，この文脈条件を以下のように取扱う．

```
var x:real;          | VAR_DECL ::= VAR_ID ':' TYPE | VAR_ID ::= ID    (c-1)
begin x:=0.1;...     | VARIABLE ::= VAR_ID | ...    | VAR_ID ::= 'x'   (c-2)
     (a)                        (b)                          (c)
```

図1　宣言と対応する構文規則．

図1 (a),(b) は Pascal のプログラムの断片と，対応する構文規則の部分である．ここで宣言に対して (c-2) の規則を対応させることにより，(c-2) を (b) に加えた規則系（有効規則系と呼ぶ）は，宣言の有効範囲内での構文を従来の (c-1) を用いるより正確に規定することができる．さらに，

```
REAL_VAR_ID ::= 'x'
```

のように識別子に属性，たとえば型を結合することができる．

3．属性付演算子順位文法．

プログラム中の構成，たとえば式について種々の属性を評価することが意味解析の主要目的である．上向き(bottom up)に評価できる，型のような属性については，以下のような評価規則系が適用できる．図2は Pascal の式の型に対する例（部分）である．

```
INTEGER ::= INTEGER_VARIABLE
          | INTEGER '*' INTEGER
          | ...

REAL    ::= REAL_VARIABLE
          | REAL '*' REAL
          | INTEGER '*' REAL
          | ...
```

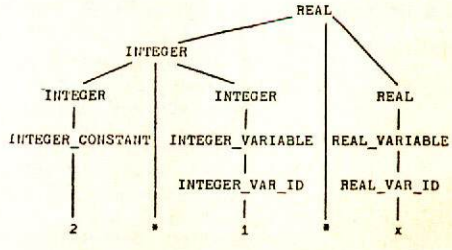

図2　型評価文法と解析木の例．

よく知られた演算子順位文法と同様の方法[1]により，この規則系にて構文解析と同時に型の評価が可能である．

4．宣言・定義による評価則生成．

新たな型を構成するなどの機能を持つ言語（たとえばPascal）では，宣言・定義により属性の基数が増える場合がある．これに対しては1，2で述べた方法を組合わせることにする．図3はPascalにおける型の定義とそれに対応する評価規則の例である．一般に型の定義には型の構成で自然に導入される演算（たとえば構造型については，いわゆる選択子（selector）による選択）の規則を対応させる．

```
var v:(* T= *)          T ::= 'v'
  record
    a:integer;          INTEGER ::= T '.' 'a'
    b:Boolean           BOOLEAN ::= T '.' 'b'
  end
    (a)                          (b)
```

図3　型の定義と対応する規則

5．宣言の有効範囲．

前節までで宣言の有効性を，対応する規則の有効性で置き換える方法を述べた．この方法では宣言の有効範囲の開閉に応じて有効規則が増減することになるが，規則に範囲を表わす特殊な非終端記号（とその規則系）を導入して修飾するという拡張が考えられる．これは通常のblock構造だけでなく，より一般の可視性制御（visibility-control）にも適用可能な柔軟性を持つが紙面の関係上詳しくは述べない．

```
R1  var a,b:real;                 REAL ::= R1 'a'
    R2  var a:integer;                  | R1 'b'
            begin a:=1;b:=0.2;    INTEGER ::= R2 'a'
        begin a:=0.1; ...
                                  /* "scope" rule */
                                  R1 ::= r1
                                       | R2
                                  R2 ::= r2
```

図4　修飾付規則の例．

6．まとめ

有効規則系の考え方をそのまま実現すれば，宣言を解析することによって規則を生成し，その時点で解析プログラム（parser）を作成することになる．このparserにより，いわゆる静的意味解析（static semantic analysis）が構文解析と同時に可能となる．その際には，parserを作成するに要する時間が問題となるが，たとえば編集機構をも組入れて宣言部を分離して処理する部分（逐次）翻訳の形での処理を行なえば，作成時間を他の編集処理中に吸収することができる．また，拡張可能言語の処理系にこの手法を応用すると核部分と拡張部分とを統一して扱うことができる．

発表者武市の研究の一部は昭和55年度文部省科学研究費補助金奨励研究（A）57523による．

[1] Aho, A.,V., Johnson, S.,C. and Ullman, J.,D.: Deterministic parsing of ambiguous grammars, Comm. ACM Vol. 18,8(1975),441-452.
[2] Bochman, G.,V.: Semantic evaluation from left to right, Comm. ACM Vol. 19,2(1976),55-62.
[3] Jensen, K., Wirth, N.: PASCAL User Manual and Report, 2nd ed., Springer-Verlag(1978).
[4] Knuth, D.,E.: Semantics of Context-free languages, Math. Systems Thory Vol. 2,2(1968),127-245.
[5] Koster, C.,H.,A.: Using the CDL compiler-compiler, in Compiler Construction an Advanced course(F.L.Bauer and J.Eickel ed.), 2nd ed., Springer-Verlag(1976).
[6] Ledgard, H.,F.: Production systems: Or can we do better than BNF?, Comm. ACM Vol. 17,2(1974),94-102.