

3L-9

パソコン通信を利用したソフトウェアの共同開発
—プログラミング言語Oscalの設計と実現—山之上 卓
九州工業大学紀太 章
コニカ相沢雅陽
富士通エフ・アイ・ビー

1. まえがき

近年、パーソナルコンピュータやワードプロセッサがいたるところで使われている。このパソコンやワープロを利用したパソコン通信が最近注目されており、現在数多くのパソコン通信ホストシステムが開局し、数多くのパソコンやワープロのユーザーが利用している[1]。このパソコン通信を利用して、原稿執筆時点でまだ一度も会ったことのない我々は、新しいプログラミング言語Oscalの設計を共同で行い、その処理系を実現することができた。(我々は物理的にも離れた位置で暮らしている。)

Oscalは、コンパイラはなぜプログラマが与えたインデントという2次元情報を無視してしまうのか、という不満から誕生した。誰でもプログラムの構造を把握する手段としてインデントを利用していると思われる。その視覚による情報と、構文との間に差がでくると、なかなか根深いバグの元になる。そこでOscalはインデント=構文という構文規則をもって、視覚と構文を一致させた。「Oscal」は、プログラムの構造(の一部)をインデントで表す言語「Occam」と、従来の手続き型言語「Pascal」の2つを合成してできた名前である。

Oscalは、インデントによってプログラムの構造を表している。このためBNFのように文脈自由文法と同じ能力を持つ記法でOscalの文法を定義することは非常に難しい。この問題を解決するために、我々は属性文法を採用したコンパイラコンパイラMYLANGを利用した[2,3]。

Oscalの開発過程の記録は、パソコン通信ホストシステムにすべて記録されており、このことも開発の上で、非常に役立った。Oscalの開発期間は、発案(1988.3.末)から処理系の実現(1988.5.末)まで約2カ月であった。

なおここで開発された処理系は、Oscalプログラムを、Pascalに変換するプリプロセッサである。

2. Oscalの開発過程

我々は、全国主要都市にアクセスポイントが設置されている全国規模のパソコン通信サービスの会員である[4]。このパソコン通信サービスの中には、100近いフォーラム(同好会)がある。1987年の末に、プログラミング言語フォーラム(以後FPL)という名前のフォーラムを相沢が開設した。FPLは種々のプロ

グラミング言語およびその処理系全般にわたって、情報交換をしたり議論しようというもので、原稿執筆時点で500人を越える会員がいる。紀太と山之上はFPLの会員であり、我々はFPLに書き込みをしているうちにたまたま知り合った。プログラミング言語Oscalは、我々が雑談をしているうちに、「こんな言語を作ったらどうか?」という話題からできた言語である。

以下にOscalの開発過程のタイムテーブルを示す。

- 3/26 紀太と相沢がプログラミング言語のインデントや、Occamの話題を始める。
- 3/28 紀太がOscalを命名、プロジェクトOscal発足
- 3/28 山之上がプロジェクトに参加を表明、MYLANGを持ち込む。
- 3/29 MYLANGの紹介や質問
- 3/31 プロジェクトOscal専用の会議室を相沢が開設。紀太が進行役に就任
- 4/1 Oscalのイメージの再確認
 - (1)インデントの深さで構造を指定。
 - (2)処理系はPascalへのトランスレータ
- 4/3 MYLANGの説明開始
- 4/4 Oscal構文の議論開始(我々以外の会員も参加)
- 4/13 相沢がOscalとPascalの比較表を提示
- 4/18 紀太がOscal文法の形式的定義の草案を作成
- 4/19 山之上がOscalのIF文をPascalのif文に変換するプログラムを作成。
- 4/19 紀太がOscalによる8-Queenプログラム例を作成
- 4/20 Oscalの構文に関する、より深い議論開始
- 5/1 構文修正案の中間集約(紀太)
- 5/1~5/8 ゴールデンウィーク
- 5/8 Oscalの改良文法草案(紀太)
- 5/12 Oscalトランスレータ第0版作成(山之上)
- 5/17 情報処理学会全国大会での発表の提案(山之上)
- 5/25 Oscalトランスレータ第0.1版作成
- 5/25 Oscalトランスレータ第0版バグ報告
- 5/27 Oscalの評価(紀太)
- 5/28 Oscalトランスレータ第0.2版作成
- 5/30 Oscalの評価とトランスレータへの要望(相沢)
- 5/31 Oscalトランスレータ第0.3版作成

3. Oscalプログラムの例

以下にエラトステネスのふるいによって素数を求めるOscalプログラムの例を示す。

```

OSCAL sieve
CONST
  size = 8190
VAR
  flags : ARRAY [0..size] OF boolean
  i, prime, k, count, iter : integer
BEGIN sieve
  writeln('10 iteration')
  FOR iter := 1 TO 10
    count := 0
    FOR i := size DOWNT0 0
      flags[i] := true
    FOR i := 0 TO size
      IF flags[i]
        prime := i + i + 3
        k := i + prime
        WHILE k <= size
          flags[k] := false
          k := k + prime
        count := count + 1
  writeln(count, ' primes')

```

4. MYLANG

属性文法はコンパイラ記述システムにおいて、現在もっとも有望な道具である。MYLANGは、属性文法を採用したコンパイラコンパイラの一つであり、これまでに、LISP, PROLOGなどのインタープリタ、日本語プログラミング言語NBSG/PPDのプリプロセッサ、簡略化された中国語文の構文解析システムなど、種々の言語処理系が、MYLANGを用いることによって実現されている。

MYLANGは、その入力記法(RTFと呼ばれる)で記述された言語処理系の定義から、目的の言語処理系を自動的に生成するものである。

MYLANGの一つの特徴は、属性文法で用いる意味関数の記述記法が、構文の記述記法と一致していることである。従ってMYLANGのユーザーは、ただ1つの記法を習得することによってMYLANGを使いこなせるようになる。(YACCの場合では、字句解析部生成系Lexの入力記法である正規表現、構文解析部生成系YACCの入力記法であるBNF、意味解析部でC言語の合計3つの記法が必要となる。)

なおRTFは副作用を持つため、厳密な意味での属性文法ではない。

5. Oscalトランスレータ

以下に、RTFで記述されたOscalトランスレータの一部を示す。

```

<節(/n)>=<ws('begin'/)><wlf>
  <最初の文(n/m)><残りの文(m/)>*
  <ws('end'/)><wlf>;
<最初の文(n/m)>=
  <gbp(/b)><f_indent(n/m)>
  (<文(m/)>+else[.back(b/)]<fail>);
<残りの文(m/)>=
  <gbp(/b)><indent(m/)>
  (<文(m/)>+else[.back(b/)]<fail>);

```

ここで<ws('begin'/)>はbeginを、<ws('end'/)>はendをそれぞれ出力することを表す。<wlf>は改行の出力を表す。<最初の文(n/m)>の定義の中の<f_indent(n/m)>は、節の最初の行の文のインデントが、1つ階層が上にある節のインデントnよりも深いことをチェックし、この文のインデントの深さをmに与える。<残りの文(m/)>の定義の中の<indent(m/)>は、節の残りの文のインデントの深さが、最初の文のインデントの深さと同じかどうかのチェックする。<gbp(/b)>は、入力バッファへのポインタをbに与える。[.back(b/)]は、入力バッファへのポインタをbに戻す。

6. Oscalの評価

実際にOscalを使用してみた感想を示す。

長所:

- 煩わしいBegin-Endやif-then-else時のセミコロンの問題がなくなりコーディングが楽。
- 特にif-then-elseはすっきりする。
- Pascalではインデントを統一しにくかったCASEなどが一定のフォーマットとなり精神衛生に良い。
- キーワードが簡潔になり、Syntactic noiseがないため見通しがよい。

短所:

- 慣れないせいか、やはりブロックの閉じ記号がないと何かもの足りない。
- インデントをきちんとしないといけなためデバッグ行を挿入する場合や、一時的にプログラムの一部分を殺したいときなどは大変。
- 長いプログラムはインデントを合わせるのが難しい。
- やはりトランスレータ方式はコンパイルに時間がかかり大変。

7. あとがき

本稿は、プログラミング言語Oscalの開発を通して、パソコン通信を利用したソフトウェアの共同開発が、どのようなものであるかということを紹介した。より詳しい開発記録はFPLに保存されている。今回開発したOscalトランスレータはNIFTY-ServeのFPLにPDSとして登録されている。また、MYLANGもPDSへの登録を準備中である。

最後にプロジェクトOscalに参加頂いたFPLの会員の皆様と、本稿の発表に快く同意していただいたNIFTY-Serve事務局に感謝の意を表します。

参考文献

- [1]マイコンBASICマガジン編集部/ツールボックス/デスクトップ編: パソコンBBS電話帳、電波新聞社(1988)。
- [2]山之上、安在: 属性付構文指示翻訳系の生成系MYLANG、情報処理学会論文誌、Vol.26, No.1, pp.195-204 (1985)。
- [3]山之上、安在、吉田、杉尾、武内、椎野: 言語処理系の生成系MYLANGによるNBSG/PPDプリコンパイラの試作、情報処理学会論文誌、Vol.28, No.1, pp.64-73(1985)。
- [4]武井一己: NIFTY-Serve徹底活用マニュアル、HBJ出版局(1988)。