OMetaのための
衛生的マクロ定義機構導入方式

星野 友宏・高桑 健太郎・渡部 卓雄
東京工業大学

概要

本研究では、プログラミング言語OMetaを用いて実装された言語処理系に対して、元の言語処理系にはできるだけ手を入れずに衛生的マクロを定義機構を導入する手法を提案する。

OMetaは、パターンマッチングのためのオブジェクト指向のプログラミング言語である。PEG(Parsing Expression Grammer)を基礎とする構文記述や、オブジェクト指向の特徴である継承やオーバーライド、親文法のルール呼び出しなどをサポートしており、言語処理系の実装に適している。

OMeta/JSのソースコード→

```
ometa Calc {
  digit = ^digit:d -> d.digitValue(),
  number   = number:n digit:d
               -> (n * 10 + d)
           | digit,
  addExpr  = addExpr:x '+' mulExpr:y
               -> (x + y)
           | addExpr:x '-' mulExpr:y
               -> (x - y)
           | mulExpr,
  mulExpr  = mulExpr:x '*' primExpr:y
               -> (x * y)
           | mulExpr:x '/' primExpr:y
               -> (x / y)
           | primExpr,
  primExpr = '(' expr:x ')' -> x
           | number,
  expr     = addExpr
}
```

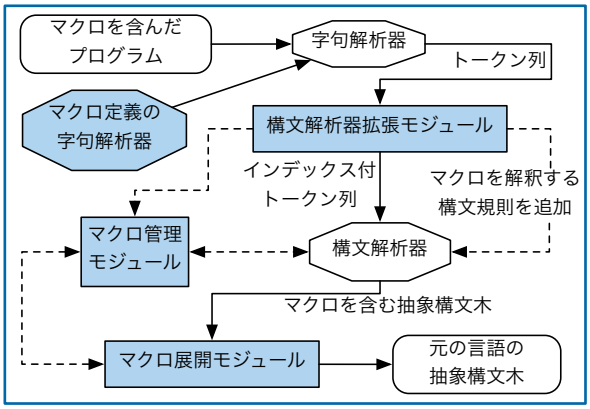
OMetaMacro

本手法に基づいて実装されたマクロ機構は、以下のような機構から構成される。

- マクロ定義を解析するための字句解析器
- 新たな規則を構文解析器に追加する構文解析器拡張モジュール
- マクロの情報を管理するモジュール
- マクロ展開モジュール

これらを既存の言語処理系と組み合わせてできるマクロ定義機構付き言語処理系の構造は右の図のようになる。構文解析器拡張モジュールにより、元の構文解析器がマクロによる新しい構文を解析できるように拡張され、マクロ展開モジュールによってマクロの衛生的な展開が行われる。

OMetaMacroを使った処理系の構造図→



例: swapマクロ

マクロを含んだプログラム

```
MACRO_FROM
swap ( Identifier:lhs , Identifier:rhs ) ;
MACRO_TO
{
    var temp = @lhs@;
    @lhs@ = @rhs@;
    @rhs@ = temp;
}
MACRO_END
var temp = 2;
var other = 3;

swap(temp, other);

print(temp);
print(other);
```

マクロ展開後のプログラム

```
var temp = 2;
var other = 3;

{
    var temp__at_5 = temp;
    temp = other;
    other = temp__at_5;
}

print(temp);
print(other);
```

マクロは、利用されている箇所の環境に応じて、適切に内部の識別子が変更され、衛生的に展開される。 マクロ付き抽象構文木を展開すると、元の言語の抽象構文木が生成される。