言語内DSLに名前束縛をもたらす

名前束縛をローカルな文法拡張とみなす

```
let a = 10 in a * a
```

"a" を
in の後の式に導入

```
var x: Int = 0
```

"x" 及び "x = _" を
このブロックに導入

　名前束縛はプログラミング言語の基本的な機能であるため、ユーザによる定義ができなかった(※1)。そのため、言語内DSLではクロージャや変数宣言のようなホスト言語の機能による名前束縛を使わざるを得なかった。
　我々は、名前束縛を「ローカルな文法拡張の導入」とみなすことでユーザ定義を可能にした。

(※1) マクロを利用すれば既存の名前束縛の文法を変えたものであれば実現可能だが、例えば代入式が使える/使えないといった既存の名前束縛の規則は変えられない。

ローカルな文法拡張

　ローカルな名前束縛を表現するために、我々は *context* というモジュール機構を提案する。contextはクラスに似たモジュール機構で、文法拡張（ここでは演算子定義）を持つ。これらの文法拡張は "そのcontext の下" でのみ有効となる。
　引数の型 C ~> T は context C の下で T 型を返す式という意味を表す。そのため、右で定義した let 式を利用して上で示した式を記述できる。C ~> T 型はメソッドボディでは関数型 C => T となる。

```
context Local<T> {
  private T v;
  Local (T v) { this.v = v; }
  T "a" () { return v; }
}
```

```
<R> R "let" "a" "=" _ "in" _
  (Int v, Local<Int> ~> R f) {
  return f.apply(new Local<>(v));
}
```

DSLユーザによる名前の指定

```
dsl LetDSL {
  <T, R, name: Id> R
   "let" name ":" T "=" _ "in" _
   (T v, Local<T, name> ~> R r) {
    return r.apply(
              new Local<>(v));
  }

  context Local <T, name: Id> {
    private T v;
    Local (T v) { this.v = v; }
    T name () { return v; }
  }
}
```

　let式のような名前束縛では、名前はDSLユーザにより与えられる。これは文法の一部を利用時に決定するということであるとみなせる。
　我々はこれに対応するために、ジェネリクスを拡張した*メタ引数*を導入した。メタ引数の実引数は構文木である。DSLユーザからの入力をこのメタ引数に束縛し、文法拡張の利用時に構文木の同一性を見ることで、これを可能にしている。
　本研究で重要な点は、DSL開発者はメタプログラミングをしていないという点である。これにより、コンパイラの停止性やDSLのコンポーザビリティを保証できる。

東京大学 市川 和央、千葉 滋

github: csg-tokyo/proteaj2
mail　: ichikawa@csg.ci.i.u-tokyo.ac.jp

東京大学
THE UNIVERSITY OF TOKYO