BiYacc: Roll Your Parser and Reflective Printer into One

Zirun Zhu[1,2], Yongzhe Zhang[1,2], Hsiang-Shang Ko[2], Pedro Martins[3], João Saraiva[4], Zhenjiang Hu[1,2]

[1] SOKENDAI (The Graduate University for Advanced Studies), Japan
[2] National Institute of Informatics, Japan [3] University of California, Irvine, U.S.A. [4] University of Minho, Portugal

BiYacc Architecture

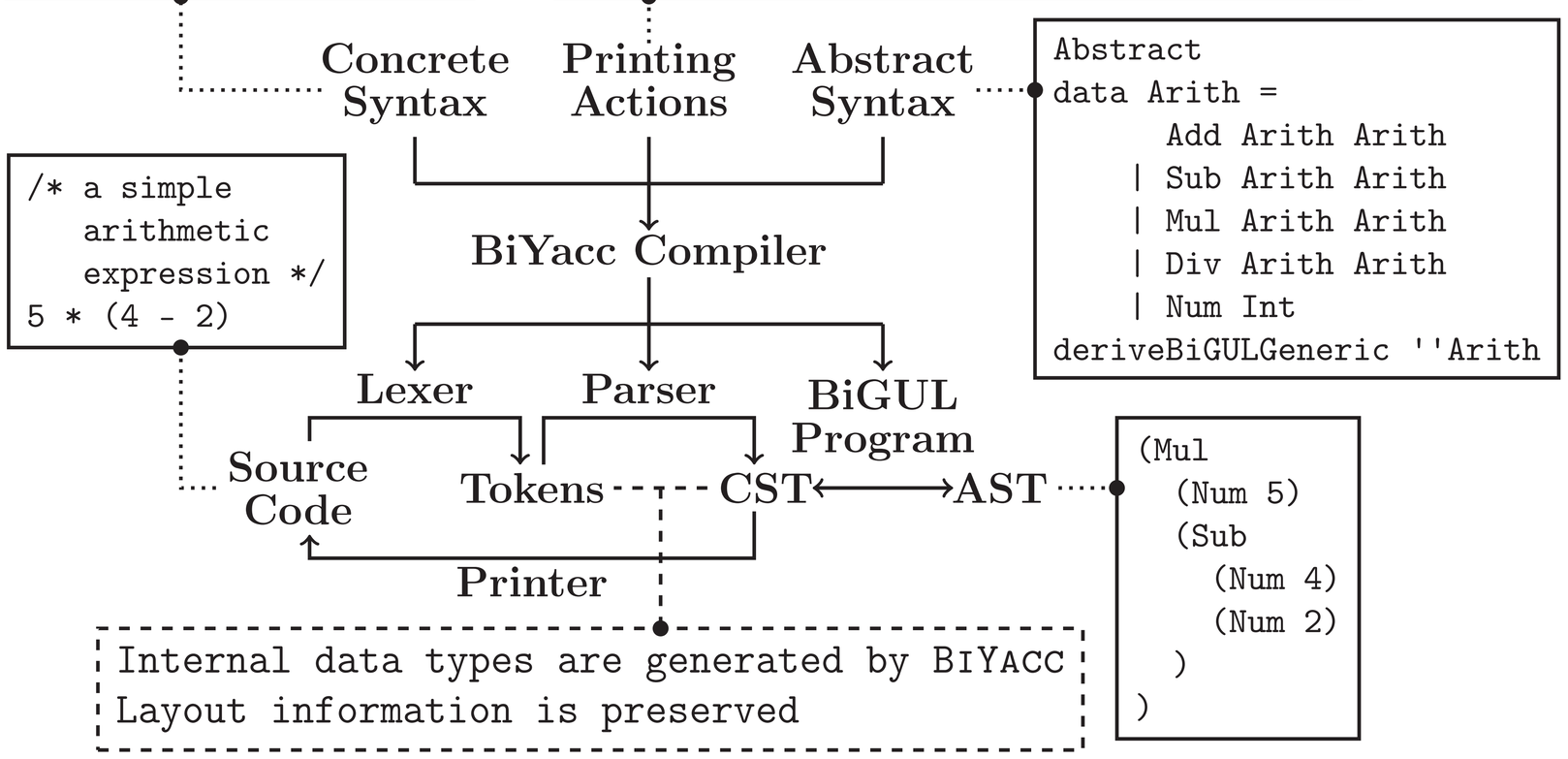
```
Concrete

Expr -> Expr '+' Term
      | Expr '-' Term
      | Term;

Term -> Term '*' Factor
      | Term '/' Factor
      | Factor;

Factor -> Int
      | '-' Factor
      | '(' Expr ')';
```

```
Actions
Arith +> Expr
Add x y +> (x +> Expr) '+' (y +> Term);
Sub x y +> (x +> Expr) '-' (y +> Term);
exp     +> (exp +> Term);
Arith +> Term
Mul x y +> (x +> Term) '*' (y +> Factor);
Div x y +> (x +> Term) '/' (y +> Factor);
exp     +> (exp +> Factor);
Arith +> Factor
Sub (Num #0) y +> '-' (y +> Factor);
Num i          +> (i +> Int);
exp            +> '(' (exp +> Expr) ')';
```




```
Abstract
data Arith =
       Add Arith Arith
     | Sub Arith Arith
     | Mul Arith Arith
     | Div Arith Arith
     | Num Int
deriveBiGULGeneric ''Arith
```

```
/* a simple
   arithmetic
   expression */
5 * (4 - 2)
```

```
(Mul
  (Num 5)
  (Sub
    (Num 4)
    (Num 2)
  )
)
```

Features

Bidirectional transformations: the forward and backward transformations between source code s and abstract syntax tree t are well-behaved.

$$print\ s\ (parse\ s) = s$$
$$parse\ (print\ s\ t) = t$$

Updating actions: unlike conventional printers, BiYacc does not print a new program from scratch, but updates the original program text with the AST, possibly preserving some syntactic structures and layout information.

Pattern matching: BiYacc's support of simultaneous pattern matching on both the CST and AST lets the user write flexible printing actions to synchronize abstract and concrete representations, and is especially useful for handling syntactic sugar.

Adaptation: printing actions only need to be written for matching ASTs and CSTs. If the AST and CST are mismatched, BiYacc automatically chooses the first printing action that matches the AST, and prints a CST from scratch like a conventional printer.

Applications

Reflecting optimizations: a variety of optimizations on ASTs (such as constant propagation and dead code elimination) can be correctly reflected to the source code.

Pombrio and Krishnamurthi's resugaring: reduction sequences on ASTs can be reflected back to the surface language.

Simple language extensions: some simple language extensions (such as 'for-each' loops in Java 5) can be implemented by desugaring into existing language constructs (such as 'for' loops).

Refactoring: semantics-preserving code reorganization (such as variable renaming and method extraction) can be performed on ASTs and an equivalent reorganizing transformation can be derived for the source code.

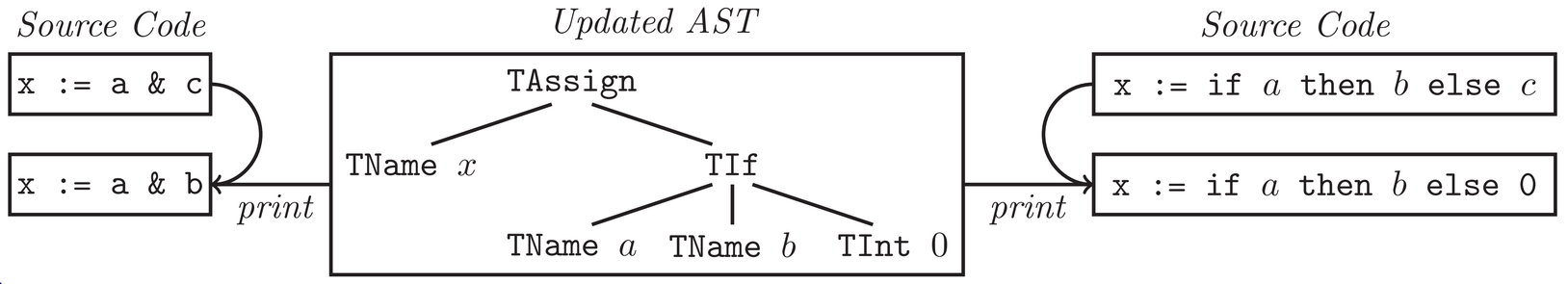
Preservation of Syntactic Sugar

Being a *reflective* printer, BiYacc takes the source code as an input, and prints the updated abstract syntax tree (AST) according to the original layout. Syntactic sugar can be directly handled.

Part of Tiger's BiYacc Language Specification

```
-- AST data type
data Tiger = TInt Int
   | TName Name
   | TAssign Tiger Tiger
   | TIf Tiger Tiger Tiger
   | ...
-- syntactic sugar for 'and' and 'or'
Tiger +> InfixExp
TIf e1 e2 (TInt #0) +> (e1 +> Exp) '&' (e2 +> Exp);
TIf e1 (TInt #1) e2 +> (e1 +> Exp) '|' (e2 +> Exp);
Tiger +> IfThenElse
TIf i e1 e2 +> 'if' (i +> Exp) 'then' (e1 +> Exp) 'else' (e2 +> Exp);
```

The following figure shows the putback transformation. By giving different source program, the same AST may be printed differently.



References

[1] Zirun Zhu, Hsiang-Shang Ko, Pedro Martins, João Saraiva, and Zhenjiang Hu. BiYacc: Roll your parser and reflective printer into one. BX'15.

[2] Hsiang-Shang Ko, Tao Zan, and Zhenjiang Hu. BiGUL: A formally verified core language for putback-based bidirectional programming. PEPM'16.

Demo Website

An interactive website with examples is available at
`http://www.prg.nii.ac.jp/project/biyacc.html`