Languages-beta: SIMPLE-3-Statements [*]

The PLanCompS Project

SIMPLE-3-Statements.cbs | PLAIN | PRETTY

Links to non-local declarations are disabled in this sample.

Language “SIMPLE”

3 **Statements**

```
Syntax  Block : block  ::=  '{' stmts? '}'
        Stmts : stmts  ::=  stmt stmts?
         Stmt : stmt   ::=  imp-stmt | vars-decl
  ImpStmt : imp-stmt   ::=  block
                         |  exp ';'
                         |  'if' '(' exp ')' block ('else' block)?
                         |  'while' '(' exp ')' block
                         |  'for' '(' stmt exp ';' exp ')' block
                         |  'print' '(' exps ')' ';'
                         |  'return' exp? ';'
                         |  'try' block 'catch' '(' id ')' block
                         |  'throw' exp ';'
```

```
Rule  ⟦ 'if' '(' Exp ')' Block ⟧ : stmt =
      ⟦ 'if' '(' Exp ')' Block 'else' '{' '}' ⟧
```

```
Rule  ⟦ 'for' '(' Stmt Exp1 ';' Exp2 ')'
        '{' Stmts '}' ⟧ : stmt =
      ⟦ '{' Stmt
        'while' '(' Exp1 ')'
        '{' '{' Stmts '}' Exp2 ';' '}'
        '}' ⟧
```

```
Semantics  exec⟦ _ : stmts ⟧ : ⇒ null-type
```

```
Rule  exec⟦ '{' '}' ⟧ = null
```

```
Rule  exec⟦ '{' Stmts '}' ⟧ = exec⟦ Stmts ⟧
```

[*]Suggestions for improvement: plancomps@gmail.com.
Reports of issues: https://github.com/plancomps/CBS-beta/issues.

```
Rule  exec⟦ ImpStmt Stmts ⟧ =
        sequential(exec⟦ ImpStmt ⟧, exec⟦ Stmts ⟧)

Rule  exec⟦ VarsDecl Stmts ⟧ =
        scope(declare⟦ VarsDecl ⟧, exec⟦ Stmts ⟧)

Rule  exec⟦ VarsDecl ⟧ = effect(declare⟦ VarsDecl ⟧)

Rule  exec⟦ Exp ‘;’ ⟧ = effect(rval⟦ Exp ⟧)

Rule  exec⟦ ‘if’ ‘(’ Exp ‘)’ Block$_1$ ‘else’ Block$_2$ ⟧ =
        if-else(rval⟦ Exp ⟧, exec⟦ Block$_1$ ⟧, exec⟦ Block$_2$ ⟧)

Rule  exec⟦ ‘while’ ‘(’ Exp ‘)’ Block ⟧ = while(rval⟦ Exp ⟧, exec⟦ Block ⟧)

Rule  exec⟦ ‘print’ ‘(’ Exps ‘)’ ‘;’ ⟧ = print(rvals⟦ Exps ⟧)

Rule  exec⟦ ‘return’ Exp ‘;’ ⟧ = return(rval⟦ Exp ⟧)

Rule  exec⟦ ‘return’ ‘;’ ⟧ = return(null)

Rule  exec⟦ ‘try’ Block$_1$ ‘catch’ ‘(’ Id ‘)’ Block$_2$ ⟧ =
        handle-thrown(
          exec⟦ Block$_1$ ⟧,
          scope(
            bind(id⟦ Id ⟧, allocate-initialised-variable(values, given)),
            exec⟦ Block$_2$ ⟧))

Rule  exec⟦ ‘throw’ Exp ‘;’ ⟧ = throw(rval⟦ Exp ⟧)
```