Unstable-Languages-beta: IMPPP-4 *

The PLanCompS Project

`IMPPP-4.cbs` | PLAIN | PRETTY

OUTLINE

Language "IMPPP"

4 Statements and blocks

Syntax

Stmt : stmt ::= block
| 'int' ids ';'
| aexp ';'
| 'if' '(' bexp ')' block 'else' block
| 'while' '(' bexp ')' block
| 'print' '(' aexps ')' ';'
| 'halt' ';'
| 'join' aexp ';'

Block : block ::= '{' stmt* '}'

---

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

Semantics execute⟦ _ : stmt* ⟧ : ⇒ null-type

Rule execute⟦ ⟧ = null

Rule execute⟦ `int` *IL* `;` *Stmt** ⟧ =
  scope(
    collateral(declare-int-vars⟦ *IL* ⟧),
    execute⟦ *Stmt** ⟧)

Otherwise execute⟦ *Stmt* *Stmt*+ ⟧ =
  sequential(execute⟦ *Stmt* ⟧, execute⟦ *Stmt*+ ⟧)

Rule execute⟦ *AExp* `;` ⟧ =
  effect(eval-arith⟦ *AExp* ⟧)

Rule execute⟦ `if` `(` *BExp* `)` $Block_1$ `else` $Block_2$ ⟧ =
  if-true-else(
    eval-bool⟦ *BExp* ⟧,
    execute⟦ $Block_1$ ⟧,
    execute⟦ $Block_2$ ⟧)

Rule execute⟦ `while` `(` *BExp* `)` *Block* ⟧ =
  while-true(eval-bool⟦ *BExp* ⟧, execute⟦ *Block* ⟧)

Rule execute⟦ `print` `(` *AExp* `)` `;` ⟧ =
  print(eval-arith⟦ *AExp* ⟧)

Rule ⟦ `print` `(` *AExp* `,` *AExps* `)` `;` ⟧ : stmt+ =
⟦ `print` `(` *AExp* `)` `;` `print` `(` *AExps* `)` `;` ⟧

Rule execute⟦ `halt` `;` ⟧ = thread-terminate(current-thread)

Rule execute⟦ `join` *AExp* `;` ⟧ =
  thread-join(lookup-index(eval-arith⟦ *AExp* ⟧))

Rule execute⟦ `{` *Stmt** `}` ⟧ = execute⟦ *Stmt** ⟧

Variable declarations

Syntax *IL* : ids ::= id (`,` ids)?

Semantics declare-int-vars⟦ _ : ids ⟧ : (⇒ environments)+

Rule declare-int-vars⟦ *I* ⟧ =
  bind(id⟦ *I* ⟧, allocate-initialised-variable(integers, 0))

Rule declare-int-vars⟦ *I* `,` *IL* ⟧ =
  declare-int-vars⟦ *I* ⟧, declare-int-vars⟦ *IL* ⟧