Unstable-Languages-beta: IMPPP-2 *

The PLanCompS Project

`IMPPP-2.cbs` | PLAIN | PRETTY

OUTLINE

2 Value expressions
Value expression sequences

Language "IMPPP"

2 Value expressions

Syntax *AExp* : aexp ::= int
| string
| id
| aexp '+' aexp
| aexp '/' aexp
| '(' aexp ')'
| id '=' aexp
| '++' id
| 'read' '(' ')'
| 'spawn' block

Type aexp-values ⇝ integers | strings

Funcon integer-add-or-string-append(_ : aexp-values, _ : aexp-values)
: ⇒ aexp-values
Rule integer-add-or-string-append(N_1 : integers, N_2 : integers) ⇝
integer-add(N_1, N_2)
Rule integer-add-or-string-append(S_1 : strings, S_2 : strings) ⇝
string-append(S_1, S_2)

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

Semantics eval-arith⟦ _ : aexp ⟧ : ⇒ aexp-values

Rule eval-arith⟦ *N* ⟧ = int-val⟦ *N* ⟧

Rule eval-arith⟦ *S* ⟧ = string-val⟦ *S* ⟧

Rule eval-arith⟦ *I* ⟧ = assigned(bound(id⟦ *I* ⟧))

Rule eval-arith⟦ $AExp_1$ '+' $AExp_2$ ⟧ =
integer-add-or-string-append(eval-arith⟦ $AExp_1$ ⟧, eval-arith⟦ $AExp_2$ ⟧)

Rule eval-arith⟦ $AExp_1$ '/' $AExp_2$ ⟧ =
checked integer-divide(eval-arith⟦ $AExp_1$ ⟧, eval-arith⟦ $AExp_2$ ⟧)

Rule eval-arith⟦ '(' *AExp* ')' ⟧ = eval-arith⟦ *AExp* ⟧

Rule eval-arith⟦ *I* '=' *AExp* ⟧ =
give(
eval-arith⟦ *AExp* ⟧,
sequential(
assign(bound(id⟦ *I* ⟧), given),
given))

Rule eval-arith⟦ '++' *I* ⟧ =
give(
integer-add(assigned(bound(id⟦ *I* ⟧)), 1),
sequential(
assign(bound(id⟦ *I* ⟧), given),
given))

Rule eval-arith⟦ `'read' '(' ')'` ⟧ = read

Rule eval-arith⟦ `'spawn'` *Block* ⟧ =
allocate-index(
thread-activate thread-joinable thunk closure execute⟦ *Block* ⟧)

Value expression sequences

Syntax *AExps* : aexps ::= aexp ('**,**' aexps)?

Semantics eval-arith-seq⟦ _ : aexps ⟧ : $(⇒ \text{aexp-values})^+$

Rule eval-arith-seq⟦ *AExp* ⟧ = eval-arith⟦ *AExp* ⟧

Rule eval-arith-seq⟦ *AExp* ',' *AExps* ⟧ = eval-arith⟦ *AExp* ⟧, eval-arith-seq⟦ *AExp* ⟧