

Unstable-Languages-beta: IMPPP-2 *

The PPlanCompS Project

IMPPP-2.cbs | PLAIN | PRETTY

OUTLINE

2 Value expressions

Value expression sequences

Language “IMPPP”

2 Value expressions

Syntax $AExp : aexp ::=$

- int
- $string$
- id
- $aexp \text{ ' + ' } aexp$
- $aexp \text{ ' / ' } aexp$
- $\text{' (' } aexp \text{ ') '}$
- $id \text{ ' = ' } aexp$
- $\text{' ++ ' } id$
- $\text{' read ' ' (' ') '}$
- $\text{' spawn ' } block$

Type $aexp\text{-values} \rightsquigarrow integers \mid strings$

Funcon $integer\text{-add-or-string-append}(_ : aexp\text{-values}, _ : aexp\text{-values})$
 $: \Rightarrow aexp\text{-values}$

Rule $integer\text{-add-or-string-append}(N_1 : integers, N_2 : integers) \rightsquigarrow$
 $integer\text{-add}(N_1, N_2)$

Rule $integer\text{-add-or-string-append}(S_1 : strings, S_2 : strings) \rightsquigarrow$
 $string\text{-append}(S_1, S_2)$

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

Semantics $\text{eval-arith}[_ : \text{aexp}] : \Rightarrow \text{aexp-values}$

Rule $\text{eval-arith}[N] = \text{int-val}[N]$

Rule $\text{eval-arith}[S] = \text{string-val}[S]$

Rule $\text{eval-arith}[I] = \text{assigned}(\text{bound}(\text{id}[I]))$

Rule $\text{eval-arith}[AExp_1 \text{ ' + ' } AExp_2] =$
 $\text{integer-add-or-string-append}(\text{eval-arith}[AExp_1], \text{eval-arith}[AExp_2])$

Rule $\text{eval-arith}[AExp_1 \text{ ' / ' } AExp_2] =$
 $\text{checked integer-divide}(\text{eval-arith}[AExp_1], \text{eval-arith}[AExp_2])$

Rule $\text{eval-arith}['(AExp ')'] = \text{eval-arith}[AExp]$

Rule $\text{eval-arith}[I \text{ '=' } AExp] =$
 $\text{give}(\text{eval-arith}[AExp],$
 $\text{sequential}(\text{assign}(\text{bound}(\text{id}[I]), \text{given}),$
 $\text{given}))$

Rule $\text{eval-arith}['++' I] =$
 $\text{give}(\text{integer-add}(\text{assigned}(\text{bound}(\text{id}[I])), 1),$
 $\text{sequential}(\text{assign}(\text{bound}(\text{id}[I]), \text{given}),$
 $\text{given}))$

Rule $\text{eval-arith}['\text{read}' (' ')] = \text{read}$

Rule $\text{eval-arith}['\text{spawn}' Block] =$
 $\text{allocate-index}(\text{thread-activate thread-joinable thunk closure execute}[Block])$

Value expression sequences

Syntax $AExps : \text{aexps} ::= \text{aexp} (' , ' \text{aexps})?$

Semantics $\text{eval-arith-seq}[_ : \text{aexps}] : (\Rightarrow \text{aexp-values})^+$

Rule $\text{eval-arith-seq}[AExp] = \text{eval-arith}[AExp]$

Rule $\text{eval-arith-seq}[AExp \text{ ' , ' } AExps] = \text{eval-arith}[AExp], \text{eval-arith-seq}[AExps]$