Languages-beta: IMP-1 [*]

The PLanCompS Project

IMP-1.cbs | PLAIN | PRETTY

Language “IMP”

1 Arithmetic expressions

Syntax *AExp* : aexp ::= num
| id
| aexp ‘+’ aexp
| aexp ‘/’ aexp
| ‘(’ aexp ‘)’

Semantics eval-arith⟦ _ : aexp ⟧ : ⇒ integers
Rule eval-arith⟦ *N* ⟧ = int-val⟦ *N* ⟧
Rule eval-arith⟦ *I* ⟧ = assigned(bound(id⟦ *I* ⟧))
Rule eval-arith⟦ $AExp_1$ ‘+’ $AExp_2$ ⟧ =
integer-add(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* ⟧

Syntax *N* : num ::= ‘-’? _decimal

Lexis *D* : decimal ::= (‘0’ – ‘9’)$^+$

Semantics int-val⟦ _ : num ⟧ : integers
Rule int-val⟦ *D* ⟧ = decimal-natural(“*D*”)
Rule int-val⟦ ‘-’ *D* ⟧ = integer-negate(int-val⟦ *D* ⟧)

Lexis *I* : id ::= (‘A’ – ‘Z’ | ‘a’ – ‘z’)$^+$

Semantics id⟦ _ : id ⟧ : ids
Rule id⟦ *I* ⟧ = “*I*”

Lexis keyword ::= ‘else’ | ‘false’ | ‘if’ | ‘true’ | ‘while’

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