

Languages-beta: SIMPLE-4-Declarations *

The P_LanCompS Project

SIMPLE-4-Declarations.cbs | PLAIN | PRETTY

OUTLINE

4 Declarations

- 4.1 Variable Declarations
- 4.2 Arrays
- 4.3 Function Declarations

Language "SIMPLE"

4 Declarations

Syntax $Decl : decl ::= \text{vars-decl} \mid \text{func-decl}$

Semantics $\text{declare}[_ : decl] : \Rightarrow \text{environments}$

4.1 Variable Declarations

Syntax $VarsDecl : \text{vars-decl} ::= \text{'var' declarators ';'}$

$Declarators : \text{declarators} ::= \text{declarator (',' declarators)}^?$

Rule $[\text{'var' Declarator ',' Declarators ';' Stmts?}] : \text{stmts} =$
 $[\text{'var' Declarator ';' 'var' Declarators ';' Stmts?}]$

Rule $[\text{'var' Declarator ',' Declarators ';' Decls?}] : \text{decls} =$
 $[\text{'var' Declarator ';' 'var' Declarators ';' Decls?}]$

Rule $\text{declare}[\text{'var' Declarator ';' }] = \text{var-declare}[Declarator]$

Syntax $Declarator : \text{declarator} ::= \text{id}$
 $\mid \text{id '=' exp}$
 $\mid \text{id ranks}$

Semantics $\text{var-declare}[_ : \text{declarator}] : \Rightarrow \text{environments}$

Rule $\text{var-declare}[Id] = \text{bind}(\text{id}[Id], \text{allocate-variable}(\text{values}))$

Rule $\text{var-declare}[Id \text{'=' } Exp] =$
 $\text{bind}(\text{id}[Id], \text{allocate-initialised-variable}(\text{values}, \text{rval}[Exp]))$

Rule $\text{var-declare}[Id \text{Ranks}] =$
 $\text{bind}(\text{id}[Id], \text{allocate-nested-vectors}(\text{ranks}[Ranks]))$

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

4.2 Arrays

Syntax $Ranks : ranks ::= '[' \text{ exps } ']' \text{ ranks?}$

Rule $\llbracket '[' \text{ Exp } ', ' \text{ Exps } ']' \text{ Ranks? } \rrbracket : ranks =$
 $\llbracket '[' \text{ Exp } ']' '[' \text{ Exps } ']' \text{ Ranks? } \rrbracket$

Compare this with p28 of the K version.

Semantics $ranks \llbracket _ : ranks \rrbracket : (\Rightarrow nats)^+$

Rule $ranks \llbracket '[' \text{ Exp } ']' \rrbracket = rval \llbracket \text{Exp} \rrbracket$

Rule $ranks \llbracket '[' \text{ Exp } ']' \text{ Ranks } \rrbracket = rval \llbracket \text{Exp} \rrbracket, ranks \llbracket \text{Ranks} \rrbracket$

Funcon $allocate_nested_vectors(_ : nats^+) : \Rightarrow variables$

Rule $allocate_nested_vectors(N : nats) \rightsquigarrow$
 $allocate_initialised_variable($
 $\quad vectors(variables),$
 $\quad vector(left_to_right_repeat(allocate_variable(values), 1, N)))$

Rule $allocate_nested_vectors(N : nats, N^+ : nats^+) \rightsquigarrow$
 $allocate_initialised_variable($
 $\quad vectors(variables),$
 $\quad vector(left_to_right_repeat(allocate_nested_vectors(N^+), 1, N)))$

4.3 Function Declarations

Syntax $FuncDecl : func_decl ::= 'function' \text{ id } '(' \text{ ids? } ')' \text{ block}$

Rule $declare \llbracket 'function' \text{ Id } '(' \text{ Ids? } ')' \text{ Block } \rrbracket =$
 $bind($
 $\quad id \llbracket \text{Id} \rrbracket,$
 $\quad allocate_variable(functions(tuples(values*), values)))$

Semantics $initialise \llbracket _ : decl \rrbracket : \Rightarrow \text{null-type}$

Rule $initialise \llbracket 'var' \text{ Declarators } ';' \rrbracket = \text{null}$

Rule $initialise \llbracket 'function' \text{ Id } '(' \text{ Ids? } ')' \text{ Block } \rrbracket =$
 $assign($
 $\quad bound(id \llbracket \text{Id} \rrbracket),$
 $\quad function \text{ closure}($
 $\quad \quad scope($
 $\quad \quad \quad match(given, tuple(patts \llbracket \text{Ids? } \rrbracket)),$
 $\quad \quad \quad handle_return(exec \llbracket \text{Block} \rrbracket)))$

Syntax $Ids : ids ::= id \text{ (', ' ids)?}$

Semantics $patts \llbracket _ : ids? \rrbracket : patterns^*$

Rule $patts \llbracket \rrbracket = ()$

Rule $patts \llbracket \text{Id} \rrbracket =$
 $pattern \text{ closure}($
 $\quad bind($
 $\quad \quad id \llbracket \text{Id} \rrbracket,$
 $\quad \quad allocate_initialised_variable(values, given)))$

Rule $patts \llbracket \text{Id } ', ' \text{ Ids } \rrbracket =$
 $patts \llbracket \text{Id} \rrbracket, patts \llbracket \text{Ids} \rrbracket$