

Unstable-Languages-beta: SIMPLE-THR-4-Declarations *

The P_LanCompS Project

SIMPLE-THR-4-Declarations.cbs | PLAIN | PRETTY

OUTLINE

4 Declarations

- 4.1 Variable Declarations
- 4.2 Arrays
- 4.3 Function Declarations

Language "SIMPLE-THR"

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 $Ranks : ranks ::= '['\ exps\ ']' \ ranks?$

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

Semantics $ranks[_ : ranks] : (\Rightarrow nats)^+$

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

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

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 $FuncDecl : func-decl ::= 'function' \ id\ '(' \ ids? \ ')' \ block$

Rule $declare[\llbracket 'function' \ Id\ '(' \ Ids? \ ')' \ Block \rrbracket] =$
 $bind($
 $\quad id[\llbracket Id \rrbracket],$
 $\quad allocate-variable(functions(tuples(values*), values)))$

Semantics $initialise[_ : decl] : \Rightarrow null-type$

Rule $initialise[\llbracket 'var' \ Declarators\ ';' \rrbracket] = null$

Rule $initialise[\llbracket 'function' \ Id\ '(' \ Ids? \ ')' \ Block \rrbracket] =$
 $assign($
 $\quad bound(id[\llbracket Id \rrbracket]),$
 $\quad function\ closure($
 $\quad \quad scope($
 $\quad \quad \quad match(given, tuple(patts[\llbracket Ids? \rrbracket])),$
 $\quad \quad \quad handle-return(exec[\llbracket Block \rrbracket]))$

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

Semantics $patts[_ : ids?] : patterns^*$

Rule $patts[_] = ()$

Rule $patts[\llbracket Id \rrbracket] =$
 $pattern\ closure($
 $\quad bind($
 $\quad \quad id[\llbracket Id \rrbracket],$
 $\quad \quad allocate-initialised-variable(values, given)))$

Rule $patts[\llbracket Id\ ',\ ' \ Ids \rrbracket] =$
 $patts[\llbracket Id \rrbracket], patts[\llbracket Ids \rrbracket]$