CBS-LaTeX Test

`SIMPLE/SIMPLE-3-Statements/SIMPLE-3-Statements.cbs`

January 12, 2021

Language "SIMPLE"

3 Statements

Syntax *Block* : block ::= `{` stmts? `}`

Stmts : stmts ::= stmt stmts?

Stmt : stmt ::= imp-stmt
| vars-decl

ImpStmt : imp-stmt ::= block
| exp `;`
| `if (` exp `)` block (`else` block)?
| `while (` exp `)` block
| `for (` stmt exp `;` exp `)` block
| `print (` exps `) ;`
| `return` exp? `;`
| `try` block `catch (` id `)` block
| `throw` exp `;`

Rule ⟦ `if (` *Exp* `)` *Block* ⟧ : stmt =
⟦ `if (` *Exp* `)` *Block* `else { }` ⟧

Rule ⟦ `for (` *Stmt* Exp_1 `;` Exp_2 `) {` *Stmts* `}` ⟧ : stmt =
⟦ `{` *Stmt* `while (` Exp_1 `) { {` *Stmts* `}` Exp_2 `; } }` ⟧

Semantics exec⟦ _ : stmts ⟧ : ⇒ null-type

Rule exec⟦ `{ }` ⟧ =
null

Rule exec⟦ `{` *Stmts* `}` ⟧ =
exec⟦ *Stmts* ⟧

Rule exec⟦ *ImpStmt Stmts* ⟧ =
sequential(exec⟦ *ImpStmt* ⟧,
exec⟦ *Stmts* ⟧)

Rule exec⟦ *VarsDecl Stmts* ⟧ =
scope(declare⟦ *VarsDecl* ⟧,
exec⟦ *Stmts* ⟧)

Rule exec⟦ *VarsDecl* ⟧ =
effect(declare⟦ *VarsDecl* ⟧)

Rule exec⟦ *Exp* `;` ⟧ =
effect(rval⟦ *Exp* ⟧)

Rule exec⟦ `if (` *Exp* `)` $Block_1$ `else` $Block_2$ ⟧ =
if-else(rval⟦ *Exp* ⟧,
exec⟦ $Block_1$ ⟧,
exec⟦ $Block_2$ ⟧)

Rule exec⟦ `while (` *Exp* `)` *Block* ⟧ =
while(rval⟦ *Exp* ⟧,
exec⟦ *Block* ⟧)

Rule exec⟦ `print (` *Exps* `) ;` ⟧ =
print(rvals⟦ *Exps* ⟧)

Rule exec⟦ `return` *Exp* `;` ⟧ =
return(rval⟦ *Exp* ⟧)

Rule exec⟦ `return ;` ⟧ =
return(null)

Rule exec⟦ `try` $Block_1$ `catch (` *Id* `)` $Block_2$ ⟧ =
handle-thrown(exec⟦ $Block_1$ ⟧,
scope(bind(id⟦ *Id* ⟧,
allocate-initialised-variable(values,
given)),
exec⟦ $Block_2$ ⟧))

Rule exec⟦ `throw` *Exp* `;` ⟧ =
throw(rval⟦ *Exp* ⟧)