Homework 3: Semantic Analysis
CS 421 Compiler Design and Construction

Asad Tariq & Fahad Shaikh

Habib University

Fall 2022

Due: November 13, 2022

1 LL(1) Grammar of TUPLE (The Ultimate Programming LanguagE)

$Program \rightarrow dt\ id\ (ParamList)\ \{Stmts\}$

$ParamList \rightarrow dt\ id\ PList$

$PList \rightarrow ,\ dt\ id\ PList \mid \epsilon$

$Stmts \rightarrow \overline{Stmts}$

$\overline{Stmts} \rightarrow DecStmt\ \overline{Stmts} \mid AssignStmt\ \overline{Stmts} \mid ForStmt\ \overline{Stmts} \mid IfStmt\ \overline{Stmts} \mid ReturnStmt\ \overline{Stmts} \mid \epsilon$

$DecStmts \rightarrow dt\ id\ OptionalAssign\ List$

$List \rightarrow ,\ dt\ OptionalAssign\ List \mid \epsilon$

$OptionalAssign \rightarrow =\ Expr\ ; \mid \epsilon$

$AssignStmt \rightarrow id = Expr\ ;$

$Expr \rightarrow T\ E`$

$E` \rightarrow +\ T\ E` \mid \epsilon$

$T \rightarrow F\ T`$

$T` \rightarrow *\ F\ T` \mid \epsilon$

$F \rightarrow (Expr) \mid id$

$ForStmt \rightarrow for\ (Type\ id\ Expr\ ;\ Expr\ relop\ Expr\ ;\ id\ {+}{+})\ \{Stmts\}$

$Type \rightarrow dt \mid \epsilon$

$IfStmt \rightarrow if\ (Expr\ relop\ Expr)\ \{Stmts\}\ OptionalElse$

$OptionalElse \rightarrow else\ \{Stmts\} \mid \epsilon$

$ReturnStmt \rightarrow return\ Expr\ ;$

1. ***Introducing attributes and semantic actions***: For the grammar used in Assignment #02 submit a L-attributed grammar to implement a semantic analyzer. Introduce attributes (synthesized, inherited, or lexical) to the grammar symbols wherever appropriate. These attributes may include *name, type (base or constructed types), size (in bytes), value (for constants),* and *scope.* Also include necessary semantic actions (rules) at appropriate points that will assign types, perform semantic error checks, and make appropriate entries in the symbol table.

Solution:
The LL(1) has been annotated as shown below alongside the rules applied to it:

$Program \rightarrow dt_{type}\ id_{name=lexical}\ (ParamList)_{pt}\ \{A\}\ \{Stmts\}$
Rule A: If lookup(name, rt) == False: `enter_to_symb_table`*, else:* ***Redeclaration Error!***
$ParamList_{pt} \rightarrow dt_{type}\ id_{name=lexical}\ \{D\}\ PList_{pt1}\ \{\text{B}\}$
$PList_{lt} \rightarrow\ ,\ dt_{type}\ id_{name=lexical}\ \{D\}\ PList_{lt1}\ |\ \epsilon_{null\_type}\ \{\text{C}\}$
$Stmts_s \rightarrow \overline{Stmts}_s$
$\overline{Stmts}_s \rightarrow DecStmt_s\ \overline{Stmts}|\ AssignStmt_s\ \overline{Stmts}|\ ForStmt_s\ \overline{Stmts}|\ IfStmt_s\ \overline{Stmts}|\ ReturnStmt_s\ \overline{Stmts}|\ \epsilon_{null\_type}$
$DecStmts_s \rightarrow dt_{type}\ id_{name=lexical}\ \{D\}\ OptionalAssign_{et}\ \{E\}\ List_{t1=type,s1=s}$
$List_{t,s} \rightarrow\ ,\ dt_{type}\ OptionalAssign_{et}\ \{E\}\ List_{t,s}\ |\ \epsilon_{null\_type}$
$OptionalAssign_{et} \rightarrow\ =\ Expr_{t1}\ ;\ |\ \epsilon\ \{\text{G}\}$
$AssignStmt_s \rightarrow id_{name=lexical}\ =\ \{F\}\ Expr_{et,s}\ ;$
$Expr_{t1} \rightarrow T_{t2}\ \{F\}\ E`_{t1}$
$E`_{t1} \rightarrow +\ T_{t2}\ E`_{t1}\ |\ \epsilon_{null\_type}$
$T_{t2} \rightarrow F_t\ \{F\}\ T`_{t2}$
$T`_{t2} \rightarrow *\ F_t\ \{H\}\ T`_{t2}\ |\ \epsilon_{null\_type}$
$F_1 \rightarrow (Expr_{t1})\ |\ id_{name=lexical}$
$ForStmt_s \rightarrow for\ (Type_t\ id_{name=lexical}\ Expr_{et,s}\ ;\ Expr_{et,s}\ relop\ \{H\}\ Expr_{et,s}\ ;\ id_{name=lexical}\ ++\ \{H\})\ \{Stmts_s\}$
$Type_t \rightarrow dt_{type}\ |\ \epsilon_{null\_type}$
$IfStmt_s \rightarrow if\ (Expr_{et,s}\ relop\ \{H\}\ Expr_{et,s})\ \{Stmts_s\}\ OptionalElse_s$
$OptionalElse_s \rightarrow else\ \{Stmts_s\}\ |\ \epsilon_{null\_type}$
$ReturnStmt_s \rightarrow return\ Expr_{et,s}\ ;$

Rules:

- *Rule A: If lookup(name, rt) == False:* `enter_to_symb_table`*, else:* ***Redeclaration Error!***
- *Rule B: If pt1 $\neq$ NULL then: pt = type + pt1, else: pt = type*
- *Rule C: If lt1 $\neq$ NULL then: lt = type + lt1, else: lt = type*
- *Rule C: If lt1 = NULL then: lt = NULL*

- *Rule D: If declarationCheck(name, s) == False then:* `enter_to_symb_table`*(name, type, s) else:* ***Redeclaration Error!***
- *Rule E: typeCheck(type, et)*
- *Rule F: Check for type and declaration of identifiers*
- *Rule G: If Expr* $\neq$ *NULL then: et = t1, else: et = NULL*
- *Rule H: Check type compatibility between operators and operands*