Progetto Compilatori

A.A. 2020/2021

Gaetano Antonucci Alessio Romano

27 gennaio 2021

Indice

1 Grammatica Utilizzata

```
      Program := VarDeclList ProcList
  VarDeclList := VarDecl VarDeclList
              | /* empty */
      VarDecl := Type IdListInit SEMI
         Type := INT
              | BOOL
              | FLOAT
              | STRING
   IdListInit := ID
              | IdListInit COMMA ID
              | ID ASSIGN Expr
              | IdListInit COMMA ID ASSIGN Expr
     ProcList := Proc
              | Proc ProcList
         Proc := PROC ID LPAR ParamDeclList RPAR ResultTypeList COLON ProcBody
              | PROC ID LPAR RPAR ResultTypeList COLON ProcBody
     ProcBody := VarDeclList StatList RETURN ReturnExprs CORP SEMI
              | VarDeclList RETURN ReturnExprs CORP SEMI
ParamDeclList := ParDecl
              | ParamDeclList SEMI ParDecl
      ParDecl := Type ParIdList
    ParIdList := ID
              | ParIdList COMMA ID
ResultTypeList := ResultType
              | ResultType COMMA ResultTypeList
   ResultType := Type
              | VOID
     StatList := Stat SEMI
              | Stat SEMI StatList
         Stat := IfStat
              | WhileStat
              | ReadlnStat
              | AssignStat
              | CallProc
       IfStat := IF Expr THEN StatList ElifList Else FI
     ElifList := Elif ElifList
              | /* empty */
```

```
      Elif :=  ELIF Expr THEN StatList
      Else :=  ELSE StatList
            |  /* empty */
 WhileStat :=  WHILE StatList RETURN Expr DO StatList OD
            |  WHILE Expr DO StatList OD
ReadlnStat :=  READ LPAR IdList RPAR
    IdList :=  ID
            |  IdList COMMA ID
 WriteStat :=  WRITE LPAR ExprList RPAR
AssignStat :=  IdList ASSIGN ExprList
  CallProc :=  ID LPAR ExprList RPAR
            |  ID LPAR RPAR
ReturnExprs :=  ExprList
            |  /* empty */
  ExprList :=  Expr
            |  Expr COMMA ExprList
      Expr :=  NULL
            |  TRUE
            |  FALSE
            |  INT_CONST
            |  FLOAT_CONST
            |  STRING_CONST
            |  ID
            |  MINUS Expr
            |  Expr PLUS Expr
            |  Expr MINUS Expr
            |  Expr TIMES Expr
            |  Expr DIV Expr
            |  NOT Expr
            |  Expr AND Expr
            |  Expr OR Expr
            |  Expr GT Expr
            |  Expr GE Expr
            |  Expr LT Expr
            |  Expr LE Expr
            |  Expr EQ Expr
            |  Expr NE Expr
            |  CallProc
```

2 Regole di Type Checking implementate

2.1 Tipi Primitivi

$$\Gamma \vdash null : null \qquad \Gamma \vdash true : boolean \qquad \Gamma \vdash false : boolean$$

$$\Gamma \vdash int : int \qquad \Gamma \vdash float : float \qquad \Gamma \vdash string : string \qquad \Gamma \vdash bool : boolean$$

2.2 Dichiarazioni di Variabili

$$\frac{(x : \tau) \in \Gamma}{\Gamma \vdash x : \tau}$$

2.3 Operazioni Unarie

$$\frac{\Gamma \vdash e : \tau_1 \quad optype1(op, \tau_1) = \tau}{\Gamma \vdash (op\ e) : \tau}$$

2.4 Operazioni Binarie

$$\frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma \vdash e_2 : \tau_2 \quad optype2(op, \tau_1, \tau_2) = \tau}{\Gamma \vdash (e_1\ op\ e_2) : \tau}$$

2.5 Chiamata a Procedura

$$\frac{\Gamma \vdash f : \tau_i^{\,i \in 1 \dots n} \to \tau_j^{\,j \in 1 \dots m} \quad \Gamma \vdash e_i : \tau_i^{\,i \in 1 \dots n}}{\Gamma \vdash f(e_i^{\,i \in 1 \dots n}) : \tau_j^{\,j \in 1 \dots m}}$$

2.6 Statement

2.6.1 if-then

$$\frac{\Gamma \vdash e : boolean \quad \Gamma \vdash stmt}{\Gamma \vdash \texttt{if}\ e\ \texttt{then}\ stmt\ \texttt{fi}}$$

2.6.2 if-then-else

$$\frac{\Gamma \vdash e : boolean \quad \Gamma \vdash stmt_1 \quad \Gamma \vdash stmt_2}{\Gamma \vdash \texttt{if}\ e\ \texttt{then}\ stmt_1\ \texttt{else}\ stmt_2\ \texttt{fi}}$$

2.6.3 if-then-elif-else

$$\frac{\Gamma \vdash e_j^{\,j \in 1 \dots m} : boolean \quad \Gamma \vdash stmt_i^{\,i \in 1 \dots 3}}{\Gamma \vdash \texttt{if}\ e_1\ \texttt{then}\ stmt_1\ (\texttt{elif}\ e_j^{\,j \in 2 \dots m}\ \texttt{then}\ stmt_2\ )_t^{\,t \in 1 \dots k}\ \texttt{else}\ stmt_3\ \texttt{fi}}$$

2.6.4 while

$$\frac{\Gamma \vdash e : boolean \quad \Gamma \vdash stmt}{\Gamma \vdash \texttt{while}\ e\ \texttt{do}\ stmt\ \texttt{od}}$$

2.6.5 while-return

$$\frac{\Gamma \vdash e\colon \mathit{boolean} \quad \Gamma \vdash \mathit{stmt}_1 \quad \Gamma \vdash \mathit{stmt}_2}{\Gamma \vdash \texttt{while}\ \mathit{stmt}_1 \texttt{->}\ e\ \texttt{do}\ \mathit{stmt}_2\ \texttt{od}}$$

2.6.6 readln

$$\frac{(x_i^{\,i\in 1\dots n}\colon \tau_i^{\,i\in 1\dots n}) \in \Gamma}{\Gamma \vdash \texttt{readln}(x_i^{\,i\in 1\dots n})}$$

2.6.7 write

$$\frac{\Gamma \vdash e\colon \tau \quad (\tau \neq \texttt{void})^1}{\Gamma \vdash \texttt{write}(e\colon \tau)}$$

2.6.8 simple-assign

$$\frac{(x\colon \tau) \in \Gamma \quad \Gamma \vdash e\colon \tau}{\Gamma \vdash x := e}$$

2.6.9 multiple-assign

$$\frac{(x_i^{\,i\in 1\dots n}\colon \tau_i^{\,i\in 1\dots n}) \in \Gamma \quad \Gamma \vdash e_j^{\,j\in 1\dots n}\colon \tau_j^{\,j\in 1\dots n}}{\Gamma \vdash x_i^{\,i\in 1\dots n} := e_j^{\,j\in 1\dots n}}$$

2.6.10 return

$$\frac{(\$\mathit{ret}\colon \tau) \in \Gamma \quad \Gamma \vdash e\colon \tau}{\Gamma \vdash \texttt{->}e}$$

2.7 Tabelle di Compatibilità

[1] Si vuole intendere che il tipo di e deve essere diverso da `void`. Per ulteriori informazioni consultare le scelte di sviluppo del compilatore.

op	operand	result
-	integer	integer
-	float	float
!	boolean	boolean

(a) optype1

op	first operand	second operand	result
+ - * /	integer	integer	integer
+ - * /	integer	float	float
+ - * /	float	integer	float
+ - * /	float	float	float
&& ||	boolean	boolean	boolean
< = > <= >= <>	integer	integer	boolean
< = > <= >= <>	integer	float	boolean
< = > <= >= <>	float	integer	boolean
< = > <= >= <>	float	float	boolean
< = > <= >= <>	string	string	boolean

(b) optype2

Figura 1: Relazioni di tipo per gli operatori primitivi. Gli operatori aritmetici lavorano sia su numeri interi sia su numeri in virgola mobile. Gli operatori logiri ! && || (not, and e or) lavorano su boolean. Gli operatori di comparazione lavorano su tipi primitivi diversi da boolean.