Documentazione De Stefano-Napolitano

NewLang

Link GitLab:

https://gitlab.com/g4660/compilatori-a.a.-2022_23/newlang-semancgen-es5/de-stefano-napolitano_es5.git

1. Sintassi e scelte progettuali

I lessemi e i token del nostro linguaggio:

L'unica variazione sta nel'uso di "real" REAL anzichè "float" FLOAT

"start:" MAIN) ; }	"def" DEF	"{" LBRACK
";" SEMI	"out" OUT	"}" RBRACK
"," COMMA	"for" FOR	":" COLON
"|" PIPE	"if" IF	"<<" ASSIGN
"var" VAR	"then" THEN	"return" RETURN
"integer" INTEGER	"else" ELSE	"true" TRUE
"real" REAL	"while" WHILE	"false" FALSE
"string" STRING	"to" TO	"+" PLUS
"boolean" BOOL	"loop" LOOP	"-" MINUS
"char" CHAR	"<--" READ	"*" TIMES
"void" VOID	"-->" WRITE	"/" DIV
	"-->!" WRITELN	"^" POW
	"(" LPAR	"&" STR_CONCAT
	")" RPAR	"=" EQ
		"<>" NE
		"!=" NE
		"<" LT
		"<=" LE
		">" GT
		">=" GE
		"and" AND
		"or" OR
		"not" NOT

Struttura del progetto:

mostreremo solo le parti rilevanti alla comprensione

- de-stefano-napolitano
  - .idea
  - src
    - main
      - java
        - esercitazione5
          - Main.java
          - parser.java
          - sym.java (Interface)
          - Yylex.java
        - nodi:
        - Visitor
          - Visitatore.java (Interface)
          - TreeMaker.java
          - ScopingVisitor.java
          - AnalisiSemantica.java
          - GenerazioneCodiceC.java
          - Env.java (Questa classe implementa i type Enviroments)
          - OpType.java
          - OptypeTable.java (questa classe implementa le optypetable mostrate nel capitolo successivo)
          - RecordSymbolTable.java (Questa classe impementa un singolo record della tabella dei simboli)
  - srcflexcup
    - newLang.cup (Questo file specifica le produzioni del linguaggio con le relative azioni semantiche)
    - newLang.flex (Questo file specifica i lessemi e i token del linguaggio)
  - test
  - test_files
    - c_out
  - tests
  - file.xml (In questo file viene salvata la visualizzazione dell'albero sintattico in formato XML)
  - pom.xml

2. Scoping

La gestione dello scoping è stata implementata all'interno della classe ScopingVisitor.

I costrutti che creano un nuovo scope sono tutti gli statements che presentano un Body.

Il nuovo Scope viene generato all'interno del Body.

Nel caso di Fundecl si richiama il body passando anche la lista di parametri che verranno aggiunti all'interno della tabella dei simboli.

Nel caso dello Statement For il Body viene esteso con l'aggiunta dell'id presente nel for.

In ProgramRoot viene gestita la Global table, dove vengono aggiunte le variabili globali e le dichiarazioni delle funzioni.

Nella classe Scoping Visitor viene gestita anche la most closely nested rule, in particolare all'interno del metodo visit di Vardecl dove viene fatta una ricerca all'interno del type environment e se la variabile è presente viene aggiunta alla tabella corrente.

3. Regole di type checking implementate

Identificatore

$$\frac{\Gamma(id) = \tau}{\Gamma \vdash id : \tau}$$

Costanti

$$\Gamma \vdash int\_const : integer$$
$$\Gamma \vdash string\_const : string$$
$$\Gamma \vdash true : boolean$$
$$\Gamma \vdash false : boolean$$

Lista di istruzioni

$$\frac{\Gamma \vdash stmt_1 : notype \quad \Gamma \vdash stmt_2 : notype}{\Gamma \vdash stmt_1; stmt_2 : notype}$$

Chiamata a funzione con o senza tipo di ritorno (espressione o istruzione) senza controllo del parametro out

$$\frac{\Gamma \vdash f : \tau_1 \times \ldots \times \tau_n \to \tau \quad \Gamma \vdash e_i : \tau_i^{i \in 1 \ldots n}}{\Gamma \vdash f(e_1, \ldots, e_n) : \tau}$$

$$\frac{\Gamma \vdash f : \tau_1 \times \ldots \times \tau_n \to notype \quad \Gamma \vdash e_i : \tau_i^{i \in 1 \ldots n}}{\Gamma \vdash f(e_1, \ldots, e_n) : notype}$$

Assegnazione

$$\frac{\Gamma(id_i) : \tau_i^{i \in 1 \ldots n} \quad \Gamma \vdash e_i : \tau_i^{i \in 1 \ldots n}}{\Gamma \vdash id_1, \ldots, id_n << e_1, \ldots, e_n : notype}$$

Estensione per l'assegnazione : aggiungiamo la possibilità di associare un integer ad un real

$$\frac{\Gamma(id_i) : real \qquad \Gamma \vdash e_i : integer}{\Gamma \vdash id_i << e_i : notype}$$

Blocco dichiarazione-istruzione
(il type environment dell'istruzione stmt viene esteso con la dichiarazione di id, prima di fare il controllo di tipo dell'istruzione stmt):

$$\frac{\Gamma[id \mapsto \tau] \vdash stmt : notype}{\Gamma \vdash \tau\ id; stmt : notype}$$

Istruzione while

$$\frac{\Gamma \vdash e : boolean \quad \Gamma \vdash body : notype}{\Gamma \vdash \textbf{while}\ e\ \textbf{loop}\ body\ : notype}$$

Istruzione for

$$\frac{\Gamma \vdash e_1 : int\_const \quad \Gamma \vdash e_2 : int\_const \quad \Gamma[id \mapsto integer] \vdash body : notype}{\Gamma \vdash \textbf{for}\ id << e_1\ \textbf{to}\ e_2\ \textbf{loop}\ body\ : notype}$$

Istruzione if-then

$$\frac{\Gamma \vdash e : boolean \quad \Gamma \vdash body : notype}{\Gamma \vdash \textbf{if } e \textbf{ then } body \ : notype}$$

Operatori unari (vedi Table 1):

$$\frac{\Gamma \vdash e : \tau_1 \quad optype1(op_1, \tau_1) = \tau}{\Gamma \vdash op1 \ e : \tau}$$

Operatori binari (vedi Table 2):

$$\frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma \vdash e_2 : \tau_2 \quad optype2(op_2, \tau_1, \tau_2) = \tau}{\Gamma \vdash e_1 \ op_2 \ e_2 : \tau}$$

Tabelle per gli operatori:

op1	operando	risultato
MINUS	integer	integer
MINUS	float	float
NOT	boolean	boolean

Table 1: Tabella per *optype1*. Esempio di uso: *optype1*(NOT, boolean) = boolean

op	operando1	operando2	risultato
PLUS, TIMES, ...	integer	integer	integer
PLUS, TIMES, ...	integer	float	float
PLUS, TIMES, ...	float	integer	float
PLUS, TIMES, ...	float	float	float
STR_CONCAT	string	string	string
AND, OR	boolean	boolean	boolean
LT, LE, GT, ...	integer	integer	boolean
LT, LE, GT, ...	float	integer	boolean
LT, LE, GT, ...	integer	float	boolean
LT, LE, GT, ...	float	float	boolean

Table 2: Tabella per *optype2*. Esempio di uso: *optype2*(TIMES, float, integer) = float

Di seguito la tabella optype2 estesa :

Per quanto riguarda la concateazione, si è previsto che essa dovesse avere almeno un operatore come stringa.

op1	operando1	operando2	risultato
LT,LE,GT,GE,NE,EQ	string	string	boolean
STR_CONCAT	string	integer	string
STR_CONCAT	string	real	string
STR_CONCAT	integer	string	string
STR_CONCAT	real	string	string