Università degli Studi di Salerno
Dipartimento di Informatica

Progetto di Compilatori

MyFun

1. Analisi lessicale

Implementazione aderente alle indicazioni fornite.

2. Analisi sintattica

Risoluzione dei conflitti della grammatica in Cup

1. Una volta inserite le regole della grammatica data, ho provato a generare il parser con Cup ed ho analizzato il file "dump.txt" prodotto in output: sono stati individuati dei conflitti.
2. Ho aggiunto le **regole di precedenza e associatività**, seguendo le specifiche date, ed il numero dei conflitti è sceso ma non si è azzerato; nello specifico ho notato che la totalità dei conflitti è causata dalle produzioni *StatList -> Stat* e *Stat -> /*empty*/*.
3. Ho risolto il conflitto causato da ***StatList -> Stat*** e ***Stat -> /* empty */*** rimuovendo quest'ultima produzione e cambiando la prima con ***StatList -> /*empty*/***, in questo modo invece di arrivare a Stat per andare in **/*empty*/** è possibile anticipare restituendo la lista vuota di Stat. A seguito di ciò, il numero dei conflitti si è **azzerato**.

3. PATTERN JAVA

Pattern Java per la visita dei nodi Abbiamo deciso di gestire la visita dell'AST servendoci di due interfacce: Visitor e Visitable. Le classi XMLVisitor, SemanticVisitor e CLangVisitor implementano un'interfaccia Visitor che obbliga queste ad implementare un metodo visit(). Queste classi sono quelle che "visitano" l'AST. Le classi corrispondenti ai nodi dell'AST invece implementano un'interfaccia Visitable che obbliga loro ad implementare un metodo accept(). Queste classi sono quelle "visitate". Prendendo in esame un nodo che si vuole visitare, si chiama su di esso il metodo accept() passandogli il visitor. All'interno del metodo accept() si va a chiamare il metodo visit del visitor passato fornendogli in input il nodo stesso.

4. Analisi semantica

4.1 Regole di type checking implementate nel formato regole di inferenza

$\forall\, \tau_i \in$ {bool, integer, real, string}, con $i \in \mathbb{N}$, sono definite le seguenti regole di type checking:

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

$$\Gamma \vdash \text{true} : \text{boolean}$$

$$\Gamma \vdash \text{false} : \text{boolean}$$

$$\Gamma \vdash \text{integer\_const} : \text{integer}$$

$$\Gamma \vdash \text{real\_const} : \text{real}$$

$$\Gamma \vdash \text{string\_const} : \text{string}$$

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

$$\frac{\Gamma \vdash f : \tau_1, \tau_2, \dots, \tau_n \to \tau \qquad \Gamma \vdash e_i : \tau_i^{\,i \in 1 \dots n}}{\Gamma \vdash f\,(e_1, e_2, \dots, e_n) : \tau}$$

$$\frac{\Gamma \vdash f : \tau_1, \tau_2, \dots, \tau_n \to notype \quad \Gamma \vdash e_i : \tau_i^{\,i \in 1 \dots n}}{\Gamma \vdash f\,(e_1, e_2, \dots, e_n) : notype}$$

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

$$\frac{\Gamma(id) = \tau \quad stmt : notype}{\Gamma \vdash id; stmt : notype}$$

$$\frac{\Gamma \vdash e : \mathit{boolean} \quad \Gamma \vdash \mathit{block} : \mathit{notype}}{\Gamma \vdash \textbf{while}\ \ e\ \textbf{loop}\ \mathit{block}\ \textbf{end loop} : \mathit{notype}}$$

$$\frac{\Gamma \vdash e : \ \mathit{boolean} \quad \Gamma \vdash \mathit{block} : \ \mathit{notype}\ \Gamma \vdash \mathit{block}_2 : \ \mathit{notype}}{\Gamma \vdash \textbf{if}\ \ e\ \textbf{then}\ \mathit{block}\ \textbf{else}\ \mathit{block}_2\ \textbf{end if} : \mathit{notype}}$$

$$\frac{\Gamma \vdash e : \ \mathit{boolean} \quad \Gamma \vdash \mathit{block} : \ \mathit{notype}}{\Gamma \vdash \textbf{if}\ \ e\ \textbf{then}\ \mathit{block}\ \textbf{end if} : \mathit{notype}}$$

$$\frac{\Gamma \vdash e : \tau_1 \qquad \mathit{optype}1(op_1 , \tau_1) = \tau}{\Gamma \vdash \ op_1 \ \mathrm{e} \ : \tau}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \qquad \Gamma \vdash e_2 : \tau_2 \qquad \mathit{optype}2(op_2 , \tau_1 , \tau_2) = \tau}{\Gamma \vdash \ \mathrm{e}_1\, op_2 \ \mathrm{e}_2 \ : \tau}$$

$$\frac{(x_1 : \tau_1) \in \Gamma \quad (x_2 : \tau_2) \in \Gamma \ \ldots \ (x_n : \tau_n) \in \Gamma \quad \mathrm{n} \in \mathrm{N}}{\Gamma \vdash \textbf{readln}\ (x_1 , \ x_2 , \ldots , \ x_n)}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \qquad \Gamma \vdash e_2 : \tau_2 \ \ldots \ \Gamma \vdash e_n : \tau_n \quad \mathrm{n} \in \mathrm{N}}{\Gamma \vdash \textbf{write}\ (e_1 , \ e_2 , \ldots , \ e_n)}$$

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

4.2 Typing relations

Op_1	Operando	Risultato
-	integer	integer
-	real	real
!	boolean	boolean

Op_2	$Operando_1$	$Operando_2$	Risultato
+ - * / ^	integer	integer	integer
+ - * / ^	integer	real	real
+ - * / ^	real	integer	real
+ - * / ^	real	real	real
div int	integer	integer	integer
div int	integer	real	integer
div int	real	integer	integer
div int	real	real	integer
&& ||	bool	bool	boolean
= <> < <= > >=	bool	bool	boolean
= <> < <= > >=	integer	integer	boolean
= <> < <= > >=	integer	real	boolean
= <> < <= > >=	real	integer	boolean
= <> < <= > >=	real	real	boolean
= <> < <= > >=	string	string	boolean

String Concat	string	string	string
String Concat	string	real	string
String Concat	real	string	string
String Concat	string	integer	string
String Concat	integer	string	string
String Concat	string	boolean	string
String Concat	boolean	string	string

Tabella di compatibilità tra tipi

Tipo	Tipi compatibili	
integer	integer	
real	real	integer
boolean	boolean	
string	string	