Alma Mater Studiorum - University of Bologna

LM Informatica

SimplanPlus Compiler

Project of the course "Compilatori & Interpreti"

Riccardo Preite, Davide Davoli, Matteo Mele, Leonardo Palumbo

A.Y. 2020/2021

Abstract

In this paper we present a **social aware** platform combined with **human activity recognition anonymization of sensitive data** of the user. Thanks to the combination of these three principles we have built an application, with private and public aspects, for places recommendation thanks to a social sharing of places, an human activity recognizer and a place prediction model.

Contents

1 Introduzione

1.1 Obiettivo

1.2 SimplanPlus

1.3 Compilatore

1.4 Interprete

1.5 Stato dell’arte

2 Grammatica

Ogni linguaggio è composto da una grammatica che definisce la struttura delle sue istruzioni, delle espressioni e dei suioi tipi. La grammatica di **SimplanPlus**, presente in `lexer/SimpLanPlus.g4`, presenta una serie di regole:

- Block;
- Declaration;
- Statement;
- Expression;
- White space, comment;
- Errors.

Grazie a `lexer/SimpLanPlus.g4` il lexer e il parser sono stati generati automaticamente. Ovviamente si è dovuto procedere all'implementazione finale del parser per fare in modo che possa creare i giusti nodi a partire dal contesto che riceve.

2.1 Blocchi

Un programma SimplanPlus inizia con un block che potrebbe avere una serie di dichiarazioni (variabili o funzioni) seguita da una possibile serie di istruzioni. Il blocco è considerato un'istruzione quindi potrebbe ripetersi più volte all'interno del programma.

```
block       : '{' declaration* statement* '}';
```

2.2 Dichiarazioni

L'utente ha la possibilita di dichiarare sia variabili, int o bool o puntatori ad int, bool o altre zone di memoria, sia funzioni di tipo int, bool o void. Duranta la dichiarazione di variabili è possibile assegnare un valore tramite un'espressione sul rhs. Le funzioni permettono l'utilizzo di parametri e presentanto un blocco nel quale scrivere le dichiarazioni e le istruzioni.

```
declaration : decFun
            | decVar ;

decFun      : (type | 'void') ID '(' (arg (',' arg)*)? ')' block ;

decVar      : type ID ('=' exp)? ';';

type        : 'int'
            | 'bool'
            | '^' type;

arg         : type ID;
```

2.3 Istruzioni

Il linguaggio SimplanPlus presenta una serie di istruzioni che permettono la manipolazione di ID o chiamate di funzioni.

- (`assignment`) di un valore ad una variabile dichiarata;
- (`deletion`), viene deallocata la zona di memoria alla quale il puntatore puntava;
- (`print`), di una espressione (variabile, ritorno di funzione o int o bool);
- (`ret`) permette ad una funzione di ritornare un valore o di ritornare al chiamante in caso di funzioni void;
- (`ite`) definisce la costruzione di un blocco if con condizione booleana e ramo else facoltativo;
- (`call`) invoca la funzione corrispondente con i relativi parametri attuali;
- (`block`) rappresenta la creazione di un blocco annidato.

```
statement   : assignment ';'
            | deletion ';'
            | print ';'
            | ret ';'
            | ite
            | call ';'
            | block;

assignment  : lhs '=' exp;

lhs         : ID | lhs '^';

deletion    : 'delete' ID;

print       : 'print' exp;

ret         : 'return' (exp)?;

ite         : 'if' '(' exp ')' statement ('else' statement)?;

call        : ID '(' (exp(',' exp)*)? ')';
```

3 Espressioni

Le espressioni possono essere utilizzate per assegnare valori ad una variabile, in condizioni booleane o anche come argomenti di una funzione.

```
exp         : '(' exp ')'                                                    #
    baseExp
     | '-' exp                                                          #negExp
     | '!' exp                                                #notExp
```

```
    | lhs                                                                                    #
      derExp
    | 'new' type                                                                          #newExp
    | left=exp op=('*' | '/')                right=exp   #binExp
    | left=exp op=('+' | '-')                right=exp   #binExp
    | left=exp op=('<' | '<=' | '>' | '>=') right=exp   #binExp
    | left=exp op=('==' | '!=')              right=exp   #binExp
    | left=exp op='&&'                       right=exp   #binExp
    | left=exp op='||'                       right=exp   #binExp
    | call                                               #callExp
    | BOOL                                               #boolExp
    | NUMBER                                                                  #valExp;

//Booleans
BOOL        : 'true'|'false';

//IDs
fragment CHAR       : 'a'..'z' |'A'..'Z' ;
ID          : CHAR (CHAR | DIGIT)* ;

//Numbers
fragment DIGIT      : '0'..'9';
NUMBER      : DIGIT+;
```

3.1 Commenti e caratteri ignorati

3.2 Errori grammaticali

4 Semantica

4.1 Ambiente

4.2 Errori semantici

5 Controllo dei tipi

5.1 Tipi

6 Analisi degli effetti

6.1 Struttura ambiente

6.2 Effetti

7 Compilatore

7.1 Compilatore SimplanPlus

8 Interprete

8.1 Interprete SimplanPlus

9 Esempi

9.1 Esempio 1