Report del Progetto di Compilatori e Interpreti

Giuseppe De Palma XXXXXX, Eduart Uzeir XXXXXX
Domenico Coriale XXXXXX e Andrew Memma XXXXXX

Corso di Laurea Magistrale in Informatica 2017/2018

Data Consegna

Indice

0.1 Introduzione 2
0.2 Descrizione del Progetto 2
0.3 Utilizzo 3

1 Analisi Preliminare **4**

2 Sintassi **5**

3 Semantica **6**
3.1 Type Checking 6

4 Generazione del Codice **7**
4.1 Generazione del Codice Intermedia 7
4.2 Generazione del Codice 7

5 Garbage Collection **8**

6 Conclusioni **9**

1 Introduzione

Breve capitolo introduttivo sul progetto. Viene discusso l'obiettivo generale del progetto e la base da cui si è partiti.

Il lavoro presentato riguarda un compilatore per un semplice linguaggio di programmazione imperativo: FOOL. L'obiettivo di tale progetto è puramente accademico, volto allo studio sullo sviluppo e funzionamento dei compilatori.
Il progetto è parte di esame del corso "Compilatori e Intepreti" della Magistrale in Informatica dell'università di Bologna. Il gruppo di lavoro, composto da quattro studenti del suddetto corso, si è dedicato allo sviluppo durante il periodo di Luglio e Agosto 2018.

2 Descrizione del Progetto

Si discute brevemente la struttura dei package e delle classi, l'implementazione etc.

FOOL è un linguaggio imperativo molto basilare che è stato parzialmente esteso con il paradigma orientato a oggetti. Questo progetto è un esercizio puramente accademico sull'implementazione di un compilatore/interprete per un linguaggio di programmazione semplificato, FOOL. Il lavoro è stato svolto partendo da una base giá funzionante e fornita dal professore, che è stata poi estesa con le caratteristiche richieste. Il progetto si basa sul tool per la generazione automatica di parser Antlr, nella sua versione 4.7. Nella sua versione iniziale FOOL prevedeva le operazioni di base di un linguaggio di programmazione imperativo (definizione e utilizzo di simboli e funzioni). Nel corso del progetto è stato esteso parzialmente con il paradigma orientato a oggetti, dando la possibilitá di definire classi e istanziarle a runtime. Si tratta appunto di un esercizio di stile senza alcuna finalitá pratica, in quanto il linguaggio si basa su programmi e funzioni composti da una singola istruzione effettiva e utilizza variabili dal valore non modificabile (di fatto soltanto costanti). Essendo queste limitazioni derivanti direttamente dalla grammatica BNF stabilita dalla consegna sono state intese come imposizioni e non modificate.

2.1 Utilizzo

Si spiega con dettaglio come lanciare e utilizzare il progetto.

3 Analisi Preliminare

4 Sintassi

Punto di inizio dello sviluppo. Si discute l'analisi lessicale. La grammatica, la generazione di token. Non dovrebbe esserci molto da dire siccome la grammatica è già praticamente data e antlr automizza molto.
Si discute l'analisi sintattica, l'AST etc. Anche qui sarà breve.
Si discute anche l'implementazione dell'AST e le altre parti inerenti alla sintassi.

5 Semantica

Si discute brevemente lo scopo dell'analisi semantica (attraversare l'albero di sintassi astratta costruito prima, gestione degli scope e controllo tipi).
Si spiega brevemente le decisioni di design per le symbol tables ed il sistema dei tipi (statico).

5.1 Type Checking

sezione importantissimo, qui è dove bisogna essere i più precisi e formali. Si discute per bene lo sviluppo e implementazione del type checking.

6 Generazione del Codice

Capitolo sulla code generation. Si discute come si trasforma il codice scritto nella grammatica ad alto livello di FOOL a codice macchina. Come si gestisce lo stack, i pointers, i frames etc etc.

Magari si può dividere questo capitolo con le due sezioni sotto, dipende da come implementiamo la code generation. Se la facciamo in due fasi passando prima per una generazione ad un codice intermedio e poi al bytecode, allora questo capitolo si scrive con le due sezioni sotto. La generazione intermedia si fa quando si usa anche un interprete, dipende da come vogliamo fare.

6.1 Generazione del Codice Intermedia

6.2 Generazione del Codice

7 Garbage Collection

Opzionalmente possiamo fare la garbage collection e verrà discussa qui.

8 Conclusioni

Brevissimo capitolo conclusivo, si fa un piccolo riassunto di tutto il report e si tirano le somme.