A.A. 2000-01

Linguaggi di Programmazione I: Paradigmi

1 Esercizi

1.1 Sintassi e semantica

1.1.1

Sia G la seguente grammatica:

$$\begin{array}{rcl} \mathit{exp} & ::= & \mathit{shorty} + \mathit{shorty} \mid \mathit{shorty} \\ \mathit{shorty} & ::= & \mathit{bit} + \mathit{bit} \mid \mathit{bit} \\ \mathit{bit} & ::= & 0 \mid 1 \end{array}$$

Quanti sono gli elementi del linguaggio *exp* definito dalla sintassi *concreta* G? Quanti sono i termini dell'insieme *exp* definito induttivamente dalla sintassi *astratta* G?

1.2 Scoping e passaggio dei parametri

1.2.1

Considerate un linguaggio di programmazione funzionale nel quale, come in ML, la dichiarazione (ed inizializzazione) di una variabile venga specificata mediante la costruzione sintattica $\mathit{let}\, x = e_1 \ \mathit{in}\, e_2$. Quale sarebbe il valore dell'espressione

$$\mathit{let}\ x = 5\ \mathit{in}\ (\mathit{let}\ x = x + 1\ \mathit{in}\ (\mathit{let}\ y = (\mathit{fn}\ z \Rightarrow x + z)\ \mathit{in}\ (\mathit{let}\ x = 7\ \mathit{in}\ y(3))))$$

se il linguaggio adottasse regole di scope *statico*? Quale nel caso invece in cui lo scoping fosse *dinamico*?

1.2.2

Considerate un linguaggio di programmazione imperativo nel quale la dichiarazione (ed inizializzazione) di una variabile venga specificata mediante la costruzione sintattica $\mathit{var}\ x = e\ \mathit{in}\ p$, e la dichiarazione di una procedura venga specificata con $\mathit{proc}\ P(x)\ \mathit{is}\ p_1\ \mathit{in}\ p_2$. Mostrare formalmente, usando una semantica operazionale *call-by-value*, il risultato dell'esecuzione del programma

$$\mathit{var}\ x = 6\ \mathit{in}\ (\mathit{proc}\ y(z)\ \mathit{is}\ r := x + z\ \mathit{in}\ (\mathit{let}\ x = 7\ \mathit{in}\ y(3)))$$

adottando prima regole di scope *statico* e poi *dinamico*.

1.2.3

Considerate un linguaggio di programmazione imperativo con assegnamenti, dichiarazione di variabili e procedure come sopra. Mostrare formalmente, usando la semantica operazionale, il risultato dell'esecuzione del programma

$$var\ x = 6\ in\ (proc\ y(z)\ is\ (x := x + 1; r := z)\ in\ y(x))$$

prima con regole di passaggio dei parametri *per valore* e poi per *nome*.

1.3 Polimorfismo

1.3.1

Siano S_1 ed S_2 due sostituzioni di tipi. La composizione $S_2 \cdot S_1$ è quella sostituzione che, applicata ad un qualunque tipo T restituisce $S_2(S_1(T))$. Scrivere esplicitamente $S_2 \cdot S_1$ supponendo che S_1 sia $\{X = X \to Y\}$ ed S_2 sia $\{X = Z, Y = V\}$.

1.3.2

Calcolare $\mathcal{W}((x : X, y : \forall Y . Y), (y(x))(y(x)))$ usando l'algoritmo $\mathcal{W}$ di inferenza dei tipi di ML.

1.4 Moduli di programma

1.4.1

(Meta-esercizio) Svolgere gli esercizi del capitolo 3.2 e 3.4 delle dispense in http://www.cs.cmu.edu/ rwh/papers/ml-notes/tr.ps.

1.5 Le pseudovariabili *this* e *super*

1.5.1

Qual è l'effetto della esecuzione del seguente programma Java?

```
class A {

    public void m () {
System.out.println("A");
    }
}

class B extends A {

    public void m () {
System.out.println("B");
    }
```

```
    public void xxx () {
super.m();
this.m();
    }
}

class C extends B {

    public void m () {
System.out.println("C");
    }
}

public class Esempio {

    public static void main (String [] args) {
B pippo = new C();
pippo.xxx();
    }
}
```

2 Soluzioni

1.1.1 Una stringa del linguaggio *exp* (sintassi concreta) è una sequenza di n bit inframezzati da "+", dove n va da 1 a 4. Vi sono 2 di tali stringhe per $n = 1$, 4 per $n = 2$, 8 per $n = 3$ e 16 per $n = 4$, per un totale di 30 stringhe diverse.

L'insieme *exp* definito dalla sintassi astratta contiene $n * n + n$ elementi, dove n è il numero degli elementi di *shorty*. Quest'ultimo contiene $m * m + m$ elementi, dove m è il numero degli elementi di *bit*, cioè due. Il resto è aritmetica.