

Grundlagen der Programmiersprachen und Compilerbau

Wintersemester 2007/2008

6. Übungsblatt

15. Januar 2008

Aufgabe 6.1

Vergleichen Sie die Programmausführung durch einen Interpreter mit den Vorgängen bei Verwendung eines Übersetzers.

- Welche Vor- und Nachteile haben die beiden Ansätze?
- Nennen Sie Beispiele für Programmiersprachen, die typischerweise durch Interpreter, durch Übersetzer oder durch Mischformen der beiden Ansätze implementiert werden.
- Sei p ein Programm (Code), dann bezeichnet $\llbracket p \rrbracket(x, y)$ die ausführbare Funktion (Semantik-Funktion) mit den Parametern x und y . Geben Sie $\llbracket \text{Interpreter} \rrbracket(p, d)$ und $\llbracket \text{Compiler} \rrbracket(p)$ an.

Aufgabe 6.2

Bei der *partiellen Auswertung* eines in einer Quellsprache \mathcal{L} geschriebenen Programms p bezüglich bekannter Eingabedaten d_1, d_2, \dots, d_n und unbekannter Eingabedaten u_1, \dots, u_m versucht man durch teilweise Abarbeitung von p ein neues Programm p' zu konstruieren, so dass $\llbracket p' \rrbracket(u_1, \dots, u_m) = \llbracket p \rrbracket(d_1, \dots, d_n, u_1, \dots, u_m)$ gilt.

Alle Befehle, die aufgrund der Abhängigkeit von unbekanntem Eingabedaten noch nicht ausgeführt werden können, werden zurückgestellt und erst in einer späteren Phase abgearbeitet. Das entstandene Programm p' heißt *residuales Programm* zu p . Ein Programm, das aus einem gegebenen Programm p zu bekannten Eingabedaten ein residuales Programm p' erzeugt, heißt *partieller Auswerter* oder *Spezialisierer*.

- Beschreiben Sie einen Spezialisierer mittels $\llbracket \cdot \rrbracket(\cdot)$ -Notation.
- Geben Sie die Spezialisierung eines Interpreters für beliebige Eingaben mithilfe eines partiellen Auswerters an. Benutzen Sie die $\llbracket \cdot \rrbracket(\cdot)$ -Notation.
- Untersuchen Sie die (evtl. mehrmalige) Spezialisierung des partiellen Auswerters selbst.
- Existiert immer für ein Programm der Programmiersprache \mathcal{L} ein partieller Auswerter?

Aufgabe 6.3

Gegeben sei das folgende Ada95 Programm. Die Parameterübergabe sei „by-value“. Der Compiler implementiert die Methodik des „Reference Counting“ (über eine für den Benutzer transparente Zählerkomponente in den Haldenobjekten).

Das Programm erzeugt insgesamt vier Haldenobjekte O1 bis O4 an den im Code kommentierten Stellen. (Guter Rat: Machen Sie sich Skizzen der Haldenzustände.)

```
( 0) type Node;
( 1) type Node_Acc is access Node;
( 2) type Node is
( 3)     record
( 4)         Content : Integer;
( 5)         Next    : Node_Acc;
( 6)     end record;
( 7) procedure test is
( 8)     P,Q,R : Node_Acc;
( 9)     procedure Inner ( A, B, C : Node_Acc ) is
(10)     begin
(11)         declare
(12)             S : Node_Acc := new Node'(8, C); -- Objekt O4
(13)         begin
(14)             P := B;
(15)             S := A;
(16)             B.Next := C;
(17)             ... -- Verwendung, aber keine Änderung der Zeiger
(18)         end;
(19)             ... -- Verwendung, aber keine Änderung der Zeiger
(20)         end Inner;
(21)     begin
(22)         P := new Node'(5, null); -- Objekt O1
(23)         Q := new Node'(6, null); -- Objekt O2
(24)         P.Next := Q;
(25)         R := new Node'(7, Q); -- Objekt O3
(26)         Inner(P, Q, R);
(27)         return;
(28)     end test;
(29) test;
```

- a) Geben Sie in der folgenden Tabelle den momentanen Wert des Referenzzählers für jedes der Objekte an, nachdem die genannte Zeile und alle durch sie implizierten Aktionen der „Reference Counting“ Methode ausgeführt wurden.

	O1	O2	O3	O4
nach Zeile (16):				
nach Zeile (18):				
nach Zeile (26):				

- b) Geben Sie ggf. die Nummern der Zeilen an, nach deren Ausführung die Reference-Counting Methode aufgrund eines Referenzzählers mit dem Wert 0 das entsprechende Objekt freigibt. Erfolgt keine Freigabe, markieren sie dies mit „X“.

Freigabe von	O1	O2	O3	O4
nach Zeile:				