Universität Karlsruhe (TH)

Institut für Innovatives Rechnen und Programmstrukturen (IPD)

Übersetzerbau WS 2003/04 http://www.info.uni-karlsruhe.de/

Dozent: Prof. Dr.rer.nat. G. Goos goos@ipd.info.uni-karlsruhe.de

Übungsleiter: Rubino Geiß rubino@ipd.info.uni-karlsruhe.de

Übungsblatt 5 Ausgabe: 09.12.2003 Besprechung: 18.12.2003

Aufgabe 1: *Strukturäquivalenz*

Gegeben seien die folgenden Typdefinitionen:

```
type t = record
    a: INT;
    b: ^t;
    c: ARRAY[1..7] of tt
end;

type t'= record
    a: INT;
    b: ^record a: INT; b: ^t' end;
    c:ARRAY[1..7] of tt'
end};

type tt'= record
    a: INT;
    b: ^record a: INT; b: ^tt end
end;

type tt = record
    a: INT;
    b: ^tt'
end};
```

1.1 *NEA bauen*

Konstruieren Sie für jeden der beiden Typen t und t' einen endlichen Automaten, so dass aus der Gleichheit der Automaten die Strukturäquivalenz der Typen folgt!

1.2 *NEA vergleichen*

Sind beide Typen strukturäquivalent?

Aufgabe 2: *Überladen von Operatoren*

Gegeben sei die folgende Formelgrammatik (konkrete Syntax)

$$E ::= E + T \mid T \qquad T ::= T * id \mid id$$

und die zugehörige Grammatik, welche die abstrakte Syntax (d.h. den Strukturbaum) beschreibt:

(1) $S ::= DE$

(2) $E ::= EE$

(3) $E ::= id$

Der Operator ('+' oder '*') sei vom Zerteiler bestimmt und ist durch ein intrinsic Attribut *E.opsym* gegeben. Benutzen Sie zur Typbestimmung die Attribute *primode*, die aus der Definition von Operanden abgeleitet werden, und *postmode*, die vom Kontext gefordert werden. Es gibt die Typen *inttype*, *realtype*, *booltype*, *stringtype* und die Operationen *intplus*, *realplus*, *boolplus* (OR), *stringplus* (concatenation), *inttimes*, *realtimes*, *booltimes* (AND). Der Typ eines Bezeichners *id.type* wird durch die Umgebung (*E.env*) mit Hilfe der Funktion *identify (id.sym, E.env)* bestimmt. *D stehe für Produktionen in denen Variablen deklariert werden.*

2.1 AG für Typanpassung

*Erweitern Sie den AST um Attributierungsregeln, die die Typen der Operatoren (*E.op*) aus den Typen der Operanden bestimmen. Benutzen Sie dazu die Funktion* coercible *bzw.* balance*, die die Anpassbarkeit zweier Typen überprüft bzw. den balancierten Typ zurückgibt.*

2.2 Funktionen für Typanpassung

Es seien folgende Anpassungen möglich:

- inttype → realtype
- inttype, realtype *und* booltype → stringtype

Entwerfen Sie die Funktionen balance *und* coercible*.*

2.3 Typanpassung unter Einbeziehung der Zuweisung

Nun entwerfen Sie analoges für folgende Erweiterung der Grammatik unter Annahme mehrdeutiger Ergebnisse von identify *(der Funktion, die die Definition eines Symbols ermittelt).*

(1) S	::=	DA	*Das unvollständige D stehe für Deklarationen*
(2) A	::=	AA	*Ein A ist ein Anweisung*
(3) A	::=	idE	*Zuweisung*
(4) E	::=	EE	
(5) E	::=	id	

Aufgabe 3: *Konturmodell*

Betrachten Sie folgendes Programm:

```
 1 program outer;
 2   var n,k : integer;
 3   procedure p ( procedure f;  var j : integer);
 4     label 1;
 5     var i : integer;
 6     procedure q;
 7       label 2;
 8     begin (* q *)
 9       n := n + 1;  if n = 4  then q;
10       n := n + 1;  if n = 7  then 2: j := j + 1;
11       i := i + 1;
12      end; (* q *)
13    begin (* p *)
14     i := 0;
15     n := n + 1;  if n = 2  then p (q, i)  else j := j + 1;
16      if n = 3  then 1: f;
17     i := i + 1;
18     writeln ('i = ', i:1);
19    end; (* p *)
20    procedure empty;  begin end; (* empty *)
21 begin (* outer *)
22  n := 1; k := 0;
23  p (empty, k);
24  writeln ('k = ', k:1);
25 end; (* outer *)
```

3.1 *Mysterien*

Welche Werte werden ausgegeben?

3.2 *Impelmentierung*

Überlegen Sie sich, wie der Zugriff auf Variablen des statischen Vorgängers implementiert werden kann.

Aufgabe 4: *Typsystem*

Beantworten Sie mit den Java-Quellen
`http://www.info.uni-karlsruhe.de/lehre/2003WS/uebau1/aufgaben/TypeRules.java` als Grundlage folgende Fragen:

4.1 *Ausprobieren*

Was ist *balance*, *coercible* und *equivalent* von

$$(proc\ proc\ real, ref\ proc\ ref\ int)$$

4.2 *Programmieren*

Erweitern Sie das Programm so, dass die balancierten Typen in obiger Notation ausgegeben werden können.

4.3 *Nachdenken*

Wie kann man die strukturelle Äquivalenz von Verbunden ausdrücken bzw. implementieren?