| Name | Kennzahl | Matrikelnummer |
|---|---|---|
| | | |

| | |
|---|---|
| | Bitte tragen Sie sofort mit Kugelschreiber<br>*Name*, *Kennzahl* und *Matrikelnummer* ein!<br>Halten Sie bitte den Studentenausweis bereit!<br>Lesen Sie die Angaben genau, bevor Sie beginnen!<br>Verwenden Sie bitte nur diesen Bogen für Ihre Antworten!<br>Die Prüfungsdauer beträgt 90 Minuten. |
| **Summe**<br>1.Bsp.<br>2.Bsp.<br>3.Bsp.<br>4.Bsp. | **Institut für Computersprachen (Prof. Brockhaus)** |

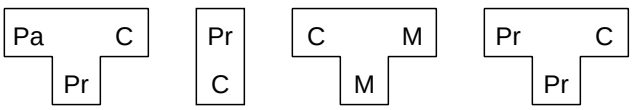
Ich trete zum ... mal zur Prüfung an.

# Prüfung aus Übersetzerbau 23.6.1995

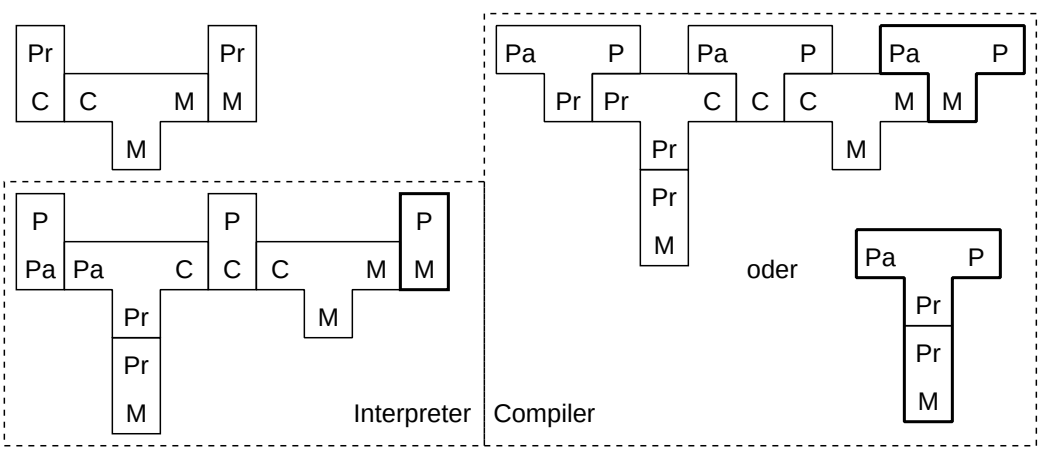
**1. 20 %**

Sie verfügen über einen in Prolog geschriebenen Pascal-Compiler, der C erzeugt, einen in C geschriebenen Prolog-Interpreter, einen C-Compiler für die Maschine M und einen in Prolog geschriebenen Prolog-Compiler, der C erzeugt.

**a) (5 %)** Stellen Sie die beschriebenen Compiler und Interpreter mit T- und I-Diagrammen dar.




**b) (15 %)** Sie haben einen in Prolog geschriebenen Pascal-Compiler, der P-Code erzeugt, und einen in Pascal geschriebenen P-Code Interpreter. Wie können diese mit Hilfe der vorhandenen Compiler und Interpreter übersetzt werden, so daß sie auf der Maschine M lauffähig sind? Stellen Sie die Vorgangsweise mittels T- und I-Diagrammen dar.



**2. 30 %** Gegeben sei folgende Grammatik, die aus verschachtelten indizierten Variablen aufgebaute Ausdrücke beschreibt:

$S \to E$
$E \to$ `id [` $L$ `]` | `num`
$L \to L$ `","` $E \mid E$

Erweitern Sie diese oder eine äquivalente Grammatik zu einer attributierten Grammatik, die die Übersetzung der Ausdrücke in Quadrupelcode beschreibt, der den Wert des Ausdrucks berechnet. Der Quadrupelcode soll im synthetisierten Attribut $S$.`code` generiert werden. Das Attribut `num.val` enthält den numerischen Wert der Zahl, `id.sym` den Namen der Arrayvariable. Zur Bestimmung der Anzahl der Elemente der $i$-ten Dimension eines Arrays $sym$ steht die Funktion `dim(`$sym, i$`)` zur Verfügung. Die Funktion `newtemp` liefert eine neu erzeugte Hilfsvariable zurück.

$S \to E$
```
S.code := E.c
S.v := E.v
```

$E \to$ `id [` $L$ `]`
```
L.sym := id.sym
t0 := newtemp
t1 := newtemp
E.v := newtemp
E.c := L.c||gen(t0 ':=' L.v '*' 4)||
  gen(t1 ':=' t0 '+adr(' id.sym ')')||gen(E.v ':=' '@' t1)
```

$E \to$ `num`
```
E.v := num.val
E.c := ''
```

$L \to L_1$ `","` $E$
```
L1.sym := L.sym
L.i := L1.i + 1
t1 := newtemp
L.v := newtemp
L.c := L1.c||E.c||gen(t1 ':=' L1.v '*' dim(L.sym,L.i))||
  gen(L.v ':=' t1 '+' E.v)
```

$L \to E$
```
L.i := 1
L.v := E.v
L.c := E.c
```

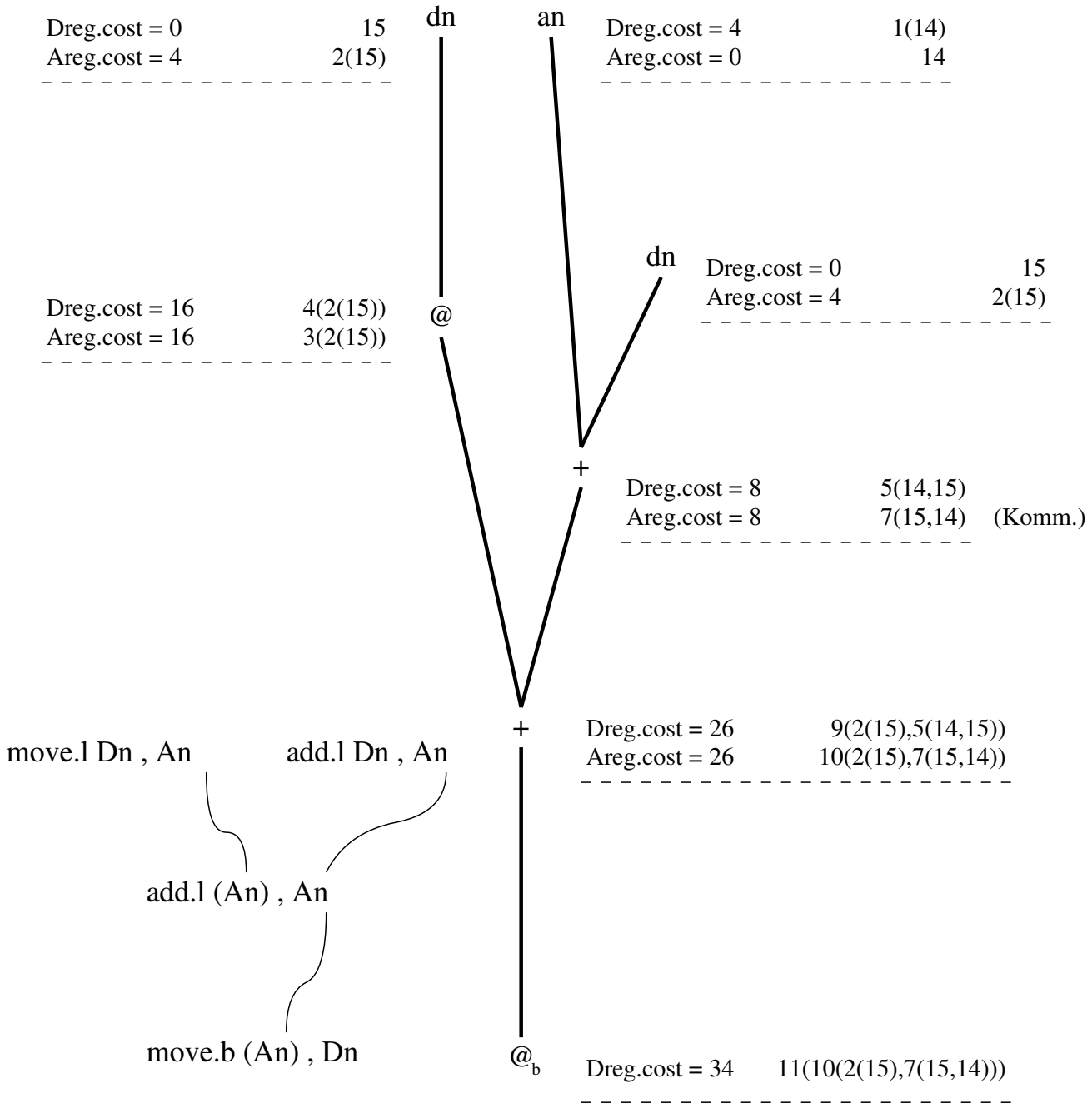
Durch Umformung der Grammatik könnten die ererbten Attribute eliminiert werden.

**3. 25 %** Gegeben sei folgendes Modula-Programmstück:

```
VAR
  a: ARRAY[0..N] OF INTEGER; b: ARRAY[1..M] OF INTEGER;
  n,j,k: INTEGER;
...
WHILE a[n+j] > k DO
  IF n < 8-j THEN
    n := n + 1;
  ELSE
    k := b[n];
  END
END;
```

Erzeugen Sie für das obige Programmstück Quadrupel-Code nach der Kontrollflußmethode.

```
Start:
    t1 := n+j
    t2 := 4*t1
    t3 := t2+adr(a)
    t4 := @t3
    if t4>k goto L1
    goto Ende
L1: t5 := 8-j (* In der WHILE-Schleife *)
    if n<t5 goto L2
    goto L3
L2: t6 := n+1 (* THEN-Zweig *)
    n := t6
    goto L4
L3: t7 := n-1 (* ELSE-Zweig *)
    t8 := t7*4
    t9 := t8+adr(b)
    t10 := @t9
    k := t10
L4: goto Start
Ende:
```

**4. 25 %**

Erzeugen Sie für den folgenden Zwischencode-Baum optimalen Maschinencode für den 68000, so daß das Ergebnis in einem Datenregister landet. Verwenden Sie dazu die im Skriptum angegebene Befehlsauswahl-Baumgrammatik.