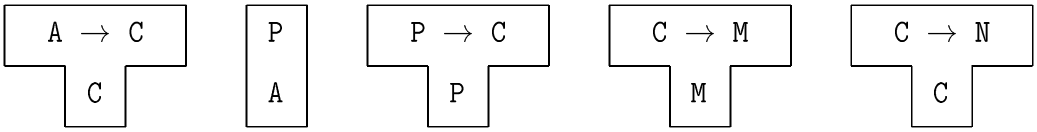
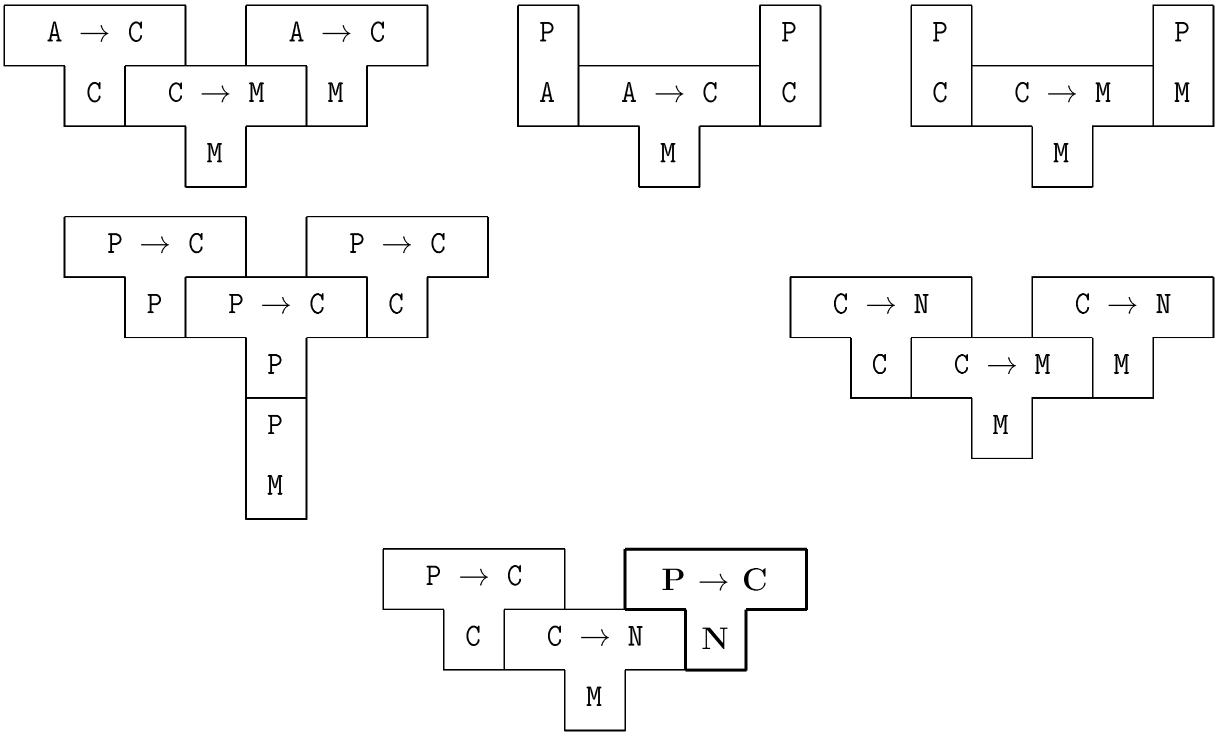
Prüfung aus Übersetzerbau 11.11.1994 Musterlösung

1. 20 % Sie verfügen über einen in C geschriebenen Algol-Compiler, der C-Code erzeugt, über einen in Algol geschriebenen Prologinterpreter und einen in Prolog geschriebenen Prolog-Compiler, der C-Code erzeugt. Außerdem gibt es einen C-Compiler für die Maschine M, der Maschinencode für M erzeugt und einen in C geschriebenen C-Compiler, der Code für die Maschine N erzeugt.

a) (5 %) Stellen Sie die vorhandenen Compiler und Interpreter mittels T- und I-Diagrammen dar.



b) (15 %) Erzeugen Sie (mittels T- und I-Diagrammen) einen Prolog-Compiler für die Maschine N, der C-Code erzeugt.



2. 30 %

Gegeben sind arithmetische Infix-Ausdrücke mit Variablen als Operanden und den Operatoren `+` und `-` (deren Operanden von links nach rechts ausgewertet werden) sowie Klammerung `(`,`)` mit der üblichen Bedeutung. Zusätzlich können in eckige Klammern geschriebene Variablenzuweisungen als Operanden auftreten. Auf der rechten Seite steht wieder ein Ausdruck und auf der linken Seite eine Variable. Der Ausdruck wird wie üblich ausgewertet und sein Wert wird der links stehenden Variable als Seiteneffekt zugewiesen. Dieser Wert ist gleichzeitig auch das Ergebnis des Operanden.

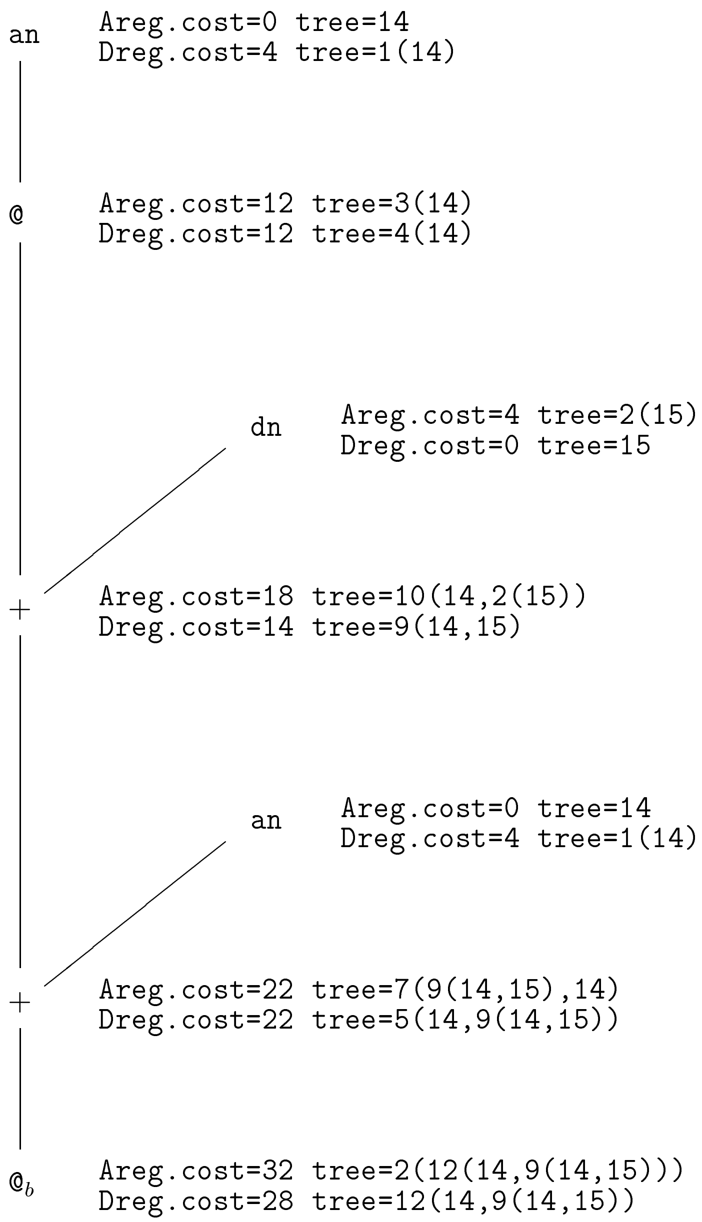
Beispiel: Nehmen Sie folgende Variableninitialisierungen an: `a=3`, `b=4`, `c=2`. Dann liegt nach Auswertung des Ausdrucks `b+[b=a-c+[d=a+c]-b]` folgende Variablenbelegung vor: `a=3`, `b=2`, `c=2`, `d=5`. Der Wert des gesamten Ausdrucks ist 6.

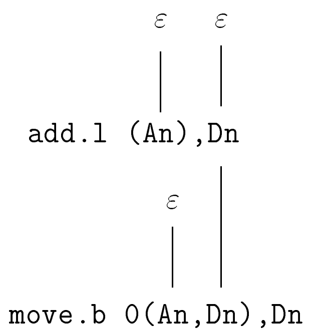
a) (10 %) Geben Sie eine Grammatik für die oben beschriebenen Ausdrücke an. Eingabe sind die Token `+`, `-`, `=`, `(`, `)`, `[`, `]` und `id`. Im Attribut `id.x` steht der Name der Variable.

b) (20 %) Erweitern Sie Ihre Grammatik zu einer Attributierten Grammatik, die die Ausdrücke berechnet und das Ergebnis in einem Attribut `val` des Startsymbols zur Verfügung stellt. Das Startsymbol erbt von seiner Umgebung ein Attribut `vbin`, das eine Tabelle mit den aktuellen Variablenbindungen enthält. Bei der Abarbeitung wird eine neue Tabelle erzeugt, die die neuen Variablenbindungen beschreibt. Diese soll im synthetisierten Attribut `vbout` des Startsymbols zurückgegeben werden. Um den aktuellen Wert einer Variablen aus der Tabelle zu bekommen, steht die Funktion `getval(tbl,name)` zur Verfügung, die als Argumente die Variablentabelle `tbl` und den Namen einer Variablen erwartet und für diese den aktuellen Wert zurückliefert. Weiters gibt es die Funktion `putval(tbl,name,val)`, die der Variablen `name` in der Tabelle `tbl` den neuen Wert `val` zuweist und die auf diese Art veränderte Tabelle als Ergebnis zurückliefert.

a)	b)
E → E_1+T	E_1.vbin:=E.vbin; T.vbin:=E_1.vbout; E.vbout:=T.vbout; E.val:=E_1.val+T.val;
E → E_1−T	E_1.vbin:=E.vbin; T.vbin:=E_1.vbout; E.vbout:=T.vbout; E.val:=E_1.val−T.val;
E → T	T.vbin:=E.vbin; E.vbout:=T.vbout; E.val:=T.val;
T → id	T.vbout:=T.vbin; T.val:=getval(T.vbin,id.x);
T → (E)	E.vbin:=T.vbin; T.vbout:=E.vbout; T.val:=E.val;
T → [id=E]	E.vbin:=T.vbin; T.vbout:=putval(E.vbout,id.x,E.val); T.val:=E.val;

3. 30 % Geben Sie für das C-statement `c=p[x+(*y)]` den Zwischencode-Baum an, wobei angenommen werden soll, daß `x` in einem Datenregister und `y` und `p` in einem Adreßregister liegen. `c` soll in einem Datenregister landen. `x` ist vom Typ `integer` (4 byte), `y` ist ein Zeiger (4 byte) auf `integer`, `p` ist ein `array of char` und `c` ist vom Typ `char` (1 byte). Erzeugen Sie mittels der im Skriptum angegebenen Befehlsauswahl-Baumgrammatik optimalen Maschinencode für den 68000.





4. 20 % Gegeben sei folgende LL(1)-Grammatik:

$$\begin{array}{lcl} \mathtt{S} & \rightarrow & \mathtt{id\ [\ L\ ]} \\ \mathtt{L} & \rightarrow & \mathtt{T\ R} \\ \mathtt{R} & \rightarrow & \mathtt{,\ T\ R} \mid \varepsilon \\ \mathtt{T} & \rightarrow & \mathtt{id} \mid \mathtt{num} \end{array}$$

Erstellen Sie die Analysetabelle mit Fehleraktionen (`skip`, `stop`) für die tabellengesteuerte Top-Down-Analyse.

First(S) = { id } Follow(S) = { $ }
First(L) = { id, num } Follow(L) = {] }
First(R) = { ",", ε } Follow(R) = {] }
First(T) = { id, num } Follow(T) = { ",",] }

	id	[	]	","	num	$
S	id[L]	skip	skip	skip	skip	stop
L	TR	skip	stop	skip	TR	skip
R	ε	ε	ε	,TR	ε	ε
T	id	skip	stop	stop	num	skip