

## Tentamen Vertalerbouw—22 november 2001

De gecorrigeerde tentamens zijn af te halen op 01-12-01 op de begane grond in de tentamenkast.

*Opmerkingen:*

- Schrijf netjes en duidelijk, met zwarte of blauwe pen.
- Zet op het eerste blad alle gegevens als naam, etc., en het totaal aantal ingeleverde bladen, en nummer de ingeleverde bladen.
- Lees de opgaven eerst goed door.
- Motiveer uw antwoorden.
- De opgaven zullen gewogen meetellen in het totaalcijfer, volgens de vermelde bestedingstijd.

### 1. (40 minuten)

a). Geef First en Follow voor alle nonterminals in de navolgende grammatica:

$G = (\{=, a, (, )\}, \{S, E, F\}, P, S)$ , met

$$P = \left\{ \begin{array}{l} S \rightarrow E S \\ , \quad S \rightarrow \\ , \quad E \rightarrow E = F \\ , \quad E \rightarrow F \\ , \quad F \rightarrow a \\ , \quad F \rightarrow (S) \\ \end{array} \right\}$$

b). Is de grammatica  $LL(1)$ ,  $LR(0)$ ,  $SLR(1)$ ,  $LALR(1)$ ,  $LR(1)$ ? Geef in geval van conflicten duidelijk aan waarom dit conflicten zijn.

### 2. (45 minuten)

Gegeven is een eenvoudig taaltje, dat syntactisch gespecificeerd wordt door:

$$\begin{array}{l} P : (id \rightarrow const . E) \\ , \quad E : T E' \\ , \quad E' : + T E' \\ , \quad E' : - T E' \\ , \quad E' : \\ , \quad T : F T' \\ , \quad T' : * F T' \\ , \quad T' : \\ , \quad F : id \\ , \quad F : const \end{array}$$

De symbolen  $id$  en  $const$  zijn terminalsymbolen, met gegeven (en door de scanner ingevulde) attributen  $id.Name$  (een unieke identificatie) en  $const.Val$  (de waarde).

Ziehier enkele voorbeeldzinnen uit dit taaltje:

```
(x->3 . 2*x*x+7*x-12)
(y->2 . 3+17*y-y*6*y+y*y*y)
```

De expressie  $E$  mag hooguit één identifier bevatten, en de waarde van die identifier staat vóór  $E$  gespecificeerd. Twee zinnen, die dus niet correct zijn, zijn:

```
(y->3 . 2*x*x+7*x-12)
(y->2 . 3+17*x-y*6*z+y*y)
```

Het is de bedoeling de syntaxregels te voorzien van attributen en rekenvoorschriften, zodanig dat de (integer-) waarde van expressie  $E$  wordt uitgerekend. Bovengenoemde foute zinnen moeten leiden tot een semantische foutmelding. U dient alle door U evt. gebruikte datastructuren en procedures (apart) te declareren.

### 3. (50 minuten)

Gegeven is het volgende Pascal 'programma':

```
PROGRAM tentamen;

TYPE arr = ARRAY [5..8] of integer;

VAR i,j: integer;
    a: arr;

FUNCTION f (j: integer): integer;
BEGIN ...
    f := a[j]*a[i]           (* 1 *)
END;

PROCEDURE p (i: integer; VAR r: integer);
    PROCEDURE q (VAR s: integer);
        BEGIN s := r+f(i)   (* 2 *)
        END;
    BEGIN r := a[i]+j;      (* 3 *)
        ...
        q(r)                (* 4 *)
    END;

BEGIN ...
    p(7,j);                 (* 5 *)
    ...
END (* tentamen *).
```

Voor het geheugenbeheer en de adresberekeningen worden de volgende registers gebruikt:

GP het base address van het activation record van het hoofdprogramma,  
LNB het base address van het huidige activation record, en  
LFA het adres van de eerste vrije geheugenlokatie.

Voor het overdragen van de omgeving van een aan te roepen procedure kan het register ENV worden gebruikt.

In de machineinstructies CALL label en RETURN van de doelmachine wordt impliciet gebruik gemaakt van een (aparte) return stack. U hoeft zich dus niet druk te maken over terugkeer-adressen!

Er zijn voldoende registers (R0, R1, R2, ...) voor het opslaan van de tussenresultaten.

Een functie implementeren we precies zo als een procedure, behalve dat een functie ook nog een resultaatwaarde oplevert (door een assignment aan de functie-identificer). De handigste plaats in de stack voor dit functieresultaat is nog vóór de parameters (dus ook met een negatief displacement tov. de LNB).

- a) Geef de layout van de activation records van  $f$  en  $q$ .
- b) Geef de te genereren (pseudo-)instructies voor de procedure-entry en exit van  $q$ .
- c) Geef de te genereren (pseudo-)instructies voor de 5 gemarkeerde statements. Controle op index-waarden, die buiten array grenzen gaan, is niet nodig!

#### 4. (45 minuten)

a) Verander de grammatica in som 1 zodanig dat deze aan de LL(1) conditie voldoet en dezelfde taal accepteert.

b) Schrijf een topdown parser met expliciete stapel, die de taal accepteert als beschreven door de aangepaste grammatica. Deze parser hoeft slechts fouten te signaleren.

Onderstaande declaraties mogen worden gebruikt, danwel aangepast, in de implementatie van de parser. Alle overige zaken dient U volledig te declareren.

TYPE

```
symbol      = (S,E,F,  
              lpar, rpar, equalsym, asym, eofs);  
tsymbol     = lpar..eofs;
```

VAR

```
sym: tsymbol;
```

PROCEDURE initscanner;

```
(* Initialisatie van de scanner *)
```

**PROCEDURE nextsym;**

**(\* Levert bij aanroep de tokenwaarde op (in de variabele sym) van het eerstvolgende symbool in de invoer \*)**

**PROCEDURE error (sy: tsymbol; str: string);**

**(\* Genereert een foutmelding in de vorm:  
"representatie van sy"+"waarde van str" \*)**

Verder is gegeven een abstract data type (ADT) stack:

**PROCEDURE initstack;**

**(\* Creeert een lege stack \*)**

**FUNCTION isempty: boolean;**

**(\* checks if stack is empty \*)**

**FUNCTION top: symbol;**

**(\* returns the symbol on top of the stack \*)**

**PROCEDURE push (s: symbol);**

**(\* pushes s onto the stack \*)**

**PROCEDURE pop;**

**(\* pops one symbol from the stack \*)**