Adres:
Postcode en Woonplaats:

Studierichting: Informatica
Jaar van eerste inschrijving: 1996

Tentamen: Vertalerbouw 1
Datum: 20-11-1998
Naam docent: J. Jongejan

Opgave 1

We voegen de volgende productieregel toe:
$\lambda \rightarrow P \perp$

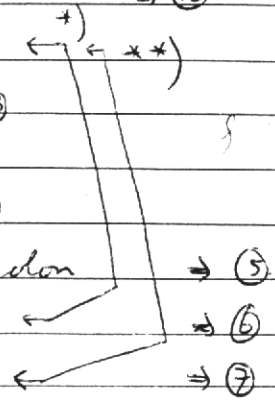
* Controle op LL(1).
Er moet gelden: $\forall (A \in N, A \rightarrow \alpha, A \rightarrow \beta \in P: \alpha \neq \beta \Rightarrow : LA(A \rightarrow \alpha) \cap LA(A \rightarrow \beta) = \phi)$

Maar er geldt:

$LA(Decls \rightarrow D) = \{int, real\}$ } niet disjunct, dus
$LA(Decls \rightarrow D\ Decls) = \{int, real\}$ } de grammatica is niet LL(

* Controle op LR(0), SLR(1) of LALR(1):

① $\lambda \rightarrow .P \perp$ ⇒ ②
$P \rightarrow .Decls\ Eqs$ ⇒ ③
$Decls \rightarrow .D$ ⇒ ④
$Decls \rightarrow .D\ Decls$ ⇒ ④
$D \rightarrow .Tp\ Lst\ semicolon$ ⇒ ⑤
$Tp \rightarrow .int$ ⇒ ⑥
$Tp \rightarrow .real$ ⇒ ⑦
② $\lambda \rightarrow P .\perp$ ⇒ -
③ $P \rightarrow Decls .Eqs$ ⇒ ⑧
$Eqs \rightarrow .Eq$ ⇒ ⑨
$Eqs \rightarrow .Eq\ Eqs$ ⇒ ⑨
$Eq \rightarrow .E\ equal\ E\ semicolon$ ⇒ ⑩
$E \rightarrow .T$ ⇒ ⑪
$E \rightarrow .T\ plus\ E$ ⇒ ⑪
$T \rightarrow .id$ ⇒ ⑫
④ $Decls \rightarrow D .$ ⇒ - (*)
$Decls \rightarrow D .Decls$ ⇒ ⑬ (**)
$Decls \rightarrow .D$ ⇒ ④
$Decls \rightarrow .D\ Decls$ ⇒ ④
$D \rightarrow .Tp\ Lst\ semicolon$ ⇒ ⑤
$Tp \rightarrow .int$ ⇒ ⑥
$Tp \rightarrow .real$ ⇒ ⑦



(5). D → Tp . Lst semicolon ⇒ (14)

Lst → . id ⇒ (15)

Lst → . id comma Lst ⇒ (15)

(6) Tp → . int ⇒ -

(7) Tp → . real. ⇒ -

(8) P → . Decls Eqs. ⇒ -

(9) Eqs → Eq. ⇒ -

Eqs → Eq . Eqs rest done? ⇒ (16)

(10) Eq → E . equal E semicolon ⇒ (17)

(11) E → T. ⇒ -

E → T. plus E ⇒ (18)

(12) T → id. ⇒ -

(13) Decls → D . Decls. ⇒ -

(14) D → Tp Lst . semicolon ⇒ (19)

(15) Lst → id. ⇒ -

Lst → id . comma Lst ⇒ (20)

(16) Eqs → Eq . Eqs. ⇒ -

(17) Eq → E equal . E semicolon ⇒ (21)

E → . T ⇒ (11)

E → . T plus E ⇒ (11)

T → . id ⇒ (12)

(18) E → T plus . E ⇒ (22)

E → . T ⇒ (11)

E → T plus E ⇒ (11)

T → . id ⇒ (12)

(19) D → Tp Lst semicolon. ⇒ -

(20) Lst → id comma . Lst ⇒ (23)

Lst → . id ⇒ (15)

Lst → . id comma Lst ⇒ (15)

(21) Eq → E equal E . semicolon ⇒ (24)

(22) E → T plus E. ⇒ -

(23) Lst → id comma Lst. ⇒ -

(24) Eq → E equal E semicolon. ⇒ -

Afdeling Wiskunde en Informatica R.U.G.

Naam: Dennis van der Laan	Geb. datum:	Bladnr.: 2
Adres:	Studierichting:	Tentamen:
Postcode en Woonplaats:	Jaar van eerste inschrijving:	Datum:
		Naam docent:

vervolg Opgave 1

Er doen zich de volgende conflicten voor:

*) shift / reduce conflict: We kunnen D reduceren tot Decls of een int-symbool weglezen.
Concrete grammatica is niet LR(0).
Er geldt: follow(Decls) = {id}, dus int ∉ follow(Decls)
Dit conflict is dus in SLR(1) oplosbaar.

**) Zelfde shift / reduce conflict als bij *), maar nu kunnen we een real weglezen. Er geldt dat ook real ∉ follow(Decls), dus ook dit conflict is SLR(1) oplosbaar.

***) shift / reduce conflict: We kunnen T reduceren tot E, of een 'plus' weglezen.
Er geldt: follow(E) = {equal, semicolon}, dus plus ∉ follow(E) → ook dit probleem is SLR(1) oplosbaar.

****) shift / reduce conflict: We kunnen id reduceren tot Lst, of een comma weglezen. Er geldt: follow(Lst) = {semicolon}, dus comma ∉ follow(Lst) dus SLR(1) oplosbaar.

Er zijn geen andere conflicten, dus de grammatica is een SLR(1) - grammatica.

En dus ...

Opgave (2).

We maken gebruik van de volgende attributen:

Tp. tp	van type type	,	synthesized
Lst. tp	van type type	,	inherited
E. tp	van type type	,	synthesized
T. tp	van type type	,	synthesized

We maken gebruik van:

procedure error (err string);
(* drukt de string 'err' af op standard error en stopt het programma *)

We krijgen dan de volgende attribuut grammatica:

P → Decls Eqs
Decls → D | D Decls
D → Tp { Lst. tp := Tp. tp } Lst Semicolon
Tp → int { Tp. tp := inttp }
Tp → real { Tp. tp := realtp }
Lst → id { storeid (id. symid, Lst tp) }
Lst_0 → id comma { Lst_1. tp := Lst_0 tp } Lst_1
{ storeid (id. symid, Lst_1. tp) }
Eqs → Eq
Eqs → Eq Eqs
Eq → E_0 equal E_1 semicolon { if E_0. tp ≠ E_1. tp then begin
error ("operandtypes do not match");
end; }
E → T { E. tp := T. tp }
E_0 → T plus E_1 { if T. tp ≠ E_1. tp then begin
error ("operandtypes do not match");
end else begin
E_0. tp := T. tp;
end; }
T → id { T. tp := gettype (id. symid) }

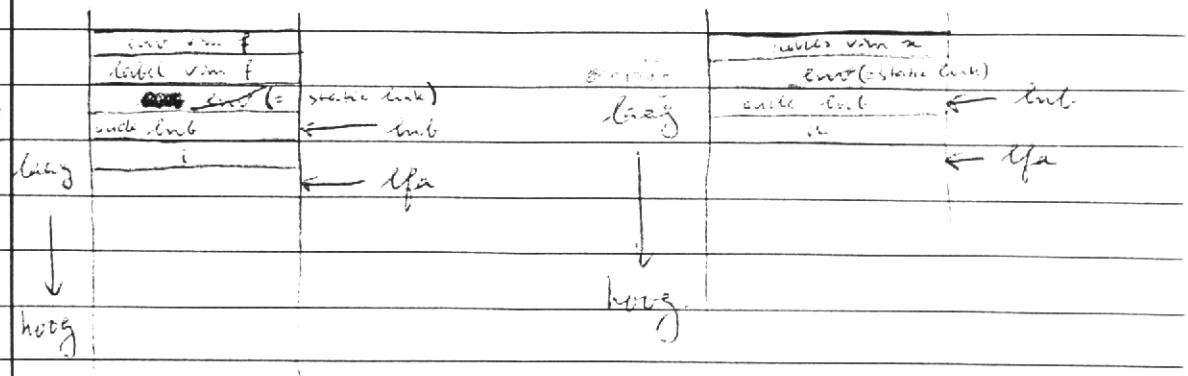
Naam: Dennis van der Haam
Adres:
Postcode en Woonplaats:

Geb. datum:
Studierichting:
Jaar van eerste inschrijving:

Bladnr.: 3
Tentamen:
Datum:
Naam docent:

Opgave 3.

a). AR van p: AR van q:



b). entry: M(lfa) := env ; sla env op
M(lfa+1) := lnk ; sla lnk + aanroepers op
lnk := lfa ; zet nieuwe lnk
lfa := lfa + 2 ; zet lfa goed

exit: lfa := lfa - 2 ; zet lfa terug
lnk := M(lnk+1) ; zet lnk terug
return ; maak aanroepers actief

c) (*1*) R0 := gf ; adres van a
R1 := M(lnk - 2) ; waarde van i
M(lfa + 1) := R1 ; zet param i van f klaar
lfa := lfa + 2 ; zet lfa verhoogd
env := M(lnk - 3) ; zet env op env van f
call M(lnk - 2) ; roep label van f
lfa := lfa - 2 ; M(lfa) bevat nu resultaat v. f
R2 := M(lfa) ; resultaat van f
M(R0) := M(R0) + R2 ; a := a + f(.)
en e hoeft dat f R0 niet gebruikt

(*2*) R0 := M(lnk - 1) ; waarde v. i
R1 := lnk - 2 ; resultaat v. f
M(R1) := R0 * R0 ; f := i * i

```
(*3*)   ~~R0 := M(gp)~~                    ; ~~waarde van a.~~
        R0 := M(lnb)                       ; lnb van omliggende block.
        R0 := M(R0+2)                      ; waarde van a.
        M(lfa+1) := R0                     ; zet a als param. voor f
        lfa := lfa+2                       ; zet lfa goed.
        env := lnb                         ; omliggende block is r.
        call label-f                       ; roep f aan
        lfa := lfa-2                       ; zet lfa terug
        R0 := M(lfa)                       ; R0 bevat resultaat van f(a)
        R1 := M(lnb-1)                     ; R1 bevat waarde van b.
        R2 := M(lnb)                       ; lnb van q.
        R2 := M(R2-1)                      ; adres van x.
        M(R2) := R1 + R0                   ; x := b + f(a).

(*4*)   M(lfa) := lnb                      ; omliggende block van f is r.
        M(lfa+1) := label-f                ; adres van f.
        lfa := lfa+2                       ; zet lfa goed
~       (env := gp)                        ; omliggende block v. p is hoofdprog
        call label-p                       ; roep p aan met f
        lfa := lfa-2                       ; zet lfa terug.

(*5*)   M(lfa) := M(M(lnb-1))              ; zet waarde v. x als param. voor
        lfa := lfa+1                       ; zet lfa goed.
        env := lnb                         ; q is omliggende block.
        call label-r                       ; roep r aan met x.
        lfa := lfa-1                       ; zet lfa terug.

(*6*)   M(lfa) := gp                       ; adres van a.
        lfa := lfa+1                       ; zet lfa goed.
~       (env := gp)                        ; omliggende block is hoofdprogr.
        call label-q                       ; roep q aan met adres van a.
        lfa := lfa-1                       ; zet lfa terug
```

Afdeling Wiskunde en Informatica R.U.G.

Naam: Dennis van der Haan	Geb. datum:	Bladnr.: 4.
Adres:	Studierichting:	Tentamen:
Postcode en Woonplaats:	Jaar van eerste inschrijving:	Datum:
		Naam docent:

Opgave 4.

Zoals bij opgave 1 is gebleken, voldoet de grammatica niet aan de LL(1) voorwaarde.
De conflicten doen zich in de volgende producties voor en de volgende oplossingen zijn hiervoor bedacht:

Decls → D
Decls → D Decls
} oplossing: Decls → D Decls2
Decls2 →
Decls2 → Decls

Lst → id
Lst → id comma Lst
} oplossing: Lst → id Lst2
Lst2 →
Lst2 → comma Lst

Eqs → Eq
Eqs → Eq Eqs
} oplossing: Eqs → Eq Eqs2
Eqs2 →
Eqs2 → Eqs

E → T
E → T plus E
} oplossing: E → T E2
E2 →
E2 → plus E.

Duidelijk is te zien dat deze grammatica nog steeds dezelfde ~~taal~~ taal voortbrengt als de ~~taal~~ grammatica uit opgave 1.
Deze nieuwe grammatica voldoet wél aan de LL(1) voorwaarde.

Nu de parser:

```
procedure match (t: tsymbol);
  begin
    if sym = t then nextsym
    else begin
      error (t, " expected");
      halt (-1);
    end;
  end;

procedure parser;
  begin initstack; initscanner; nextsym;
    push (eofs);
    push (P);
    while top ≠ eofs do begin
      if top = [semicolon, id, comma, int, real, equal, plus]
      then begin
        match (sym);
        pop;
      end else if top = P then begin
        pop;
        push (Eqs);
        push (Decls);
      end else if top = Decls then begin
        pop;
        push (Decls2),
        push (D),
      end else if top = D then begin
        pop;
        push (semicolon);
        push (Lst);
        push (Tp);
      end else if top = Tp then begin
        if sym = int then begin
          pop; push (int);
        end else begin
          pop; push (real);
        end;
      end else if top = Lst then begin
        pop;
        push (Lst2);
```

vervolg Opgave 4.

```
;
end else if top = Decls2 then begin
  if sym in [int, real] then begin
    pop;
    push (Decls);
  end else begin
    pop;
  end;
end else if top = Lst2 then begin
  if sym = comma then begin
    pop; push (Lst);
  end else begin
    pop;
  end;
end else if top = Eqs then begin
  pop;
  push (Eqs2);
  push(Eq);
end else if top = Eqs2 then begin
  if sym = id then begin
    pop; push (Eqs);
  end else begin
    pop;
  end;
end else if top = Eq then begin
  pop;
  push (semicolon); push(E); push(equal); push(E);
end else if top = E then begin
  pop;
  push (E2); push(T);
end else if top = T then begin
  pop;
  push (id);
end else if top = E2 then begin
  if sym = id then begin
    pop; push (E);
  end else begin pop; end;
```

```
end;
end; {while}
  if sym ≠ eofs then begin
     error (eofs, " expected");
     halt (-1);
  end; (* parser *)

begin (* main *)
  initscanner;
  initStack;
  nextsym;    (* lees eerste symbool van invoer *)
  parser ;
end.  (* main *)
```