MATHEMATICAL CENTRE SYLLABUS 21, 1975, X, 151-165.

RECURSIEVE PROCEDURES EN EENVOUDIGE INDUCTIE ASSERTIES

M.M. FOKKINGA
Technische Hogeschool, Delft

1. INLEIDING

Al meermalen in dit colloquium is het woord "assertie" gevallen. Een *assertie* is een uitspraak op een plaats in een programmatekst, die in die kontekst geïnterpreteerd moet worden. Inderdaad, goed gekozen asserties vormen een machtig hulpmiddel voor de dokumentatie van een programma en voor de *overtuiging* voor de correctheid ervan. Maar meer nog, door middel van asserties kan men op grond van de programmatekst de -partiële- correctheid van een programma formeel *bewijzen*.
Het tweetal asserties dat een programmatekst omsluit noemen we een correctheidsbewering voor dat programma. Onder partiële correctheid wordt dan verstaan dat de assertie aan het einde van de tekst geldig is ná terminatie van het programma, mits vóóraf de assertie aan het begin van de tekst geldig was. Totale correctheid garandeert bovendien dat het programma inderdaad termineert.
In een formeel bewijs gaan we er van uit dat we weten welke correctheidsbeweringen waar zijn -en dus bewezen geacht mogen worden- voor de elementaire opdrachten en welke correctheidsbeweringen we mogen doen voor een samengestelde tekst op grond van alreeds bewezen correctheidsbeweringen voor de tekstgedeeltes. Deze uitgangspunten zijn geformuleerd als respectievelijk de axioma's en de samenstellings- of afleidingsregels voor het systeem waarin we werken. Als er géén andere correctheidsbeweringen waar zijn dan degene die in dit systeem bewezen kunnen worden, dan noemen we het systeem *volledig* en kunnen we zeggen dat de afleidingsregels en axioma's de semantiek van de programmeertaal volledig vastleggen.
Zo is voor de taal PASCAL door HOARE [7] een systeem van axioma's en regels gegeven dat *per definitie* van bijna alle taalconstructies (n.l. uitgezonderd de goto en de arithmetiek) de semantiek vastlegt.

Enige voorbeelden mogen dit verduidelijken. Laten we in het vervolg met $p, p_1, p_2, \ldots, q, q_1, q_2, \ldots$ eigenschappen (predicaten) aanduiden.

Voorbeeld 1

Een zinvol axioma voor de meervoudige toekenningsopdracht is

ASS: $\{p(\bar{E}(\bar{x}))\}\ \bar{x} := \bar{E}(\bar{x})\ \{p(\bar{x})\}$

en als samenstellingsregels zijn heel zinvol

WH: $$\frac{\{B \wedge p\}\ S\ \{p\}}{\{p\}\ \underline{\text{while}}\ B\ \underline{\text{do}}\ S\ \{p \wedge \neg B\}}$$

SEQ: $$\frac{\{p_1\}\ S_1\ \{q\} \qquad \{q\}\ S_2\ \{p_2\}}{\{p_1\}\ S_1; S_2\ \{p_2\}}$$

CONS: $$\frac{\{p\} \text{ impliceert } \{p_1\} \qquad \{p_1\}\ S\ \{q_1\} \qquad \{q_1\} \text{ impliceert } \{q\}}{\{p\}\ S\ \{q\}}$$

(Deze axioma's en regels zijn intuïtief gerechtvaardigd -ingeval er in $\bar{E}(\bar{x})$ geen neveneffecten optreden- : we zijn er zeker van dat we hiermee geen beweringen kunnen afleiden die tegenstrijdig zijn met een "werkelijke" uitvoering van een programma.)
Voor de berekening van de grootste gemene deler*) van A en B volgens de algoritme van Euclides kunnen we gebruik maken van het programma:

```
{A>B≥0}
{A>B≥0 ∧ [A,B]=[A,B]}
(m,n):= (A,B);
{m>n≥0 ∧ [m,n]=[A,B]}
while n≠0
do {m>n≥0 ∧ [m,n]=[A,B] ∧ n≠0}
   {n>m-n · m+n≥0 ∧ [n, m-n · m÷n]=[A,B]}
```

*) [A,B] staat voor: de grootste gemene deler van A en B.

```
        (m,n):= (n, m-n · (m÷n));
        {m>n≥0 ∧ [m,n]=[A,B]}
    od;
    {m>n=0 ∧ [m,n]=[A,B]}
    {m=[A,B]}
    GGD:= m
    {GGD=[A,B]}
```

waarvan nu tevens de -partiële- correctheid bewezen is m.b.v. de methode van de asserties, omdat voor ieder -welgevormd- tekstgedeelte de omsluitende asserties een bewezen bewering vormen volgens de gegeven axioma's en regels.*)

"Formeler" opgeschreven luidt het bewijs als volgt:

(i)	{ Li }	Si	{ Ri }	Vi
(1)	{A>B≥0 }	impliceert	{A>B≥0 ∧ [A,B]=[A,B]}	lemma
(2)	{R1 }	(m,n):= (A,B)	{m>n≥0 ∧ [m,n]=[A,B]}	ASS
(3)	{n≠0 ∧ R2}	impliceert {n>m-n·m÷n≥0	∧ [n,m-n·m÷n]=[A,B]}	lemma
(4)	{R3 }	(m,n):= (n,m-n·(m÷n))	{m>n≥0 ∧ [m,n]=[A,B]}	ASS
(5)	{L3 }	S4	{R4 }	3,4 : C+S
(6)	{R2 }	while n≠0 do S4 od	{R4 ∧ n = 0 }	5 : WH
(7)	{R6 }	impliceert	{m = [A,B] }	lemma
(8)	{R7 }	GGD:= m	{GGD = [A,B] }	ASS
(9)	{A>B≥0 }	HET VOLL. PROGR: S2;S6;S8	{GGD = [A,B] }	1,2,6,7,8:C+S

Voorbeeld 2

Vermaard is al het axioma voor de toekenningsopdracht,
ASS: $\{p(E(x))\}\ x:= E(x)\ \{p(x)\}$. Maar intuïtief gerechtvaardigd is ook het volgende axioma ASS': $\{p(x)\}\ x:= E(x)\ \{\exists x_0 : p(x_0) \wedge x = E(x_0)\}$. Mits E geen neveneffecten heeft zal ASS' geen tegenstrijdigheden opleveren met de "werkelijkheid", en zelfs is ze naar ons gevoel ook volledig, d.w.z. iedere ware bewering over de toekenningsopdracht x:= E(x) kan in de vorm van ASS' geschreven worden. Dus voortaan kunnen we met recht ASS' als definitie van de semantiek nemen.

---

*) Het is ook mogelijk de correctheidsbewering {A,B>0} ... {GGD=[A,B]} te bewijzen.

Maar ASS is ook volledig (en gelijkwaardig met ASS'; ga na welke p je moet kiezen in ASS' om ASS uit ASS' af te leiden, en omgekeerd) en wordt meestal gebruikt in praktische toepassingen.

Voorbeeld 3

Voor recursieve procedures heeft Hoare de volgende regel opgesteld:

(ALS:) {p} Body-van-P {q} is af te leiden volgens het systeem van axioma's en regels, met de *hypothese* {p} call P {q} voor ieder voorkomen van call P in de Body-van-P

(DAN:) {p} call P {q}

Deze bewijsregel is gerechtvaardigd door bijvoorbeeld uit de premisse een bewijs van {p} Body-van-P {q} af te leiden met inductie naar de recursiediepte (rd). Want bij rd=1 leidt een evaluatie van de body niet tot een aanroep op P en hebben we derhalve de hypothese niet nodig en staat in de premisse al een bewijs voor dit geval. En bij grotere rd leiden de binnenaanroepen call P tot een recursiediepte < rd, dus mogen we daarvoor de inductiehypothese aannemen. Als daarmee {p} Body-van-P {q} bewezen kan worden, zoals in de premisse staat, dan kunnen we volgens het principe van volledige inductie stellen

{p} Body-van-P {q}

voor evaluaties die tot willekeurige recursiediepte leiden, dus {p} call P {q}.
Maar deze afleidingsregel kan bijv. ook afgeleid worden uit een andere bewijsregel, Scott's inductieregel, die sterker is dan de boven gegeven regel van HOARE. Scott's inductieregel is reeds in hoofdstuk 8 van dit colloquium uitvoerig behandeld DE BAKKER [1]; een andere naam ervoor is *computational induction*. Zie bijv. ook MANNA, NESS & VUILLEMAN [8], DE BAKKER & DE ROEVER [4], DE BAKKER & MEERTENS [3].

2. DOELSTELLING EN ENIG FORMALISME

We zullen ons nu bezig houden met theoretische aspecten van de methode van inductieve asserties. Wat we hopen te bereiken is inzicht in de kracht en de werking van de "eenvoudige" assertiemethode, met name voor recursieve

procedures. Dat is: we leggen ons de beperking op om alléén correctheidsbeweringen voor *elementaire* opdrachten in de premisse van een regel toe te laten. Dus Hoare's regel voor recursieve procedures en Scott's inductieregel worden niet gebruikt. We streven niet na een zo sterk mogelijk bewijs te geven -daarvoor verwijs ik naar DE BAKKER & MEERTENS [3]-, maar juist enig inzicht te verschaffen.

We zullen zien dat er voor iedere recursieve procedure (stel van recursieve procedures) een regel te formuleren is, die volledig is, maar waarvan het aantal beweringen in de premisse dan en slechts dan eindig is als de recursieve procedure "regulier" is. Bovendien is zo'n regel een karakterisering voor de recursieve procedure P in de volgende zin:

> als voor enig ander programma T de correctheidsbewering van de regel ook voor T geconcludeerd kan worden uit de premisse, voor alle specificaties van de asserties
> dan en slechts dan
> is P een uitbreiding van T wat het invoer-uitvoer-gedrag betreft.

Allereerst hebben we enig formalisme nodig om onze beweringen ondubbelzinnig te kunnen formuleren. Maar voor de leesbaarheid zal ik het zoveel mogelijk beperken; de lezer vergeve mij de mogelijke onvolkomenheden.
De *programma's* die we beschouwen zijn stelsels van recursieve procedures gegeven door declaraties van de volgende vormen:

vb. $\{P \Leftarrow A_1;A_2;P;A_3 \cup A_4$

of

vb. $\begin{cases} P_1 \Leftarrow A_1;P_2;A_1;P_2 \cup A_1;P_2 \\ P_2 \Leftarrow ((A_4;P_2) \cup A_3);\ A_2 \end{cases}$

waarbij we veronderstellen dat het in-uitvoergedrag van de hoofdletters $A_{1,2\ldots}$ bekend is. Zoals gewoonlijk wordt het in-uitvoergedrag van een samenstelling van opdrachten uit dat van de onderdelen verkregen d.m.v. een aaneenschakeling van de gedragingen in geval van een ;-samenstelling en d.m.v. een niet-deterministische keuze tussen de gedragingen van de onderdelen in geval van een $\cup$-samenstelling. Voor proceduresymbolen moeten we de copy-rule toepassen, d.w.z. het in-uitvoergedrag van een procedure wordt volkomen gegeven door dat van zijn body.

Opmerking 1. Het niet-deterministisch gedrag bepaald door $\cup$ is gemakkelijk

deterministisch te maken door vóór ieder daarmee samengesteld onderdeel S_i een "opdracht" B_i te plaatsen met zó een gedrag dat voor iedere invoer hoogstens één B_j die invoer doorlaat en alle andere B_k géén uitvoer geven.

Voorbeeld

Laten de gedragingen van B_1 en B_2 elkaar "uitsluiten", dan is het gedrag van $S_1 \cup S_2$ mogelijk niet-deterministisch, maar dan komt het gedrag van $B_1;S_1 \cup B_2;S_2$ overeen met het deterministisch gedrag van

if B_1 **then** S_1 **else** S_2 (of beter: **if** B_1 **then** S_1 **else if** B_2 **then** S_2 **d**↑)

Opmerking 2. We zullen ook voor de elementaire opdrachten niet-deterministische gedragingen toelaten, d.w.z. voor een invoer kunnen er verscheidene uitvoeren gespecificeerd zijn. Hiermee zijn de programma's in het algemeen dus in-uitvoerrelaties geworden i.p.v. in-uitvoerfuncties. We zullen dan ook met $x(S)y$ aanduiden dat x en y in de in-uitvoerrelatie zitten bepaald door S.

We kunnen voor opdrachten S_1 en S_2 beweren dat de relatie bepaald door S_1 geheel omvat wordt door die van S_2, i.e. $\forall x,y: x(S_1)y \rightarrow x(S_2)y$. We formuleren dit als "$S_1 \subseteq S_2$" en het inclusieteken heeft dan zijn gewone betekenis voor de relaties, i.e. verzamelingen, (S_1) en (S_2).
Nu het formalisme voor correctheidsbeweringen. Als in Hoare's notatie $\{p\}\ S\ \{q\}$ gesteld wordt, dan betekent dit

$$\forall x,y: p(x) \wedge x(S)y \rightarrow q(y),$$

ofwel, als we de predicaten ook noteren in relatievorm,

$$\forall x,y: x(p)x \wedge x(S)y \rightarrow x(S)y \wedge y(q)y,$$

zodat volgens onze afspraken voor de ;-samenstelling

$$\forall x,y: x(p;S)y \rightarrow x(S;q)y \ , \quad \text{i.e.}$$
$$p;S \subseteq S;q$$

waarbij uit de kontekst duidelijk moet zijn wat de interpretatie voor de asserties p en q is.
Wanneer A en C inclusies zijn van de vorm $S_1 \subseteq S_2$ of $p;S_1 \subseteq S_1;q$ e.d. en de interpretatie voor de asserties (kleine letters p, q,...) wordt gesymboliseerd met de aanduiding I, dan zetten we $A \models_I C$ voor "A impliceert C" en we laten de aanduiding I weg als "voor alle interpretaties voor de

asserties, A de bewering C impliceert".

Voorbeeld

Hoare's regel WH luidt nu:

$$p;B;A \subseteq A;p \models p;(\underline{\text{while}}\ B\ \underline{\text{do}}\ A) \subseteq (\underline{\text{while}}\ B\ \underline{\text{do}}\ A);p;\overline{B}$$

Definiëren we de while opdracht als de recursieve procedure W

$$W \Leftarrow B;A;W \cup \overline{B}$$

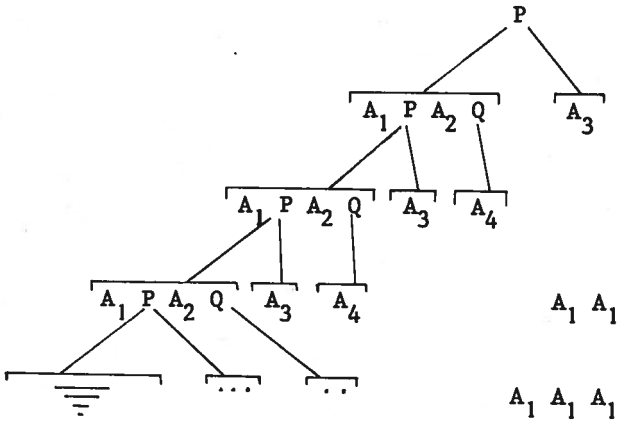
dan kunnen we stellen $p;B;A \subseteq A;p \models p;W \subseteq W;p;\overline{B}$.

3. FORMULERING VAN DE REGEL

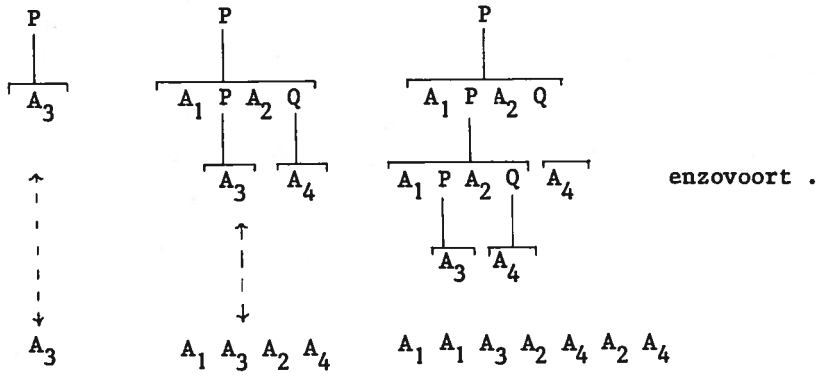
Beschouw eens het programma

$$P \Leftarrow A_1;P;A_2;Q \cup A_3$$
$$Q \Leftarrow A_4$$

Dan kunnen we een "boom" opzetten die alle mogelijke evaluaties (van P) bevat, n.l.



Dit soort bomen komt veelvuldig voor in formele talentheorie; meestal wordt slechts dat gedeelte gegeven dat precies één "woord" laat afleiden, bijv.



Ieder van de laatstgeschetste zullen we een *afleidingsboom* noemen en de eerstgeschetste de *volledige afleidingsboom*. Uit de definitie van de copy-rule als interpretatie voor procedureaanroepen en uit uw gevoel voor wat de "woorden" van een volledige afleidingsboom zijn, volgt dat voor iedere in- en uitvoer x(P)y er een woord w is in de volledige afleidingsboom voor P, zodat x(w)y, en omgekeerd. D.w.z. de relatie (P) is omvat door de vereniging van de relaties bepaald door de woorden uit de boom, en omgekeerd. Noteren we de verzameling van woorden uit een boom B als $L(B)$ en de daardoor bepaalde in-uitvoerrelatie als $x(L(B))y$ voor desbetreffende x en y dan hebben we

Stelling 1. $P \subseteq L(B_P)$, $L(B_P) \subseteq P$, ofwel $P = L(B_P)$.

Terminologie. Beschouw in de voorgaande figuren nog eens de (volledige) afleidingsboom en het woord $A_1\ A_3\ A_2\ A_4$. We zeggen dat in dat woord A_3 *direkt volgt* op A_1, A_2 direkt volgt op A_3, enz. Definieer nu voor de overeenkomstige voorkomens van die symbolen in de boom dezelfde "direkt na" relatie. We zeggen ook dat A_1 een *linkersymbool* is in dat woord, en A_4 een *rechtersymbool*. Definieer nu ook voor de overeenkomstige voorkomens van die symbolen in de boom zo een "is linker voorkomen" en "is rechter voorkomen" eigenschap. Dan kunnen we definiëren dat de woorden van een boom precies de rijen van symbolen zijn die in de boom achtereenvolgens direkt na elkaar voorkomen, beginnend met een linkervoorkomen en eindigend met een rechter-

voorkomen. We zullen dit in het vervolg als definitie aannemen.

We geven nu de definitie van een stelsel correctheidsbeweringen (voor elementaire opdrachten) met betrekking tot correctheidsassertie-symbolen p_{in}, p_{ex} en programma P. Als we dit stelsel, noem het A, nemen als premisse van de regel met conclusie $p_{in};P \subseteq P;p_{ex}$, dan is deze regel de gezochte volledige, karakteriserende regel. Het bewijs daarvan volgt in het volgende hoofdstuk. Nu eerst de

Definitie

Laat $\{p_i\}_{i\in\mathbb{N}}$ een verzameling nieuwe, i.e. nog nergens voorkomende predicaatletters zijn. Associeer met ieder voorkomen van een opdrachtsymbool in de boom B-(voor-P) precies één zo'n predicaatletter, dan bestaat A uit

1) voor direkt opvolgende symbolen A en A' met geassocieerde p en p' de bewering $p;A \subseteq A;p'$
2) voor ieder linker/rechter voorkomen van symbolen A/A' met geassocieerde p/p' de bewering $p_{in} \subseteq p/p';A' \subseteq A';p_{ex}$.

4. STELLINGEN EN BEWIJZEN

De wezenlijke stelling met bewijs ligt besloten in het

Hoofdlemma

(i) $A \models p_{in};L(B) \subseteq L(B);p_{ex}$.

(ii) $p_{in};L(B) \subseteq L(B);p_{ex} \models_{*} A$
(*: mits de asserties $\{p_i\}_{i\in\mathbb{N}}$ in A geschikt geïnterpreteerd worden)

(iii) voor willekeurig programma T
$A \models p_{in};T \subseteq T;p_{ex}$ impliceert $T \subseteq L(B)$.

Bewijs

(i) Voor willekeurige interpretatie I voor de asserties redeneren we als volgt. Zij $A_1;A_2;\ldots;A_n$ een woord uit $L(B)$, dan hebben we per definitie $p_{in} \subseteq p_1$, $p_1;A_1 \subseteq A_1;p_2,\ldots,p_n;A_n \subseteq A_n;p_{ex}$ in A en hiermee kunnen we in n stappen concluderen dat hun geldigheid die van $p_{in};A_1;\ldots;A_n \subseteq A_1;\ldots;A_n;p_{ex}$ impliceert, i.e. $A \models_I p_{in};A_1;\ldots;A_n \subseteq A_1;\ldots;A_n;p_{ex}$. Dit geldt voor alle woorden uit $L(B)$, dus (met enkele stappen) $A \models_I p_{in};L(B) \subseteq L(B);p_{ex}$. □

(ii) Merk op dat de restrictie in de keuze voor de interpretatie van de p_i in A ook zó gelezen kan worden: als $p_{in};L(B) \subseteq L(B);p_{ex}$ geldt dan kan er

een interpretatie voor de asserties p_i in Á gevonden worden zó dat Á ook geldig is.
Deze geschikte interpretatie is bijvoorbeeld de volgende (in DE BAKKER & MEERTENS [3] staat een karakterisering wélke het allemáál kunnen zijn):

$x(p)x \leftrightarrow$ "x kan als resultaat verkregen worden uitgaande van een invoer die aan p_{in} voldoet, door een berekening door $A_1,\ldots,A_{i-1}$ waarbij $A_1;\ldots;A_i$ een beginstuk van een woord uit de boom B is en p met A_i geassocieerd is", i.e. $\exists x_0$: $x_0(p_{in};A_1;\ldots;A_{i-1})x$ en p is met A_i geassocieerd.

Inderdaad, nu is de geldigheid van de $p_{in} \subseteq p$ en $p;A \subseteq A;p'$ in Á gemakkelijk aan te tonen, en voor de $p;A \subseteq A;p_{ex}$ redeneren we als volgt.
Voor alle x en y: als $x(p;A)y$, i.e. $x(p)x \wedge x(A)y$, dan per definitie $x_0(p_{in};A_1;\ldots;A_{n-1})x \wedge x(A)y$ waarbij $A_n \equiv A$ en $A_1;\ldots;A_n$ een woord uit B is, dus $x_0(p_{in};L(B))y$ en derhalve volgens de premisse $x_0(L(B);p_{ex})y$, dus i.h.b. $y(p_{ex})y$ en aangezien ook nog geldt $x(A)y$ volgt $x(A;p_{ex})y$. □

(iii) We tonen $T \subseteq L(B)$ aan. Dus zij $x_0(T)y_0$ dan moeten we $x_0(L(B))y_0$ aantonen.
Beschouw daartoe het antecedent met de volgende interpretatie voor de asserties p_i in Á en p_{in},p_{ex} (een interpretatie die van x_0 afhangt):

$x(p)x \leftrightarrow x_0(A_1;\ldots;A_{i-1})x$ waarbij p met A_i geassocieerd is en $A_1\ldots A_i$ een beginstuk van een woord uit de boom B is,
$x(p_{in})x \leftrightarrow x=x_0$,
$x(p_{ex})x \leftrightarrow x_0(A_1;\ldots;A_n)x$ voor een woord $A_1\ldots A_n$ uit B.

Nu is het gemakkelijk de geldigheid van de beweringen in Á aan te tonen:

- als $p;A \subseteq A;p'$ in Á is, dan voor alle x en y:
  als $x(p;A)y$ dan $x(p)x \wedge x(A)y$, dus $x_0(A_1;\ldots;A_{i-1})x \wedge xAy$ met $A_i \equiv A$, dus $x_0(A_1;\ldots;A_i)y \wedge x(A)y$, dus $x(A)y \wedge y(p')y$, i.e. $x(A;p')y$.
- als $p_{in} \subseteq p$ dan triviaal.
- als $p;A \subseteq A;p_{ex}$ in Á is, dan net als hierboven.

Welnu, volgens het antecedent volgt uit Á de geldigheid van $p_{in};T \subseteq T;p_{ex}$. Omdat verondersteld was $x_0(T)y_0$ hebben we zelfs $x_0(p_{in})x_0 \wedge x_0(T)y_0$ dus $x_0(T;p_{ex})y_0$ en i.h.b. $y_0(p_{ex})y_0$ hetgeen volgens de gekozen interpretatie voor de asserties betekent $x_0(A_1\ldots A_n)y_0$ voor een of ander woord uit B, dus $x_0(L(B))y_0$. □

Nu volgt onmiddellijk de volgende

Stelling 2

(i) Zowel (a) $A \models p_{in};P \subseteq P;p_{ex}$

Als ook (b) $p_{in};P \subseteq P;p_{ex} \models_{*} A$

(*: mits de asserties in A geschikt gekozen worden).

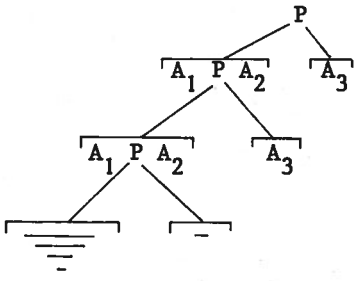
(ii) Voor willekeurig programma T

$A \models p_{in};T \subseteq T;p_{ex}$ slals*) $T \subseteq P$.

Bewijs

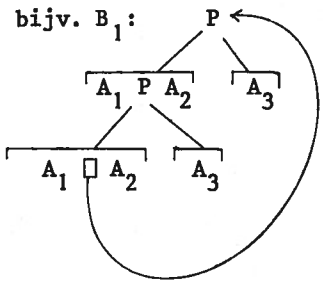
Stelling 1, $P = L(B)$, toegepast op Hoofdlemma (i), (ii) geeft (i(a)(b)). En (ii,⇒) volgt zo ook uit Hoofdlemma (iii), terwijl (ii⇐) volgt uit Hoofdlemma (i) tesamen met hetvolgende feit: als een in-uitvoerbewering geldt voor relatie (P), dan geldt die zeker voor iedere deelrelatie (T), i.e. $T \subseteq P$, $p_{in};P \subseteq P;p_{ex} \models p_{in};T \subseteq T;p_{ex}$. □

We hebben hiermee ons doel bereikt om een regel te formuleren die volledig en karakteriserend is. Dit is eigenlijk niet verwonderlijk, we hebben immers inductieve asserties geplaatst in iedere mogelijke evaluatie van de procedure. Maar het is wel onbevredigend dat het aantal beweringen in de premisse oneindig is, zo gauw het stel procedures recursief is. We kunnen proberen een vereenvoudiging in het stel beweringen aan te brengen. Dat komt er op neer in de volledige afleidingsboom voor P een vereenvoudiging aan te brengen door bijv. daartoe geschikte symbolen te schrappen, mét alles wat eruit voortsproot, en in plaats daarvan een verwijzing te geven naar een voorkomen van dat symbool elders in de boom. Noemen we deze handeling "opknopen", dan is voor de procedure $P \Leftarrow A_1;P;A_2 \cup A_3$ de volledige afleidingsboom B:

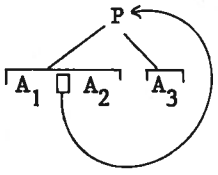


*) Afkorting van "als" gelezen van links naar rechts en "als" gelezen van rechts naar links.

en een opgeknoopte volledige afleidingsboom is bijv. B_1:



en ook B_2:



We definiëren de relatie "volgt direkt na" en de eigenschappen "is linker/rechter voorkomen" zó dat ze behouden blijven voor de voorkomens van symbolen die onder het opknopen onaangeroerd blijven. Omdat we de weggeschrapte symbolen met elders voorkomende identificeren, definiëren we bovendien dat voorkomens van symbolen "direkt na" elkaar volgen, als hun identificaties in de oorspronkelijke boom "direkt na" elkaar volgen. (Deze en ook de andere definities kunnen gemakkelijk geformaliseerd worden, zie FOKKINGA [5],[6].)

Voorbeeld

Ten gevolge van het opknopen is in B_1 de bovenste A_1 nu óók "direkt na" de onderste A_1, en is de bovenste A_2 nu óók "direkt gevolgd door" de onderste A_2. De bovenste A_3 ligt nu óók "direkt tussen" de onderste A_1 en A_2.

Herinner u nu de definitie dat de woorden van een boom precies de rijen van symbolen zijn die achtereenvolgens direkt na elkaar voorkomen, beginnend met een linker- en eindigend met een rechtersymbool.

Voorbeeld. $L(B_1) = \{A_1^n;A_3;A_2^m \mid m=n(\mathrm{mod}2)\}$, $L(B_2) = \{A_1^n;A_3;A_2^m\}$.

Met de definitie van Á onveranderd (maar wel opgeknoopte volledige afleidingsbomen toelatend) blijft het Hoofdlemma en het bewijs ervan correct ! Als nu ook nog stelling 1, $P = L(B)$, blijft gelden na opknoping van bomen, dan zou ook "de stelling", stelling 2, nog gelden. Maar in het algemeen is de verzameling van woorden echt uitgebreid, en daarmee hun totale in-uitvoergedrag.
Laat B nu eens een volkomen willekeurige opgeknoopte volledige afleidingsboom zijn, niet gerelateerd aan P, en Á het stel inductieve beweringen, gebaseerd op boom B. Mogelijkerwijs is Á eindig. Het Hoofdlemma blijft,

zoals gezegd, geldig. Om toch de stelling te concluderen is een noodzakelijk en voldoende voorwaarde dat zowel $P \subseteq L(B)$ alsook $P \supseteq L(B)$ geldt.

Willen we dat "$A \therefore p_{in};P \subseteq P;p_{ex}$" een regel is *met algemene geldigheid*, dus *voor iedere specificatie van de elementaire opdrachten in* P en A, dan zijn de condities $P \subseteq L(B)$ en $P \supseteq L(B)$ gelijkwaardig met $L(B\text{-voor-}P) \subseteq L(B)$ en $L(B\text{-voor-}P) \supseteq L(B)$, i.e. $L(B\text{-voor-}P) \equiv L(B)$, waarbij we nu syntactische inclusie en gelijkheid bedoelen en B-voor-P de volledige afleidingsboom voor P is.
Onder deze voorwaarde, $L(B\text{-voor-}P) \equiv L(B)$, geldt nu

Stelling 3

A kan precies dan eindig gekozen worden, als $L(B\text{-voor-}P)$ regulier is (en dan noemen we P *regulier*).

Bewijs

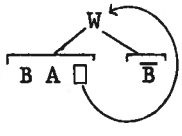
A is eindig precies dan als B eindig is (in het aantal voorkomens van elementaire opdrachtsymbolen). Als B eindig is, kunnen we B als eindige automaat opvatten die precies $L(B)$ accepteert en volgt er dat $L(B) \equiv L(B\text{-voor-}P)$ regulier is.
Omgekeerd, als $L(B\text{-voor-}P)$ regulier is, dan kunnen we een eindige opgeknoopte volledige afleidingsboom B construeren aan de hand van een grammatica voor $L(B\text{-voor-}P)$ in reguliere vorm, zó dat $L(B) \equiv L(B\text{-voor-}P)$.
De A bij deze boom is dan eindig.

5. EEN VOORBEELD

We besluiten met een uitwerking voor de while opdracht. Die opdracht kan gedefinieerd worden als een recursieve procedure

$$W \Leftarrow B;A;W \cup \overline{B} .$$

Beschouwd als een contextvrije grammatica is hij reeds in reguliere vorm.
Een eindige boom voor W is:
En volgens de definitie is het hierop gebaseerde stel inductiebeweringen:



A: $p_{in} \subseteq p_B \quad p_B;B \subseteq B;p_A \quad p_A;A \subseteq A;p_B \quad p_A;A \subseteq A;p_{\overline{B}}$

$p_{in} \subseteq p_{\bar{B}} \quad p_{\bar{B}};\bar{B} \subseteq \bar{B};p_{ex}$

Hiermee gelijkwaardig is

A': $p_{in} \subseteq p_B \quad p_B;B;A \subseteq A;p_{\overline{B}} \quad p_B;B;A \subseteq A;p_{\overline{B}}$

$p_{in} \subseteq p_{\overline{B}} \quad p_{\overline{B}};\overline{B} \subseteq p_{ex}$

en zelfs

A'': $p_{in} \subseteq p \quad p;B;A \subseteq A;p$

$p;\overline{B} \subseteq p_{ex}$

En daarmee vinden we volgens de stellingen 2 en 3:

(i) zowel $p_{in} \subseteq p,\ p;B;A \subseteq A;p,\ p;\overline{B} \subseteq p_{ex} \models p_{in};W \subseteq W;p_{ex}$

als ook $p_{in};W \subseteq W;p_{ex} \models_{*} p_{in} \subseteq p,\ p;B;A \subseteq A;p,\ p;\overline{B} \subseteq p_{ex}$

(ii) voor willekeurig programma T

$p_{in} \subseteq p,\ p;B;A \subseteq A;p,\ p;\overline{B} \subseteq p_{ex} \models p_{in};T \subseteq T;p_{ex}$ slals $T \subseteq W$.

en derhalve kunnen we als regel -die volledig en karakteriserend is- formuleren:

$$\frac{p_{in} \subseteq p,\ p;B;A \subseteq A;p,\ p;\overline{B} \subseteq p_{ex}}{p_{in};W \subseteq W;p_{ex}} \quad \text{voor } W \Leftarrow B;A;W \cup \overline{B}$$

hetgeen precies Hoare's regel is waarin CONS reeds is opgenomen.

LITERATUUR

[1] DE BAKKER, J.W., *Syllabus van het colloquium programmacorrectheid*, hoofdstuk 8, Mathematisch Centrum Amsterdam, 1974.

[2] DE BAKKER, J.W. & MEERTENS, L.G.L.T., *Simple recursive program schemes and inductive assertions*, Mathematical Centre Report MR 142, Amsterdam, 1972.

[3] DE BAKKER, J.W. & MEERTENS, L.G.L.T., *On the completeness of the inductive assertion method*, Prepublication, Mathematical Centre Report IW 12/73, Amsterdam, 1973.

[4] DE BAKKER, J.W. & DE ROEVER, W.P., *A calculus for recursive program schemes*, in Proc. IRIA Symposium on Automata, Formal Languages and Programming, M. Nivat (ed), North-Holland, Amsterdam, 1972.

[5] FOKKINGA, M.M., *Inductive assertion patterns and recursive procedures*, Report T.H. Delft, 1973.

[6] FOKKINGA, M.M., *Inductive assertion patterns and recursive procedures*, in Proc. Symp. on Programming, april 1974, to appear in: Lecture notes in comp. science, Springer-Verlag, Berlin.

[7] HOARE, C.A.R. & WIRTH, N., *An axiomatic definition of the programming language PASCAL*. Report E.T.H. Zürich.

[8] HOARE, Z, NESS, S. & VUILLEMIN, J., *Inductive methods for proving properties of programs*, in Proc. ACM Conf. on Prov. Ass. about Progr., January 1972.