Een paar praktische stellinkjes

0 Standaardeindiging

stelling (standaardeindiging 0) : Het volgende programmafragment eindigt, heeft als repetitie-invariant $0 \leq n \wedge n \leq N$ en als variante functie $N - n$:

$$
\begin{array}{ll}
 & \{\ 0 \leq N\ \} \\
 & n := 0 \\
; & \mathsf{do}\ n \neq N\ \rightarrow\ n := n+1\ \mathsf{od} \\
 & \{\ n = N\ \}
\end{array}
$$

bewijs: Opgave.
□

We breiden nu bovenstaand programmafragment uit door assignments toe te voegen aan een nieuwe variabele m, aldus:

$$
\begin{array}{ll}
 & \{\ 0 \leq N\ \} \\
 & n, m := 0, N \\
; & \mathsf{do}\ n \neq N\ \rightarrow\ n, m := n+1,\ m-1\ \mathsf{od} \\
 & \{\ n = N\ \}
\end{array}
$$

Omdat iedere verhoging van n gepaard gaat met een even grote verlaging van m is nu ook "$m+n$ is constant" een invariant van de repetitie. Die constante is, uiteraard, de som van de beginwaarden van m en n, dat is N. Kortom, ook een invariant is:

(0) $\quad m+n = N$.

Uit deze nieuwe invariant en de oude invariant $0 \leq n \wedge n \leq N$ volgt bovendien dat ook $0 \leq m \wedge m \leq N$ invariant is.

Maar nu – in de context van (0) – geldt dat $n \neq N$ gelijkwaardig is met $0 \neq m$, en dus mogen we de guard van de repetitie straffeloos vervangen door $0 \neq m$. Hier wil "straffeloos" zeggen: zonder dat het de correctheid en/of de eindiging aantast: dezelfde invarianten blijven van kracht; we hebben immers alleen maar een expressie vervangen door een gelijkwaardige expressie?

De rol van variabele n wordt nu overgenomen door m: de assignments aan n mogen we nu weglaten. Aldus verkrijgen we dan een nieuw stellinkje.

stelling (standaardeindiging 1) : Het volgende programmafragment eindigt, heeft als repetitie-invariant $0 \leq m \wedge m \leq N$ en als variante functie m:

$$
\begin{array}{l}
\quad \{\ 0 \leq N\ \} \\
\quad m := N \\
;\ \mathsf{do}\ 0 \neq m\ \rightarrow\ m := m-1\ \mathsf{od} \\
\quad \{\ n = N\ \}
\end{array}
$$

□

Natuurlijk is het mogelijk van dit tweede schemaatje de invariantie en de eindiging apart te bewijzen, los van het eerste schema. Hier hebben we dit tweede schema echter uit het eerste verkregen door middel van een (zogenaamde) *programmatransformatie*. Het tweede schemaatje erft nu de gewenste eigenschappen van het eerste, als gevolg van de transformatie, zonder dat hiervoor nieuwe bewijzen nodig zijn.

Tenslotte zij opgemerkt dat de genoemde invarianten en eindiging ook geldig blijven als aan de schemaatjes assignments worden toegevoegd aan *andere* variabelen dan n respectievelijk m. Aldus vertegenwoordigen de twee stellinkjes een hele *klasse* van programmeervoorbeelden.

1 Twee standaard φ-schema's

We behandelen hier twee standaardschema's voor twee veel voorkomende vormen van de techniek "constante door variabele vervangen". De eerste is van toepassing als de postconditie van het te maken programma kan worden geschreven als:

$$r = \varphi \cdot N \quad .$$

Door nu constante N te vervangen door een variabele n verkrijgen we als kandidaat voor de invariant:

$$r = \varphi \cdot n \quad .$$

In het volgende stellinkje is verwoord wat er bij de uitwerking hiervan allemaal komt kijken. Voor de eindiging doet men een beroep op standaardeindiging 0 uit de vorige paragraaf.

stelling (φ-schema 0) : Gegeven is een functie φ die in ieder geval is gedefinieerd op het interval $[\,0\,..\,N\,]$. Dan is het volgende program-

mafragment correct:

$$
\begin{array}{l}
\text{P0:}\quad 0 \le n \ \wedge\ n \le N \\
\text{P1:}\quad r \ = \ \varphi{\cdot}n \\
\\
\quad \{\ 0 \le N\ \} \\
\quad n, r := 0, \varphi{\cdot}0 \\
;\ \{\ \mathsf{inv}\colon \text{P0} \wedge \text{P1}\,;\ \mathsf{vf}\colon N - n\ \} \\
\quad \mathsf{do}\ n \neq N\ \ \rightarrow\ \ r := \varphi{\cdot}(n{+}1) \\
\qquad\qquad\qquad\quad ;\ \{\ (\text{P0} \wedge \text{P1})\,(n := n{+}1)\ \} \\
\qquad\qquad\qquad\quad\ \ n := n{+}1 \\
\quad \mathsf{od} \\
\quad \{\ r \ = \ \varphi{\cdot}N\ \}
\end{array}
$$

□

In het algemeen is (de definitie van) functie φ van dien aard dat $\varphi{\cdot}0$ en $\varphi{\cdot}(n{+}1)$ geen in programma's toegelaten expressies zijn. (Als dat wel zo was was er ook geen programmeerprobleem: dan zou de assignment $r := \varphi{\cdot}N$ al het probleem oplossen.) Voor de praktische toepassing van het schema dient men dan ook wel toegelaten expressies voor $\varphi{\cdot}0$ en $\varphi{\cdot}(n{+}1)$ uit te rekenen. Merk hierbij op dat uit de bewijsverplichtingen volgt dat $r := \varphi{\cdot}(n{+}1)$ dient te voldoen aan:

$$
\begin{array}{l}
\{\ \text{P0} \wedge \text{P1} \wedge n \neq N\ \} \\
r := \varphi{\cdot}(n{+}1) \\
\{\ (\text{P0} \wedge \text{P1})\,(n := n{+}1)\ \}
\end{array}
$$

Dit betekent dat bij het uitrekenen van een geschikte expressie voor $\varphi{\cdot}(n{+}1)$ gebruik mag worden gemaakt van de preconditie: $\text{P0} \wedge \text{P1} \wedge n \neq N$. Concreet betekent dit dat het voldoende is $\varphi{\cdot}(n{+}1)$ uit te drukken in $\varphi{\cdot}n$, die immers mag worden vervangen door r, op grond van P1. We proberen dus toegelaten expressies E en F te berekenen die voldoen aan:

$$
\begin{array}{lcl}
\varphi{\cdot}0 & = & E \\
\varphi{\cdot}(n{+}1) & = & \varphi{\cdot}n \oplus F \quad \text{, voor alle } n\colon 0 \le n < N \quad .
\end{array}
$$

Hierin is $\oplus$ een of andere (toegelaten) binaire operator die de waarden van $\varphi{\cdot}n$ en F combineert. Deze betrekkingen vormen in feite een *recursieve* definitie voor φ en worden ook wel *recurrente betrekkingen* genoemd. Zulke betrekkingen vormen de kern van de oplossing van het programmeerprobleem.

Als we eenmaal zulke recurrente betrekkingen hebben verkregen, mogen de assignments aan r als volgt worden gecodeerd in het programma: $r := \varphi\cdot 0$ wordt $r := E$ en, gebruikmakend van P1, wordt $r := \varphi\cdot(n+1)$ nu: $r := r \oplus F$.

* * *

Het vorige schema is gebaseerd op het vervangen van de constante N in de postconditie door een variabele n. Voor het vervangen van een constante 0 in de postconditie door een variabele kan ook schema worden geconstrueerd. In dit geval wordt de postconditie geschreven als $r = \varphi\cdot 0$, en krijgen we behoefte aan toegelaten uitdrukkingen voor $\varphi\cdot N$ en $\varphi\cdot(m-1)$. (De variabele hebben we hier m gedoopt.) Eindiging volgt uit standaardeindiging 1 uit de vorige paragraaf.

stelling (φ-schema 1): Gegeven is een functie φ die in ieder geval is gedefinieerd op het interval $[0\,..\,N]$. Dan is het volgende programmafragment correct:

P0: $0 \le m \;\wedge\; m \le N$
P1: $r = \varphi\cdot m$

$\{\; 0 \le N \;\}$
$m, r := N, \varphi\cdot N$
; $\{$ inv: P0 $\wedge$ P1 ; vf: m $\}$
do $m \neq 0 \;\rightarrow\; r := \varphi\cdot(m-1)$
; $\{\; (\mathrm{P0} \wedge \mathrm{P1})\,(m := m-1) \;\}$
$m := m-1$
od
$\{\; r = \varphi\cdot 0 \;\}$

□

Voor de toepassing hiervan proberen we nu recurrente betrekkingen voor φ uit te rekenen van deze vorm:

$$\begin{array}{lcl} \varphi\cdot N & = & E \\ \varphi\cdot(m-1) & = & \varphi\cdot m \oplus F \quad \text{, voor alle } m : 0 < m \le N \quad . \end{array}$$

Als we zulke recurrente betrekkingen hebben verkregen, mogen de assignments aan r als volgt worden gecodeerd in het programma: $r := \varphi\cdot N$ wordt $r := E$ en, gebruikmakend van P1, wordt $r := \varphi\cdot(m-1)$ nu: $r := r \oplus F$.

2 Het φ/ψ-schema

Als in het "φ-schema 0", in de betrekking $\varphi\cdot(n{+}1) = \varphi\cdot n \oplus F$, de uitdrukking F te ingewikkeld (of te inefficiënt) is om direct in een programma te kunnen gebruiken, kunnen we een nieuwe functie ψ introduceren, gedefinieerd door:

$$\psi\cdot n \;=\; F \quad \text{, voor alle } n : 0 \le n \le N \quad .$$

Vervolgens proberen we dan een bruikbare recurrente betrekking voor ψ af te leiden. Als dit lukt verkrijgen we resultaten van de volgende vorm:

$$\begin{array}{lcl} \varphi\cdot 0 & = & E \\ \varphi\cdot(n{+}1) & = & \varphi\cdot n \oplus \psi\cdot n \quad \text{, voor alle } n : 0 \le n < N \quad . \\ \psi\cdot 0 & = & G \\ \psi\cdot(n{+}1) & = & \psi\cdot n \otimes H \quad \text{, voor alle } n : 0 \le n < N \quad . \end{array}$$

Deze recurrente betrekkingen kunnen dan als volgt worden toegepast in een programma.

stelling (φ/ψ-schema 0a): Gegeven zijn functies φ en ψ die in ieder geval zijn gedefinieerd op het interval $[\,0\,..\,N\,]$ en die voldoen aan bovenstaande recurrente betrekkingen. Dan is het volgende programmafragment correct:

P0: $0 \le n \;\wedge\; n \le N$
P1: $r = \varphi\cdot n$
P2: $s = \psi\cdot n$

$\{\; 0 \le N \;\}$
$n, r, s := 0, E, G$
; $\{$ inv: P0 $\wedge$ P1 $\wedge$ P2 ; vf: $N - n$ $\}$
do $n \neq N \;\rightarrow\; r := r \oplus s$
; $\{\; (\mathrm{P0} \wedge \mathrm{P1})\,(n := n{+}1) \;\wedge\; \mathrm{P2} \;\}$
$s := s \otimes H$
; $\{\; (\mathrm{P0} \wedge \mathrm{P1} \wedge \mathrm{P2})\,(n := n{+}1) \;\}$
$n := n{+}1$
od
$\{\; r = \varphi\cdot N \;\}$

□

* * *

Soms lukt het niet een, overal goedgedefinieerde, functie ψ te kiezen waarvoor $F = \psi\cdot n$, maar lukt het wel als we ψ zó kiezen dat $F = \psi\cdot(n{+}1)$. Als het vervolgens lukt een recurrente betrekking voor deze ψ af te leiden, kan het resultaat deze vorm krijgen:

$$\begin{array}{lcl} \varphi\cdot 0 & = & E \\ \varphi\cdot(n{+}1) & = & \varphi\cdot n \oplus \psi\cdot(n{+}1) \quad \text{, voor alle } n : 0 \le n < N \quad . \\ \psi\cdot 0 & = & G \\ \psi\cdot(n{+}1) & = & \psi\cdot n \otimes H \quad \text{, voor alle } n : 0 \le n < N \quad . \end{array}$$

Op basis van deze recurrente betrekkingen is dan het volgende programma een goede oplossing voor het programmeerprobleem.

stelling (φ/ψ-schema 0b): Gegeven zijn functies φ en ψ die in ieder geval zijn gedefinieerd op het interval $[\,0\,..\,N\,]$ en die voldoen aan bovenstaande recurrente betrekkingen. Dan is het volgende programmafragment correct:

P0: $0 \le n \;\wedge\; n \le N$
P1: $r = \varphi\cdot n$
P2: $s = \psi\cdot n$

$\{\; 0 \le N \;\}$
$n, r, s := 0, E, G$
; $\{$ inv: P0 $\wedge$ P1 $\wedge$ P2 ; vf: $N - n$ $\}$
do $n \neq N \;\rightarrow\; s := s \otimes H$
; $\{\; (\mathrm{P0} \wedge \mathrm{P2})\,(n := n{+}1) \;\wedge\; \mathrm{P1} \;\}$
$r := r \oplus s$
; $\{\; (\mathrm{P0} \wedge \mathrm{P1} \wedge \mathrm{P2})\,(n := n{+}1) \;\}$
$n := n{+}1$
od
$\{\; r = \varphi\cdot N \;\}$

Merk op dat in deze variant de volgorde van de assignments aan r en s anders is dan in de vorige versie.

□

Het is belangrijk voortdurend voor ogen te houden dat zowel functiewaarde $\varphi\cdot n$ als $\psi\cdot n$ goedgedefinieerd moeten zijn, voor alle voorkomende waarden

van n; in bovenstaande uitwerkingen zijn dat de waarden van n bepaald door $0 \leq n \leq N$. De keuze tussen "φ/ψ-schema 0a" en "φ/ψ-schema0b" wordt bepaald door zulke overwegingen van goedgedefinieerdheid.

opgaven

We willen een efficiënt programma maken voor de berekening van een polynoom $(\Sigma i : 0 \leq i < N : a \cdot i * X^i)$, voor gegeven $N, 0 \leq N$, integer array $a[0 \,..\, N)$ en integer constante X.

0. Schrijf een formele specificatie op voor dit probleem.

1. Construeer een programma door uit te gaan van een functie φ, gedefinieerd door: $\varphi \cdot n = (\Sigma i : 0 \leq i < n : a \cdot i * X^i)$, voor alle $n : 0 \leq n \leq N$. Het zal nodig zijn om X^n "bij te houden"; doe dit door een tweede functie ψ in te voeren, met $\psi \cdot n = X^n$, hiervoor ook recurrente betrekkingen af te leiden en vervolgens een extra invariant $s = \psi \cdot n$ in te voeren.

2. Idem, voor een functie φ, gedefinieerd door:
   $\varphi \cdot n = (\Sigma i : n \leq i < N : a \cdot i * X^i)$, voor alle $n : 0 \leq n \leq N$. Hier lukt het versterken van de invariant niet zo best: waarom niet?

3. Idem, voor een functie φ, gedefinieerd door:
   $\varphi \cdot n = (\Sigma i : n \leq i < N : a \cdot i * X^{(i-n)})$, voor alle $n : 0 \leq n \leq N$. Nu lukt het juist heel goed, zonder versterken van de invariant! (De langs deze verkregen oplossing staat bekend als Horner's schema.)

4. Het "φ/ψ-schema 0a" en het "φ/ψ-schema 0b" zijn verfijningen van het oorspronkelijke "φ-schema 0". Formuleer soortgelijke "φ/ψ-schema's", uitgaande van het "φ-schema 1".

Eindhoven, 22 december 2004

Rob R. Hoogerwoord
faculteit der Wiskunde en Informatica
Technische Universiteit Eindhoven
postbus 513
5600 MB Eindhoven