Enkele berekeningspatronen bij het ontwerpen van repetities

Tom Verhoeff*

14 February 2003

1 Inleiding

In deze notitie bespreek ik enkele berekeningspatronen[1] die in het vak *Ontwerp van Algoritmen 1* [3] aan bod komen bij het ontwerp[2] van repetities. Het gaat hierbij om de vorm van invarianten en bijbehorende berekeningen, en hun samenhang.

Beschouw het volgende programmafragment, een repetitie met gat $\mathcal{E}$.

{ inv: P }
do B
$\rightarrow$ { $P \wedge B$ }
  $v := \mathcal{E}$
; { $P(n := n + 1)$ }
  $n := n + 1$
  { P }
od
{ $P \wedge \neg B$ }

Hierin is $\mathcal{E}$ een nader te bepalen *programma*-expressie, terwijl invariant P en guard B al bepaald zijn. We staan aanvankelijk niet stil bij het vinden van P en B, initialisatie, finalisatie en terminatie (eindiging). Zie daarvoor §6 en Appendix A.

Als voorbeeld gebruiken we de volgende specificatie voor machtsverheffen.

|[**con** X, N: int;
  var v: int;
▷ { pre: $0 \leq N$ }
  Power
  { post: $v = X^N$ }
]|

*Software Constructie, Fac. Wiskunde & Informatica, TU Eindhoven, T.Verhoeff@tue.nl

[1] Deze term is ingegeven door de term ontwerppatronen [1]. Meyer stelt in [2, p. 72] dat een succesvol ontwerppatroon een direct (her)bruikbare software-component moet zijn en niet alleen een beschrijving op papier. Daarom spreken we hier van berekeningspatronen.

[2] Het gaat hier om, al rekenend, vinden van programmaonderdelen, niet om verificatie achteraf.

2 Het basispatroon

De aanpak om $\mathcal{E}$ te vinden is altijd dezelfde: Ga de correctheid van het programma bewijzen en vind al doende een geschikte $\mathcal{E}$ die het bewijs sluitend maakt.

Het eerste patroon, dat ik het **basispatroon** noem, kan altijd toegepast worden, ongeacht de vorm van de invariant P. We bepalen $\mathcal{E}$ door gewoon het *Axiom of Assignment* toe te passen op $v := \mathcal{E}$ met de betreffende pre- en postconditie. Te bewijzen is dan[3]:

$$[\ P \wedge B \quad \Rightarrow \quad P(n := n+1)(v := \mathcal{E})\ ]$$

Zonder verdere kennis te gebruiken omtrent de vorm van P is zo'n bewijs het beste[4] als volgt op te zetten.

$$\begin{array}{ll} & |[\ P \wedge B \\ & \triangleright \\ & \quad P(n := n+1)(v := \mathcal{E}) \\ = & \quad \{\ \text{def. } P, \text{subst.}\ \} \\ & \quad \ldots \\ = & \quad \{\ \bullet \text{ Kies } \mathcal{E} = \ldots\ \} \\ & \quad \ldots \\ = & \quad \{\ \ldots\ \} \\ & \quad \text{true} \\ &]| \end{array}$$

Berekeningen van de volgende vorm zijn veelal *af te raden*, omdat je niet vrijelijk in elke stap beschikt over de preconditie $P \wedge B$.

$$\begin{array}{ll} & \quad P(n := n+1)(v := \mathcal{E}) \\ = & \quad \{\ \text{def. } P, \text{subst.}\ \} \\ & \quad \ldots \\ \Leftarrow & \quad \{\ \bullet \text{ Kies } \mathcal{E} = \ldots\ \} \\ & \quad \ldots \\ \Leftarrow & \quad \{\ \ldots\ \} \\ & \quad P \wedge B \end{array}$$

En het volgende is vrijwel *nooit handig*, omdat de formules zo lang worden en het meeste ervan (m.n. $P \wedge B \Rightarrow \ldots$) vaak niet wijzigt.

$$\begin{array}{ll} & \quad P \wedge B \ \Rightarrow\ P(n := n+1)(v := \mathcal{E}) \\ = & \quad \{\ \text{def. } P, \text{subst.}\ \} \\ & \quad \ldots \\ = & \quad \{\ \bullet \text{ Kies } \mathcal{E} = \ldots\ \} \\ & \quad \ldots \\ = & \quad \{\ \ldots\ \} \\ & \quad \text{true} \end{array}$$

[3] Gemakshalve nemen we aan dat gedefinieerdheid geen rol speelt.

[4] De contextaanname $P \wedge B$ kan soms verzwakt worden, zie §7.

3 Het conjunctiepatroon

Als we wel iets van de vorm van P weten, dan kan dat mogelijk uitgebuit worden in de opzet van de berekeningen. We spreken van het **conjunctiepatroon** als P de vorm van een conjunctie heeft, zeg $P0 \wedge P1$. Passen we hierop het basispatroon toe, dan leidt dit tot de volgende berekening.

$$
\begin{array}{ll}
& |[\ P0 \wedge P1 \wedge B \\
& \triangleright \\
& \quad (P0 \wedge P1)(n := n+1)(v := \mathcal{E}) \\
= & \quad \{\ \text{subst. distribueert over } \wedge\ \} \\
& \quad P0(n := n+1)(v := \mathcal{E}) \ \wedge\ P1(n := n+1)(v := \mathcal{E}) \\
= & \quad \{\ \text{def. } P0 \text{ en } P1, \text{subst.}\ \} \\
& \quad \ldots \\
= & \quad \{\ \bullet \text{ Kies } \mathcal{E} = \ldots\ \} \\
& \quad \ldots \\
= & \quad \{\ \ldots\ \} \\
& \quad \text{true} \\
&]|
\end{array}
$$

Om de breedte van de berekening te beperken, splitsen we deze liever in twee afzonderlijke berekeningen voor $i \in \{0, 1\}$ van de vorm:

$$
\begin{array}{ll}
& |[\ P0 \wedge P1 \wedge B \\
& \triangleright \\
& \quad Pi(n := n+1)(v := \mathcal{E}) \\
= & \quad \{\ \text{def. } Pi, \text{subst.}\ \} \\
& \quad \ldots \\
= & \quad \{\ \bullet \text{ Kies } \mathcal{E} = \ldots\ \} \\
& \quad \ldots \\
= & \quad \{\ \ldots\ \} \\
& \quad \text{true} \\
&]|
\end{array}
$$

Uiteraard moeten beide keuzen van $\mathcal{E}$ compatibel zijn. Vaak komt v niet voor in zowel $P0$ als $P1$. In dat geval kan er geen dilemma voor $\mathcal{E}$ zijn, omdat $\mathcal{E}$ slechts voorkomt indien v voorkomt.

Als contextaanname hebben we hier in beide deelberekeningen de volledige invariant en de guard B opgenomen. Vaak kan per berekening volstaan worden met een zwakkere aanname (zie ook §7 en het volgende voorbeeld). Als bij de berekening t.b.v. Pi alleen $Pi \wedge B$ aangenomen hoeft te worden, dan is Pi in isolatie een invariant. Als $P0$ en $P1$ beide in isolatie invariant zijn, dan is $P0 \wedge P1$ ook een invariant. Maar omgekeerd, als $P0 \wedge P1$ een invariant is, dan hoeven noch $P0$, noch $P1$ in isolatie invariant te zijn.

Wanneer we in het vervolg over een invariant P spreken, dan bedoelen we daarmee vaak niet dat P *in isolatie* een invariant is, maar dat het een *conjunct*

uit een invariant is. Eventueel zijn er dan nog *andere* conjuncten nodig om de invariantie van P te kunnen aantonen. Soms zullen zulke andere conjuncten vooraf al bekend zijn (denk aan grenzen op variabelen), in andere gevallen worden ze pas ontdekt tijdens het uitvoeren van de berekeningen (denk aan versterken van invarianten [4, §4.3]).

Merk op dat $a \leq b \leq c$ equivalent is met $a \leq b \ \wedge \ b \leq c$ en dat ook hierop dus het conjunctiepatroon toegepast kan worden.

Bijvoorbeeld voor machtsverheffen kunnen we hebben:

$$\begin{array}{lcl} B & : & n \neq N \\ P & : & 0 \leq n \leq N \end{array}$$

Gesplitste behandeling van deze P geeft de twee berekeningen:

$$\begin{array}{l} |[\ 0 \leq n \\ \triangleright \\ \quad (0 \leq n)(n := n+1)(v := \mathcal{E}) \\ = \quad \{\ \text{subst.}\ \} \\ \quad 0 \leq n+1 \\ = \quad \{\ 0 \leq n \text{ uit context}\ \} \\ \quad \text{true} \\ ]| \end{array}$$

en

$$\begin{array}{l} |[\ n \leq N \ \wedge \ n \neq N \\ \triangleright \\ \quad (n \leq N)(n := n+1)(v := \mathcal{E}) \\ = \quad \{\ \text{subst.}\ \} \\ \quad n+1 \leq N \\ = \quad \{\ n \leq N \text{ en } n \neq N \text{ uit context}\ \} \\ \quad \text{true} \\ ]| \end{array}$$

Hieruit blijkt dat $0 \leq n$ en $n \leq N$ in isolatie invariant zijn.

4 Het koppatroon

We spreken van het **koppatroon**[5] als (de betreffende conjunct uit) de invariant de volgende vorm heeft:

$$v \quad = \quad F$$

waarbij v een programmavariabele is en F een uitdrukking in termen van de *andere* programmavariabelen, d.w.z. F hangt niet af van v.

In termen van ons programmafragment schrijven we dat liever als

$$v \quad = \quad \varphi.n$$

waarbij φ een geschikt gekozen functie is. Op college [3] (maar niet in [4]) wordt dit het φ-schema[6] genoemd. Bij het voorbeeld van machtsverheffen kunnen we hebben:

$$\varphi.n \quad = \quad X^n$$

Als je op zo'n invariant het basispatroon toepast, dan krijg je een berekening van de volgende vorm.

$$\begin{array}{ll}
|[& P \wedge B \\
\triangleright & \\
 & P(n := n+1)(v := \mathcal{E}) \\
= & \quad \{ \text{def. } P \} \\
 & (v = \varphi.n)(n := n+1)(v := \mathcal{E}) \\
= & \quad \{ \text{subst., } v \text{ hangt niet af van } n \text{ en } \varphi.n \text{ niet van } v \} \\
 & \mathcal{E} = \varphi.(n+1) \\
= & \quad \vdots \\
 & \mathcal{E} = \ldots \\
= & \quad \{ \bullet \text{ Kies } \mathcal{E} = \ldots \} \\
 & \text{true} \\
]| &
\end{array}$$

Bijvoorbeeld voor machtsverheffen met $P : v = X^n$ berekenen we:

$$\begin{array}{ll}
|[& P \wedge B \\
\triangleright & \\
 & P(n := n+1)(v := \mathcal{E}) \\
= & \quad \{ \text{def. } P \} \\
 & (v = X^n)(n := n+1)(v := \mathcal{E}) \\
= & \quad \{ \text{subst.} \} \\
 & \mathcal{E} = X^{n+1} \\
= & \quad \{ \text{eig. machtsverheffen} \} \\
 & \mathcal{E} = X^n * X \\
= & \quad \{ v = X^n \text{ uit context (zie } P) \} \\
 & \mathcal{E} = v * X \\
= & \quad \{ \bullet \text{ Kies } \mathcal{E} = v * X \} \\
 & \text{true} \\
]| &
\end{array}$$

[5] Tegenhanger van het staartpatroon, zie ook §8.

[6] Deze naam komt van Wim Feijen.

Op zich is deze berekening correct, maar de redactie kan beter.

Wat in de vorm van deze berekening opvalt is dat de linkerkant van de formules, te weten $\mathcal{E} = \ldots$, niet wijzigt. Dat deel kun je ook 'buiten haakjes halen'. In [4] doet zich dit voor het eerst voor in de berekening bovenaan p. 58, waar het 'buiten haakjes halen' zonder toelichting gebeurt, en de berekening pas in de conclusie gerelateerd wordt aan het invariantie-bewijs.

Het universele stuk van de berekening bij het koppatroon, dat daarom ook niet telkens vermeld hoeft te worden, luidt:

$$
\begin{array}{ll}
& |[\ P \wedge B \\
& \triangleright \\
& \quad P(n := n+1)(v := \mathcal{E}) \\
= & \quad \{\ \text{def. } P\ \} \\
& \quad (v = \varphi.n)(n := n+1)(v := \mathcal{E}) \\
= & \quad \{\ \text{subst., } v \text{ hangt niet af van } n \text{ en } \varphi.n \text{ niet van } v\ \} \\
& \quad \mathcal{E} = \varphi.(n+1) \\
= & \quad \{\ \bullet \text{ Kies } \mathcal{E} = \varphi.(n+1)\ \} \\
& \quad \text{true} \\
&]|
\end{array}
$$

Let wel dat de keuze $\mathcal{E} = \varphi.(n+1)$ niet betekent dat we letterlijk $\varphi.(n+1)$ kiezen voor $\mathcal{E}$. We mogen ook iets kiezen dat ermee gelijkwaardig is. De herschrijving van $\varphi.(n+1)$ dient ertoe om te komen tot een *programma*-expressie. Die herschrijving vermelden we wel.

De berekening bij het koppatroon $v = \varphi.n$ wordt als volgt opgeschreven.

$$
\begin{array}{ll}
& |[\ P \wedge B \\
& \triangleright \\
& \quad \varphi.(n+1) \\
= & \quad \{\ \ldots\ \} \\
& \quad \vdots \\
= & \quad \{\ \ldots\ \} \\
& \quad \ldots \\
&]|
\end{array}
$$

De berekening dient uit te monden in een geschikte programma-expressie, die we dan voor $\mathcal{E}$ kunnen invullen.

Zo passen we het koppatroon toe bij machtsverheffen met $\varphi.n = X^n$:

$$
\begin{array}{ll}
& |[\ P \wedge B \\
& \triangleright \\
& \quad \varphi.(n+1) \\
= & \quad \{\ \text{def. } \varphi\ \} \\
& \quad X^{n+1}
\end{array}
$$

$$
\begin{array}{ll}
= & \{ \text{ eig. machtsverheffen } \} \\
& X^n * X \\
= & \{ \ v = X^n \text{ uit context (zie } P) \ \} \\
& v * X \\
]|
\end{array}
$$

Aan de context van deze berekening kun je direct zien onder welke preconditie het resultaat geldig is. Dit is van belang voor de volgorde waarin de statements uiteindelijk in het programma opgeschreven mogen worden. Weglaten van de context, of het half afmaken van de berekening verhoogt de mentale belasting (zowel van de schrijver als van de lezer). Hoewel de volgende berekening de kern vangt in een recurrente betrekking voor φ, is de stap naar de programmatekst groter dan bij de voorgaande berekening.

$$
\begin{array}{ll}
& \varphi.(n+1) \\
= & \{ \text{ def. } \varphi \ \} \\
& X^{n+1} \\
= & \{ \text{ eig. machtsverheffen } \} \\
& X^n * X \\
= & \{ \text{ def. } \varphi \ \} \\
& \varphi.n * X
\end{array}
$$

5 Het staartpatroon

We spreken van het **staartpatroon**[7] als (de betreffende conjunct uit) de invariant de volgende vorm heeft:

$$F \quad = \quad C$$

waarbij F een uitdrukking in termen van de programmavariabelen is en C een constante, d.w.z. C hangt niet af van de programmavariabelen.

In termen van ons programmafragment schrijven we dat liever als

$$F.v.n \quad = \quad C$$

waarbij F een geschikt gekozen *functie* is. Bij het voorbeeld van machtsverheffen kunnen[8] we hebben:

$$
\begin{array}{rcl}
F.v.n & = & v * X^{N-n} \\
C & = & X^N
\end{array}
$$

Als je op zo'n invariant het basispatroon toepast, dan krijg je een berekening van de volgende vorm.

[7] De naam is geïnspireerd door het optreden bij staartinvarianten, zie ook §8.

[8] Het is wat mooier om $F.v.n = v * X^n$ te nemen met $n := n - 1$ in de body.

$$
\begin{array}{ll}
|[& P \wedge B \\
\triangleright & \\
 & P(n := n+1)(v := \mathcal{E}) \\
= & \quad \{ \text{def. } P \} \\
 & (F.v.n = C)(n := n+1)(v := \mathcal{E}) \\
= & \quad \{ \text{subst., } C \text{ hangt niet af van } v \text{ en } n \} \\
 & F.\mathcal{E}.(n+1) = C \\
= & \quad \{ F.v.n = C \text{ uit context (zie } P) \} \\
 & F.\mathcal{E}.(n+1) = F.v.n \\
= & \quad \vdots \quad \text{gereken aan } F.v.n \\
 & F.\mathcal{E}.(n+1) = F.(\ldots).(n+1) \\
= & \quad \{ \bullet \text{ Kies } \mathcal{E} = \ldots \} \\
 & \text{true} \\
]| &
\end{array}
$$

Het staartpatroon is lastiger dan het koppatroon, omdat er meer vrijheid is om te rekenen, waardoor men de weg nogal eens kwijt raakt. We hebben hier met opzet ervoor gekozen om zo vlug mogelijk C te elimineren en dan te rekenen aan $F.v.n$. Als de betrokken operaties *inversen* of zekere *schrapeigenschappen* hebben, dan kan het ook wel anders, zoals uit Appendix B blijkt. Maar in het algemeen is het beter zich daar niet door te laten verleiden.

Bijvoorbeeld voor machtsverheffen met $P : v * X^{N-n} = X^N$ berekenen we[9]:

$$
\begin{array}{ll}
|[& P \wedge n \neq N \quad \wedge n \leq N \\
\triangleright & \\
 & P(n := n+1)(v := \mathcal{E}) \\
= & \quad \{ \text{def. } P \} \\
 & \left(v * X^{N-n} = X^N\right)(n := n+1)(v := \mathcal{E}) \\
= & \quad \{ \text{subst.} \} \\
 & \mathcal{E} * X^{N-n-1} = X^N \\
= & \quad \{ v * X^{N-n} = X^N \text{ uit context (zie } P) \} \\
 & \mathcal{E} * X^{N-n-1} = v * X^{N-n} \\
= & \quad \{ \text{eig. machtsverheffen, } N-n > 0 \text{ vanwege } n < N \text{ o.g.v. context} \} \\
 & \mathcal{E} * X^{N-n-1} = v * \left(X * X^{N-n-1}\right) \\
= & \quad \{ \text{associativiteit van } * \} \\
 & \mathcal{E} * X^{N-n-1} = (v * X) * X^{N-n-1} \\
= & \quad \{ \bullet \text{ Kies } \mathcal{E} = v * X \} \\
 & \text{true} \\
]| &
\end{array}
$$

Op zich is deze berekening[10] correct, maar de redactie kan beter.

[9] Om delen door nul uit te sluiten is de contextaanname versterkt met $n \leq N$. Dit moet volgen uit de preconditie van $v := \mathcal{E}$, wat kan door ook de invariant te versterken met $n \leq N$. $X \neq 0$ zou ook volstaan. Maar daar gaat deze uiteenzetting niet over.

[10] We hebben dezelfde programma-expressie gevonden als bij behandeling via het koppatroon.

Wat in de vorm van deze berekening opvalt is dat de linkerkant van de formules, te weten $F.\mathcal{E}.(n+1) = \ldots$, niet wijzigt. Dat deel kun je 'buiten haakjes halen'. In [4] wordt deze observatie gedaan bij een speciaal geval van het staartpatroon op pp. 75–76.

Het universele stuk van de berekening bij het staartpatroon laat zich moeilijker in isolatie beschrijven dan bij het koppatroon. We doen daarom geen poging. Het hoeft niet telkens vermeld te worden. De herschrijving van het rechterlid $F.v.n$ tot $F.(\ldots).(n+1)$ vermelden we wel.

De berekening bij het staartpatroon $F.v.n = C$ wordt als volgt opgeschreven.

$$
\begin{array}{l}
|[\ P \wedge B \\
\triangleright \\
\quad F.v.n \\
=\quad \{\ \ldots\ \} \\
\quad \vdots \\
=\quad \{\ \ldots \} \\
\quad F.(\ldots).(n+1) \\
]|
\end{array}
$$

De berekening dient uit te monden in een vorm van F met als rechterargument $n+1$ en als linkerargument een geschikte programma-expressie, die we dan in het programma voor $\mathcal{E}$ kunnen invullen.

Zo passen we het staartpatroon toe bij machstverheffen met $F.v.n = v * X^{N-n}$:

$$
\begin{array}{l}
|[\ n \neq N \quad \wedge n \leq N \\
\triangleright \\
\quad F.v.n \\
=\quad \{\ \text{def. } F\ \} \\
\quad v * X^{N-n} \\
=\quad \{\ \text{eig. machtsverheffen, } N-n>0 \text{ vanwege } n<N \text{ o.g.v. context}\ \} \\
\quad v * \left(X * X^{N-n-1}\right) \\
=\quad \{\ \text{associativiteit van } *\ \} \\
\quad (v * X) * X^{N-n-1} \\
=\quad \{\ \text{def. } F\ \} \\
\quad F.(v * X).(n+1) \\
]|
\end{array}
$$

Merk op dat hierbij geen beroep meer is gedaan op P uit de context. Dit is bij het staartpatroon in het algemeen het geval. De invariant $P : F.v.n = C$ is gebruikt in het universele deel om C te elimineren. De guard $B : n \neq N$ en andere delen van de totale invariant (die in de een of andere vorm in de preconditie van de toekenning $v := \mathcal{E}$ kunnen terechtkomen) spelen mogelijk wel een rol (hier bijv. $n \leq N$ i.v.m. $N - n > 0$).

6 Initialisatie, finalisatie, terminatie

We hebben de berekeningspatronen geïdentificeerd in het kader van invariantie. Laten we nu stilstaan bij initialisatie, finalisatie en terminatie (eindiging).

Beschouw invariant $v = \varphi.n$, waarbij sprake is van het koppatroon. Voor postconditie $v = \varphi.N$ is finalisatie rechtstreeks te realiseren met guard $B : n \neq N$, immers

$$[\, v = \varphi.n \;\wedge\; n = N \;\;\Rightarrow\;\; v = \varphi.N \,]$$

Initialisatie van de invariant $v = \varphi.n$ vergt berekening van $\varphi.A$, waarbij A een geschikte beginwaarde voor n is. Bijvoorbeeld voor machtsverheffen met $\varphi.n = X^n$ en initialisatie van n met $n := 0$ berekenen we:

$$\begin{array}{l} |[\; 0 \leq N \\ \triangleright \\ \quad \varphi.0 \\ = \quad \{ \text{ def. } \varphi \} \\ \quad X^0 \\ = \quad \{ \text{ eig. machtsverheffen } \} \\ \quad 1 \\ ]| \end{array}$$

We zien in deze berekening hetzelfde patroon als bij invariantie volgens het koppatroon. Deze vorm van berekening is in het algemeen van toepassing in de situatie

$$\{\, Q \,\}\; v := \mathcal{E} \;;\, n := A \;\{\, v = \varphi.n \,\}$$

Dit is zowel het geval bij initialisatie (A is een geschikte "kleine" beginwaarde voor n) als bij invariantie (A is $n + 1$ en Q is $v = \varphi.n \;\wedge\; B$). De vorm van de berekening is ingegeven door de vorm van de betreffende postconditie $v = \varphi.n$; de vorm van de preconditie is hier niet van belang.

Beschouw nu invariant $F.v.n = C$, waarbij sprake is van het staartpatroon. Als we hebben $C = F.V.N$, dan kan initialisatie van deze invariant rechstreeks met $v, n := V, N$, immers

$$[\, \text{true} \;\;\Rightarrow\;\; (F.v.n = F.V.N)(v, n := V, N) \,]$$

Bijvoorbeeld voor machtsverheffen met $Fv.n = v * X^n$ en postconditie $v = X^N$ is $F.v.n = F.1.N$ een geschikte invariant (immers na afloop met $n = 0$ geldt $v = F.v.n = F.1.N = X^N$, zie ook hieronder bij finalisatie). De initialisatie kan dan met $v, n := 1, N$.

Voor postconditie $v = F.V.N \ldots$ Finalisatie vergt berekening van een guard B met $[\neg B \Rightarrow F.v.n = v]$. In dit geval hebben we te maken met een staartinvariant[11].

[11] Aan de invariant alleen kun je niet zien dat het om een staartinvariant gaat. De postconditie speelt daarbij ook een rol.

Het staartpatroon biedt meer vrijheid dan het koppatroon. Immers bij het koppatroon $v = \varphi.n$ is de waarde van v eenduidig vastgelegd door de waarde van n, terwijl bij het staartpatroon $F.v.n = C$ meer combinaties[12] toegelaten kunnen zijn. Bij het staartpatroon is sprake van een (algemene) relatie, terwijl bij het koppatroon sprake is van een beperkte relatie, namelijk een functie.

Die extra vrijheid geeft het staartpatroon een tweetal voordelen boven het koppatroon:

1. Grotere stappen, meerdere soorten stappen mogelijk; vf-verlagend statement hoeft niet eenduidig vooraf gekozen te worden.

2. Vanzelfsprekende vroegtijdige beeindiging.

Beschouw machsverheffen met postconditie $v = X^N$ en staartinvariant $v * x^n = X^N$ (er is nu een extra programmavariabele x). Initialisatie gaat met $v, x, n := 1, X, N$. Laten we vervolgens de guard voor finalisatie uitrekenen:

$$
\begin{array}{ll}
 & v * x^n = v \\
= & \quad \{ \text{ gevalsonderscheid naar gelang } v = 0, \text{eig.} * \} \\
 & v = 0 \ \vee \ x^n = 1 \\
= & \quad \{ \text{ eig. machtsverheffen } \} \\
 & v = 0 \ \vee \ x = 1 \ \vee \ n = 0
\end{array}
$$

Dus de sterkst mogelijke guard is

$$v \neq 0 \ \wedge \ x \neq 1 \ \wedge \ n \neq 0$$

Wat betreft de body zijn er twee eenvoudige manieren om $v * x^n$ vormbehoudend te manipuleren:

$$
\begin{array}{lcll}
v * x^n & = & (v * x) * x^{n-1} & \text{als } 0 < n \\
v * x^n & = & v * (x * x)^{n \mathbf{div} 2} & \text{als } n \text{ even}
\end{array}
$$

Beide manieren verlagen n mits aanvankelijk $0 < n$; de tweede manier is sneller, maar minder universeel toepasbaar. Gezien $n \neq 0$ in de guard, kunnen we in de body dus volstaan met de selectie:

$$
\begin{array}{l}
\textbf{if } \text{true} \quad \rightarrow v, n := v * x, n - 1 \\
[] \ n \ \textbf{mod} \ 2 = 0 \rightarrow x, n := x * x, n \ \textbf{div} \ 2 \\
\textbf{fi}
\end{array}
$$

De eerste guard van de selectie kan versterkt worden tot $n \ \textbf{mod} \ 2 \neq 0$ om zo snelheid af te dwingen.

[12] Of er ook meer combinaties bij executie optreden is een ander verhaal.

7 Integratie

8 Slotopmerkingen

De naamgeving kop- versus staartpatroon suggereert dualiteit, maar daar is slechts gedeeltelijk sprake van. Bij het koppatroon $v = \varphi.n$ begin je voor het ontwerp van de body te rekenen aan $\varphi.(n+1)$ en is het doel een programma-expressie, bijvoorbeeld in termen van $\varphi.n$. Bij het staartpatroon $F.v.n = C$ begin je te rekenen aan $F.v.n$ (dus zonder substitutie) en is het doel van de vorm $F.(\ldots).(n+1)$ met op de puntjes een programma-expressie.

Ik hoop met deze inventarisatie een bijdrage te hebben geleverd aan het voorkomen van zogenaamde *kop-noch-staart-berekeningen* (:-).

8.1 Open eindjes

- Andere namen: koppatroon→functionele patroon, staartpatroon→(homogeen) relationele patroon?
- Patronen sluiten elkaar niet uit. Voor de ene conjunct kan het staartpatroon van toepassing zijn, terwijl voor een andere conjunct in dezelfde invariant het koppatroon gebruikt wordt.
- Uitleggen wanneer een recurrente betrekking staartrecursief heet.
- Koppatroon is vaak (altijd?) te herschrijven tot een overeenkomstig staartpatroon dat aanleiding geeft tot hetzelfde programma. Maar pas op: bij machtsverheffen heb je aan $0 = v - \varphi.n$ i.h.a. niet veel. Omgekeerd, vanuit staartpatroon een overeenkomstig koppatroon bepalen, gaat soms (vereist i.h.a. schrapeigenschap).
- Zuinigere contexten opschrijven en programma-onderdelen samenstellen.
- Stappenplannen (kop, staart; nu impliciet in de volgorde van de onderdelen in de samenvatting).
- Staartpatroon noteren als $C = E$?
- Voordeel staartpatroon boven koppatroon (komt later bij slope search): doorzochte gebied versus te doorzoeken gebied; het tweede is soms eenvoudiger te beschrijven dan het eerste.
- Niets over selectie vermeld.

A Samenvatting van de berekeningspatronen

Ik vat hier de patronen samen in het volgende formaat:

Vorm van invariant Welke vorm de invariant heeft.

Bron Hoe je aan zo'n invariant kan komen.

Finalisatie Welke vorm de finalisatie heeft en hoe je de guard berekent.

Initialisatie Welke vorm de initialisatie heeft.

Invariantie Welke vorm de invariantie-berekening heeft.

Het ontbreekt ons aan een goede notatie om de patronen in volle algemeenheid te beschrijven. En het is maar de vraag of zo'n beschrijving de beginner zou helpen. De lezer dient de patronen dus voldoende te begrijpen om ze te kunnen toepassen in de onderhavige situatie, met meer of minder variabelen, of met een andere variante functie en verlaging ervan.

We gaan uit van de volgende generieke repetitie.

$$
\begin{array}{ll}
 & \{\ \text{pre: } Q\ \} \\
 & \mathit{init} \\
; & \{\ \text{inv: } P\ \} \\
 & \textbf{do } B \\
 & \rightarrow \{\ P \wedge B\ \} \\
 & \quad \mathit{body} \\
 & \quad \{\ P\ \} \\
 & \textbf{od} \\
; & \{\ P \wedge \neg B\ \} \\
 & \mathit{fini} \\
 & \{\ \text{post: } R\ \}
\end{array}
$$

Hierin is Q de preconditie, *init* het initialiserende statement, P de invariant, B de guard, *body* het herhaalde statement, *fini* het finaliserende statement en R de postconditie. Tenzij anders vermeld leggen we de patronen uit aan de hand van de *body*

$$v := \mathcal{E}\ ;\ n := n + 1$$

met $\mathcal{E}$ een nader te bepalen *programma*-expressie en $n := n + 1$ het statement dat de variante functie verlaagt. I.h.a. kunnen meer variabelen een rol spelen en kan de variante functie op een andere manier verlaagd worden.

A.1 Basispatroon

Vorm van invariant Er is geen specifieke vorm. Kan altijd toegepast worden, maar als een ander patroon ook van toepassing is, dan heeft dat de voorkeur.

Bron Elke invariant past bij dit patroon. Denk aan technieken als

1. een conjunct laten vallen uit de postconditie,

2. een constante in de postconditie vervangen door een nieuwe programmavariabele.

Finalisatie Probeer skip voor *fini* en zoek guard B zó dat

$$[\ P \wedge \neg B \quad \Rightarrow \quad R\ ]$$

Hiervoor is geen algemene berekeningsvorm. Als de invariant echter is verkregen uit de postconditie door

laten vallen van conjunct dan ontkenning van die conjunct als guard nemen;

variatie van constante dan guard B zó kiezen dat uit $\neg B$ volgt dat de waarde van de geïntroduceerde programmavariabele gelijk is aan de constante in de postconditie.

Initialisatie Kies voor init

$$v := \mathcal{E}' \ ;\ n := A$$

met geschikte beginwaarde A voor n en bepaal $\mathcal{E}'$ als volgt:

$$\begin{array}{l}
|[\ Q \\
\triangleright \\
\quad P(n := A)(v := \mathcal{E}') \\
= \quad \{\ \text{def. } P, \text{subst.}\ \} \\
\quad \ldots \\
= \quad \{\ \bullet \text{ Kies } \mathcal{E}' = \ldots\ \} \\
\quad \ldots \\
= \quad \{\ \ldots\ \} \\
\quad \text{true} \\
]|
\end{array}$$

Invariantie Wordt bij voorkeur als volgt opgezet:

$$\begin{array}{l}
|[\ P \wedge B \\
\triangleright \\
\quad P(n := n + 1)(v := \mathcal{E}) \\
= \quad \{\ \text{def. } P, \text{subst.}\ \} \\
\quad \ldots \\
= \quad \{\ \bullet \text{ Kies } \mathcal{E} = \ldots\ \} \\
\quad \ldots \\
= \quad \{\ \ldots\ \} \\
\quad \text{true} \\
]|
\end{array}$$

Guard B in de context hoeft niet gebruikt te worden, evenmin als invariant P.

A.2 Conjunctiepatroon

Vorm van invariant $P : P0 \wedge P1$.

Bron
1. Conjuncten kunnen uit de postconditie komen, eventueel na generalisatie.
2. Een conjunct kan ook later geïntroduceerd worden als **versterking**, bijvoorbeeld als grens op een programmavariabele of bij introductie van een verse programmavariabele om een niet-programma-expressie te omzeilen (zie koppatroon).
3. Merk op dat begrenzing $a \leq n \leq b$ te splitsen is als $a \leq n \ \wedge \ n \leq b$.

Finalisatie
1. Mogelijk is de postconditie R al te realiseren op grond van één van de conjuncten en de ontkenning van de guard. Verder als basispatroon.
2. Als de postconditie R ook de vorm van een conjunctie heeft, dan kunnen mogelijk delen van de postconditie gerealiseerd worden o.g.v. delen uit de invariant eventueel samen met de ontkenning van de guard.
3. Het later toevoegen van conjuncten aan de invariant brengt geen nieuwe bewijsverplichtingen met zich mee voor finalisatie.

Initialisatie Handel elke conjunct afzonderlijk af, vergelijk met invariantie.

Invariantie Handel elke conjunct afzonderlijk af:

$$
\begin{array}{l}
|[\ P0 \wedge P1 \wedge B \\
\triangleright \\
\quad Pi(n := n+1)(v := \mathcal{E}) \\
= \quad \{ \text{ def. } Pi, \text{subst. } \} \\
\quad \ldots \\
= \quad \{ \bullet \text{ Kies } \mathcal{E} = \ldots \} \\
\quad \ldots \\
= \quad \{ \ldots \} \\
\quad \text{true} \\
]|
\end{array}
$$

De contextaanname mag *alle* conjuncten en de guard bevatten, maar deze hoeven niet gebruikt te worden.

A.3 Koppatroon

Vorm van invariant $P : v = E$, waarbij v een programmavariabele is en E een uitdrukking in termen van de *andere* programmavariabelen, d.w.z. E hangt niet af van v. We schrijven daarom liever $P : v = \varphi.n$ voor geschikte functie φ. Dit staat ook bekend als het φ-schema.

Bron

1. Variatie van constante toepassen op (een conjunct uit) de postconditie R van de vorm $v = \varphi.N$. Hierbij wordt een nieuwe programmavariabele n geïntroduceerd.
2. Tijdens het ontwerpen treedt een expressie $\varphi.n$ op, die men in het programma zou willen gebruiken, maar die zelf geen programma-expressie is. Introduceer dan een nieuwe programmavariabele v en versterk de invariant met $v = \varphi.n$ (zie ook conjunctiepatroon).

   Als alternatief kan ook $v = \varphi.(n-1)$ o.i.d. genomen worden, mits v vóór gebruik wordt bijgewerkt t.b.v. $P(n := n+1)$, opdat $v = \varphi.n$ geldt.

Finalisatie Als de postconditie R de vorm $v = \varphi.N$ heeft, probeer dan guard $B : n \neq N$ en neem skip voor *fini*. Als v niet in de postconditie R vóórkomt, dan is er geen bewijsverplichting m.b.t. finalisatie.

Initialisatie Neem voor *init* het statement $v, n := \varphi.A,\ A$, waarbij A een geschikte beginwaarde is voor n en bereken

$$
\begin{array}{l}
|[\ Q \\
\triangleright \\
\quad \varphi.A \\
= \quad \vdots \\
\quad \ldots \\
]|
\end{array}
$$

De kleinste toegelaten waarde van N is veelal een geschikte keuze voor A. De postconditie speelt geen directe rol bij initialisatie.

Invariantie Neem $\mathcal{E} = \varphi.(n+1)$ en bereken daartoe

$$
\begin{array}{l}
|[\ P \wedge B \\
\triangleright \\
\quad \varphi.(n+1) \\
= \quad \vdots \\
\quad \ldots \\
]|
\end{array}
$$

De berekening dient uit te monden in een geschikte programma-expressie, die we voor $\mathcal{E}$ kunnen invullen.

A.4 Staartpatroon

Vorm van invariant $P : E = C$, waarbij E een uitdrukking in termen van de programmavariabelen is en C een constante, d.w.z. C hangt niet af van de

programmavariabelen. We schrijven liever $P : F.v.n = C$ voor geschikte functie F.

Merk op dat C wel kan afhangen van programma*constanten*, bijvoorbeeld **con** N: int.

Als bovendien (een conjunct van) de postconditie R de vorm $v = C$ heeft, dan spreken we van een **staartinvariant**. De invariant drukt de te berekenen waarde C uit als *nog uit te voeren berekening* F op v en n, de zongenaamde staart.

Bron (*Dit is nog niet naar mijn zin.*)

1. Als (een conjunct van) de postconditie R de vorm $v = C$ of zelfs $G.v = C$ heeft, zoek dan een generalisatie van C van de vorm $F.v.n$ met een nieuwe programmavariabele n. De vorm van $F.v.n$ dient behouden te blijven onder geschikte wijzigingen van v en n, die de variante functie verlagen. Voor verdere eisen aan de generalisatie F zie finalisatie en initialisatie.
2. Het is vaak nuttig om voor een paar waarden van de programmaconstanten die in C voorkomen, het probleem handmatig uit te werken, om zo te zien welke vormen een rol spelen.
3. Als (een conjunct van) de postconditie R de vorm $v = F.V.N$ heeft voor geschikte waarden V en N, probeer dan $C = F.V.N$.

Finalisatie Onderstel postconditie $R : v = C$.

1. Neem skip voor *fini* en bepaal guard B uit:

$$\begin{array}{ll} & F.v.n = v \\ \Leftarrow & \vdots \\ & \neg B \end{array}$$

2. Bepaal guard B zó dat

$$[\, \neg B \quad \Rightarrow \quad F.v.n = \mathcal{E}'' \,]$$

voor een geschikte programma-expressie $\mathcal{E}''$. Kies $v := \mathcal{E}''$ voor *fini*.

Initialisatie $\ldots\ v, n := V, N$

Invariantie De berekening wordt als volgt opgeschreven:

$$\begin{array}{l} |[\ P \wedge B \\ \triangleright \\ \quad F.v.n \\ = \quad \vdots \\ \quad F.(\ldots).(n+1) \\ ]| \end{array}$$

De berekening dient uit te monden in een vorm van F met als rechterargument $n+1$ en als linkerargument een geschikte programma-expressie, die we voor $\mathcal{E}$ kunnen invullen.

De invariantie-berekening hangt *niet* af van de vorm van de postconditie R.

Mogelijk nog specialisaties op staartpatroon onderscheiden.

B Missers bij het staartpatroon

Bij het staartpatroon is er veel gelegenheid om het anders te doen. Hier zijn drie minder geslaagde berekeningen voor machstverheffing met $P : v * X^{N-n} = X^N$. De eerste twee benutten de inverse van vermenigvuldigen (ga na!). De eerste loopt min of meer vast door P niet te gebruiken:

$$
\begin{array}{ll}
|[& P \wedge B \quad \wedge n \leq N \\
\triangleright & \\
& P(n := n+1)(v := \mathcal{E}) \\
= & \quad \{ \text{def. } P \} \\
& \left(v * X^{N-n} = X^N\right)(n := n+1)(v := \mathcal{E}) \\
= & \quad \{ \text{subst.} \} \\
& \mathcal{E} * X^{N-n-1} = X^N \\
= & \quad \{ \text{eig. machtsverheffen, } N-n > 0 \text{ vanwege } n \leq N \text{ in context} \} \\
& \mathcal{E} = X^{n+1} \\
= & \quad \{ \bullet \text{ Kies } \mathcal{E} = ? \} \\
& \text{true} \\
]| &
\end{array}
$$

Je kan je nog wel redden door op te merken dat uit P volgt $v = X^n$, dus $\mathcal{E} = v * X$. Maar dan had je beter meteen het koppatroon kunnen gebruiken.

De tweede gebruikt P wel, maar rekent aan *beide* leden:

$$
\begin{array}{ll}
|[& P \wedge B \quad \wedge n \leq N \\
\triangleright & \\
& P(n := n+1)(v := \mathcal{E}) \\
= & \quad \{ \text{def. } P \} \\
& \left(v * X^{N-n} = X^N\right)(n := n+1)(v := \mathcal{E}) \\
= & \quad \{ \text{subst.} \} \\
& \mathcal{E} * X^{N-n-1} = X^N \\
= & \quad \{ v * X^{N-n} = X^N \text{ uit context (zie } P) \} \\
& \mathcal{E} * X^{N-n-1} = v * X^{N-n} \\
= & \quad \{ \text{eig. machtsverheffen, } N-n > 0 \text{ vanwege } n \leq N \text{ in context} \} \\
& \mathcal{E} = v * X \\
= & \quad \{ \bullet \text{ Kies } \mathcal{E} = v * X \} \\
& \text{true} \\
]| &
\end{array}
$$

Dit lijkt ideaal, omdat je meteen ziet wat $\mathcal{E}$ kan zijn. Maar helaas kun je vaak niet zoveel tegen elkaar laten wegvallen, omdat de betrokken operatoren niet voldoende schrapeigenschappen hebben.

Tenslotte nog een versie die juist aan het *linkerlid* rekent:

$$
\begin{array}{ll}
|[& P \wedge B \quad \wedge n \leq N \\
\triangleright & \\
& P(n := n+1)(v := \mathcal{E}) \\
= & \quad \{ \text{def. } P \} \\
& \left(v * X^{N-n} = X^N\right)(n := n+1)(v := \mathcal{E}) \\
= & \quad \{ \text{subst.} \} \\
& \mathcal{E} * X^{N-n-1} = X^N \\
= & \quad \{ v * X^{N-n} = X^N \text{ uit context (zie } P) \} \\
& \mathcal{E} * X^{N-n-1} = v * X^{N-n} \\
= & \quad \{ \text{eig. machtsverheffen, } N-n > 0 \text{ vanwege } n \leq N \text{ in context} \} \\
& \frac{\mathcal{E}}{X} * X^{N-n} = v * X^{N-n} \\
= & \quad \{ \bullet \text{ Kies } \mathcal{E} \text{ zó dat } \mathcal{E}/X = v, \text{ ofwel } \mathcal{E} = v * X \} \\
& \text{true} \\
]| &
\end{array}
$$

In al deze berekeningen zijn er zorgen m.b.t. $X = 0$, omdat delen door X, de inverse van vermenigvuldigen met X, dan niet gedefinieerd is. Merk op dat we die zorg niet hadden bij het koppatroon.

Referenties

[1] E. Gamma, et al. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995.

[2] B. Meyer. *Object-Oriented Software Construction*. *Second Edition*. Prentice Hall, 1997.

[3] 2IA10, *Ontwerp van Algoritmen 1*. TUE, Informatica. Internet: `www.win.tue.nl/~wstomv/edu/2ia10/`.

[4] A. Kaldewaij. *Programming: The Derivation of Algorithms*. Prentice Hall, 1990.