

Ontwerp van Algoritmen 1: OpgavenBundel

Rob R. Hoogerwoord

22 november 2006

Contents

0	Inleiding	1
1	Propositie- en predikatenrekening	2
1.0	Inleiding	2
1.1	Notatie	2
1.2	Basisregels	3
1.3	Afgeleide regels	4
1.4	Sterker en Zwakker	5
1.5	Rekenen in context	7
1.6	Toegift	9
1.7	Opgaven	9
1.8	Quantoren: basisregels	11
1.9	Opgavenselectie	12
2	Skip, assignment en compositie	14
3	Specificatie-oefeningen	17
4	Selectie	19
5	Bewijs- en specificatie-oefeningen	21
6	Div en mod	23
7	Repetitie	24
8	Het stappenplan	29
9	Programmeeropgaven	30
9.0	Basistechnieken	30
9.1	Invariant versterken	31
9.2	Geneste repetities	32
9.3	Staartinvarianten	32
9.4	Gemengde opgaven	33

Chapter 0

Inleiding

De opgaven in deze bundel zijn gegroepeerd naar verschillende thema's. Per thema zijn de opgaven (min of meer) geordend naar oplopende moeilijkheid. Hier en daar zijn samenvattinkjes toegevoegd van belangrijke stukjes theorie. Als bijlage zijn toegevoegd: de tentameneisen voor dit vak en voorbeelden van tentamenopgaven.

Voor alle opgaven geldt: “Boek” verwijst naar *Programming: The Derivation of Algorithms* van Anne Kaldewaij. Bij voorkeur worden de opgaven opgelost door te rekenen!

Smaakmaker

0. Gegeven is een natuurlijke constante N en een integer functie f , gedefinieerd op het interval $[0..N)$; dit betekent dat $f \cdot i$ (uitsluitend) is gedefinieerd voor alle i met $0 \leq i < N$. We definiëren nu een functie sum , als volgt: $sum \cdot p \cdot q$ is de som van alle functiewaarden van f op het deelinterval $[p..q)$, voor alle p en q die voldoen aan $0 \leq p \leq q \leq N$. In formule: $sum \cdot p \cdot q = (\sum i : p \leq i < q : f \cdot i)$.

Schrijf, met gebruikmaking van het geleerde bij het vak *Programmarealisatie* en zo zorgvuldig mogelijk, een programma ter berekening van de maximale waarde van $sum \cdot p \cdot q$ over alle paren p en q waarvoor $0 \leq p \leq q \leq N$. Hoe efficiënt is uw programma?

Chapter 1

Propositie- en predikatenrekening

1.0 Inleiding

Dit is een samenvatting van de rekenregels voor proposities en predikaten, zoals behandeld in het vak Logica & Verzamelingen. Enige vertrouwde met dit vak wordt dan ook verondersteld. De regels voor quantoren worden apart behandeld. We besluiten deze paragraaf met een selectie uit de opgaven.

Voor het praktische gebruik van de logica is niet van belang dat de verzameling regels zo *klein mogelijk* is – op welke manier dan ook –; integendeel: hoe meer regels ons ter beschikking staan des te effectiever we er mee kunnen werken. Om louter didactische redenen zijn de regels toch verdeeld in twee klassen. De *afgeleide regels* kunnen uit de *basisregels* worden afgeleid; dit is goede oefenstof. De term *basisregels* bedoelt overigens niet te suggeren dat deze een minimale collectie zouden vormen: dat is niet zo⁰.

Voor de praktijk is de opdeling in basisregels en afgeleide regels van geen belang: alle regels mogen door elkaar worden gebruikt. Vandaar ook dat alle regels namen hebben.

1.1 Notatie

De bewering “ P is een tautologie” korten we af tot $[P]$. (We spreken dit uit als “overall P ”.) De rechte haken maken geen deel uit van de eigenlijke formule¹: $[P]$ is zelf geen formule maar een *uitspraak over* formule P .

In plaats van het symbol $\stackrel{val}{\equiv}$ gebruiken we gewoon $=$ en in plaats van $\stackrel{val}{|}\equiv$ gebruiken we \Rightarrow met rechte haken: $P \stackrel{val}{|}\equiv Q$ schrijven we als $[P \Rightarrow Q]$.

In berekeningen – met tussen accoladen vermelding van de gebruikte regels – gebruiken we $=$ en \Rightarrow , zonder rechte haken: hier is toch geen verwarring mogelijk. Zie paragraaf 1.4 voor voorbeelden.

Van Logica & Verzamelingen weten we dat de bewering “ P is een tautologie” even waar is als de bewering $P \stackrel{val}{=} \text{true}$. In onze schrijfwijze wordt dat dus: (de bewering) $[P]$ is even waar als (de bewering) $P = \text{true}$.

In plaats van het symbol \Leftrightarrow (“bi-implicatie”) gebruiken we het symbol \equiv (“equivalent”). Van Logica & Verzamelingen weten we dat $P \stackrel{val}{=} Q$ even waar is als $P \Leftrightarrow Q \stackrel{val}{=} \text{true}$,

⁰Uit een regel kan zijn *duale* vaak worden afgeleid, maar om een keuze te vermijden zijn ze dan toch beide opgenomen; voorbeeld: de Morgan.

¹Het is wel mogelijk $[\cdot]$ in het formalisme op te nemen – $[P]$ is immers eigenlijk ook een propositie –, maar hier doen we dat niet; zie verder opgaven 30 t/m 33.

en dat is weer even waar als “ $P \Leftrightarrow Q$ is een tautologie”. In onze schrijfwijze wordt dat dus: $P = Q$ is even waar als $P \equiv Q = \text{true}$, en dat is weer even waar als $[P \equiv Q]$. Zowel $P = Q$ als $[P \equiv Q]$ drukken dus gelijkheid van P en Q uit.

Het gebruik van zowel $P = Q$ als $[P \equiv Q]$ heeft het voordeel dat we die vorm kunnen hanteren die het beste past bij onze behoeften. Bijvoorbeeld, $P = Q$ herinnert ons aan de regel van Leibniz: als $P = Q$ mogen we in elke formule P door Q vervangen (en vice versa). De andere vorm is juist handig in de rekenregels. Bijvoorbeeld: we hebben dat $P \equiv \text{true}$ gelijkwaardig is met P ; deze regel kunnen we zó schrijven: $P \equiv \text{true} = P$ maar ook zó: $[P \equiv \text{true} \equiv P]$. Nu krijgen we zomaar een extra eigenschap cadeau! Omdat \equiv associatief en commutatief is, is dit immers hetzelfde als $[P \equiv P \equiv \text{true}]$, waaruit volgt dat $P \equiv P$ gelijk is aan true .

Voor substitutie gebruiken we het symbool $:=$ (“wordt”): $P(x := E)$ is de formule die uit P is verkregen door *alle* vrije voorkomens van de variabele x in P te vervangen door de formule E . Substitutie kan worden herhaald; bijvoorbeeld: $P(x := E)(y := F)$ is de formule verkregen door de substitutie $y := F$ toe te passen op $P(x := E)$. Opeenvolgende substituties dienen dus *van links naar rechts* te worden uitgewerkt. Simultane substitutie van twee variabelen wordt zó genoteerd: $P(x, y := E, F)$.

We herhalen hier de *bindingsvolgorde*² van de operatoren, in volgorde van afnemende bindingskracht:

$$\begin{array}{l} \neg \\ \wedge, \vee \\ \Rightarrow, \Leftarrow \\ \equiv, \neq \\ = \end{array}$$

Voor \Leftarrow en \neq zijn geen rekenregels opgenomen; al hun eigenschappen kunnen worden afgeleid uit de regels voor \Rightarrow respectievelijk \equiv , en de volgende definities:

$$P \Leftarrow Q = Q \Rightarrow P, \text{ en:}$$

$$P \neq Q = \neg(P \equiv Q) .$$

1.2 Basisregels

In de volgende regels staan de hoofdletters P, Q, R voor willekeurige predikaten. De logische operatoren $\wedge, \vee, \equiv, \neq$ zijn associatief and commutatief, maar dit wordt niet in de regels uitgedrukt: deze eigenschappen gebruiken we stilzwijgend. Let op: \Rightarrow en \Leftarrow zijn *noch* commutatief *noch* associatief!

Alle regels drukken tautologieën uit, en wel gelijkheden; vandaar dat ze alle van de vorm $[\dots \equiv \dots]$ zijn.

nulelement:

$$\begin{array}{l} [P \vee \text{true} \equiv \text{true}] \\ [P \wedge \text{false} \equiv \text{false}] \end{array}$$

²Ook wel *prioriteiten* genoemd.

eenheidselement:

$$\begin{aligned} [P \vee \text{false} &\equiv P] \\ [P \wedge \text{true} &\equiv P] \\ [P &\equiv \text{true} \equiv P] \end{aligned}$$

dubbele negatie:

$$[\neg\neg P \equiv P]$$

uitsluitingen:

$$\begin{aligned} [P \vee \neg P &\equiv \text{true}] \\ [P \wedge \neg P &\equiv \text{false}] \end{aligned}$$

de Morgan:

$$\begin{aligned} [\neg(P \vee Q) &\equiv \neg P \wedge \neg Q] \\ [\neg(P \wedge Q) &\equiv \neg P \vee \neg Q] \end{aligned}$$

distributie:

$$\begin{aligned} [P \wedge (Q \vee R) &\equiv (P \wedge Q) \vee (P \wedge R)] \\ [P \vee (Q \wedge R) &\equiv (P \vee Q) \wedge (P \vee R)] \\ [P \vee (Q \equiv R) &\equiv (P \vee Q) \equiv (P \vee R)] \end{aligned}$$

implicatie:

$$[P \Rightarrow Q \equiv \neg P \vee Q]$$

equivalentie:

$$[P \equiv Q \equiv (P \Rightarrow Q) \wedge (Q \Rightarrow P)]$$

1.3 Afgeleide regels

constanten:

$$\begin{aligned} [\neg \text{true} &\equiv \text{false}] \\ [\neg \text{false} &\equiv \text{true}] \end{aligned}$$

nulelement voor \Rightarrow :

$$\begin{aligned} [P \Rightarrow \text{true} &\equiv \text{true}] \\ [\text{false} \Rightarrow P &\equiv \text{true}] \end{aligned}$$

eenheidselement voor \Rightarrow :

$$\begin{aligned} [P \Rightarrow \text{false} &\equiv \neg P] \\ [\text{true} \Rightarrow P &\equiv P] \end{aligned}$$

uitsluiting voor \equiv :

$$[P \equiv \neg P \equiv \text{false}]$$

haasje over:

$$[\neg P \equiv Q \equiv P \equiv \neg Q]$$

idempotentie:

$$\begin{aligned} [P \vee P &\equiv P] \\ [P \wedge P &\equiv P] \end{aligned}$$

absorptie:

$$\begin{aligned} [P \vee (P \wedge Q) &\equiv P] \\ [P \wedge (P \vee Q) &\equiv P] \end{aligned}$$

complement(absorptie):

$$\begin{aligned} [P \vee (\neg P \wedge Q) &\equiv P \vee Q] \\ [P \wedge (\neg P \vee Q) &\equiv P \wedge Q] \end{aligned}$$

\wedge/\Rightarrow -associatie:

$$[P \wedge Q \Rightarrow R \equiv P \Rightarrow (Q \Rightarrow R)]$$

\Rightarrow/\vee -associatie:

$$[(P \Rightarrow Q) \vee R \equiv P \Rightarrow Q \vee R]$$

contrapositie:

$$[\neg P \Rightarrow \neg Q \equiv Q \Rightarrow P]$$

de Morgan voor \Rightarrow :

$$[\neg(P \Rightarrow Q) \equiv P \wedge \neg Q]$$

uit het ongerijmde:

$$[P \equiv \neg P \Rightarrow P]$$

implicatie voor gevorderden:

$$\begin{aligned} [P \Rightarrow Q &\equiv P \equiv P \wedge Q] \\ [P \Rightarrow Q &\equiv Q \equiv P \vee Q] \end{aligned}$$

Golden Rule:

$$[P \wedge Q \equiv P \vee Q \equiv P \equiv Q]$$

1.4 Sterker en Zwakker

Als $[P \Rightarrow Q]$ zeggen we dat P *sterker* is dan Q . Eigenlijk is dit taalgebruik niet juist, want iedere P is nu ook sterker dan zichzelf; er geldt immers $[P \Rightarrow P]$. We zouden eigenlijk moeten zeggen P is *ten minste zo sterk als* Q , maar omdat dit zo'n mond vol is, en omdat het nu eenmaal is ingeburgerd, gebruiken we toch gewoon *sterker dan*. Ook zeggen we wel dat Q *zwakker* is dan P ; dit drukt hetzelfde uit als “ P is sterker dan Q ”, net zo als $[Q \Leftarrow P]$ hetzelfde is als $[P \Rightarrow Q]$. Kortom de relaties “sterker dan” en “zwakker dan” zijn *reflexief*³.

Zoals we weten is $[P]$ hetzelfde als $[P \equiv \text{true}]$. Maar dit is hetzelfde als $[(P \Rightarrow \text{true}) \wedge (\text{true} \Rightarrow P)]$, en dit kunnen we, omdat $[P \Rightarrow \text{true}]$, vereenvoudigen tot

³Ze zijn ook anti-symmetrisch en transitief: het zijn dus ordeningen.

$[\text{true} \Rightarrow P]$, wat we ook wel schrijven als $[P \Leftarrow \text{true}]$. In woorden: $[P]$ is hetzelfde als de mededeling dat P ten minste zo zwak is als true .

Om $[P]$ te bewijzen kunnen we dus volstaan met P te versterken tot true ; daarom mogen we in een berekening, die begint met P en die ten doel heeft in true te eindigen, behalve gelijkheden ook versterkingen gebruiken. Een bewijs van $[P]$ door middel van berekening kan er dus zó uit zien:

$$\begin{aligned}
 & P \\
 = & \quad \{ \text{reden waarom } [P \equiv Q_0] \} \\
 & Q_0 \\
 \Leftarrow & \quad \{ \text{reden waarom } [Q_0 \Leftarrow Q_1] \} \\
 & Q_1 \\
 = & \quad \{ \text{reden waarom } [Q_1 \equiv Q_2] \} \\
 & Q_2 \\
 & \vdots \\
 \Leftarrow & \quad \{ \text{reden waarom } [\dots \Leftarrow Q_n] \} \\
 & Q_n \\
 = & \quad \{ \text{reden waarom } [Q_n \equiv \text{true}] \} \\
 & \text{true} .
 \end{aligned}$$

Kortom, we mogen wat we moeten bewijzen gaandeweg versterken: als we iets sterkers dan P bewijzen dan bewijzen we natuurlijk P zelf ook.

Pas Op!: We moeten natuurlijk wel *zuinig* zijn met versterkingen, anders lopen we het gevaar dat we een *te sterke* formule verkrijgen, die niet meer uit true kan worden afgeleid. Dat is weliswaar niet fout, maar wel loopt onze bewijspoging in zo'n geval muurvast. Hier is een (extreem en dus flauw) voorbeeldje van dit verschijnsel:

$$\begin{aligned}
 & P \\
 \Leftarrow & \quad \{ [P \Leftarrow \text{false}] \text{ (false is sterker dan alle predikaten)} \} \\
 & \text{false} \\
 \Leftarrow & \quad \{ ??? \text{ (nu komen we nooit meer tot true!)} \} \\
 & ???
 \end{aligned}$$

□

De volgende regels vinden toepassing bij het verzwakken van predikaten. Ze kunnen natuurlijk ook worden gebruikt om predikaten te versterken, namelijk door ze van rechts naar links te lezen: $P \Rightarrow Q$ mag je immers ook schrijven als $Q \Leftarrow P$.

\wedge/\vee -verzwakking:

$$\begin{aligned} & [P \wedge Q \Rightarrow P] \\ & [P \Rightarrow P \vee Q] \end{aligned}$$

monotonie van \wedge en \vee :

$$\begin{aligned} & \text{als } [P \Rightarrow Q] \text{ dan ook } [P \wedge R \Rightarrow Q \wedge R] \\ & \text{als } [P \Rightarrow Q] \text{ dan ook } [P \vee R \Rightarrow Q \vee R] \end{aligned}$$

Negatie is juist *niet* monotoon, maar wel *anti-monotoon*: als we P verzwakken wordt $\neg P$ juist sterker! Dit is al verwoord in de regel “contrapositie”, die we hier voor de volledigheid herhalen, zij het in een herschreven vorm.

contrapositie:

$$[P \Rightarrow Q \equiv \neg P \Leftarrow \neg Q]$$

* * *

Bij het programmeren zijn we vaak geïnteresseerd in het bepalen van een zogenaamde *zwakste preconditione*. Dit komt altijd neer, voor een of ander predikaat Q , op het bepalen van het zwakste van alle predikaten X die sterker zijn dan Q ; dat wil dus zeggen: het bepalen van het zwakste van alle predikaten X waarvoor geldt $[X \Rightarrow Q]$. Dit is eenvoudig: die zwakste X is Q zelf! Immers, er geldt $[Q \Rightarrow Q]$, dus (de linker) Q is sterker dan (de rechter) Q , en voor elk predikaat X waarvoor geldt $[X \Rightarrow Q]$ geldt ook dat Q zwakker is dan X .

Het *oplossen* van de vergelijking $[X \Rightarrow Q]$ is dus niet moeilijk; het probleem is vaak *het in deze vorm* brengen van een vergelijking: X links van een implicatie en rechts ervan iets waar X niet in voorkomt. Hier is, bijvoorbeeld, een patroon dat nogal eens voorkomt:

$$\begin{aligned} & X \wedge P \Rightarrow Q \\ = & \quad \{ \wedge/\Rightarrow\text{-associatie} \} \\ & X \Rightarrow (P \Rightarrow Q) \\ = & \quad \{ \text{implicatie} \} \\ & X \Rightarrow (\neg P \vee Q) \ , \end{aligned}$$

en dus is $\neg P \vee Q$ de zwakste oplossing van $[X \wedge P \Rightarrow Q]$.

Niet iedere vergelijking heeft een zwakste oplossing; het is zelfs niet zo dat iedere vergelijking überhaupt oplossingen heeft. Zie bijvoorbeeld opgave 34.

1.5 Rekenen in context

Een uitspraak van de vorm $[P \Rightarrow Q]$ kunnen we bewijzen door $P \Rightarrow Q$ uit *true* af te leiden. Als echter de manipulaties vooral *rechts* van de implicatie plaatsvinden en (de informatie in) P slechts af en toe in het betoog een rol speelt, dan kunnen we de berekening korter opschrijven door Q uit *true* af te leiden, *onder aanname van P* . We laten dus

al die voorvoegsels $P \Rightarrow$ weg en voeren de berekening uit *in de context van de aanname* P^4 . Daar waar P een rol speelt halen we P aan in de hint van de betreffende rekenstap. Vooral als P groot is sparen we zo een hoop schrijfwerk uit en wordt de berekening overzichtelijker.

Bij Logica & Verzamelingen werd een aanname in een “vlag” geplaatst en werd de context aangegeven door een wel heel duidelijke maar niet zo handige “vlaggestok”. Wij geven vlag en context aan met een hakenpaar $\llbracket \dots \triangleright \dots \rrbracket$; het openingshaakje \llbracket markeert het begin van een context, het sluihaakje \rrbracket markeert het einde van een context, en het symbool \triangleright scheidt de aanname(n) van de afleiding.

Hier is een voorbeeld, dat is ontleend aan Ontwerp van Algoritmen 1; laat P het volgende predikaat zijn –met x, y en n geheeltallige variabelen–:

$$P : \quad y = x^n \quad .$$

We zijn nu wel eens geïnteresseerd⁵ in een expressie E met de eigenschap: $[P \wedge 0 \leq n \Rightarrow P(n, y := n+1, E)]$. We bepalen nu zo’n expressie door (te proberen) deze eigenschap te bewijzen en aldus te bepalen welke eisen aan E worden gesteld. Onder aanname van $P \wedge 0 \leq n$ leiden we hiertoe af:

$$\begin{aligned} & \llbracket P \wedge 0 \leq n \\ & \triangleright \quad P(n, y := n+1, E) \\ & = \quad \{ \text{definitie } P \} \\ & \quad (y = x^n) (n, y := n+1, E) \\ & = \quad \{ \text{substitutie} \} \\ & \quad E = x^{n+1} \\ & = \quad \{ \text{algebra, met de aanname } 0 \leq n \} \\ & \quad E = x^n * x \\ & = \quad \{ \text{aanname } P \} \\ & \quad E = y * x \\ & = \quad \{ \bullet \text{ kies } E = y * x \} \\ & \quad \text{true} \\ & \rrbracket \quad . \end{aligned}$$

Aldus concluderen we dat de eigenschap geldt als we E gelijk kiezen aan $y * x$. Alleen in de voorlaatste stap van deze berekening is gebruik gemaakt van de aanname P . De eerste stap betreft alleen het vervangen van de naam P door de formule waar P voor staat: dat staat los van die aanname. De laatste stap, met hint “ \bullet kies $E = y * x$ ” is een *ontwerpbeslissing*: hier maken we een keuze voor de gezochte expressie E ; ter markering van zo’n stap gebruiken we een “vette punt” \bullet , die aangeeft dat er iets nieuws wordt ingevoerd.

We zien in dit voorbeeld ook dat we de onderdelen van een conjunctie gewoon “los” gebruiken: wat bij Logica & Verzamelingen “ \wedge -eliminatie” heet passen we hier stilzwijgend toe.

⁴Bij Logica & Verzamelingen heette dit “ \Rightarrow -introductie”.

⁵Waarom is hier niet van belang.

1.6 Toegift

In een oudere versie van dit stukje heb ik, geheel tegen mijn bedoeling, verzuimd de regel van de “uitsluitingen” bij de basisregels op te nemen. De studenten kennen deze regels van Logica & Verzamelingen als “uitgesloten derde”:

$$[P \vee \neg P \equiv \text{true}] ,$$

en als “tegenspraak”:

$$[P \wedge \neg P \equiv \text{false}] .$$

Als gevolg van deze omissie ondervonden zij moeilijkheden bij het maken van de opgaven: ze hadden de regel nodig. Pas later ontdekte ik mijn verzuim; toen ik de studenten hierop wees en aangaf dat deze regel er ook bij hoort, verschaftte J. Sterrenburg echter het volgende bewijs in termen van de eerdere regels – de opgave over idempotentie was al met succes opgelost –:

$$\begin{aligned} & P \vee \neg P \\ = & \quad \{ \text{implicatie} \} \\ & P \Rightarrow P \\ = & \quad \{ \text{idempotentie (opgave)} \} \\ & (P \Rightarrow P) \wedge (P \Rightarrow P) \\ = & \quad \{ \text{equivalentie} \} \\ & P \equiv P \\ = & \quad \{ \text{eenheidselement} \} \\ & \text{true} . \end{aligned}$$

Schitterend! De duale $[P \wedge \neg P \equiv \text{false}]$ volgt hier, met behulp van de Morgan, eenvoudig uit. Kortom, sommige studenten zijn hierin al vaardiger dan ik. Waarvan akte!

1.7 Opgaven

1. Bewijs alle afgeleide regels (in paragraaf 1.3) met gebruikmaking van alleen de basisregels.
2. Bewijs de verzwakkings- en monotonieregels uit paragraaf 1.4.
3. Bewijs: $[\text{true} \neq \text{false}]$.
4. Vereenvoudig: $P \wedge \neg P \Rightarrow R$, en ook: $P \vee \neg P \Rightarrow R$.
5. Vereenvoudig: $P \equiv \text{false}$.
6. Bewijs: $[(P \neq Q) \equiv \neg P \equiv Q]$.
7. Bewijs: \neq is commutatief en associatief.
8. Idem: \wedge distribueert over \neq .

9. Geef een tegenvoorbeeld waaruit blijkt dat \Rightarrow niet associatief is.
10. Idem: dat \wedge niet distribueert over \equiv .
11. Idem: dat \equiv niet distribueert over \vee .
12. Bewijs: $P \Rightarrow (Q \wedge R) = (P \Rightarrow Q) \wedge (P \Rightarrow R)$; waarom is dit nuttig?
13. Idem: $P \vee Q \Rightarrow R = (P \Rightarrow R) \wedge (Q \Rightarrow R)$; waarom is dit nuttig?
14. Bewijs: $(P \wedge R) \vee (Q \wedge \neg R) = (P \vee \neg R) \wedge (Q \vee R)$.
15. Idem: $P \equiv Q = (P \wedge Q) \vee (\neg P \wedge \neg Q)$.
16. Idem: $(P \Rightarrow Q) \wedge (Q \Rightarrow R)$ is sterker dan $P \Rightarrow R$.
17. Geef een voorbeeld van een predikaat P en expressies E en F zodanig dat $P(x:=E)(y:=F)$ en $P(y:=F)(x:=E)$ en $P(x, y:=E, F)$ alle *verschillend* zijn.
18. Geef een voorbeeld van een predikaat P en expressies E en F zodanig dat $P(x:=E)(y:=F)$ en $P(y:=F)(x:=E)$ en $P(x, y:=E, F)$ alle *gelijk* zijn.
19. Wat is het zwakste van alle predikaten X waarvoor $[X]$?
20. Wat is het *zwakste* van alle predikaten X die voldoen aan: $[P \Rightarrow (X \Rightarrow Q)]$?
21. Bepaal de *sterkste* en de *zwakste* van alle predikaten X die voldoen aan: $[P \Rightarrow X \equiv \text{true}]$; evenzo voor: $[X \Rightarrow P \equiv \text{true}]$.
22. Idem, voor: $[X \Rightarrow P \equiv P]$, en voor: $[P \Rightarrow X \equiv P]$.
23. Bepaal, met a een geheel-tallige variabele, het zwakste van alle predikaten X waarvoor: $[0 \leq a \wedge X \Rightarrow 0 \leq a-1]$.
24. Beargumenteer dat voor geheel-tallige x en y geldt: $[x < y+1 \equiv x \leq y]$. Dit noemen we eigenschap \heartsuit .
Bewijs vervolgens, voor geheel-tallige i en n en onder de aanname $0 \leq n$, dat: $[0 \leq i < n+1 \equiv 0 \leq i < n \vee i = n]$; geef aan waar de aanname $0 \leq n$ nodig is en waar eigenschap \heartsuit .
Idem, voor: $[0 \leq i < n+1 \equiv 0 = i \vee 0 \leq i-1 < n]$.
25. opgave 8 op pagina 12 uit het Boek; vereenvoudig de antwoorden.
26. Bereken en vereenvoudig: $(x^2 + y^2)(x, y := x+y, x-y)$
27. Vergelijk de uitkomsten van:
 $(x + a \cdot n < 10)(x, n := a \cdot n, n+1)$,
 $(x + a \cdot n < 10)(x := a \cdot n)(n := n+1)$, en:
 $(x + a \cdot n < 10)(n := n+1)(x := a \cdot n)$.
28. Bereken en vereenvoudig: $P(x, n := x + a \cdot n, n+1)$, waarbij P het predikaat $x = (\sum i: 0 \leq i < n: a \cdot i)$ is, en waarbij $0 \leq n$ gegeven is.

Voor de liefhebbers:

29. Beargumenteer dat het met alleen de basisregels onmogelijk is te bewijzen dat: $\text{true} \neq \text{false}$. In opgave 3 is wel bewezen: $[\text{true} \neq \text{false}]$. Leg (informeel) uit waarom hieruit redelijkerwijs toch volgt dat true en false verschillend zijn.
30. De bewering “ P is een tautologie” is waar of niet waar: dit is dus eigenlijk ook een propositie. Als we nu $[\cdot]$ als gewone operator in de calculus opnemen, kunnen we formules maken zoals $[[P]]$, $\neg[P]$ en $[P] \vee [Q]$. Bespreek het verschil in betekenis tussen $\neg[P]$ en $[\neg P]$, en tussen $[P]$ en $\neg[\neg P]$. Geef voorbeelden die deze verschillen illustreren.
31. Beargumenteer waarom $[[P]] = [P]$ een goede regel zou moeten zijn, in de met $[\cdot]$ uitgebreide calculus.
32. Beargumenteer: $P \wedge Q$ is een tautologie als en alleen als P en Q elk afzonderlijk tautologieën zijn. Geef dit in een formule met $[\cdot]$ weer.
33. Geef een tegenvoorbeeld waaruit blijkt dat $[P] \vee [Q]$ niet gelijk is aan $[P \vee Q]$. Is één van beide sterker dan de ander?
34. Voor welke predikaten P en Q heeft deze vergelijking (in X) oplossingen: $[X \vee P \Rightarrow Q]$? Is er dan ook een zwakste oplossing?

1.8 Quantoren: basisregels

We gebruiken de volgende regels voor universele en existentiële quantificatie en voor sommatie. Hierbij zijn P , Q en R predikaten en is F een integer expressie; in P , Q , R en F kan variabele i (vrij) voorkomen.

leeg domein:

$$\begin{aligned}(\forall i: \text{false}: P) &= \text{true} \\(\exists i: \text{false}: P) &= \text{false} \\(\Sigma i: \text{false}: F) &= 0\end{aligned}$$

eenpuntsdomein – E is een expressie waarin i niet voorkomt –:

$$\begin{aligned}(\forall i: i = E: P) &= P(i := E) \\(\exists i: i = E: P) &= P(i := E) \\(\Sigma i: i = E: F) &= F(i := E)\end{aligned}$$

domeinsplitsing:

$$\begin{aligned}(\forall i: P \vee Q: R) &= (\forall i: P: R) \wedge (\forall i: Q: R) \\(\exists i: P \vee Q: R) &= (\exists i: P: R) \vee (\exists i: Q: R) \\(\Sigma i: P \vee Q: F) &= (\Sigma i: P: F) + (\Sigma i: Q: F)\end{aligned}$$

35. Motiveer de regels voor leeg domein: waarom zijn ze zo gekozen?
36. Geef een (eenvoudig) tegenvoorbeeld dat laat zien dat de regel voor domeinsplitsing voor Σ niet altijd geldt.

37. Onder welke voorwaarde, op te leggen aan P en Q , geldt de regel voor domeinsplitsing voor Σ wel? Vat deze voorwaarde in een formule. Waarom is zo'n voorwaarde voor \forall en \exists niet nodig?
38. Bewijs, als gegeven is $0 \leq n$, deze regels voor termafplitsing:
 $(\forall i: 0 \leq i < n+1: P) = (\forall i: 0 \leq i < n: P) \wedge P(i:=n)$, en ook:
 $(\forall i: 0 \leq i < n+1: P) = P(i:=0) \wedge (\forall i: 0 \leq i < n: P(i:=i+1))$.
39. Pas beide vormen van termafplitsing toe op $(\forall i: 0 \leq i < n+1: a \cdot i \leq a \cdot n)$.
40. Beschouw de formule $Q \cdot n$, gedefinieerd door: $Q \cdot n = (\forall i, j: 0 \leq i < j < n: P \cdot i \cdot j)$.
- Voor welke (geheel-tallige) waarde(n) van n is het domein van de dummies in $Q \cdot n$ leeg?
 - Voor welke waarde(n) van n is het domein een éénpuntsdomein? Wat is dan de waarde van $Q \cdot n$?
 - Geef een berekening waarin, uitgaande van $Q \cdot (n+1)$, geval $0 = i$ wordt afgesplitst en waarin vervolgens, m.b.v. van dummytransformatie(s), alle domeinen weer ondergrens 0 krijgen. Aan welke voorwaarde moet n voldoen?
41. Idem, voor $Q \cdot n$ met: $Q \cdot n = (\forall i, j: 0 \leq i \leq j < n: P \cdot i \cdot j)$.

1.9 Opgavenselectie

De volgende opgaven uit de voorafgaande collectie moet men in ieder geval zelfstandig kunnen maken.

propositierekening

- Van opgave 1: nulelement en eenheidselement voor \Rightarrow , haasje over, idempotentie, complement(absorptie), absorptie, de Morgan voor \Rightarrow .
- opgave 3.
- opgave 4.
- opgave 6.
- opgave 9.
- opgave 12.
- opgave 13.
- opgave 14.
- opgave 24.

sterker en zwakker

- opgave 2.
- opgave 16.
- opgave 19.
- opgave 20.
- opgave 22.
- opgave 23.

substitutie

- opgave 17.
- opgave 18.
- opgave 25.
- opgave 26.
- opgave 27.
- opgave 28.

quantoren

Alle opgaven in deze rubriek.

Chapter 2

Skip, assignment en compositie

samenvatting: over Hoare triples

Uit een statement S en predikaten P en Q vormen we het Hoare-triple $\{P\}S\{Q\}$. De *operationele* betekenis van $\{P\}S\{Q\}$ is:

Uitvoering van S voert een (begin)toestand die aan P voldoet in eindig veel stappen over in een (eind)toestand die aan Q voldoet.

De *formele* betekenis van $\{P\}S\{Q\}$ wordt gegeven door de volgende

definities:

$\{P\} \text{skip} \{Q\}$ betekent: $[P \Rightarrow Q]$.

$\{P\} x := E \{Q\}$ betekent: $[P \Rightarrow Q(x := E)]$.

$\{P\} x, y := E, F \{Q\}$ betekent: $[P \Rightarrow Q(x, y := E, F)]$.

$\{P\} S; T \{R\}$ betekent: Er bestaat een predikaat Q zodanig dat zowel $\{P\}S\{Q\}$ als $\{Q\}T\{R\}$.

Het woord “betekent” betekent hier “is per definitie gelijkwaardig met”.

□

nota bene: Let op het gebruik van de rechte haken. Wat betekenen ze ook al weer?

□

* * *

We geven nog wat algemene eigenschappen. Deze hebben gebruikswaarde, dus men doet er goed aan ze te kennen.

De uitspraak $\{P\}S\{Q_0 \wedge Q_1\}$ is gelijkwaardig met $\{P\}S\{Q_0\}$ én $\{P\}S\{Q_1\}$ tesamen. Daarom kunnen we $\{P\}S\{Q_0 \wedge Q_1\}$ bewijzen door $\{P\}S\{Q_0\}$ en $\{P\}S\{Q_1\}$ afzonderlijk te bewijzen; aldus delen we een bewijsverplichting op in twee

eenvoudigere bewijsverplichtingen. Het betekent ook dat we eenvoudig een assertie aan een geannotéerd programma kunnen toevoegen: we hoeven dan alleen te bewijzen dat de nieuwe assertie een correcte postconditie is van de voorafgaande statement, onafhankelijk van zijn mede-asserties. Daarom schrijven we assertie $\{ Q_0 \wedge Q_1 \}$ ook wel zó op: $\{ Q_0 \} \{ Q_1 \}$, wat vooral handig is als Q_1 later is of wordt toegevoegd.

Als de preconditionie P zelf ook een conjunctie is, zeg $P_0 \wedge P_1$, kan de uitspraak $\{ P_0 \wedge P_1 \} S \{ Q_0 \wedge Q_1 \}$ worden gesplitst in $\{ P_0 \wedge P_1 \} S \{ Q_0 \}$ en $\{ P_0 \wedge P_1 \} S \{ Q_1 \}$. Kortom, ook bij het bewijzen van zo'n deeluitspraak mag de *gehele* preconditionie worden gebruikt.

opgaven

42. Wat is, voor vaste Q , het zwakste predikaat X dat voldoet aan $\{ X \} \text{skip} \{ Q \}$?
43. Boek, pagina 17, opgave 0.
44. Boek, pagina 17, opgave 1.
45. Wat is, voor vaste Q en E , het zwakste predikaat X dat voldoet aan $\{ X \} x := E \{ Q \}$?
46. Bewijs:
- $\{ x = n^2 \} x := x + 2 * n + 1 \{ x = (n+1)^2 \}$
 - $\{ 0 \leq n < 100 \} n := n + 1 \{ 0 \leq n \leq 100 \}$
47. Boek, pagina 20, opgave 0: (ii), (iii), (vi), (vii), (ix) en (xiii).
48. Bereken, voor elk van de volgende keuzen voor predikaat P , een expressie E die voldoet aan: $\{ P \wedge 0 \leq n \} x, n := E, n + 1 \{ P \}$.
- P is $x = 12 * n + 5$
 - P is $x = 2^n$
 - P is $x = n!$
 - P is $x = (\sum i : 0 \leq i < n : i)$
49. Functie f is gedefinieerd op de gehele getallen. Bereken een expressie E die voldoet aan:
- $\{ \text{true} \} x, n := E, 0 \{ x = (\sum i : 0 \leq i < n : f \cdot i) \}$
 - $\{ \text{true} \} x, n := E, 1 \{ x = (\sum i : 0 \leq i < n : f \cdot i) \}$
50. Variabele b is een boolean variabele. Predikaat P is gegeven door:
 $b \equiv (\exists i : 0 \leq i < n : f \cdot i = 0)$, waarbij f een gegeven functie op de gehele getallen is.
 Bepaal een (zo eenvoudig mogelijke) expressie E die voldoet aan:
- $\{ \text{true} \} b, n := E, 0 \{ P \}$
 - $\{ P \wedge 0 \leq n \} b, n := E, n + 1 \{ P \}$
 - $\{ P \wedge 0 \leq n \wedge \neg b \} b, n := E, n + 1 \{ P \}$
 - $\{ P \wedge 0 \leq n \wedge f \cdot n = 0 \} b, n := E, n + 1 \{ P \}$

51. Hoe kunnen we, voor gegeven statements S , T en predikaat R het zwakste predikaat X bepalen waarvoor $\{X\} S; T \{R\}$?
52. Boek, pagina 22, opgave 0.
53. Boek, pagina 23, opgave 2.
54. Predikaat Q_0 is gegeven als: $x = n^3$. Bewijs deze uitspraak:

$$\begin{array}{l} \{ Q_0 \} \{ Q_1 \} \{ Q_2 \} \\ x := x + y \\ ; y := y + z \\ ; z := z + 6 \\ ; n := n + 1 \\ \{ Q_0 \} \{ Q_1 \} \{ Q_2 \} \end{array}$$

Doe dit door geschikte predikaten Q_1 en Q_2 af te leiden en door bij elke puntkomma tussenpredikaten te bepalen. Formuleer *alle* bewijsverplichtingen, ook de triviale.

opmerking: Dit is de body van het op college behandelde programma voor de tabel van derdemachten. Raadpleeg uw aantekeningen hierover alleen ter controle of als het niet lukt, maar probeer de opgave eerst zelf!

55. Gegeven is een (constante) functie f en variabelen x , y , en z . Predikaat P is gedefinieerd als: $f \cdot x \neq f \cdot y$. Bepaal boolean expressies B en C zodanig dat de volgende drie uitspraken waar zijn: $\{P \wedge B\} x := z \{P\}$ en $\{P \wedge C\} y := z \{P\}$ en $[P \Rightarrow B \vee C]$.

Chapter 3

Specificatie-oefeningen

56. Variabelen x en y zijn geheeltallig. Geef een specificatie van een programmafragment S dat (de waarden van) x en y niet verandert en dat verder voldoet aan:
- a) S berekent de som van x en y .
 - b) S bepaalt of onder x en y een 0 voorkomt.
57. Gegeven zijn een positief natuurlijk getal N en een integer array $A[0..N)$. Verder zijn k en r integer variabelen en is b een boolean variabele. Formuleer predikaten die de volgende uitspraken weergeven:
- a) Alle elementen van A zijn positief.
 - b) De waarde van b geeft aan of alle elementen van A positief zijn.
 - c) Alle elementen van A zijn priemgetallen.
 - d) In A komt geen nul voor.
 - e) In A komen ten minste 2 nullen voor.
 - f) Alle elementen van A zijn gelijk.
 - g) r is het maximum van A .
 - h) Op plaats k staat (een voorkomen van) het maximum van A .
 - i) b geeft aan of het maximum van A precies één keer in A voorkomt.
 - j) r is de som van de even elementen van A .
 - k) r is de som van de elementen van A met een even index.
58. Gegeven zijn een integer array $A[0..1000)$ en een integer variabele x . Geef een specificatie van een programmafragment S dat deze variabelen niet verandert en dat verder voldoet aan:
- a) S berekent de som van alle positieve elementen van array A .
 - b) S bepaalt of x in array A voorkomt.
 - c) S bepaalt of de som van (de elementen van) A groter is dan 371.
 - d) S bepaalt hoe vaak x in array A voorkomt.
59. Specificeer een programmafragment S dat

- a) voor gegeven integer N en integer array $A[0..N)$ het maximum van array A berekent.
 - b) vaststelt of een gegeven integer array $A[0..N)$ constant is.
 - c) voor gegeven integer N en integer array $A[0..N)$ berekent hoe vaak het maximum van A in A voorkomt.
60. Stel een specificatie op voor een programma ter berekening van de som van die elementen van een gegeven integer array die een kwadraat zijn.
61. Stel een specificatie op voor een programma ter berekening van de som van die elementen van een gegeven integer array waarvan de indices een kwadraat zijn.

Chapter 4

Selectie

Voor alle volgende opgaven over programmaatjes geldt de spelregel: formuleer altijd *eerst* alle bewijsverplichtingen.

62. (tail distribution)(prima thuiswerk!) Bewijs dat:

```
{ P }
if B0 → S0
[] B1 → S1
fi
; T
{ R }
```

even waar is als:

```
{ P }
if B0 → S0 ; T
[] B1 → S1 ; T
fi
{ R }
```

63. Boek, pagina 27, opgave 0.

64. Boek, pagina 27, opgave 3.

65. Construeer een “gelijkwaardig” – d.w.z. met hetzelfde effect – programmafragment, zonder gebruik van de operatoren `max` en/of `min` voor:

```
{ 0 ≤ y ≤ x }
y := (y+a) max 0
; x := x max y
{ 0 ≤ y ≤ x }
```

66. In deze opgave is m een (integer) variabele en zijn A , B , C en D constanten.

(a) Construeer een specificatie voor een programmafragment S dat de waarde van m vervangt door het maximum van m en A .

- (b) Construeer zo'n programmafragment, zonder gebruik van de operatoren `max` en/of `min`.
- (c) Construeer een programmafragment, zonder gebruik van de operatoren `max` en/of `min`, dat aan m het maximum toekent van A , B , C en D .

67. Bereken, voor elk van de volgende selecties, een zo zwak mogelijke preconditionie P :

- (a) $\{ P \}$
`if $x \leq y \rightarrow m := x \parallel y \leq x \rightarrow m := y$ fi`
 $\{ m = x \min y \}$
- (b) $\{ P \}$
`if $x \leq y \rightarrow m := x \parallel y \leq x \rightarrow m := y$ fi`
 $\{ m = x \max y \}$
- (c) $\{ P \}$
`if $x < y \rightarrow \text{skip} \parallel y < x \rightarrow \text{skip}$ fi`
 $\{ \text{true} \}$
- (d) $\{ P \}$
`if $\text{true} \rightarrow x := 0 \parallel \text{true} \rightarrow x := 1$ fi`
 $\{ x = 1 \}$

68. Bereken "passende" boolean expressies B en C voor:

$$\{ x \geq 1 \wedge y \geq 1 \}$$
`if $B \rightarrow x, y := y, x$`
 `$\parallel C \rightarrow x := x - y$`
`fi`
 $\{ x \geq 1 \wedge y \geq 1 \}$

Chapter 5

Bewijs- en specificatie-oefeningen

69. Bewijs, netjes en waar mogelijk d.m.v. berekeningen:

- (a) $+$ distribueert over \max
- (b) \min distribueert over \max
- (c) \max distribueert over \max
- (d) $(-x) \max (-y) = -(x \min y)$
- (e) $[u \leq x \min y \equiv u \leq x \wedge u \leq y]$
- (f) $[(\max i :: F) \leq u \equiv (\forall i :: F \leq u)]$

70. Gegeven: $P \cdot x$ is een predikaat waarin y niet voorkomt, en $Q \cdot y$ is een predikaat waarin x niet voorkomt. Bewijs de gelijkwaardigheid van:

$$(\forall x : P \cdot x : (\forall y : Q \cdot y : R \cdot x \cdot y)) \text{ en: } (\forall y : Q \cdot y : (\forall x : P \cdot x : R \cdot x \cdot y)) .$$

(Hint: doe het met vlagvertoon, zoals in Logica, deel B.)

71. (instantiatie, calculatieneel) Bewijs, door te rekenen:

$$[(\forall i : P \cdot i : Q \cdot i) \wedge P \cdot x \Rightarrow Q \cdot x] \text{ en: } [P \cdot x \wedge Q \cdot x \Rightarrow (\exists i : P \cdot i : Q \cdot i)] .$$

72. Gegeven is een integer functie f , op de natuurlijke getallen.

(a) Geef in woorden weer wat deze formule uitdrukt:

$$(\forall j : 0 \leq j : (\forall i : 0 \leq i < j : f \cdot i \leq f \cdot j)) .$$

(b) Bewijs dat die formule gelijkwaardig is met deze:

$$(\forall j : 0 < j : f \cdot (j-1) \leq f \cdot j) .$$

73. Gegeven zijn een integer functie f op de natuurlijke getallen, een natuurlijke constante N , en variabelen a , b en p . Gegeven is dat alle functiewaarden van f verschillend zijn. P_0 en P_1 zijn predikaten, als volgt:

$$P_0 : a \leq p \leq b$$

$$P_1 : (\forall i : 0 \leq i \leq N : f \cdot i \leq f \cdot p)$$

- (a) Druk het gegeven dat alle waarden van f verschillend zijn in een formule uit.
- (b) Geef in woorden weer wat P_0 en P_1 samen uitdrukken.

(c) Bewijs de volgende uitspraak:

$$\begin{array}{l} \{ P_0 \wedge P_1 \wedge a \neq b \} \\ \text{if } f \cdot a < f \cdot b \rightarrow a := a + 1 \\ \square f \cdot b < f \cdot a \rightarrow b := b - 1 \\ \text{fi} \\ \{ P_0 \wedge P_1 \} \end{array}$$

(d) Wat kan men concluderen uit $P_0 \wedge P_1 \wedge a = b$?

Chapter 6

Div en mod

74. Gegeven zijn gehele getallen A en B waarvoor geldt $0 < B$. We beschouwen deze vergelijking, in onbekenden q, r :

$$q, r: A = q * B + r \wedge 0 \leq r < B .$$

Bewijs nu:

- (a) Voor alle A en B heeft deze vergelijking (ten minste) een oplossing.
- (b) Voor alle A en B heeft deze vergelijking ten hoogste één oplossing, d.w.z.: de oplossing is uniek.

75. Voor geheel getal A en positief getal B zijn $A \operatorname{div} B$ en $A \operatorname{mod} B$ gedefinieerd als de (unieke) oplossing van de vergelijking:

$$q, r: A = q * B + r \wedge 0 \leq r < B .$$

Bewijs de volgende eigenschappen van div en mod , met behulp van deze definitie; hierin is $0 < B$ en zijn x, y en k gehele getallen:

- (a) $[0 \leq x < B \equiv (x \operatorname{mod} B) = x]$
- (b) $[0 \leq x < B \equiv (x \operatorname{div} B) = 0]$
- (c) $[0 \leq x \equiv 0 \leq x \operatorname{div} B]$
- (d) $(x+B) \operatorname{mod} B = x \operatorname{mod} B$
- (e) $(x+B) \operatorname{div} B = (x \operatorname{div} B) + 1$
- (f) $(x+k*B) \operatorname{mod} B = x \operatorname{mod} B$
- (g) $(x+k*B) \operatorname{div} B = (x \operatorname{div} B) + k$
- (h) $(x \operatorname{mod} B) \operatorname{mod} B = x \operatorname{mod} B$
- (i) $(x+y) \operatorname{mod} B = (x \operatorname{mod} B + y \operatorname{mod} B) \operatorname{mod} B$
- (j) $[x \operatorname{mod} B = 0 \equiv x \operatorname{div} B = x/B]$
- (k) $[1 \leq x \equiv x \operatorname{div} B < x]$

Chapter 7

Repetitie

Ook hier geldt de spelregel: formuleer altijd *eerst* alle bewijsverplichtingen.

eindiging informeel

76. Bewijs en bespreek daarbij informeel ook de eindiging:

$$\begin{array}{l} \{ 0 \leq A \wedge 0 < B \} \\ q, r := 0, A \\ ; \{ A = q * B + r \wedge 0 \leq r \} \\ \text{do } B \leq r \rightarrow q := q + 1 \\ \quad ; r := r - B \\ \text{od} \\ \{ A = q * B + r \wedge 0 \leq r < B \} \end{array}$$

77. Boek, pagina 37, opgave 0.

78. Boek, pagina 37, opgave 1.

79. Boek, pagina 37, opgave 2.

80. Beschouw een “schaakbord” van afmeting $M \times N$ eenheidsvelden, waarbij zowel M als N oneven zijn. Geef een eenvoudig argument waarom dit schaakbord niet netjes met 2×1 tegels kan worden belegd. Wat is hier de invariante eigenschap?

81. We beschouwen een $M \times N$ “schaakbord”, waarbij zowel M als N oneven zijn. Van het schaakbord is (precies) één van de hoekvelden verwijderd. Is het nu mogelijk het zo verkregen bord netjes met 2×1 tegels te beleggen? Beargumenteer!

82. We beschouwen een $M \times N$ “schaakbord”, waarbij zowel M als N even zijn. Van het schaakbord zijn twee diagonaal tegenover elkaar gelegen hoekvelden verwijderd. Is het nu mogelijk het zo verkregen bord netjes met 2×1 tegels te beleggen? Beargumenteer!

83. Bewijs, door een invariant te kiezen en vervolgens alle bewijsverplichtingen te formuleren en na te komen:

$$\begin{array}{l}
\{ 0 \leq N \} \\
x, r := 1, 0 \\
; \{ \dots ? \dots \} \\
\text{do } x < N \rightarrow x := 2 * x \\
\quad ; r := r + 1 \\
\text{od} \\
\{ N \leq 2^r \wedge (\forall i: 0 \leq i < r: 2^i < N) \}
\end{array}$$

Formuleer in gewoon Nederlands wat de postconditie uitdrukt.

84. Bepaal geschikte expressies E en F en bewijs dan:

$$\begin{array}{l}
\{ A < 0 \wedge 0 < B \wedge -B \leq A \} \\
q, r := E, F \\
; \{ A = q * B + r \wedge 0 \leq r \} \\
\text{do } B \leq r \rightarrow q := q + 1 \\
\quad ; r := r - B \\
\text{od} \\
\{ A = q * B + r \wedge 0 \leq r < B \}
\end{array}$$

85. Boek, pagina 37, opgave 3.

86. Boek, pagina 37, opgave 4.

87. Bespreek informeel de eindiging van:

$$\begin{array}{l}
\{ x \geq 0 \wedge y \geq 0 \} \\
\text{do } y \neq 0 \rightarrow y := y - 1 \\
\quad \square x \neq 0 \rightarrow x, y := x - 1, y + 1 \\
\text{od} \\
\{ x = 0 \wedge y = 0 \}
\end{array}$$

88. We definiëren $ggd \cdot x \cdot y$ als het maximum van alle gemeenschappelijke delers van de positieve natuurlijke getallen x en y . De functie ggd heeft dan de volgende eigenschappen:

- (0) $ggd \cdot x \cdot y = x$, als $x = y$
- (1) $ggd \cdot x \cdot y = ggd \cdot y \cdot x$
- (2) $ggd \cdot x \cdot y = ggd \cdot (x - y) \cdot y$, als $x > y$
- (3) $ggd \cdot x \cdot y = ggd \cdot x \cdot (y - x)$, als $y > x$

(a) Bewijs deze eigenschappen.

(b) Bewijs, met P_0 , P_1 en P_2 net als bij het op het college behandelde programma voor ggd :

$$\begin{array}{l}
\{ X \geq 1 \wedge Y \geq 1 \} \\
x, y := X, Y \\
; \{ P_0 \wedge P_1 \wedge P_2 \} \\
\text{do } x > y \rightarrow x := x - y \\
\quad \square y > x \rightarrow x, y := y, x \\
\text{od} \\
\{ x = ggd \cdot X \cdot Y \}
\end{array}$$

Bespreek informeel ook de eindiging.

89. We beschouwen dit programmafragment, met W en Z constanten:

```

{  $W \geq 0 \wedge Z \geq 0 \wedge W+Z \geq 1$  }
 $w, z := W, Z$ 
; {  $\dots ? \dots$  }
do  $w \geq 2$            $\rightarrow w, z := w-2, z+1$ 
   $\square w \geq 1 \wedge z \geq 1$   $\rightarrow z := z-1$ 
   $\square z \geq 2$            $\rightarrow z := z-1$ 
od
{  $\{w, z\} = \{0, 1\}$  }

```

- Bewijs de correctheid van de postconditie; bespreek daarbij informeel ook de eindiging.
- Bewijs dat $w = 0 \equiv$ “ W is even” ook een correcte postconditie van dit programmafragment is. Formuleer hiertoe alle *extra* bewijsverplichtingen.
- Waar doet deze opgave aan denken?

90. Boek, pagina 50, opgave 4.

91. Bewijs:

```

{ true }
do  $a > b \rightarrow a, b := b, a$ 
   $\square b > c \rightarrow b, c := c, b$ 
   $\square c > d \rightarrow c, d := d, c$ 
od
{  $a \leq b \leq c \leq d$  }

```

Bespreek informeel ook de eindiging.

eindiging formeel

Vanaf hier dient eindiging *formeel* te worden bewezen, dat wil zeggen: met behulp van een *variante functie* en de bewijsregels hiervoor.

92. Beschouw het programma:

```

{  $N \geq 0$  }
 $x := 1 ; n := 0$ 
; do  $n \neq N \rightarrow x := (n+1) * x ; n := n+1$ 
  od
{  $n = N$  } {  $x = N!$  }

```

- Bewijs dat $n = N$ een correcte postconditie is, op de volgende manier:
 - geef eerst een volledig geannoteerd programma;
 - som alle bewijsverplichtingen op;
 - geef de bijbehorende bewijzen.

Hint: de asserties hoeven alleen over n te gaan.


```
{ A ≥ 0 }
q, r := 0, A+1
; do r ≥ 4 → x, y := r mod 4, r div 4
      ; q, r := q+y, x+y
od
; r := r-1
{ q = A div 3 ∧ r = A mod 3 }
```

100. (met dank aan Tom Verhoeff)

- (a) Is het mogelijk een 30×30 “schaakbord” netjes te beleggen met 3×4 tegels?
- (b) Is het mogelijk een 30×30 “schaakbord” netjes te beleggen met 1×4 tegels?
- (c) Is er een (logisch) verband tussen deze twee vragen?

Chapter 8

Het stappenplan

1. Stel een formele specificatie op, van de vorm:
[[“variabelendeclaratie” \triangleright { *Pre* } *S* { *Post* }]].
2. Kies een oplossingsstrategie. Als in de oplossing een repetitie wordt gebruikt, kies dan een mogelijke invariant *P*. Hiertoe maak je bijvoorbeeld gebruik van “constante door variabele vervangen” of van “weglaten van een conjunct”.
3. Ga na of met de voorgestelde invariant het doel kan worden bereikt, d.w.z. bepaal een boolean expressie *B* zodanig dat $[P \wedge \neg B \Rightarrow Post]$. Kies dan die *B* als guard.
4. Ontwerp een statement *Init* zó dat $\{ Pre \} Init \{ P \}$.
5. Kies een variante functie en een statement die de variante functie verlaagt; ontwerp dan een repetitiebody die bovendien *P* invariant laat.
6. Geef (voor zover nog nodig) het eindigingsbewijs.
7. Schrijf tenslotte het gehele programma op.

Chapter 9

Programmeeropgaven

9.0 Basistechnieken

Volg steeds het stappenplan. Eén van beide technieken, “weglaten van een conjunct” en “constante door variabele vervangen”, is steeds toepasbaar voor het verkrijgen van een bruikbare invariant. Beslis zelf welke.

101. Construeer een programma ter berekening van de som van de elementen met even index in een gegeven integer array.
102. Construeer een programma ter berekening van de som van de positieve elementen van een gegeven integer array.
103. Construeer een programma ter berekening van het grootste natuurlijke getal dat niet groter is dan \sqrt{N} , voor gegeven natuurlijk getal N .
104. Construeer een programma dat, voor een gegeven boolean functie f met domein $[0..N)$, bepaalt of:
 - (a) f ergens in zijn domein de waarde `true` heeft;
 - (b) f ergens in zijn domein de waarde `false` heeft;
 - (c) f overal in zijn domein de waarde `true` heeft;
 - (d) f overal in zijn domein de waarde `false` heeft.

In welke mate komen deze vier gevallen overeen?

105. Construeer een programma dat berekent of in een gegeven integer array het getal 7 voorkomt. Is er een overeenkomst met de vorige opgave?
106. Construeer een programma ter berekening van het aantal zevens in een gegeven integer array.
107. Boek, pagina 56, opgave 0. (Hint: herschrijf $x = {}^3\log N$ tot $N = 3^x$.)
108. (Linear Search) Gegeven is een boolean array f met domein $[0..N)$ en gegeven is dat f in ten minste één punt van zijn domein de waarde `true` heeft. Construeer een programma ter berekening van de *kleinste* index (in $[0..N)$) waar f de waarde `true` heeft.

109. Construeer een programma dat bepaalt of een gegeven integer X een kwadraat is. (Denk goed na over (de vorm van) de specificatie!)
110. Beschouw predikaat R gegeven door
 $R: r = (\#i: 0 \leq i < N: f \cdot i = f \cdot (N-1-i))$,
 voor gegeven array $f[0..N)$, met $N \geq 0$.
- (a) Kies $\varphi \cdot n = (\#i: 0 \leq i < n: f \cdot i = f \cdot (N-1-i))$ en ontwikkel het bijbehorende φ -programma met R als postconditie.
- (b) Ga na dat de keuze $\varphi \cdot n = (\#i: 0 \leq i < n: f \cdot i = f \cdot (n-1-i))$ in dit geval niet handig is.
111. Voor gegeven integer array $f[0..N)$ en constante X zijn we geïnteresseerd in een programma ter berekening van $(\sum i: 0 \leq i < N: a \cdot i * X^{N-1-i})$.
- (a) Kies $\varphi \cdot n = (\sum i: 0 \leq i < n: a \cdot i * X^{n-1-i})$ en ontwikkel het bijbehorende φ -programma (bekend als Horner's schema).
- (b) Kies $\varphi \cdot n = (\sum i: 0 \leq i < n: a \cdot i * X^{N-1-i})$ en ontwikkel het bijbehorende φ -programma. Ga na dat het hierbij zinvol is een extra invariant $s = X^{N-n}$ te gebruiken, en doe dat ook.
112. Boek, pagina 62, opgave 3.
113. Boek, pagina 62, opgave 5.
114. Construeer een programma dat bepaalt of een gegeven integer array *constant* is. (Wat betekent dat?)

9.1 Invariant versterken

115. De rij $F[0..\infty)$ van (zogenaamde) Fibonacci getallen is als volgt recursief gedefinieerd, voor alle natuurlijke i :
 $F_0 = 0$, $F_1 = 1$, en $F_{i+2} = F_i + F_{i+1}$.
 Construeer een programma ter berekening van F_N , voor een gegeven constante N , $0 \leq N$.
116. Construeer een programma, met lineaire rekentijd, ter berekening van $(\sum i: 0 \leq i < N: X^i)$, voor gegeven constanten X en N , $0 \leq N$.
117. In de volgende deelopgaven zijn N en f constanten, met $0 \leq N$ en $f[0..N)$ een integer array; steeds wordt alleen een postconditie gegeven.
- (a) $r = (\sum i, j: 0 \leq i < j < N: f \cdot i + f \cdot j)$
- (b) $r = (\sum i, j: 0 \leq i < j < N: f \cdot i * f \cdot j)$
- (c) $r = (\#i, j: 0 \leq i < j < N: f \cdot i = 0 \wedge f \cdot j = 1)$
- (d) $r = (\min i, j: 0 \leq i \leq j < N: f \cdot i - f \cdot j)$
- (e) $r = (\#j: 0 \leq j < N: (\sum i: 0 \leq i < j: f \cdot i) \leq f \cdot j)$
- (f) $r = (\#j: 0 \leq j < N: (\forall i: 0 \leq i < j: f \cdot i \leq f \cdot j))$

$$(g) \quad r = (\Sigma j: 0 \leq j < N: (\#i: 0 \leq i < j: f \cdot i \leq f \cdot j))$$

Bij (g) lukt het niet een bruikbare versterking van de invariant te kiezen: waarom niet? Wat is het verschil met (f)? (Opgave (g) hoeft hier niet te worden afgemaakt; zij komt terug als opgave 128 in de volgende paragraaf.)

118. Gegeven zijn N en f als in de vorige opgave. Bovendien is h een gegeven functie van type $\text{Int} \rightarrow \{-3, 7\}$; er geldt dus $h \cdot x = -3 \vee h \cdot x = 7$, voor alle gehele x .

$$(a) \quad r = (\#i, j: 0 \leq i < j < N: h \cdot (f \cdot i) = h \cdot (f \cdot j))$$

$$(b) \quad r = (\#i, j: 0 \leq i \leq j < N: h \cdot (f \cdot i) = h \cdot (f \cdot j))$$

Waarin verschilt dit met (a)?

$$(c) \quad r = (\Sigma j: 0 \leq j < N: (\#i: 0 \leq i < j: h \cdot (f \cdot i) = h \cdot (f \cdot j)))$$

Waarin verschilt dit met (a)?

119. Boek, pagina 71, opgave 1. Hint: er zijn twee versterkingen nodig.

120. Construeer een programma dat vaststelt of een gegeven array *ascending* –ieder element is *ten hoogste* alle opvolgers– is.

121. Construeer een programma dat de *lengte* berekent van een langste segment dat louter nullen bevat, in een gegeven array.

122. Construeer een programma dat de *lengte* berekent van een langste constante segment, in een gegeven array.

123. Boek, pagina 71, opgave 2.

124. Boek, pagina 71, opgave 3.

125. Boek, pagina 71, opgave 4.

126. (voor de liefhebbers) Boek, pagina 71, opgave 5.

9.2 Geneste repetities

127. Construeer een programma dat vaststelt of van een gegeven array alle elementen verschillend zijn.

128. Opgave 117(g) uit de vorige paragraaf.

129. Construeer een programma ter berekening van al die elementen uit een gegeven integer array die een kwadraat zijn.

9.3 Staartinvarianten

130. Boek, pagina 78, opgave 1.

131. De rij $F[0..∞)$ van Fibonacci getallen is als volgt recursief gedefinieerd, voor alle natuurlijke i :

$$F_0 = 0 \text{ , } F_1 = 1 \text{ , en } F_{i+2} = F_i + F_{i+1} .$$

We definiëren nu een functie G , als volgt, voor natuurlijke a, b, i :

$$G \cdot a \cdot b \cdot i = a * F_i + b * F_{i+1} .$$

Leid een recursieve definitie voor G af en gebruik die voor de constructie van een programma ter berekening van F_N , voor een gegeven N , $0 \leq N$.

132. Boek, pagina 79, opgave 3.

133. Boek, pagina 79, opgave 4.

134. Gegeven zijn een constante N , $0 \leq N$, en een integer array $f[0..N)$. Functie G is als volgt gedefinieerd, voor natuurlijke m, n waarvoor bovendien geldt $m+n \leq N$:

$$G \cdot m \cdot n = (\Sigma i : 0 \leq i < n : 2^i * f \cdot (i+m)) .$$

(a) Leid recurrente betrekkingen voor G af; doe dit zó dat er geen machtsverheffen in voorkomt.

(b) Construeer hiermee een programma ter berekening van:

$$(\Sigma i : 0 \leq i < N : 2^i * f \cdot i) .$$

135. Als de vorige opgave, maar met functie G gedefinieerd door:

$$G \cdot n = (\Sigma i : n \leq i < N : 2^{i-n} * f \cdot i) .$$

136. Boek, pagina 62, opgave 5.

137. Construeer een programma voor het berekenen van het aantal delers van een gegeven positief natuurlijk getal. Stel eerst een specificatie op.

138. (voor de liefhebbers) Boek, pagina 80, opgave 5.

9.4 Gemengde opgaven

In onderstaande opgaven dient men zelf te beoordelen welke techniek geschikt is voor het verkrijgen van een oplossing.

139. Boek, pagina 81, opgave 0.

140. Boek, pagina 81, opgave 1.

141. Boek, pagina 81, opgave 2.

142. Boek, pagina 82, opgave 4.

143. Boek, pagina 82, opgave 6.

144. Boek, pagina 82, opgave 7.

145. Construeer een programma voor de berekening van de maximale lengte der segmenten, van een gegeven array, waarvan alle elementen verschillend zijn.

Appendix: spelregels deeltentamen

In de tentamenperiode in januari, na de vijfde week van het trimester, wordt een deeltentamen afgenomen. Deelname hieraan is niet verplicht.

Het cijfer behaald voor dit deeltentamen, kan, mits het ten minste een 6 – zes – is, in de plaats worden gesteld voor het resultaat van de eerste opgave van de reguliere tentamens in maart en in mei, in *hetzelfde* studiejaar waarin dat cijfer voor het deeltentamen is behaald. Na afloop van het studiejaar vervalt dit recht.

De volgende *kennis* en *vaardigheden* worden in het deeltentamen getoetst.

0. Voor een gegeven (gedeeltelijk) geannoteerd programma *alle* bewijsverplichtingen formuleren. De belangrijke (GCL-)constructies zijn: `skip`, assignments, sequentiële compositie, selectie, repetitie.
1. Voor incomplete maar reeds geannoteerde programmafragmenten passende expressies uitrekenen en hierbij het correctheidsbewijs voltooien.
2. Uit een informele probleembeschrijving een formele specificatie opstellen.
3. Rekenen met predikaten en gequantificeerde formules met één dummy; in het bijzonder: de regels voor leeg domein, eenpuntsdomein, domein- en termafsplittingsen, en de regel voor distributie van een operator over een quantor.
4. De techniek constante door variabele vervangen, het standaard φ -schema, eenvoudig invariant versterken. Hiermee eenvoudige programmeeropgaven correct oplossen en documenteren.
5. Eenvoudige recurrente betrekkingen afleiden en toepassen in het standaard φ -schema.
6. Van eenvoudige repetities de eindiging bewijzen.

Tip: Ken ook het “Stappenplan”!

Appendix: Tentameneisen

Hieronder wordt opgesomd welke *vaardigheden* nodig zijn voor het succesvol beoefenen van dit vak; het draait hier vooral om *kunde*, maar natuurlijk vereist die kunde wel enige *kennis*: om bewijsverplichtingen te kunnen formuleren moet men de bewijsregels kennen, om met formules te kunnen rekenen moet men de calculus kennen, en om invarianten te kunnen formuleren moet men de relevante technieken kennen.

0. Voor een gegeven (gedeeltelijk) geannoteerd programma *alle* bewijsverplichtingen formuleren. De gebruikte taal is GCL, de belangrijke constructies zijn: `skip`, assignments, sequentiële compositie, selectie, iteratie en binnenblokken.
1. Uit een informele probleembeschrijving een formele specificatie opstellen.
2. Rekenen met predikaten en gequantificeerde expressies; in het bijzonder: termafspelingen en dummytransformaties.
3. Eenvoudige recurrente betrekkingen afleiden, generaliseren en staartrecursief maken.
4. Technieken voor het verkrijgen van invarianten beheersen: constante door variabele vervangen, conjunct weglaten, staartinvariant opstellen, invariant versterken.
5. Aandacht besteden aan het goed-gedefinieerd zijn van expressies. In de praktijk is dit vooral van belang bij:
 - gebruik van `div` en `mod`: delers moeten positief zijn.
 - array-indices: deze moeten binnen de arraygrenzen liggen.
6. Uitgaande van een specificatie systematisch een programma construeren, waarvan de correctheid duidelijk blijkt. (Eenvoudige) programma's met geneste repetities kunnen construeren.
7. Eindigingsbewijzen leveren.
8. De volgende standaardalgoritmen kennen en kunnen toepassen: Linear Search, Bounded Linear Search, Binary Search.

Bij het tentamen worden in ieder geval 0 en 6 expliciet getoetst. De onder 6 genoemde *correctheid* vereist dat men 4 en 5 beheerst: hierin te kort schieten kost punten; het leveren van eindigingsbewijzen valt hier ook onder, maar is voor alle duidelijkheid nog eens apart vermeld. De frase *systematisch... construeren* geeft aan dat programma's niet uit de mouw worden geschud maar worden afgeleid; hiertoe dienen 2, 3 en 4. **Tip:** Ken het "Stappenplan"!

Een "oplossing" (van een programmeeropgave) die slechts een programma is, voldoet niet aan de eisen en wordt dan ook niet gehonoreerd. Een substantiële aanzet tot een oplossing, die wel aan de eisen voldoet maar zonder programmatekst, wordt gedeeltelijk gehonoreerd.

Verwijzingen naar standaardschema's zijn toegestaan, maar dat moet wel expliciet zijn. Voorbeelden: de veel voorkomende invariant $0 \leq n \leq N$ uit het standaard φ -schema moet wel worden vermeld; bij initialisatie en invariantie vermeldde men dan: "standaard". Bij

eindiging volgens het φ -schema vermeldde men wel de variante functie en bij eindigingsbewijs vermeldde men: “standaard”. **Nota Bene:** “standaard” is geen toverwoord; misbruik wordt bestraft!

* * *

Tenslotte: Succesvol programmeren vereist, naast beheersing van de genoemde vaardigheden, een nette en *nauwgezette* stijl van werken: elk detail telt. Een enkele detailfout in een overigens correcte en verzorgde oplossing wordt in de regel door de vingers gezien, maar een (al te gortige) opeenstapeling van slordigheden leidt tot afkeuring van de gehele oplossing, zelfs als hieruit (enige) beheersing van de technieken blijkt. Bovendien: slordigheid leidt vrijwel steeds (en volkomen onnodig) tot incorrecte oplossingen.

Nauwgezetheid is een noodzakelijke (maar bij lange na niet voldoende) voorwaarde voor succesvol programmeerwerk: Hoe geavanceerd moderne tools en hoe snel moderne computers ook mogen zijn: het blijven machines!

Oude Tentamens

Tot besluit geven we hier een selectie van in de afgelopen jaren gebruikte tentamenopgaven. Het hoeft geen betoeg dat ook deze uitstekend oefenmateriaal vormen.

TECHNISCHE UNIVERSITEIT EINDHOVEN

Faculteit der Wiskunde en Informatica

Deeltentamen Ontwerp van Algoritmen 1 (2IA10), woensdag 21 januari 2004, 9.00–12.00 uur.

Let op: Noteer op het eerste vel de naam van uw instructeur. U mag geen boeken, eigen aantekeningen of calculators gebruiken. Draag zorg voor net en overzichtelijk werk.

1. In onderstaand programmafragment is x een integer variabele. Bewijs de navolgende bewering; formuleer hiertoe alle bewijsverplichtingen:

```

{ 1 ≤ x }
x := x + 1
; if 2 ≤ x → x := x + 1
  [] x ≤ 0 → x := x - 1
  [] x = 2 → skip
fi
{ 2 ≤ x }

```

2. Gegeven is een integer array $f[0..N]$, met $0 \leq N$. Van array f is gegeven dat alle elementen nul of één zijn, dat wil zeggen: $(\forall i: 0 \leq i < N: 0 \leq f \cdot i < 2)$.

Bewijs de correctheid van het volgende programma, dat berekent of in array f méér enen dan niet-enen voorkomen:

```

|| con N : lnt ; f[0..N] : array of lnt ; var n, r : lnt ; b : Bool
▷ { pre: 0 ≤ N ∧ (∀ i : 0 ≤ i < N : 0 ≤ f · i < 2) }
  n, r := 0, 0
; do n ≠ N → if f · n = 0 → skip
              [] f · n = 1 → r := r + 1
              fi
              ; n := n + 1
od
; b := 2 * r > N
{ post R : b ≡ (# i : 0 ≤ i < N : f · i = 1) > (# i : 0 ≤ i < N : f · i ≠ 1) }
||

```

Formuleer hiertoe éérs alle bewijsverplichtingen, en geef vervolgens (beknopte) bewijzen. Bespreek kort en informeel waarom de repetitie eindigt.

3. Gegeven is een integer array $f[0..N]$, met $0 \leq N$. We zijn geïnteresseerd in een programma voor het berekenen van de maximale waarde van $sum \cdot p \cdot N$, voor alle $p: 0 \leq p \leq N$. Hierbij is de functie sum gedefinieerd door:

$$sum \cdot p \cdot q = (\sum i: p \leq i < q: f \cdot i) \quad , \text{ voor alle } p, q: 0 \leq p \leq q \leq N \quad .$$

- a) Stel voor dit probleem een formele specificatie op.
 b) Ontwerp (en documenteer) een programma dat aan deze specificatie voldoet.
 Tip: Vervang beide voorkomens van N door een variabele.

TECHNISCHE UNIVERSITEIT EINDHOVEN
 Faculteit der Wiskunde en Informatica

Deeltentamen Ontwerp van Algoritmen 1 (2IA10), zaterdag 15 januari 2005,
 9.00 – 12.00 uur.

Let op: Noteer op het eerste vel de naam van uw instructeur. U mag geen boeken, eigen aantekeningen of calculators gebruiken. Draag zorg voor net en overzichtelijk werk.

1. Stel een formele specificatie op voor: een programma ter berekening van, voor een gegeven *niet-leege* integer array, een index in het domein van dat array waar het array zijn maximale waarde heeft.
2. Gegeven is een integer array $f[0..N]$, met $0 \leq N$. We gebruiken een hulpfunctie mx , die is gedefinieerd door:

$$mx \cdot q = (\max i : 0 \leq i < q : f \cdot i) \quad , \text{ voor alle } q : 0 \leq q \leq N \quad .$$

Bewijs de correctheid van het volgende programmafragment.

```

{ 0 ≤ N }
n, r, s := 0, 0, -∞
; do n ≠ N → if f · n ≤ s → skip
              [] f · n > s → s := f · n
              fi
              ; r := r + s
              ; n := n + 1
od
{ R: r = (Σ q: 0 < q ≤ N: mx · q) }
```

Formuleer hiertoe éérst alle bewijsverplichtingen, en geef vervolgens (beknopte) bewijzen. Bespreek kort waarom de repetitie eindigt.

3. Specificeer en construeer een programma ter berekening van de som van alle elementen die kleiner zijn dan 10, van een gegeven integer array.
-

waardering	opgave 1: 3 punten
	opgave 2: 4 punten
	opgave 3: 3 punten

TECHNISCHE UNIVERSITEIT EINDHOVEN

Faculteit der Wiskunde en Informatica

Deeltentamen Ontwerp van Algoritmen 1 (2IA14), dinsdag 17 januari 2006,
9.00 – 12.00 uur.

Let op: Noteer op het eerste vel de naam van uw instructeur. U mag geen boeken, eigen aantekeningen of calculators gebruiken. Draag zorg voor net en overzichtelijk werk.

1. Stel een formele specificatie op voor: een programma ter berekening van, voor een gegeven *niet-leege* integer array, het aantal posities in het domein van dat array waar het array zijn maximale waarde heeft.

2. Bewijs de correctheid van het volgende programmafragment:

```

{ 0 ≤ A ∧ 0 < B }
q, r := 0, A
; do B ≤ r → q := q + 1
      ; r := r - B
od
{ A = q * B + r ∧ 0 ≤ r < B }

```

Formuleer hiertoe alle bewijsverplichtingen en geef vervolgens (beknopte) bewijzen.

3. Gegeven is een constante N , $1 \leq N$, en een (constant) integer array $f[0..N)$. Construeer een programma ter bewerkstelling van de volgende postconditie – waarin dus twee variabelen, m en r , een waarde moeten krijgen –:

R: $m = (\max i: 0 \leq i < N: f \cdot i) \wedge r = (\sum i: 0 \leq i < N \wedge f \cdot i < m: f \cdot i)$.

waardering	opgave 1: 3 punten
	opgave 2: 4 punten
	opgave 3: 3 punten

TECHNISCHE UNIVERSITEIT EINDHOVEN

Faculteit Wiskunde en Informatica

Examen Ontwerp van Algoritmen 1 (2IA10), maandag 15 maart 2004, 9.00–12.00 uur.

Noteer op het eerste vel de naam van uw instructeur. U mag geen boeken, ander studiemateriaal of calculators gebruiken.

1. **Let op!** Voor deze opgave mag U, desgewenst, het resultaat van het (in januari) afgelegde deeltentamen opvoeren, mits dat resultaat ten minste 6 is. Dan schrijft U hier alleen op: “resultaat deeltentamen”. Als U dit niet doet en een oplossing voor de opgave inlevert blijft het resultaat van het deeltentamen buiten beschouwing.

We beschouwen het volgende programmafragment, met W en Z gegeven constanten:

```

{ 0 ≤ W ∧ 0 ≤ Z }
w, z := W, Z
; do w ≥ 2 → w, z := w - 2, z + 1
  [] z ≥ 1 → z := z - 1
od
{ post R: (w = 0 ∨ w = 1) ∧ (“W is even” ≡ w = 0) }
```

Bewijs de correctheid van de postconditie R ; formuleer hiertoe de benodigde invarianten, een variante functie, en *alle* bewijsverplichtingen. Tip: Eén van de benodigde invarianten is: “ W is even” \equiv “ w is even”.

2. Gegeven is een integer array $A[0..N]$, met $0 \leq N$. Ontwerp een programma voor het berekenen van de uitdrukking:

$$(\min i, j : 0 \leq i \leq j < N : A \cdot i - A \cdot j) .$$

3. De rij $F[0..\infty)$ van Fibonacci getallen is als volgt recursief gedefinieerd, voor alle natuurlijke i :

$$F_0 = 0 \quad , \quad F_1 = 1 \quad , \quad \text{en} \quad F_{i+2} = F_i + F_{i+1} .$$

We definiëren nu een functie G , als volgt, voor natuurlijke a, b, i :

$$G \cdot a \cdot b \cdot i = a * F_i + b * F_{i+1} .$$

Leid een recursieve definitie voor G af en gebruik die voor de constructie van een programma ter berekening van F_N , voor een gegeven N , $0 \leq N$.

waardering	opgave 1: 10 punten
	opgave 2: 10 punten
	opgave 3: 10 punten
	eindcijfer: totaalscore div 3 .

TECHNISCHE UNIVERSITEIT EINDHOVEN

Faculteit Wiskunde en Informatica

Examen Ontwerp van Algoritmen 1 (2IA10), woensdag 12 mei 2004, 9.00–12.00 uur.

Noteer op het eerste vel de naam van uw instructeur. U mag geen boeken, ander studiemateriaal of calculators gebruiken.

1. **Let op!** Voor deze opgave mag U, desgewenst, het resultaat van het (in januari) afgelegde deeltentamen opvoeren, mits dat resultaat ten minste 6 is. Dan schrijft U hier alleen op: “resultaat deeltentamen”. Als U dit niet doet en een oplossing voor de opgave inlevert blijft het resultaat van het deeltentamen buiten beschouwing.

We beschouwen het volgende programmafragment, waarin N , B en integer array $f[0..N)$ gegeven constanten zijn.

```

{ 0 ≤ N ∧ 1 ≤ B ∧ (∀i: 0 ≤ i < N: 0 ≤ f·i ≤ 1) }
a, q, r, n := 0, 0, 0, N
; do n ≠ 0 → n := n - 1 ; d := f·n
           ; a, q, r := 2*a + d, 2*q, 2*r + d
           ; if r < B → skip
           [] B ≤ r → q, r := q + 1, r - B
           fi
od
{ post R: a = q*B + r ∧ 0 ≤ r < B }
```

Bewijs de correctheid van de postconditie R; formuleer hiertoe de benodigde invarianten, een variante functie, en *alle* bewijsverplichtingen.

2. Gegeven is een integer array $f[0..N)$, met $0 \leq N$. Verder is gegeven dat in f alleen de waarden 3 of 7 voorkomen; dat wil zeggen: $(\forall i: 0 \leq i < N: f \cdot i = 3 \vee f \cdot i = 7)$. Ontwerp een programma voor het berekenen van de uitdrukking:

$$(\#i, j: 0 \leq i < j < N: f \cdot i = f \cdot j) \ .$$

3. Gegeven is een integer array $f[0..N)$, met $1 \leq N$. Ontwerp een programma voor het berekenen van de uitdrukking:

$$(\max i: 0 \leq i \wedge i^2 < N: f \cdot (i^2)) \ .$$

waardering	opgave 1: 10 punten
	opgave 2: 10 punten
	opgave 3: 10 punten
	eindcijfer: totaalscore div 3 .

TECHNISCHE UNIVERSITEIT EINDHOVEN

Faculteit Wiskunde en Informatica

Examen Ontwerp van Algoritmen 1 (2IA10), donderdag 10 maart 2005, 9.00 – 12.00 uur.

Noteer op het eerste vel de naam van uw instructeur. U mag geen boeken, ander studiemateriaal of calculators gebruiken.

1. Let op! Voor deze opgave mag U, desgewenst, het resultaat van het (in januari 2005) afgelegde deeltentamen opvoeren, mits dat resultaat ten minste 6 is. Dan schrijft U hier alleen op: “resultaat deeltentamen”. Als U dit niet doet en een oplossing voor de opgave inlevert blijft het resultaat van het deeltentamen buiten beschouwing.

We beschouwen het volgende programmafragment, waarin N een gegeven constante is en $f[0..N]$ een integer array:

```

{ 0 ≤ N }
n, r := N, 0
; do n ≠ 0 → n := n - 1 ; h := f · n + r
           ; if h ≤ 0 → r := h
           [] 0 ≤ h → r := 0
           fi
od
{ post R: r = ( min p: 0 ≤ p ≤ N : ( Σ i: 0 ≤ i < p : f · i ) ) }

```

Bewijs de correctheid van de postconditie R ; formuleer hierbij *alle* bewijsverplichtingen. Hint: Vervang in R beide voorkomens van de constante 0 door variabele n .

2. Gegeven is een integer array $f[0..N]$, met $0 \leq N$. Ontwerp een programma voor het berekenen van de uitdrukking:

$$(\max i, j: 0 \leq i \leq j < N : A \cdot i + A \cdot j) \ .$$

3. Een oneindige rij natuurlijke getallen $F[0..\infty)$ is als volgt recursief gedefinieerd, voor alle natuurlijke n :

$$F_0 = 0 \ , \ F_{2*n+1} = n + F_{2*n} \ , \ \text{en} \ F_{2*n+2} = 2 * F_{n+1} \ .$$

Construeer een efficiënt programma ter berekening van F_N , voor gegeven N , $0 \leq N$.

waardering	opgave 1: 10 punten
	opgave 2: 10 punten
	opgave 3: 10 punten
	eindcijfer: $(1 + \text{totaalscore}) \text{ div } 3 \ .$

TECHNISCHE UNIVERSITEIT EINDHOVEN

Faculteit Wiskunde en Informatica

Examen Ontwerp van Algoritmen 1 (2IA10), woensdag 27 april 2005, 9.00–12.00 uur.

Noteer op het eerste vel de naam van uw instructeur. U mag geen boeken, ander studiemateriaal of calculators gebruiken.

1. **Let op!** Voor deze opgave mag U, desgewenst, het resultaat van het (in januari 2005) afgelegde deeltentamen opvoeren, mits dat resultaat ten minste 6 is. Dan schrijft U hier alleen op: “resultaat deeltentamen”. Als U dit niet doet en een oplossing voor de opgave inlevert blijft het resultaat van het deeltentamen buiten beschouwing.

We beschouwen het volgende programmafragment, waarin N , B en integer array $f[0..N)$ gegeven constanten zijn.

```

{  $0 \leq N \wedge 1 \leq B \wedge (\forall i: 0 \leq i < N: 0 \leq f \cdot i \leq 1)$  }
 $a, q, r, n := 0, 0, 0, N$ 
; do  $n \neq 0 \rightarrow n := n - 1; d := f \cdot n$ 
      ;  $a, q, r := 2 * a + d, 2 * q, 2 * r + d$ 
      ; if  $r < B \rightarrow$  skip
      []  $B \leq r \rightarrow q, r := q + 1, r - B$ 
      fi
od
{ post R:  $a = q * B + r \wedge 0 \leq r < B$  }
```

Bewijs de correctheid van de postconditie R; formuleer hiertoe de benodigde invarianten, een variante functie, en *alle* bewijsverplichtingen.

2. Gegeven is een integer array $f[0..N)$, met $0 \leq N$. Verder is gegeven dat in f alleen de waarden 3 of 7 voorkomen; dat wil zeggen: $(\forall i: 0 \leq i < N: f \cdot i = 3 \vee f \cdot i = 7)$. Ontwerp een programma voor het berekenen van de uitdrukking:

$$(\#i, j: 0 \leq i < j < N: f \cdot i < f \cdot j) .$$

3. Een oneindige rij natuurlijke getallen $F[0..\infty)$ is als volgt recursief gedefinieerd, voor alle natuurlijke n :

$$F_0 = 0, \quad F_{2 * n + 1} = n + F_{2 * n}, \quad \text{en} \quad F_{2 * n + 2} = 2 * F_{n + 1} .$$

Construeer een efficiënt programma ter berekening van F_N , voor gegeven N , $0 \leq N$.

waardering	opgave 1: 10 punten
	opgave 2: 10 punten
	opgave 3: 10 punten
	eindcijfer: $(1 + \text{totaalscore}) \text{ div } 3$.

TECHNISCHE UNIVERSITEIT EINDHOVEN

Faculteit Wiskunde en Informatica

Examen Ontwerp van Algoritmen 1 (2IA10), donderdag 16 maart 2006, 9.00–12.00 uur.

Noteer op het eerste vel de naam van uw instructeur. U mag geen boeken, ander studiemateriaal of calculators gebruiken.

1. Let op! Voor deze opgave mag U, desgewenst, het resultaat van het (in januari 2006) afgelegde deeltentamen opvoeren, mits dat resultaat ten minste 6 is. Dan schrijft U hier alleen op: “resultaat deeltentamen”. Als U dit niet doet en een oplossing voor de opgave inlevert blijft het resultaat van het deeltentamen buiten beschouwing.

Gegeven zijn een constante N en een integer functie f op het domein $[0..N]$. Gegeven is dat alle functiewaarden van f verschillend zijn:

```

{ 0 ≤ N }
p, q := 0, N
; do p ≠ q → if f·p < f·q → p := p + 1
                [] f·q < f·p → q := q - 1
                fi
od
{ post R: 0 ≤ p ≤ N ∧ f·p = (max i: 0 ≤ i ≤ N: f·i) }

```

Bewijs de correctheid van postconditie R ; formuleer hierbij *alle* bewijsverplichtingen.

2. Gegeven is een integer array $f[0..N]$, met $0 \leq N$. Ontwerp een programma voor het berekenen van de uitdrukking:

$$(\#i, j: 0 \leq i < j < N: f \cdot i = f \cdot j) .$$

3. Een oneindige rij natuurlijke getallen $F[0..∞)$ is als volgt recursief gedefinieerd, voor alle natuurlijke n :

$$F_0 = 0 \quad , \quad F_1 = 1 \quad , \quad F_{2*n} = F_n \quad , \quad \text{en} \quad F_{2*n+1} = F_n + F_{n+1} .$$

Construeer een programma ter berekening van F_N , voor gegeven N , $0 \leq N$.

waardering	opgave 1: 10 punten
	opgave 2: 10 punten
	opgave 3: 10 punten
	eindcijfer: $(1 + \text{totaalscore}) \text{ div } 3 .$

TECHNISCHE UNIVERSITEIT EINDHOVEN

Faculteit Wiskunde en Informatica

Examen Ontwerp van Algoritmen 1 (2IA10), donderdag 11 mei 2006, 9.00–12.00 uur.

Noteer op het eerste vel de naam van uw instructeur. U mag geen boeken, ander studiemateriaal of calculators gebruiken.

1. Let op! Voor deze opgave mag U, desgewenst, het resultaat van het (in januari 2006) afgelegde deeltentamen opvoeren, mits dat resultaat ten minste 6 is. Dan schrijft U hier alleen op: “resultaat deeltentamen”. Als U dit niet doet en een oplossing voor de opgave inlevert blijft het resultaat van het deeltentamen buiten beschouwing. We beschouwen het volgende programmafragment, waarin N een gegeven constante is en $f[0..N)$ een integer array:

```

{ 0 ≤ N }
n, r := N, 0
; do n ≠ 0 → n := n - 1 ; h := f · n + r
           ; if h ≤ 0 → r := h
           [] 0 ≤ h → r := 0
           fi
od
{ post R: r = ( min p: 0 ≤ p ≤ N : ( Σ i: 0 ≤ i < p : f · i ) ) }

```

Bewijs de correctheid van de postconditie R ; formuleer hierbij *alle* bewijsverplichtingen. Hint: Vervang, ter verkrijging van een mogelijke invariant, in R beide voorkomens van de constante 0 door variabele n .

2. Gegeven is een integer array $f[0..N)$, met $0 ≤ N$. Verder is gegeven dat in f alleen de waarden 3 of 7 voorkomen; dat wil zeggen: $(\forall i: 0 ≤ i < N: f \cdot i = 3 \vee f \cdot i = 7)$. Ontwerp een programma voor het berekenen van de uitdrukking:

$$(\#i, j: 0 ≤ i < j < N: f \cdot i = f \cdot j) \ .$$

3. Een oneindige rij natuurlijke getallen $F[0..∞)$ is als volgt recursief gedefinieerd, voor alle natuurlijke n :

$$F_0 = 0 \ , \ F_{2*n+1} = n + F_{2*n} \ , \ \text{en} \ F_{2*n+2} = 2 * F_{n+1} \ .$$

Construeer een efficiënt programma ter berekening van F_N , voor gegeven N , $0 ≤ N$.

waardering	opgave 1: 10 punten
	opgave 2: 10 punten
	opgave 3: 10 punten
	eindcijfer: $(1 + \text{totaalscore}) \text{ div } 3$.