MF werkexemplaar

Over "intermittent" en "inductive" assertions

Maarten Fokkinga, 29 maart 1985

Intermittent assertions en inductive assertions zijn termen die in de literatuur gebruikt worden om twee verschillende ~~besso~~ methoden aan te duiden voor het bewijzen van de korrektheid van een programma. Wellicht dat een betere terminologie luidt "data gerichte" en "syntaxis gerichte" methoden. (De laatste methode is de bekende methode met invarianten à la Hoare.) Ik zet hieronder beide methoden uiteen.

***

Zij prog een programma of programmafragment. Een door prog opgeroepen berekening is een rij toestanden $s_0, s_1, ---$ waarbij iedere toestand s een tweetal $L{:}g$ is, met L een label of programmapunt ("waar de besturing is") en g een ("daarbij horende") geheugeninhoud is. De begin-toestand is gegeven door het initiele geheugen; de label is de startlabel van het programma. Verder ontstaat iedere toestand uit de voorgaande door vanuit het programmapunt met het geheugen één elementaire aktie van het programma uit te voeren.

Willen we de korrektheid van een programma bewijzen, dan moeten we iets bewijzen over alle mogelijke berekeningen die door ~~een~~ het programma worden opgeroepen. Met name moeten we aantonen dat als de initiele geheugeninhoud aan gegeven preconditie P voldoet, de finale geheugeninhoud aan gegeven postconditie Q voldoet (en ook dat de berekening ook eindig is). Omdat de door een programma opgeroepen berekeningen in het algemeen in lengte kunnen varieren en vaak onbegrensd lang (hoewel alle eindig) kunnen zijn, moet zo'n bewijs op de een of andere manier inductief zijn. Dat wil zeggen, de berekening wordt opgedeeld in deelberekeningen en die deelberekeningen worden net zo weer opgedeeld in delen, enzovoorts, en dan wordt het korrektheidsbewijs geleverd over de berekeningen met inductie naar de zojuist gevormde structuur: voor de elementaire berekeningsstappen moet er iets expliciet bewezen worden, en aannemende dat de delen korrekt zijn moet aangetoond worden dat de samenstelling van die deelberekeningen tot een groter geheel ook korrekt is.

De vraag rijst nu, hoe moeten we een berekening opdelen? Welnu, daar zijn twee verschillende strategieen voor, leidende tot de "intermittent" en de "inductive" assertions. Hier zijn ze.

<u>Syntaxis gerichte opdeling.</u> Deel de berekening zó op dat ieder deel de berekening is die door een syntactisch deel van de programma tekst wordt opgeroepen. Met name wordt de door een repetitie opgeroepen berekening opgedeeld in delen die precies door de repetitie-romp worden opgeroepen. Korrektheid van een deelberekening kan dan geformuleerd worden als de korrektheid van een syntactisch onderdeel van het programma. Dus op ieder punt in de programmatekst (of alleen op de cruciale punten!) komen asserties te staan. Zij L zo'n cruciaal punt, en zij P de assertie die bij L komt te staan. Dan moeten we dus aantonen dat <u>iedere</u> <u>keer</u> wanneer de besturing bij L komt ~~[illegible]~~ de geheugeninhoud aan P voldoet.

<u>Data gerichte opdeling.</u> Deel de berekening zó op dat ieder deel een zinvolle bewerking is op de data die door de gehele berekening wordt bewerkt. Met name bij gestructureerde data zou zo'n onderdeel van de berekening precies kunnen korresponderen met de bewerking op een onderdeel van de data. (Meestal zal de syntactische structuur van de programmatekst al aansluiten bij de structuur van de data; in dat geval is deze opdeling tevens een syntaxis gerichte opdeling! Maar <u>soms</u>, zoals bij de DSW-algoritme en, algemener, bij door transformatie ontstane programma's,

sluit de syntactische structuur niet aan bij de structuur van de bewerkte data!)
Stel nu dat we zo'n opdeling van de berekening hebben bedacht, hoe formuleren we dan ~~de korrekt~~ de korrektheid? Bezie de volgende schematische berekening; het ~~deel~~ aangegeven stuk is ~~zo~~ een deel

$$L_0 : g_0 ,\ \dots\ , \underbrace{L' : g' \ \dots\ L'' : g''} ,\ \dots$$

van de opsplitsing. Stel dat voor dit type onderdelen de korrektheid "bij conditie P' aan 't begin geldt conditie P'' aan 't eind" bewezen moet worden (met inductie! dus voor onderdelen van dit type wordt de korrektheid al aangenomen als inductie hypothese!). Dit kan dan geformuleerd worden als

$$L' : P' \quad \text{leidt tot} \quad L'' : P'' \qquad \text{---- } (*)$$

of, $L' : P'$ impliceert dat ooit eens $L'' : P''$ (in formules $L' : P' \Rightarrow \Diamond L'' : P''$). Hierin heeft $L : P$ de interpretatie: "de besturing is op punt L en de geheugeninhoud voldoet aan assertie P".

Formules van de vorm (*) kunnen nu aan de hand van de programmatekst bewezen worden: Neem aan dat P' geldt op L'; propageer dan P' door de op L' volgende statements heen (bijvoorbeeld op de bekende manier, met name met gebruik van de regel $\{P[x/expr]\}\ x:=expr\ \{P\}$) en pas steeds zo mogelijk een inductiehypothese toe (die je in staat stelt van de ene label verder te gaan met propageren vanaf een andere label} met andere assertie).

Een voorbeeld

De moeilijkheid bij het vinden van goede (eenvoudige) voorbeelden is dat bij elegante (gestructureerde, goede) programma's de syntactische structuur meestal overeenkomt ~~bij~~ met de structuur van de bewerkte data. Ik presenteer hier een elegant programma, waarvoor er weliswaar ook makkelijk een syntaxis gericht korrektheidsbewijs geleverd kan worden, maar waarvoor er ook een elegant data gericht korrektheidsbewijs is. Het programma moet het aantal knopen in een binaire boom berekenen. Het luidt als volgt.

```
    p, t, s := R, 0, ∅;
L:  do  p ≠ nil  -A->  push(s, p↑.r);  p := p↑.l;  t := t+1
    []  p = nil ∧ s ≠ ∅  -B->  pop(s, p)
    od
```


Dit programma kan ontstaan zijn door een voor de hand liggende recursieve procedure te transformeren. Je zou ~~dus~~ kunnen zeggen dat de structuur van de programmatekst niet direkt aansluit bij de structuur van de data: binaire bomen. We zullen zien dat het korrektheidsbewijs volgens de datagerichte methode wel direkt aansluit bij de structuur van binaire bomen. Maar allereerst geven we een syntaxis-gericht korrektheidsbewijs.

De invariant van de repetitie luidt:

$$n(R) = t + n(p) + (\Sigma x \text{ in } s.\ n(x))$$

waarbij n ~~de~~ funktie is die bij willekeurige pointer P het aantal knopen (nodes) in de boom P geeft. De stapelvariabele s wordt dus zó gebruikt dat daarin nog de te tellen subbomen staan. Dat deze bewering door de initialisatie wáár wordt gemaakt, en door de takken (A en B) van de repetitie in stand wordt gehouden, is gemakkelijk aan te tonen. Dat laten we over aan de lezer. Na terminatie geldt bovendien de negatie van de gezamelijke tests, dus $s = \emptyset$ en $p = nil$, dus $n(R) = t + 0 + 0 = t$. [Vraag aan de lezer: waarom termineert de repetitie? Geef een geheeltallige uitdrukking die per slag van de repetitie afneemt maar --op grond van een of andere invariant-- niet negatief kan zijn.]

Voor het data-gerichte korrektheidsbewijs gaan we als volgt te werk. We bewijzen de uitspraak

$$\forall P, T, S.\quad L:\{t=T,\ p=P,\ s=S\}$$
$$\text{leidt tot}$$
$$L:\{t=T+n(P),\ p=\underline{nil},\ s=S\}$$

Deze uitspraak zegt dat gekomen bij het testen van de herhalingsconditie(s) met $p=P$ ooit eens weer dat punt bereikt wordt maar zó dat precies heel boom P bij t is bijgeteld. Let wel, in het algemeen is de repetitie dan nog niet geëindigd! Die eindigt pas met bovendien $s=\emptyset$.

We bewijzen de uitspraak met inductie naar de structuur van de boom P. Dat een boom P' onderdeel is van een boom P'' noteren we met $P' < P''$. De relatie $<$ op bomen is een welordening en dus geschikt voor een inductieargument. (Wie onbekend is met deze termen leze $P' < P''$ als "het aantal knopen in P' is kleiner dan dat in P''"; het bewijs is

dan met inductie naar het aantal knopen in P.)
Hier is het bewijs.

<u>Ingeval</u> P de kleinst mogelijke boom is (geen voorgangers volgens de relatie $<$ heeft), moet gelden $P = nil$ en is de begin-toestand $L:\{t=T,\ p=P,\ s=S\}$ tevens de eindtoestand $L:\{t=T=T+0=T+n(P),\ p=nil,\ s=S\}$.
<u>Ingeval</u> P niet de kleinst mogelijke boom is (wel voorgangers volgens de relatie $<$ heeft), nemen we als inductie hypothese aan dat

$$\forall P'<P,\ T', S'.\quad L:\{t=T',\ p=P',\ s=S'\}$$
$$\text{leidt tot}$$
$$L:\{t=T'+n(P'),\ p=nil,\ s=S'\}$$

Welnu, hier volgt de redenering dat de begintoestand tot de eindtoestand leidt.

$L:\ \{t=T,\ p=P,\ s=S\}$
  -- $P \neq nil$ gesteld, dus via tak A leidt dit tot
$L:\ \{t=T+1,\ p=P\uparrow.l,\ s=S \frown P\uparrow.r\}$
  -- $P\uparrow.l < P$, dus vlgs ind hyp. (met $P', T', S' = P\uparrow.l, T+1, S\frown P\uparrow.r$):
$L:\ \{t=T+1+n(P\uparrow.l),\ p=nil,\ s=S\frown P\uparrow.r\}$
  -- $p=nil,\ s\neq\emptyset$ dus via tak B leidt dit tot

L: $\{t = T + 1 + n(P\uparrow.\ell),\ p = P\uparrow.r,\ s = S\}$

— $P\uparrow.r < P$, dus vlgs ind hyp (met $P', T', S' = P\uparrow.r,\ T+1+n(P\uparrow.\ell),\ S$):

L: $\{t = T + 1 + n(P\uparrow.\ell) + n(P\uparrow.r) = T + n(P),\ p = nil,\ s = S\}$

En hiermee is het bewijs voltooid.

We zien dat het korrektheidsbewijs lineair is in de lengte van de programmatekst, want beide takken A en B zijn slechts eenmaal in beschouwing genomen. Voorts is de inductiehypothese tweemaal gebruikt, korresponderende met de struktuur van binaire bomen; (bij n-aire bomen zou de hypothese n maal gebruikt worden). De triviale boom (nil) korrespondeert met een triviaal stuk in het bewijs. Kortom, het bewijs is data-gericht. Omdat het bewijs data-gericht is, zijn de formuleringen van de asserties ook data-gericht: er komen géén andere notaties in voor dan degene die al bekend zijn op moment van probleem- en programmaformulering. Dit staat in tegenstelling met de invariant van het syntaxis-gerichte bewijs: de notatie

$$\Sigma\, x \in s.\ ----$$

hebben we speciaal voor het korrektheidsbewijs te hulp moeten roepen; hij is vreemd aan de probleem-

stelling en aan het begrip "stapel". [Overigens, ~~de~~ het syntaxis-gerichte bewijs blijft goed ook als we pop(S,p) veranderen in "haal een of ander element uit s en ken dat toe aan p". Het data-gerichte bewijs gaat dan niet meer op, maar de berekening kan ook niet meer direkt aansluitend bij binaire bomen opgedeeld worden.]

De voordelen van de datagerichte methode komen wel heel nadrukkelijk te voorschijn bij het bewijs van de DSW-algoritme. De data-gerichte asserties zijn véél eenvoudiger te formuleren (en korrekt te bewijzen) dan de syntaxis-gerichte asserties. Met name de "verborgen stapel" hoeft niet benoemd en tot in detail vastgelegd en gebruikt te worden. Voor meer informatie, zie [Jansen en Fokkinga, 198?]. Wie meer wil lezen over "intermittent assertions", raadplege [Burstall 1974] en [Manna en Waldinger 1978]; ik heb dit verhaal geschreven zonder de inhoud van dik literatuur te kennen (of te herinneren).

* * *

Jansen, P.G., Fokkinga, M.M. (198?). In de maak.

Burstall, R.M.: Program proving as hand simulation with a little induction. In. Proc IFIP Congress 1974 pp 308-312.

Manna, Z., Waldinger, R.: Is "sometime" sometimes better than "always"? Intermittent assertions in proving progr. corr. CACM 21 (78) 159-17