Sorteren met behulp van formele procedures korrekt bewezen

Maarten M Fokkinga, 22 maart 1985.

Beschouw hetvolgende programma, afkomstig van Matthijs Kuiper, RUU.

```
( proc p = ( proc (int, int) void  q) void:
        if in = ∅
        then  q (-∞, +∞)
        else  int v; read (in, v);
              proc q' = (int m, n) void:
                    if m < v < n
                    then  q (m, v); write (uit, v); q (v, n)
                    else  q (m, n)  fi;
              p (q')
        fi;
  proc q0 = (int m, n) void : skip;
  p (q0)
)
```

Hierin is 'in' de naam voor de invoerfile, $in=\emptyset$ is een test of de invoerfile leeg is, en 'uit' is de naam voor de uitvoerfile. Als initieel geldt $in=\mathbb{IN}$ ~~dan geldt~~ en $uit=\emptyset$, dan geldt na afloop $in=\emptyset$ en $uit = sorted(\{x \in \mathbb{IN}\})$. Dus alle invoergetallen worden zonder duplicaten in gesorteerde volgorde afgedrukt.

We geven een korrektheidsbewijs met de bekende assertie-methode. Het bijzondere hierbij is dat in asserties niet alleen "gewone" beweringen zoals $in = I$ komen te staan, maar ook "ongewone" beweringen zoals $\{P\}q(m,n)\{Q\}$ (als een bewering over een parameter q).

Allereerst geven we de specificatie voor p; de korrektheid ervan zullen we later aantonen. De specificatie is precies zo sterk, dat daarmee het vereiste korrektheidsbewijs voor de romp van p geleverd kan worden. Voor de toepassing van p in het hoofdprogramma, de aanroep $p(q_0)$, had de specificatie best zwakker mogen wezen. Omdat dus de specificatie sterk genoeg is voor het korrektheidsbewijs van de romp, vormt deze specificatie de kern van de uitleg (verklaring, dokumentatie) van de werking van p. Hij luidt als volgt (een toelichting volgt erna).

waarom

$$pspec(p) = \quad \{in=I,\ uit=\emptyset,\ qspec(I_0,q)\}\ \ p(\{I_0, I\}\ q)\ \ \{in=\emptyset,\ uit=sorted\{x \in I_0 ++ I\}\}$$

met

$$qspec(I_0,q) = \quad \{uit=U\}\ \ q(\{U\}\ m, n)\ \ \{uit=U ++ sorted\{x \in I_0 \mid m<x<n\}\}$$

De notatie $\{P\}\ p(\{x,y\}\ a,b)\ \{Q\}$ kan ook gelezen worden als $\forall x,y,a,b.\ \{P\}\ p(a,b)\ \{Q\}$; de x en y (en a en b) zijn dus gebonden in deze formule. Voorts is ++ een notatie voor "gevolgd door" waarbij we voor 't gemak zowel rijen alsook elementen als operanden toelaten. In woorden zegt pspec(p):

als geldt in=I en uit=∅ en qspec(I_0, q)
dan vestigt de aanroep p(q)
de geldigheid van uit=sorted$\{x \in I_0{+}{+}I\}$ en in=∅

en dit alles voor willekeurige I_0, I en q. Als pspec(p) geldig is (of als hypothese wordt aangenomen), dan mogen we dus tot de korrektheid van

{in=I, uit=∅, qspec(I_0, q)}
p(q)
{in=∅, uit=sorted$\{x \in I_0{+}{+}I\}$}

konkluderen, voor willekeurige expressies I_0, I, q; zo'n stap in het korrektheidsbewijs zullen we instantiatie noemen. Bijvoorbeeld, in het hoofdprogramma,

{in=$\mathbb{N}$, uit=∅, qspec(∅, q_0)}
p(q_0) --instantiatie pspec(p) met I, I_0, q = $\mathbb{N}$, ∅, q_0
{in=∅, uit=sorted$\{x \in \mathbb{N}\}$}

ambigue verwijzing

Beschouw nu eens qspec(I_0, q). De informele interpretatie hiervan luidt

als uit=U geldt,
dan vestigt de aanroep q(m,n)
de geldigheid van uit=U++sorted$\{x \in I_0 \mid m<x<n\}$

en hierbij zijn U, m en n nog vrij te kiezen (per aanroep van q!). Let wel, I_0 is niet per aanroep van q vrij te kiezen; I_0 ligt vast voor q -- maar is wel universeel gekwantificeerd in pspec(p)! Binnen de romp van p zal I_0 zó gebruikt worden dat het de rij van alreeds ingelezen getallen voorstelt; deze getallen staan in de stapel van variabelen, alle genaamd v, en toegankelijk via de procedures q.

Zij r een procedure gedeclareerd als proc r = (a): romp. Een specificatie {P} r({x,} a){Q} is korrekt als we kunnen aantonen dat ({x,} a): {P} romp {Q} korrekt is, waarbij we ~~zelfs~~ voor de recursieve aanroepen de specificatie voor r zelf als inductie-hypothese mogen aannemen. We zullen zo'n korrektheidsbewijs in de programmatekst als volgt opnemen ("annoteren"):

```
proc r
   : {P} r({x,} a) {Q}
   = ({x,} a): {P} romp {Q}
```

Tenslotte nog deze conventie. Wanneer S een programmadeel is en P is een assertie zó dat S en P "interferentie-vrij" zijn (dwz de variabelen die door S veranderd worden --al of niet expliciet--, komen niet in P voor), dan maken we gebruik van de invariantie van P over S, {P}S{P}, zonder dat nog verder te rechtvaardigen (dus ook het ontbreken van interferentie wordt niet aangetoond).

Het geannoteerde programma ziet er nu als volgt uit

```
( {in=IN, uit=∅}
  proc P
    : pspec(P)
    = ({I₀, I,} q) void:
      ({in=I, uit=∅, qspec(I₀,q)}
       if in=∅
       then {I=∅, uit=∅}
              q(-∞, +∞)   --instantiatie qspec(I₀,q) met U,m,n=∅,-∞,+
              {uit=∅++sorted{x∈I₀ | -∞<x<+∞}}
              {uit=sorted{x∈I₀++I}}
       else {in≠∅ zeg in=i++I'=I, uit=∅, qspec(I₀,q)}
              int v; read(in, v);
              {in=I', v=i, uit=∅, qspec(I₀,q)}
              proc q'
                : qspec(I₀++v, q')
                = ({U,} m, n) void:
                  ({uit=U}
                   if m<v<n
                   then {uit=U}
                          q(m, v);  --instantiatie qspec(I₀,q)
                          {uit=U++sorted{x∈I₀ | m<x<v}}
                          write(uit, v);
                          {uit=U++sorted{x∈I₀ | m<x<v}++v}
                          ⋮
```

```
        ⋮
24              q(v,n) --instantiatie qspec(I0,q)
25              {uit=U++sorted{x∈I0|m<x<v}++v ++
26                   sorted{x∈I0|v<x<n}}
                -- m<v<n
27              {uit=U++sorted{x∈I0++v|m<x<n}}
28          else {uit=U}
29              q(m,n) --instantiatie qspec(I0,q)
30              {uit=U++sorted{x∈I0|m<x<n}}
31              -- not (m<v<n)
32              {uit=U++sorted{x∈I0++v|m<x<n}}
33          fi
34          {uit=U++sorted{x∈I0++v|m<x<n}}
35        ); --end q'
36      {in=I', v=i, uit=∅, qspec(I0++v, q')}
37      {in=I', uit=∅, qspec(I0++i, q')}
38      p(q') --instantiatie pspec(p) met I,I0,q = I', I0++i, q'
39      {in=∅, uit=sorted{x∈(I0++i)++I'}}
40      {in=∅, uit=sorted{x∈I0++I}}   -- I = i++I'
41    fi
42    {in=∅, uit=sorted{x∈I0++I}}
43  ); --end p
44 {in=IN, uit=∅}
45 proc q0
46   : qspec(∅,q0)
47   = ({U,}m,n) void:
48     ({uit=U} skip {uit=U++sorted{x∈∅|m<x<n}}); --end q0
49 {in=IN, uit=∅, qspec(∅,q0)}
   ⋮
```



```
   ⋮
50     p(q0) --instantiatie pspec(p) met I0,I,q := ∅, IN, q0
51     {in=∅, uit=sorted{x∈∅++IN}}
52     {in=∅, uit=sorted{x∈IN}}
53  ) --end program
```

Wanneer eenmaal geschikte pspec(p) en qspec(I_0,q) gevonden en geformuleerd zijn, is het korrektheids-bewijs zelf verder recht-toe recht-aan. Een informele uitleg van "de werking" van dit programma zal, als die uitleg korrekt is, ~~bovenstaand~~ bewijs op de voet volgen, zo is mijn stellingname. Opmerkelijk is ~~dan~~ nog ~~ook~~ dat de specificatie die voor q' bewezen wordt, niet qspec(I_0++i, q') luidt, maar qspec(I_0++v,q'). Pas in regel 36/37 vindt de overgang tot qspec(I_0++i,q') plaats; (eventueel hadden we de overgang van I_0++v tot I_0++i nog later kunnen doen plaats vinden).

dat lijkt toch een interessante exercitie

Hetzelfde programma nu in een funktionele taal

Wellicht is het interessant hetzelfde programma in een funktionele taal te zien, met de vertaling van het korrektheidsbewijs. Het enige opvallende is de grotere beknoptheid. We kiezen de SASL, KRC, TWENTEL stijl van funktionele programma's.

$$
\begin{array}{lll}
\underline{def} & p\ q\ [\,] = & q\ (-\infty)\ (+\infty) \\
 & p\ q\ (i{:}I') = & p\ q'\ I' \\
 & & \underline{where}\quad q'\ m\ n = m < i < n \rightarrow q\ m\ i \mathbin{++} [i] \mathbin{++} q\ i\ n \\
 & & \qquad\qquad\qquad\qquad q\ m\ n
\end{array}
$$

?

$p\ q_0\ IN$ $\underline{where}$ $q_0\ m\ n = [\,]$?

We beweren dat (sorted $\{x \leftarrow IN\}$) wordt opgeleverd. We bewijzen daartoe

$$
\begin{array}{ll}
\forall I_0, I. & q\ m\ n = \text{sorted}\ \{x \leftarrow I_0 \mid m < x < n\} \\
 & \Rightarrow\quad p\ q\ I = \text{sorted}\ \{x \leftarrow I_0 \mathbin{++} I\}
\end{array}
$$

Het bewijs is met inductie: we nemen bovenstaande bewering al voor waar aan voor de recursieve aanroepen van p. (We passen dus zg. "computational induction" toe. Of vergis ik me hierin?).

geen idea

logici noemen het inductie over de lengte v/d berekening

Hier is het bewijs.

Zij I_0 en I willekeurig,

$\underline{case}$ $I = [\,]$

$$
\begin{array}{lll}
p\ q\ [\,] & = q\ (-\infty)\ (+\infty) & \text{-- pas nu aanname over } q \text{ toe} \\
 & = \text{sorted}\ \{x \leftarrow I_0 \mid -\infty < x < +\infty\} & \\
 & = \text{sorted}\ \{x \leftarrow I_0 \mathbin{++} I\} & \text{-- } I = [\,]
\end{array}
$$

$\underline{case}$ $I = i{:}I'$

$$
\begin{array}{l}
p\ q\ (i{:}I') = \\
= p\ q'\ I' \\
\quad \underline{where}\quad q'\ m\ n = m<i<n \rightarrow q\ m\ i \mathbin{++} [i] \mathbin{++} q\ i\ n \\
\qquad\qquad\qquad\qquad\qquad q\ m\ n \\
\qquad\qquad\qquad \text{-- pas nu aanname over } q \text{ driemaal toe} \\
\qquad\qquad\qquad = m<i<n \rightarrow \text{sorted}\ \{x \leftarrow I_0 \mid m<x<i\} \mathbin{++} [i] \mathbin{++} \\
\qquad\qquad\qquad\qquad\qquad \text{sorted}\ \{x \leftarrow I_0 \mid i < x < n\} \\
\qquad\qquad\qquad\qquad \text{sorted}\ \{x \leftarrow I_0 \mid m < x < n\} \\
\qquad\qquad\qquad = m<i<n \rightarrow \text{sorted}\ \{x \leftarrow I_0 \mathbin{++} [i] \mid m<x<n\} \\
\qquad\qquad\qquad\qquad \text{sorted}\ \{x \leftarrow I_0 \mathbin{++} [i] \mid m<x<n\} \\
\qquad\qquad\qquad = \text{sorted}\ \{x \leftarrow I_0 \mathbin{++} [i] \mid m<x<n\} \\
= \text{-- vlgs inductie hypothese met } I_0, I := I_0 \mathbin{++} [i],\ I' \\
\quad \text{sorted}\ \{x \leftarrow (I_0 \mathbin{++} [i]) \mathbin{++} I'\} \\
= \text{sorted}\ \{x \leftarrow I_0 \mathbin{++} I\} \qquad\qquad \text{-- } I = [i] \mathbin{++} I'
\end{array}
$$

Dus in beide $\underline{cases}$ is bewezen

$p\ q\ I = \text{sorted}\ \{x \leftarrow I_0 \mathbin{++} I\}$

en daarmee zijn we klaar.