Het Acht Koninginnen Probleem (gericht aan Docenten, niet: Studenten)

Maarten Fokkinga, 10 febr 87

Aan de orde komt

- representatie-keuze;
- specificatie → implementatie: efficientieverbetering;
- filter promotion (preselection, pruning of seach space);
- uitkomst (semantiek) v.e. funct. progr. versus
  het operationele gedrag ervan onder lazy evaluation;
- backtracking;
- efficientie beschouwingen (qua grootte-orde!)
  zowel onafhankelijk van de evaluatiemethode,
  alsook enigszins afhankelijk ervan;
- implementatie in imperatieve taal,
  mbv coroutines, formele procedures, en "gewoon".
- progr. transformatie

Representatiekeuze:

Koningin: (i,j). Plaatsing stel Koninginnen: $[\ldots,(i,j),\ldots]$
(er zijn vele alternatieven mogelijk).

Dus:

$$\text{slaat}^{\text{niet}}\ (i,j)\ (i',j') = i \neq i' \ \&\ j \neq j' \ \&\ |i-i'| \neq |j-j'|$$

```
veilig K p = and [slaatniet K K' | K'←p]
goed [] = True
goed (K:p) = veilig K p & goed p
```

Specificatie Acht Koninginnen Probleem:
Gevraagd [p | p ← pl 8 ; goed p] of de hd ervan.

```
pl 0 = [[]]
pl n = [(i,j):p | i,j ← [1..8]; p ← pl (n-1)]
```

Eff. beschouwing:
pl 8 heeft 64^8 elten, die moeten allen getest worden, of slechts een fractie ervan. Dit kost te veel tijd/stappen.

Spec → impl: alles wat er nu volgt.

Eff. verbetering: filter promotion / preselection, (progr. transf.)
Verklein de lijsten pl n door het filter 'goed' te vervroegen naar (te "promoten" naar) de def van pl n:
Doel: pl' n = [p | p ← pl n; goed p].

```
pl' 0 = [[]]
pl' n = [(n,j):p | j ← [1..8]; p ← pl' (n-1); veilig (n,j) p]
```

Eff. verbetering - ad hoc (= voor dit geval).
Veronderstel (i) één plaatsing gevraagd, (ii) mach. rekent niets meer uit dan strikt noodzakelijk (bijv. lazy evaluation).
Dan "j ← ..; p ← .." bespaart op j's, "p ← .. ; j ← .." bespaart op p's. Voortbrengen van p's is duurder. Dus nu:

```
pl'' 0 = [[]]
pl'' n = [(n,j):p | p ← pl'' (n-1); j ← [1..8]; veilig (n,j) p]
```

(Abstracte) interpretatie: gevraagde uitkomst wordt inderdaad gespecificeerd/uitgedrukt.
Operationele interpretatie: onder lazy evaluation vertoont de berekening een backtracking gedrag.

Implementaties (met één globale var p voor plaatsingen):
Gemakkelijk mbv coroutines. In een ad hoc notatie:

```
var p: array [1..8] of 1..8;
coroutine pl(n: int);
    if n=0 → return
    [] n>0 → for each return of pl(n-1) do
                 for j from 1 to n do
                     if veilig(n,j) then p[n]:=j; return fi
    fi;
...
for each return/the first return of pl(8) do write(p[1..8])
```

Zonder coroutines, kan het rechtstreeks mbv. formele proc's.
(Motivatie hier NIET vermeld!)

```
var p: array [1..8] of 1..8;
proc pl(n: int; proc verdere acties);
    if n=0 → verdere acties
    [] n>0 → pl(n-1, proc: for j:=1 to n do
                               if veilig(n,j) then p[n]:=j; verdere acties fi)
    fi;
...
pl(8, proc: write(p[1..8]) )
```

Zonder coroutines of formele procedures. We transformeren eerst het programma tot een elementsgewijze iteratieve versie.

Doel: ext_pl(n, p) (met $p \in pl(n)$) genereert alle uitbreidingen (extensies) van p die in pl(8) zitten, formeel:

(*) $[p' \mid p \leftarrow pl''(n);\ p' \leftarrow ext\_pl\ n] = pl''8$

```
ext_pl 8 p = [p]
ext_pl n p = [p' | j←[1..8]; veilig (n+1,j) p; p'← ext_pl (n+1) ((n+1,j):p)]
```

Volgens (*) is nu $pl''8 = ext\_pl\ 0\ []$.

De vertaling naar Pascal is nu:

```
var p: array [1..8] of 1..8
proc ext_pl (n: int);
    if n=8 → write(p[1..8])
    [] n<8 → for j:=1 to 8 do
                  if veilig (n+1,j) then p[n+1]:=j; ext_pl (n+1) fi
    fi;
...
ext_pl (0)
```

Opgaven ter oefening

1. Beschouw de eerste versie van pl en de volgende drie varianten pl_i:

$pl_2\ n = [(i,j):p \mid j,i \leftarrow [1..8];\ p \leftarrow pl_2(n-1)]$

$pl_3\ n = [(i,j):p \mid p \leftarrow pl_3(n-1);\ i,j \leftarrow [1..8]]$

$pl_4\ n = [(i,j):p \mid i \leftarrow [1..8];\ p \leftarrow pl_4(n-1);\ j \leftarrow [1..8]]$

$pl_5\ n = [p ++ [(i,j)] \mid i,j \leftarrow [1..8];\ p \leftarrow pl_5(n-1)]$

Geef voor ieder van $pl_1 8, \ldots, pl_5 8$ de eerste negen elementen (dus de eerste negen plaatsingen van 8 koninginnen).

2. Kies als representatie: [j1, ..., j8] indien er Koninginnen staan op posities (1, j1), ..., (8, j8). Pas de definities aan.
3. Kiest als representatie voor één Koningin: (i, c) waarbij i het rij-nummer en c de kolom-letter is. Pas de definities aan.
4. Geef een recursieve definitie voor 'veilig' (zonder de functie 'and' te gebruiken).
5. Wijzig de imperatieve programma's zó dat niet alle maar slechts één oplossing wordt afgedrukt. (Bij coroutines makkelijk, bij de andere "moeilijk" dan wel "inefficient".) (Moraal: hiep hiep hoera voor lazy evaluation.)
6. Los op analoge wijze als bij het Acht Koninginnen probleem hetvolgende op: gevraagd alle partities (x, y, z) van n zodanig dat $1 \leq x \leq y \leq z$ en $x+y+z = n$. Wijs de onderwerpen uit "aan de orde komt" (pag 1) expliciet aan, m.n. de specificatie, filter promotion, operationele gedrag, implementatie.
7. Idem voor: gevraagd één/alle deelverzamelingen van een gegeven een lijst natuurlijke getallen, zó dat hun som een gegeven limiet S niet overschrijdt.

Literatuur

Wirth, N., Program development by stepwise refinement. C ACM 14 (1971) 221 ff.

Wirth, N., Algorithm + DataStructures = Programs. Chapter 3.4, 3.5.

Fokkinga, M.M., Backtracking and ..... , Memorandum INF 86-18

Fokkinga, M.M., Transformatie van Specificatie tot Implementatie.
In: Softw Spec Techn, (eds Schoenmakers, Poiters), te verschijnen in 1987.