MMF werkexemplaar

Bereikbaarheidstest voor grafen

Maarten Fokkinga, 4 sept 1985

We geven een functioneel programma dat louter een "uitgeklede" versie is (zowel t.a.v. programmatekst alsook t.a.v. ontstaansgeschiedenis) van een programma uit een (lezenswaardig) verhaal van Gerrit van der Hoeven.

* * *

Zij een graaf gegeven door een lijst Knoop van de knopen en een functie Pijl zó dat voor knopen k en k' geldt (Pijl k k') = er is een pijl van k naar k'. Zij voorts een vaste knoop endk gegeven. We definieren voor knopen k: <u>een pad van</u> k is een rij knopen $k_1, k_2, \ldots, k_n$ zó dat $k_1 = k$ en $k_n =$ endk en voor alle i met $1 \leq i < n$ geldt (Pijl k_i k_{i+1}). Gevraagd wordt een functie Bereikbaar zó dat (Bereikbaar k) = er is een pad van k.

We presenteren nu een zo "zuinig mogelijk" programma met een zo zuinig mogelijk korrektheidsbewijs. In feite wordt de bekende "olievlek-methode" functioneel geprogrammeerd.

We definieren een functie F die we alleen zullen gebruiken in contexten K ++ F K' K'' waarbij K, K', K'' voldoen aan het predikaat P(K, K', K''):

P(K, K', K'') :=

K, K', K'' vormt een partitie van Knoop zó dat

(i) endk in K ∪ K'

(ii) voor iedere k in K is er een pad;

(iii) voor iedere k' in K' is er een pad;

(iv) voor iedere k'' in K'' geldt voor ieder pad dat het, alvorens door een knoop in K te gaan, alreeds door een knoop in K' is gegaan.

(In termen van de "olievlekmethode": K is de vlek, K' de rand om de vlek, en K'' de rest.) Wanneer P(K, K', K'') geldt en K' = {}, dan kan er vanuit K'' geen pad zijn (want een pad eindigt in endk, dus gaat door K) en zijn K dus alle knopen van waaruit endk bereikbaar is. Definieren we dus

```
F [] K'' = []
```

dan is de gevraagde funktie te definieren door

```
bereikbaar k0 =
    some {k0 = k | k <- [] ++ F [endk] (Knoop -- [endk])}
```

Er rest ons nog F te definieren voor niet-lege eerste parameter~~, zo dat de grootte van die parameter bij recursieve aanroepen afneemt (en we) op den duur afneemt~~. Hiertoe roepen we de hulp van hetvolgende lemma in.

<u>Lemma</u>. Zij K, K', K'' zo dat $P(K, K', K'')$ geldt. Zij k'_0 in K' en zij $dK'' = \{k'' \mid k'' \leftarrow K''; \text{pijl } k'' \; k'_0\}$. Dan geldt ook $P(K \cup \{k'_0\},\ K' - \{k'_0\} \cup dK'',\ K'' - dK'')$.

<u>bewijs</u>

De verificatie van (i), (ii) en (iii) is heel eenvoudig. We bewijzen nu (iv). Zij $[k'', \ldots]$ een willekeurig pad vanuit willekeurige k'' in K''. Stel dat dit pad door $K \cup \{k'_0\}$ gaat; als het door K gaat dan gaat het volgens (iv) ook al eerder door een k'. Dus het pad heeft de vorm $[k'', \ldots, k', \ldots, k, \ldots]$ waarbij de aangegeven voorkomens de eerste in K' resp $K \cup \{k'_0\}$ zijn (en eventueel ~~...~~ vallen k' en k samen). Als nu $k' = k'_0$ dan moet zijn voorganger in dK'' dus in $K' - \{k'_0\} \cup dK''$ liggen; en als $k' \neq k'_0$ dan ligt k' zelf in $K' - \{k'_0\}$ dus in $K' - \{k'_0\} \cup dK''$.

Q.E.D.

Dus we definieren

$$F\ (k'_0 : K')\ \ K'' = k'_0 : F\ (K' \mathbin{++} dK'')\ (K'' \mathbin{--} dK'')$$

<u>where</u> $dK'' = \{k'' \mid k'' \leftarrow K''; \text{pijl } k'' \; k'_0\}$

Het is eenvoudig in te zien dat de recursie termineert: de grootte van de vereniging (concatenatie) van de twee argumenten neemt per recursie-stap met één af; vroeg of laat zal de eerste parameter dus leeg zijn.

De complexiteit is ook eenvoudig af te leiden. Zij n ~~het~~ aantal knopen in $\text{Knoop} = K \cup K' \cup K''$. Per β-reductie stap van F vindt hetvolgende plaats:

- een ++ en een -- en een { } -operatie, ieder een werklast van hoogstens n eenheden gevend (vroeg of laat: lazy! evaluation),
- hoogstens n aanroepen van ($\text{pijl } k'' \; k'_0$) (bij vaste k_0) en voorts

~~Dus de tijdscomplexiteit ligt in $\Theta(n^2)$~~.

Iedere stap heeft dus een werklast in $\Theta(n)$ + hoogstens n Pijl-evaluaties; er zijn hoogstens n stappen. Dus de tijdscomplexiteit ligt in: hoogstens n^2 Pijl-operaties + $\Theta(n^2)$ werk.

Functie F heeft nog meer --belangrijke-- eigenschappen die in het korrektheidsbewijs niet aan bod zijn gekomen. Met name worden de knopen in de resultaatlijst van F opgesomd naar toenemende lengte van een korste pad vanuit betreffende knoop. Deze eigenschap is cruciaal in een korrektheidsbewijs van de volgende variant van F:

```
G []     K'' = []
G (w':W') K'' = w' : G (W'++ dW') (K''--dK'')
             where  dK'' = {k'' ← K''; Pijl k'' (hd w')}
                    dW'  = {k'':w'  |  k'' ← dK''}
```

(W is mnemonisch voor Wandeling, i.e. Pad). In iedere ~~aanroep~~ kontekst W ++ G W' K'' geldt nu:

Zij K, K' = map hd W, map hd W'

dan: P (K, K', K'')

& W = {een korste pad van k | k ← K}

& W' = {een korste pad van k' | k' ← K'}

& W' is geordend naar toenemende lengte der elementen

& de elementen in W' hebben 'n lengteverschil hoogstens 1.

De korrektheid kan hiermee bewezen worden; het bewijs heeft dezelfde structuur als bij F. Merk wel op dat bij F's recursieve aanroep K' ++ dK'' vervangen mag worden door dK'' ++ K', maar dat bij G's recursieve aanroep W' ++ dW' <u>niet</u> vervangen mag worden door dW' ++ W' !

—//—