MF

Tekenen in SASL

Maarten Fokkinga, 31 jan 1984

Abstract Een SASL representatie voor tekeningen en bewerkingen daarmee wordt gegeven. Als voorbeeld wordt een kort en bondig programma gepresenteerd voor de zg. Hilbert krommen (ook wel eens Peano krommen genoemd).

* * *

Al het navolgende is louter een omzetting in SASL termen van het artikel

A.J. Cole: A note on space filling curves, Software - Practice and Experience, 13 (1983) pp 1181-1189.

In dat artikel staan bovendien nog meer voorbeelden en literatuur-verwijzingen. We zullen daarom kort zijn.

We zullen straks een representatie voor tekeningen kiezen, en de volgende bijhorende funkties implementeren. Denk voorlopig aan zoiets als een lijst van lijnstukken ter representatie van een tekening.

rot t h = "t (tegen de klok in) gedraaid over een hoek h"

scale t (fx, fy) = "de t' verkregen uit t door alle x-coordinaten met fx, en de y-coordinaten met fy te vermenigvuldigen"

shift t (vx, vy) = "de t' ontstaan uit t door alle x-coordinaten met vx, en de y-coordinaten met vy te vermeerderen"

join t0 t1 = "de vereniging van t0 en t1"

cat t0 t1 = "de concatenatie van t0 en t1, dwz. t0 en t1 middels het lijnstuk van (eindpunt t0) tot (beginpunt t1) ge-joined"

point (x, y) = "de tekening bestaande uit het punt (x, y)"

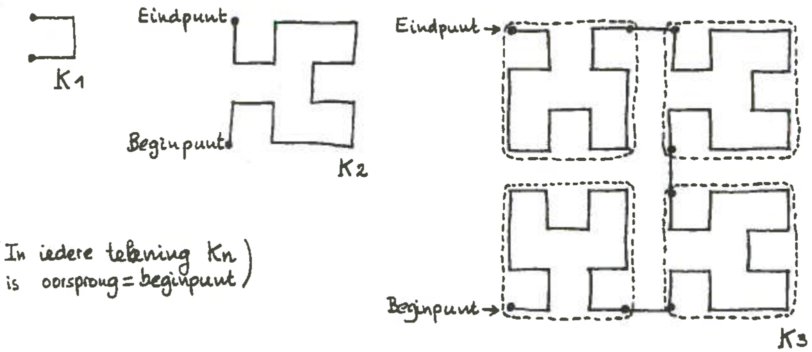
Merk op dat tekeningen een begin- en eindpunt hebben; daarvan wordt in cat gebruik gemaakt. Bijgevolg is join niet ~~..~~ commutatief; (de naam union zou wel commutativiteit suggereren en is daarom niet gekozen). Merk bovendien op dat (scale t (-1, +1)) de gespiegelde (t.o.v. de y-as) van t oplevert.

De uitbreiding van cat tot een niet-lege lijst van tekeningen is standaard:

```
catAll (t:nil)  = t
catAll (t: tl)  = cat t (catAll tl)
```

Dus catAll (t0, ..., tn) = (cat t0 (cat t1 ... tn)).

Met bovenstaande hulpmiddelen zijn veel recursief opgebouwde tekeningen goed uit te drukken. Als voorbeeld programmeren we nu de Hilbert-krommen, soms ook wel Peano-krommen genoemd.



De kromme K_n van orde n en breedte $b = 2**n - 1$ is louter de concatenatie van vier krommen $K' = K\ (n-1)$ met breedte $b' = 2**(n-1) - 1 = (b-1)$ div 2, die ieder op geschikte manier geroteerd, verschoven en geschaald zijn. Dit geldt zelfs voor $n=1$, waarbij K 0 (met breedte 0) uit louter één punt bestaat. Omwille van de efficientie maken we de breedte b tot een extra parameter van K.

```
K 0 0  =  point (0, 0)
K n b  =  catAll ( scale (rot K' π/4) (-1,1)
                 , shift K' (b'+1, 0)
                 , shift K' (b'+1, b'+1)
                 , shift (scale (rot K' -π/4) (-1,1)) (b', b)
                 )
```

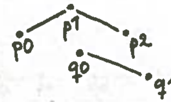


```
  where  b' = (b - 1) div 2
         K' = K (n-1) b'
```

De initiele aanroep is (K n (2**n - 1)).

Bovenstaande funktie K is recursief, maar niet iteratief (= last action recursion = tail recursion). Cole geeft in zijn artikel een tail recursieve formulering van de funktie K. (Onder sommige implementaties is tail recursion efficienter dan niet-tail recursion.)

Rest ons nog een representatie voor tekeningen te bedenken en de funkties rot, join, cat etc daarvoor te concretiseren. Mij lijkt een lijst van 'gebroken lijnstukken' wel zinvol, waarbij een "gebroken lijnstuk" op zich een lijst van (hoek- of begin- resp eind-)punten is. Voorbeeld:



≈ ((p0, p1, p2), (q0, q1))

Zo'n lijst kan bijna letterlijk als besturing van de tekentafelmachine dienen. Verdere uitwerking wordt aan de lezer overgelaten.

—//—