MF werkexemplaar

<u>Nederlandse telwoorden voor getallen</u>

Maarten Fokkinga, 17 april 1985.

We geven een SASL-achtige funktie die ~~voor~~ willekeurig natuurlijk getal in het nederlands omzet; bijvoorbeeld 123.456.789 = "honderddrieentwintig miljoen vierhonderdzesenvijftig duizend zevenhonderdnegenentachtig".

* * *

Ik vond het een plezierige oefening om de nederlandse telwoorden te analyseren en een programma te schrijven dat getallen in het nederlands omzet. Dat plezier wil ik de lezer niet onthouden, vandaar deze notitie. De kunst is om zoveel mogelijk regelmatigheid in de nederlandse weergave van getallen te vinden en die in het algoritme uit te buiten en om de onregelmatigheden toch zo elegant mogelijk te formuleren.

Officieel hoort willekeurig getal in het nederlands als één woord geschreven te worden. Omwille van de leesbaarheid zullen wij echter de telwoorden van tienmachten groter dan honderd als ~~eigen~~ zelfstandige (naam)woorden behandelen die van omringende woorden door spaties gescheiden worden, zie het eerder genoemde voorbeeld 123.456.789. (Dat we honderd of zelfs alle telwoorden

niet net zo behandelen geeft weer een extra onregelmatigheid en dus een extra uitdaging!)

Uit de volgende getalnamen moeten de nederlandse ~~getallenaamingen~~ telwoorden worden gevormd:

nul, een, twee, ---, negen;
tien, elf, twaalf, ---, negentien;
twintig, dertig, veertig, ..., negentig;
honderd, duizend, miljoen, biljoen = miljard = $10^{6.2}$,
triljoen = $10^{6.3}$, quadriljoen = $10^{6.4}$, quintiljoen = $10^{6.5}$.

miljard$_{Ned}$ = 10^9

We kunnen desgewenst ook nieuwe namen vormen door aaneenvoeging. Bijvoorbeeld, als we miljoen niet hadden gekend dan hadden we kunnen zeggen duizendduizend, en quintiljoenquintiljoen = $10^{6.5} * 10^{6.5} = 10^{60}$.

We analyseren nu hoe de structuur van een ~~getal-benaming~~ telwoord eruit ziet. In de volgende bespreking gebruiken we 0, 1, 2, ---, 99, H, D, M, B, T, Q, ... zowel voor de geruggereerde getalwaarden alsook voor de ~~namen~~ telwoorden ervan; bijvoorbeeld 12.345.678 wordt uitgesproken als 12 M 3 H 45 D 6 H 78. Definieer voorts

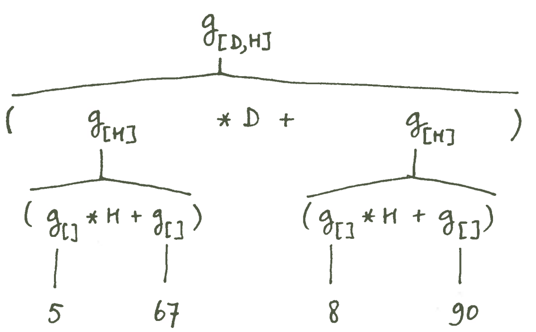
L_D = [.. Q, T, B, M, D, H] ;

We laten steeds L en t:L een staartstuk van deze lijst aanduiden (L = lijst, t = tienmacht). Het meest creatieve werk is voltooid met het opstellen van de volgende grammatica voor nederlandse ~~getalbenamingen~~ telwoorden. Hierin zijn alle g_L nonterminals en de symbolen), (, +, * zijn eigenlijk onzichtbaar – in de nederlandse tekst – maar geven wel de eenduidige outleding en betekenis goed weer.

(1) $g_{[\,]} ::= 0 \mid 1 \mid 2 \mid \ldots \mid 99$

(2) $g_{t:L} ::= (g'_L * t + g''_L)$

Hier is een voorbeeld van een productie en het geproduceerde getal (567.890).



$((5 * H + 67) * D + (8 * H + 90))$

5 H 67 D 8 H 90

Dat de outleding van ~~getalbenamingen~~ telwoorden zo eenvoudig is en/of ondubbelzinnig hangt mijns inziens samen met het volgende feit:

(3) in L_0 is elk ~~niet-laatste tienmacht~~ getal hoogstens het kwadraat van ~~de eropvolgende tienmacht~~ zijn rechterbuur:

$Q \leq T^2,\quad T \leq B^2,\quad B \leq M^2,\quad M = D^2,\quad D < H^2$

$Q < T^2,\ T < B^2,\ B < \text{Miljard}^2,\ \text{Miljard} \leq M^2,\ M = D^2,\ D < H^2$

Helaas is de grammatica nog niet korrekt: er worden nog on-nederlandse ~~getalbenamingen~~ telwoorden geproduceerd. Summanden gelijk aan 0 verdwijnen namelijk in de nederlandse uitspraak. Bijvoorbeeld, 500.091 wordt uitgesproken als 5 H D 91 en niet als 5 H 0 D 0 H 91. We corrigeren dit door de regel

(2) $g_{t:L} ::= (g'_L * t + g''_L)$

te vervangen door

(2) $g_{t:L} ::= (g'_L * t + g''_L)$ mits g'_L en $g''_L > 0$
$\quad\mid (g'_L * t)$ mits $g'_L > 0$
$\quad\mid g''_L$

Dat de grammatica nu slechts legale nederlandse ~~getalbenamingen~~ telwoorden produceert neem ik verder voor waar aan. (Aangezien het nederlands niet geformaliseerd is kan ik zoiets ook niet formeel bewijzen!)

Overigens genereert de grammatica misschien niet álle legale ~~benamingen~~ telwoorden. 3.456 wordt uitgesproken als 3 D 4 H 56 maar ook als 34 H 56 en in dit geval worden beide ook geproduceerd uit $g_{[H]}$; maar ~~als~~ MD wordt niet door de grammatica geproduceerd terwijl dat misschien wel goed nederlands is (en hetzelfde betekent als 1 D M). (Nota bene, 1 M 1 D is legaal en betekent waarschijnlijk iets anders dan 1 MD.) Maar gelukkig worden er wel voor ieder getal een ~~benaming~~ telwoord geproduceerd, getuige de volgende stelling.

<u>Stelling</u> (Volledigheid)

$g_{[]}$ genereert ~~benamingen~~ telwoorden voor alle getallen < 100,
$g_{t:L}$ genereert ~~benamingen~~ telwoorden voor alle getallen $< t^2$.

<u>bewijs</u>

Met volledige inductie naar de lengte van L in g_L.
<u>Geval</u> $L = []$: triviaal volgens regel (1).
<u>Geval</u> $L \neq []$ zeg $L = t:L'$. Zij g willekeurig $< t^2$. Er geldt $g = g' * t + g''$ met $g', g'' = g$ <u>div</u> t, g <u>rem</u> t.
Als nu $L' = []$ dus $t = 100$, dan zijn ~~volg~~ wegens $g < t^2$ zowel g' alsook g'' beide < 100, dus genereerbaar uit

$g_{[]}$ i.e. $g_{L'}$.
Als echter $L' \neq []$ zeg $L' = t':L''$, dan is volgens feit (3) $t \leq t'^2$ en dus zijn g' en g'' beide $< t'^2$ en dus per inductiehypothese genereerbaar uit $g_{L'}$.
In beide gevallen is dus $g = g' * t + g''$ met g' en g'' genereerbaar uit $g_{L'}$, dus volgens regel (2') is g genereerbaar uit $g_{t:L'}$ i.e. g_L.
(Einde bewijs)

Gebaseerd op bovenstaand bewijs is het nu eenvoudig een algoritme te geven dat voor willekeurige getal g een nederlandse ~~benaming~~ telwoord ervoor oplevert.

```
tot100 = ["nul", "één", ---, "negenennegentig"]
        -- er zijn talloze manieren om tot100 zonder
        -- al te veel schrijfwerk te definieren!
L0 = [[10^6.5, "quintiljoen"], [10^6.4, "quadriljoen"], ---, [10^2, "honderd"]]

nederlands g  =  g > (hd (hd L0))^2  ->  "erg groot"
                 ned L0 g
ned [] g  =  tot100 (g+1)                  -- lijst subscriptie! start bij 1.
ned ([t,nt]:L) g  =  g'=0  ->  ned L g''
                     g''=0 ->  ned L g' ++ nt
                     ned L g' ++ nt ++ ned L g''
                     where g', g'' = g div t, g rem t
```

In bovenstaand algoritme worden nog geen spaties --volgens onze afspraak-- voortgebracht. Dat kan wel als volgt.

(i) wijzig in de tweede clausule voor ned alle drie voorkomens van ++ in ++ "␣" ++ .

(ii) geef bij de hoofdaanroep van ned niet L_0 maar L_0-zonder-z'n-laatste-element mee, en wijzig de eerste clause voor ned in

```
ned [] g =
      g < 100  ->  tot100 (g+1)          -- zo als voorheen
      g' = 0   ->  n''
      g'' = 0  ->  n' ++ "honderd"
      n' ++ "honderd" ++ n''
      where  g', g'' = g div 100,  g rem 100
             n', n'' = tot100 (g'+1),  tot100 (g''+1)
```

of wijzig die clause in

```
ned [] g  =  tot1000 (g+1)
tot1000 = ["nul", "een", ... ,"negenhonderdnegenennegentig"]
```

We zien hierboven dat er voor de eerste duizend getallen een inruil mogelijk is voor opslagruimte tegen berekeningstijd en vice versa. Desgewenst kan dat ook voor tot100,

alhoewel me dat minder nuttig lijkt. We kunnen bijvoorbeeld tot100 definieren als een funktie (aan te roepen met argument g in plaats van g+1):

```
tot100  g  =
    g' = 0   ->  n''
    g'' = 0  ->  n'
    g' = 1   ->  ["elf", "twaalf", ..., "negentien"] g''
    n'' ++ "en" ++ n'
    where  g', g'' = g div 10,  g rem 10
           n', n'' = tientallen g', eentallen (g''+1)
           tientallen = ["tien", "twintig", ..., "negentig"]
           eentallen = ["nul", "een", ..., "negen"]
```

De onregelmatigheid van ~~getalbenamingen~~ telwoorden tot honderd wordt toch vrijwel op dezelfde manier behandeld als de regelmatigheid vanaf honderd!

We besluiten met een oefening voor de lezer: schrijf een algoritme dat voor ieder getal álle door de grammatica voortgebrachte ~~getalbenamingen~~ telwoorden oplevert (of nondeterminisch één ervan). Wenk: bewijs eerst dat $g_{[]}$ uitsluitend voor getallen < 100 ~~benamingen~~ telwoorden voortbrengt en $g_{t:L}$ uitsluitend voor getallen $< t^2$. Nog een voor de hand liggende oefening is de inverse van (ned L_0)

te schrijven, dwz een funktie getal zó dat

$$\text{getal}\ (\text{ned}\ L_0\ g) = g \qquad \text{voor alle}\ g < (\text{hd}\ (\text{hd}\ L_0))^2$$