Simo K. Kivelä, 15.4.2003

Luonnollisten lukujen laskutoimitusten määrittely Peanon aksioomien pohjalta

Aksioomat

Luonnolliset luvut voidaan määritellä Peanon aksioomien avulla. Tarkastelun kohteena on tällöin joukko, jolle annetaan nimi $\mathbb{N}$ ja jonka alkioilla oletetaan olevan seuraavat ominaisuudet:

1) Joukossa $\mathbb{N}$ on alkio u.

2) Jokaista joukon $\mathbb{N}$ alkiota p kohden on olemassa yksikäsitteinen alkio $s(p)$, jota sanotaan alkion p *seuraajaksi*.

3) Alkio u ei ole minkään alkion seuraaja.

4) Jos $s(p) = s(q)$, niin $p = q$.

5) Olkoon S on jokin joukon $\mathbb{N}$ osajoukko, jolla on ominaisuudet a) $u \in S$, b) jos $p \in S$, niin myös $s(p) \in S$. Tällöin $S = \mathbb{N}$.

Näitä viittä ominaisuutta sanotaan *Peanon aksioomiksi* ja joukkoa $\mathbb{N}$ *luonnollisten lukujen joukoksi*.

Aksioomien määrittämä jounon $\mathbb{N}$ rakenne tulee ymmärrettävämmäksi seuraavasti:

Alkio u saakoon nimekseen '1'. Aksiooman 2 mukaan tällä on seuraaja $s(u)$, joka saakoon nimekseen '2'. Tällä on edelleen seuraaja $s(s(u))$, jonka nimi olkoon '3'. Näin jatketaan. Kaikkiaan syntyy ketju joukon $\mathbb{N}$ alkioita, ts. eräs joukon $\mathbb{N}$ osajoukko. Tällä on aksioomassa 5 joukolta S vaaditut ominaisuudet, jolloin siis ketju on sama kuin $\mathbb{N}$. Joukossa $\mathbb{N}$ ei siis ole muita alkioita kuin u ja tämän seuraajat: u, $s(u)$, $s(s(u))$, $s(s(s(u)))$ jne.

Nimien antaminen luvuille

Seuraavassa on tarkoitus määritellä joukon $\mathbb{N}$ alkioille yhteen- ja kertolasku. Jotta alkioita voidaan myös käsitellä niiden nimien avulla, määritellään seuraavat Mathematican apufunktiot:

```
In[1]:= luku[nimi_String] := Nest[s, u, ToExpression[nimi] - 1]
```

```
In[2]:= nimi[p_s] := ToString[p /. {s → Function[x, x + 1], u → 1}];
        nimi[u] := ToString[1]
```

Näistä edellinen saa argumentikseen luvun nimen ja palauttaa luvun seuraajafunktion *s* avulla esitettynä. Jälkimmäinen on tämän käänteisfunktio. Esimerkiksi:

```
In[3]:= luku["7"]
Out[3]= s[s[s[s[s[s[u]]]]]]

In[4]:= nimi[s[s[s[s[u]]]]]
Out[4]= 5

In[5]:= nimi[luku["12"]]
Out[5]= 12

In[6]:= luku[nimi[s[s[u]]]]
Out[6]= s[s[u]]

In[7]:= nimi[u]
Out[7]= 1

In[8]:= luku["1"]
Out[8]= u
```

Aksioomista seuraa myös, että samalle alkiolle ei tässä anneta kahta eri nimeä: Jos nimet ovat erilaiset, kyseessä on myös kaksi eri alkiota.

Laskutoimitusten märittely

Luonnollisten lukujen yhteen- ja kertolasku on määriteltävä induktiivisesti, ts. ensin määritellään ne laskutoimitukset, joissa toisena argumenttina on *u*. Yleinen tapaus palautetaan tämän jälkeen rekursiivisesti edeltäjää koskevaan laskutoimitukseen. Tällöin aksioomasta 5 seuraa, että laskutoimitus tulee määritellyksi kaikille joukon ℕ alkioille.

Laskutoimitus on itse asiassa kahden muuttujan funktio. Esimerkiksi yhteenlasku saa kaksi argumenttia, yhteenlaskettavat, ja funktion arvona on näiden summa. Määriteltävien funktioden nimet olkoot `summa` ja `tulo`:

Yhteenlasku:

```
In[9]:= summa[p_, u] := s[p]
        summa[p_, s[q_]] := s[summa[p, q]]
```

Kertolasku:

```
In[11]:= tulo[p_, u] := p
         tulo[p_, s[n_]] := summa[tulo[p, n], p]
```

Esimerkkinä olkoon lukujen 7 ja 8 yhteen- ja kertolasku:

Nimien perusteella muodostetut luvut ovat

```
In[13]:= a = luku["7"]
Out[13]= s[s[s[s[s[s[u]]]]]]
```

```
In[14]:= b = luku["8"]
Out[14]= s[s[s[s[s[s[s[u]]]]]]]
```

Näiden summa on

```
In[15]:= c = summa[a, b]
Out[15]= s[s[s[s[s[s[s[s[s[s[s[s[s[s[u]]]]]]]]]]]]]]
```

ja sitä vastaava nimi

```
In[16]:= nimi[c]
Out[16]= 15
```

Lukujen tulo on

```
In[17]:= d = tulo[a, b]
Out[17]= s[s[
           s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[
                                                    s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[
                                                    u]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]
                                     ]]]]]]]]]]]]]]]]]]]
```

ja tämän luvun nimi

```
In[18]:= nimi[d]
Out[18]= 56
```

Erityisesti:

```
In[19]:= tulo[u, a] // nimi
Out[19]= 7
```

```
In[20]:= tulo[a, u] // nimi
Out[20]= 7
```

Myös monimutkaisempia lausekkeita voidaan laskea:

```
In[21]:= a = luku["2"]; b = luku["3"]; c = luku["4"]; d = luku["5"];
```

```
In[22]:= v1 = summa[tulo[a, summa[b, c]], tulo[d, tulo[a, a]]]
Out[22]= s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[
                                                s[s[s[u]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]
```

```
In[23]:= nimi[v1]
Out[23]= 34
```

Mathematastican rakenteessa olevat rajoitukset

Edellä olevalla tavalla määritellyissä laskutoimituksissa on syytä välttää liian suuria lukuja. Määritelmät ovat nimittäin rekursiivisia ja Mathematicalla on rajoitettu rekursioiden syvyys. Rajoitus ilmenee muuttujasta

```
In[24]:= $RecursionLimit

Out[24]= 300
```

Esimerkki epäonnistumisesta:

```
In[25]:= e = luku["260"];

In[26]:= v2 = summa[a, e]

Out[26]= s[s[
   s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[
                                                  s[s[s[s[s[s[s[s[s[s[s[
                                                  s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[
                                                  s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[
                                                  s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[
                                                  s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[
                                                  s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[
                                                  s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[
                                                  s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[
                                                  s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[
                                                  s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[
                                                  s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[
                                                  s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[
                                                  s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[
                                                  u]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]
                                                  ]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]
                                                  ]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]
                                                  ]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]
                                                  ]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]
                                                  ]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]
                                                  ]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]
     ]]]
```

In[27]:=
```
v3 = summa[e, a]
```

Out[27]=
```
s[s[
  s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[
                                  s[s[s[s[s[s[s[s[s[s[s[
                                  s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[
                                  s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[
                                  s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[
                                  s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[
                                  s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[
                                  s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[
                                  s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[
                                  s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[
                                  s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[
                                  s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[
                                  s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[
                                  s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[
                                  u]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]
                                  ]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]
                                  ]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]
                                  ]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]
                                  ]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]
                                  ]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]
                                  ]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]
    ]]]
```

In[28]:=
```
v2 // nimi
v3 // nimi
```

Out[28]= 262

Out[29]= 262

Esimerkin laskeminen epäonnistuu toisinpäin, mutta onnistuu toisinpäin, koska yhteenlaskun määritelmä ei ole symmetrinen (vaikka määritelmän tuloksena syntyvä yhteenlasku onkin). Rekursiorajaa voidaan muuttaa, minkä jälkeen edellä olevat laskut voi ajaa uudelleen:

In[30]:=
```
$RecursionLimit = 300
```

Out[30]= 300

Helpomman algebran rakentaminen määritelmien varaan

Funktioiden summa ja tulo kirjoittaminen moneen kertaan laskettavaan lausekkeeseen tekee lausekkeiden syöttämisen hankalaksi. Lukujen neliösumma esimerkiksi olisi

In[31]:=
```
v4 = summa[tulo[a, a], summa[tulo[b, b], summa[tulo[c, c], tulo[d, d]]]]
```

Out[31]=
```
s[s[
  s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[
                                 s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[
                                 u]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]
                 ]]]]]]]]]]]]]]]]]
```

In[32]:=
```
v4 // nimi
```

Out[32]=
```
54
```

Ottamalla käyttöön laskutoimituksille sopivat symbolit päästään hieman yksinkertaisempaan esitykseen. Symboli ⊕tarkoittaa funktiota nimeltään `CirclePlus` ja symboli ⊗ funktiota `CircleTimes`. Kun nämä asetetaan tarkoittamaan edellä määriteltyjä laskutoimituksia, voidaan lausekkeet syöttää niiden avulla:

In[33]:=
```
CirclePlus = summa
CircleTimes = tulo
```

Out[33]=
```
summa
```

Out[34]=
```
tulo
```

In[35]:=
```
v5 = a⊗a⊕(b⊗b⊕(c⊗c⊕d⊗d))
```

Out[35]=
```
s[s[
  s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[
                                 s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[
                                 u]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]
                 ]]]]]]]]]]]]]]]]]
```

In[36]:=
```
v5 // nimi
```

Out[36]=
```
54
```

Syötteestä on voitu jättää sulut pois kertolaskujen ympäriltä, koska Mathematicassa on sisäänrakennettuna operaatioiden suoritusjärjestys: ⊗ suoritetaan ensin, ⊕ sitten. Kaikkia sulkuja sen sijaan ei voida jätää pois, koska funktiota `summa` ei ole määritelty useammalle kuin kahdelle argumentille; sama koskee funktiota `tulo`.

In[37]:=
```
v6 = a⊗a⊕b⊗b⊕c⊗c⊕d⊗d
```

Out[37]=
```
summa[s[s[s[u]]], s[s[s[s[s[s[s[s[u]]]]]]]],
 s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[u]]]]]]]]]]]]]]], s[s[s[
   s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[u]]]]]]]]]]]]]]]]]]]]]]]]]
```

Tilanne voidaan korjata antamalla lisämääritelmät, joissa kolmen tai useamman argumentin summa ja tulo palautetaan kahden argumentin summaksi ja tuloksi. (Yhdellä alaviivalla merkitty symboli tarkoittaa mitä tahansa yhtä argumenttia, kahdella alaviivalla (pitempi alaviiva) merkitty vastaavasti yhtä tai useampaa argumenttia.)

In[38]:=
```
summa[x_, y_, z__] := summa[x, summa[y, z]]
tulo[x_, y_, z__] := tulo[x, tulo[y, z]]
```

Tämän jälkeen lasku onnistuu ilman sulkuja:

In[40]:=
```
v6 = a⊗a⊕b⊗b⊕c⊗c⊕d⊗d
```

Out[40]=
```
s[s[
  s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[
                                 s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[s[
                                 u]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]
                 ]]]]]]]]]]]]]]]]]
```

In[41]:=
```
v6 // nimi
```
Out[41]=
```
54
```

Harjoitustehtäviä

1) Luonnolliset luvut voidaan yhtä hyvin nimetä siten, että alkio *u* saa nimekseen '0', ts. pienin luonnollinen luku ei olekaan ykkönen vaan nolla. Tämä heijastuu kuitenkin laskutoimitusten määrittelyyn. Muunna määritelmät siten, että ne vastaavat tätä tilannetta.

2) Luonnollisille luvuille voidaan eräissä tapauksissa määritellä vähennyslasku: suuremmasta luvusta voidaan vähentää pienempi, mutta ei pienemmästä suurempaa eikä yhtä suuria lukuja toisistaan (jos käytössä on määritelmä, missä pienin luonnollinen luku on ykkönen). Määrittele vähennyslasku rekursiivisesti seuraajaesityksiin perustuen. Huomaa, että Mathematican aritmeettisia operaatioita ei saa käyttää, vaan määritelmän on perustuttava yksinomaan seuraajaesityksiin. Jos annettujen lukujen erotus ei ole määritelty, tulee tästä antaa ilmoitus.

3) Täydennä edellä kehitettyä notaatiota ($\oplus$, $\otimes$) potenssiinkorotussymbolilla valitsemalla tähän tarkoitukseen sopivan laskujärjestyksen antama symboli ja testaa määrittelysi toimivuus. (Ks. Mathematican dokumentaatiota: Operator Precedence.)