Tavoitteet ja menetelmät ASSET-92-järjestelmässä

Jorma Sajaniemi, Markku Tukiainen, Jarmo Väisänen

Joensuun yliopisto
Tietojenkäsittelyopin laitos
PL 111
80101 Joensuu

Tiivistelmä

ASSET-92 on rakenteinen taulukkolaskentajärjestelmä, jossa käyttäjän käsittelemät tiedot muodostuvat yksittäistä solua suuremmista kokonaisuuksista. Tässä raportissa esitetään ASSET-92-järjestelmän tapa käsitellä tavoitteita ja niihin liittyviä menetelmiä. ASSET-92-järjestelmässä tavoite kuvaa halutusta laskennan tuloksesta vain eräitä yleisiä ominaisuuksia, kuten lähtö- ja tulosrakenteiden tyypit. Sen sijaan itse laskennan suoritustapa kuvataan menetelmänä, joita voi olla kullakin tavoitteella useita. Tavoitteiden ominaisuuksista erikoisin on paloittuvuus, joka kuvaa kokonaisuuden jakamista osiin ja tämän vaikutusta laskentaan.

Sisällysluettelo

1. Johdanto

Taulukkolaskennan käyttäjät eivät miellä taulukkopohjassa olevia soluja yksittäisinä tietoalkioina, vaan muodostavat mielessään näistä suurempia loogisia kokonaisuuksia (Saariluoma & Sajaniemi, 1989). Rakenteinen taulukkolaskenta (Hassinen & al., 1988) eroaa perinteisestä taulukkolaskennasta siinä, että se antaa käyttäjille mahdollisuuden käsitellä tietoja tällaisina kokonaisuuksina. Siten rakenteinen taulukkolaskenta vastaa perinteistä taulukkolaskentaa paremmin käyttäjien mielikuvaa käsiteltävistä tiedoista.

Rakenteisessa taulukkolaskennassa kaikki laskenta kohdistetaan periaatteessa rakenteiden muodostamiin kokonaisuuksiin. Laskennallisia menetelmiä kuvattaessa voidaan tosin viitata yksittäisiin soluihin, mutta laskennallisten tavoitteiden kohteena ovat aina kokonaiset rakenteet.

ASSET-92 on rakenteinen taulukkolaskentajärjestelmä, jonka tarkoitus on havainnollistaa rakenteisen taulukkolaskennan mahdollisuuksia ja toimia erilaisten ideoiden testipenkkinä. Järjestelmästä ollaan laatimassa prototyyppiä käyttäen työvälineinä C++ -ohjelmointikieltä (Ellis & al., 1990) ja InterViews-käyttöliittymäkirjastoa (Linton & al., 1989).

ASSET-92-järjestelmän käyttäjän ei yleensä tarvitse kuvata laskennan suorittamista yksityiskohtaisesti, koska järjestelmä sisältää joukon valmiiksi määriteltyjä laskennallisia tavoitteita (esimerkiksi kumuloituva summa) (Tukiainen & al., 1993), joita käyttäjä voi suoraan soveltaa haluamiinsa rakenteisiin. Käyttäjä voi myös kuvata omia tavoitteitaan, jolloin niille on annettava myös vähintään yksi menetelmä laskennan suorittamiseksi.

Tässä raportissa tarkastellaan ASSET-92-järjestelmän tapaa esittää tavoitteet ja menetelmät. Jatkossa ei käsitellä käyttöliittymän yksityiskohtia eikä lausekkeiden ja muiden syntaktisten olioiden yksityiskohtaista syntaksia niiltä osin kuin syntaksi on B-ASSET-92-kielen mukaista. Tämän kielen yksityiskohtainen kuvaus löytyy raportista Sajaniemi & al. (1992).

Ennen tavoitteiden ja menetelmien käsittelyä luodaan lyhyt katsaus ASSET-92-järjestelmän tuntemiin rakennetyyppeihin.

2. ASSET-92-järjestelmän tuntemat rakenteet

ASSET-92-järjestelmän käsittelemät rakennetyypit ovat kenno, jono ja taulu. Nämä kaikki koostuvat soluista. Kunkin solun arvo voi olla kokonaisluku, reaaliluku, merkkijono tai totuusarvo, joita kutsutaan solun kannalta skalaarityypeiksi ja rakenteen kannalta solutyypeiksi. Yhdessä rakenteessa olevissa soluissa on kaikissa samantyyppinen arvo.

Kukin rakenne on kokonaisuudessaan joko laskennan tulos tai sisältää soluja, joihin käyttäjä voi syöttää haluamiaan arvoja. Jälkimmäisiä rakenteita kutsutaan syöttörakenteiksi.

Rakenteista yksinkertaisin on kenno, joka koostuu yhdestä ainoasta solusta. Kennolla, kuten kaikilla rakenteilla, on nimi. Kuvassa 1 on esimerkki kennosta.

Jono on yksiulotteinen kokoelma soluja. Jonon ulottuvuudella on nimi (esimerkiksi *Kuu*), jota ei kuitenkaan käytetä laskennassa hyväksi. Jonon alkioihin viitataan indekseillä, joita on kahta lajia. Absoluuttiset indeksit ovat kokonaislukuja ykkösestä jonon sisältämien solujen lukumäärään saakka. Mikäli soluja on vain yksi, kutsutaan ulottuvuutta triviaaliksi. Kutakin absoluuttista indeksiä vastaa todellinen indeksi (esimerkiksi absoluuttista indeksiä 2 voi vastata todellinen indeksi *Helmikuu*). Todellinen indeksi voi olla tyhjä ja lisäksi todellisten indeksien ei tarvitse olla yksikäsitteisiä edes ulottuvuuden sisällä. Käyttäjälle näytetään normaalisti todelliset indeksit. Laskennassa voidaan käyttää hyväksi sekä absoluuttisia että todellisia indeksejä. Kuvassa 1 on esimerkki jonosta.

Laskennan kannalta ajateltuna jonoilla ei ole suuntaa. Näytöllä kukin jono esitetään joko vaakajonona tai pystyjonona. Käyttäjä voi muuttaa jonon suuntaa halutessaan. Suunnan muuttamisella ei ole mitään vaikutusta muiden jonojen suuntiin tai muiden arvojen laskemiseen.

Taulu on kaksiulotteinen kokoelma soluja. Taulun ulottuvuuksilla on todelliset nimet (esimerkiksi *Tuote* ja *Vuosi*), joiden rinnalla voidaan käyttää ulottuvuuksien universaaleja nimiä *row* ja *col*. Taulun alkioihin viitataan käyttämällä rivi- ja sarakeindeksejä, jotka voivat olla jonojen indeksien tapaan absoluuttisia tai todellisia. Kuvassa 1 on esimerkki taulusta.

Käyttäjä voi halutessaan paloitella jonoja ja tauluja. Paloittelu tarkoittaa ulottuvuuden jakamista yhtenäisiin alueisiin. Jos jonon absoluuttiset indeksit ovat 1..15, niin tämä voidaan paloitella esimerkiksi osiin 1..8, 9..10, 11..11 ja 12..15. Taulu voidaan paloitella joko vain toisen ulottuvuutensa suhteen tai molempien ulottuvuuksiensa suhteen. Kuvassa 2 on esimerkki paloitellusta taulusta.

LvvProsentti
22

Vihannesmyynti											
Kuu											
Tammi	Helmi	Maalis	Huhti	Touko	Kesä	Heinä	Elo	Syys	Loka	Marras	Joulu
11	12	13	14	15	16	17	18	19	20	21	22

Myynti					
		Vuosineljännes			
		1.	2.	3.	4.
Tuote	Ruuvi	101	102	103	104
	Mutteri	105	106	107	108
	Naula	109	110	111	112
	Vasara	113	114	115	116
	Hohtimet	117	118	119	120
	Saha	121	122	123	124

Kuva 1: Kenno, jono ja taulu

Myynti			
		Vuosineljännes	
		1.	2.
Tuote	Ruuvi	101	102
	Mutteri	105	106
	Naula	109	110
	Vasara	113	114
	Hohtimet	117	118
	Saha	121	122

Myynti			
		Vuosineljännes	
		3.	4.
Tuote	Ruuvi	103	104
	Mutteri	107	108
	Naula	111	112
	Vasara	115	116
	Hohtimet	119	120
	Saha	123	124

Kuva 2: Paloiteltu taulu

Paloittelulla on kaksi tarkoitusta. Ensinnäkin se antaa mahdollisuuden siirtää rakenteen eri palat eri kohtiin näyttöä. Tämä palojen siirtäminen ei vaikuta millään tavalla laskentaan, vaan laskenta käsittelee aina koko rakennetta. Toiseksikin paloittelu voi vaikuttaa laskentaan tavalla, jota käsitellään tarkemmin seuraavassa luvussa. On huomattava, että paloittelun mahdollinen vaikutus laskentaan on täysin riippumaton siitä, siirretäänkö paloja erilleen toisistaan vai ei.

3. Tavoitteiden esittäminen

ASSET-92-järjestelmässä tavoitteen kuvaus sisältää vain tiedon tavoitteen yleisistä ominaisuuksista. Näitä ovat muun muassa argumenttien ja tulosrakenteiden rakennetyypit. Sen sijaan itse laskentamenetelmä - se kuinka tavoitteen tarkoittama tulos todella lasketaan - ei kuulu tavoitteen kuvaukseen. Siten tavoite tulee laskennallisesti määritellyksi vasta sille annettavan menetelmän kautta.

Tavoitteen kuvauksessa on seuraavat osat:

- tavoitteen nimi (goal)
- argumenttien rakennetyypit ja nimet (arguments)
- argumenteille asetettavat vaatimukset (argument requirements)
- tulosrakenteen rakennetyyppi ja nimi (result)
- ulottuvuuksien periytyvyys (dimension)
- laskennan paloittaisuus (split usage method)

Kullakin tavoitteella on yksikäsitteinen nimi, joka on muodoltaan B-ASSET-92-kielen tunnus.

Argumentit ovat muodollisten parametrien asemassa olevia rakenteiden nimiä, jotka sidotaan tavoitetta sovellettaessa todellisiin, sovelluksessa jo olemassaoleviin rakenteisiin. Laskennassa voidaan viitata argumenttien lisäksi suoraan todellisiin rakenteisiin, joten laskenta ei perustu pelkästään argumentteina saatavien rakenteiden sisältämiin tietoihin. Kustakin argumentista kerrotaan sen rakennetyyppi (kenno, jono, taulu) ja muodollinen, laskennassa käytettävä nimi, mutta sen sijaan ei kerrota argumentin solutyyppiä (kokonaisluku, reaaliluku, merkkijono, totuusarvo). B-ASSET-92-kieli antaa lausekkeiden arvoksi *ERR*, mikäli tavoitetta sovellettaessa tietojen tyypit eivät ole sopivia.

Argumenteille asetettavat vaatimukset kuvaavat yleensä argumenttien keskinäisiä kokosuhteita. Jos argumentteina on kaksi jonoa, niin voidaan esimerkiksi vaatia, että niissä tulee olla yhtä monta solua. Argumenteille asetettava vaatimus on muodoltaan B-ASSET-92-kielen totuusarvon palauttava skalaarilauseke. Siinä voidaan viitata argumenttien lisäksi todellisiin rakenteisiin. Jos jokin viitattu todellinen rakenne ei ole olemassa tavoitetta sovellettaessa, on tavoitteen arvo kenno, jossa on *ERR*.

Tulosrakenteen rakennetyyppi ilmoittaa tavoitteen soveltamisen seurauksena syntyvän rakenteen rakennetyypin. Tulosrakenteen nimi on luonteeltaan muodollinen argumentti ja sitä voidaan tarvittaessa käyttää laskennan antavan menetelmän kuvauksessa. Tavoitteen tuloksena syntyvälle todelliselle rakenteelle antaa järjestelmä aina uuden yksikäsitteisen nimen, jonka käyttäjä voi muuttaa haluamakseen.

Ulottuvuuksien periytyvyys ilmoittaa ensinnäkin sen, miltä argumenteilta tulosrakenne perii todelliset indeksinsä ja ulottuvuuksien nimet (inherited from) sekä sen salliiko tavoite paloittelun periytyvän argumenteista (splitting inherited).

Jos tulosrakenne on kenno, ei ulottuvuuksien periytymistä ilmoiteta, koska sillä ei ole mitään merkitystä. Jonon tapauksessa ilmoitetaan yksi ulottuvuus ja taulun tapuksessa ilmoitetaan riviulottuvuus (row dimension) ja sarakeulottuvuus (col dimension) erikseen.

Todellisten indeksien ja ulottuvuuden nimien mahdollinen periytyminen ilmoitetaan antamalla se argumentti, jolta tiedot periytyvät. Jos argumentti on taulu, on lisäksi mainittava kumpaa sen ulottuvuuksista tarkoitetaan. Siten indeksien ja ulottuvuuksien nimien periytyminen ilmoitetaan jollain seuraavista tavoista: jonoargumentti, tauluargumentti.row tai tauluargumentti.col.

Paloittelun periytyminen ilmoittaa peritäänkö argumentilta indeksien ja ulottuvuuden nimien lisäksi paloittelu. Vaihtoehtoja on kaksi: joko paloittelu peritään (yes) tai ei peritä (no).

Tavoitetta soveltaessaan käyttäjä voi vielä muuttaa ulottuvuuksien periytymistä joko vaihtamalla rakennetta ja ulottuvuutta, josta periytyminen tehdään, tai ilmoittamalla, että periytymistä ei tietyn ulottuvuuden suhteen käytetä. Lisäksi käyttäjä voi muuttaa paloittelun periytymisominaisuutta.

Vihannesmyynti											
Kuu											
Tammi	Helmi	Maalis	Huhti	Touko	Kesä	Heinä	Elo	Syys	Loka	Marras	Joulu
11	12	13	14	15	16	17	18	19	20	21	22

Kumulointi koko vuodelle											
Kuu											
Tammi	Helmi	Maalis	Huhti	Touko	Kesä	Heinä	Elo	Syys	Loka	Marras	Joulu
11	23	36	50	65	81	98	116	135	155	176	198

Kumulointi puolivuosittain											
Kuu											
Tammi	Helmi	Maalis	Huhti	Touko	Kesä	Heinä	Elo	Syys	Loka	Marras	Joulu
11	23	36	50	65	81	17	35	54	74	95	117

Kuva 3: Kumuloituva summa jonosta ja sen paloista

Laskennan paloittaisuus kuvaa sen suoritetaanko laskenta koko argumenttirakenteille vai erikseen niiden paloille. Esimerkiksi kumuloituva summa voidaan kohdistaa lähdejonon paloihin, jolloin tuloksessa kumulointi aloitetaan aina alusta jokaisen paloittelukohdan kohdalla. Tästä on esimerkki kuvassa 3, jossa toinen tulosrakenne on saatu laskemalla kumuloitava summa koko rakenteesta ja toinen on saatu laskemalla kumuloituva summa paloista.

Kohdistettaessa laskenta paloihin voidaan tulosta laskettaessa joutua käyttämään joidenkin argumenttien ja välitulosten paloja useaan kertaan. Esimerkiksi kuvassa 4 tuloksena on paloihin kohdistetut painotetut rivisummat. Tässä joudutaan samoja painokertoimia soveltamaan useasti eri lähtölukujen kanssa. Esimerkiksi painokertoimien ylempää palaa käytetään lähtölukujen vasemmanpuoleisen yläpalan, keskimmäisen yläpalan ja oikeanpuoleisen yläpalan kanssa.

Laskennan paloittaisuus ilmoittaa sen, millä tavalla argumenttien paloja sovelletaan toistensa kanssa. Vaihtoehtoina tulevat kyseeseen esimerkiksi seuraavat (Sajaniemi & al., 1993):

- paloittelu ei vaikuta laskentaan lainkaan (none)
- argumenteilla tulee olla yhtä monta palaa kussakin ulottuvuudessa ja kutakin palaa käytetään kerran yhdessä muissa argumenteissa olevien vastinpalojensa kanssa (simple)
- edelläolevan esimerkin tapaus (each column group by column)

Lähtöluvut					
		Tehdas			
Tuote	Muki	10	20	30	40
	Kippo	11	21	31	41
	Kannu	12	22	32	42
	Patsas	13	23	33	43

Kertoimet		
Tuote	Muki	4
	Kippo	3
	Kannu	2
	Patsas	1

Tulos				
		Tehdasryhmä		
Tuote	Muki	120	120	160
	Kippo	96	93	123
	Kannu	68	64	84
	Patsas	36	33	43

Kuva 4: Paloittain lasketut painotetut rivisummat; tulos on taulu

Mikäli laskenta on paloittaista, voi tulosrakenteen rakennetyyppi poiketa tavoitteen kuvauksessa mainitusta rakennetyypistä. Voidaankin ajatella, että tulosrakenteen rakennetyyppi ilmoittaa yksittäisen tulospalasen rakennetyypin. Edellisessä esimerkissä olleen tavoitteen "painotetut rivisummat" tulos on varsinaisesti rakennetyypiltään jono. Tämä käy selvästi ilmi kuvasta 5, jossa lähtölukujen sarakeulottuvuuden paloittelu on poistettu. Rakenteen paloittelussa tapahtuvat muutokset voivat siis muuttaa tulosrakenteen rakennetyyppiä.

Paloihin kohdistuva laskenta ei ole aina mahdollista ja lisäksi se tuottaa toisinaan arvaamattomia tuloksia. Tällaisissa tapauksissa laskenta antaa tulosrakenteen kaikille soluille arvoksi *ERR*.

Paloihin kohdistuvan laskennan yksityiskohtia ja mahdollisia vaihtoehtoja käsitellään tarkemmin raportissa Sajaniemi & al. (1993).

Esimerkkinä tavoitteen kuvauksesta olkoon kumuloituva summa, jonka laskenta kohdistetaan lähdejonoon kokonaisuudessaan.

```
Goal:                      KumuloituvaKokonaisSumma
Arguments:                 sequence Jono
Argument requirements:     -
Result:                    sequence s
Dimension
  inherited from:          Jono
  splitting inherited:     yes
Split usage method:        none
```

Lähtöluvut					
		Tehdas			
Tuote	Muki	10	20	30	40
	Kippo	11	21	31	41
	Kannu	12	22	32	42
	Patsas	13	23	33	43

Kertoimet		
Tuote	Muki	4
	Kippo	3
	Kannu	2
	Patsas	1

Tulos		
Tuote	Muki	400
	Kippo	312
	Kannu	216
	Patsas	112

Kuva 5: Paloittain lasketut painotetut rivisummat; tulos on jono

Tavoitteen nimessä esiintyy sana "kokonais" kuvaamassa sitä, että laskenta tehdään koko rakenteelle eikä sen osille. Vastaavan paloittaisesti laskettavan tavoitteen nimenä käytetään siten muotoa "KumuloituvaSumma".

Muina esimerkkeinä olkoot TauluSumma, joka laskee taulun (palasen) summan ja Paikka-kohtainenTauluEro, joka laskee kahdesta taulusta niiden vastinsolujen erotukset.

```
Goal:                   TauluSumma
Arguments:              table Taulu
Argument requirements: -
Result:                 chamber c
Split usage method:     simple

Goal:                   PaikkakohtainenTauluEro
Arguments:              table Vahennettava, Vahentaja
Argument requirements: Vahennettava.row# = Vahentaja.row# and
                        Vahennettava.col# = Vahentaja.col#
Result:                 table t
Row dimension
  inherited from:       Vahennettava.row
  splitting inherited: yes
Col dimension
  inherited from:       Vahennettava.col
  splitting inherited: yes
Split usage method:     none
```

4. Menetelmien esittäminen

ASSET-92-järjestelmässä jokainen menetelmä liittyy johonkin olemassaolevaan tavoitteeseen. Siten menetelmää ei voida kuvata ellei vastaava tavoite ole jo olemassa. Tavoitetta kuvattaessa ei sille ole tietenkään vielä yhtään menetelmää. Tavoitteen soveltaminen edellyttää vähintään yhden menetelmän olemassaoloa.

Samaan tavoitteeseen liittyvien eri menetelmien on syytä laskea täsmälleen sama lopputulos. Järjestelmä ei kuitenkaan tarkasta menetelmien yhtäpitävyyttä.

Menetelmän kuvauksessa on seuraavat osat:

- vastaavan tavoitteen nimi (plan for goal)
- menetelmän nimi (plan)
- tarvittavat alitavoitteet (required subgoals)
- tuloslauseke (expression)

Menetelmien nimet, jotka ovat muodoltaan B-ASSET-92-kielen tunnuksia ovat yksikäsitteisiä tavoitteen sisällä. Kunkin tavoitteen yhdellä menetelmällä voi olla tyhjä nimi. Tätä menetelmää käytetään oletusarvoisesti tavoitetta sovellettaessa.

Tarvittavissa alitavoitteissa kuvataan joukko välituloksia, joiden avulla varsinainen laskenta tehdään. Kukin välitulos ilmoitetaan seuraavassa muodossa:

välitulosrakenne=tavoite(rakenne1,...)

Välitulosrakenteiden nimet ovat muodoltaan B-ASSET-92-kielen tunnuksia ja ne eivät saa olla samoja kuin argumenttien nimet. Kuvauksissa esiintyvien rakenteiden nimien tulee olla argumentteja, tämän saman menetelmän aiemmin kuvattuja välitulosrakenteita tai todellisia rakenteita.

Alitavoitteissa lueteltujen tavoitteiden ei tarvitse välttämättä olla olemassa menetelmän kuvaamisen aikana. Sen sijaan menetelmää sovellettaessa alitavoitteiden tulee olla olemassa ja niille tulee kullekin olla olemassa vähintään yksi menetelmä.

Tuloslauseke kuvaa menetelmän sisäisen laskennan ja on muodoltaan normaalisti B-ASSET-92-kielen menetelmälauseke, jossa ei saa esiintyä tavoitelausekkeita. Tuloslausekkeessa voidaan viitata argumentteihin, välitulosrakenteisiin, todellisiin rakenteisiin ja tulosrakenteeseen. Tulokseksi voidaan kirjoittaa myös yhdestä tavoitteesta muodostuva tavoitelauseke, kunhan sen argumentteina on pelkästään kuvattavan tavoitteen argumentteja, välitulosrakenteita ja todellisia rakenteita.

Alitavoitteissa ja tuloslausekkeessa mainittujen todellisten rakenteiden ei tarvitse olla olemassa sen paremmin menetelmää määriteltäessä kuin sovellettaessakaan. Jos alitavoite tai tuloslauseke viittaa olemassaolemattomaan tai väärää rakennetyyppiä olevaan rakenteeseen, on menetelmän arvo kenno, jossa on *ERR*.

Menetelmän kuvauksen esimerkkinä on seuraavassa oletusmenetelmät edellisessä luvussa kuvatuille tavoitteille.

```
Plan for goal:          KumuloituvaKokonaisSumma
Plan:
Required subgoals:      -
Expression:             sum(Jono[1..i]); i=1..Jono#

Plan for goal:          TauluSumma
Plan:
Required subgoals:      -
Expression:             sum(Taulu)

Plan for goal:          PaikkakohtainenTauluEro
Plan:
Required subgoals:      -
Expression:             Vahennettava[i,j] - Vahentaja[i,j];
                          i=1..Vahentaja.row#, j=1..Vahentaja.col#
```

5. Tavoitteiden ja menetelmien yhdistäminen: esimerkki

Esimerkkinä tavoitteiden ja menetelmien käytöstä tarkastellaan kumuloituvaa erotusta. Siinä kahdesta jonosta halutaan laskea vastinsolujen erotuksen yhteiskertymän kehittyminen jonojen alusta niiden loppuun.

Tämä tavoite voidaan laskea (ainakin) kahdella eri tavalla. Nämä tavat tarvitsevat alitavoitteita, joista ensimmäinen on paloittaisesti laskettava kumuloituva summa.

```
Goal:                   KumuloituvaSumma
Arguments:              sequence Jono
Argument requirements: -
Result:                 sequence s
Dimension
inherited from:         Jono
  splitting inherited: yes
Split usage method:     simple

Plan for goal:          KumuloituvaSumma
Plan:
Required subgoals:      -
Expression:             sum(jono[1..i]); i=1..Jono#
```

Toinen tavoite, jota tullaan käyttämään alitavoitteena on kahden jono vastinsolujen väliset erotukset. Laskennan tulos on sama tehtiinpä se paloittaisesti tai ei.

```
Goal:                   PaikkakohtainenJonoEro
Arguments:              sequence Vahennettava, Vahentaja
Argument requirements: Vahennettava# = Vahentaja#
Result:                 sequence s
Dimension
  inherited from:       Vahennettava
  splitting inherited: yes
Split usage method:     none

Plan for goal:          PaikkakohtainenJonoEro
Plan:
Required subgoals:      -
Expression:             Vahennettava[i] - Vahentaja[i];
                          i=1..Vahentaja#
```

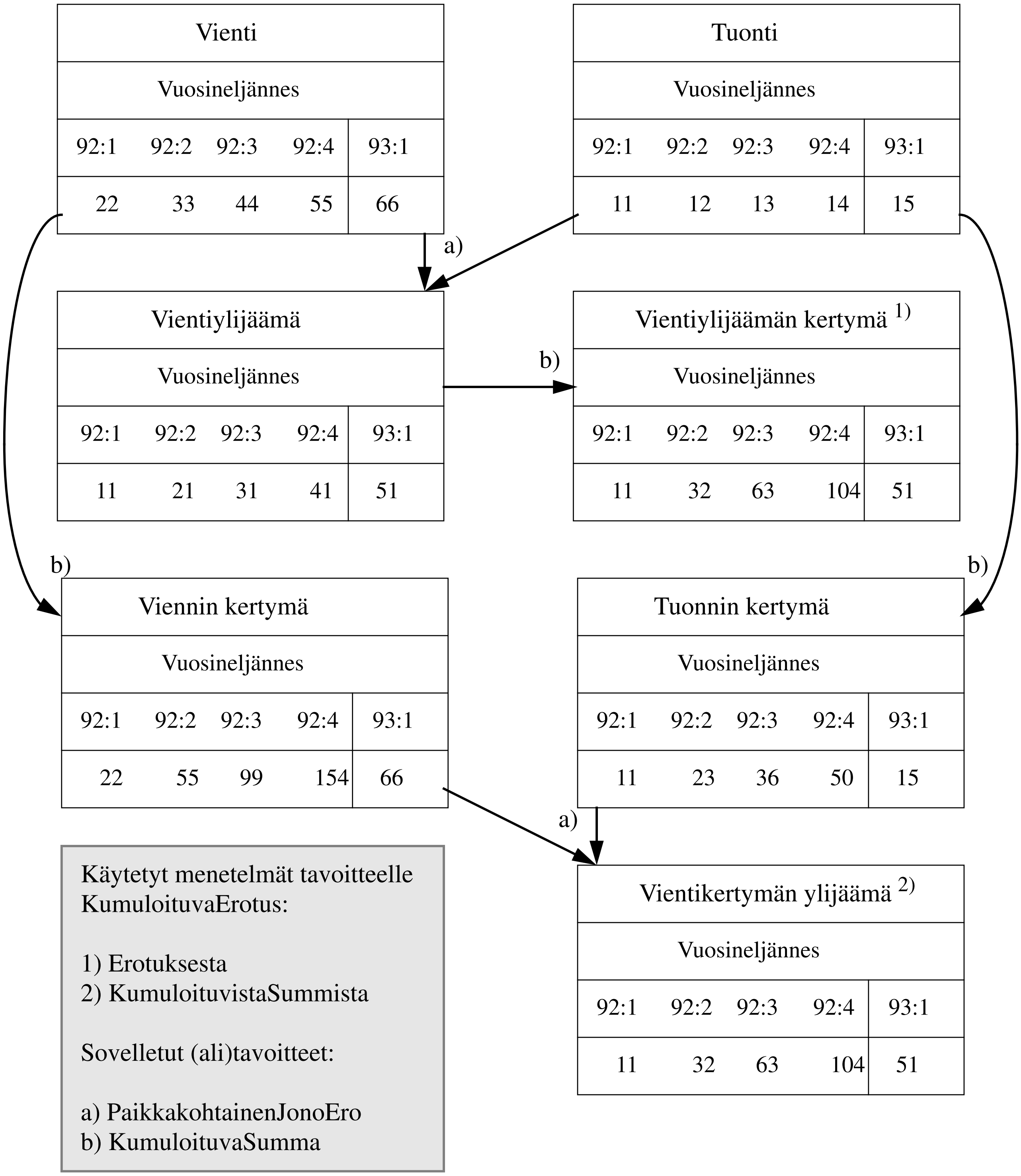
Kumuloituva erotus voidaan nyt laskea ottamalla lähtöjonoista ensin paikkakohtainen erotus ja laskemalla siitä kumuloituva summa. Toisaalta täsmälleen sama tulos saadaan ottamalla ensin lähtöjonoista kumuloituvat summat ja laskemalla sitten niiden paikkakohtaiset erotukset.

```
Goal:                   KumuloituvaErotus
Arguments:              sequence Vahennettava, Vahentaja
Argument requirements: Vahennettava# = Vahentaja#
Output:                 sequence s
Dimension
  inherited from:       Vahennettava
  splitting inherited: yes
Split usage method:     simple

Plan for goal:          KumuloituvaErotus
Plan:                   Erotuksesta
Required subgoals:      sequence a = PaikkakohtainenJonoEro(
                          Vahennettava,Vahentaja)
Expression:             sequence s = KumuloituvaSumma(a)
```

```
Plan for goal:          KumuloituvaErotus
Plan:                   KumuloituvistaSummista
Required subgoals:      sequence a1 = KumuloituvaSumma(Vahennettava),
                        sequence a2 = KumuloituvaSumma(Vahentaja)
Expression:             PaikkakohtainenJonoEro(a1,a2)
```

Esimerkki näiden tavoitteiden soveltamisesta on kuvassa 6.



Kuva 6: Tavoitteen laskenta kahdella eri menetelmällä

6. Tavoitteen soveltaminen

Tavoitetta sovellettaessa käyttäjän on annettava tavoitteen nimen lisäksi argumentteina toimivat todelliset rakenteet. Todellisten rakenteiden tulee periaatteessa olla tavoitteen kuvauksessa mainittua rakennetyyppiä. Jonoa vastaavaksi todelliseksi rakenteeksi voidaan kuitenkin antaa kenno, jolloin kenno tulkitaan jonoksi, jolla on triviaali ulottuvuus. Kenno voidaan tulkita myös tauluksi, jonka molemmat ulottuvuudet ovat triviaaleja. Lisäksi jono voidaan tulkita tauluksi, jonka toinen ulottuvuus on jonon ulottuvuus ja toinen ulottuvuus on triviaali. Sen, käytetäänkö jonon ulottuvuutta taulun rivi- vai sarakeulottuvuutena, määrää käyttäjä tavoitetta soveltaessaan.

Jos argumenttiin sidotun todellisen rakenteen rakennetyyppi ei ole edellä kuvatun tulkinnankaan jälkeen sopiva, niin tavoitteen soveltaminen onnistuu, mutta sen tuloksena on kenno, jonka solun arvona on *ERR*. Tavoitteen soveltamista pidetään sallittuna, koska rakenteiden nimiä voidaan myöhemmin muuttaa ja siten samanniminen rakenne voi olla eri ajanhetkinä eri rakennetyyppiä.

ASSET-92-järjestelmää voidaan käyttää kahdella tasolla. Aloittelijatasolla toimittaessa järjestelmä pyrkii tekemään ratkaisuja käyttäjän puolesta. Asiantuntijatasolla käyttäjälle annetaan puolestaan suurempi vapaus suorittaa valintoja. Tavoitteelle käytettävän menetelmän valinnassa tämä tulee esiin siten, että mikäli tavoitteella on useita menetelmiä ja käyttäjä toimii aloittelijatasolla, valitsee järjestelmä käytettävän menetelmän. Jos tavoitteella on nimetön menetelmä, valitsee järjestelmän sen, mutta muuten järjestelmä voi valita minkä menetelmän tahansa. Jos käyttäjä taas toimii asiantuntijatasolla, antaa järjestelmä käyttäjän valita käytettävän menetelmän.

Tavoitteen soveltamiseen tarvittavien seikkojen tultua valmiiksi käyttäjä voi vielä muuttaa omia ja järjestelmän tekemiä valintoja sekä tavoitteen muutettavissa olevia ominaisuuksia. Tässä vaiheessa käyttäjä voi muuttaa todellisia argumentteja sekä ulottuvuuksien ja paloittelun periytymistä.

Tulosrakenteen muodostamiseksi järjestelmä huolehtii seuraavista toimenpiteistä:

- argumenttien rakennetyyppien tarkastaminen
- argumenteille asetettujen vaatimusten tarkastaminen
- alitavoitteiden muodostaminen
- tulosrakenteen koon määrääminen
- ulottuvuuksien periyttäminen
- tulosrakenteen paloittelu
- tulosrakenteen solujen arvottaminen

Mikäli argumentit eivät ole vaadittuja rakennetyyppejä (aiemmin mainituin poikkeuksin) tai argumenteille asetetut vaatimukset kuvaavan lausekkeen arvo on *false*, tulee tulosrakenteen arvoksi kenno, jossa on *ERR*.

Laskennassa tarvittavat alitavoitteet kuvaavat välituloksia sisältävät rakenteet, jotka on oltava olemassa laskennan onnistumiseksi. Mikäli jokin välitulos ei ole jo olemassa, muodostaa järjestelmä sen samalla tavalla kuin normaalin tavoitteen soveltamisen yhteydessä. Siten asiantuntijatasolla toimittaessa järjestelmä voi joutua pyytämään käyttäjää valitsemaan menetelmiä alitavoitteille ja pyytämään hyväksyntää tehdyille valinnoille. Aloittelijatasolla toimittaessa ei eri alitavoitteisiin liittyviä hyväksymispyyntöjä tehdä.

Mikäli alitavoitteen mukainen välitulos muodostetaan tavoitteen soveltamisen yhteydessä, ei välitulosrakennetta näytetä käyttäjälle vaan se jää piiloon. Tällaista rakennetta kutsutaan verhotuksi. Käyttäjällä on kuitenkin mahdollisuus saada verhotut rakenteet näkyviin.

Tulosrakenteen koko määrätään menetelmän tuloslausekkeen ja laskennan paloittaisuus-ominaisuuden avulla.

Ulottuvuuksien ja indeksien nimet peritään niiltä rakenteilta, jotka on mainittu tavoitteen kuvauksessa ellei käyttäjä ole tätä muuttanut. Indeksi periytyy vain silloin, kun tulosrakenteen ulottuvuuden koko on sama kuin perittävällä ulottuvuudella. Muuten indeksi ei periydy, mutta tilanne ei kuitenkaan ole virhe.

Tulosrakenteen paloittelu määräytyy seuraavasti. Ensinnäkin tulosrakenne perii paloittelun samoin kuin ulottuvuudet. Mikäli tavoitteen laskenta on paloittaista, paloitellaan tulosrakenne toisaalta erillisinä laskettujen tulospalojen mukaan. Jos nämä kaksi paloitteluperiaateetta ovat molemmat käytössä ja niiden antamat paloittelut eroavat toisistaan, on tulosrakenteen kaikkien solujen arvo *ERR*.

Tulosrakenteen soluille saadaan arvot tuloslausekkeen avulla.

Edellä kuvatut tulosrakenteen muodostamisen vaiheet on tehtävä nimenomaan kuvatussa järjestyksessä. Erityisesti ulottuvuuksien periyttäminen edellyttää, että tulosrakenteen koko on tiedossa. Tulosrakenteen koko ja indeksit voivat puolestaan riippua välituloksista.

7. Rakenteissa tapahtuvat muutokset

Muissa rakenteisssa tapahtuvat muutokset voivat aiheuttaa muutoksia tavoitteen tulosrakenteessa. Tulosrakenteen muodostamisen yhteydessä tehdyt toimenpiteet voidaan jakaa kahteen luokaan: kerran tehtäviin ja jatkuvasti tehtäviin.

Seuraavia toimenpiteitä ei toisteta automaattisesti enää tavoitteen alkuperäisen soveltamisen jälkeen (vaikka käyttäjä voi halutessaan pyytää näiden tekemistä uudelleen):

- menetelmän valinta
- käyttäjän muutettavissa olevien ominaisuuksien hyväksyttäminen
- argumenttien sitominen todellisiin rakenteisiin
- alitavoitteiden mukaisten välitulosten muodostaminen
- välitulosten nimien sitominen todellisiin rakenteisiin

Argumenttien ja välitulosten sitominen todellisiin rakenteisiin perustuu järjestelmän käyttämään rakenteiden sisäiseen nimentään. Vaikka käyttäjä muuttaisi rakenteiden nimiä, ei sillä ole vaikutusta näihin sidontoihin.

Seuraavat toimenpiteet tehdään periaatteessa jokaisen missä tahansa rakenteessa tapahtuneen muutoksen jälkeen:

- argumenttien rakennetyyppien tarkastaminen
- argumenteille asetettujen vaatimusten tarkastaminen
- tulosrakenteen koon määrääminen
- ulottuvuuksien periyttäminen
- tulosrakenteen paloittelu
- tulosrakenteen solujen arvottaminen

Käytännössä ei tietenkään tarvitse tehdä kaikkia ylläkuvattuja toimenpiteitä jokaisen muutoksen jälkeen. Esimerkiksi tulosrakenteen koon uudelleenmääräämistä ei tarvitse tehdä, mikäli muutos ei kohdistu niiden argumenttien, välitulosrakenteiden tai todellisten rakenteiden tietoihin, joihin viitataan tuloslausekkeessa.

Uudelleenlaskennan aiheuttavia muutoksia ovat rakenteiden sisältämien solujen arvojen muuttumisen lisäksi ulottuvuuksien ja indeksien todellisten nimien muuttuminen, rakenteiden rakennetyypin tai koon muuttuminen, paloittelun muuttuminen, rakenteiden poistaminen ja lisääminen sekä rakenteiden nimien muuttaminen. Rakenteiden nimien muuttamisella on tosin merkitystä vain silloin, kun rakenteeseen viitataan jossain lausekkeessa suoraan todellisella nimellä (eikä argumentti- tai välitulossidonnan kautta).

Argumentteihin ja välituloksiin sidottujen todellisten rakenteiden rakennetyypit voivat muuttua, jos käyttäjä muuttaa jonkin tavoitteen kuvausta tai muuttaa sellaisen ulottuvuuden paloittelua, jonka suhteen laskenta on paloittaista. Tällöin voi erityisesti käydä niin, että rakenne, joka on aiemmin ollut solu tai taulu, muuttuu jonoksi. Jos lisäksi rakenne on sidottu muodolliseen taulutyyppiseen argumenttiin, ei järjestelmä tiedä kumpi "taulun" ulottuvuuksista on tulkittava triviaaliksi, joten järjestelmän on kysyttävä se käyttäjältä. Jos ulottuvuuden paloitteluun tehdään myöhemmin lisää muutoksia, ei tätä kysymystä enää esitetä käyttäjälle, vaan järjestelmä toimii aiemmin saamansa vastauksen perusteella.

Käyttäjä voi muokata perittyjä ulottuvuuksien nimiä ja todellisia indeksejä. Tällöin periytyminen katkeaa muokatun olion kohdalta. Jos käyttäjä editoi esimerkiksi yhtä todellista indeksiä, niin muiden (samaan ulottuvuuteen kuuluvien) todellisten indeksien periytyminen säilyy ennallaan.

Käyttäjä voi muuttaa tulosrakenteen sellaisen ulottuvuuden paloittelua, joka ei periydy miltään muulta rakenteelta, mikäli lisäksi tavoitteen laskenta ei ole paloittaista.

Käyttäjä ei voi suoraan muokata tulosrakenteiden solujen arvoja.

Viiteluettelo

Ellis, M. A. & Stroustrup, B. (1990). The Annotated C++ Reference Manual. Addison-Wesley.

Hassinen, K., Sajaniemi, J. & Väisänen, J. (1988). Structured Spreadsheet Calculation. 1988 IEEE Workshop on Languages for Automation, Computer Society Press, 129-133.

Linton, M. A., Vlissides, J. M. & Calder, P. R. (1989). Composing User Interfaces with InterViews. Computer 22,2, 8-22.

Saariluoma, P. & Sajaniemi, J. (1989). Visual Information Chunking in Spreadsheet Calculation. Int. J. Man-Machine Studies 30, 475-488.

Sajaniemi, J., Väisänen, J., Hassinen K. & Tukiainen, M. (1992). B-ASSET-92-kielen määrittely. Joensuun yliopisto, Tietojenkäsittelyopin laitos, Raportti B-1992-3.

Sajaniemi, J., Tukiainen, M. & Väisänen, J. (1993). Paloittelu ASSET-92-järjestelmässä. Valmisteilla Joensuun yliopiston Tietojenkäsitelyopin raporttisarjaan B.

Tukiainen, M., Väisänen, J. & Sajaniemi, J. (1993). ASSET-92-järjestelmään sisältyvät valmiit tavoitteet ja menetelmät. Valmisteilla Joensuun yliopiston Tietojenkäsittelyopin raporttisarjaan B.