B-ASSET-92-kielen määrittely

Jorma Sajaniemi, Jarmo Väisänen, Kari Hassinen, Markku Tukiainen

Joensuun yliopisto
Tietojenkäsittelyopin laitos
PL 111
80101 Joensuu

Tiivistelmä

Tässä raportissa kuvataan rakenteisen taulukkolaskentajärjestelmän ASSET-92 lausekkeiden esittämiseen käytettävä kieli B-ASSET-92. Kieli esitetään antamalla kielen abstrakti syntaksi täsmällisessä muodossa ja kuvaamalla siihen liittyen kielen semantiikka. Abstraktin syntaksin esittämisellä pyritään antamaan selkeä kuva kielen ominaisista piirteistä. Konkreettinen syntaksi esitetään sanallisen kuvauksen ja esimerkkien avulla.

Sisällysluettelo

1. Johdanto

ASSET-92 on rakenteinen taulukkolaskentajärjestelmä, jossa käsiteltävät tiedot muodostuvat yksittäistä solua suuremmista kokonaisuuksista (Hassinen & al., 1988). ASSET-92-järjestelmän tarkoitus on havainnollistaa rakenteisen taulukkolaskennan mahdollisuuksia ja toimia erilaisten kokeilujen alustana. Järjestelmää ei ole toistaiseksi toteutettu kuin vain osittain. ASSET-92 erottelee selkeästi laskennalliset tavoitteet ja niiden toteutukseen käytettävät menetelmät. Tässä raportissa kuvataan järjestelmän kieli B-ASSET-92, jolla ilmaistaan mm. menetelmien tarvitsemat laskentasäännöt. Samalla tulevat esille ne tiedot, jotka tarvitaan menetelemien ja tavoitteiden kuvaamiseen, mutta näiden esitystapaa järjestelmän käyttöliittymässä ei jatkossa käsitellä. Tavoitteiden ja menetelmien tarvitsemia tietoja ja niiden käsittelyä kuvataan tarkemmin raportissa Sajaniemi & al. (1992).

Kieli esitetään antamalla kielen abstrakti syntaksi ja siihen liittyvä semantiikka. Abstraktin syntaksin esittämisellä pyritään antamaan selkeä kuva kielen ominaisista piirteistä. Kielen konkreettinen syntaksi, joka kuvaa sen, miltä kielen rakenteet näyttävät esimerkiksi paperille kirjoitettuina, esitetään vain esimerkkien ja sanallisen kuvauksen avulla. Konkreettista syntaksia ei esitetä formaalisti.

2. B-ASSET-92-kielen käytöstä

B-ASSET-92-kielellä voidaan ilmaista laskennallisia menetelmiä sekä kokonaisten tietorakenteiden (kenno, jono, taulu) että tietorakenteiden osina olevien solujen tasolla. Kokonaista rakennetta kuvataan lausekkeella ja rakenteen osana olevaa solua skalaarilausekkeella.

Menetelmiä määritellessään järjestelmän käyttäjä kirjoittaa kielen mukaisia lausekkeita vain kokonaisten tietorakenteiden tasolla. Järjestelmä itse sen sijaan näyttää sekä kokonaisia tietorakenteita että niiden osia kuvaavia lausekkeita käyttäjän pyytäessä jonkin tiedon laskentamenetelmän selitystä.

Käyttäjän halutessa määritellä uuden tavoitteen pyytää järjestelmä häntä antamaan seuraavat tiedot:

- tavoitteen nimi
- tavoitteen argumenttien nimet ja rakennetyypit (kenno, jono, taulu)
- argumenttien kokoja koskeva rajoitus, joka ilmaistaan totuusarvon palauttavana B-ASSET-92-skalaarilausekkena
- tavoitteen tulosrakenteen nimi ja rakennetyyppi
- ulottuvuuksien periytyvyys
- laskennan paloittaisuus

Vastaavasti käyttäjän halutessa määritellä uuden menetelmän pyytää järjestelmä antamaan seuraavat tiedot:

- vastaavan tavoitteen nimi
- menetelmän nimi
- menetelmän käyttämät välirakenteet ja näiden suhde argumentteihin tavoitetasolla ilmaistuna
- tulosrakenteen arvon antava B-ASSET-92-lauseke

Edellä annettavat rakenteiden nimet ovat luonteeltaan muodollisia parametreja ts. ne eivät ole minkään laskentapohjassa todella esiintyvän rakenteen nimiä. Menetelmää sovellettaessa argumenttien nimet korvautuvat niiden todellisten rakenteiden nimillä, jotka käyttäjä antaa tavoitteen lähtörakenteiksi. Välirakenteiden nimet korvautuvat sellaisten todellisten (valmiiksi olemassaolevien tai menetelmän soveltamisen takia luotujen) rakenteiden nimillä, joilla on vaadittu suhde argumentteihin. Tulosrakenteen nimi korvautuu puolestaan syntyvän todellisen tulosrakenteen nimellä.

Tulosrakenteen arvon antava lauseke voi viitata argumentteihin, välirakenteisiin, tulosrakenteeseen ja menetelmän soveltamisen aikana olemassaoleviin todellisiin rakenteisiin.

Esimerkkinä tavoitteiden ja menetelmien määrittelystä olkoot seuraavat kumuloituvan kokonaissumman, kumuloituvan kokonaiserotuksen, välieron, kokonaisrivisummien ja rivien välisummien määrittelyt :

```
Goal:                       kumuloituva_kokonaissumma
Arguments:                  sequence s
Argumen requirements:       -
Result:                     sequence r
Dimension
  inherited from:           s
  splitting inherited:      yes
Split usage method:         none

Plan for goal:              kumuloituva_kokonaissumma
Plan:                       A
Required subgoals:          -
Expression:                 if i=1 then s[i] else s[i]+r[i-1]; i=1..s#


Goal:                       kumuloituva_kokonaiserotus
Arguments:                  sequence s1, s2
Argument requirements:      s1# = s2#
Result:                     sequence r
Dimension
  inherited from:           s1
  splitting inherited:      yes
Split usage method:         none

Plan for goal:              kumuloituva_kokonaiserotus
Plan:                       A
Required subgoals:          sequence a1 = kumuloituva_kokonaissumma(s1),
                            sequence a2 = kumuloituva_kokonaissumma(s2)
Expression:                 a1[i]-a2[i]; i=1..a1#


Goal:                       väliero
Arguments:                  sequence s
Argument requirements:      -
Result:                     sequence r
Dimension
  inherited from:           -
  splitting inheritaed:     -
Split usage method:         none

Plan for goal:              väliero
Plan:                       A
Required subgoals:          -
Expression:                 s[i]-s[i-1]; i=2..s#


Goal:                       kokonaisrivisumma
Arguments:                  table t
Argument requirements:      -
Result:                     sequence s
Dimension
  inherited from:           t.row
  splitting inherited:      yes
Split usage method:         none
```

```
Plan for goal:            kokonaisrivisumma
Plan:                     A
Required subgoals:        -
Expression:               sum(t[i,1..t.col#]); i=1..t.row#


Goal:                     rivien_välisummat
Arguments:                table t
Argument requirements:    -
Result:                   sequence s
Dimension
  inherited from:         t.row
  splitting inherited:    yes
Split usage method:       simple

Plan for goal:            rivien_välisummat
Plan:                     A
Required subgoals:        -
Expression:               sum(t[i,1..t.col#]); i=1..t.row#
```

Kokonaisrivisumma ja rivien välisummat eroavat toisistaan ainoastaan laskennan paloittaisuuden suhteen. Edellinen ei ole paloittainen, jonka johdosta se tuottaa kutakin riviä kohti yhden tuloksen: rivin solujen summan. Jälkimmäisessä laskenta suoritetaan kullekin palalle erikseen, jonka johdosta yhdestä rivistä voi syntyä useita tuloksia. Asiaa havainnollistaa kuva 1.

Laskennan paloittaisuutta käsitellään tarkemmin raportissa Sajaniemi & al. (1993).

3. B-ASSET-92-lausekkeiden syntaksi ja semantiikka

3.1 Yleistä

Käsiteltävät tietorakenteet jakaantuvat kolmeen rakennetyyppiin: kenno, jono ja taulu. Nämä koostuvat soluista, joissa kussakin on jokin skalaariarvo. Rakenteen solujen sisältämät skalaariarvot voivat olla joko (käyttäjän muokattavissa olevia) vakioita tai ne määräytyvät rakenteen menetelmän mukaisesti. Sillä, kumpi tapa on rakenteen kohdalla käytössä, ei ole mitään merkitystä rakennetta edelleen käyttävän muun menetelmän kannalta.

Kenno koostuu yhdestä solusta.

Jono on yksiulotteinen kokoelma soluja. Solut indeksoidaan [1..n], missä n on jonoon kuuluvien solujen lukumäärä, n>0. (Jonon esitys vaakasuorana "rivinä" tai pystysuorana "sarakkeena" on esitystason kysymys eikä sillä ole mitään tekemistä jonoon kohdistuvan laskennan kanssa.)

Taulu on kaksiulotteinen kokoelma soluja. Solut indeksoidaan [1..n, 1..m], missä n*m on solujen lukumäärä, n>0 ja m>0. Ensimmäistä ulottuvuutta nimitetään riviulottuvuudeksi ja toista sarakeulottuvuudeksi. (Taulu esitetään n rivillä ja m sarakkeella, mutta tämä on esitystason kysymys eikä sillä ole suoranaisesti mitään tekemistä tauluun kohdistuvan laskennan kanssa.) Taulun ulottuvuuksista voidaan käyttää myös taulukohtaisia ulottuvuuksien nimiä ("otsikkotunnuksia"), jotka ovat mielivaltaisia merkkijonoja. Käyttäjä voi esimerkiksi nimetä taulun *Myynti* riviulottuvuuden nimellä *Tuote* ja sarakeulottuvuuden nimellä *Kuu*.

Ulottuvuutta kutsutaan triviaaliksi, jos se on 1..1. Muuten ulottuvuuden sanotaan olevan epätriviaali.

Solujen kokonaislukuina esitettäviä indeksejä kutsutaan solujen absoluuttisiksi indekseiksi. Näiden ohella voidaan käyttää solujen todellisia indeksejä, jotka ovat mielivaltaisia merkkijonoja ja jotka vaihtelevat eri tauluissa ja jonoissa. Käyttäjä voi antaa esimerkiksi jonon *Kausimyynti* todellisiksi indekseiksi *Tammi*, *Helmi*, *Maalis* jne.

Jono ja taulu voidaan paloitella jakamalla yksi ulottuvuus tai molemmat ulottuvuudet yhtenäisiin lukualueisiin (esimerkiksi ulottuvuus 1..16 osiin 1..5, 6..13, 14..14, 15..16). Jos tavoitteen laskenta ei ole paloittainen, ei argumenttien ja välitulosten paloittelulla ole vaikutusta tavoitteen laskentaan. Jos tavoitteen laskenta on paloittainen, lasketaan tulos kustakin palaryhmästä erikseen indeksoimalla palasten solut uudelleen (laskennan ajaksi) alkaen ykkösestä. Vastinpalaryhmien muodostamista käsitellään raportissa Sajaniemi & al. (1993).

Myynti													
		Kuu											
		Tammi	Helmi	Maalis	Huhti	Touko	Kesä	Heinä	Elo	Syys	Loka	Marras	Joulu
Tuote	Appelsiini	11	12	13	14	15	16	17	18	19	20	21	22
	Mandariini	21	22	23	24	25	26	27	28	29	30	31	32
	Banaani	31	32	33	34	35	36	37	38	39	40	41	42
	Mango	41	42	43	44	45	46	47	48	49	50	51	52
	Omena	51	52	53	54	55	56	57	58	59	60	61	62
	Päärynä	61	62	63	64	65	66	67	68	69	70	71	72

Vuosimyynti		
Tuote	Appelsiini	198
	Mandariini	318
	Banaani	438
	Mango	558
	Omena	678
	Päärynä	798

Puolivuotismyynti			
		Col	
Tuote	Appelsiini	81	117
	Mandariini	141	177
	Banaani	201	237
	Mango	261	297
	Omena	321	357
	Päärynä	381	417

Kuva 1: Ei-paloittainen ja paloittainen laskenta

Tavoitetta sovellettaessa voidaan sille antaa todellisiksi argumenteiksi seuraavan luettelon mukaisesti myös muuntyyppisiä rakenteita, kuin mitä tavoitteen määrittelyssä on sanottu.

Vaadittu tyyppi	Todellinen tyyppi	Tulkinta
Jono	Kenno	Kenno tulkitaan jonoksi, jolla on triviaali ulottuvuus.
Taulu	Kenno	Kenno tulkitaan tauluksi, jonka molemmat ulottuvuudet ovat triviaaleja.
Taulu	Jono	Jono tulkitaan tauluksi, jonka toinen ulottuvuus on jonon ulottuvuus ja jonka toinen ulottuvuus on triviaali. Sen, käytetäänkö jonon ulottuvuutta taulun rivi- vai sarakeulottuvuutena, määrää käyttäjä tavoitetta soveltaessaan.

Skalaarityypit ovat kokonaisluku, reaaliluku, totuusarvo ja teksti. Kunkin rakenteen kaikilla soluilla on oltava sama tyyppi, jota kutsutaan ko. rakenteen solutyypiksi.

Arvo *ERR* ei ole mitään tyyppiä. Jos syntaktisesti virheettömän lausekkeen osien tyypit eivät noudata jatkossa kuvattavia vaatimuksia, niin lauseke hyväksytään, mutta sen arvo on *ERR*. Erityisesti tästä seuraa, että jos lausekkeessa yksikin operandi on *ERR*, niin koko lausekkeen arvo on *ERR*.

3.2 Rakenteisen arvon palauttavat lausekkeet

Rakenteinen arvo on rakennetyypiltään kenno, jono tai taulu. Menetelmän kuvaava menetelmälauseke palauttaa jonkin edellämainittua rakennetyyppiä olevan arvon.

plan-expression = *expression*

expression = *scalar-expression* |
structure-indicator |
slice |
constructor

Jos menetelmälausekkeen palauttaman tuloksen tyyppi ei ole tavoitteen tulosrakenteen rakennetyyppi, on menetelmälausekkeen arvo *ERR*. Tuloksen tyyppi saa kuitenkin olla skalaarityyppi, mikäli tulosrakenteen rakennetyyppi on kenno.

structure-indicator = *identifier*

Rakenneindikaattorin arvo on viitattu rakenne.

Mikäli viitattua rakennetta ei ole, on rakenneindikaattorin arvo *ERR*.

slice = *sequence-slice* |
table-slice

sequence-slice =	*sequence-identifier:*	*identifier,*
	indexes:	*index-pair-list*
table-slice =	*table-identifier:*	*identifier,*
	rows:	*index-pair-list,*
	columns:	*index-pair-list*
index-pair-list =	*first-index:*	*index,*
	last-index:	*index,*
	other-pairs:	*other-index-pairs*
other-index-pairs =	*empty* |	
	index-pair-list	

Viipaleen arvo on rakenneindikaattorin ilmaisema rakenne kokonaisuudessaan tai viitatun jonon (taulun) niiden solujen muodostama jono (taulu), joiden indeksit osuvat annetuille väleille. Jos annetuille väleille ei osu yhtään solua, on viipaleen arvo tyhjä, jos viipale on funktion parametri, ja *ERR*, jos viipale on menetelmän palauttama arvo. Viipaleessa solujen todelliset indeksit säilyvät alkuperäisinä, mutta niiden absoluuttiset indeksit muuttuvat alkamaan ykkösestä.

Jonoon kuuluvan solun indeksi osuu annetuille väleille, jos indekseissä on indeksipari, jonka alkuindeksi edeltää ja loppuindeksi seuraa solun indeksiä.

Tauluun kuuluvan solun indeksi osuu annetuille väleille, jos riviosassa on indeksipari, jonka alkuindeksi edeltää ja loppuindeksi seuraa solun rivi-indeksiä, ja sarakeosassa on indeksipari, jonka alkuindeksi edeltää ja loppuindeksi seuraa solun sarakeindeksiä.

Mikäli viitattua rakennetta ei ole tai rakenteessa ei ole viipaleessa käytettyjä indeksejä, on viipaleen arvo *ERR.*

Konkreettisessa syntaksissa merkitään viipaleen indeksit hakasulkujen väliin, erotetaan indeksiparilistat pilkulla, indeksiparit &-merkillä ja indeksiparin indeksit kahdella peräkkäisellä pisteellä. Siten seuraavat ovat kelvollisia viipaleita:

```
LvvProsentti
Kausimyynti
Kausimyynti[1..i]
Myynti[1..5&7..8,Tammi..i]
```

empty =

Tyhjä ei koostu mistään.

constructor =	*sequence-constructor* | *table-constructor*	
sequence-constructor =	*expression:*	*scalar-expression,*
	index-variable:	*identifier,*
	index-bound-1:	*scalar-expression,*
	index-bound-2:	*scalar-expression*
table-constructor =	*expression:*	*scalar-expression,*
	row-variable:	*identifier,*
	row-bound-1:	*scalar-expression,*
	row-bound-2:	*scalar-expression,*
	column-variable:	*identifier,*
	column-bound-1:	*scalar-expression,*
	column-bound-2:	*scalar-expression*

Jonomuodostajan arvo on jono, joka saadaan laskemalla lauseke useita kertoja sitomalla indeksimuuttuja nousevassa järjestyksessä kaikkiin kokonaislukuihin, jotka ovat suurempia tai yhtäsuuria kuin pienempi indeksirajoista ja pienempiä tai yhtäsuuria kuin suurempi indeksirajoista.

Taulumuodostajan arvo on taulu, joka saadaan laskemalla lauseke useita kertoja sitomalla rivimuuttuja nousevassa järjestyksessä kaikkiin kokonaislukuihin, jotka ovat suurempia tai yhtäsuuria kuin pienempi rivirajoista ja pienempiä tai yhtäsuuria kuin suurempi rivirajoista ja jokaista edellistä kohti sitomalla sarakemuuttuja nousevassa järjestyksessä kaikkiin kokonaislukuihin, jotka ovat suurempia tai yhtäsuuria kuin pienempi sarakerajoista ja pienempiä tai yhtäsuuria kuin suurempi sarakerajoista.

Rajojen tulee olla tyypiltään kokonaislukuja tai tekstejä. Mikäli raja on teksti sen tulee esiintyä täsmälleen kerran lausekkeessa esiintyvien rakenteiden todellisten indeksien joukossa, jolloin se tarkoittaa kyseistä todellista indeksiä vastaavaa absoluuttista indeksiä. Jos raja esiintyy edellä mainittujen indeksien joukossa useammin kuin kerran tai ei esiinny niiden joukossa kertaakaan, niin muodostajan arvo on *ERR*.

Konkreettisessa syntaksissa merkitään lausekkeen perään puolipiste, indeksimuuttujan perään yhtäsuuruusmerkki, rajojen väliin kaksi pistettä ja sarakemuuttujan eteen pilkku. Siten seuraavat ovat kelvollisia jono- ja taulumuodostajia:

```
sum(Kausimyynti[1..kk]); kk=Syys..Loka
s[i]-s[i-1]; i=2..s#
i; i=3..6
sum(Myynti[i,1..t.col#]); i=1..Myynti.row#
Myynti[i,j]/sum(Myynti); i=1..Myynti.row#, j=1..Myynti.col#
```

3.3 Skalaariarvon palauttavat lausekkeet

Skalaarilausekkeen arvo on jotain skalaarityyppiä, eli kokonaisluku, reaaliluku, totuusarvo tai teksti, tai se on *ERR*.

scalar-expression =	*number* | *truth-value* | *text* | *cell-reference* | *index-coercion* | *unary-expression* | *binary-expression* | *function-application* | *conditional-expression* | *case-expression* | *goal-expression*

Skalaarilausekkeen arvona on jonkin skalaarityypin arvo. Jos skalaarilauseke on tavoitelauseke ja tavoite ei palauta kennoa, on lausekkeen arvo *ERR*.

Teksti voidaan ilmaista vakiotekstillä, soluviittauksella, ehdollisella lausekkeella ja valinta-lausekkeella. Erityisiä tekstioperaattoreita ei ole.

number =	*integer* | *real*
integer =	*{...}*
real =	*{...}*

Kokonaisluvun arvo on luku itse. Reaaliluvun arvo on luku itse.

Konkreettisessa syntaksissa kokonaisluku koostuu numeroista. Siten seuraavat ovat kelvollisia kokonaislukuja:

```
0
1
009
```

Konkreettisessa syntaksissa reaaliluku koostuu numeroista ja täsmälleen yhdestä desimaali-pisteestä. Desimaalipisteen edessä tai perässä on oltava vähintään yksi numero. Siten seuraavat ovat kelvollisia reaalilukuja:

```
0.0
78.666
567.
.0
```

Lausekkeessa esitettävät kokonaisluvut ja reaaliluvut eivät voi olla negtiivisia. Sen sijaan laskennan tuloksena syntyvät luvut voivat olla negatiivisia.

truth-value = *{false, true}*

Totuusarvon arvo on itse totuusarvo.

Konkreettisessa syntaksissa totuusarvoja merkitään .false ja .true .

text = *{...}*

Tekstin arvo on teksti itse.

Teksti koostuu tyhjettä lukuunottamatta näkyvistä, toteutuksessa käytettävästä merkistöstä riippuvista merkeistä. Seuraavat merkit sisältyvät jokaiseen toteutukseen:

```
abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ
1234567890 !"'#$%&/()=?+.,:;-_*
```

Konkreettisessa syntaksissa teksti ympäröidään heittomerkein ('). Heittomerkki esitetään kahdella heittomerkillä. Siten seuraavat ovat kelvollisia tekstejä:

```
'B-ASSET-92 syntaksi'
'Kirja ''RobinsonCrusoe'''
''''
```

Mikäli teksti esiintyy indeksissä tai rakennemuodostajan indeksi-, rivi- tai sarakerajan lausekkeena, on muodoltaan tunnuksen kaltainen ja ei ole sama kuin ympäröivän rakennemuodostajan indeksi-, rivi- tai sarakemuuttujan tunnus, ei konkreettinen syntaksi vaadi tekstin ympärille heittomerkkejä. Siten konkreettinen syntaksi sallii muodon *Myynti[Omena,i];i=Huhti..Touko*, joka on täysin ekvivalentti pitemmän muodon *Myynti['Omena',i];i='Huhti'..'Touko'* kanssa.

cell-reference = *chamber-cell-reference* |
sequence-cell-reference |
table-cell-reference

chamber-cell-reference = *chamber-identifier:identifier*

sequence-cell-reference = *sequence-identifier:identifier,*
index: *index*

table-cell-reference = *table-identifier:* *identifier,*
row: *index,*
column: *index*

Soluviittauksen arvo on viitatun solun sisältö. Mikäli viitattua rakennetta ei ole tai soluviittauksessa käytetty indeksi ei esiinny rakenteen asianomaisessa ulottuvuudessa täsmälleen kerran, on soluviittauksen arvo *ERR*.

Konkreettisessa syntaksissa merkitään soluviitausta hakasuluilla, joiden sisällä on indeksit pilkuilla erotettuina. Siten seuraavat ovat kelvollisia soluviittauksia:

```
LvvProsentti[]
Kausimyynti[5]
Myynti[Omena,Huhti]
```

Mikäli kennosolun viittaus ei esiinny indeksissä, rakennemuodostajan indeksi-, rivi- tai sarakerajan lausekkeena, argumenttina tai menetelmälausekkeena sallii konkreettinen syntaksi viittaukseksi pelkän kennon nimen. Siten konkreettinen syntaksi sallii muodon *Kuukausi*sum(Kausimyynti)/12,* joka on täysin ekvivalentti pitemmän muodon *Kuukausi[]*sum(Kausimyynti)/12* kanssa, vaikka se vaatii muodot *Kausimyynti[Kuukausi[]]* ja *Myynti[i,j]/12;i=1..Omena,j=1..Kuukausi[].*

identifier =	*{...}*

Tunnukset koostuvat käytettävän merkistön mukaisista kirjaimista, numeroista ja _-merkeistä siten, että tunnuksessa ei voi olla pelkästään numeroita. Tunnus ei saa olla varattu sana. Siten seuraavat ovat kelvollisia tunnuksia:

```
Myynti
Lvv_Prosentti
123_
_
```

index =	*scalar-expression*

Indeksin ilmaisevan lausekkeen tyypin on oltava joko kokonaisluku tai teksti. Jos lausekkeen tyyppi on kokonaisluku, niin indeksi tarkoittaa absoluuttista indeksiä. Jos lausekkeen tyyppi on teksti, indeksi tarkoittaa todellista indeksiä.

index-coercion =	*absolute-index* |	
	actual-index	
absolute-index =	*struct-dim-ind:*	*structure-dimension-indicator,*
	index:	*actual-index-denotation*
actual-index-denotation =	*scalar-expression* |	
	empty	
actual-index =	*struct-dim-ind:*	*structure-dimension-indicator,*
	index:	*absolute-index-denotation*
absolute-index-denotation =	*scalar-expression*	

Absoluuttisen indeksin arvona on viitatun ulottuvuuden todellista indeksiä vastaava absoluuttinen indeksi, jos todellisen indeksin osoitus ei ole tyhjä, tai viitatun ulottuvuuden viimeinen absoluuttinen indeksiarvo, jos todellisen indeksin osoitus on tyhjä.

Todellisen indeksin arvona on viitatun ulottuvuuden absoluuttista indeksiä vastaava todellinen indeksi.

Jos annettu indeksi ei esiinny viitatussa ulottuvuudessa tai se esiintyy siinä useammin kuin kerran, on indeksimuunnoksen arvo *ERR*.

structure-dimension-indicator =sequence-dimension-indicator |
table-dimension-indicator

sequence-dimension-indicator =structure-indicator

table-dimension-indicator =	*table-indicator:*	*structure-indicator,*
	dim-indicator:	*dimension-indicator*

dimension-indicator = *text* |
row-or-column-indicator

row-or-column-indicator = *{row, col}*

Jonon ulottuvuusindikaattori tarkoittaa viitatun jonon (ainoaa) ulottuvuutta. Mikäli viitattu rakenne ei ole jono tai rakennetta ei ole, on jonon ulottuvuusindikaattorin arvo *ERR*.

Taulun ulottuvuusindikaattori tarkoittaa viitatun taulun rivi- tai sarakeulottuvuutta. Mikäli viitattu rakenne ei ole taulu tai rakennetta ei ole, on taulun ulottuvuusindikaattorin arvo *ERR*.

Ulottuvuusindikaattorina oleva teksti ilmaisee todellisen rivi- tai sarakeotsikkotunnuksen. Mikäli viitatulla rakenteella ei ole tekstin mukaista rivi- tai sarakeotsikkotunnusta tai sekä rivi- että sarakeotsikkotunnukset ovat kumpikin tekstin mukaisia, niin ulottuvuusindikaattorin arvo on *ERR*.

Rivi-tai-sarakeindikaattori row viittaa riviin ja col sarakkeeseen.

Konkreettisessa syntaksissa laitetaan indeksimuunnoksessa olevan epätyhjän indeksin eteen < ja perään >. Absoluuttisessa indeksissä laitetaan merkin < eteen #. Taulun ulottuvuusindikaattorissa erotetaan rakenneindikaattori ja ulottuvuusindikaattori pisteellä. Siten seuraavat ovat kelvollisia indeksimuunnoksia:

`Kausimyynti#`	(Arvo 12)
`Myynti.row#`	(Arvo 6)
`Myynti.Kuu#`	(Arvo 12)
`Kausimyynti#<Helmi>`	(Arvo 2)
`Kausimyynti<2>`	(Arvo 'Helmi')
`Myynti.row#<Omena>`	(Arvo 5)
`Myynti.Kuu<5>`	(Arvo 'Touko')

unary-expression =	*operator:*	*unary-operator,*
	operand:	*scalar-expression*

Unaarisen lausekkeen arvo saadaan soveltamalla operaattoria operandiin.

binary-expression =	*operator:*	*binary-operator,*
	operand1:	*scalar-expression,*
	operand2:	*scalar-expression*

Binaarisen lausekkeen arvo saadaan soveltamalla operaattoria operandeihin.

unary-operator = *{+, -, not}*

Unaaristen operaattorien soveltamisella on tavanomainen merkityksensä.

Unaaristen operaattorien + ja - operandien tulee olla kokonaislukuja tai reaalilukuja ja niiden tulos on operandin tyyppiä. Operaattorin *not* operandin tulee olla totuusarvo ja sen tulos on totuusarvo.

binary-operator = *{+, -, *, /, div, mod, and, or, =, <>, <, <=, >, >=}*

Binaaristen operaattorien soveltamisella on tavanomainen merkityksensä. Operaattori *div* tarkoittaa kokonaislukujakoa ja *mod* jakojäännöstä. Negatiivisten lukujen tapauksessa *div* ja *mod* lasketaan etumerkin määräytymistä lukuunottamatta kuten vastaaville positiivisille luvuille.

Binaaristen operaattorien operandien ja tulosten tyyppien välillä vallitsee seuraava yhteys:

Operaattori	Operandit	Tulos
+, -, *	Kokonaislukuja	Kokonaisluku
	Kokonaisluku ja reaaliluku	Reaaliluku
	Reaalilukuja	Reaaliluku
/	Kokonaislukuja	Reaaliluku
	Kokonaisluku ja reaaliluku	Reaaliluku
	Reaalilukuja	Reaaliluku
div, mod	Kokonaislukuja	Kokonaisluku
and, or	Totuusarvoja	Totuusarvo
=, <>, <, <=, >, >=	Kokonaislukuja	Totuusarvo
	Kokonaisluku ja reaaliluku	Totuusarvo
	Reaalilukuja	Totuusarvo

Mikäli operandien tyypit eivät ole operaattorien vaatimusten mukaisia, on lausekkeen arvo *ERR*.

Konkreettisessa syntaksissa unaariset operaattorit merkitään operandin eteen, binaariset operaattorit operandien väliin ja operaattoreilla on seuraavat presedenssit:

Korkein presedenssi:	unaariset operaattorit
	and, or
	=, <>, <, <=, >, >=
	*, /, div, mod
	+, -
Matalin presedenssi:	ehdollisuus, valinta

Samalla presedenssitasolla olevat operaattorit lasketaan vasemmalta oikealle. Konkreettisessa syntaksissa voidaan käyttää myös sulkuja (ja) osoittamaan laskentajärjestys. Siten seuraavat ovat kelvollisia unaarisia ja binaarisia lausekkeita:

```
-Kausimyynti[12]
LvvProsentti[]*(--Kausimyynti[12]+5)
```

function-application =	*function:*	*function,*
	arguments:	*arguments*
function =	*{sum, avg, min, max, err}*	
arguments =	*empty* |	
	non-empty-arguments	
non-empty-arguments =	*argument:*	*argument,*
	arguments:	*arguments*
argument =	*expression*	

Funktiosovelluksen arvo saadaan soveltamalla funktiota argumentteihin.

Funktion argumenttien tyypeille asetettavat vaatimukset riippuvat sovellettavasta funktiosta seuraavasti:

Funktio	Argumentit	Tulos
sum, min, max	Kokonaislukuja ja kokonaislukuja sisältäviä viipaleita	Kokonaisluku
	Kuten edellä, mutta joukossa myös reaalilukuja	Reaaliluku
avg	Kokonaislukuja, reaalilukuja ja näitä sisältäviä viipaleita	Reaaliluku
err	-	ERR

Mikäli argumentit eivät ole funktion vaatimusten mukaisia, on funktiosovelluksen arvo *ERR*.

Konkreettisessa syntaksissa merkitään argumenttien ympärille sulut (ja) ja argumentit erotetaan pilkuilla. Jos argumentteja ei ole, ei sulkuja merkitä. Siten seuraavat ovat kelvollisia funktiosovelluksia:

```
sum(Kausimyynti[1..12])
min(Myynti,Kausimyynti)
err
```

conditional-expression =	*condition:*	*scalar-expression,*
	then-part:	*scalar-expression,*
	else-part:	*scalar-expression*

Ehdollisen lausekkeen arvo on then-osan arvo, mikäli ehdon arvo on true, ja muutoin else-osan arvo.

Ehdon tyypin tulee olla totuusarvo. Then- ja else-osien tyyppien ja tuloksen tyypin välillä vallitsee seuraava yhteys:

Then- ja else-osat	Tulos
Kokonaislukuja	Kokonaisluku
Kokonaisluku ja reaaliluku	Reaaliluku
Reaalilukuja	Reaaliluku
Totuusarvoja	Totuusarvo
Tekstejä	Teksti

Mikäli ehdon, then-osan ja else-osan tyypit eivät ole edelläkuvattujen vaatimusten mukaisia, on ehdollisen lausekkeen arvo *ERR*.

Konkreettisessa syntaksissa merkitään ehdon eteen if, then-osan eteen then ja else-osan eteen else. Siten seuraavat ovat kelvollisia ehdollisia lausekkeita:

```
if Kausimyynti[Helmi] > 0 then Kausimyynti[Helmi] else 0
if LvvProsentti[] > .16 then LvvProsentti[]*Kausimyynti[i] else err
```

case-expression =	*selector:*	*scalar-expression,*
	case-list:	*case-list,*
	else-part:	*scalar-expression*
case-list =	*case-values:*	*case-values,*
	expression:	*scalar-expression,*
	other-cases:	*other-cases*
other-cases =	*empty* |	
	case-list	
case-values =	*case-value:*	*integer,*
	other-c-values:	*other-case-values*
other-case-values =	*empty* |	
	case-values	

Valintalausekkeen arvo on sen skalaarilausekkeen arvo, johon liittyvässä ehtoarvojen listassa esiintyy valitsimen arvo. Jos valitsimen arvo ei esiinny missään listassa, on valintalausekkeen arvo else-osassa esiintyvän skalaarilausekkeen arvo.

Saman kokonaisluvun esiintyminen useammin kuin kerran saman valintalausekkeen ehtoarvona on syntaktinen virhe.

Valitsimen tyypin on oltava kokonaisluku.Valintalistan ja else-osan lausekkeiden tyyppien ja tuloksen tyypin välillä vallitsee seuraava yhteys:

Valintalistan ja else-osan lausekkeet	Tulos
Kokonaislukuja	Kokonaisluku
Kokonais- ja reaalilukuja	Reaaliluku
Reaalilukuja	Reaaliluku
Totuusarvoja	Totuusarvo
Tekstejä	Teksti

Mikäli valitsimen, valintalistan ja else-osan lausekkeiden tyypit eivät ole edelläkuvattujen vaatimusten mukaisia, on valintalausekkeen arvo *ERR*.

Konkreettisessa syntaksissa merkitään valitsimen eteen case, ehtoarvojen listan eteen if, valintalistan lausekkeen eteen then ja else-osan eteen else. Siten seuraaava on kelvollinen valintalauseke:

```
case Lkm[]   if 5 then 20
             if 6,8,11 then Lkm[]*Kausimyynti['Omena']
             else 0
```

3.4 Tavoitelausekkeet

goal-expression =	*goal:* *arguments:*	*goal,* *arguments*
goal =	*predefined-goal* | *user-defined-goal*	
predefined-goal =	*{...}*	
user-defined-goal =	*identifier*	

Tavoitteet ovat ennaltamääriteltyjä tai käyttäjän määrittelemiä laskennallisia tavoitteita. Ennaltamääritellyt tavoitteet kuvataan raportissa Tukiainen & al. (1993).

Käyttäjä ei voi kirjoittaa lausekkeisiin tavoitteita vapaasti. Hän voi käyttää tavoitetta tuloslausekkeena vain, mikäli lauseke koostuu yhdestä tavoitelausekkeesta, jonka argumentit ovat rakenneindikaattoreita. Sen sijaan järjestelmä voi käyttää tavoitteita vapaasti selittäessään rakenteen tai sen osan laskentaa käyttäjälle.

Konkreettisessa syntaksissa merkitään argumenttien ympärille sulut (ja) ja argumentit erotetaan pilkuilla. Jos argumentteja ei ole, ei sulkuja merkitä. Siten seuraavat ovat kelvollisia tavoitelausekkeita:

```
kumuloituva_kokonaiserotus(Kausimyynti,EdellinenKausimyynti)
parametriton_tavoite
```

Viiteluettelo

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

Sajaniemi, J., Tukiainen, M. & Väisänen, J. (1992). Tavoitteet ja menetelmät ASSET-92-järjestelmässä. Joensuun yliopisto, Tietojenkäsittelyopin laitos, Raportti B-1992-4.

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.