**T A R T U  Ü L I K O O L**

MATEMAATIKA-INFORMAATIKATEADUSKOND

Arvutiteaduse instituut

Infotehnoloogia eriala

**Marbel Tamm**

# Vaimse võimekuse testide automaatne genereerimine

Magistritöö (30 EAP)

Juhendaja: Margus Niitsoo, MSc

Juhendaja: Rene Mõttus, PhD

Autor: ……..… “…..“ mai 2010

Juhendaja: …….. “…..“ mai 2010

Lubada kaitsmisele

Professor .....…… “……“ ……. 2010

TARTU 2010

# SISUKORD

# SISSEJUHATUS

Inimestele pakub huvi oma võimeid teistega võrrelda. Üheks erinevuse mõõdupuuks on intelligentsus ehk rahvakeeli tarkus. Intelligentsust defineeritakse kui väga üldist vaimset suutlikkust. Reeglina seostatakse seda erinevate probleemide ja ülesannete eduka lahendamise oskusega [1]. Intelligentsust on keeruline mõõta mingil fikseeritud absoluutsel skaalal, kus ühes äärmuses on geenius ja teises vaimselt alaarenenud inimene. Ei saa väita, et isikul A puudub intelligentsus täiel määral või isikul B on maksimaalne intelligentsuse tase. Intelligentsuse taset saab paika panna ainult inimeste üksteisega võrdlemise teel.

Intelligentsuse olemuse küsimustele on vastuseid otsitud aastaid. Psühholoog Charles Spearman (1863-1945) märkas, et tema poolt läbi viidud vaimse võimekuse uuringutes said testitavad reeglina eri ülesannetes suhteliselt sarnaseid tulemusi. Nimelt inimesed, kes olid keskmisest edukamad üht tüüpi IQ testides, olid tihti edukad ka kõikide teiste lahendamisel. Samuti kehtis vastupidine nähtus: testis üht ülesannet kehvemalt sooritanud isikud said madalaid tulemusi väga erineva sisuga ülesannetes. Sellest lähtuvalt võib üksikute ja konkreetsete võimete asemel üritada vaadelda üldvõimekust (*g faktor*), mis mõjutab soorituse edukust kõigis vaimset pingutust nõudvates valdkondades [2].

Intelligentsuse hindamiseks on koostatud mitmeid teste, mille õigete vastuste konfitsendi alusel pannakse inimesed pingeritta ning omistatakse neile IQ väärtus. IQ kirjeldab inimese paiknemist võrdlusgrupi keskmise tulemuse suhtes nii, et 15 IQ punkti vastab ühele standardhälbele. Ehkki suurem osa IQ teste mõõdavad mingil määral üldist vaimset võimekust, on siiski täheldatud, et leidub ülesandeid, mille lahendamise võime korreleerub Spearmani g-ga oluliselt tugevamalt kui teistel. Teisisõnu eristavad sedatüüpi ülesanded inimesi eriti hästi just üldise võimekuse osas. Näiteks kuuluvad sellesse klassi nn maatrikstüüpi ülesanded, milles tuleb inimesel esmalt avastada talle esitatud kujundite põhjal seaduspära ning seejärel selle seaduspära üle kanda puuduvate kujundite leidmiseks. Peamiselt seostatakse taolisi ülesandeid siiski mitte Spearmani enda, vaid tema õpilase Raveniga. Seetõttu nimetatakse neid ülesandeid Raveni maatriksiteks [3].

Maatrikstüüpi testide põhimõte ja välimus on järgmine: ülesanne koosneb 3x3 suurusest tabelist, kus kaheksa lahtri sisse on paigutatud teatud reegli alusel abstraktsed objektid ning

üheksas lahter (alumine parem) on tühi. Lahendaja ülesandeks on mõista, mis seaduspärasuse või loogika põhjal toimuvad tabeli ruutudevahelised muutused, ning seejärel valida viimasesse täitmata lahtrisse kaheksast valikvastuse variandist õige.

Mugava ja lihtsa kasutuse tõttu on maatrikstüüpi testid praktikas ühed kõige laiemalt tarvitatavatest vaimse võimekuse mõõtmise vahenditest. Nimelt võimaldavad ainult geomeetrilistest kujunditest koosnevad ülesanded testi läbi viia väga erineva kultuurilise, vanuselise ja haridusliku taustaga inimeste seas, sest pildiline esitus ei nõua eelteadmis [4, 5, 6]. Erinevalt individuaalset tüüpi intelligentsuse määramise testidest, kus iga testitavaga tegeletakse eraldi, saab suurele inimgrupile lahendamiseks anda ühed ja samad ülesanded ning hiljem tulemusi analüüsida. See võimaldab säästa ressursse, sest inimeste ükshaaval testimine on väga kallis ja ajamahukas.

Tänu maatrikstüüpi ülesannete eelnimetatud headele omadustele on oluline uurida nende koostamise ja edasi arendamise võimalusi. Tänapäevani on teadaolevalt kõik teaduslikud testid, mida on kasutatud inimeste üldvõimekuse mõõtmiseks, olnud koostatud käsitsi. On üritatud luua arvutisüsteeme, mis suudaksid juba loodud testides kitsama valdkonna ülesandeid edukalt lahendada [7]. Ülesannete genereerimise uurimine on aga jäänud taha-plaanile. Samas on testide mitmekesistamiseks kasulik omada ülesannete automaatse genereerimise võimalust, et luua hulgaliselt unikaalseid ülesandeid. Kuna samade ülesannete korduv kasutamine põhjustab tulemuste moonutuse nn õppimisefekti tõttu, võimaldaks generaator testida inimesi korduvalt. Lisaks saaks vajadusel teste kasutada adaptiivsel moel, kus järgnevate testiülesannete raskusaste sõltub eelnevate ülesannete lahendamise edukusest. Käesolev töö käsitlebki testigeneraatori võimalikkust ja selle konstrueerimisel kerkivaid probleeme.

Magistritöö koosneb kuuest osast. Esimeses jaotises antakse lugejale intelligentsustestide olemuse mõistmise jaoks psühhholoogia-alast taustinformatsiooni ning seletatakse lühidalt maatrikstüüpi ülesannete generaatori vajalikkust koos loodava programmi funktsionaalsete nõudmistega. Teises peatükis esitatakse ülevaade abstraksete kujundite loomisest ning tuuakse välja neile rakenduvad piirangud. Kolmandas jaotises kirjeldatakse ülesannetes kasutatavaid reegleid, nende koosmõju nii üksteise kui erinevate kujunditega. Neljandas peatükis esitatakse mitmeid võimalusi kujundite ja seoste sobitamiseks inimesele visuaalselt meelepärasteks

ülesanneteks. Viiendas jaotises on toodud detailsem algoritm terviklike testide koostamiseks. Viimane peatükk on pühendatud magistritöö käigus loodud testiprogrammile (*demo*), mille loomisel rakendati praktikas käesoleva magistritöö teoreetilisi ideid. Rakendusega lisanduvad kasutamisjuhised, klassidiagrammid ja vajaminevad programmid (näiteks *Flash Player*).

# 1. VAIMSE VÕIMEKUSE TESTID

Psühholoogid jagavad inimestevahelised psühholoogilised erinevused laias laastus kahte suuremasse gruppi. Ühte kategooriasse kuuluvad kognitiivsed võimed (*intellektuaalne intelligentsus*), mis seostuvad indiviidi üldise suutlikkusega tulla toime erinevate vaimset pingutust nõudvate ülesannetega ja mõelda ratsionaalselt. Teise alamjaotusesse paigutatakse isiksuse omadused (nn *suur viisik* – ekstravertsus, neurootilisus, meelekindlus, avatus ja sotsiaalsus), mis väljendavad oskust mõista iseenda ja teiste tundeid, tunda huvi ümbritseva maailma vastu ning käituda vastavalt vaimsele tajule [8, lk 175 - 179]. Kuna käesolev magistritöö on seotud ainult vaimsete võimetemõõtmisega, siis edaspidi on intelligentsuse all mõeldud kognitiivseid võimeid.

Intelligentsust võib kirjeldada mitmeti. C. Spearman leidis, et erinevat tüüpi ülesannete lahendamise oskus kipub taanduma ühele muutujale – üldvõimekusele (*general intelligence* ehk *g-faktor*). Üldvõimekusele lisanduvad igale ülesandetüübile unikaalsed erivõimed (*specific factors* ehk *s-faktorid*). Seega, kui erinevad vaimset pingutust nõudvad ülesanded mõõdavad nii üld- kui ka spetsiifisemaid võimeid, siis kõikide ülesannete tulemuste kokku liitmisel võimendub üldvõimekus spetsiifilistest enam. Spearmani intelligentsuse mudelit on erinevate autorite poolt kritiseeritud ja täpsustatud. Näiteks L. Thurstone (1887 – 1955) arvates jaotuvad võimed väiksemateks alamgruppideks:

1. verbaalne võimekus,
2. ruumiline ettekujutus,
3. numbriline anne,
4. taju kiirus,
5. kõne voolavus,
6. mõtlemine,
7. töömälu.

R. Catell (1905 – 1998) aga rühmitas intelligentsuse kaheks osaks järgmiselt: voolav ning kristalliseerunud võimekus. Voolav intelligentsus tähistab inimese arutlusoskust ja erinevates olukordades toimetulekut. Kristalliseerunud intelligentsus kajastab omandatud teadmiste ja oskuste rakendamist konkreetsetes situatsioonides. Ehkki intelligentsuse liigitamiseks on

erinevaid viise, nõustutakse, et seni tehtud uuringute tulemused viitavad üldvõimekuse eksiteerimisele [8 lk 281 - 294].

Inimeste võimete mõõtmiseks on mitmeid viise. Üheks võimaluseks on individuaalne vaimse võimekuse test, mille käigus püütakse hinnata testi sooritaja intelligentsust üksikute ülesannete seeriatega (näiteks *Wechsler* ja *Stanford-Binet* testid). Individuaalsed testid võivad sisaldada peale kirjalike ülesannete ka pusle ja klotside mänge, samuti suulist küsimus-vastus seanssi. Kui lahendaja ei saa ülesandega hakkama, siis esitatakse talle uued lihtsamad ülesanded. Kuna iga testitava jaoks on üks testija, siis on kogu testimisprotsess väga kallis [9].

Väiksema ajakuluga saab võimete taset kindlaks teha grupitestiga, kus inimrühmale antakse lahendamiseks komplekt erinevatest kirjalikest ülesannetest ning hiljem analüüsitakse tulemusi. Nii individuaal– kui grupitestid on standardiseeritud, st testiküsimused ning tulemuste analüüsiprotsess on ühesugune kõikide uuritavate puhul. Mida rohkem on erinevaid testitavaid inimesi, seda enam hakkab testitulemustest välja kujunema normaaljaotus ehk Gaussi kõver. Normaaljaotus kirjeldab olukorda, kus suurem osa resultaate on keskmise taseme väärtuse lähedased, mõlemasuunalisi kõrvalekaldeid aga vähem. Intelligentsustesti läbinud inimrühma tulemusi saab võtta etaloniks iga järgmise isiku võimete hindamisel. Seega võib individuaalseid ja grupiteste pidada peamiseks teaduslikuks vaimse võimekuse uurimise vahendiks [10].

Võttes aluseks Spearmani välja toodud üldvõimekuse summeerumise igas vaimset pingutust nõudvates testides, võib järeldada, et intelligentsustaseme täpsemaks määramiseks tuleb inimestele lahendamiseks anda võimalikult erinevaid ülesandeid. Siiski ei ole intelligentsus-testid vaid komplekt juhuslikke küsimusi. Ühe testi koostamise peale kulub palju vaeva ja tööd. Pärast erinevate ülesannete väljamõtlemist tuleb test normeerida ehk anda lahendada inimrühmale ning seejärel testitulemustest leida üldpopulatsiooni jaotus ning arvutada standardhälve jm. Kogu normeerimise protsess on kallis ja ajakulukas ning seetõttu on kasutuses olevate testide arv suhteliselt väike. Teaduslikes intelligentsuse uuringutes kasutatakse enamasti kolme järgmist Raveni maatriksitel põhinevat testikomplekti.

- Värvilised kasvava raskusastmega ülesanded (*Coloured Progressive Matrices*) on mõeldud pigem lastele, kellele standardversioon ei sobi.

- Standardsed kasvava raskusastmega maatriksid (*Standard Progressive Matrices*) on keskmise raskusastmega ja seega ka kõige populaarsemad.
- Edasijõudnutele mõeldud testid (*Advanced Progressive Matrices*) sobivad hindamiseks keskmisest kõrgemate võimetega inimestele, kellele standardversiooni keerukustase on liiga madal, sest saavutatakse maksimumi lähedasi tulemusi ja mistõttu ei suudeta testitavate omavahelisi erinevusi usaldatavalt tuvastada [11].

Ehkki kirjeldatud Raveni teste on kasutatud aastaid IQ mõõtmiseks, on intelligentsuse uuringute jaoks sarnaseid ülesandekomplekte juurde loodud üksikuid. Põhjuseks võib oletada inimressursi ja aja puudumist. Antud probleemi saaks leevendada maatriksteste automaatselt genereeriva süsteemiga, sest arvutiprogrammiga saadud erinevate piltide valik oleks mahukam ning lihtsustaks testide komplekteerimist.

Teaduslike vaimsete võimete uuringute jaoks on testide koostamisel ja kasutuskõlblikeks muutmisel alati mängus inimkomponent, kes vaatab genereeritud ülesanded üle. Ülesannetevalikust selekteeritakse testidesse välja võimalikult erinevaid ülesandetüüpe. Näiteks nõuab mõne ülesande lahendamine numbrilist arvutust, teine aga järeldusvõimet. Sellest lähtuvalt piisaks esialgu programmist, mis genereeriks sobivaid ülesandeid piisavalt tihti.

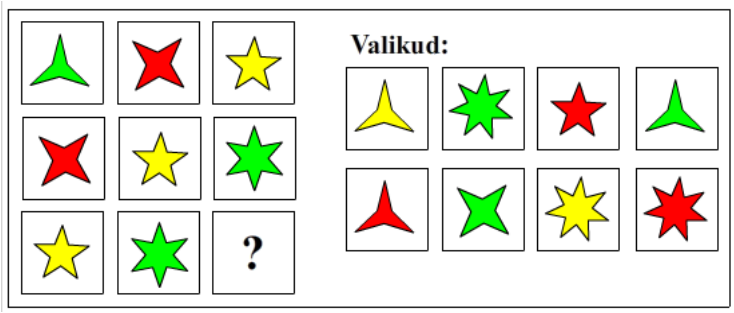
Piisava kasuteguri garanteerimiseks kehtivad ülesannetegeneraatorile teatavad üldisemad funktsionaalsed nõuded. Nimelt on loodava süsteemi eesmärgiks luua elemente (kujundeid) ning esitada neid visuaalselt viisil, mis võimaldaks tajuda ülesandes kasutatud reegleid. Lisaks on vaja, et genereeritud test oleks üheselt mõistetav, kujundid arusaadavad ning üldine pilt piisavalt lihtsa ülesehitusega. Lahendaja peab saama keskenduda vaid seoste mõistmisele ega sattuma segadusse ülesannetes juhuslikult tekitatud joonistest.

Kuna maatrikstüüpi testides on kasutatud ainult kujundeid ning esitatud need kindla seaduspärasuse alusel, siis võib kogu testiülesande koostamise protsessi jaotada nelja põhikomponendi vahel. Esiteks tuleb uurida, milliseid abstraktseid kujundeid ülesannetes esitada. Arvutiga võib koostada väga palju erinevaid kujutisi, aga vähesed on neist maatrikstüüpi testides kasutatavad. Teiseks komponendiks on reeglite kogum. Ülesande ruutudes kujundite muutusi vaadates peab lahendaja mõistma kujunditele rakendatud seoste

olemust, näiteks kujundi nurkade arvu suurenemist. Reeglid peavad olema lahendajale arusaadavad. Järgmine etapp on sobitada abstraksed kujundid ja reeglid omavahel visuaalselt meelepärasteks ülesanneteks. Võimalusi on selleks mitmeid: jaotada kujund tükkideks ning värvida reeglite alusel vastavad alamosad, liita kujundid kokku jne. Loodud ülesannetes ei tohi esineda konflikte seoste ja kujundite vahel ega eksisteerida mitut erinevat lahenduskäiku.

Viimaseks komponendiks on valikvastuste genereerimine. Iga ülesande juurde antakse kaheksa valikut, mis võiksid olla piisavalt keerulised, et lahendaja ei saaks koheselt kõiki valesid elimineerida. Näiteks on väga lihtne keerulisi ülesandeid lahendada nendesse süvenemata, kui kolme valiku kujundid ei sobi ülesandesse üldse (kujundid on täiesti võõrad), kaks valikut on ülesandes juba esinenud piltidega ning ülejäänud valede kujundite värv on vale.

Kasutades kõiki nelja komponenti, saab generaatori abil moodustada sisult keerulisi, kuid visuaalselt lihtsaid maatrikstüüpe ülesandeid (näide on esitatud joonisel 1). Kirjeldatud komponendid on pikemalt lahti seletatud järgmistes peatükkides.



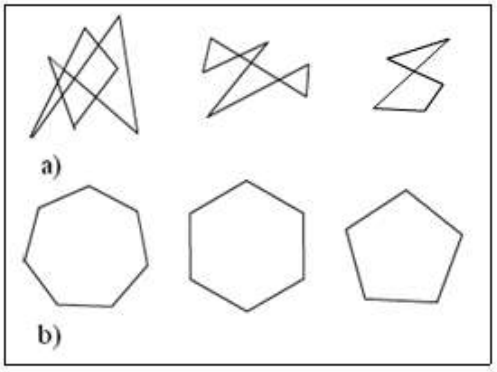

Joonis 1. Maatrikstüüpi ülesanne.

# 2. ABSTRAKTSED KUJUNDID

Maatrikstüüpi intelligentsustestide baasmaterjaliks on abstraktsed kujundid. Nimelt võetakse kujundid ja rakendatakse nende peal erinevaid reegleid. Ülesannetes võivad kujundid jääda samasuguseks (muutub vaid värv, asend) või deformeeruda (näiteks suureneb nurkade arv). Ehkki testides on põhirõhk kujundite vahelistel seostel, tuleb neile endile ka tähelepanu pöörata. Järgnevalt kirjeldatakse kujunditele eksisteerivaid kitsendusi ning tuuakse välja erinevaid võimalusi ülesandesse visuaalselt kasutuskõlblike jooniste saamiseks.

## *2.1 Piirangud abstraktsetele kujunditele*

M. Meo, M. J. Roberts ja F. S. Marucci poolt läbi viidud uuringu ("*Element salience as a predictor of item difficulty for Raven's Progressive Matrices*") tulemusi analüüsides selgus, et parimad tulemused saadi ülesannetes, kus kujundeid oli lihtne identifitseerida [12]. Seega võib kujundite lihtsust pidada üheks kriteeriumiks. Kui eesmärgiks on teha väga keerulisi ülesandeid, siis võib lihtsuse piirangut ignoreerida. Näiteks võtab joonisel 2 esitatud nurkade arvu muutumise reegli mõistmine rohkem aega just esimese (a) variandi puhul.

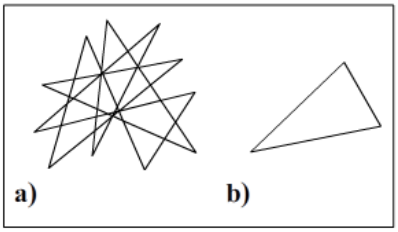



Joonis 2. Kujundi nurkade arvu vähenemine.

Siiski on vaja, et testides kasutatud kujundid oleksid selged ning üheselt mõistetavad ega tõmba endale üleliigset tähelepanu. Kuna maatrikstüüpi testides on eesmärgiks hinnata inimeste oskust taibata abstraktseid reegleid ning osata neid rakendada probleemide lahendamiseks, siis peab testi lahendaja saama vaevata keskenduda seoste mõistmisele, mitte sattuma segadusse jooniste keerukuse tõttu. Seetõttu võib kujundite keerukuse uuringu põhjal järeldada, et ülesannetes on soovitatav kasutada objekte, mida enamik inimesi on pärast ühe

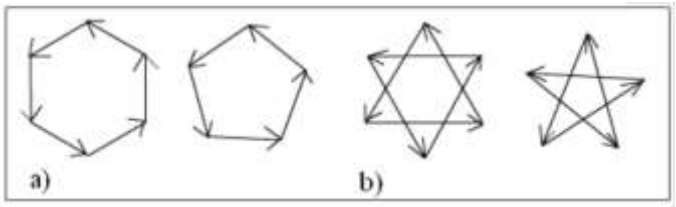
sekundi vaatamist suutelised peast järgi joonistama. Näiteks sobivad tuttavad kujundid nagu ring, kolmnurk ja ruut.

Üsna suur keerukust mõjutav tegur on kujundis kasutatud joonte arv: mida vähem jooni, seda lihtsam ja arusaadavam on objekt (joonis 3). Seega, kui lasta arvutil genereerida üks kinnine ebakorrapärane kujund, ei tohiks tulemusel olla servi liiga palju. Näiteks võiks piirduda kuni kümne piirjoonega. Joonisel 3 (a) esitatud kujundi negatiivse küljena võib mainida ülesande lahendamisele kuluvat liigset energiat ning testi sooritaja ehmatamist segase pildi näitamisega. Pealegi on ülesannetes eesmärgiks mõista rakendatud reeglit. Näiteks on oluline kujundi tippude arvu vähenemine, mitte nurkade kokku lugemise reaktsiooniaja mõõtmine. Seetõttu on mõttekam keskenduda pigem keerukamate reeglite loomisele, sest intelligentsuse mõju on madala taseme kognitiivsetes oskustes (kujundi tippude kokku lugemine jne) väiksem, kui kõrgema taseme (abstraktsete) seoste märkamises [13, lk 18, 74].




Joonis 3. Joonte arv kujundites. Variant (b) on lihtsam variandist (a).

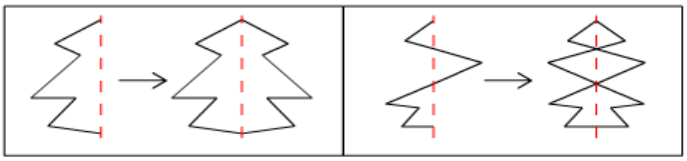
Üldiselt peetakse sümmeetrilisi kujundeid visuaalselt meeldivamateks. Kui jooned on üksteisest võrdsetel kaugustel, siis tundub kujund meeldejäävam ning lihtsam. Näiteks võttes ringi ning paigutades sellele teatud arvu punkte (rohkem kui kaks) ja ühendades need omavahel, saame tuntud objektid (joonis 4).




Joonis 4. Kujundid, kus tipud on ühendatud iga järgmise (a) või ülejärgmisega punktiga (b). Noole suund näitab järgmist tippu, kuhu joon tõmmata.

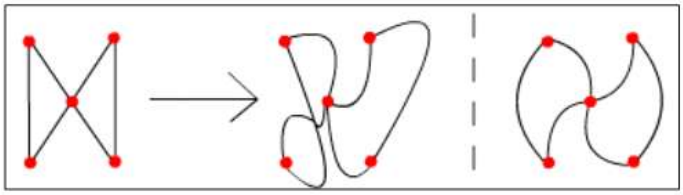
Lisaks kõikide punktide asetamisele võrdsetele kaugustele võib sümmeetrilisust saavutada ka peegeldamise võttega. Kui ühendada suvaliselt valitud punktid nii, et jooned omavahel ei

lõikuks ning seejärel kopeerida sama kujund vertikaalis teisele poole telgjoont, saame tulemuseks korrapärase kujundi (joonis 5). Algus- ja lõpppunktid tuleb asetada keskjoonest võrdsetele kaugustele, vastasel juhul ei teki kinnist kujundit.



Joonis 5. Peegeldusel saadud kujundid.

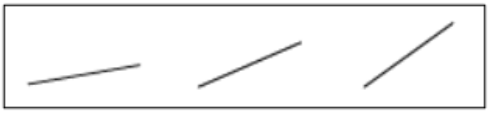
Kujundit moodustatavad jooned ei tarvitse olla kõik sirged. Võib kasutada ka kõverjooni. Paraku ei pruugi juhusliku kõverjoone kasutamine tagada ülesande kvaliteeti. Näiteks on joonisel 6 toodud pilt vasakul pool oleva kujundi joonte muutmisest, tekitades nii juurde kaks uut kujundit. Esimene saadud objektidest on liiga abstraktne ega sobi ülesannetesse, teine aga on visuaalselt lihtsam ning testis kasutajasõbralikum. Seetõttu viitab antud näide asjaolule, et parema tulemuse annavad ühtlane kaare kumerus ja omavahel mitte lõikuvad jooned.



Joonis 6. Sirgete joonte muutmine kõverateks.

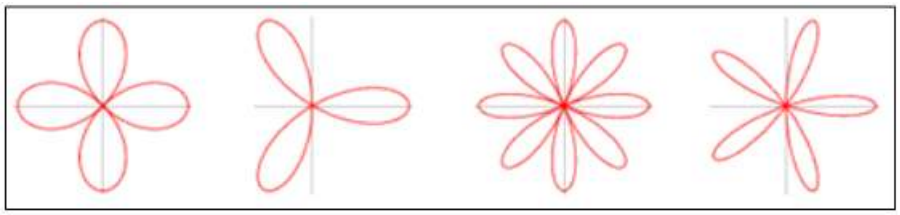
## *2.2 Matemaatilised funktsioonid kujundite tekitamiseks*

Kujundeid saab joonistada erinevate matemaatiliste funtsioonide alusel. Saadud objektide sobivuse määravad samuti eelpool nimetatud üldpõhimõtted, peamiselt kujundi lihtsus. Päris juhuslike valemite kasutamine ei ole mõistlik. Näiteks lineaarvõrrandite sirgeid saaks testides esitada erineva tõusu muutusega (joonis 7), aga ülesanne ise jääks kesiseks.



Joonis 7. Sirge tõusu muutmise ülesanne.

Küllaltki mitmekesiseid resultaate saab näiteks tsükloididest (ingl. keeles *cyclodal curves*), mille hulka kuuluvad epitsükloid, hüpotsükloid jm. Mõned näited ülesannetes kasutatavast Rhodonea kõverast on joonisel 8.

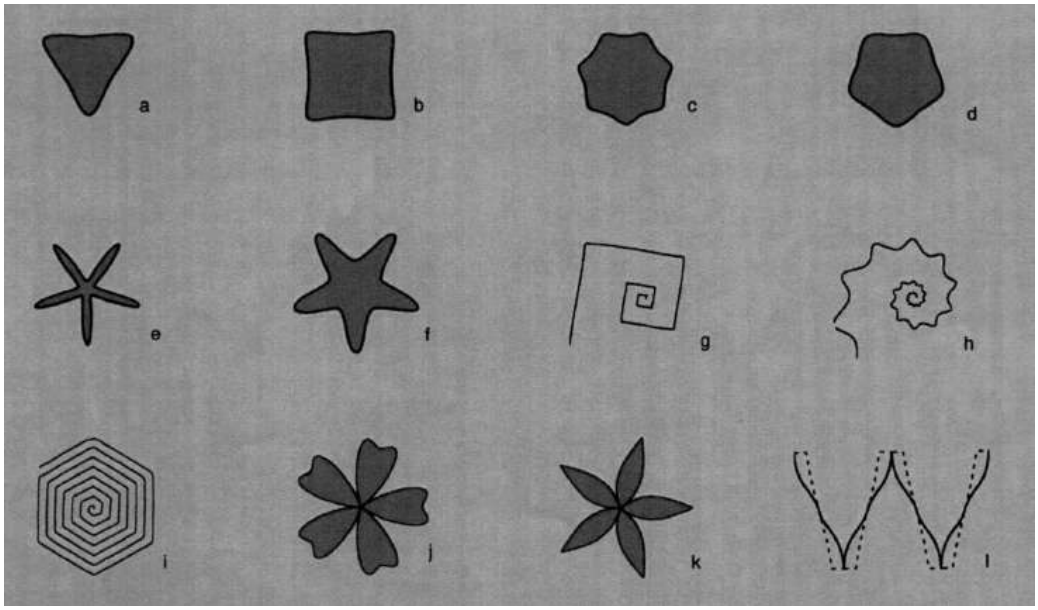


Joonis 8. Rhodonea kõverad [14].

Sobivalt lihtsaid kujundeid on võimalik saada 1999 J. Gielise poolt avaldatud "supervalemi" (*superformula*) abil. Gielis oli inspireeritud looduses leiduvatest kujuditest ning leidis, et suur osa naturaalsetest objektidest on kujutatavad polügoonidega. Loodud supervalemi abil esitas ta uue geomeetrilise lähenemisviisi abstraktsete kujundite paremaks modelleerimiseks ja mõistmiseks. Tegemist on võrrandiga, mille graafikutena esituvad väga mitmekesised ning vaheldusrikkad kujundid. Erinevad resultaadid tekivad nelja numbri kombinatsioonist. Polaar-koordinaatides on valem järgmine:

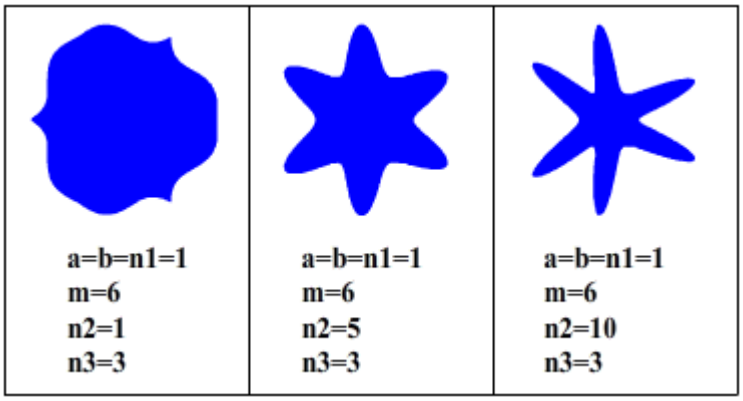
$$r(\varphi) = \left[ \left| \frac{\cos\left(\frac{m\varphi}{4}\right)}{a} \right|^{n2} + \left| \frac{\sin\left(\frac{m\varphi}{4}\right)}{b} \right|^{n3} \right]^{-\frac{1}{n1}},$$

kus $r$ on raadius ja $\varphi$ nurk [15]. Supervalemiga saadavaid võimalikke näidiskujundeid on esitatud joonisel 9.

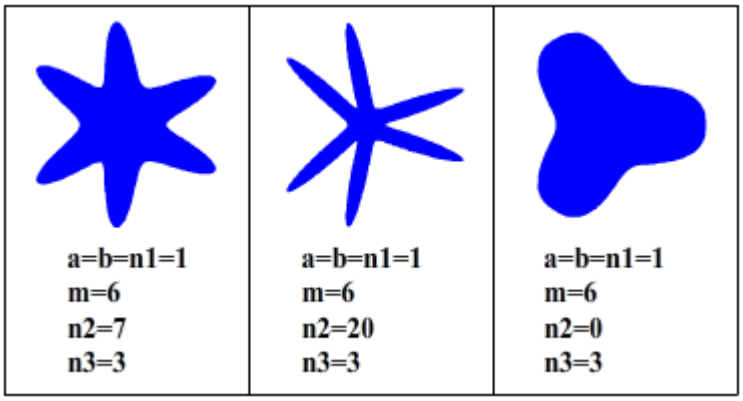



Joonis 9. Supervalemi näited [15].

Sümmeetrilisi ning lihtsaid kujundeid saab kasutada kõigi maatriksis rakendatavate reeglitega, alustades deformeerimisega (nurkade arvu suurendamine, joonte kumerus) ja lõpetades üldisemate muutustega (kujundi suurus, värv). Kahjuks ei kehti see matemaatiliste valemite abil koostatud kujunditele. Kui matemaatilistes funktsioonides suurendada mõne parameetri väärtust, siis üldiselt kajastub muutus valemiga moodustatavas graafikus ehk kujundis. Näiteks on joonisel 10 Gielise supervalemi abil esitatud kolm erinevat kujundit. Neid kujundeid moodustavas valemis on ainult ühte parameetrit modifitseeritud – *n2* on joonisel vasakult poolt lugedes esimesel kujundil 1, teisel 5 ja kolmandal 10. Reegliks oleks sellisel juhul muutuja *n2* suurendamine viie ühiku võrra. Kahjuks ei ole võimalik toodud näidet ülesandena kasutada, sest lahendaja näeb vaid loodud kujundeid ega tea parameetrite väärtustest midagi. Kui valikvastustesse panna mõned teised *n2* väärtusel saadud kujundid (joonis 11), siis on eksimisvõimalus suur. Seetõttu on ülesannetes võrranditega esitatud kujunditel kasutatavad ainult üldisemad objektimuutuse reeglid nagu värv, suurus jne.




Joonis 10. Gielise supervalemiga loodud kujundid - parameeter *n2* muutub viie ühiku võrra.

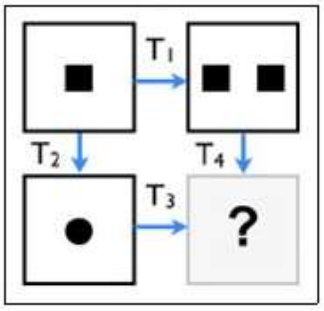



Joonis 11. Võimalikud valed vastused joonisel 10 esitatud kujundiülesandele.

# 3. REEGLID

Iga Raveni testiülesanne sisaldab abstraktseid kujundeid, mis muutuvad reas ja/või tulbas kindlal viisil. Üldiselt on muutuvaid parameetreid mitu, näiteks kujundi asukoht, värv, suurus. Testi lahendajal tuleb aru saada esitatud kujundite omavahelistest seostest ning saadud teadmiste abil jõuda õige vastuseni. M. Kunda, K. McGreggor ja A. Goel kirjeldasid maatriksi ülesandeid järgmiselt: „Raveni Progressiivseid Maatrikseid võib vaadelda piltide seeriana (järjestatud reas ja tulbas), kus mõni tundmatu teisendus (*transformatsioon*) T muudab ühe pildi teiseks lähedalasuvaks pildiks“ [16]. Näiteks on joonisel 12 toodud ülesandes neli transformatsiooni. Muutuste T1 ja T2 kohta saadud info põhjal tuleb leida kujundi(te) teisenduste analoogiad T3 ja T4 jaoks. Seejärel saab puuduva pildi tuletada T3 ja T4 rakendamisega vastavatele olemasolevatele joonistele. Antud ülesandes on järgmised muutused:

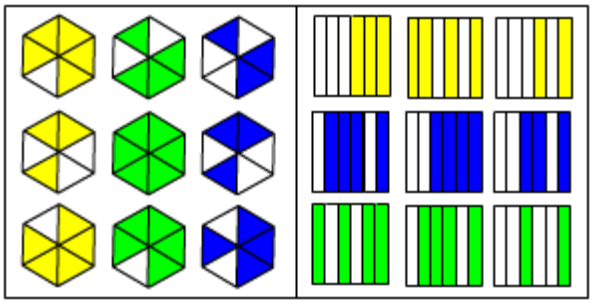
- T1 ja T3 – kujundi kahekordistumine, olgu selleks reegel A;
- T2 ja T4 – kujundelemendi/elementide transformeerumine, olgu selleks reegel B.




Joonis 12. Piltide transformeerumine muutusega T [16].

Võttes aluseks muutustest T1-T4 tuletatud reeglid A ja B, saab genereerida puuduva pildi (alumine parem ruut). Reegel A määrab, et vastuses on kaks ühesugust kujundit ning reegel B kujundi(te) muutust ruudust ringiks. Tulemus loetakse õigeks vaid siis, kui see ei ole vastuolus ühegi ülesandes rakendatud reegliga. Seega tuleb testi lahendajal leida piltide vahel eksisteerivatele seostele T1-T4 kindel tähendus.

Olemasolevaid teste vaadates võib ülesannetes märgata reeglitüüpide erinevate variatsioonide kasutust. Näiteks on joonisel 13 esitatud ülesannetes samad seosed - „loogiline JA“ ning „3 eri värvi“, kuid ülesanded on siiski erinevad.



Joonis 13. Kaks ülesannet reeglitega „Loogiline Ja“ ning „3 eri värvi“.

E. Hunt (1975) paigutas ülesanded (teisisõnu reeglid) kahte suuremasse gruppi:

- gestalt (tervik, mis on rohkem kui osade summa);
- analüütilist laadi.

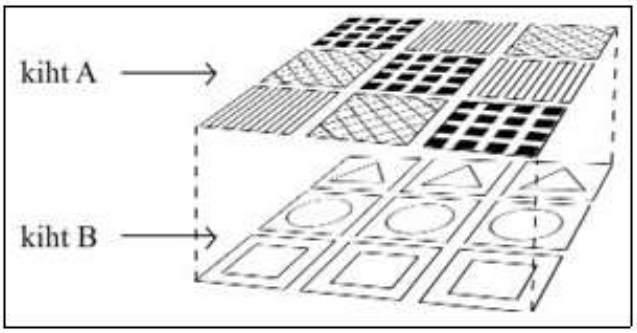
Tegemist oli siiski pigem lahendusalgoritmide jaotamisega, kuid liikudes lahenduselt tagasi probleemile, avastame, et sama grupeeringut saab kasutada ka ülesannete koostamisel. Esimesse rühma kuuluvaid maatriksi teste kirjeldas Hunt järgmiselt: „*lahendab probleeme, kasutades ülesandemaatriksi visuaalset üldpilt.*“, teise aga „*rakendab ülesandemaatriksis esitatud kujundite tunnustele loogilisi tehteid*“ [17]. Alljärgnevalt on täpsemalt kirjeldatud lahendusalgoritmide kahte jaotusgruppi, kuid vaadelduna reeglite seisukohast.

## *3.1 Dekoratiivsed reeglid*

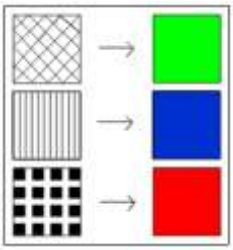
Paljud ülesanded kujutavad maatrikstüüpi testides mosaiikpilti, kus kõik pildiosad kokku moodustavad ühe terviku. M. J. Lawson ja J. R. Kirby soovisid uurida intelligentsust terviklike ülesannete peal eraldi, st ülesandeid lahendades tuli analüütilist mõtlemist võimalikult palju vältida. Suunamaks inimesi maatrikstüüpi testides kasutama pigem üldpildi olemust, andsid Lawson ja Kirby lahendajatele ette instruktsiooni: „*Me hakkame lahendama ülesandeid, kus sa pead välja nuputama, mis on pildilt puudu. Iga ülesanne on põhimõtteliselt ühe puuduva tükiga muster. Sa pead valikutest välja valima tüki, et panna see tühimikku nii, et pilt või muster oleks tervik/lõpetatud. Vaata igat pilti ja proovi välja valida puuduv tükk. Sa pead valima tüki, mis täiendab pilti, mis tekitaks mulje heast mustrist.*“. Seega on osad ülesanded lahendatavad visuaalse analoogia arutluse teel, kus tuleb kindlaks määrata kasutuses olev muster[18]. Seetõttu on gestalt tüüpi seoseid mugavam käsitleda dekoratiivse seose nime all.

### 3.1.1 Mustriline paigutus

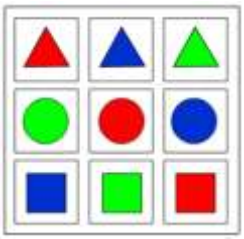
Kasutatud mustrit saab testiülesannetes käsitleda abstraktsete kujundite peale paigutatud eraldiseisva kihina (joonis 14, kiht A). Selline pealmine pind määrab ära vastavad parameetrid, mis rakenduvad allpool olevatele kujunditele (iga ruut omaette). Näiteks võttes joonisel 14 kujutatud kihi A, määrates seal parameetrite väärtusteks kolm erinevat värvi (joonis 15) ning rakendades neid kihil B olevatele kujunditele, saame tulemuseks nö värvireegliga dekoreeritud ülesande (esitatud joonisel 16).




Joonis 14. Dekoratiivne reeglikiht A ülesande B peal.



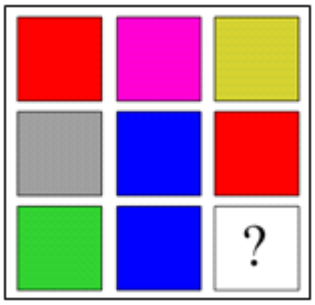
Joonis 15. Dekoratiivse kihi parameetrite väärtustamine.



Joonis 16. Dekoratiivne värvireegel rakendatuna kujundite ülesandele.

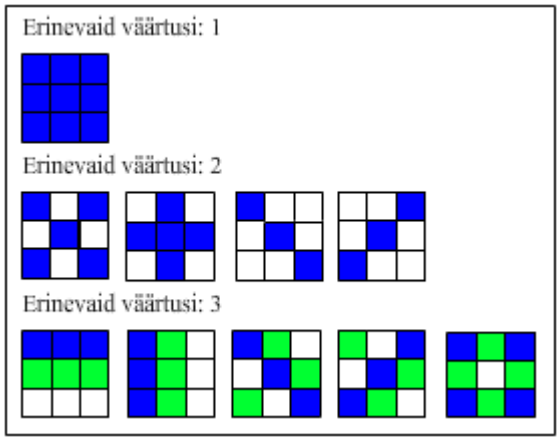
Kõikide dekoratiivsete reeglite korral on kasutuses mustriline paigutus (ingl. keeles *layout*), mille alusel pannakse paika parameetrite väärtuste maatriks. Joonisel 14 esitatud kiht A sisaldab vaid ühte võimalikest asetuse skeemidest. Erinevaid võimalusi ei ole palju, sest iga

pildi puhul tuleb säilitada loogiliselt järeldatav puuduv alumine parem ruut. Näiteks ei ole võimalik joonisel 17 esitatud maatriksis üheselt määrata küsimärgiga tähistatud ruudu värvi.




Joonis 17. Ebasobiv mustripaigutus värvide näitel.

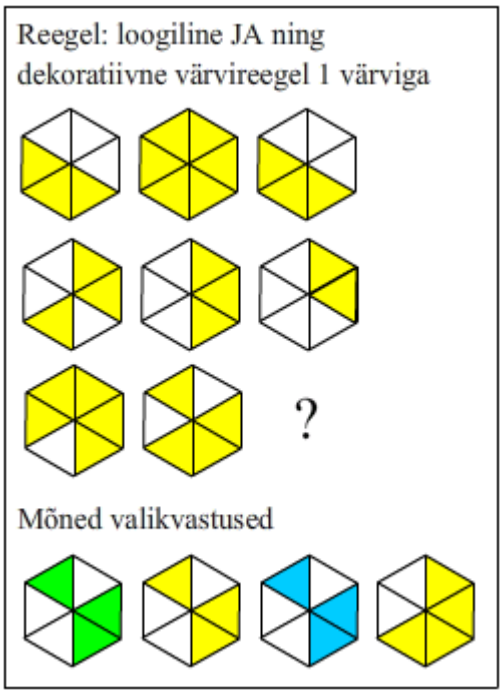
Mustrilistes paigutustes on üldiselt piirdutud kuni kolme varieeruva parameetri väärtusega. Mida rohkem on korraga kasutuses erinevaid paigutusobjekte 3x3 suuruses maatriksis, seda raskem on tekitada sobivat ja loogilist asetuse skeemi. Joonisel 18 on välja toodud hetkel ülesannetes enim rakendatud variandid.




Joonis 18. Võimalikud paigutuse skeemid, kus värvide asukohale pannakse värvile vastav parameetriväärtus.

Võib tekkida küsimus, kas ainult ühe väärtusega dekoratiivne reegel on kasutatav - sisemiselt ei muutu ülesande üheksas ruudus midagi ning tegu justkui polekski reegliga. Tegelikult vaadates kogu maatriksi probleemi koos valikvastustega, on selge, et lahendajal tuleb siiski arvestada vastava lisatingimusega. Vastuste seast on vaja välja selekteerida need valikud, kus on samuti kasutatud konkreetse dekoratiivse kihi parameetreid. Näiteks saab joonisel 19 õigeks vastuseks olla vaid üks kollastest kujunditest. Samuti rikastab antud juhtum ülesannete

väljanägemist. Isegi kui valikute seas peaks juhtuma, et kõik variandid kasutavad sama parameetrit (kõik on kollased), siis vähemalt välimuse poolest on samalaadsed ülesanded erinevad (mõni on rohelist, teine sinist värvi).

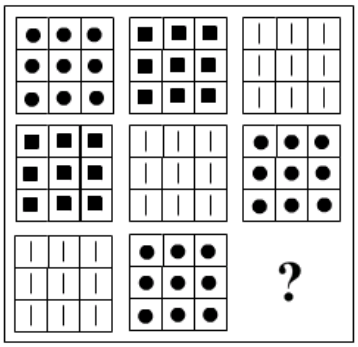



Joonis 19. Ühe parameetri väärtusega dekoratiivne reegel.

### 3.1.2 Dekoratiivse reegli väljund

Dekoratiivse reegli väljundiks on parameetrite väärtuste 3x3 mõõtmeline maatriks. Lahendajale ülesannet ette kuvades saab maatriksist järgi vaadata kujundi(te) joonistamiseks vajalikke väärtusi, näiteks värv, suurus ja positsioon. Kõigi dekoratiivsete seoste korral kehtib lõpliku väljundi (maatriksi) saamiseks sama tegevuskäik. Lühidalt on algoritmi seletus järgmine: valida ülesandes rakendatava dekoratiivse seose jaoks mustriline paigutus ning väärtustada maatriksis kõik parameetrid vastavalt asetusskeemile, näiteks panna joonis 18. mustrites eri värvide kohale kujundite erinevad suurused.

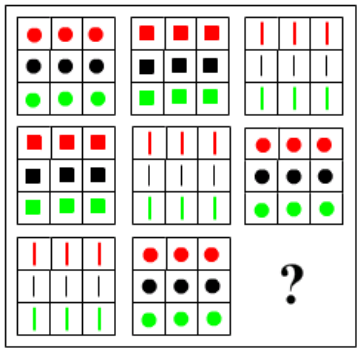
Enamasti kasutatakse dekoratiivse reegli väljundit kogu ülesande peal ühe tervikuna, st iga väljundimaatriksi osa rakendub ainult konkreetsele testiülesande ruudule (joonis 16 reegliga jooniselt 15). Samas võib üks pildiruut olla juhtumisi ise omakorda 3x3 maatriks (joonis 20).




Joonis 20. Ülesande iga ruut on 3x3 maatriks.

Sellistel puhkudel saab gestalt mustreid kasutada ülesande ruudustiku asemel ühe pildiruudu sees. Nimetagem dekoratiivse reegli ruudusisest kasutamist *introvertseks* dekoratiivse reegli rakendusviisiks. Näiteks on joonisel 21 iga ülesande ruudu sees kasutatud järgmist värvimaatriksit:

| | | |
|---|---|---|
| punane | punane | punane |
| must | must | must |
| roheline | roheline | roheline |




Joonis 21. Dekoratiivne värvireegel ruudusisesel lähenemisel.

Kogu reegel rakendub täpselt üheksa korda – kord iga suure ülesanderuudu sees. Kuna dekoratiivse seose väljund on suurusega 3x3, siis on reegli introvertse kasutamise tingimuseks üheksa kujundi nähtav olemasolu igas ülesande ruudus. Nii saab kõikidele ruudusisestele objektile panna külge konkreetsed mustrilised väärtused. Vastasel juhul võib ülesandes tekkida liiga suuri tühimikke, mistõttu ei ole seose muster ebapiisava visuaalse info tõttu arusaadav.

### 3.1.3 Dekoratiivsete reeglite omadused

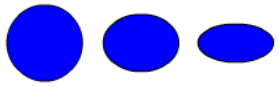
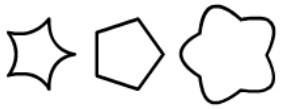
Mustrilise paigutuse üheks suurimaks plussiks võib lugeda kerget täiendamisvõimalust. Nimelt saab olemasolevatele alati juurde lisada uusi sobivaid asetusviise, mis rakenduvad automaatselt kõikidele eksisteerivatele dekoratiivsetele reeglitele. Samuti on lihtne luua uusi seoseid, sest reeglites tuleb ainult määrata võimalike kasutatavate parameetrite väärtused (näiteks erinevad värvid). Paraku saab manipuleerida vaid kujundi visuaalsete omadustega, nagu pöörlemine, värv, skaleerimine jne. Ülesannete dekoratiivsetes seostes ei saa kasutada matemaatilisi tehteid, sest kujundite konkreetsete parameetrite täpseid väärtusi ei osata visuaalse vaatluse teel hinnata. Näiteks võib kergesti eksida õige vastuse leidmisel pöörlemiskraadide liitmistehte $23° + 77° = 100°$ korral. Õiged tunduvad nii $90°$, $100°$ kui ka $110°$, kuna täpsed algandmed ei ole teada. Seepärast kuuluvad kõik kujundite animeerimisega (pidev liikumine) tegelevad reeglid dekoratiivsete alla.

Järgnevalt on tabelis 1 loetletud peamisi kujundite modifitseerimise parameetreid koos võimalike väärtustega. Parameetrite väärtused tuleb määrata piisavalt suurte vahedega, et kujundite muutused oleksid inimesele silmaga eristatavad.

Tabel 1

**Dekoratiivsed reeglid**

| **Nimetus** | **Kirjeldus** | **Võimalikud parameetri väärtused** | **Näide** |
|---|---|---|---|
| Värv (*color*) | Kinniste kujundite sisu või sirgjoonte puhul jooned on värvitud | Värvideks on punane, must, sinine, roheline, kollane, roosa, hall |  |
| Joone paksus (*line thickness*) | Kujundite piirjooned on erineva paksusega | 1.0; 3.0; 5.0 px |  |

| | | | |
|---|---|---|---|
| Pööramine (*rotation*) | Kujundid on pööratud keskpunkti suhtes | Üldiselt 0, 90, 180, 270 kraadi, kuid iga kujundi puhul võib see olla erinev, näiteks sektorringil *x**360/*y*, kus *x* on pööramissammu kordaja ja *y* sektorite arv |  |
| Positsioon (*position*) | Kujundi paiknemine ruudu sees | 9 erinevat positsiooni. Ruut tuleb jaotada 3x3 maatriksiks, iga maatriksi osa on üheks väärtuseks |  |
| Läbipaistvus (*alpha*) | Kujundid võivad osaliselt tagatausta näidata | 25%, 50%, 75% või 100% kujundi selgust |  |
| Skaleerimine (*scale*) | Kujundid deformeeruvad ühe telje suunas | 0%, 25%, 50%, 75% originaalsest suurusest vertikaalis ja/või horisontaalis |  |
| Kujundi valimine (shapes) | Valitakse ülesannetesse kujundeid | Valikutes on lihtsamad kujundid nagu ring, hulknurk (3-9 nurka) jne |  Näiteülesanne on joonisel 15 |
| Kaare kuju (curve) | Kujundi servad on kaardus | (, \| või ) |  |

### 3.1.4 Animeeritud dekoratiivsed reeglid

Täiesti omaette grupi moodustavad ainult arvutiga esitatavad animeeritud reeglid. Sarnaselt visuaalselt paigal seisvatele (*staatilistele*) dekoratiivsetele seostele kasutatakse liikuvates (*dünaamilistes*) reeglites kujundite üldisemaid parameetreid - värv, joone paksus vms. Dünaamiliste seoste peamiseks eripäraks on kujundites toimuvate muutuste kiirus. Tavaliselt kulub muutuse tsüklile algolekust lõppolekusse 0 (muutust ei toimu), 0.5, 1 või 1.5 sekundit.

Näiteks deformeerub joon ülesande esimeses reas ühe piksli laiuselt kolmeni kiiresti (poole sekundiga), järgmises aga aeglasemalt (tervele tsüklile kulub üks sekund). Iga reegli puhul võib teha erandeid ning lisaks kiirusele määrata ka suuna. Näiteks pöörleb kujund mõnes ruudus päripäeva, teistes vastassuunas. Seega on animeeritud reegli väljundmaatriksis fikseeritud muutuste konstantsed sammud - erinevus hetke ja uue aja vahel. Kujundite visuaalsete omaduste liikuv efekt saavutatakse muutumatute väärtuste lisamisega hetke väärtustele iga teatud ajavahemiku tagant. Näiteks kasutades horisontaalset liikumist kahe erineva väärtusega, võime defineerida järgmise maatriksi: 1, -5, 1, -5, ..., -5. Vastavalt maatriksi väärtustele liiguvad kujundid osades ruutudes ühe piksli võrra edasi paremale, teistes aga viis pikslit vasakule iga kümnendik sekundi järel.

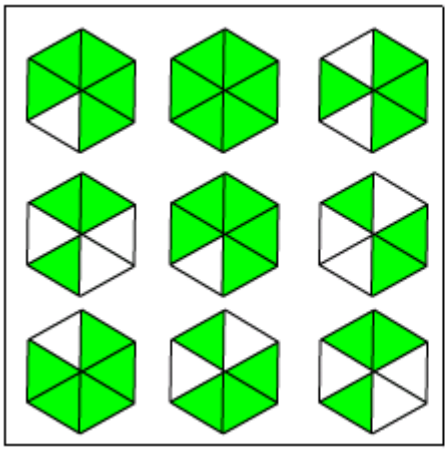
## *3.2 Analüütilised reeglid*

Teise reeglite kategooriasse kuuluvad analüütilised seosed, kus ülesande lahendamisel ei piisa vaid visuaalsest vaatlemisest ja kena mustri otsimisest. Kasutusel on hoopis algoritmid, mis „*tegelevad kujundi abstraktsete tunnuste muutmisega kasutades tehteid nagu konstantsus, täiendus, eemaldus, laiendus, kontraktsioon, lisamine, tükeldamine, liikumine, kompositsioon ning dekompositsioon*“ (E. Hunt) [17]. Lawson ja Kirby soovisid sarnaselt dekoratiivsete seoste leidmise võimekuse analüüsimisele uurida inimeste analüütiliste reeglite leidmise oskusi. Uuringugruppi kuuluvatele katsealustele anti enne testi lahendamist lugemiseks järgmine instruktsioon: „*Me hakkame lahendama ülesandeid, kus Sa pead välja nuputama, mis on pildilt puudu. Iga ülesande jaoks on reegel, mis ütleb Sulle, mis peaks minema tühimikku. Sa pead välja mõtlema selle reegli ja seejärel välja mõtlema puuduva tüki. Kõigepealt püüa aru saada reeglist ja see aitab sind puuduva tüki leidmisel.*“ [18]. Seega tuleb ülesannetes tähele panna kujunditega toimuvaid muutusi ning välja nuputada kujundite-vahelise transformatsiooni seaduspärad.

### 3.2.1 Reeglite rakendamise suund

Kõikide analüütiliste reeglite puhul ei ole võimalik kasutada dekoratiivsetele seostele sarnaseid mustrikihte. Vastasel juhul muutub ülesanne liiga segaseks. Näiteks ei toimu joonisel 22 olevate sektor-kuusnurkade muutused mitte kõrvuti asetsevate, vaid pigem diagonaalis paiknevate ruutude vahel. See aga raskendab oluliseltloogilise „JA“ reegli

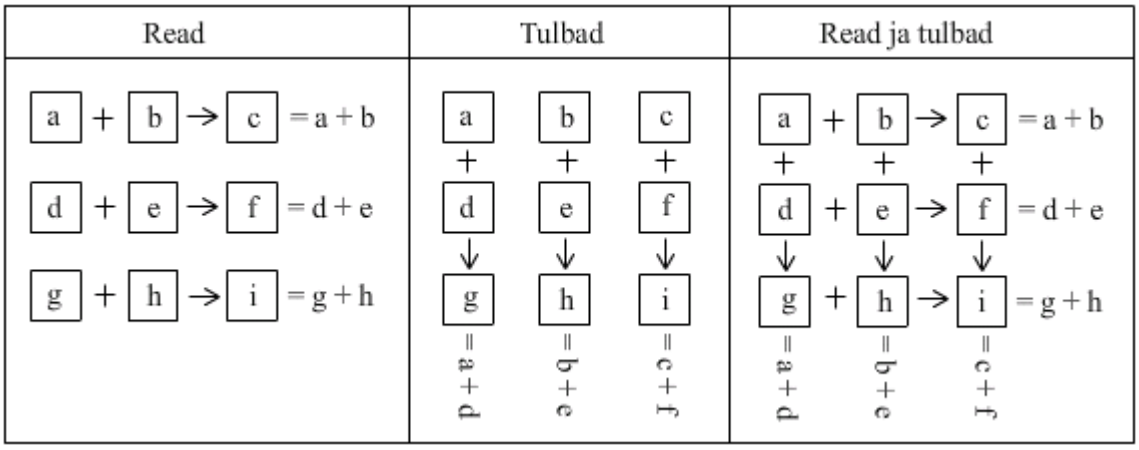
märkamist. Joonise kõrval on välja toodud illustreeriv juhend kuusnurkade komplektidesse jaotumise kohta. Kui sellist abijuhendit ei ole antud, siis tuleb lahendajal proovida erinevaid ruutude kõrvutamise kombinatsioone ning valesid ruute võrreldes võib leida enda jaoks hoopiski muu esialgsest erineva seaduspärasuse. Esitatud probleemi vältimiseks on analüütilised reeglid rakendatud kas ainult veergudele (vasakult paremale), tulpadele (ülevalt alla) või siis mõlemale korraga (joonis 23). Viimane variant ei tee ülesannet raskemaks, vaid pigem lihtsamaks, sest lahendaja võib probleemile läheneda ükskõik millises suunas.



| | | |
|---|---|---|
| 1 | 2 | 3 |
| 1 | 2 | 3 |
| 1 | 2 | 3 |

Rakendatud on dekoratiivreeglite ühte paigutusviisi (tähistatud kolme erineva värviga) ning loogiline reegliülesanne on kohtadel järgmiselt: 1 + 2 = 3.

Joonis 22. Reegel „Loogiline JA“ dekoratiivse reegli paigutusega.




Joonis 23. Analüütilise reegli erinevad rakendamise suunad.

### 3.2.2 Reeglite alamgrupeering

Kõik analüütilised reeglid baseeruvad matemaatilisel tehtel, näiteks liitmisel (arvude puhul) või konjuktsioonil (bittide korral). Vastavalt reeglis kasutatava matemaatilise tehte tüübile võib käesolevad reeglid jaotada omakorda kahte suuremasse alamkategooriasse:

- loogikaalgebra ehk binaarloogika (n *konjuktsioon, disjunktsioon*);
- elementaaralgebra (n *liitmine, lahutamine*).

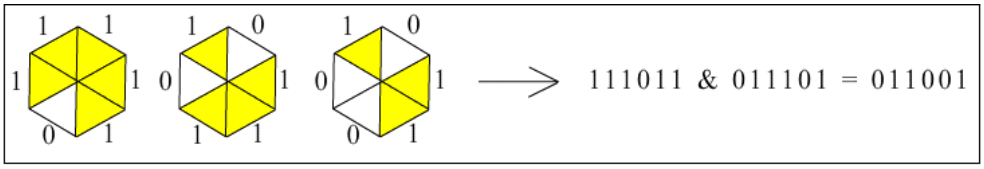
Mõlema reeglirühma tehteid võib ülesannetes rakendada kahel erineval viisil:

- summa ehk kahest parameetrist tekib kolmas: $A + B = C$;
- järjend ehk väärtused muutuvad järjestikku: $A \rightarrow B \rightarrow C$.

Alljärgenevalt on kirjeldatud mõlemat reegliterühma.

#### 3.2.2.1 Binaarloogika

Binaarloogikat (*Boolean logic*, rajajaks George Boole, inglise matemaatik ja loogik) kasutatakse peamiselt arvutiteaduses ja digitaalelektroonikas. Lihtsustatult võib öelda, et loogikaalgebra tegeleb üksikute bittide võrdlemisega. Erinevate funktsioonide kasutamisega määratakse loogilise tehte tulem [19]. Neid funktsioone saab edukalt rakendada maatrikstüüpi testides. Nimelt vaadates Raveni kasvava raskusastmega (progressiivseid) ülesandeid, on märgata, et osad kujundid kord eksisteerivad ja siis jälle kaovad sama koha pealt, mõni joon on vasakule ning järgmise pildi peal paremale kaldu. Selliste „nähtuste“ erinevaid olekuid on ainult kaks (ing. keeles lihtsustatult *on/off*). Seetõttu saab neid arvutimaailmas esitada bittidena: 1 – esimene seisund, 0 – teine seisund. Joonisel 24 on toodud näide sektorringi teisendamisest bitijadaks.

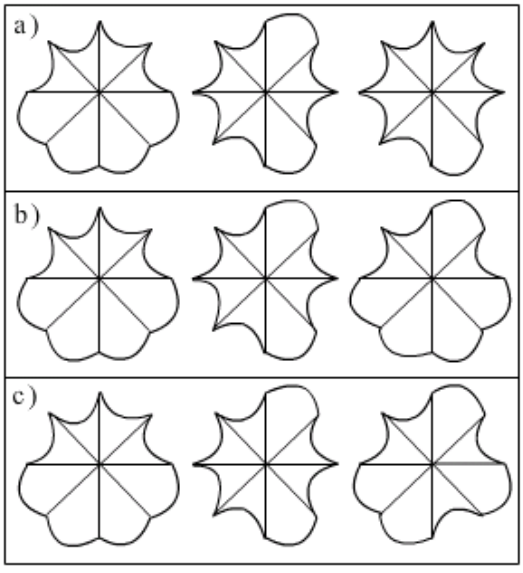



Joonis 24. Sektorringi ülesanne esitatud bittidena, kus värvitud sektor tähistab 1 ja värvimata 0 bitti.

#### *3.2.2.1.1 Binaarloogika teisendused*

Iga loogikatehte jaoks on koostatud tõeväärtustabel, mis sisaldab konkreetse binaarvalemi kõikvõimalikke väärtusi. Testi sooritajal tuleb binaarloogika maatriksi ülesandeid lahendades defineerida enda jaoks kujundi kaks erinevat olekut ning seejärel ülesandes esitatud olekute võrdlemise teel jõudma tõeväärtustabelini. Viimase abil saab juba leida õige vastuse. Testiülesannetes võib teoorias kasutada kõiki olemasolevaid binaarfunktsioone alates loogilisest "JA" ning lõpetades implikatsiooniga. Alljärgnevalt on toodud kolme populaarseima ning tuntuima loogikatehte seosed koos näidetega.

- Loogiline korrutamine (JA, *AND*) on võrdne ühega ainult siis, kui kõik argumendid on võrdsed ühega. Loogilist korrutamist nimetatakse ka konjunktsiooniks (*conjunction*). Konjunktsiooni näiteülesanne on esitatud joonisel 25 (a).
- Loogiline liitmine (VÕI, *OR*) on üks siis, kui kas või üks argumentidest võrdub ühega. Loogilist liitmist nimetatakse ka disjunktsiooniks (*disjunction*). Disjunktsiooni näiteülesanne on esitatud joonisel 25 (b).
- Loogiline antivalentsfunktsioon (Välistav VÕI, *XOR*) on üks siis, kui ainult üks argumentidest võrdub ühega. Välistava "või" näiteülesanne on joonisel 25 (c).

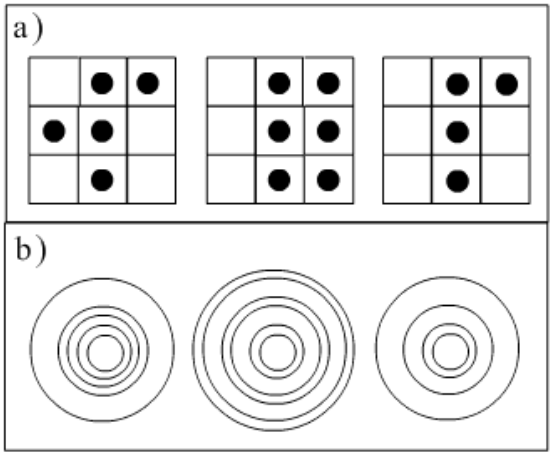



Joonis 25. Näiteülesanded binaarloogika reeglitest, kus väljapoole kaar tähistab bitti 1 ja sissepoole bitti 0. a) Loogiline JA; b) Loogiline VÕI; c) Välistav VÕI

Kahjuks ei saa ülesannetes iga joont või objekti teisendada bitiks. Binaarloogika kasutamise tingimuseks on ülesannetes selgelt eristatavad kaks erinevat olekut. Alljärgnevalt on toodud mõned näited.:

- Konstantne kujundelement nähtav – nähtamatu.
- Kinnine kujundelement seest värvitud – värvimata.
- Kui elemendiks on joon, siis pööratud 90° (joon on horisontaalis – vertikaalis, kaldu paremale - vasakule). Dekoratiivse pöörde reeglit (*rotation*) ei saa enam pärast joone binaarset esitust enam kasutada, sest tekib visuaalne konflikt.
- Kujundi joone kuju (kaar sisse - välja).

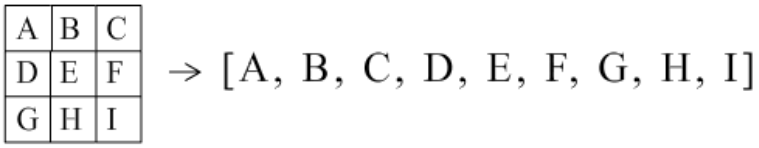
Eksisteerib nõue, et ülesandes oleks iga objekti olemasolu kindlasti aimatav, eriti kui on tegu nähtavuse muutmisega. Näiteks on joonisel 26 toodud kaks ühesugust ülesannet erineva visuaalse esitusviisiga. Pildil (a) on lahendajale koheselt mõistetav, et ühes ülesande ruudus on üheksa täppi, millest osad on nähtamatud. Samas (b) variandis ei pruugi olla selge, et ruudus on tegelikult üheksa ringi. Ehkki programmiliselt võttes on alumine (b) ülesanne igati korrektne, siis visuaalselt on see inimesele liiga raskesti jälgitav - ringide suuruste vahe ei ole piisavalt suur ning järjestikku nähtamatute ringide arv on silmaga mõõtes petlik, sest ei saa aru, kas on nüüd üks või kaks ringi nähtamatud.




Joonis 26. Reegel “Loogiline JA” esitatud kahel erineval viisil.

#### *3.2.2.1.2 Binaarloogika reegli väljund*

Sarnaselt dekoratiivsetele reeglitele on binaarloogika seoste väljundiks 3x3 suurune arvumaatriks. Maatriksi iga arv tähistab vastava ülesanderuudu bitijada. Ülesande info asukohalist paiknemist väljundmaatriksi jadas on esitatud selgitava pildina joonisel 27, kus iga täht tähistab vastava ruudu bittide infot (näiteks A = 110010 jne).



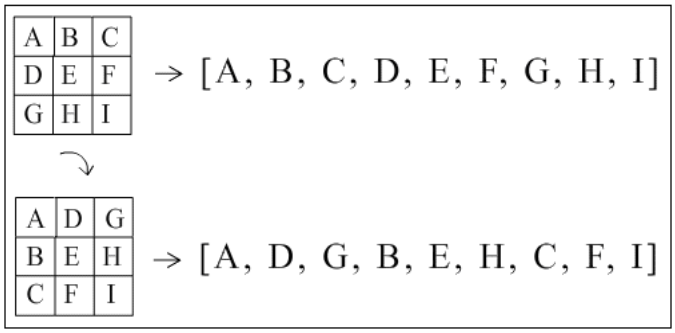

Joonis 27. Bittidejada vastavalt 3x3 maatriksi asukohtadele.

Seega koosneb tulemusmaatriks üheksa elemendilisest jadast, kus iga liige on omakorda bitijada. Kõik ülesande bitijadad on võrdse pikkusega. Näiteks joonise 25 puhul on pikkuseks kaheksa, joonisel 26 aga üheksa. Pikkus on varieeruv lõigus [4, 9]. Välja on jäetud arvud ühest kolmeni, sest väiksematega ei puugi ülesanded tulla üheti mõistetavad, eriti ühebitistega. Nimelt võib pildil tulla esimesel real *1 + 0 = 1*, teisel *0 + 0 = 0* ja kolmandal *1 + 1 = "?"*. Sellisel juhul võib vastus viimasele olla nii 1 (reegel *OR*) kui ka 0 (reegel *XOR*). Kuna kogumaatriks koosneb üheksast elemendist, siis ei ole mõtet ka suuremaid jadasid võtta - saja sektoriga ring näeb liiga detailne välja ning nõuab ülemäärast tähelepanu ja aega ülesande lahendamisel.

Väljundmaatriksi leidmine sõltub binaarloogika reegli rakendamise suunast. Kuna suundasid on kolm, siis jaotub tegevuskäik samuti kolmeks. Kõige lihtsam on leida ridamisi rakendatud loogikareegli tulemit. Nimelt tuleb juhuslikult valida ülesande ridade esimestele ruutudele soovitud pikkusega bitijadad. Seejärel saab õiged bitijada viimasesse tulpa arvutada reegli rakendamisel esimese ja teise tulba väärtuste vahel. Tuleb jälgida, et ridadesse ei satuks ühesugused arvukombinatsioonid, näiteks $A + B = C$ ja $B + A = C$. Samuti on vaja, et tehete muutujad oleksid omavahel erinevad, st $A \neq B$. Vastasel juhul ei pruugi teatud väärtuste korral ülesandest välja lugeda rakendatud reegli terviklikku tõeväärtustabelit.

Tulpade suuna kasutamisel leitakse väljundmaatriks analoogselt ridade võttele. Maatriksis on vaja vaid konkreetsed positsioonid umber paigutada (joonis 28). Kuna mõlemad reegli

rakenduse suunad on sarnased, siis ei ole käesolevas töös tulpade suuna tulemuse arvutamist pikemalt seletatud.



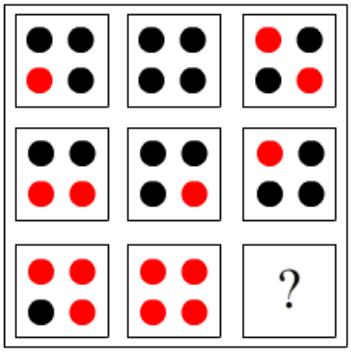

Joonis 28. Ridade ümbertõstmine tulpadeks.

Ridade ja tulpade üheaegse suuna kasutamise korral võib reeglile *X* valida kõrvale teise reegli *Y*, kusjuures on lubatud *Y* = *X*, st võib kasutada mõlemas suunas sama reeglit. Mõlema suuna kasutamisel on juba raskem leida maatriksisse sobivaid väärtusi, sest seose *X* jaoks ülesande ridadesse leitud numbrid ei pruugi kehtida reegel *Y*-ga tulpades. Lahenduseks on valida maatriksisse neli bitijada kohtadele A, B, D ja E (joonis 27). Antud positsioonid on sõltumatud, st nendele kohtadele valitud väärtused määravad ülejäänud maatriksi positsioonide tulemused (joonis 23). Kuna esialgu valitud neljast väärtusest moodustub kokku kuus võrrandit (kolm veerus ja kolm reas), siis on sobiva arvukomplekti leidmise õnnestumise protsent üsna väike. Suurem osa neljast pikast bitijadast koosnevad kombinatsioonid tekitavad võrrandites konflikte. Seetõttu tuleb kogu ülesanne taandada alamülesanneteks, st leida sobivad ühebitised alamhulgad, mille puhul konflikte ei esine. Kuna ühebitiseid nelikuid on vaid kuusteist tükki, saadakse sobivad variandid läbivaatuse teel. Seejärel tuleb leitute seast järjest võtta tulemused ja need omavahel kokku kombineerida. Lihtsustatuna võetakse tulemus ja jaotatakse üheksa ruudu peale laiali, seejärel võetakse järgmine jne.

#### *3.2.2.1.3 Binaarne bitilülitus*

Bitireegleid saab rakendada ka jadamisi, mustri $A \rightarrow B \rightarrow C$ alusel. Nimelt liikudes ülesande ridades vasakult paremale, võib muuta ruutudes mõned bitid teise väärtuse peale. Bitt 1 asemele tuleb 0 ja vastupidi ehk teisisõnu sees olev bitt lülitatakse välja ning väljas olev sisse. Binaarse bitilülituse näiteülesanne on toodud joonisel 29. Esitatud ülesandes tuleb ülesanderuut jaotada omakorda neljaks (2x2 maatriks). Igas saadud väikeses ruudus on üks

ring. Kui ring on musta värvi, siis asub maatriksi bitijadas samal kohal väärtus 1. Punane ring tähistab maatriskis vastavat kohta bitiga 0. Seega vahetatakse bitimuutusel vastaval kohal oleva ringi värvi.




Joonis 29. Binaarlüliti bitimuutus ridades järgmiselt: esimesel korral 3., teisel korral 2. ja 4. bit.

Tegemist on vaid veergudele või tulpadele rakendatava reegliga. Mõlemas suunas korraga samal positsioonil olevate bittide muutmine ei ole visuaalselt märgatav, sest ühes suunas lülitatakse bit teise olekusse, järgmises suunas aga tagasi esimesse olekusse. Binaarse bitilülituse väljundmaatriksi leidmine on lihtne: iga rea jaoks tuleb valida juhuslik bitijada, seejärel muuta selles juhuslikult mõned bitid teise olekusse ning saadud resultaadil veel sama tegevust korrata.

### 3.2.2.2 Elementaaralgebra

Elementaarmatemaatika põhiteheteks on liitmine, lahutamine, korrutamine, jagamine. Neid funktsioone saab kasutada maatrikstüüpi testides selliste kujundikomplektidega, kus on võimalik midagi kokku loendada, näiteks nurkade arv. Üldiselt piirdutakse tehetes väikeste arvudega (numbrid lõigus [0, 9]) samal põhjusel, mida on mainitud binaarloogika reeglite peatükis. Nimelt kui inimene on mõistnud reegli olemust, siis suuremate väärtuste puhul muutub ülesande lahendamine mõttetult ajakulukamaks - vaja kontrollida seose kehtivust iga alamosakese peal.

#### *3.2.2.2.1 Elementaaralgebra kujundite muutused*

Abstraktsetel kujunditel ei ole palju parameetreid, mida saaks kasutada elementaaralgebra seostega. Põhjuseks on kujundite visuaalsete omaduste hindamine. Nimelt võiks defineerida objektide piirjoonte jämedused pikslites ning seejärel neid liita või lahutada. Kuid visuaalselt ei ole võimalik täpselt kindlaks teha, kas joon on 3 pikslit või 4 pikslit lai. Seetõttu tuleb piirduda kergesti loendatavate tunnustega. Alljärgnevas tabelis 2 on välja toodud peamised elementaaralgebra seoste poolt muudetavad kujundite omadused.

Tabel 2

**Elementaaralgebra reeglid**

| Nimi | Kirjeldus | Piirangud | Näide |
|---|---|---|---|
| Nurkade arv | Kõikidel nurksetel kujunditel saab kokku lugeda tippude arvu. | Piirang: [3,9]. Antud reegli rakendamisel saab kasutada vaid positiivseid arve alates kolmest, sest väiksema või negatiivse numbriga ei saa joonistada ühtegi hulknurka. |  |
| Kujundi alamosakeste arv koos positsiooni märgiga | Alamosakesi võib jaotada kahte gruppi - osad paiknevad näiteks kujundi sees, teised väljas. | Antud juhul on tegu erandiga, kus saab kasutada ka negatiivseid arve. Seega toimub väärtuse valik lõigus [-9, 9]. Märk numbri sees määrab, millisesse gruppi osakesed kuuluvad. |  |

| | | | |
|---|---|---|---|
| Elementide arv kujundikomplektis | Määrab, kui palju tervikelemente on ühes maatriksi ruudus. Kujundi tüüp ei oma mitte mingit tähtsust. | Arvudeks on suvaline number lõigus [0, 9]. |  |
| Sektorite arv | Osakestest kokku moodustunud tervikelemendi kujunditel on võimalik kokku lugeda sektorite arvu. | Kasutusel on ainult positiivsed arvud lõigus [1, 9]. |  |

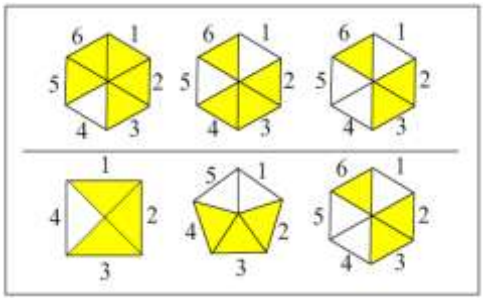
#### *3.2.2.2.2 Reegli väljundmaatriks*

Sarnaselt kõigile eelpoolkirjeldatud reeglitele on elementaaralgebra seoste rakendamise esialgseks väljundiks 3x3 maatriks arvudest. Olenemata reegli kasutusviisist (summa või järjend) tuleb alati jälgida, et maatriksi väärtused ei ületaks eespool toodud tabelis määratud piirangulimiite. Näiteks ei tohi nurkade loendamise seose tulemusse sattuda number 2 jne.

Esimene reegli rakendusviis ehk summa meenutab natuke binaarloogika seost. Sarnaselt loogikaalgoritmile leitakse konkreetsetele maatriksipositsioonidele sobilikud väärtused, mis soovitud elementaarfunktsiooni(de) rakendamisel annaksid ülejäänud puuduvad väljundi väärtused. Seega, kahest parameetrist tekib kolmas: $A + B = C$. Teine reegli rakendusviis ehk jadad meenutab binaarlülitit. Nimelt valitakse väljundmaatriksis ülesande ridade või tulpade esimesteste ruutude positsioonidele juhuslikud väärtused. Seejärel rakendadakse neile soovitud elementaarmatemaatika funktsioon, näiteks $A \text{ –(*2)→ } B \text{ –(*2)→ } C$. Saadud tulemus kantakse maatriksisse. Viimased kolm väljundi tulemust saadakse viimati leitud resultaatidele uuesti mõne elementaarfunktsiooni tehte sooritamisega.

## *3.3 Reeglite kokkusobivus kujunditega ja teiste reeglitega*

Paljudes ülesannetes on võimalik kasutada rohkem kui ühte reeglit korraga. Paraku ei sobi kõik seosed omavahel. Kõige suuremaks konfliktide allikaks võib lugeda asjaolu, et paljude reeglitega muudetakse samu konkreetseid parameetreid. Näiteks võib sektor-ringis kasutada välisserva joontel analüütilist „Loogiline Ja“ või hoopiski dekoratiivset „3 eri kõverat“. Sellisel juhul kaotaks esimesena kehtestatud seos oma olemuse, sest viimasega on andmed uute väärtustega üle kirjutatud. Kui jagatud infovälja probleem kõrvaldada, siis dekoratiivsed reeglid võivad teiste seostega koos eksisteerida. Kahjuks ei saa sama öelda analüütiliste binaarloogika ja elementaaralgebra tehete kooskasutamise kohta. Nimelt kehtib loogika-algoritme kasutatavates ülesannetes nõue, et igas ruudus oleks sama palju bitte, näiteks ringi värvimisel sektoreid. Rakendades ülesandele lisaks sektorite arvu suuredamise ühe võrra, muutub kogu tulemus mõistetamatuks, kus binaarsest loogikareeglist ei tarvitse visuaalselt enam midagi alles olla. Näiteks on joonisel 30 ülemises reas esitatud tavaline binaarne „Loogiline JA“. Iga kujundi sektorile on antud indeks, mille järgi sektoreid eristada. Joonise alumises osas on samale ülesandele lisatud analüütiline sektorite muutmise reegel. Viimase seose tõttu on „ebavajalikud“ sektorid eemaldatud - vasakpoolseimal kujundil ei ole sektoreid indeksitega 5 ja 6. Seetõttu on tulemuses binaarset loogikareeglit võimatu mõista – osa info on lihtsalt kaduma läinud.

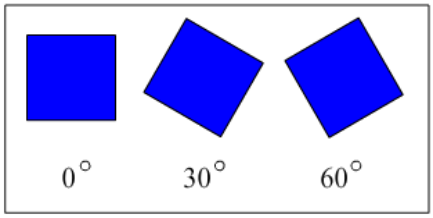


Joonis 30. Ülemise rea bitiülesandele on lisatud kujundi nurkade suurendamise reegel (tulemus alumisel real).

Analoogne info puudulikkuse probleem võib tekkida dekoratiivse introvertse reegli koostöös binaarse nähtavusreegliga. Nimelt on võimalik ülesande objektid bittideks defineerida nähtavuse alusel: nähtamatu tähistab bitti 0 ja nähtav bitti 1. Kui nüüd ülesandes rakendada mõni dekoratiivne reegel introvertselt, siis võib tekkida olukordi, kus ruudustiku mõnedes

osades pole kordagi objekti kasutatud (on nähtamatud ehk bitiga 0). Probleem tekib siis, kui just viimases inimesele lahendamiseks mõeldud osas läheb seda objekti vaja. Sellisel juhul ei ole võimalik teada, milline on õige vastus, sest nähtavatest objektidest ei moodustu läbinähtavat dekoratiivset mustrit. Infot ehk esindatud kujundeid on ülesande lahendamiseks liiga vähe.

Eelpool mainitud konflikte saab arvutis vältida. Paraku aga tuleb paljude probleemide avastamiseks kasutada inimeste abi. Nimelt võivad ülesannetes kaks erinevat reeglit olla programmiliselt igati korrektselt rakendatud, kuid siiski eksiteerib oht, et visuaalselt jääb üks neist seostest märkamata. Näiteks ei ole mõtet kasutada loogikatehteid üheaegselt animeeritud pöörlemisega, sest liikuvaid kujundeid on keeruline omavahel võrrelda. Samuti ei sobi iga kujund kõikide reeglitega. Näiteks ei saa ringi puhul öelda, kas viimane on pööratud või mitte. Programmiliselt oleks muutusi võimalik kontrollida piltidel olevate kujundite piksleid võrreldes. Sellisel juhul laheneks ringi pöörlemise probleem, kuid näiteks keerates ruutu 30 ja 60 kraadi, saame ikkagi visuaalselt vähe eristatava ülesande (joonis 31) .




Joonis 31. Ruudu 30° ja 60° pööretel on esmapilgul raske vahet teha.

Seega tuleb kõiki reegleid omavahel võrrelda ja määrata, kas vastav paar võib ülesandes esineda või mitte. Käesolevas töös kirjeldatud seoste kokkusobivus on esitatud tabelis 3.

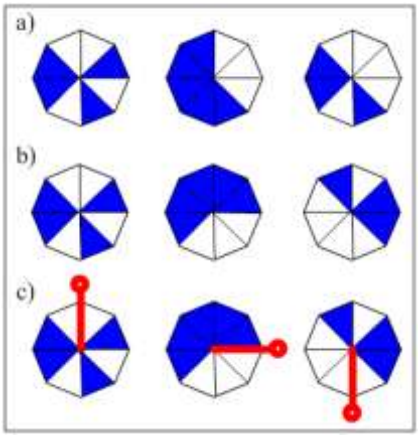
Tabel 3

**Reeglite sobivustabel**

| | Binaarloogika | Bitilülitus | Kõik elementaaralgebra seosed | Kujundi pööramine | Kujundi animeeritud pööramine | Ülejäänud dekoratiivreeglid |
|---|---|---|---|---|---|---|
| Binaarloogika | ? | ? | - | * | - | + |
| Bitilülitus | ? | ? | - | * | - | + |
| Kõik elementaaralgebra seosed | - | - | ? | * | + | + |
| Kujundi pööramine | * | * | * | - | - | + |
| Kujundi animeeritud pööramine | - | - | + | - | - | + |
| Ülejäänud dekoratiivreeglid | + | + | + | + | + | - |

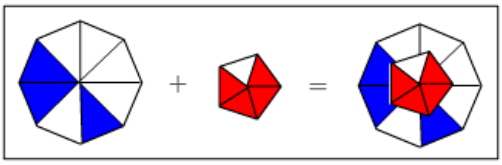
Esitatud tabelis on kasutatud järgmisi tingmärke:

- "-" – reeglid ei või ülesandes koos eksisteerida (näiteks animeeritud pööramine binaarloogikaga).
- "+" – reeglid võivad ülesandes koos eksisteerida.
- "?" – reeglid võivad ülesandes koos eksisteerida vaid juhul, kui ei jagata omavahel ühiseid andmevälju, st mõlemad seosed ei muuda kujundites samu parameetreid.
- "*"– reeglid võivad ülesandes koos eksisteerida vaid juhul, kui ülesandel on küljes mõned lahendamist lihtsustavad lisaelemendid. Näiteks joonisel 32 on binaarloogika probleemist (a) tehtud kaks ühesugust uut ülesannet: (b) ja (c). Mõlemal uuel ülesandel on juures 90° pööramise reegel, kuid viimane on inimesele palju selgem tänu pööret eristavale püstkriipsule.




Joonis 32. Ülesanded, kus reegliteks on a) binaarloogika, b) binaarlogika ja pööre, c) binaarloogika ja pööre koos lisaelemendiga.

Mõned testiülesanded võivad olla ülesannete komplektid, st sisaldada rohkem kui ühte iseseisvat ülesannet (joonis 33). Sellistel juhtudel ei ole eraldiseisvate ülesannete reeglid omavahel põimunud ning reeglitetabel ei kehti. Ühes ülesandes võib kasutuses olla binaarloogika, teises aga animeeritud pööremine.

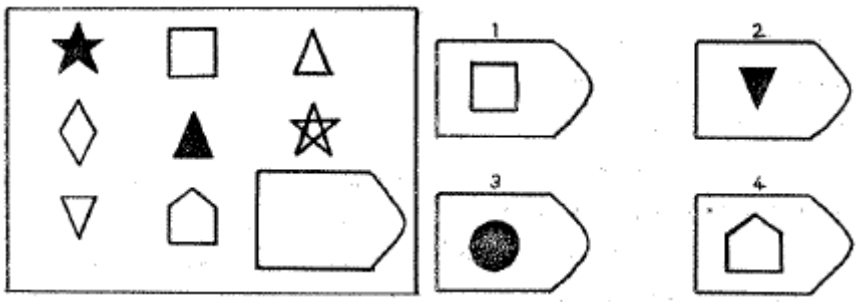



Joonis 33. Vahel on võimalik mitu iseseisvat probleemi üheks ülesandeks liita.

## *3.4 Ülesande keerukus vastavalt reeglile*

Raveni tüüpi testiülesannetes on täheldatud, et osad probleemid on teistest keerukamad. Paljud inimesed on peatunud sellel teemal ning uurinud, millised konkreetsed tegurid muudavad piltidevahelised seosed raskemini mõistetavaks. Lahedajate vigu analüüsides jõudis P. Carpenter järeldusele, et reeglite arv on vastavuses testisooritajate vigadega. Mida rohkem reegleid oli ülesandes rakendatud, seda sagedamini tehti lahendamisel vigu. Põhjuseks toodi välja mitme reegli jälgimise raskust. Töömälu on mitme reegli puhul rohkem hõivatud, sest õige vastuseni jõudmiseks on vaja meeles hoida kõiki leitud seoseid. Rakendatud reeglite arvu suurenemisel kasvab üldiselt ka ülesandes kasutatud kujundite arv või kujundi muutuvate parameetrite arv. Tulemuseks on raskused kujundite grupeerimisel – osadele kehtib üks reegel, teistele teine [7].

Ülesanded teeks kindlasti raskemaks segavate faktorite olemasolu, näiteks mõned asjasse mittepuutuvad väiksemad kujundid. Paraku ei saa ebaolulisi lisaomadusi kasutada, sest lahendaja võib neis siiski märgata mõnd reeglit, mida esialgselt sinna ei planeeritud. Näiteks on joonisel 34 esitatud ülesandes segadust tekitav musta värvi kujundite kasutamine. Toodud ülesande vastusteks võib olla nii 1. valik (neli nurka) kui ka 2. ja 3. (värvitud seest mustaks).




Joonis 34. Kaheti mõistetav ülesanne [18].

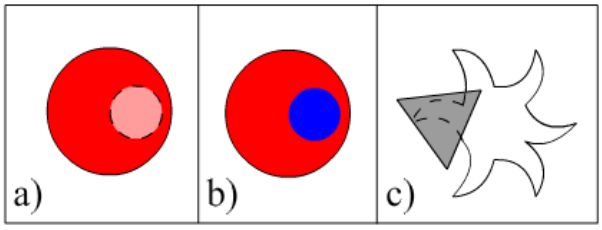
# 4. VISUALISEERIJAD

Peatükkides 2 ja 3 kirjeldatud kujundid ning seosed tuleb kombineerida visuaalselt üheselt mõistetavateks ülesanneteks. Kahjuks ei ole võimalik teste täies mahus programmiliselt iseseisvalt genereerida, sest lõpptulemus võib lahendajale paista liiga abstraktsena. Kui lasta arvutil joonistada juhuslikud kujundid ning rakendada neile reeglitekomplektid nii, et ei tekiks peatükkides mainitud konflikte seoste ja kujundite vahel, siis võivad loodud ülesannetes ilmneda kaks järgnevalt kirjeldatud probleemi.

Kujundite kattuvus

Väikeseid kujundeid ei tohi paigutada suuremate kujundite taha (joonis 35.a), sest lahendaja ei tea alumise kaetud objektiga arvestada. Nähtavuse probleem ei lahene ka siis, kui panna pisemad objektid alati kõige pealmisteks kujunditeks (joonis 35 .b). Nimelt võib juhtuda, et varjatakse suurema kujundi olulised tunnused, näiteks tipud, kaared vms (joonis 35.c). Joonisel (c) kujutatud probleemi saaks lahendada, võttes aluseks kujundite oluliste tunnuste olemasolu. Kuna väiksemal kujundil (kolmnurgal) ei ole ühtegi lisaomadust, siis võib kujund alumiseks liikuda. Kahjuks kerkib oluliste tunnuste kattuvuse probleem uuesti esile, kui kolmnurgal ilmnevad ülesande lahendamiseks vajalikke omadusi.

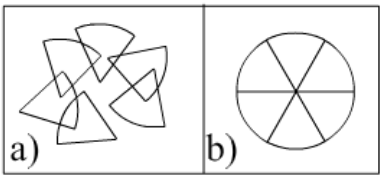
Probleemi üheks võimalikuks lahenduse viisiks oleks igale kujundile prioriteedi seadmine. Määratud väärtus näitab kujundi ruumilise kõrguse (*z-telje*) positsiooni. Paraku nõuab kujundite prioriteetide seadmine inimesepoolset sekkumist. Nimelt tuleb üle vaadata probleemsed kujundikombinatsioonid ning igale kujundile defineerida sobiv kõrgus. Näiteks joonisel (c) võiks kolmnurga kõrguseks seada väärtuse 0 ning tähekesele 1. Joonistades kujundeid kõrguse asukoha järgi, saame tulemuse, kus teatud juhtudel paikneb väiksem objekt suuremast ülevalpool (joonis 35.b) või allpool (joonisel 35.c kujundite kõrgused ära vahetada).




Joonis 35. Kujundite kattuvus.

Enamasti asümmeetrilised ülesanded

Kui paigutada programmiliselt kujundeid ülesande ruutudesse juhuslikult, siis suure tõenäosusega ei pruugi tulemus olla sümmeetriline. Näiteks satub lahendajale väga harva joonisel 36 (b) esitatud pilt, kui kujundelemendiks võtta ringi üks sektor ning kopeerida seda teistele positsioonidele.




Joonis 36. Kuue sektori juhuslik paigutus võib harva anda tulemuseks sümmeetrilise ringi.

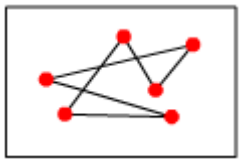
Väga keeruline on maatrikstüüpi testide genereerimiseks implementeerida universaalset arvutisüsteemi. Inimestele visuaalselt aktsepteeritavate ülesannete saamiseks tuleb programmi lisada palju erinevaid tingimusi – vaja on määrata kujundite muudetavad parameetrid (näiteks kõikidel ei ole võimalik muuta servi), omavahel sobivad kujundid jne. Tunduvalt efektiivsem lähenemine oleks lasta programmi kasutajal koostada erinevate ülesandetüüpide mallid (ingl. keeles *template*), mis sisaldaksid ülesandes kasutatavaid reeglite ja võimalike valitavate kujundite gruppe ning kirjeldaks siis täpselt, kuidas neid kujundeid ja reegleid omavahel kombineerida. Näiteks võib reeglitena lubada „loogiline JA“, „bitilülitus“, kujunditena kolmnurka, ringi ja ruutu ning anda kujundite ülesanderuutu paigutamiseks fikseeritud positsioonid. Moodustatud malle võib nimetada visualiseerijateks. Ehkki ülesannete väljanägemine on suures osas inimese poolt määratletud, on tulemused siiski väga mitmekesised. Andmed – mallist juhuslikult valitud abstraktsete kujundite ja seoste komplektid – on programmiliselt genereeritud ja varieeruvad. Kuna igale visualiseerijale saab määrata spetsiifilised ülesande esituse viisid, siis võimaldab see lahendajatele ette anda sisu mõttes sama ülesande, aga igale inimesele visuaalselt eri vormis. Näiteks joonisel 13 esitatud ülesannete puhul ei ole lahendajatel koheselt võimalik naabri pealt õiget vastust maha vaadata.

Järgmistes alampeatükkides on toodud mõned visualiseerijate näited. Sarnaselt reeglite numbrilistele piirangutele (näiteks tippude arv lõigus [3-9]) eksisteerivad visualiseerijates erinevad kitsendused, kus maksimaalseks numbriväärtuseks on üheksa. Kuna terves ülesandes on kujundite näitamiseks üheksa ruutu, siis saaks igas ruudus esitada sama kujundelementi,

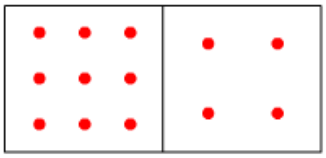
modifitseerides vaid ühte kujundi parameetrit. Näiteks muutub ülesandes kujundi sektorite arv ühest üheksani. Suuremate arvude korral muutuks ülesanne raskesti tajutavaks.

## *4.1 Maatriksi punktidest kujund*

Erinevate punktide vahele tõmmatud sirged moodustavad kujundeid. Kui punktid valida juhuslikult, siis suure tõenäosusega on tulemus väga segane (näiteks joonis 37). Seetõttu on vaja sümmeetrilisema objekti saamiseks fikseerida tippude asukohad. Selleks sobib nö maatriksi paigutus (joonis 38).



Joonis 37. Suvalised punktide asukohad



Joonis 38. Maatriksi paigutus

Maatriksis on võimalik punktide vahele tõmmata jooni kahel erineval viisil:

a) ainult naaberpunktide vahel – ükski joon ei lõiku;
b) suvaliste punktide vahel – jooned võivad lõikuda.

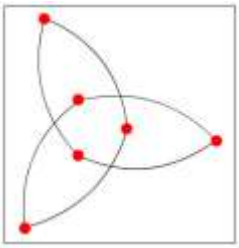
Käesoleva visualiseerija poolt esitatud kujunditele on võimalik rakendada järgmisi reegleid.

- Binaarloogika ja bitilülitus – valides juhuslikult välja kuni üheksa punkti ning tõmmates nende vahele järjest sirgeid, saab ülesandes bittidena defineerida joonte nähtavust: bit 0 – ei ole kriipsu, 1 – on. Kuna sirgeid on võimalik painutada, siis saab alternatiivina bittideks defineerida ka kumerust ja nõgusust või sirge-kumera või sirge-nõgusa paare.
- Maatriksi punktide vaheliste sirgete juures on peamiselt kasutuses numbri suurenemine, st tõmmatavate joonte arv muutub. Ülesannetes on võimalik kasutada nurkade muutmist, kuid sellisel juhul tuleb juurde panna lisakontrolle, et näiteks kolme naaberpunkti vahele tõmmatud kaks sirget ei moodustaks visuaalselt ühte joont.

- Dekoratiivsetest reeglitest on kasutatavad värv, joone paksus, pööramine ja kaare kuju. Tuleb jälgida, et pööratava seose rakendamisel oleks muutus märgatav.

## *4.2 Paaris arv punkte ringis*

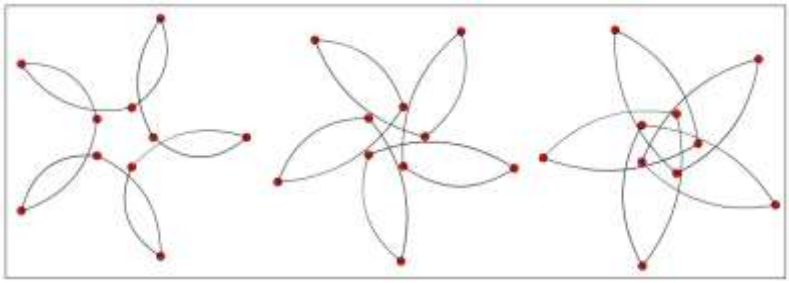
Sarnaselt eelmisele visualiseerijale kasutatakse selle visualiseerija puhul samuti punktide vahele tõmmatud jooni. Üldine paigutus on järgmine: võetakse kaks erineva suurusega ringi ning asetatakse nendele kindel arv punkte võrdsetele kaugustele. Seejärel hakatakse jooni tõmbama vaheldumisi sisemise ja välimise ringi punktide vahele. Illustreeriv näide on joonisel 39.



Joonis 39. Jooned punktide 3*2 vahel.

Visualiseerijas rakendatavad reeglid on järgmised.

- Analüütilistest reeglitest on rakendatavad vaid numbri suurenemise seosed. Reegli arvumaatriksit kasutatakse kujundi joonistamiseks. Maatriksi alusel on joonistamise variante kolm.
  a) Ringil asuvate punktide arv – ülesande ühes ruudus on 3*2, teises ruudus 4*2 kujund.
  b) Samm järgmise punkti valikuks. Alati ei tarvitse järgmise joone tõmbamisega liikuda kohe esimese mittekasutatud punktini, vaid järjestuses võib osa punkte vahele jätta. Erinevad näited on esitatud joonisel 40,



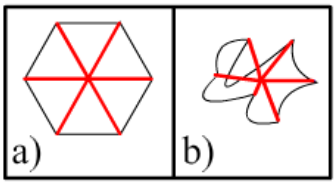
Joonis 40. Jooned 5*2 punktide vahel hüpetega 0, 1 ja 2.

c) Kaugus ringide vahel. Kuna ringide vahesid on visuaalselt täpselt mõõta suhteliselt raske, siis tuleb paika panna piiratud arv erievaid võimalusi, mis on üksteisest kergesti eristatavad.

- Dekoratiivsetest reeglitest on rakendatavad värv, joone paksus, pööramine ja skaleerimine. Koos analüütiliste reeglite (b) rakendamise variandiga kaasnevad siiski mõned kitsendused. Nimelt muutub pöörlemisel ülesanne liiga jälgimatuks – liikuval pildil on punktide vahele tõmmatud joonte uurimine raskendatud.

## *4.3 Kujundi jaotamine osadeks*

Terve kujundi saab jagada osadeks. Näiteks võib ringi tükeldada sektoriteks (joonis 41.a). Osakeste arv $x$ on juhuslik number lõigust [1, 9]. Kui jaotatavat kujundit ei tükelda, siis $x = 1$. Kasutatavateks objektideks võivad olla kõik kinnised kujundid, mille piirjooni ei läbitaks jagava(te) sirge(te)ga. Ebasobiva kujundi näide on esitatud joonisel 41 (b).

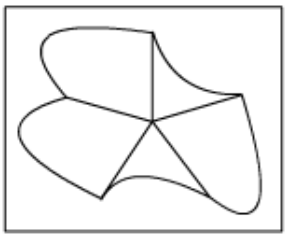



Joonis 41. Alamosakeste jaotamiseks sobilik (a) ja mittesobiv (b) kujund.

Järgnevalt on kirjeldatud reegleid, mida saab kasutada kujundit tükeldavas visualiseerijas.

- Binaarloogika ja bitilülitus – alamosakesi saab seest värvida. Seega saab bitid defineerida järgmiselt: 1 → värvitud, 0 → värvimata osake.
- Analüütiline numbriline suurendamine – alamosakeste arv võib muutuda. Näiteks on ühes ruudus kujund jaotatud kolmeks, järgmises neljaks jne.
- Kõiki dekoratiivseid reegleid saab rakendada terviklikule kujundile. Näiteks on ühes ülesande ruudus kujundi kõik jooned viie ja teises ruudus kolme piksli laiused. Olenevalt jaotatavast kujundist ei pruugi siiski iga dekoratiivne reegel sobida kujundi peale. Näiteks ei saa joonisel 42 esitatud kujundi piirjoontele rakendada dekoratiivset „Kaare kuju“ reeglit. Samuti kehtib nii staatilise kui animeeritud pööramise reegli rakendamisel lisatingimus. Nimelt peab vähemalt üks alamosake olema värvimata ja teine värvitud. Vastasel juhul ei ole muutus märgatav. Seega tuleb kujundis kasutada

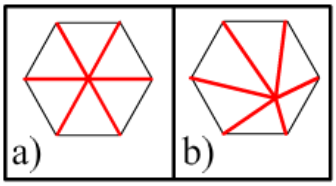
vähemalt kahte osakest. Dekoratiivsetest reeglitest saab introvertselt rakendada ainult värvimist, vahel harva ka läbipaistvust.



Joonis 42. Kõikidele sektoriteks jaotatud kujundites ei saa rakendada dekoratiivset „Kaare kuju“ muutmise reeglit.

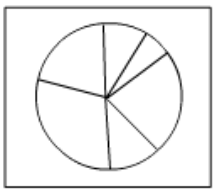
Kujundit on võimalik tükeldada mitmel erineval viisil. Eri võtted omakorda laiendavad või kohati hoopiski piiravad reeglite rakendamist. Alamosakesteks jaotamise gruppe on viis.

1. Punktist hargnemine, tekitades sektoreid. Kuna kujundis saab pärast punktist jaotamist eristada sisemisi ja välimisi jooni, siis on võimalik nendele joontele üheaegselt rakendada erinevaid binaarseid või dekoratiivset kaaremuutmise reegleid. Samuti võib ülesande visuaalselt keerukamaks teha järgmiste võtetega.
    a) Keskpunkt ei asu täpselt kujundi keskel, vaid lähemal mõnele servajoonele (joonis 43). Kindlasti tuleb jälgida, et ei tekiks piirjoone lõikumise probleeme.




Joonis 43. Sektorite jaotuspunkt ei pruugi paikneda kujundi keskel.

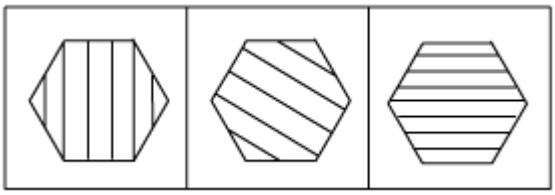
    b) Sektorid on ebavõrdse kraadisuurusega, st mõni on teistest laiem (joonis 44).



Joonis 44. Sektorid võivad olla erineva suurusega.

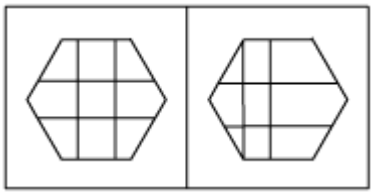
2. Triibud – kujund on jaotatud ribadeks. Ülesannete mitmekesistamiseks võib kasutada triibutamiseks erinevaid kaldenurki (joonis 45). Kui kujundis on jaotusjooned

üksteisest piisavalt kaugel, siis saab sarnaselt eelmisele punktile rakendada tõmmatud joonte peal kaare muutmisega kasutatavaid reegleid. Samuti ei tarvitse kõik triibud olla võrdse laiusega. Mida väiksemad on kujundi mõõtmed (vastavalt ülesande ruudu suurusele), seda hoolikamalt tuleb järgida, et triipude vahe ei muutuks visuaalselt liiga kitsaks. Mida rohkem jooni ehk osasid on kujundil, seda suurem oht on saada ülesandesse visuaalselt kasutuskõlbmatu objekt.



Joonis 45. Kujundi triibutamine erineva kaldenurga all.

3. Ruudustik. Tegemist on triipude analoogiaga. Osa jaotusjooni on tõmmatud vertikaalselt, osa horisontaalselt (joonis 46).



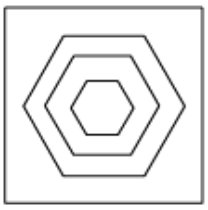
Joonis 46. Kujundi jaotamine ruudustikuga.

4. Juhuslikult tõmmatud jaotusjooned (joonis 47). Sarnaselt eelmistele tükeldamisviisidele tuleb jälgida, et alamosakesed ei tekiks liiga väikesed. Samuti ei ole soovitav muuta tõmmatud sirgeid reeglitega kõverjoonteks. Vastasel korral võib tekkida raskusi mõnede niigi väikese osade äratundmisega.

Joonis 47. Juhuslikult tõmmatud joontega on oht saada liiga väikeseid alamosakesi.

5. Kujundi piirjoonte järgi. Lõpptulemus meenutab kujunditekomplekti, kus ühesugused erinevate mõõtmetega kujundid on asetatud üksteise peale (joonis 48). Kahjuks ei saa sellise jaotusviisi juures kasutada pöörlemise reeglit, sest isegi pärast mõne sektor ära värvimist ei ole kujundi keeramist näha. Sarnaselt triibutamisele tuleb siingi jälgida

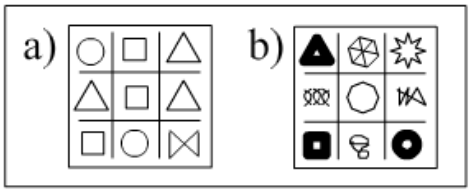
joonte visuaalset paiknemiset. Liiga tihedalt üksteise kõrval olevaid jooni on keeruline silmaga eristada.



Joonis 48. Kujundi tükeldamine kujundi piirjoonte järgi.

## 4.4 Kujundid ruudustikus

Ühte ülesande ruutu võib paigutada üheksa kujundit 3x3 maatriksi laadse paigutusega. Kõikidel kujunditel on seega kindel ruudusisene positsioon. Likvideerimaks objektide kattuvuse probleemi tuleb kõik kujundid teha ühe suurusega: kujundi laius ja kõrgus on kolmandik ülesanderuudu küljepikkusest. Kuna kujundid on siiski väikesed, on kasutusel lihtsamad objektid, näiteks kolmnurk, ring, ruut jne. Vastasel juhul on joonistatud objekti visuaalselt raske tuvastada. Näiteks tunduvad väike ring ja sama suur üheksatipuline hulknurk liiga ühesugused. Samal põhjusel ei saa kasutada väikeste kujundite peal joonte laiuste varieerimist (joonis 49).




Joonis 49. Sobilikud ja lihtsad (a) vs kõlbmatud (b) kujundid.

Visualiseerijas rakendatavad reeglid on järgmised.

- Binaarloogika ja bitilülitus. Bitte võib väärtustada kahel erineval viisil.
    1. Nähtavus - kuna iga kujundi jaoks on olemas oma koht, siis märkab lahendaja ülesannet uurides peatselt, et vastaval positsioonil võiks olla kujund isegi siis, kui see on puudu. Seega saab bitid defineerida järgmiselt: 1 → nähtav, 0 → nähtamatu objekt.
    2. Dekoratiivsed parameetrid - võimalik on bittidena kasutada ka kahte erinevat dekoratiivse reegli parameetrit. Näiteks defineerida bittideks kujundi värvid või

kujundi tüübid (ruut = 1, ring = 0 vms). Dekoratiivsete reeglite parameetrite bittidena kasutamise korral ei või ükski kujund olla nähtamatu.

- Analüütiline numbriline suurendamine kahel erineval viisil.
    1. Kujundite arv - binaarsete reeglite kasutamisvõimalusest loobumisel saab igas ruudus määrata, kui palju kujundeid seal sees on. Objektid võivad olla jaotatud juhuslikult (positsioon ei ole tähtis) või siis järjest alates mõnest nurgast. Näiteks ülemisest vasakust, täites esmalt terve veeru ning alles seejärel alustades uuelt realt.
    2. Hulknurkade tippude arv – igas ruudus võib olla kasutatud erinevaid hulknurki. Näiteks on ühes ülesanderuudus vaid kolmnurgad, teises ristkülikud. Piiranguks on maksimaalselt kuus nurka. Vastasel korral tundub väiksete mõõtmetega kujund kaugelt vaadates pigem ringina.
- Kõik dekoratiivsed reeglid, välja arvatud joone paksus ning pööramise introvertne kasutamine. Kujundite piirjoonte laius ei oleks probleemiks suurte ülesande ruutude ehk joonistatava ala korral. Samuti saaks heal juhul igat pööratavat kujundit keerata. Paraku ei ole väikeste mõõtmetega kujunditel võimalik visuaalselt selgesti eristada selle detaile. Seetõttu tuleb nendest seostest loobuda, ehkki programmiliselt oleks kõik korras.

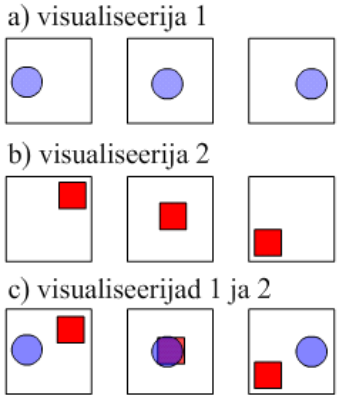
## *4.5 Individuaalsed kujundid*

Ülesande ruudu sees võib kujutada ainult ühte abstraktset kujundit. Kasutada võib igasuguseid kujundeid alates tavapärastest ringidest ja ruutudest, lõpetades supervalemiga saadud objektidega. Kuna kujundeid on ruudus vaid üks, siis võib kasutatava objektiga suhteliselt paindlikult ringi käia. Näiteks võib muuta kujundi positsiooni või suurust. Erinevalt teistest visualiseerijates ei ole käesolevas visualiseerijas vaja arvestada ülejäänute ülesanderuutu jagavate kujunditega. Kujundite valik ruutudesse võib toimuda nii juhuslikkuse alusel kui ka konkreetsete piirangutega. Näiteks joonistatakse etteantud arvu 6 põhjal kuusnurk või kuuetipuline täht.

Visualiseerijas saab rakendada järgmisi reegleid.

- Kõik dekoratiivsed reeglid. Siiski tuleb tähelepanu pöörata kujundi keeramise sobivusele ning vajadusel sellele juurde lisada mõni abistav objekt, näiteks ringile üks sirge.

- Kahe ruudu liitmise/lahutamise reegel. Tegemist on ühega vähestest visualiseerijatest, kus on võimalik kaks kujundit kokku panna või hoopiski tükkideks jaotada. Pealmine objekt võiks pooleldi läbi paista või olla seest värvimata, et lahendaja näeks ka alumise kujundi olemasolu, vältimaks probleemi, et üks varjab teist. Reegli miinuseks on asjaolu, et objekti ennast ei või enam nähtavuse seosega muuta.
- Analüütilistest reeglitest on kasutatav vaid kujundi nurkade või sektorite arvu muutmise seos.
- Visualiseerija dubleerimine, st ülesandesse võib kokku panna kaks (harva ka rohkem) erinevat üksikut objektireeglite komplekti. Näiteks on joonisel 50 esitatud eraldi ruudu ja ringi visualiseerijad. Joonisel (c) on visualiseerijad kokku liidetud. Seose rakendamise tingimuseks on järjekordselt ühe kujundi läbipaistvus teise suhtes. Ülesanderuudu liitmise või lahutamise reegel taandub põhimõtteliselt samuti käesolevasse gruppi, kui erandkorras kasutada kujundi täieliku nähtavuse muutmist – ühes ruudus on nähtamatu, teises nähtav ning kolmandas pooleldi läbipaistev.

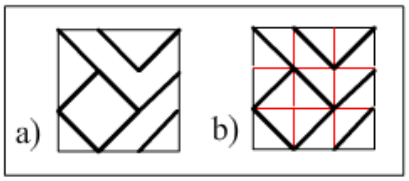



Joonis 50. Visualiseerija dubleerimine.

## *4.6 Kaldega kriipsud maatriksis*

Ülesande ruut on jaotatud üheksaks või neljaks alamosaks, st vastavalt 3x3 või 2x2 maatriksiks. Iga ruudu sees on üks diagonaalne sirge, mis võib olla kaldu vasakule või

paremale. Üheksaelemendiline ruut on lahendajale piisavalt segane ning vajab seetõttu abijooni, et osataks eristada ruudustikku (joonis 51).



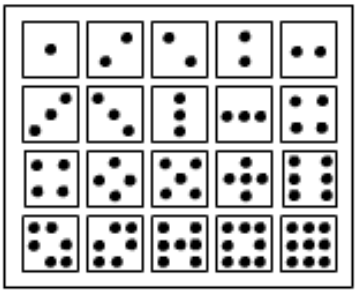

Joonis 51. Kaldega kriipsud maatriksis – a) abijoonteta, b) abijoontega

Visualiseerijas saab rakendada järgmisi reegleid.

- Binaarloogika ja bitilülitus – sarnaselt „kujundid ruudustikus“ visualiseerijale saab kaldkriipsude nähtavust kasutada bittidena: 1 → nähtav, 0 → nähtamatu objekt. Teise alternatiivina on võimalik bitid defineerida vastavalt kalde suunale: 1 → vasakule (\), 0 → paremale (/). Mõlemat varianti – nähtavus ja kalle – saab korraga rakendada vaid tingimusel, kui kalded on dekoratiivse reeglina, st mõnes ruudus on kõik kriipsud joonistatud ühele poole, teistes aga teisele poole.
- Analüütilistest seostest võib käesolevale visualiseerijale rakendada vaid vähtavate joonte arvu suurendamist. Valikuvõimaluste piiramise põhjuseks on sirgjoone kasutamine kujundina. Kriipsud võivad olla jaotatud juhuslikult (positsioon ei oma tähtsust) või järjest alates mõnest nurgast. Näiteks ülemisest vasakust, täites esmalt terve veeru ning alles seejärel alustades uuelt realt. See välistab pea kõigi muude reeglite sisuka kasutamise.
- Dekoratiivsetest reeglitest on kasutatavad värvi, joone paksuse ning kaare kuju muutmine (kriips võib ka kumer olla).

## *4.7 Täringu paigutus*

Ühte ülesanderuutu on mitme kujundi paigutamiseks eelnevalt välja toodud maatriksi lahendused. Teise võimalusena saab kasutada nö täringupaigutust. Kujundid asetatakse ülesanderuutudes mängutäringu täppide positsioonidele. Joonisel 52 on tuntud täringule lisatud juurde mõned sobilikud variandid.



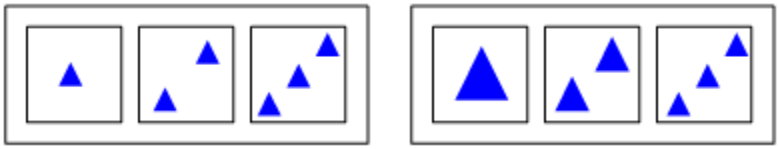
Joonis 52. Erinevad täringupaigutuse viisid

Visualiseerijas saab rakendada järgmisi reegleid.

- Analüütilistest reeglitest sobib ainult kujundite arvu suurendamine. Kui objekte valida dekoratiivse kujundivaliku seosega ja selle tõttu satuvad juhuslikult ülesannetesse erinevad hulknurgad, siis võib tekkida mulje, nagu oleks tegu kujundite nurkade arvu muutmisega. Tegelikult on ülesanne petlik. Käesolev reegel jääb siiski pigem teiste visualiseerijate jaoks.
- Kõik dekoratiivsed reeglid koos üldiste piirangutega. Välja jääb standardne kujundi suuruse muutmise seos, sest suurenenud objektid ei tarvitse ülesanderuutu ära mahtuda.
- Käesoleva visualiseerija puhul on võimalik objektide suurustega manipuleerida dekoratiivsest „kujundi suuruse“ muutmise seosest erinevalt. Nimelt täiesti omaette reeglina on rakendatavad järgmised võimalused.

1. Kõik kujundid on ühesuurused olenemata palju neid ruudus on. Üksik kujund on igal pool sama suur kui objektid üheksakesi ruudus;
2. Kujundid muutuvad väiksemaks sõltuvalt objektide arvust ülesanderuudus. Mida rohkem objekte on ruudus, seda väiksemalt need ojektid joonistatakse.

Kirjeldatud kuundite suuruse muutmise seoste näiteülesanded on joonisel 53.



Joonis 53. Kujundi suurusega manipuleerimine.

# 5. TERVIKLIKU ÜLESANDE KOOSTAMINE

Terviklike ülesannete koostamisel on võimalik lähtuda kahest erinevast lähenemisviisist: visualiseerijapõhine või reeglitepõhine meetod. Esimesel juhul võetakse aluseks konkreetne visualiseerija ning rakendadatakse sellele erinevaid visualiseerijas kindlaks määratud reegleid. Teisel juhul valitakse välja omavahel sobivad reeglid ja seejärel otsitakse üles visualiseerija, mille peal oleks võimalik soovitud seoseid kehtestada. Käesolevas töös on vaadeldud visualiseerijapõhist lähenemist.

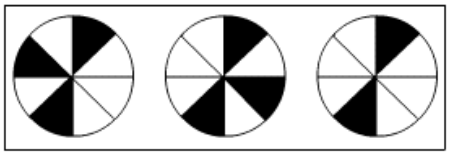
Ülesande koostamine toimub visualiseerijapõhisel lähenemisviisil kolmest järjestikkusest põhietapist:

- visualiseerija valimine,
- reeglite rakendamine visualiseerijas,
- valikvastuste genereerimine.

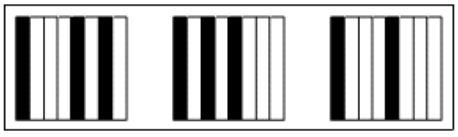
Alljärgnevalt kirjeldatakse iga etappi detailsemalt.

## *5.2 Visualiseerija valimine*

Tegemist on ülesande koostamise kõige väiksema ja lihtsama etapiga. Kõik erinevad visualiseerijad on paigutatud nimekirja, milles tuleb juhuslikult välja valida üks liige. Võib juhtuda, et valituks osutunud visualiseerijal on alamliikmeid. Näiteks kuuluvad joonistel 54 ja 55 esitatud sarnased ülesanded ühte „Kujundi jaotamine osadeks“ visualiseerija gruppi – reeglid on samad, kuid välimus on erinev. Seega tuleks alamgrupi liikmete vahel teha uuesti juhuslik valik.



Joonis 54. Sektor-kera reegliga „loogiline JA“



Joonis 55. Tulbad reegliga „loogiline JA“

Iga ülesande koostamisel läbitakse täpselt ühesugused etapid. Seetõttu võib juhtuda, et kõikide testiülesannete peale kokku on mõni visualiseerija tüüp esindatud teistest sagedamini. Selle vältimiseks võib kasutatud visualiseerija eemaldada valikute nimekirjast.

## *5.3 Reeglite rakendamine*

Reeglite rakendamine konkreetsele visualiseerijale jaguneb neljaks komponendiks:

1. kasutatavate reeglite nimekiri,
2. ülesandes rakenduvate reeglite arv,
3. valitud reegli rakendamine visualiseerijas,
4. järgmise reegli valimine ning selle rakendamine punkti 3 alusel.

Lühidalt toimub kogu protsess järgmiselt: visualiseerijas rakendatakse järjest ükshaaval juhuslikult valitud reegleid seni, kuni enam ei saa ühtegi uut reeglit lisada või on ülesandes soovitud reeglitehulk juba kasutatud.

Alljärgnevalt kirjeldatakse iga rakendusprotsessi komponenti detailsemalt.

Visualiseerija reeglite nimekiri

Iga visualiseerija puhul on eelnevalt kindlaks määratud sellega koos rakendatavad reeglid. Niiviisi saab ülesannetes vältida võimalikke visuaalseid konflikte ja raskesti jälgitavaid kujundi-reegli paare. Näiteks ei ole otstarbekas kasutada „Kaldega kriipsud maatriksis“ visualiseerijas animeeritud pööramist. Seega saab ülesannetes rakendada ainult visualiseerija reeglite nimekirjas olevaid seoseid. Nimekirjas ei ole märgitud, kas erinevad seosed sobivad omavahel üheaegseks kehtestamiseks. Sobivuse kontroll toimub hilisemas etapis.

Esialgne reeglite rakendusarv

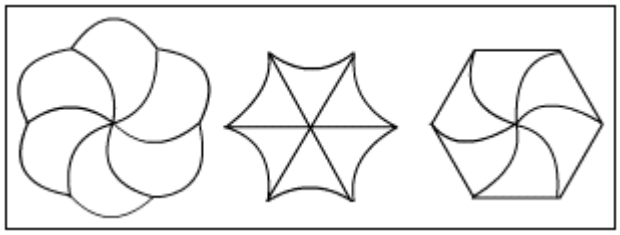
Üldiselt saab visualiseerijates rakendada rohkem kui ühte reeglit samaaegselt. Kasutatavate reeglite teoreetiline arv on väärtus lõigust [1, $k$], kus $k$ on visualiseerija poolt ette antud reeglite koguarv. Valitavate reeglite arv määrab suuresti ülesande raskuse. Tegemist on esialgse hinnanguga, sest reeglid ei tarvitse omavahel sobituda. Näiteks ei saa ülesannetes kujundi piirjoonte peal korraga kasutada bittide loogilisi tehteid ning dekoratiivset kaare muutmise seost.

Reegli rakendamine

Visualiseerijates tuleb iga reegli rakendamisel läbida alljärgnevad etapid.

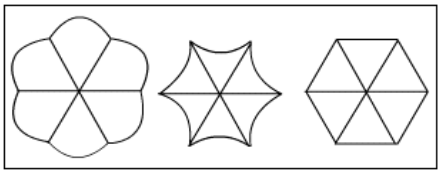
Samm 1. Kõigepealt tuleb juhuslikult võtta üks reegel visualiseerijas määratud seoste nimekirjast. Vältimaks konkreetse reegli korduvkasutust uuel seoste rakendamise ringil, tuleb valitud reegel seoste nimekirjast eemaldada.

Samm 2. Järgnevalt tuleb reeglite nimekirjast eemaldada kõik ülejäänud selle reegliga mitte kokkusobivad võimalused. Kasutades reeglite sobivustabelit, vaadatakse kogu reeglite nimekiri läbi ning eemaldatakse seosed, mis kindlasti ei saa ülesandes eksisteerida koos valitud reegliga. Mõne visualiseerija puhul võib nimekirjas olla mõni reegl topelt või rohkemgi. Näiteks saab sektorringis kasutada „Joone kumeruse“ seost nii sisemiste kui välimiste joonte peal, sest reeglid opereerivad antud juhul erinevate objektidega (joonis 56). Seetõttu on reegel esindatud nimekirjas kaks korda.

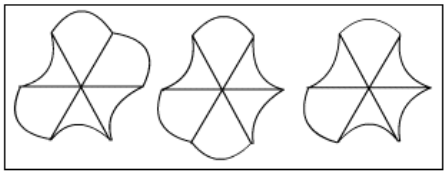


Joonis 56. Sektorring kahekordse „joone kumerus“ reegliga – üks on rakendatud välisjoontele, teine sisejoontele.

Samm 3. Võib juhtuda, et mõne visualiseerija puhul on vaja erandkorras eemaldada reeglite nimekirjast seos, mis tegelikult reeglitetabeli järgi võiks ülesandes eksisteerida üheaegselt hetkel rakendatava seosega. Põhjuseks on reeglitega samade kujundiparameetrite muutmine. Näiteks võib sektorringi välimistele joontele rakendada dekoratiivset „Kaare kuju“ muutmise reeglit (joonis 57). Samas võib kujundi välisjoonte peal kasutada binaarset tehet „loogiline Ja“, kus bit 0 on defineeritud välispidise kaarega ning bit 1 sisemise kaarega (joonis 58). Kuna mõlemad seosed modifitseerivad välisjooni, siis ei saa kirjeldatud kahte reeglit korraga kasutada. Küll aga saaks kujundi välimistele kaartele rakendada „Kaare kuju“ reeglit ja sektorid värvida binaarse „loogilise JA“ tehtege.



Joonis 57. Sektor-ring reegliga „kumer joon“ rakendatuna kujundi välisjoontele.



Joonis 58. Sektor-ring reegliga „loogiline JA“ rakendatuna kujundi välisjoontele.

Kui reegliga saab muuta kujundi erinevaid parameetreid, siis valitakse kasutatav parameeter juhuslikult. Näiteks võib sektor-ringis binaarloogika reegli rakendamise korral kasutada reegli väljundmaatriksi väärtusi sektorite värvimiseks või välisjoonte kaare joonistamiseks. Muidugi saab valida vaid teiste rakendatud reeglite poolt muutmata kujundiparameetrite vahel. Kui ei leidu enam ühtegi hõivamata kujundiparameetrit, siis tuleb reegel vahele jätta ning liikuda järgmise etapi juurde.

Samm 4. Järgmise reegli rakendamiseks on vaja täita kaks tingimust.

a) Esialgne planeeritud reeglite rakendusarv on suurem juba kasutatud reeglite arvust, st ülesandesse on vaja lisada veel vähemalt üks reegel;
b) Leidub veel kasutamata reegleid. Kuna reeglite nimekiri lüheneb iga reegli kasutususega vähemalt ühe võrra, siis ühel hetkel võib juhtuda, et ühtegi ülesandes rakendatatavat reeglit ei eksisteeri.

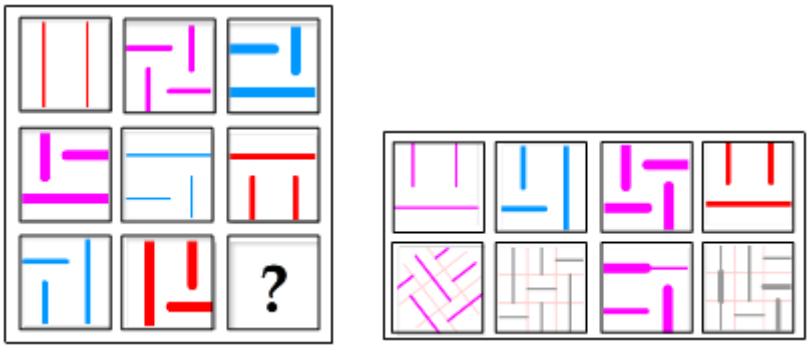
Kirjeldatud nõuete täitmise korral korratakse kogu reegli rakenduse protsessi algusest peale uue reegliga. Vastasel juhul minnakse ülesande koostamise viimase etapi – valikvastuste genereerimise - juurde.

## *5.4 Valikvastuste genereerimine*

Valikute genereerimist saab vaadelda täiesti eraldiseisva ülesande koostamise osana. Ainsaks eeltingimuseks on ülesande (üheksa pildi) olemasolu.

Valikvastuseid on Raveni ülesannete puhul reegline kokku kaheksa, millest üks on õige. Valikud võiksid olla piisavalt keerulised ja sundima lahendajat arvestama kõigi ülesandes rakendatud reeglitega. Näiteks on joonisel 59 esitatud ülesanne, kus on kasutatud järgmisi reegleid:

- dekoratiivne värvireegel kolme värviga: roosa, sinine, punane;
- dekoratiivne joone paksuse reegel kolme erineva joone jämedusega;
- bitlülitus tulpadel järgmiste muutustega: esimesel korral muudetakse teine, kolmas ja neljas bit, teisel korral kõik neli bitti.




Joonis 59. Ülesanne visualiseerijast „Kaldega kriipsud maatriksis“ koos liiga lihtsa valikvastuste komplektiga.

Vaadates joonisel 59 esitatud ülesande valikvastuseid, saab lahendaja koheselt välja elminineerida järgmised ruudud:

- alumisest reast vasakult lugedes esimese, teise ja neljanda - kriipse on ruutudes liiga palju;
- ülemisest reast sinist ja punast värvi ruudud – vastavalt dekoratiivsele värvireeglile;
- allesjäänud kolmest roosast ruudust saab eemalda jämedate joontega valikud – vastavalt dekoratiivsele joone muutmise reeglile.

Pärast nimetatud valikute eemaldamist jääb alles vaid üks ruut: ülemiselt realt vasakult esimene valik. Ühtlasi osutub see ruut õigeks. Sellised valikvastused teevad ülesande oodatust oluliselt lihtsamaks, sest lahendaja ei tarvitse arvestada bitilülituse reegliga.

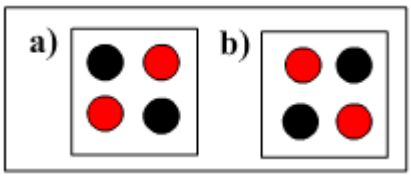
Kirjeldatud probleemi vältimiseks tuleb kasutusse võtta erinevaid valikvastuste koostamise meetmeid. Loodud programm kasutab järgmist võtete kombinatsiooni.

- Valikvastustest maksimaalselt üks variant võib olla ülesandes esitatud muu juhuslikult valitud ruut, va õige vastus. Kui lisada olemasolevaid rohkem, siis võib õige vastuse leidmine osutuda liiga lihtsaks.
- Maksimaalselt neli valikut on sarnased õigele vastusele. Valiku saamiseks võetakse õige vastus ning muudetakse juhslikult mõne reeglite poolt modifitseeritud parameetri väärtust. Näiteks võib joonisel 59 esitatud ülesande puhul olla muudetavaks parameetriks värv, joone paksus või bitid. Soovitav on dekoratiivsete ja analüütiliste reeglite üheaegse kasutamise korral muuta alati kujundil defineeritud bitte. Vastasel juhul tekivad erinevate värvidega, kuid sama kujundiga variandid, mille seast õige leidmine ei ole keeruline. Kui ülesandes on kasutatud mitut reeglit, siis võib muuta rohkem kui ühte parameetrit. Muudetud parameetrite väärtused tuleb võtta ülesandes kasutatud parameetri väärtuste seast. Näiteks ei ole soovitav joonisel 59 esitatud ülesande puhul värvida valikvastuse kujund roheliseks, sest loodud vale vastus muutuks liiga ilmseks.
- Maksimaalselt neli valikut võivad olla juhuslike ülesandes esinenud ruutudepaaride kombinatsioonid. Nimelt võetakse kaks juhuslikku ruutu ning kombineeritakse need omavahel. Tulemuses on mõned parameetri väärtused pärit ühelt ruudult, ülejäänud teiselt ruudult. Sarnaselt eelmisele punktile on soovitav alati muuta bittide väärtusi.
- Ülejäänud valikud on nö mutandid - luuakse uued valikud, kus kõik ülesandes rakendatud reeglitega muudetavad parameetrid on juhuslike väärtustega. Näiteks tuleks joonisel 59 esitatud ülesande puhul võtta „kaldega kriipsud maatriksis“ visualiseerija, muuta juhuslikult bitid, värvida kujund kas roosaks, siniseks või punaseks ning määrata joone paksus.

Kirjeldatud võtted on esitatud prioriteedi järjekorras. Seetõttu võib juhtuda, et esimeste sammudega saadakse kõik valikvastused loodud ning mutante ei lähegi vaja. Samas leidub ülesandeid, kus ei ole võimalik tekitada õigest vastusest nelja erinevat varianti. Sellisel juhul võib tekkida valikvastustesse tühimikke, mis on vaja täita mutantidega.

Iga uue genereeritud valikvastuse korral tuleb kontrollida ega loodud kujund ole sama juba mõne olemasoleva vastusevariandiga. Näiteks võib taoline olukord tekkida joonisel 60

esitatud (a) kujundiga. Muutus on märgata 90 kraadise pööramisega (b), kuid 180 kraadi keeramise korral saadakse esialgsega võrdne tulemus (a).




Joonis 60. Limiteeritud pööamisvõimalusega kujund.

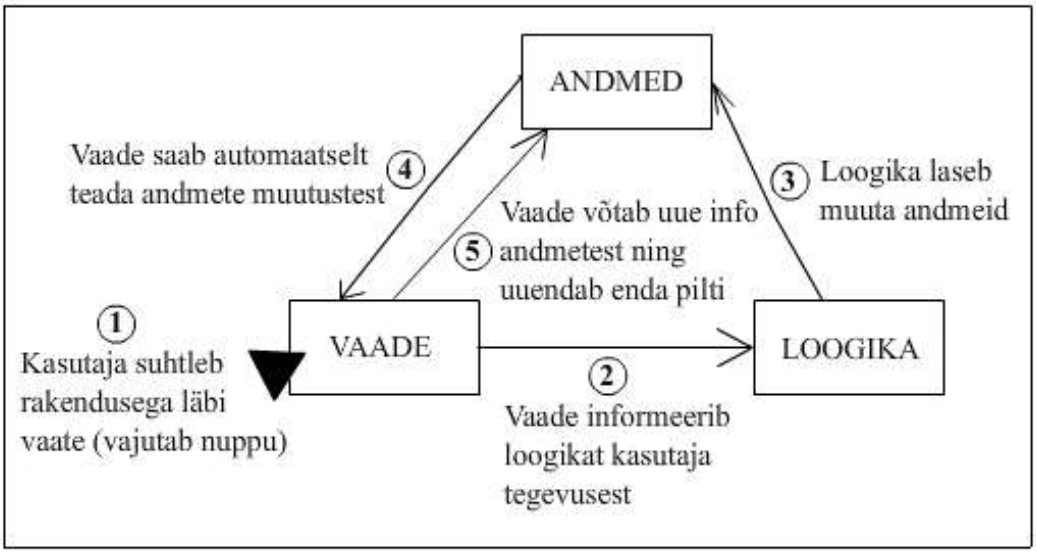
# 6. RAKENDUS

Magistritöö teoreetilise osa kõrvale on valmis tehtud näiteprogramm, kus on rakendatud eelpool mainitud algoritme, kujundeid ja reegleid. Visualiseerijate olemusest ülevaate saamiseks on programmis esindatud kõik visualiseerijad vähemalt ühe reegli ja kujundiga. Vastasel juhul ei ole täpselt teada, kas vastava visualiseerijaga on üldse võimalik genereerida rahuldavaid ülesandeid. Praktilise töö skoobist on välja jäetud ajanõudlike kujundite implementeerimine, näiteks juhuslikult genereeritud kujundi jaotamine osadeks (peatükk 4.3).

Rakendus on valminud Adobe Flash *swf* formaadis ning kood kirjutatud *Actionscript 3.0* skriptikeelega.

## *6.1 Programmi üldine struktuur*

Tegemist on üsna lihtsustatud programmistruktuuriga. Nimelt on rakenduses kasutatud MVC arhitektuuridisaini mustrit (*Model-View-Controller design pattern*). MVC võimaldab üksteisest lahus hoida kasutaja vaate, andmed ja üldise juhtimisloogika. Vastav skeem on esitatud joonisel 61. Selline lähenemine võimaldab rakenduses kergesti juurde lisada või muuta erinevaid reegleid, visualiseerijaid ja kujundeid.




Joonis 61. Üldine MVC skeem.

Loodud rakenduses on MVC disainimustrit mugandatud - loogika osa genereerib valmis kõik vajalikud ülesannete andmed (kujundid, nende paigutus jne) ning annab info edasi vaatele

pärast kasutaja vastavat interaktsiooni (näiteks nupuvajutust „*next question*“). Seega on joonisel 61 toodud skeemis neljas ja viies samm pandud kokku üheks info objektiks, mille loogika edastab vaatele.

Järgnevalt on kõiki kolme programmiosa detailsemalt kirjeldatud.

## *6.2 Kontroller-loogika (controller)*

Tegemist on programmi nö ajuga, mis kontrollib kogu rakenduse käitumist. Üldine skeem on esitatud joonisel 62. Kontrolleri peamiseks tööülesandeks on kasutaja tegevusele vastamine. Näiteks lastakse vaatel kuvada uus ülesanne pärast „*next question*“ nupuvajutust.

Kuna programmi on võimalik edasi arendada, siis on mõned väiksemad tuleviku ideed lisatud loogikaskeemile. Eristamaks neid alamprotsesside töötlemise osasid olemasolevatest, on plaanitavad uuendused esitatud sinise värviga.

Konfiguratsioon (*configuration*).
Vahel on vaja, et programm genereeriks kindlaid ülesandeid. Näiteks kui on soov tekitada vaid ühe konkreetse visualiseerijaga ülesandeid. Selleks tuleks programmile ette anda ülesande joonistamiseks vajalik info. Kuna mitteprogrammeerijatel ei ole programmi koodi lihtne muuta, siis on mõttekas hoiustada kõiki muudetavad andmed programmiväliselt. Üheks võimaluseks on kasutada *XML* formaadis olevat konfiguratsioonifaili. Üldine skeem võiks olla alljärgnev:

```
<questions>
    <question id="0">
        <pattern name = „name of pattern"/>
        <rules>
            <rule name = „name of rule" >
                <shapes>
                    <shape name = „name of shape" values="some values,
                    separated with comma"/>
                    ...
                <shapes/>
```

```
                    <layout name = „name of layout“ values=“some values,
                    separated with comma“/>
                </rule>
                ...
            <rules/>
        </question>
        ...
    </questions>
```

Lahtiseletatuna sisaldavad *XML* lõigud järgmist informatsiooni:

- „questions“ – kõikide ülesannete kirjelduste kogum.
- „question“ – ühe ülesande kirjeldus.
- „pattern“ – visualiseerija nimi. Iga ülesande jaoks saab defineerida vaid ühe visualiseerija.
- „rules“ – ülesandes rakendatavate reeglite kogum. Iga reegli jaoks tuleb määrata seose nimi, soovi korral kasutatavatave kujundite jm lisaandmed.
- „shapes“ – ülesande reeglis kasutatavad kujundid. Iga kujundi puhul tuleb kirja panna selle nimi. Paljudele kujunditele saab määrata täpsustavaid andmeid. Näiteks on võimalik määrata hulknurkade tippude arvu. Vastav info läheb „values“ parameetriks.
- „layout“ – sarnaselt reegliga kasutatud kujunditele võib dekoratiivsete reeglite jaoks määrata seose paigutusmustri (read, tulbad jne).

Kuna konfiguratsioonifail on väliselt muudetav, siis võib selles esineda vigu – valed reeglite nimed, puuduvad andmed vms. Seetõttu tuleb rakenduses alati kontrollida, kas määratud info on kasutatav. Konflikti korral infot lihtsalt ignoreeritakse ning jätkatakse programmi tööd, st ülesandesse valitakse juhuslikult väärtused.

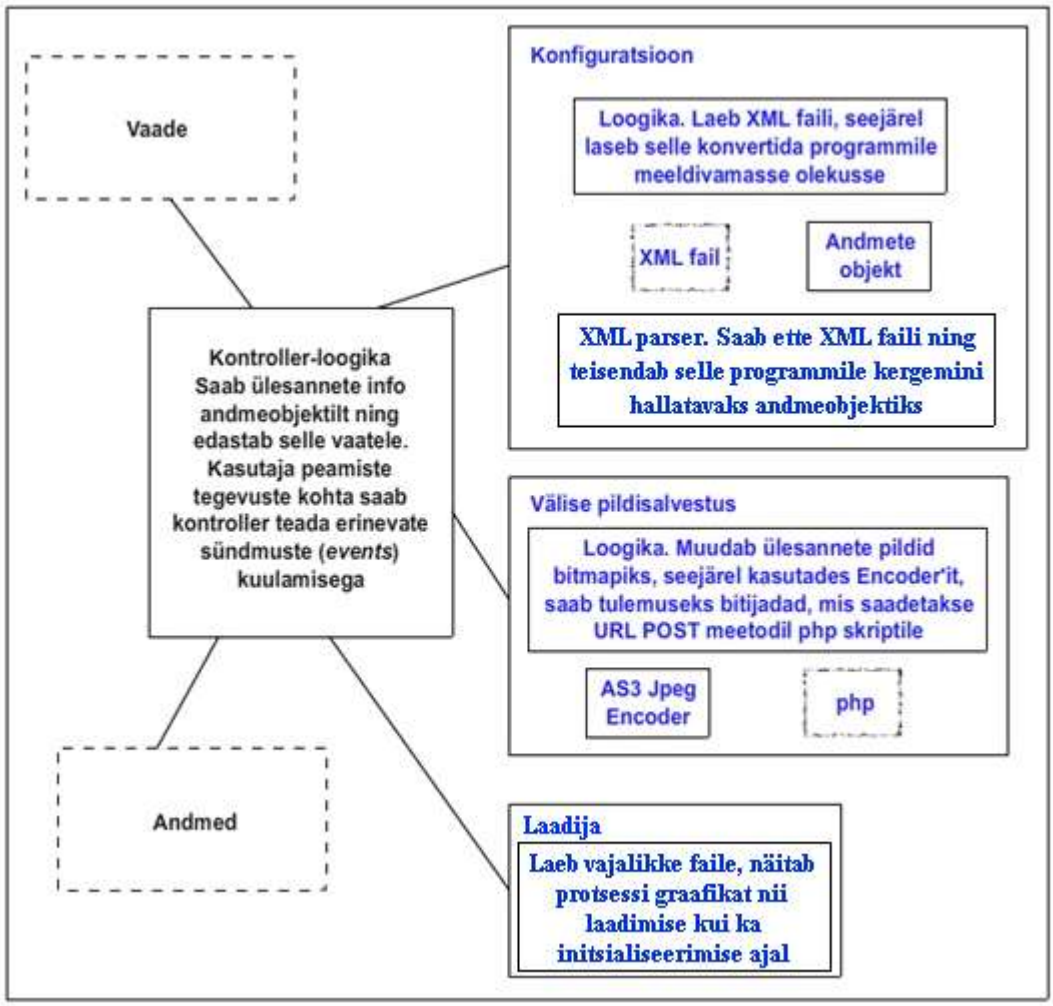
Laadija (*loader*)

Programmi graafika ja lisafailide allalaadimise ajaks tuleks kasutajale näidata pilti koos pidevalt uueneva infoga. Sellisel juhul teaks programmi kasutaja, kui palju kulub veel rakenduse käivitumise lõpuni aega. Tulevikus võib rakenduses näidata 3D vm andmemahult keerukamaid ülesandeid, mille genereerimine võib võtta tavapärasest rohkem aega. Seetõttu

oleks mõttekam programmi käivitamise alguses teha valmis kogu aeganõudvam töö. Vältimaks CPU liigset kasutust, tuleks ülesannete koostamine jaotada erinevate kaadrite peale.

Ülesanete väline hoiustamine (*external jpeg save*)

Ülesannete hilisemaks analüüsimiseks on vaja genereeritud ülesannet hoiustada programmi-väliselt. Flash ise ei ole suuteline vastavat operatsiooni tegema. Seetõttu tuleb kasutada muid abivahendeid. Üheks võimalikuks lahenduseks on järgmine meetod: kogu küsimuse pilt teisendatakse *bitmap* pildiks ning saadetakse *URL POST* andmena serveri skriptile (näiteks mõni *php* skript), mis salvestab info mujale. Flashipoolse tegevuse lihtsustamiseks saab kasutada näiteks „*AS3 Jpeg Encoder*“ nimelist teeki.



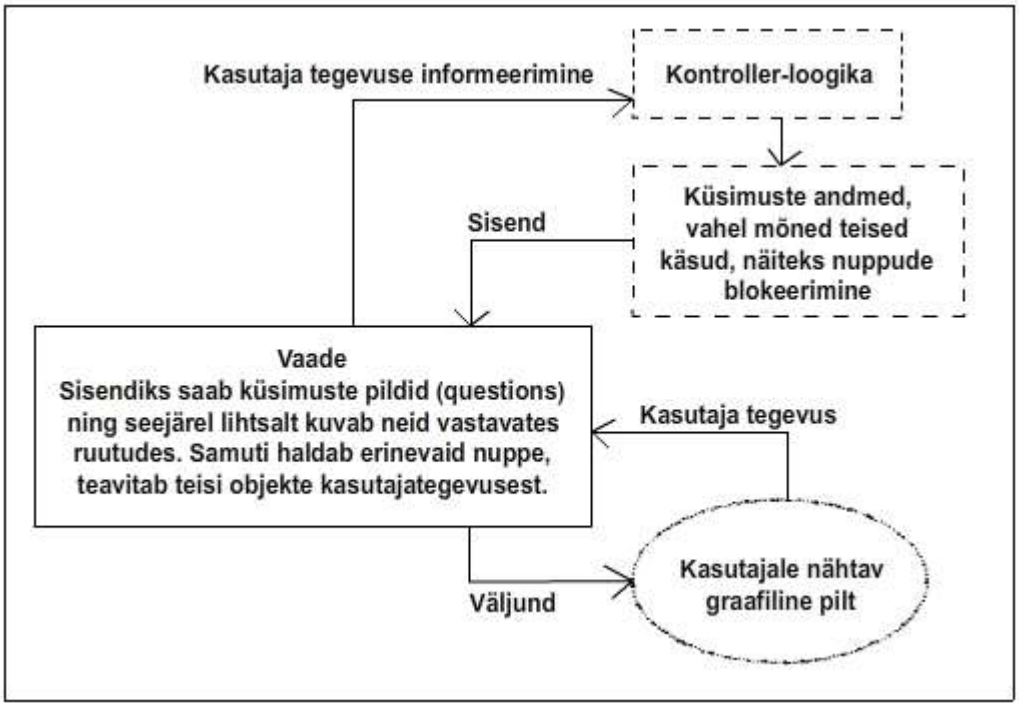

Joonis 62. Üldine kontroller-loogika skeem.

## 6.3 Vaade (viewer)

Vaade on teisisõnu kasutajaliides (*UI – user interface*), mis tegeleb rakenduse visuaalse näitamisega. Vaate peamisteks ülesanneteks on:

- teavitada kontroller-objekti kasutaja tegevusest;
- näidata ülesandeid visuaalselt;
- kontrollida nuppe nii graafilise kui loogilise poole pealt. Näiteks võimaldab nupu välja lülitamist.

Kogu ülesannete näitamiseks vajalik info ning käsud tulevad kontroller-objektilt. Vaatel tuleb lihtsalt kuvada vastav pilt. Seetõttu on tegemist programmi kõige lihtsama osaga, millel puudub keeruline sisemine loogika. Üldine skeem on esitatud joonisel 63.




Joonis 63. Vaate üldine skeem.

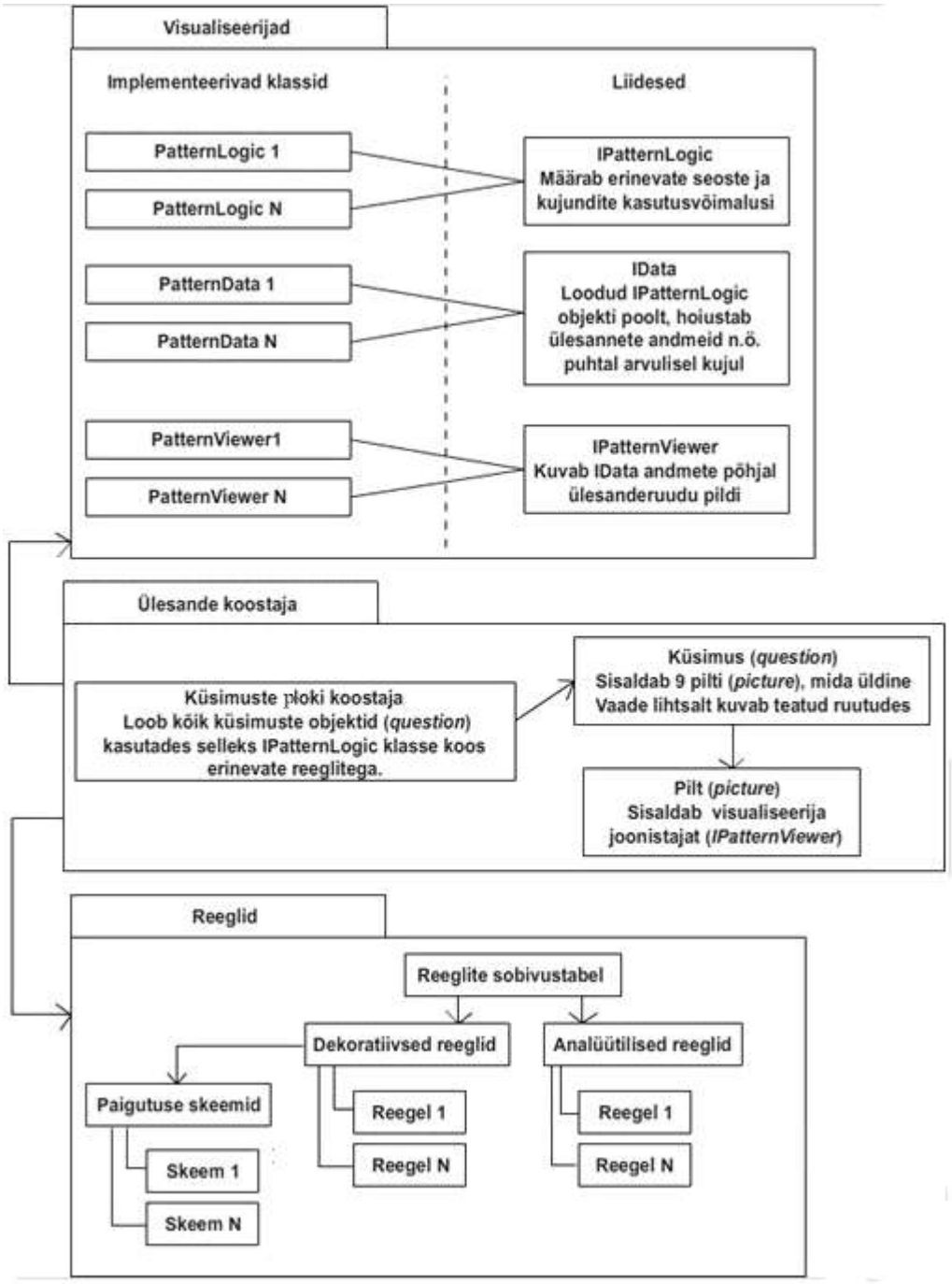
## 6.4 Ülesannete andmed (data)

Tegemist ei ole puhtalt ainult küsimuste hoidlaga, vaid ka nende genereerimise funktsionaalsusega. Võib öelda, et antud komponent on kõige tähtsam ja keerulisem osa kogu programmis. Selle üldine skeem on esitatud joonisel 64. Komponendis võib eristada kahte peamist alamtükki: ülesannete genereerija (peatükk 5) ning genereeritud küsimuste andmete

objektid (peatükk 4). Esimene osa kasutab konfiguratsioonifaili infot ning vajadusel leiab juhuslikud väärtused ise. Eraldi hoitakse reegleid, paigustusskeeme ning visualiseerijaid. Viimaste seas võib omakorda märgata väiksemat *MVC* mudelit.

- Igal visualiseerijal on olemas loogika kasutatavate seoste ja kujundite rakendamiseks (*IPatternLogic*).
- Visualiseerija poolt valmistatud infot hoitakse eraldi andmeobjektis (*IData*). Näiteks on sektror-ringi värvitud sekrorite bitijadad omaette objektis.
- Vastavalt andmeobjektile (*IData*) kuvatakse ülesande pilt visualiseerija joonistaja poolt (*IPatternViewer*).

Liideste (*interfaces*) kasutamine võimaldab ülesande genereerijal erinevaid visualiseerija objekte käsitleda võimalikult anonüümselt.




Joonis 64. Ülesande andmete üldine skeem.

# KOKKUVÕTE

Üldise intelligentsuse mõõtmiseks on loodud erinevaid teste. Üheks populaarseimaks on osutunud maatrikstüüpi geomeetrilised ülesanded, kus lahendajal tuleb mõista ülesandes esitatud abstraktsete kujundite vahelisi seoseid ning seejärel valida õige vastus. Olemasolevad ülesanded on teadaolevalt inimeste endi poolt välja mõeldud. Kuna ülesandeid ei ole palju juurde loodud, siis ei paku sama testi korduvkasutamine inimesele huvi. Pilt-testide automaatseks genereerimiseks aga puudub praegu arvutisüsteem. Käesolevas töös uuriti vastava süsteemi võimalikkust.

Töös anti lühike ülevaade maatrikstüüpi ülesannetesse sobivatest kujunditest. Toodi välja erinevaid võtteid, kuidas arvutiga joonistada testides kasutatavaid elemente.

Lisaks kirjeldati erinevaid reegleid ning nende rakendamisviise. Samuti uuriti nii seoste kui kujundite omavahelist sobivust. Tulemuseks olid kriteeriumid, mis likvideerisid ülesannetest võimalikud seostevahelised konfliktid. Reeglite rakendamise jaoks toodi täpsemad juhised üldisemate algoritmidega.

Abstraktsete kujundite ja reeglitekomplektide visuaalse esitusviisi kohta leiti erinevaid variante. Tervikliku testiülesande koostamisel kirjeldati ülesande genereerimise etappe koos lahenduskäikudega.

Töös esitatud teoreetiliste mõtete ja lahenduste illustreerimiseks valmis käesoleva magistritöö kõrvale maatrikstüüpi ülesandeid genereeriv testiprogramm. Loodud rakendust on võimalik edasi arendada. Nimelt ei tarvitse ülesannetes piirduda tasapinnaliste kujunditega, vaid võiks rakendada erinevaid reegleid ka ruumiliste (*3D*) objektide peal. See mitmekesistaks genereeritud ülesandeid ning aitaks inimestel arendada ruumilist mõtlemist. Samuti võiks kaaluda helidega ülesandeid, kus iga ülesanderuudu peal vajutamine mängiks teatud heli. Näiteks kõlaksid ülesande esimese rea ruutudel kõrged helid ja järgmistel madalamad. Taoline esitusviis sobiks just musikaalsetele inimestele.

# KASUTATUD KIRJANDUS


[1] **Gottfredson, L**. Mainstream Science on Intelligence: An Editorial With 52 Signatories, History, and Bibliography, 1994. American Scientist Vol. 85, No. 5, lk. 440-447.

[2] **Spearman, C**. General Intelligence' Objectively Determined and Measured, 1904. American Journal of Psychology 15, lk 201-293.

[3] **Neisser, U.** Rising Scores on Intelligence Tests. http://www.americanscientist.org/issues/pub/rising-scores-on-intelligence-tests, 1997. (viimati vaadatud 15.06.2010)

[4] **Rushtona, J, Skuy, M, Fridjhon, P**. Performance on Raven's Advanced Progressive Matrices by African, East Indian, and White engineering students in South Africa, 2003. Intelligence, Vol. 31, No. 2, lk 123-37.

[5] **Babcock, R**. Analysis of age differences in types of errors on the Raven's Advanced Progressive Matrices, 2002. Intelligence, Vol. 30, No. 6, lk 485-503.

[6] **Lynn, R, Irwing, P**. Sex differences on the progressive matrices: A meta-analysis, 2004. Intelligence, Vol. 32, No. 5, lk 481-498.

[7] **Carpenter, P.** What One Intelligence Test Measures: A Theoretical Account of the Processing in the Raven Progressive Matrices Test, 1990. Psychological Review, 97, lk 404-431.

[8] **Maltby, J, Day, L, Macaskill, A**. Personality, Individual Differences and Intelligence, 2006. Prentice Hall, lk175 – 179, 281 – 294.

[9] Individual vs Group IQ Testing http://wilderdom.com/intelligence/IQIndividualGroupTesting.html (viimati vaadatud 15.06.2010)

[10] **Allik, J, Mõttus, R, Kööts, L**. Alaealiste ja noorte kinnipeetavate isiksuseseadumuste ja kognitiivsete võimete hindamine, 2009. http://www.just.ee/orb.aw/class=file/action=preview/id=43782/UURIMISRAPORT.pdf (viimati vaadatud 15.06.2010)

[11] **Paul, S.** The Advanced Raven's Progressive Matrices: Normative Data for an American University Population and an Examination of the Relationship with Spearman's g. http://www.questia.com/googleScholar.qst?docId=95186374

(viimati vaadatud 15.06.2010)

[12] **Meo, M, Roberts, M, Marucci, F.** Element salience as a predictor of item difficulty for Raven's Progressive Matrices, 2007. Intelligence Vol. 35, No. 4, lk 359-368 **.**

[13] **Jensen, A.** The g Factor: The Science of Mental Ability (Human Evolution, Behavior, and Intelligence), 1998. Praeger Publishers, lk. 18, 74.

[14] Rose picture
http://mathworld.wolfram.com/Rose.html
(viimati vaadatud 15.06.2010)

[15] **Gielis, J**. A generic geometric transformation that unifies a wide range of natural and abstract shapes, 2003.
http://www.danger-island.com/~dav/code/superformula/Superformula.pdf
(viimati vaadatud 15.06.2010)

[16] **Kunda, M, McGreggor, K, Goel, A**. Addressing the Raven's Progressive Matrices Test of "General" Intelligence.
http://www.cc.gatech.edu/grads/b/bmcgregg/AddressingRavens.pdf
(viimati vaadatud 15.06.2010)

[17] **Hunt, E**. Quote the raven? Nevermore! In Gregg, L.W. ed. Knowledge and cognition, 1974, lk 129–158. Hillsdale, NJ:Erlbaum.

[18] **Lawson, M, Kirby, J.** Strategy training and reasoning, 1979.
http://www.aare.edu.au/79pap/1979_3_5.pdf
(viimati vaadatud 15.06.2010)

[19] Controlling The Real World With Computers
http://www.learn-c.com/boolean.htm
(viimati vaadatud 15.06.2010)

# LISA

## *Lisa 1 – Ülesannete generaator*

CD andmekandja peal on loodud ülesannete generaatori rakendus koos Flash Player-iga. Samuti on programmiga kaasa antud rakenduse arhitektuuri klassidiagrammid.

# Generating Raven Matrices Automatically

Master's Thesis

Marbel Tamm

## Summary

Although most of the IQ tests measure general mental skills, observations have been made that matrix-type tests are one the best because those tests contain various problems that require very varied mental abilities like logical thinking, problem solving, abstract reasoning etc.

The main principle of matrix-type tests is following: a task consists of a 3x3 size table, where abstract objects are placed according to certain rules into each of the eight cells and the ninth cell (bottom on the right) is left empty. The aim of the person taking the test is to understand the patterns and logic contained in the 8 cells in the table and to use that to choose the correct pattern for the ninth cell among 8 given choices.

Because of comfortable and simple usage it is very important to explore the possibilities of composing and elaborating matrix-type tests. To date, all available tasks have been created by hand. There is no computer system to generate tests with pictures. This master's thesis attempts to partially remedy that situation.

In the first part of the thesis the author gave more background information about general intelligence. Also the functional requirements of the generator were described.

The process of generating matrix problems has been divided into four large phases:

- patterns used in tasks
- applicable rules
- combining rules and patterns into tasks
- generating a selection of answers

What kind of patterns to use in tests was examined in the second part of the thesis. A computer system could generate very different shapes, but only few of them are applicable in the matrix-type tests. Different ways of creating acceptable patterns were given.

The third part of the research handles several rules and putting them into practice. This thesis introduced methods of applying those described rules in tests. Accurate instructions with algorithms were shown to enforce the rules. While all rules do not match all the patterns, the suitability of relations and patterns was individually analyzed. Outcome was a group of parameters that resolve potential conflicts between relations in tasks.

The fourth part of the thesis shows, how to combine abstract patterns and relations into unique tasks. Various problems were described which may occur while generating a test, for example overlapped elements. Then a method of creating visually complex but still clear and interesting tasks was proposed.

Last phase of generating tasks described creating complete tests along with 7 false solutions. An algorithm was made to compose selection of answers.

To illustrate the theoretical thoughts and solutions of current master's thesis, a program that generates matrix-type problems was also created following all the principles and ideas laid out in the previous chapters.