Gebruikers documentatie voor *manipulate*  
  
Hans van Staveren [sater@xs4all.nl](mailto:sater@xs4all.nl)

# Inleiding

Het programma *manipulate* manipuleert schema's. Het kan op veel verschillende manieren bestaande schema's aan elkaar plakken, splitsen, vernummeren, draaien, balanceren, uitbreiden enzovoorts. Hoewel het is bedoeld voor een vermoedelijk zeer kleine doelgroep probeer ik het goed genoeg te documenteren dat het bruikbaar is zonder telefoontjes of Emails naar mij.

Manipulate kan schema's lezen in de volgende formaten:

1. ASC formaat, of TXT, of CSV. Deze formaten zijn gedocumenteerd in een apart document.
2. Pairs 4.1 binair formaat. Formaat is niet gedocumenteerd.
3. Deense Bridge Federatie formaat, DBF.
4. MVT formaat van het Engelse programma Jeanie. Niet gedocumenteerd.
5. GEN formaat. Dit beschrijft schema's zoals de tabelletjes in Movements-a fair approach(hierna te noemen MAFA). Gedocumenteerd in een appendix van dit document.

*Manipulate* kan schema's schrijven in de eerste drie formaten. Pairs binair formaat schrijven heeft beperkingen. Plus meer... TODO.

Wat *manipulate* slechts beperkt kan is nieuwe schema's verzinnen(maar zie de stanza optie!). Het is voornamelijk een hulpmiddel voor bewerking van bestaande schema's. Ook bevat het geen interactieve editor. Voor handmatige bewerking is het CSV formaat in combinatie met een spreadsheet programma prima geschikt.

# Installatie

Manipulate is geschreven in de taal *perl*. Iedereen met een Unix/Linux of soortgelijke achtergrond zal dit kennen en hebben. Voor de Windows gebruikers is er een gratis en uitstekende implementatie, genaamd Strawberry Perl.

Download en installeer dat. Start vanuit de groep Strawberry Perl het item Perl(Command Line). Probeer eerst even het volgende commando te typen:  
perl -v  
als dat versie informatie van perl geeft is het programma werkend. Als het niet lukt wordt dat vaak gefixed door in en uit te loggen.

Maak een map waarin je *manipulate* installeert. Als het programma is geinstalleerd open dan een Command Prompt window in die folder(zie ook Aanroep) en geef dan eens het volgende commando:  
perl manipulate gen:end:2

Wat dat betekent leg ik nog wel uit, maar je hoort nu een klein schemaatje als uitvoer op je scherm te krijgen.  
4 2 3 3 0

4 1 A 3 2 A

4 2 B 1 3 B

4 3 C 2 1 C

Als dat werkt kun je in deze map al je schema werk doen, maar er is een alternatief:  
maak een bestand genaamd manipulate.bat en zet dat ergens in je PATH, bijvoorbeeld in de strawberry\perl\bin folder. Dit bestand moet bevatten:  
@ECHO OFF  
perl <volledige pad naar het script 'manipulate'> %\*  
dus als je *manipulate* in de folder d:\werk\schema hebt geinstalleerd wordt dat  
@ECHO OFF  
perl d:\werk\schema\manipulate %\*  
en vanaf dat moment kun je in iedere willekeurige map werkend *manipulate* gebruiken alsof het een standaard commando is. Dat betekent dat je in alle voorbeelden in dit document waarin staat:  
perl manipulate ....  
het woord perl kunt weglaten.

# Globale werking

*Manipulate* leest eerst een of meerdere schema files en plakt die naast of onder elkaar. Daarna kan een extra tafel of tafels worden toegevoegd. Rondes kunnen worden gedraaid, de balans kan worden verbeterd(door middel van het programma balans van Peter Smulders!), verschillende vernummeringen kunnen plaatsvinden, het schema kan wat worden verkleind en daarna kan het schema weer worden gesplitst in deelschema's, en het schema, of die deelschema's, worden dan weggeschreven.

Dit alles gaat ook in die volgorde, en in de volgende sectie zullen de verschillende opties ook in die volgorde worden uitgelegd.

# Aanroep

Het programma heeft geen windows user interface maar wordt aangeroepen vanaf een command prompt. In Unix achtige omgevingen dus gewoon vanaf je favoriete shell, en in Windows vanuit een *Command Prompt* window. Tip: In Windows Vista/7 of later kun je met een SHIFT right-click een command window krijgen in een bepaalde folder. Nederlands “Opdracht venster” geloof ik.

De aanroep zal er zo uitzien:  
perl manipulate [optie of invoerfile] [optie of invoerfile] ...

Als je geen optie hebt gegeven om het resulterende schema weg te schrijven komt het als uitvoer op je scherm. We zullen verderop de uitvoeropties bespreken.

Opties en invoerfiles kunnen door elkaar worden gegeven, maar de invoerfiles worden pas gelezen als alle opties zijn verwerkt. In de praktijk maakt de volgorde dus bijna nooit uit. Invoerfiles die nog niet bestaan, maar waarvoor een recept bestaat worden gemaakt als nodig. Zie hoofdstukje *Recepten*. Invoerfiles worden gezocht in de folder waarin *manipulate* staat en alle subfolders.

Opties bestaan uit een woord, soms gevolgd door argumenten, gescheiden door een dubbele punt. Voorbeeld: split:vertical:7,6 is een optie met twee argumenten: het woord vertical en de lijst(komen we op terug) 7,6

Deze woorden mogen worden afgekort tot bijvoorbeeld spl:v:7,6  
Zolang de afkorting uniek is is hij toegestaan. Als je te ver afkort kun je foutmeldingen krijgen.  
Als je een onbekend woord gebruikt krijg je een lijst van toegestane mogelijkheden.

Soms kan er verwarring ontstaan omdat een invoer file dezelfde naam kan hebben als een optie. Dat kan alleen met opties zonder argumenten. De optie volgen door een dubbele punt zonder iets erachter haalt de verwarring weg.

# Opties algemeen

Er zijn een aantal opties die niet echt onder een sectie horen:

## Minmeeting en maxmeeting

Deze opties geven grenzen aan voor het aantal malen dat deelnemers elkaar mogen tegenkomen. Er wordt bij het inlezen van de schema's uiteraard op van alles gecontroleerd, en deze opties maken het mogelijk grenzen aan toelaatbaarheid te stellen. Als deze opties niet worden gebruikt is het minimum aantal keren dat deelnemers elkaar moeten tegenkomen 0, en het maximum aantal keren dat ze elkaar mogen tegenkomen 1 voor een paren schema, en 2 voor een individueel schema. Voorbeelden:  
minmeeting:1 Alle deelnemers moeten elkaar ontmoeten  
maxmeeting:4 Voor een groot (monster)schema: maximum vier keer tegen dezelfde

## Debug

De mogelijkheid om debugging dynamisch aan te zetten op verschillende onderwerpen en niveaus. Zonder kennis van de code niet te gebruiken. Voorbeeld:  
debug:ascio:8

# Opties invoer

Opties die de invoer regelen. Sommige van deze opties zijn eigenlijk sub-schema's, en worden dan gemaakt/gelezen tussen de andere invoerschema's. Neem bijvoorbeeld de *generate* optie. Dat is in wezen equivalent aan het inlezen van een schema. Dus de aanroep:  
perl manipulate schema1.asc generate:mitchell:5 schema2.mvt  
zal drie schema's gaan combineren, schema1 in ASC formaat, gevolgd door een mitchell van 5 tafels, en dan schema2 in Jeanie formaat. Meer hierover verderop.

## Type

Geeft aan of de schema's paren of individueel zijn. Deze optie is alleen nodig voor schema's waaraan dat anders niet te zien is. In de huidige implementatie geldt dat alleen voor binaire Pairs 4.1 schema's. Voorbeelden:  
type:pairs afgekort ty:p  
type:individual afgekort ty:i

Let wel dat het programma weliswaar beide typen schema's aankan, maar natuurlijk niet gemengd. Ook zijn er een paar opties die alleen werken met paren schema's.

## In

Geeft een invoer file aan. Simpel voorbeeld:  
in:schev16.csv

Dit voorbeeld is volledig equivalent aan alleen de filenaam. Maar de volgende mogelijkheid is:  
in:24indi06.hhj:pairs

Wat naast een filenaam ook een invoertype aangeeft(voor dit voorbeeld zie ook de pairs optie). Maar krachtiger wordt het als je patronen gaat gebruiken:  
in:schev\*.csv

Dit vindt alle schema's van het csv type die beginnen met het woord schev. Zoals gezegd worden schema's ook in alle subfolders gezocht. Als er meer dan één schema matcht krijg je een lijstje te zien en kan je kiezen.

Dit patroon matchen maakt het handig om voor schema's een soort standaardnaamgeving te maken, dat maakt het makkelijk om schema's te vinden die aan bepaalde eigenschappen voldoen.

## Combine

Als het programma meerdere deel-schema's als invoer krijgt worden ze gecombineerd tot één groot schema. De manier waarop is in te stellen. Standaard doet het programma wat ik noem horizontaal combineren. De deel-schema's moeten dan evenveel rondes bevatten, en de tafels worden dan achter elkaar gezet. Als je dat als ASC schema beziet worden de schema's dus horizontaal achter elkaar gezet, het aantal regels blijft gelijk, maar de regels worden langer. Met andere woorden het aantal ronden blijft gelijk maar er komen tafels(en deelnemers) bij.

Ook mogelijk is verticaal combineren. Dan moeten de deel-schema's evenveel tafels hebben, en worden de ronden achter elkaar gezet. Als ASC schema blijven de regels dus even lang, maar er komen er meer. Met andere woorden het aantal tafels blijft gelijk maar er komen ronden(en spelsets) bij.

Voorbeelden:  
combine:horizontal   
combine:vertical

## Barometer

Je kunt van ieder schema een barometer maken met de barometer optie. Geen argumenten. Van alle ingelezen schema's wordt de spelset genegeerd, en in ronde 1 wordt spelset 1 gespeeld aan alle tafels, in ronde 2 spelset 2, enzovoorts.

## Pairs

Om een Pairs 4.1 binair schema te lezen moet je twee dingen doen. Je moet de hierboven genoemde *type* optie gebruiken, en je moet de filenaam in een *pairs* optie gebruiken. Voorbeeld:  
pairs:21SHWL07.HHJ  
In de pairs filenaam zitten twee getallen, het eerste is het aantal deelnemers, het tweede is het aantal ronden. Zonder deze filenaam conventie zijn pairs 4.1 binaire schema's vrijwel onleesbaar. Hoog tijd dus om ze zo snel mogelijk af te schaffen.

Deze optie is er een die, zoals boven genoemd, zich gedraagt als een invoerfile.

## Generate

Het is mogelijk om bepaalde ingebouwde schema's te genereren, ipv in te lezen. De ingebouwde schema's zijn

1. Mitchell. Oneven aantal tafels standaard, bij even de keus tussen relay- en skip-mitchell, en bij de skip-mitchell nog de keus uit twee rondes voor de skip. Voor oneven en skip-mitchells zijn Scheveningen-varianten mogelijk.
2. Expanded Mitchell. Tot 3 rondes expansie.
3. Double Weave Mitchell. Alleen voor viervouden tafels.
4. Web Mitchell. Voor even aantal tafels. Amerikaans schema.
5. Endless Howell. Deze kunnen gemaakt worden voor ieder aantal tafels.
6. Vasneg schema's. Voor aantallen rondes van 5,7,11,13 of 17. Aantal tafels van aantal ronden tot het dubbele. De 1-X en X-X variant(kenners weten wat ik bedoel).
7. Null schema. Een schema met 0 rondes. Bedoeld om “uit te breiden” met de stanza optie.

Voor de Mitchell oneven is het simpel:  
generate:mitchell:7

Het (eerste) argument is dus aantal tafels, en dat is voor alle andere gegenereerde schema's ook zo. Voor een even aantal tafels zijn er drie mogelijkheden:  
generate:mitchell:8  
generate:mitchell:8:3  
generate:mitchell:8:4  
De eerste maakt een relay-mitchell van 8 rondes, de twee andere een skip-mitchell van 7 rondes, waarbij de skip respectievelijk na ronde 3 en ronde 4 is.

Bij de Mitchells zijn geen arrow-switches standaard, NZ paarnummers zijn laag, OW paarnummers hoog. Je kunt ook een variant maken met de spellen vast aan tafel(NZ omhoog, OW omlaag), dat kan alleen met oneven en skip-mitchells:  
generate:s-mitchell:7  
generate:s-mitchell:8:3

Als het aantal tafels een viervoud is kun je ook een Double Weave Mitchell maken. Daarbij wordt niet geleend:  
generate:dw-mitchell:12

De expanded Mitchell:  
generate:expanded-mitchell:8:9  
maakt een expanded Mitchell van 8 tafels, 9 rondes. De arrow-switches volgens MAFA zijn geimplementeerd. *Balans* kan het beter…

Een web-mitchell kun je maken voor ieder even aantal tafels, waarbij het aantal boardsets minimaal de helft van het aantal tafels is, en maximaal één minder dan het aantal tafels. Bij het spelen heb je twee kopieen van de spellen nodig:  
generate:web-mitchell:16:9  
maakt een schema van 16 tafels met 9 spelsets. NZ en OW apart.

De Endless Howell(of Barometer Howell) heeft maar één argument: aantal tafels. Het hoogste paarnummer zit vast aan tafel 1NZ. Rest zoals MAFA beschrijft op pagina 223. Voorbeelden:  
generate:endlesshowell:18  
gen:en:20

Voor de aantallen tafels van 2 tot en met 48 worden de initiele posities gebruikt als in MAFA pagina 224. Maar in meer gecompliceerde situaties gebeurt dat niet. Zie commentaar bij *draai* optie.

De Vasneg schema’s hebben 3 argumenten: aantal tafels, aantal ronden, 1-of-X. Voor een beschrijving van de schema's verwijs ik naar Wekowijzer 116, waarin Marc ze heeft beschreven. Er wordt ook gedraaid in ronde 2 zoals daar beschreven, maar dat verandert als de gebruiker zelf draai aanwijzingen geeft, of balans aanroept. Wordt in die sectie verder uitgelegd. Voorbeelden:  
generate:vasneg:7:7:1  
gen:v:15:11:X

Het null schema:  
gen:null:8

Deze optie is er een die, zoals boven genoemd, zich gedraagt als een invoerfile.

## Matrix

Tijdens het inlezen van alle schema's kan een ontmoetingentabel worden gemaakt. Met behulp van de *matrix* optie kan die in een bepaalde file worden weggeschreven. Altijd een tegenstander tabel, en in individueel ook een partner tabel. Standaard wordt er geen file geschreven. Voorbeeld:  
matrix:matrix.txt

Deze tabellen worden, nogmaals, gemaakt tijdens invoer. Dus als je het schema stevig omwerkt geeft de file matrix.txt de gegevens van de invoer, niet de uitvoer! Als je de nummers van de uitvoer wilt hebben moet je die uitvoer nog een keer door *manipulate* halen.

# Optie stanza

***Experimenteel, nog niet af!***

Het is mogelijk om een schema uit te breiden met extra rondes, met nieuwe boards. De huidige implementatie kan dit al, maar er wordt nog getest. Aan verder testen en uitbreiden zal worden gewerkt, met dank aan Frans Schiereck.

Momenteel werkt het als volgt:  
stanza:7  
voegt aan het huidige schema een extra stanza van 7 rondes toe waarin alle paren tegenstanders krijgen die ze nog niet hebben ontmoet.

# Optie extratafel

Je kunt proberen een schema uit te breiden met één of meer tafels. De manier waarop dat gebeurt is als volgt: met de extra tafel komen twee extra paren(dit kan niet voor individuele schema's dus). Het even extra paar blijft aan de extra tafel zitten, het oneven paar verdringt iedere ronde een ander paar van zijn tafel en stuurt die naar de extra tafel. Je hebt nog enige zeggenschap over welke paren mogen worden verdrongen. Eerst even een simpel voorbeeld:  
perl manipulate gen:end:9 extratafel:1

Dit genereert een Endless Howell van 9 tafels en plakt er daarna een tafel aan. Dat wordt een eigenaardig schema.

In het volgende voorbeeld iets nuttigers:  
perl manipulate gen:vasneg:7:7:X extratafel:2:o1-14

Wat gebeurt hier? Er komen 2 extra tafels, maar die o1-14 is nieuw. Dit is een voorbeeld van een lijst, in dit geval de oneven nummers van 1 tot en met 14. In plaats van o1-14 had ik ook kunnen schrijven 1,3,5,7,9,11,13 wat hetzelfde is. Deze lijst geeft aan welke paren in aanmerking komen om van tafel te worden gestuurd. Het blijkt dat dit precies hetzelfde schema oplevert als  
perl manipulate gen:vasneg:9:7:X  
omdat dit (toevallig) hetzelfde mechanisme gebruikt dat in die Vasneg’s wordt gebruikt voor uitbreiding.

Een derde argument kan wat karakters bevatten die nog wat aansturen. Als het cijfer 1 in dat argument voorkomt beginnen die twee extra paren tegen elkaar.

De extratafel optie kan flink wat tijd kosten bij ingewikkelde schema's, en lukt ook niet altijd. De implementatie is echter zo dat als het faalt het ook echt niet op deze manier kan. Alle mogelijkheden worden geprobeerd, en de beste wordt genomen. Als het nodig mocht zijn om te lenen van tafels waaraan al geleend is wordt dat gemeld. Het kan dan blijkbaar niet zonder.

# Optie draai

Met de draai optie kun je bij tafels NZ en OW verwisselen. De optie heeft drie vormen, de eerste is gewoon  
draai:  
en doet niets. Klinkt niet echt nuttig, maar de reden is als volgt: bij sommige schema's(Vasnegs, Expanded Mitchells, Endless Howells en sommige GEN files) zijn al standaard draaiingen ingebouwd. Deze draaiingen worden uitgevoerd als er maar één invoer schema is, en als de *draai* en *balans* opties niet worden gebruikt. Het idee is dat slechts in triviale gevallen het programma zelf mag draaien, in alle andere gevallen moet hij het aan de bediener overlaten. Dus door de optie *draai* zonder argumenten te gebruiken zet je de standaard draaiingen uit. Vergelijk bijvoorbeeld de volgende twee commando’s:  
perl manipulate gen:vasn:5:5:1 Draai in ronde 2  
perl manipulate gen:vasn:5:5:1 draai: Geen draai

De volgende versie is met één argument:  
draai:2,9  
en draait alle tafels in de rondes 2 en 9. Die 2,9 is weer een lijst, dus iets als 1,6-11 mag ook(draait in totaal 7 rondes).  
perl manipulate gen:vasn:5:5:1 draai:2  
geeft weer een standaard Vasneg, met ronde 2 gedraaid. De draai optie zet de standaard draaiing uit, maar doet hem daarna zelf weer.

De derde versie is met twee argumenten:  
draai:12,13:27-30,32 draai:10,11:31  
Deze twee draaiingen komen uit MAFA bij één of andere Reduced Interwoven Howell. Je kunt dus meerdere draaiingen opgeven, en die worden in volgorde uitgevoerd. Hier staat dat in de rondes 12 en 13 alle tafels moeten worden gedraaid waar paar 27,28,29,30 en 32 zitten, en dat in rondes 10 en 11 de tafel moet worden gedraaid waar paar 31 zit.

# Optie balans

Om te beginnen: *manipulate* weet niets af van de balans van een schema, en kan die zelf dus ook niet verbeteren. Hiervoor wordt gebruik gemaakt van het programma *balans* van Peter Smulders. Het enige wat de optie  
balans:  
doet is dat programma aanroepen op een redelijk standaard manier. Dit kan alleen voor paren schema's momenteel. Het schema zoals door alle andere opties en invoer schema’s is opgebouwd wordt aangeboden aan het *balans* programma van Peter, en de suggesties worden geimplementeerd, dus er worden een aantal tafels gedraaid.

Dit is niet alles wat het programma *balans* kan, en power-users wordt sterk aangeraden om de documentatie van Peter Smulders goed te lezen. Je kunt uiteraard altijd met *manipulate* een schema opbouwen en uitschrijven, en daarna dat resultaat door *balans* halen met de opties die jij kiest, en het resultaat daarvan zo gewenst weer verder met *manipulate* behandelen.

Soms kun je de opties voor *balans* vanuit *manipulate* aangeven:  
balans:-r1,-s10000  
houdt de eerste ronde vast en doet meer samples.

Om deze optie te laten werken moet het programma *balans* uiteraard zijn geinstalleerd en te vinden in of vanuit de huidige folder/directory.Je kunt met behulp van de PATH variabele dit ook voor elkaar krijgen. Simpel gezegd werkt het als je *balans* vanaf de command prompt kan aanroepen.

# Opties hernummeren

Er zijn drie opties die allerlei hernummeringen doen: *spelvast*, *spelervast*, en *renumber*. Het blijkt zeer ingewikkeld om dat allemaal te combineren, dus momenteel kun je maar één van de drie in één aanroep van *manipulate* gebruiken. De *renumber* optie heeft meerdere varianten en die kun je meer dan één keer gebruiken.

## Spelvast

De optie *spelvast* is bedoeld om een schema om te werken naar Nederlandse standaarden, dwz dat de spellen zoveel mogelijk op tafel blijven liggen. Simpel voorbeeld:  
perl manipulate gen:mit:7 spelvast:  
bouwt een 7 tafel Mitchell om naar wat wij een Scheveninger noemen.

Niet in alle gevallen kan het zo simpel zijn. Sommige schema's kunnen slechts spelvast worden op een bepaalde set tafels in een bepaalde set ronden. In het algemeen zijn dat de laagst genummerde tafels en ronden. Dus een uitgebreider voorbeeld:  
spelvast:7:6  
maakt het schema spelvast op tafels 1-7 in rondes 1-6. Vermoedelijk heeft dit schema op tafels 8 en hoger leentafels, en gebeurt er in ronde 7 iets woests met de spellen.

Een schema met meer spelsets dan tafels zal lege tafels gaan bevatten, zoals bijvoorbeeld in het bekende multiplex schema van 6 rondjes en 10 paren.

Deze optie controleert of op al die tafels wel verschillende spellen liggen zodat het kan en krimpt eventueel dat aantal tafels in. Probeer eens dit:  
perl manipulate gen:mitch:6 spelvast:

Het blijkt dat je voor dit schema eigenlijk  
spelvast:5  
had moeten gebruiken, maar dit wordt door *manipulate* gezien en opgevangen.

Je kunt ook een schema spelvast maken met meer tafels dan spelsets, zodat je een aantal groepjes tafels krijgt met vaste spellen. Maar dat gebeurt alleen maar met een expliciet argument van aantal tafels, niet zonder dat. Voorbeeld:  
perl manipulate gen:mit:6 ren:boar:1-6:4-6,1-3 out:mit6rot  
perl manipulate gen:mit:6 mit6rot spelvast:12  
maakt een spelvast schema van 12 tafels waarin 2x2 groepjes van 6 tegen elkaar spelen.

## Spelervast

Deze optie is bruikbaar voor minstens twee dingen: zorgen dat een bepaalde deelnemer vast zit op een bepaalde tafel, bijvoorbeeld in verband met een handicap, of zorgen dat een “Nederlands” schema wordt omgebouwd naar internationaal. Eerste simpele voorbeeld:  
spelervast:3:1  
zorgt ervoor dat deelnemer nummer 3 vast blijft op tafel 1. De rest van het schema wordt hernummerd om plaats te maken.

Het is ook mogelijk om het schema gelijktijdig uit te breiden met een extra tafel speciaal voor die deelnemer. Dan blijven de anderen op hun tafels en is er elke ronde een tafel leeg. Voorbeeld:  
perl manipulate gen:end:6 spelervast:1:7  
breidt een 6-tafel Howell uit naar 7 tafels voor paar 1. Uiteraard zou het in dit geval een stuk makkelijker zijn om paar 1 nummertje 12 te geven, die blijft al zitten...

Meer in het algemeen kunnen de argumenten lijsten getallen zijn, dus bijvoorbeeld:  
spelervast:1,6:7-8  
plaatst deelnemer 1 op tafel 7 en deelnemer 6 op tafel 8. De lijsten moeten dus even lang zijn, allebei 2 in dit voorbeeld. Wat gebeurt er nu als 1 tegen of met 6 speelt? Dan mag 1 blijven zitten, want die staat vooraan in de lijst.

Dat brengt me tot het laatste voorbeeld:  
perl manipulate gen:vasn:7:7:X spelervast:o1-14:1-7  
heeft twee lijsten van allebei 7 lang. Dit zorgt ervoor dat dit Vasneg schema wordt omgebouwd tot een soort draaiende Mitchell, waarbij de NZ paren blijven zitten, en OW en de spellen bewegen.

## Usp

De optie *usp* is equivalent aan  
renumber:contestant:auto:auto renumber:board:auto:auto  
en wat *dat* betekent leest u in de volgende sectie. Usp staat natuurlijk voor Universele Start Positie, weer zo’n mooie Nederlandse gewoonte.

## Renumber

Het programma kan alles naar alles hernummeren. *Renumber* krijgt 3 argumenten:

1. Type hernummering: *Contestant, Board*, *Round, Table*
2. De lijst met nummers die moeten veranderen
3. De lijst met nummers waarnaar ze moeten veranderen

De twee lijsten moeten weer even lang zijn. Simpel voorbeeld:  
renumber:round:1,2:2,1  
verwisselt rondes 1 en 2 in het schema. En voor een schema van 11 rondes:  
renumber:round:1-11:11-1  
speelt het hele schema achterstevoren. Als je bijvoorbeeld ronde 4 wilt weghalen hernummer je zo dat ronde 4 de laatste ronde wordt, en gebruikt de *curtail* optie uit de volgende sectie om hem helemaal weg te halen.

Stel dat je voor de tweede zitting van een toernooi twee Vasneg’s speelt van 7 tafels, maar je wilt de oneven paren verwisselen:  
perl manipulate gen:v:7:7:x gen:v:7:7:x ren:co:o15-28,o1-14:o1-28  
Doet het in 1 aanroep. Genereert twee vasnegs, combineert ze naast elkaar(default) en hernummert alle oneven paren.

Bij de *contestant* en *board* optie kunnen allebei de lijsten het woord *auto* zijn. De eerste lijst wordt dan de lijst van deelnemers of spelnummers in tafelvolgorde van de eerste ronde, en de tweede lijst alle nummers in volgorde. Dit maakt dan automatisch de Universele Start Positie. Verder is *Pair* een synoniem voor *Contestant*.

# Kleiner maken

Een paar mogelijkheden om het schema weer in te krimpen.

## Curtail

Het is mogelijk een schema in te korten door de laatste ronde(n) te schrappen. In internationaal verband vaak toegepast bij Mitchells, maar in Nederland zeer ongebruikelijk omdat dan vaak niet alle spelers dezelfde spellen hebben gespeeld. In ieder geval kun je het doen door middel van  
curtail:8  
die schrapt alle ronden na ronde 8.

## Krimp en paarweg

Je kunt de twee hoogste paren, en dus vermoedelijk een tafel, verwijderen met de krimp optie. Simpel voorbeeld:  
krimp:  
haalt 2 paren, en wellicht 1 tafel weg. Meer tegelijk kan ook:  
krimp:4  
verwijdert de 8 hoogste paren, en wellicht 4 tafels. Vanwaar dat wellicht? Het programma laat de hoogste tafels vervallen als daar geen enkele ronde iemand zit. Dus als het hoogste paarnummer aan de hoogste tafel vast zit, zoals bij Vasneg’s, verdwijnt de tafel. Maar dus niet altijd.

Als de twee paren die eenzaam aan tafel overblijven toevallig dezelfde spellen moeten spelen worden ze tegen elkaar ingedeeld, en als ze NZ en OW zaten blijft dat gehandhaafd. Als ze in dezelfde richting zaten worden ze at random tegen elkaar ingedeeld. Ze komen altijd op de laagste van de twee tafels waaraan ze zaten.

Als het goed is dan maakt krimp precies ongedaan wat extratafel heeft gemaakt.

Het is mogelijk twee paren weg te halen die niet de hoogste zijn:  
paarweg:16:32  
verwijdert de paren 16 en 32. Zoals boven worden tafels gecombineerd. Bij wegschrijven van het schema worden paren hernummerd om de gaten weg te halen.

## Leegweg

Soms hebben schema's lege tafels, en soms maakt *manipulate* lege tafels, bijvoorbeeld bij spelvast of krimp. De optie leegweg: haalt lege tafels weg. Alle ontmoetingen worden naar lagere tafels geschoven. Aan het eind van het programma worden hoog genummerde tafels die in alle rondes leeg zijn automatisch verwijderd. In combinatie met spelvast kan de volgende truuk:

Stel dat het schema niet alle spellen even vaak speelt per ronde, maar je wilt het begin in ieder geval spelvast. Doe dan het volgende:  
perl manipulate <......> spelvast:<aantal tafels> leegweg:  
dat maakt eerst een tafel aan per spelset per keer dat die het meest gespeeld word. Aan het eind zitten daar veel lege tafels tussen. De leegweg optie haalt die weer weg. Het effect is een spelvast schema op de lagere tafels, en wat extra tafels daarna.

# Opties uitvoer

Tijdens de uitvoer fase kan het schema ook worden gesplitst in deelschema’s. Er wordt dan een nieuwe filenaam gemaakt voor elk deelschema. Dit is echter niet mogelijk voor uitvoer in Pairs 4.1 binair formaat, want dan kun je niks met de filenaam. De mogelijkheid tot splitsen kan dus niet samen met dit soort uitvoer.

## Split

Net zoals schema’s horizontaal en verticaal kunnen worden gecombineerd kunnen ze zo ook worden gesplitst. Voorbeeld:  
split:vertical:7,6  
neemt een schema van 13 ronden, en splitst het in tweeën. Eerste zitting 7 ronden, tweede 6 ronden. Er wordt uiteraard gecontroleerd of dat zinnig is, in dit geval dus of er geen spellen zijn die zowel in de eerste als de tweede zitting voorkomen. De filenamen bevatten in dit geval de string r1-7 en r8-13.

De andere kant uit splits je op tafels, dus  
split:horizontal:11,11,6  
splitst een schema van 28 tafels in 2x11+6. Hier wordt gecontroleerd of er geen deelnemers over de grens van hun sectie hoppen. De filenamen bevatten in dit geval de string t1-11, t12-22 en t23-28.

Splitsen kan ook beide richtingen uit tegelijk. In ieder subschema worden paren en spellen hernummerd naar beneden toe. Dus als bij een subschema van 4 tafels na splitsing de paren 1,12,18,19,23,34,47 en 50 op die tafels spelen worden ze in die volgorde hernummerd naar 1 t/m 8.

Je kunt in plaats van een list ook het woord auto geven, of helemaal geen argument en dan zoekt *manipulate* zelf de (kleinste) grenzen uit. Let op, dit werkt alleen als je maar één kant uit splitst, en als je een barometer schema automatisch verticaal splitst krijg je wat je verdient .

## Out

Hiermee specificeer je de naam en eventueel het type van het uitvoerbestand. Er zijn drie types:

1. ASC files met extensie anders dan .csv voldoen aan de ASC file specificatie en gebruiken spaties als scheiders en meestal hoofdletters voor spelsets.
2. ASC files met extensie .csv zijn bedoeld om in een spreadsheet te bewerken. De velden worden gescheiden met een puntkomma, en de spelsets worden als getal weergegeven. Ook dit voldoet aan de ASC specificatie.
3. Pairs 4.1 binaire bestanden zijn wat ze zijn. Niet te lang meer gebruiken.

Simpel voorbeeld:  
out:toernooi.asc  
gebruikt de naam toernooi.asc en gebruikt formaat #1. De filenaam kan als *split* actief is langer worden. Als je verwacht wat handmatige correcties aan het schema te gaan doen gebruik bijvoorbeeld:  
out:tussenversie.csv  
dan krijg je formaat #2.

Als je toch nog Pairs 4.1 binair wilt gebruiken moet je om te beginnen een correcte filenaam bedenken. De eerste twee posities zijn het aantal deelnemers, positie 7 en 8 het aantal ronden, de filenaam precies 8+3 karakters lang. Dan doe je  
out:24INDI07.XYZ:pairs  
en dat doet de truuk. *Split* kan niet met deze uitvoer.

Als je geen *out* optie gebruikt wordt het resulterende schema of schema’s op standaard uitvoer (is meestal het scherm) geschreven. Deze standaard uitvoer kan ook worden doorgesluisd naar een bestand d.m.v. het teken '>'. Dus dit kan ook:  
perl manipulate gen:end:10 >barom10.asc

# Lijsten

Een lijst is een aantal elementen gescheiden door kommaas. Minimum aantal elementen is 1, dus de kleinst mogelijke lijst is bijvoorbeeld:  
1  
en slechts iets groter:  
1,2

Een element kan dus een getal zijn, maar ook een reeks:  
1-20

En zo’n reeks kan gekwalificeerd worden, niet alles in die reeks maar bijzondere getallen uit die reeks:  
e20-1  
dit zijn de even getallen van 20 tot 1, dus in aflopende volgorde.

In het algemeen ziet zo’n kwalificatie er als volgt uit: <deler>D<rest>R<reeks>, voorbeeld:  
7D3R1-50  
Dit zijn de getallen  
3,10,17,24,31,38,45  
dus de getallen van 1 t/m 50 die rest 3 hebben bij deling door 7.

Het moge duidelijk zijn dat *e* equivalent is aan 2D0R, en *o* aan 2D1R.

# Recepten

Niet alle schema's hoeven als invoerfile beschikbaar te zijn. Een schema kan ook gemaakt worden als er een recept voor is. Als aanwezig leest *manipulate* de file *recepten* om te weten wat er te maken is en hoe. Files waarvoor een recept aanwezig is en die er nog niet zijn worden gemaakt. Als de subfolder *recept* bestaat komen ze daar te staan*.* Ze blijven daarna staan, en worden een volgende keer als file gebruikt en niet opnieuw gemaakt. Een kort voorbeeld van twee recepten:  
xxx  
 manipulate gen:vasn:9:7:1 out:TEMP1  
 balans -r 1 TEMP1  
 manipulate renumber:ro:1-7:7-1 \_balans.txt  
yyy  
 manipulate xxx extra:1

De file xxx wordt gamaakt door twee aanroepen van *manipulate* en een van *balans*. De file yyy wordt gemaakt door daaraan dan nog een tafel toe te voegen. Tussenfiles kunnen worden gebruikt, en de namen TEMP0 tot en met TEMP9 worden vervangen door unieke namen, en na afloop worden die tussenfiles weer weggegooid.

Recepten kunnen ook geparametriseerd zijn op aantal tafels of deelnemers. Hierna een voorbeeld. De woorden T, C en P staan voor tafels, contestants en paren, de laatste twee zijn synoniemen. C en P zijn twee keer het aantal tafels. Daarmee kan bijvoorbeeld dit:  
mit2x[Te2-20].asc  
 manipulate gen:mi:[T] ren:bo:1-[T]:[T/2+1]-[T],1-[T/2] out:TEMP1  
 manipulate gen:mi:[T] TEMP1 spelvast:[2\*T]  
waarbij in de definitie staat dat dit een dubbel Mitchell schema is voor 2x 2-20 tafels(even). Dat wordt in twee stapjes gemaakt, eerst een normale relay Mitchell waarvan de spellen worden hernummerd. Tussen de [] staan expressies die worden uitgerekend en ingevuld. Daarna wordt die hernummerde Mitchell samen met een niet hernummerd broertje gecombineerd en spelvast gemaakt op alle tafels.

# Formaat GEN files

De GEN files zijn bedoeld om schema’s uit MAFA makkelijk te kunnen invoeren. In het algemeen bevatten die schema’s de eerste ronde, en bevatten een impliciet recept voor de volgende rondes. Dit kan door *manipulate* worden gelezen en de bijbehorende schema’s gegenereerd.

Hoe ziet nu zo’n GEN file eruit? Laten we eens beginnen met een simpel voorbeeld:  
#Naam: barom20  
#  
20\_ 1  
19 2  
3 18  
4 17  
16 5  
15 6  
14 7  
13 8  
9 12  
11 10

Dit is de 10 tafel Endless Howell, zoals ook gegenereerd door gen:en:10. Afgezien van wat commentaar bovenin geeft het een regel per tafel voor de eerste ronde. Dus tafel 1 is paar 20 tegen paar 1, tafel 10 is 11 tegen 10. Dat het een paren schema is is duidelijk omdat er maar twee deelnemers staan, dat het barometer is volgt uit het feit dat er geen spelset bijstaat. Paar 20 heeft een \_ (underscore) achter het getal. Dat is het equivalent van het onderstrepen in MAFA, dus paar 20 is stationair. Alles bij elkaar is dit voldoende informatie om het hele schema te maken. Er zijn 19 bewegende paren, dus er komen 19 rondes. Dit alles gaat automatisch.

Een wat ingewikkelder voorbeeld:  
#Naam: i\_stz\_hwll\_3x7  
#  
7; 14; 8 10 1  
4 15 12 13 2  
11 3 6 18 3  
21; 5 9 20 4  
16 19 17 2 5  
1

Dit is een individueel schema, er zijn namelijk 4 deelnemers per tafel. Het is ook geen barometer want er staan spellen op, als getallen. Dit schema staat in MAFA, pag 365. Het is vrijwel hetzelfde als wat daar staat, met de uitzondering van die puntkommaatjes achter 7, 14 en 21. Deze geven aan dat dit een schema is in 3 stanzaas. Er zijn drie groepjes van 7 spelers, die onderling elkaar volgen, en als ze dan 7 rondes hebben gespeeld roteren de groepjes. Dat betekent die puntkomma: voorlopig einde groepje. Het is voorlopig omdat het schema niet stopt maar drie keer wordt gespeeld, totaal 21 rondes, 5 tafels. Dit kun je natuurlijk met de split:vertical:7,7,7 of gewoon split:v optie weer makkelijk in drie zittingen splitsen.

Nog een voorbeeld met een ander symbool: MAFA pag 170  
#Naam: rd\_ihwll\_2x13+4

#Draai:12,13:27-30

27\_ 1 A

14 26. A

25 23 B

13. 22 C

12 28\_ D

21 11 E

24 20 F

8 10 G

19 7 H

18 29\_ I

6 17 J

9 5 K

4 3 L

30\_ 16 L

15 2 M

Hier zien we drie nieuwe dingen en een oude. Het oude is de \_ achter 27 t/m 30, dat zijn stationaire paren. We zien achter paar 13 en 26 een punt. Een punt is een definitief einde, dus dit is geen stanza schema. Dit schema wordt dus 13 rondes en 15 tafels. Zo te zien, spellen nu als letters, spelen tafel 1 en 2 samen, en tafel 13 en 14. Ook zien we in het commentaar een *draai* optie. Bij het inlezen van dit schema wordt, onder de reeds bovengenoemde voorwaarden, een automatische draai gedaan, dus in ronde 12 en 13 switchen alle stationaire paren van windrichting.

Nog een voorbeeld:

#Naam: rd\_stz\_hwll\_2x12-s1

12\_ 1 A

16 19 B

6 11 C

22 20 D

14 15 E

17 21 F

5 9 G

2 3 H

10 8 I

18 23 J

4 7 K

13 24\_ A

Hier zie je een schema met twee stationaire paren, 12 en 24. Die zorgen er impliciet voor dat de groepjes van 1-11 en 13-23 termineren, het is dus alsof er een punt staat achter 11 en 23. Die punt kun je er bij zetten, maar het hoeft niet. Dit schema is dus 11 rondjes, 12 tafels.

# Extra programmaas

## Maak-toernooi

Momenteel is er één extra programma: *maak-toernooi*

Op dezelfde manier aan te roepen, dus  
perl maak-toernooi

Dit programma gebruikt 4 of meer argumenten

1. Naam toernooi
2. Aantal paren
3. Aantal zittingen
4. Aantal rondes/zitting
5. (Optioneel) spelvast
6. (Optioneel) usp
7. (Optioneel) vastbegin:filenaam

Voorbeeld:  
perl maak-toernooi gbc 48 2 11

Maakt schema's voor een toernooi van 2 zittingen van 11 rondes voor 48 paren. Compleet toernooi schema wordt gemaakt, het wordt door *balans* gehaald, en daarna gesplitst per zitting.

*Maak-toernooi* maakt gebruik van een lijst schema's die staat in de file *toernooi-schema's.txt* waarin voor elk schema staat welke groepen erin zitten, en hoeveel tafels en hoeveel rondes het zijn. In principe kun je die file aanpassen met je favoriete basis schema's. Maar speel er eerst even mee voor je het speelgoed kapotmaakt 

schema's van een oneven(liefst priem) aantal rondes werken meestal beter. Er zijn grenzen aan hoeveel zittingen je kunt spelen, daar wordt op getest.

Als de toernooinaam gbc is eindigen de schema's in files met de naam gbc-r<ronde>-<ronde>.asc

Met de spelvast optie wordt het resulterende schema hernummerd om de spellen te laten liggen.

Met de usp optie worden de paren hernummerd naar universele startpositie

De vastbegin optie is bedoeld voor het volgende scenario: je wilt een serie schema's maken voor een toernooi waar je nog niet precies weet hoeveel paren er komen. Maar het zou wel fijn zijn als je in de eerste ronde geen nieuwe loopbriefjes aan de hele zaal hoeft uit te delen. Deze optie stelt je in staat in de eerste zitting alle groepen behalve de laatste vast te zetten, zodat in de eerste zitting de schema's daar identiek zijn. Dit maakt de balans van die schema's (iets) slechter! Voorbeeld:  
perl maak-toernooi tn60 60 2 7 vastbegin:vb  
perl maak-toernooi tn62 62 2 7 vastbegin:vb

Het argument van vastbegin is een filenaam. Voor de eerste oproep van de serie moet die file er *niet* zijn. De eerste aanroep maakt de file, de rest gebruikt hem.

## Loopbriefje

Inderdaad.