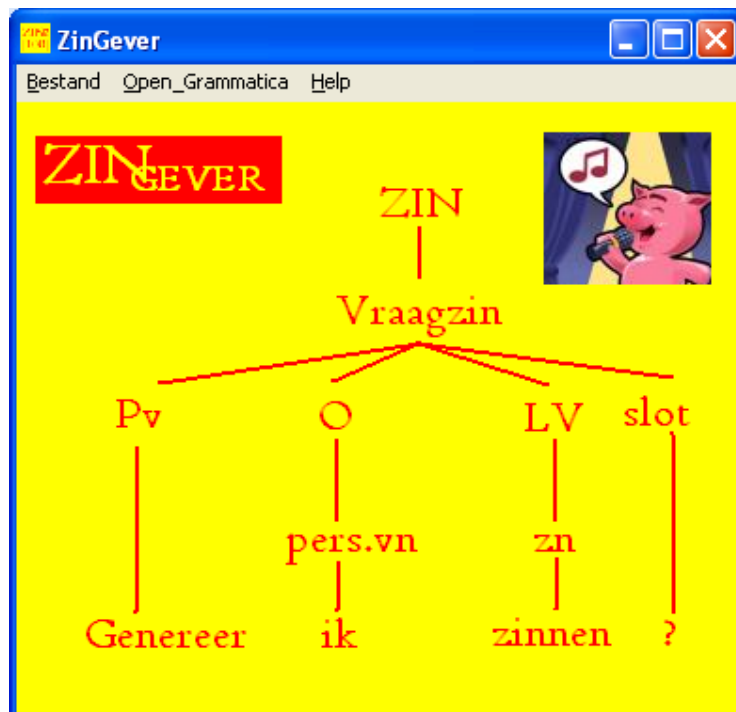


ZinGever



Oktober 2006
A.M. Stoop

0. Inleiding

Voor je ligt een handleiding bij een computerprogramma dat je zal helpen om inzicht te krijgen in structuren die verborgen zitten in “zinnen”. Deze “zinnen” zijn de uitvoer van het programma; de invoer wordt gevormd door een reeks van grammaticaregels die gebaseerd zijn op de BNF-notatie. Als je zelf de uitvoer correct acht, zullen de regels misschien wel goed zijn. Dat wil echter nog niet zeggen dat ze perfect zijn of wetenschappelijk iets aantonen.

Het computerprogramma bevindt zich nog in een ontwikkelingsstadium en zal op een aantal punten nog aanzienlijk verbeterd moeten worden. De basisfunctie werkt echter uitstekend: op basis van een probabilistische, contextvrije grammatica kan het reeksen van tekens produceren. Deze reeksen van tekens kunnen *woorden*, maar ook *zinnen* zijn. Om die reden is het programma **ZinGever** genoemd.

De tweede reden waarom het deze naam draagt, is dat het werken met grammatica's je inzicht geeft in de structuren van gegevens.

In dit boekje vind je eerst een klein beetje theorie over het bovengenoemde type grammatica. Daarna wordt het leuk: je zult merken dat je met een dergelijke grammatica leuke en leerzame dingen kunt doen. Leerzaam, natuurlijk, omdat we op de eerste plaats wél in een schoolomgeving zitten, maar ook omdat je snel meer inzicht zult krijgen in hoe reeksen van tekens in elkaar kunnen zitten. Probeer maar eens een reeks optelsommetjes voor je kleine neefje uit groep 4 te maken of keer-sommen (vermenigvuldigingen) voor je nichtje uit groep 6. Op een hoger niveau kun je denken aan vierkantsvergelijkingen die je vrij recent nog gehad hebt. Probeer ook eens om de kalenderdata in regels te vangen, zonder dat er dagen als 30 februari uitrollen of maak een grammatica voor een digitale klok. Vang de gehele getallen van 0 tot en met 100 eens in een reeks grammaticale regels, maar schrijf ze dan wel met letters. Als klap op de vuurpijl kun je proberen om een deelverzameling van het Nederlands of een andere moderne taal in grammaticaregels vast te leggen.

En, o ja, leuk. Leuk? Ja, dat ook, maar hoe? Je zult zien dat het maken van grammaticaregels een makkie is, maar dat de eindresultaten niet altijd even logisch zijn. Soms zijn ze zelfs lachwekkend, vooral als je met talige structuren bezig bent. Houd je van puzzelen in je eigen puzzels? Hoe komt het dat je “*achtenachtig*” en “*drietien*” krijgt als je “*achtentachtig*” en “*dertien*” als resultaat verwacht? Wat doe je verkeerd als **ZinGever** “*het vlieg en de paard*” creëert, als je toch echt “*de vlieg en het paard*” bedoelde te maken? Helemaal hilarisch wordt het als je een complimenten- of scheldwoordgrammatica maakt. Maar daar komen we later aan toe.

1. Theorie: Wat is een *probabilistische, contextvrije grammatica*?

Wat is een *probabilistische, contextvrije grammatica*? Laten we de term van achter naar voren ontleden:

- **Grammatica:** laat je niet afschrikken door het voorblad. Ja, deels heeft het programma te maken met grammatica's zoals je die kent van de natuurlijke talen, maar aan de andere kant is een grammatica niets anders dan een reeks van regels waarin de structuur van zinnen of woorden van een natuurlijke, een kunstmatige of een computertaal wordt vastgelegd. In dit geval worden de regels vastgelegd in gestructureerde (= formele) regels. Je zou het zelfs een beetje wiskundig kunnen noemen.

Het doel van het opstellen van deze regels is om grammaticale van ongrammaticale zinnen te onderscheiden. Heb je dat in regels vastgelegd, dan zou een computer het werk voor je kunnen doen.

Verder probeert men in de regels de structuur van zinnen in kaart te brengen. Onderzoek naar de structuur van taal kan bijvoorbeeld een antwoord bieden op de vraag of er zoiets als een *universele grammatica* bestaat, hoe taal door kinderen geleerd wordt, en wat de belangrijkste verschillen tussen talen zijn. Om op deze zeer abstracte vragen concrete antwoorden te krijgen is een abstract en wiskundig model van taal noodzakelijk.

Toonaangevend op dit gebied is het werk van Noam Chomsky, die in de jaren vijftig en zestig van de vorige eeuw een taalmodel voorstelde waarin de grammaticale zinnen van een taal worden geproduceerd (gegenereerd) op basis van een aantal abstracte regels.

De eenvoudigste van deze regels is de zogenaamde *herschrijfregel* die zegt dat een categorie C mag worden herschreven als een rijtje categorieën $C_1, \dots, C_n$. Voorbeelden van herschrijfregels voor een klein stukje van het Nederlands vind je in onderstaande figuur.

ZIN → **Vraagzin**
Vraagzin → **Pv O LV slot**
O → **pers.vn**
LV → **zn**

Pv → *Genereer*
pers.vn → *ik*
zn → *zinnen*
slot → ?

De BNF-notatie van bovenstaande grammatica is:

ZIN :: Vraagzin
Vraagzin :: Pv O LV slot
O :: pers.vn
LV :: zn

Pv :: Genereer
pers.vn :: ik
zn :: zinnen
slot :: ?

Als je de regels vergelijkt met het plaatje op de kft zie je wat het resultaat is van de regels van de grammatica. Je ziet dat het begrip *grammatica* al heel anders uitgelegd wordt dan je gewend was.

ZIN wordt *beginsymbool* genoemd: het is de wortel van de boomstructuur. Alle andere symbolen die links van een $\rightarrow$ (= het herschrijfsymbool) kunnen staan heten: *hulpsymbolen*. De schuingedrukte symbolen heten: *eindsymbolen*. Zij zijn altijd de blaadjes aan de boom en kunnen dus nooit links van een herschrijfsymbool (= $\rightarrow$) staan.

In schema:

<i>Taalkunde-teken</i>	<i>BNF-teken</i>	<i>Betekenis</i>
$\rightarrow$	::=	is gedefinieerd als
Teken of reeks tekens	< ... >	Hulpsymbool: naam staat op puntjes tussen de punthaken
<i>Teken of reeks tekens</i>	Teken of reeks tekens	Eindsymbool

Een zin is natuurlijk niet altijd een vraagzin. De grammatica kan uitgebreid worden door achter het hulpsymbool **Vraagzin** een of-teken te plaatsen met daarachter een alternatief **Zin**. Daarna moet je voor **Zin** een nieuwe herschrijfgregel maken, waarbij je gebruik kunt maken van al bestaande symbolen. Resultaat:

ZIN :: Vraagzin | Zin
Zin :: O Pv LV slot

Het schema kan uitgebreid worden met:

<i>Taalkunde-teken</i>	<i>BNF-teken</i>	<i>Betekenis</i>
		of

- **Contextvrije grammatica:** een grammatica die syntactische structuren beschrijft door middel van een stelsel abstracte herschrijfgregels. Wanneer een grammatica slechts gebruik maakt van herschrijfgregels van de vorm $C \rightarrow C_1 \dots C_n$ ($n \leq 0$), waarbij precies één categorie C wordt herschreven tot 0 of meer dochters, noemen we dit een *contextvrije grammatica*. Chomsky veronderstelt dat zulke grammatica's te eenvoudig zijn om de grammatica's van natuurlijke taal goed te beschrijven. Toch zijn zulke grammatica's binnen de computerwereld nog populair, omdat er een groot aantal

technieken voor het automatisch ontleden van contextvrije grammatica's bestaat. Bovendien is *contextvrije grammatica* vaak voldoende om een behoorlijk groot deel van een taal te kunnen beschrijven. Voor die fenomenen die niet eenvoudig of zelfs helemaal niet met contextvrije regels te beschrijven zijn, stelt Chomsky het gebruik van transformaties voor. Maar over transformaties gaan we het hier niet hebben.

- **Probabilistische**, contextvrije grammatica: je kunt je voorstellen dat de computer niet meer weet wat hij moet kiezen als je achter: **zn** → ***zinnen*** andere zelfstandig naamwoorden plaatst, bijvoorbeeld:

zn → ***zinnen*** | ***onzin*** | ***grappen*** (je begreep al dat | “of” betekent)

Door bij ieder alternatief in de of-regel een kans van verschijning (hoe vaak wil ik dat een onderdeelje voorkomt in een structuur) te zetten, kan de computer wel kiezen.

zn → **60 *zinnen*** | **20 *onzin*** | **20 *grappen***

De computer gooit met een denkbeeldige dobbelsteen (met honderd vlakken) en kiest op grond van de uitslag de juiste herschrijving. Is die bijvoorbeeld **58**, dan wordt het *zinnen*; is die **68**, dan wordt het *onzin*. Komt de dobbelsteen boven de 80 dan wordt het *grappen*. Deze dobbelsteen zorgt voor de naamgeving *probabilistisch*.

Opdracht 1: vergelijk regels met boomstructuur

Ga voor jezelf na of je de relatie begrijpt tussen de grammatica van de vorige bladzijde en de tekening op de voorkant.

2. Herschrijfregels in ZinGever

Omdat een computer alleen met heel exact geschreven regels kan werken, moeten de regels uit de herschrijfgrammatica hierboven aan nog meer wetten voldoen. Vergelijk de oude regels hierboven met de nieuwe die ernaast staan:

Herschrijfregels in oude vorm

Herschrijfregels in nieuwe vorm

1	ZIN → Vraagzin	ZIN : [100] Vraagzin .
2	Vraagzin → Pv O LV slot	Vraagzin : [100] Pv O LV slot .
3	O → pers.vn	O : [100] pers.vn .
4	LV → zn	LV : [100] zn .
5	Pv → <i>genereer</i>	Pv : [100] <i>*genereer</i> .
6	pers.vn → <i>ik</i>	pers.vn : [100] <i>*ik</i> .
7	zn → <i>zinnen</i> <i>onzin</i> <i>grappen</i>	zn : [60] <i>*zinnen</i> ; [20] <i>*onzin</i> ; [20] <i>*grappen</i> .
8	slot → ?	slot : [100]*?.

1. Achter iedere herschrijfregel **moet** een punt staan.
2. Iedere kans **moet** tussen rechte haken '['] staan.
3. De som van de kansen op een regel **moet** precies 100 (honderd) zijn.
4. Het herschrijftteken '→' is lastig intypen. Dat is voortaan een dubbele punt ':'. De blaadjes onderaan de takken (zie regel 5 tot en met 8) **moeten** met een sterretje '*' beginnen. Anders gaat de computer tevergeefs op zoek naar het woord als hulpsymbool, dus: als iets dat voor een dubbele punt staat.
6. Let erop dat je nooit hulp- en eindsymbolen door elkaar rechts van het herschrijfsymbool mag plaatsen. De volgende regel is dus **verboden**:

O : [50] **pers.vn**; [50] **ik*.

We noemen hem *ongebalanceerd* (= onevenwichtig).

7. Het of-teken 'I' (zie regel 7) moet je vervangen door een puntkomma ';'. Ook die is wat gemakkelijker op je toetsenbord te vinden. Het of-teken mag je trouwens ook plaatsen tussen hulpsymbolen of reeksen van hulpsymbolen. Regel 2 mag je als volgt uitbreiden:

2a Vraagzin : [50] **Pv O slot**; [50] **Pv O LV slot**.

Nu worden er twee soorten vraagzinnen gegenereerd: één soort zonder een lijdend voorwerp (*genereer ik?*) en één soort met lijdend voorwerp, beide met een kans van 50%.

Met deze nieuwe vorm kan de pc uit de voeten. Mocht je een foutje maken, dan zal het computerprogramma **ZinGever** dat aangeven, zodat je het kunt verbeteren.

In schema:

<i>Taalkunde- teken</i>	<i>BNF- teken</i>	<i>ZinGever</i>	<i>Betekenis</i>
→	::=	:	is gedefinieerd als
Teken of reeks tekens	< ... >	Teken of reeks tekens	Hulpsymbool: naam staat op puntjes tussen de punthaken
<i>Teken of reeks tekens</i>	Teken of reeks tekens	Teken of reeks tekens voorafgegaan door *	Eindsymbool
		;	of
		[]	Geeft kans van verschijning aan
		.	Verplichte regelafsluiter

Opdracht 2: tel de resultaten

1. Teken de boomstructuur van de voorkant na, maar vervang de herschrijving van **zn** door die van regel 7 hierboven.
2. Zet bij alle vertakkingen de kansen.
3. Hoeveel verschillende zinnen kunnen er met deze mini-grammatica gemaakt worden?
4. Schrijf die zinnen op.
5. Welke zin(nen) komen erbij als je regel **2** vervangt door regel **2a**?

Het kan geen kwaad om af en toe wat commentaar in je grammatica te zetten.

Dat doe je door eerst twee schuine strepen vooraan op een regel te plaatsen. Op deze wijze kun je wat orde in de chaos aanbrengen.

Bij de opbouw van een grammatica-bestand kun je het beste de volgende indeling aanhouden:

1. ZINNEN op regel 1 (of: iets anders als je geen spaties wil zien in de uitvoer)
2. // commentaar; bijvoorbeeld: hieronder staat de regel met het beginsymbool
// de som van de kansen moet per regel 100 zijn
3. beginsymbool: [kans_1] hulpsymbool_1 ; ... [kans_n] hulpsymbool_n .
4. // commentaar: hieronder volgt de herschrijving van de hulpsymbolen
5. hulpsymbool_1 : ...
6.
7. // commentaar: hieronder volgen de regels met de eindsymbolen
8. hulpsymbool_x : [kans_1] *eindsymbool_1 ; ... [kans_n] *eindsymbool_n .

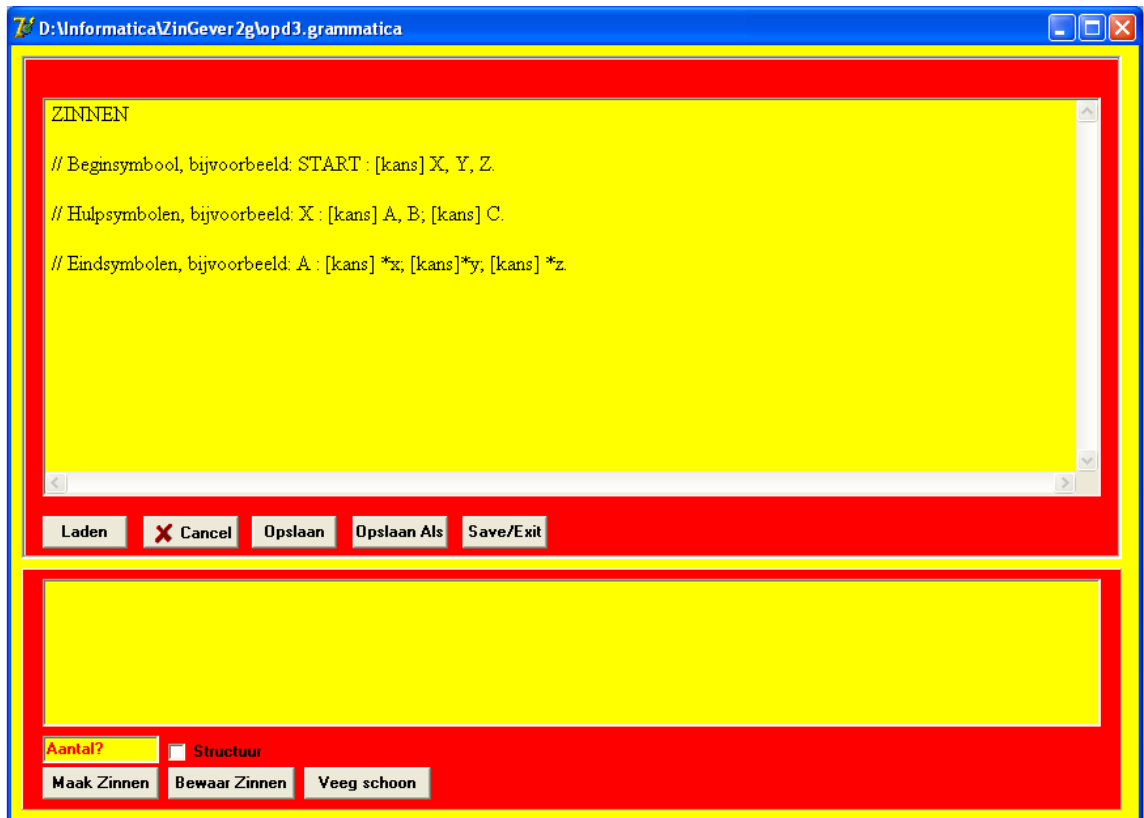
3. Eindelijk naar ZinGever

Ga naar www.stoopned.net, kies **Informatica** en klik in het menu links op **ZinGever**.
Download de zipfile en pak hem uit.

Klik vervolgens op **ZinGever**.

Opdracht 3: genereer ik zinnen?

1. Kies in ZinGever **Bestand**, en dan **Nieuw** (voortaan zul je hier lezen: **Bestand | Nieuw**).
2. Voordat je gaat regels invoeren, moet je een nieuwe bestandsnaam opgeven, bijvoorbeeld **opd3**. Klik dan op **Opslaan**. Er wordt een nieuw scherm geopend.



3. Je ziet bij de naam van het venster dat er een bestand is gemaakt met de naam: **opd3.grammatica**. Verder zie je in het bovenste venster dat er al ZINNEN in regel 1 staat. Wijzig je dit woord, dan zullen er geen spaties in de resultaten van de grammatica staan. Daaronder staan drie commentaarregels.
4. Voer de grammaticaregels uit het vorige hoofdstuk in. Neem de oude regel 2. Voor de overzichtelijkheid mag je regels wit laten tussen groepjes regels die bij elkaar horen.



5. Klik op: **Opslaan** of op: **Save/Exit**. In het laatste geval moet je het bestand weer opnieuw openen.
6. Kies nu: **Maak_Zinnen**. Het programma wil eerst van je weten hoeveel zinnen je wilt laten maken en stelt een maximum van drie voor. Dit mag je veranderen.
7. Stel, het aantal zinnen dat je wilt maken in, bijvoorbeeld: 10.
8. Klik op: **Geef zinnen**.
9. De syntaxis wordt gecontroleerd. Zit daar een fout in, dan krijg je een melding. Verbeter dan eerst de fout. In bovenstaand scherm zit de fout van een punt in een hulpsymbool (pers.vn). Wijzig dit in pers_vn (op twee plaatsen).
10. Een tweede fout zit in de herschrijving van **Vraagzin**. De hulpsymbolen moeten door een komma gescheiden worden van elkaar:

Vraagzin : [100] Pv, O, LV, slot.

11. De grammaticaregels worden nu geteld. Klik door op **ok** en bekijk het resultaat. Je zult misschien zien dat sommige zinnen vaker voorkomen. Klopt het aantal *verschillende* zinnen met je antwoord op vraag 3 van opdracht 2?
12. Sluit nu het venster af. Klik op: **Open_Grammatica** en vervang de grammaticaregel 2 door regel 2a. Klik weer op: **Geef zinnen**. Klopt het aantal verschillende zinnen met je antwoord op vraag 5 van opdracht 2?
13. Via **Bewaar Zinnen** kun je je resultaat opslaan in een bestand.

Opdracht 4: sommetjes maken

We gaan een mini-grammatica maken voor je oom die aan een basisschool in groep 3 les geeft. Hij heeft 30 eenvoudige optelsommetjes nodig om een proefwerk te kunnen maken voor zijn leerlingen.

Een voorbeeld van zo'n sommetje is: $4 + 2 =$

De uitkomst bedenken de leerlingen zelf wel. Je ziet natuurlijk al snel dat de regels een boompje moeten kunnen maken, als onderstaand:

```

      Som
      |
-----|-----
cijfer plus cijfer is
 |    |    |    |
 4    +    2    =

```

1. Maak een grammatica in *ZinGever* en genereer die 30 sommetjes. Sla de grammatica op onder de naam: **sommen**.
2. Hoe kun je het probleem van de kopieën snel opvangen?
3. Breid de grammatica uit met andere bewerkingen: aftrekken, vermenigvuldigen en delen (-, x, :) (Nee, geen smiley!).
4. Probeer tenslotte je grammatica uit te breiden tot sommen van getallen tot 100. Bedenk daarbij dat je getallen onder de 10 zonder 0 schrijft (dus: **9** en niet: **09**), maar dat je nul wél nodig hebt voor de tientallen.
5. Sla je eindresultaat op in **sommen2**.

Opdracht 5: getallen in letters

We gaan een mini-grammatica maken voor je iemand die Nederlands als vreemde taal wil leren. We beperken ons tot de getallen in het Nederlands. Je grammatica moet in **ZinGever** een aantal woorden, zoals de volgende opleveren:

```

Twaalf =
Vijfentwintig =
Honderd drieëndertig =
Zeshonderd vierenvijftig =

```

Zie voor gedetailleerde regels de bijlage bij opdracht 5.
Sla de grammatica op onder de naam: **vijf**.

Opdracht 6: simpele zinnen

1. Probeer nu een grammatica te schrijven voor korte, Nederlandse zinnestjes die bestaan uit een eigennaam, gevolgd door een werkwoord in de verleden tijd. Gebruik werkwoorden die geen lijdend voorwerp of meewerkend voorwerp bij zich kunnen verdragen, zoals: blaffen, groeien, klimmen, zitten, slapen. Bijvoorbeeld:

Jan sliep

Tip: kijk op het voorblad. Maak gebruik van je kennis van de woordsoorten.

Onderwerp $\rightarrow$ zn

Pv $\rightarrow$ zww_ovt

zn $\rightarrow$ Jan | Piet | Elsje | Anna

Kies je eigen namen en je eigen werkwoorden. Sla de grammatica op onder de naam: **simpelzin**.

2. Breid je grammatica uit met de persoonlijke voornaamwoorden (ik, jij, hij, wij, jullie, zij) die als Onderwerp kunnen voorkomen. Genereer een aantal zinnen en beantwoord dan de volgende vraag:
Welk probleem doet zich voor als je kijkt naar het getal van Onderwerp en Pv?

Opdracht 7: kwadratische vergelijkingen

Maak een grammatica waarmee kwadratische vergelijkingen gegenereerd kunnen worden. Ze moeten de volgende vorm hebben:

$$ax^2 + bx + c = d$$

Hierbij zijn a, b, c en d natuurlijk getallen uit de verzameling {1, 2, 3, 4, 5, 6, 7, 8, 9, 0}, maar let op: je schrijft natuurlijk nooit:

$$0x^2 + 1x + 0 = d$$

Bedenk daar een elegante oplossing voor. Je mag het kwadraatje gewoon laag schrijven:

$$ax^2 + bx + c = d$$

Sla je grammatica op onder de naam: **kwadraat**.

Opdracht 8: scheldwoordgenerator

Maak een grammatica die zelfstandignaamwoordgroepen maakt die bestaan uit één of meer bijvoeglijke naamwoorden, gevolgd door één zelfstandig naamwoord.

Kies voor de zelfstandige naamwoorden dieren, zoals: aap, olifant, slang, rat, ezel, enzovoort. Neem voor de bijvoeglijke naamwoorden niet al te gunstige woorden, zoals: lelijk, vies, vuil, et cetera.

Sla je grammatica op onder de naam: **schelden**.

Opdracht 9: Haakjes wegwerken

Maak een grammatica die zinnnetjes maakt als de volgende:

$$(1x + 20)(5x + 2)$$

Ook hier moet je rekening houden met de gedachte dat voor de x nooit een 0 voor hoeft te komen.

Sla je grammatica op onder de naam: **haakjes**.

Bijlage bij opdracht 5

Schrijfwijze van samengestelde hoofd- en rangtelwoorden

bron: www.kun.nl/e-ans/index.php3

[7-5-1]



- 1 De volgende hoofdtelwoorden schrijft men in één woord:
 - alle hoofdtelwoorden tot 100, bijv.: *eenentwintig*, *zevenenveertig*, *achtennegentig*
 - alle veelvouden van *honderd* en *duizend* die gevormd worden met een hoofdtelwoord tot 100 of met een veelvoud van 100, bijv.: *driehonderd*, *zestienhonderd*, *vijfenzestighonderd*; *zevenduizend*, *vierennegentigduizend*, *vijfhonderdduizend*
- 2 De volgende hoofdtelwoorden schrijft men niet aan elkaar:
 - alle hoofdtelwoorden boven 100 (maar niet de veelvouden van 100), bijv.: *honderd (en) een*, *tweehonderd vijfendertig*, *negentienhonderd tachtig*, *vijftienduizend negenhonderd (en) twaalf*
 - alle veelvouden van *duizend* die gevormd worden met één van de hier vlak boven genoemde hoofdtelwoorden, bijv.: *honderd (en) een duizend*, *achthonderd (en) zevenentwintig duizend*
 - alle veelvouden van *een miljoen*, *een miljard*, enz., bijv.: *eenentwintig miljoen*, *dertig miljard*