Werkcollegeopdrachten
Discrete Wiskunde

Periode 2 en 3

J. Foppele

Opleiding Informatica
Noordelijke Hogeschool Leeuwarden

Oktober 2013

Woord vooraf

In dit dictaat staan een groot aantal opdrachten die horen bij het vak Discrete Wiskunde.
De opdrachten dienen ter oefening en verheldering van de stof die in het instructiecollege aan bod komt en die beschreven staat in het dictaat Discrete Wiskunde (door D. Bruin en J.M. Jansen).

De opdrachten moeten deels gemaakt worden m.b.v. Amanda. Het Amanda-programma is via BlackBoard te downloaden. Waar mogelijk moet je de opdrachten waarbij Amanda gebruikt wordt voorafgaand aan het werkcollege maken. Verder is het werkcollege bedoeld om zelfstandig met de opdrachten aan de slag te gaan. De docent kun je om uitleg of assistentie vragen.

Per week is aangegeven welke opdrachten je moet maken. Op het werkcollege volgend op de aangegeven week moet je de opdrachten ter goedkeuren aanbieden aan de docent. Als de opdrachten naar behoren zijn gemaakt, worden deze afgetekend in onderstaand schema.

Let op: Alleen als alle opdrachten tijdig zijn afgetekend, kun je deelnemen aan de betreffende deeltoetsen aan het eind van periode 2 en periode 3.

Aftekenlijst werkcollegeopdrachten

Naam: ……..

Periode	weeknr.	Opdracht(en)	Datum voldaan	Paraaf docent
2	46	2.1		
2	47	2.2		
2	48	2.3		
2	49	2.3		
2	50	2.4		
2	51	2.5		
2	02	Uitloop		
3	06	3.1		
3	07	3.2		
3	09	3.3		
3	10	3.4		
3	11	3.5		
3	12	3.6		
3	13	Uitloop		

Werkplaats Diwi opdracht 2.1

Logica

Opgave 1

Maak waarheidstabellen voor

a (~p ∨ q) ∧ p
b (p -> q) ∨ q
c (p ∧ q) ∨ (p -> q)
d (p -> r) ∧ (q -> r)
e (r -> p) \ / (r ∧ ~p)

Opgave 2

Geef de ontkenning van de volgende uitspraken

a p ∧ q
b p ∨ ~q
c p -> q

Opgave 3

De operaties + is associatief. Dit betekent dat (a + b) + c hetzelfde resultaat heeft als a + (b + c). Ga na voor de volgende operaties na of ze ook associatief zijn: -, *, /, div, mod, ^

Opgave 4

De exclusive-or operatie ⊕ is gedefiniëerd door de volgende waarheidstabel

p	q	p ⊕ q
False	False	False
False	True	True
True	False	True
True	True	False

Bedenk een formule voor p ⊕ q (dus mbv. ∧, ∨ en ~)

Opgave 5

De nand operatie (not and) is gedefiniëerd door de volgende waarheidstabel

p	q	p nand q
False	False	True
False	True	True
True	False	True
True	True	False

Bedenk een formule voor p nand q (dus mbv. ∧, ∨ en ~)

Kennismaking met Amanda

Het volgende practicum kan worden gedaan met Amanda, Amanda++ of Amanda2.0. Aangeraden wordt om zelf uit te zoeken welk programma je het makkelijkste vind werken.

Opgave 6

Typ zelf een aantal rekenkundige expressies in en ga na of Amanda zich houdt aan de regels voor prioriteiten en associatie.
bv:
5 + 3
12 * 8
3 * 5 + 1
7 - 3 - 1
7 - (3 - 1)
7 ^ 2 ^ 3
(7 ^ 2) ^ 3

Amanda bevat een groot aantal ingebouwde functies. Via de help kun je informatie over deze functies vinden. Probeer informatie over de functies trunc en round te vinden.
Evalueer verder de volgende expressies:
sin 1.5
cos 0
sin (pi//2)

Opgave 7: div en mod

Evalueer de volgende expressies
10 div 3
10 / 3
10 // 3
10 mod 3

Wat voor operatie is mod? Probeer een verband aan te geven tussen div (/) en mod.

Opgave 8: Booleaanse expressies

Evalueer de volgende expressies
1 < 3
1 > 3
2 = 1 + 1
4 ~= 2 + 3
~True
~False
1 < 3 \/ 4 > 6
1 < 3 /\ 4 > 6

Opgave 9: Lijsten

Evalueer de volgende expressies
[1,2,3]
hd [1,2,3]
tl [1,2,3]
sum [1,2,3]
prod [1,2,3]
[1..10]

sum [1..10]
[1,3..20]
[1..10] ! 0
[1..10] ! 3
take 3 [1..10]
drop 3 [1..10]
and [1 < 3, 5 < 7, 4 ~= 9]
or [7 < 3, 5 > 7, 4 ~= 9]

Probeer nu zelf een expressie te vinden die de som van alle 3-vouden van 3 tot 99 berekent.

Opgave 10: Strings

Evalueer de volgende expressies
“hallo allemaal”
hd “hallo”
tl “hallo”
“hallo” < “hello”
Wat denk je dat de Amanda representatie van een string is?

Opgave 11: Tuples

Een lijst moet elementen van dezelfde soort bevatten, tuples kunnen elementen van verschillende soort bevatten. Evalueer de volgende expressies
(1,”hallo”)
(True,”aap”,1)

Opgave 12: Foute expressies

Evalueer de volgende expressies en probeer de foutmelding te begrijpen
4 = (1 + 3
1 = +3
[1,”hallo”]
1 < 2 < 3
1 ! 3
1 @ 2
1 = “hallo”

Werkplaats Diwi opdracht 2.2

Functies

Het doel van dit practicum is zelf een aantal functies te definiëren.

Opgave 1

Open nu de file diwi1.ama (bevat gem, verhoog etc.....) in Amanda. Je kunt deze file vinden op blackboard. Run de file indien nodig.
Typ nu de volgende expressies in de command line van het console-venster:

```
> gem 4 7

> verhoog 13

> ing 3 6
```

Opgave 2

We voegen nu zelf een aantal functiedefinities toe aan diwi1.ama. Definiëer de volgende functies:

`cirkelOppervlak r`, die gegeven een straal r de opervlakte van een cirkel met straal r berekent. De constante pi is al gedefiniëerd in Amanda.

`opbrengst b r n`, berekent gegeven een bedrag b een rente percentage r, de opbrengst na n jaar.

Activeer de file opnieuw en test de functies uit. Vergeet bij het verlaten van Amanda niet de file te saven.

Opgave 3

Probeer nu zelf de volgende functie.

```
f x = 1, if x = 1
    = 2, if x = 2
```

Ga na wat er gebeurt als je `f 3` evalueert.

Opgave 4

Definiëer een functie `alleGelijk` met 3 argumenten. alleGelijk geeft True als resultaat als alle 3 argumenten dezelfde waarde hebben, anders False.

Opgave 5

Definiëer een functie `alleVerschillend` met 3 argumenten. alleVerschillend geeft True als resultaat als alle 3 argumenten een verschillende waarde hebben, anders False.

Opgave 6

Definiëer een functie `hoeveelGelijk` met 3 argumenten, die het (grootste) aantal gelijke getallen als resultaat teruggeeft.

Lijsten

Opgave 1

Test zelf uit wat er gebeurt bij `take` en `drop` als n negatief is, of groter dan het aantal elementen van de lijst.

Opgave 2

Definiëer een functie `derde` die je het derde element van een lijst geeft zonder ! te gebruiken.

Opgave 3

Definiëer een functie `plak3 xs ys zs` die drie lijsten aan elkaar plakt.

Opgave 4

Hoe selecteer je uit een lijst het derde t/m het negende element?

Opgave 5

Lijsten kunnen ook met elkaar vergeleken worden. Probeer te begrijpen wat er gebeurt als je twee lijsten vergelijkt? bv. wat is het resultaat van [1,2,3] < [1,1,4]?
Bekijk ook wat er gebeurt als je strings met elkaar vergelijkt.

Werkplaats Diwi opdracht 2.3

Lijstcomprehensies

Opgave 1

Definiëer mbv. een lijstcomprehensie een functie `verhoog xs`, die alle elementen van de lijst xs van getallen met 1 verhoogt.

Opgave 2

Definiëer mbv. een lijstcomprehensie de functie `tussen3en7 xs`, die van de lijst xs alleen die elementen met een waarde tussen 3 en 7 oplevert.

Opgave 3

Definiëer mbv. lijstcomprehensies de functie `sumsqr` die het volgende berekent:

```
sumsqrs n = 1^2 + 2^2 + .. + n^2
```

Opgave 4

Definiëer mbv. een lijstcomprehensie een functie die alle getallen uit een lijst vervangt door drietallen, 1 * getal, 2 * getal, 3 * getal.
Voorbeeld:

```
> fie [1,3,4,5]
[1,2,3,3,6,9,4,8,12,5,10,15]
```

Opgave 5

d is een deler van n als de deling n / d geen rest heeft (dus als n mod d = 0). Schrijf mbv. een lijstcomprehensie een functie `delers n`, die een lijst van alle delers van n oplevert.

Een getal is een priemgetal als het getal maar twee delers heeft nml. 1 en zichzelf. Schrijf nu een functie `priemgetal n`, die bepaalt of n een priemgetal is (maak gebruik van de functie delers uit de vorige opgave).

Een getal heet perfect als het gelijk is aan de som van zijn delers (kleiner dan het getal zelf). Bv. 6 is een perfect getal (6 = 1 + 2 + 3). Probeer het volgende perfecte getal te vinden.

De grootste gemene deler (ggd) van twee getallen is het grootste getal dat beide getallen deelt. Definiëer een functie `ggd n m`, die de grootste gemene deler van n en m bepaalt (maak gebruik van delers en max).

Opgave 6

Definiëer een functie `pythagoras n` die voor gegeven n een lijst van alle pythagorische drietallen (x,y,z) met x,y,z tussen 1 en n berekent. Voor x,y,z moet dus gelden x^2 + y^2 = z^2. Denk om de efficiency, elk drietal mag maar een keer voorkomen.

Werkplaats Diwi opdracht 2.4

Recursie

Opgave 1a

Definiëer een functie `haakjes` die voor een invoerstring bestaande uit haakjes nagaat of deze goedgevormd is.
(()) is goedgevormd
(())() is goedgevormd
())() is niet goedgevormd
Tip: gebruik een extra parameter die het aantal openhaakjes telt.

Opgave 1b

Breid de bovenstaande functie uit zodat er ook andere haakjes gebruikt kunnen worden.
[()] is goedgevormd
[{}](){} is goedgevormd
[} is niet goed gevormd
([]{{})} is niet goed gevormd
Tip: gebruik een extra parameter een lijst (stack) waarop alle begin haakjes die je tot dusver gezien hebt worden opgeslagen. Als je een sluithaakje ziet moet het beginhaakje boven op de stack staan, anders is er een fout opgetreden.

Opgave 2

Definiëer de functies `bintonum` en `numtobin` voor conversie tussen decimale en binaire getallen. Een binair getal wordt gerepresenteed door een lijst van nullen en enen.

Opgave 3

Definiëer een functie `remove x xs` die alle voorkomens van x uit xs verwijdert.

```
> remove 3 [4,3,5,6,3,9]
[4,5,6,9]
```

Definiëer ook een functie `removeOnce x xs` die alleen het eerste voorkomen van x uit xs verwijdert.

```
> removeOnce 3 [4,3,5,6,3,9]
[4,5,6,3,9]
```

Opgave 4

Definiëer een functie `subset xs ys` die True teruggeeft als xs een deelverzameling is van ys. Maak hierbij gebruik van de voorgedefiniëerde functie `member`.

```
> subset [1,4,2] [1,2,3,5,4]
True
```

Opgave 5

Definiëer de functie `last` die het laatse element van een niet lege lijst teruggeeft.

Opgave 6

Definiëer een functie `langstebeginplateau xs` die van een lijst het langste beginstuk geeft van elementen die gelijk zijn aan het eerste element.

```
> langstebeginplateau [1,2,3]
[1]
> langstebeginplateau [1,1,1,1,2,3]
[1,1,1,1]
```

Sorteren

Er betaan diverse methoden om een lijst van getallen te sorteren. In het diktaat worden selection sort en quicksort beproken. Selection sort sorteert een lijst door herhaald het kleinste element uit de lijst te verwijderen tot er geen elementen meer over zijn. Bij quicksort wordt de lijst steeds op een slimme manier in tweeën gedeeld tot we lijstjes van lengte één of nul overhouden. Hieronder zullen we nog een aantal alternatieve sorteermethoden bekijken.

Opgave 7: Insertion sort

Bij insertion sort maken we gebruik van een functie insert die een getal op de juiste plaats in een gesorteerde lijst inserteerd:

```
> insert 3 [1,2,5,6,7,8]
[1,2,3,5,6,7,8]

> insert 3 []
[3]
```

Geef zelf een recursieve definitie van insert. Vul hiertoe onderstaande definitie aan:

```
insert x [] = [x]
insert x (y:ys) = ..
```

We kunnen een lijst nu sorteren door te beginnen met een lege lijst er daar één voor één de elementen van de te sorteren lijst in te inserteren. Vul nu de onderstaande definitie weer aan:

```
isort [] = []
isort (x:xs) = .. (isort xs)
```

Opgave 8: Merge sort

Een andere sorteermethode is merge sort. Mergesort werkt net zoals quicksort volgens het ‘verdeel en heers’ principe. Ook bij mergesort wordt de lijst steeds in tweeën gedeeld. Bij quicksort werd dit op een slimme manier gedaan, de getallen in de ene helft waren allemaal kleiner dan de getallen in de tweede helft. Bij mergesort hakken we de lijst steeds precies middendoor waarna de twee helften met een recursieve aanroep gesorteerd worden en daarna worden ge-merged.
Mergen is het samenvoegen van twee gesorteerde lijsten tot één gesorteerde lijst.

```
> merge [1,3,5,6,8,9] [2,4,5,6,8]
[1, 2, 3, 4, 5, 5, 6, 6, 8, 8, 9]
```

Voltooi de onderstaande definitie van merge:

```
merge xs [] = xs
```

```
merge [] xs = xs
merge (x:xs) (y:ys) = ..            , if x < y
                    = ..            , otherwise
```

Zoals al gezegd wordt bij mergesort de lijst steeds in tweeën verdeeld waarna de twee helften worden gesorteerd en daarna gemerged. Het sorteren van de twee helften gebeurt ook weer door deze in tweeën te verdelen en daarna de helften weer te sorteren en te mergen. Dit in tweeën delen gaat net zolang door tot de overgebleven lijsten lengte 0 of 1 hebben gekregen. Aan dit soort lijsten valt natuurlijk niet veel te sorteren. Voltooi nu de onderstaande definitie van mergesort:

```
msort [] = []
msort [x] = [x]
msort xs = ..
where es = .. (take m xs)
      ts = .. (drop m xs)
      m = ..
```

Controleer wat er gebeurt als we in dze definitie de tweede regel weg laten. Kun je dit verklaren?

Werkplaats Diwi opdracht 2.5

Recursieve datastructuren en Bomen

Opgave 1 (7.8)

Definieer een functie `spiegelBoom` die een boom spiegelt in een verticale as door het midden.

Opgave 2 (7.9)

Definieer functies, `inorder`, `preorder` en `postorder` voor het afdrukken van een boom.

Opgave 3 (7.10)

Definieer een functie `maxDepth` die het aantal elementen in het langste pad een boom terug geeft.

Opgave 4 (7.11)

Definieer een functie `isVolledig` die na gaat of een boom volledig is.

Opgave 5 (7.12)

Definieer een functie `isEvenwichtig` die na gaat of een boom evenwichtig is.

Opgave 6 (7.13)

Definieer een functie `maakBoom` die een lijst van getallen omzet in een evenwichtige boom. Hint: doe dit door de lijst steeds te halveren.

Opgave 7 (7.15)

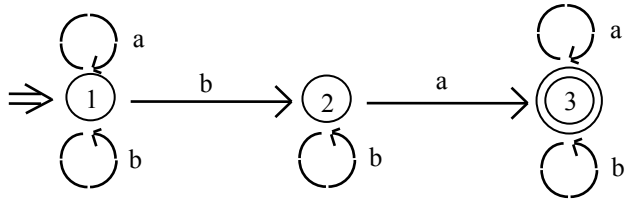
Definiëer een functie `sorteer` die een lijst van getallen sorteert door deze eerst om te zetten in binaire zoekboom en deze zoekboom daarna in in-order volgorde af te drukken.

Werkplaats Diwi opdracht 2.6

Automaten

Opgave 1 (opgave 5.2 diktaat)

Gegeven is de volgende eindige automaat:



a. Is de automaat deterministisch ?
b. Beschrijf de taal van de automaat

Opgave 2 (opgave 5.3 diktaat)

Maak een eindige automaat voor de volgende talen van strings bestaande uit a's en b's door de betekenis van de toestanden goed te definieren:

a. alle strings waarin precies 3 b's voorkomen
b. alle strings waarin na een a altijd precies twee b's komen
c. alle strings waarin een even aantal a's en een even aantal b's voorkomt

Opgave 3 (opgave 5.4 diktaat)

Maak voor elk van de volgende talen een Amanda functie die test of een string tot die taal hoort:

a. alle strings waarin minstens 3 b's voorkomen
b. alle strings waarin na een a altijd precies een b komt
c. alle strings waarin een even aantal a's en een even aantal b's voorkomt

Probeer de functies ook direct in Amanda te programmeren mbv lijstcomprehensies.

Opgave 4 (opgave 5.6 diktaat)

a. Maak een nondeterministische eindige automaat voor de taal van alle strings bestaande uit a's en b's waarin aabaab voorkomt
b. Maak voor de automaat uit onderdeel a. een equivalente deterministische automaat

Werkplaats Diwi opdracht 3.1

Grammatica's

Opgave 1

Gegeven is de volgende grammatica:
X = ab | a X b
Geef ontleedbomen voor de volgende strings:
a. aabb
b. aaabbb
c. aaaaaaabbbbbbb
d. beredeneer dat de taal van X = { a^n b^n | n >= 1 } waarbij a^n betekent n a's achter elkaar.

Opgave 2

Gegeven is de volgende grammatica:
S = X | c S
X = ab | a X b
Geef ontleedbomen voor de volgende strings:
a. caabb
b. ccaaabbb
c. cccccccab
d. wat is de taal van X ?

Opgave 3

Gegeven is de volgende grammatica:
S = d X | c S
X = a X | Y
Y = b Y | ε
Geef ontleedbomen voor de volgende strings:
a. cdaa
b. ccdaaabbb
c. cccccccdab
d. Geef een reguliere expressie en een eindige automaat voor de taal S

Opgave 4

Stel voor de volgende talen grammatica's op (a^n betekent n a's achter elkaar):
a. { a^n b^n | n >= 0 }
b. { a^n b^m | n >= 0 /\ m >= 0 /\ m <= n }
c. { (a^n b^n)* c* | n >= 0 }

Opgave 5

De volgende grammatica kan ook alle rekenkundige expressies aan:
cijfer = 0 | 1 | 2 .. | 9
getal = cijfer | cijfer getal
operator = + | -| * | /
expressie = getal | (expressie) | expressie operator expressie
Geef zoveel mogelijk ontleedbomen voor de volgende rekenkundige expressies en vermeld ook telkens de waarde van de onderdelen:
a. 89 + 007 * 3
b. (8 -90) / 7 * 8
c. (78 + 9 + 78) -(6)
d. Wat zou het nadeel zijn van deze grammatica ?

Werkplaats Diwi opdracht 3.2

Cryptografie

Een leuke manier om een tekst te coderen (omzetten in geheimschrift) is door ieder karakter bitgewijs te x-oren met een van tevoren gegeven byte (een karakter neemt precies 1 byte in beslag). Je kunt nu weer decoderen door te x-oren met hetzelfde byte (ga dit na)!

Opgaven

Maak de volgende functies:

bin c: zet een karakter om naar een lijst van 8 nullen en enen (de binaire representie van de code van het karakter).

```
> bin 'a'
[0, 1, 1, 0, 0, 0, 0, 1]
```

dbin bs: doet het omgekeerde van bin.

```
dbin [0, 1, 1, 0, 0, 0, 0, 1]
'a'
```

xor b d: berekent de xor van b en d (b en d zijn 0 of 1).

```
> xor 1 0
1
```

xorbyte bs ds: x-ort twee lijsten van 8 bits.

```
> xorbyte (bin 'a') (bin 's')
[0, 0, 0, 1, 0, 0, 1, 0]
```

enc b c: codeert het karakter c, door te x-oren met het karakter b.

```
> enc 'A' 't'
'5'
```

Als je het goed gedaan hebt, geldt er nu:

```
> enc 't' (enc 't' 'a')
'a'
```

encrypt b txt: codeert de string txt, door alle karakters te x-oren met het karakter b.

```
> encrypt 't' "hallo allemaal"
T
```

Ook nu moet er weer gelden:

```
> encrypt 't' (encrypt 't' "niet te geloven")
niet te geloven
```

Je kunt dit programma practisch bruikbaar maken mbv. de volgende voorgedefiniëerde Amanda functies:

```
fread fn
fwrite fn txt
```

fread leest de file met de naam fn in en geeft het resultaat als een string:

```
> fread "test.dat"
"hallo allemaal"
```

fwrite schrijft de string txt naar de file fn en geeft True terug als het gelukt is:

```
> fwrite "test.dat" "hallo allemaal"
```
True

Werkplaats Diwi opdracht 3.3

Backtracking

Opgave

Vind m.b.v. backtracking het (unieke) getal n bestaande uit 9 cijfers met de volgende eigenschappen:

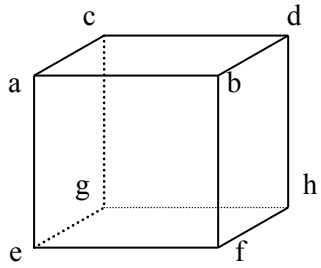
- de cijfers 1 t/m 9 komen precies één keer voor
- de eerste k cijfers van n zijn deelbaar door k voor k loopt van 1 t/m 9

Werkplaats Diwi opdracht 3.4

Backtracking

Opgave

Bij een magische kubus zijn de getallen 1 t/m 8 verdeeld over de hoekpunten zodat de som van elk vlak 18 is.



Bijvoorbeeld met a=1 , b=4, c=8, d=5, e=6, f=7, g=3 en h=2 hebben we een magische kubus want alle vlakken hebben een som van 18.

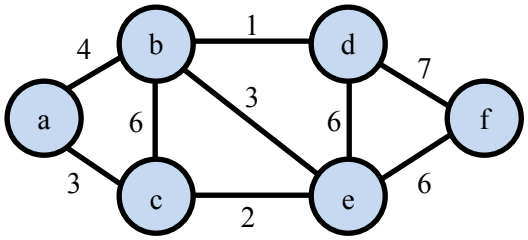
Schrijf nu een Amandafunctie met behulp van backtracking die een lijst van alle magische kubussen oplevert.

Werkplaats Diwi opdracht 3.5

Grafen

Opdracht 1

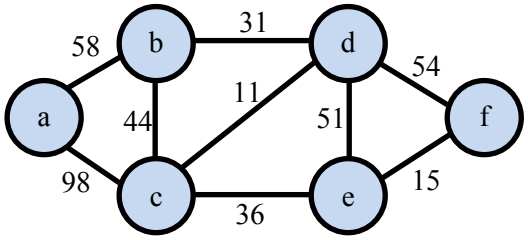
Pas zowel het algoritme van Kruskal als het algoritme van Prim toe op de volgende graaf.



Opdracht 2

Geef het kortste pad tussen de volgende punten:

a. punt b naar punt e
b. punt b naar punt f
c. punt a naar punt f



Werkplaats Diwi opdracht 3.6

Grafen

Opgave 1 (opgave 8.3 diktaat)

Schrijf een amandafunctie kortstepad die de kortste afstand berekent tussen een gegeven beginpunt en een gegeven eindpunt via een gegeven graaf.

Voorbeeld:
kortstepad "a" "c" [("a", [("b", 3), ("c", 6)]), ("b", [("a", 3), ("c", 1)]), ("c", [("a", 6), ("b", 1)])]

het beginpunt is nu "a"
het eindpunt is "b"
de graaf is [("a", [("b", 3), ("c", 6)]), ("b", [("a", 3), ("c", 1)]), ("c", [("a", 6), ("b", 1)])]

Gebruik hierbij de volgende opzet:

```
kortstepad begin eind graaf = vergroot [(begin, 0)] eind graaf

vergroot verzameling eind graaf
= hd eindafstanden                  ,if eindafstanden ~= []
= vergroot verzameling1 eind graaf  ,otherwise
where
  eindafstanden = [a | (x, a) <- verzameling; x = eind]
  buren = [(y, a1+a2) | (x, a1) <- verzameling;
                        (x1, lijst) <- graaf;
                        x1 = x;
                        (y, a2) <- lijst
                        ~(member (map fst verzameling) y)]
  verzameling1 = verzameling ++ ...
```