

# Keuzedeel: doorstroom mbo-hbo

J.B. Bergmann (docent AO, Drenthe College, #6786, Emmen, Assen), v0.22,

27 oktober 2019

## Samenvatting

Dit is het lesmateriaal en allerlei andere relevante informatie voor het Keuzedeel Doorstroom MBO HBO van de opleiding Applicatie-ontwikkeling van het Drenthe College.



Figuur 1: Verschillende MBO en HBO scholen

# 1 Vooraf

In het studiejaar 2014-2015 is door een aantal MBO en HBO scholen in Noord Nederland onderzoek gedaan naar een betere aansluiting tussen de MBO en HBO opleidingen. Enkele relevante conclusies voor het vak Doorstroming MBO - HBO bij de opleiding Applicatieontwikkeling staan in onderstaande tabellen samengevat:

8 Analyseren			
Verskil hbo met mbo	Resultaat	Deelvaardigheden / technieken	Gedrag
<ul style="list-style-type: none"> <li>• abstracter</li> <li>• theoretischer</li> <li>• deductief i.p.v. inductief</li> <li>• transfer van kennis</li> <li>• onderbouwen en beargumenteren van de aanpak en de oplossing van opdrachten of problemen</li> </ul>	<ul style="list-style-type: none"> <li>• student kan informatie systematisch en correct analyseren</li> <li>• student kan algemene regels toepassen op specifieke situaties en verschijnselen verklaren</li> <li>• student kan argumenteren en abstract denken</li> </ul>	<ul style="list-style-type: none"> <li>verband leggen tussen theorie en praktijk</li> <li>probleem definiëren</li> <li>informatie vergelijken en verbanden leggen</li> <li>informatie beoordelen</li> <li>feiten en meningen onderscheiden</li> <li>conclusies trekken</li> </ul>	<ul style="list-style-type: none"> <li>legt verband tussen theorie en praktijk</li> <li>stelt vooraf probleem of vraag vast</li> <li>vergelijkt en verbindt informatie uit verschillende bronnen</li> <li>bepaalt of informatie volledig en betrouwbaar is en onderscheidt hoofd- en bijzaken</li> <li>let erop of het om meningen of feiten gaat</li> <li>trekt logische conclusies uit verzamelde informatie</li> </ul>

Figuur 2: Verschillende analysevaardigheden gevraagd op MBO en HBO

## Inhoud van de set generieke studievoordigheden schematisch weergegeven



Figuur 3: Verschillende studievoordigheden gevraagd op HBO

Zie eventueel de [website](#) van het onderzoek en deze [introductie](#) video.

Er kan onderscheid gemaakt worden tussen generieke ofwel *algemene* en *vakspecifieke* vaardigheden. In figuur 3 worden verschillende generieke studievoordigheden genoemd. Ze zijn wel te onderscheiden maar niet altijd gescheiden: deze vaardigheden lopen in elkaar over.

Voor Applicatieontwikkelaars zijn met name meer abstractere, analytische en mathematische denkwijzen naast sociale en persoonlijke vaardigheden van belang. Een probleemstelling dient niet alleen op creatieve wijze van

verschillende kanten te worden onderzocht maar vergeet voor het bedenken en vastleggen van mogelijke oplossingen nu eenmaal ook abstracte denkwijzen. Er dient veelal vanuit een specifieke probleemsituatie gegeneraliseerd te worden waarna deze algemenere oplossing weer wordt toegepast in het concrete geval. Een gebouwde softwareapplicatie wordt veelal op verschillende manieren in een domeingebied ingezet. Bijvoorbeeld een kaartverkoop-programma voor theaters dat ook kan worden gebruikt voor bioscopen en popconcerten. Hoewel het in deze zalen gaat om stoelbezetting kunnen er (grote) verschillen zijn. Uit implementatieonderzoek kan dan blijken dat een theaterpakket niet geschikt is voor een bioscoop en vice versa.

Een andere voorbeeld is dat een klein, regionale bierproducent zijn groei wil versnellen via een beter informatie-systeem. Wordt zo'n vraagstuk aan een MBO'er voorgelegd dan zal hij zo'n situatie veelal zien als afzonderlijke registraties voor voorraad, order- en financiële administratie. Er worden diverse, losstaande (deel-)oplossingen gebouwd voor deze subadministraties. Een HBO student dient na te gaan of er geen bekende min of meer standaard-werkmethodieken zijn voor zo'n productiebedrijf. Deze zijn beschreven in de bedrijfseconomische c.q. logistieke literatuur onder ERP, MRP-I en MRP-II modellen. Communicatie met vakspecialisten (niet-informatici) komt dan ook veelvuldig voor.

Van de softwarebouwer wordt gevraagd dat hij/zij naast communicatief vermogen (o.a. teamwerk, presentaties) beschikt over analytisch denkvermogen.

Wat is analytisch denkvermogen? Volgens de website [Leren.nl](http://Leren.nl) is dat: "Analytisch denken is het systematisch ontleden van een complex probleem in zijn elementen. Analytisch denken is nauw verwant aan kritisch denken. Je maakt duidelijk onderscheid tussen hoofd- en bijzaken, tussen symptomen en oorzaken, tussen feiten en opvattingen. Je bepaalt eerst de hoofdlijnen en detailleert later. Je ziet trends en patronen in ogenschijnlijk losstaande gegevens. Je gaat logisch en methodisch te werk."

In deze lesmodule wordt aan deze competenties gewerkt door aandacht te besteden aan enkele relevante onderwerpen uit de **predicatenlogica, grafentheorie, verzamelingenleer en toestandsdiagrammen**. Steeds is daarbij gepoogd studenten aan de hand van oefeningen en opdrachten de praktische toepassingen er van te laten zien en uit te werken. De MBO student is een doener die met name in zijn opleiding heeft gewerkt aan praktische en concrete kleinschalige softwareprojecten. Een HBO student heeft meer theoretische kennis van technieken en methoden en werkt veel in projectgroepen aan **grotere softwareapplicaties**.

Door het volgen van dit keuzedeel wordt de MBO-afstudeerder enkele handvatten aangereikt om zowel algemene als vakspecifieke vaardigheden verder te ontwikkelen. Zie hierna bij Onderwerpen.

## 2 introductie

In de softwareontwikkeling speelt ‘de logica’ een centrale rol. En toch wordt het vak niet apart onderwezen in het MBO-onderwijs. In het HBO-onderwijs wordt op een ‘abstracter’ of algemenere manier nagedacht over hoe een softwarepakket moet werken. Daarbij stuit je al snel op allerlei algemene technieken die hun oorsprong vinden in de wiskunde of de logica.

Het gebruik van wiskunde en logica biedt een aantal voordelen:

1. De systeemeigenschappen kun je daardoor preciezer (nauwkeuriger) specificeren. Natuurlijke taal is niet precies (ambigu). Een programmeertaal vereist dat een computeralgoritme of de betekenis van een keyword eenduidig is. In PHP bijvoorbeeld gebruik je een for-loop om exact aan te geven hoe vaak je een of meerdere statement(s) uitvoert. Weet je dat aantal vooraf nog niet dan gebruik je een while-loop. Bij een if-statement geef je exact aan onder welke voorwaarde(n) je een of meerdere statements uitvoert. De betekenis van woorden (keywords) als ‘for’, ‘while’ and ‘if’ zijn bij programmeurs exact bekend. De vertaler (compiler of interpreter) geeft voorspelbare resultaten. En ‘leren programmeren’ is o.a. het bekend worden met zulke taalconstructies. In een natuurlijke of mensentaal als het Nederlands hebben woorden vaak meerdere betekenissen en moet uit de context bepaald worden welke betekenis het woord in deze context heeft. Neem het woord ‘bank’. De zin ‘Jan zit op de bank’ kan op verschillende manieren uitgelegd worden:
  - (a) a. zit Jan op een bankje in het park?
  - (b) b. zit Jan op een gebouw waarin een financiële instelling is gehuisvest?
  - (c) c. is Jan op kantoor actief als bankmedewerker?
  - (d) d. of zit Jan bij zichzelf thuis voor de tv op een zitbank?
  - (e) e. etc.
2. Nauwkeurig en formeel geformuleerde systeemeigenschappen bieden de mogelijkheid om ze door een computerprogramma te laten verifiëren. Dit voorkomt veel fouten en bespaard veel tijd.
3. Stapsgewijze manier van denken: opsplitsen van het grote vraagstuk in deelproblemen, ...
4. Planning maken.

Tenslotte, onderscheid tussen ICT en AO. In dit lesprogramma richten we ons op de vaardigheden van een AO-er.

## 3 Onderwerpen

In dit keuzedeel worden een aantal methoden en technieken kort behandeld zodat je als AO-er enige ideeën krijgt van de mogelijkheden. Wij zullen aandacht besteden aan o.m.:

1. waarheidstabellen (Booleaanse logica)
2. redeneermechanisme Propositielogica (tot aan tussenkopje ‘Syntaxis en semantiek’). Vooral de twee wetten van De Morgan krijgen aandacht.
3. grafentheorie
4. research vaardigheden,
5. studievvaardigheden,
6. begrijpelijk lezen,
7. schrijfvaardigheden,
8. samenwerking, interviewtechnieken en presenteren: vlog en reflectieverslag

9. ...

We hebben de nadruk gelegd op verschillende modellen en zienswijzen die door software-ontwikkelaars worden gehanteerd. Het gaat daarbij niet zo zeer om de concrete programmeertalen, hulpmiddelen, etc. (die worden behandeld in het reguliere studieprogramma van AO) maar vooral om de principes van logica en wiskunde. Daarnaast wordt ook aandacht besteed aan samenwerking in projectgroepen, presentaties, lezen en schrijven. Dit is overeenkomstig de voorgeschreven werkwijze in het relevante Kwalificatiedossier K0125 (voornamelijk de generieke studievaardigheden). Deze is integraal opgenomen in de bijlagen. Voor de logica is (nog) geen KD beschikbaar.

Omdat daarnaast een aantal gewenste analysevaardigheden ook worden onderwezen op basis van K0205 Voorbereiding HBO Wiskunde voor de techniek is deze eveneens integraal opgenomen in de bijlagen. Als toekomstige HBO-student kun je zo eenvoudig lezen wat er van je wordt verwacht.

## 4 Waarheidstabellen

### 4.1 inleiding

Waarheidstabellen worden gebruikt om te onderzoeken of een formule bestaande uit een of meerdere condities *vervulbaar* is. Daarnaast kunnen ze worden gebruikt om te bepalen of een redenering en de daaruit volgende gevolgtrekking *geldig* is. Tevens kan worden onderzocht of bijvoorbeeld twee formules logisch gelijkwaardig (equivalent) zijn. In een waarheidstabel worden *alle* mogelijkheden om aan de *propositievariabelen* waarheidswaarden toe te kennen geïnventariseerd. Elke *rij* in de tabel correspondeert met een waardering (valuatie), terwijl elke *kolom* overeenkomt met een formule. In de tabel wordt opgenomen of de formule onder de gegeven valuatie waar is of niet.

Stel: P en Q zijn bepaalde ‘uitspraken’ die een waarheidswaarde hebben van WAAR/TRUE of ONWAAR/FALSE. Bijvoorbeeld P: ‘Jan is 17 jaar’ en Q: ‘Quintess is 18 jaar oud’. Er zijn nu 4 combinaties denkbaar:

P	Q	P	&	Q
T	T	T	T	T
T	F	T	F	F
F	T	F	F	T
F	F	F	F	F

Deze waarheidstabel kan worden opgesteld zonder dat we weten welke van de uitspraken P en Q in werkelijkheid klopt. Het is een overzicht van alle denkbare combinaties. Met andere woorden: zo’n waarheidstabel is onafhankelijk van de *inhoud* van P en Q.

studie-aanwijzing: Als je zo’n tabel probeert op te zetten doe dat dan per rij.

Indien de feitelijke leeftijden van Jan en Quintess bekend zijn, kan vastgesteld worden welk van de 4 regels de juiste is. Op basis daarvan kan dan de conclusie worden getrokken of de *samengestelde uitspraak* ‘Jan is 17 jaar EN Quintess is 18 jaar oud’ ofwel  $P \& Q$  WAAR of ONWAAR is. Stel: Jan is 17 jaar en Quintess is 19 jaar. er geldt dan dat regel 2 van de waarheidstabel actueel of geldig is.

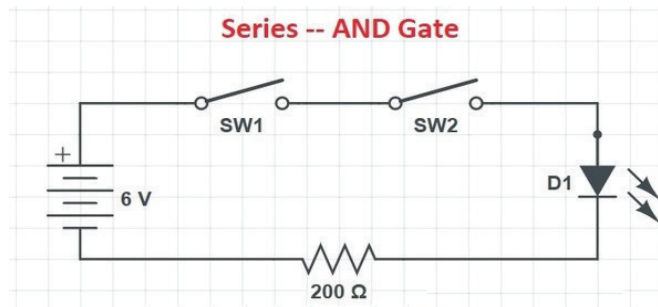
Let op: het moet je duidelijk zijn dat welke letters voor de condities (proposities) worden gebruikt er niet toe doen. En het maakt ook niet uit of de tabel wordt gevuld met 1 en 0 in plaats van met T en F. Kortom in de volgende tabel staat qua logica hetzelfde als in de voorgaande:

A	B	A	&	B
1	1	1	1	1
1	0	1	0	0
0	1	0	0	1
0	0	0	0	0

Welk van deze varianten je gebruikt, is vooral afhankelijk wat jezelf prettig vindt. Wat dat betreft kun je het vergelijken met de naamgeving van variabelen in broncode. Of daar een bepaalde variabele de naam ‘som’ heeft, is niet relevant voor de werking van de software. Het gaat om de inhoud (betekenis) en niet om de naamgeving. In deze tekst zullen we doorgaans gebruik maken van de eerste vorm.

Met andere woorden, in een AND beslissing moeten beide condities WAAR zijn om de gehele statement WAAR te laten zijn. AND heet ook wel : "logisch product".

In de elektromechanica wordt een AND-functie toegepast door schakelaars in serie te zetten. Er kan dan immers alleen stroom lopen als beide schakelaars zijn ingedrukt. In de elektronica heet een circuit dat een AND-functie vervult een AND-gate.



Figuur 4: AND-gate

Belangrijk is dat met waarheidstabellen vooral de *structuur* en *completeheid* van de redenering c.q. gevolgtrekking gecheckt wordt. De completeheid wil hier zeggen: alle mogelijkheden worden onderzocht en de structuur ligt vast in de wijze van redeneren.

denkoefening: hoe zou jij de waarheidstabel voor AND gaan bouwen in software: schrijf dat op in pseudocode.

## 4.2 enkele logische operatoren: NOT, AND, OR, XOR

De eenvoudigste operator is de *ontkenning*. Deze keert de logische waarde van een uitspraak om: Kortom: als een

P	$\sim$ P
T	<b>F</b>
F	<b>T</b>

uitspraak de waarde TRUE heeft dan wordt deze door het gebruik van de NOT-operator FALSE. En vice versa: FALSE wordt TRUE.

De AND-operator hebben we hiervoor al besproken in de inleiding. Daar gebruiken we het &-teken maar vaak wordt daarvoor  $\wedge$  gebruikt. We gaan verder met de OR-operator. Hiervoor gebruikt men veelal het symbool  $\vee$ .

P	Q	P $\vee$ Q
T	T	<b>T</b>
T	F	<b>T</b>
F	T	<b>T</b>
F	F	<b>F</b>

En de XOR-operator, die veelal wordt weergegeven als  $\oplus$ :

P	Q	P $\oplus$ Q
T	T	<b>F</b>
T	F	<b>T</b>
F	T	<b>T</b>
F	F	<b>F</b>

De gelijkwaardigheid-operator, veelal aangegeven met  $\equiv$  of  $\leftrightarrow$ , heeft de volgende betekenis:

Men noemt dit ook wel de logische gelijkheid. Zie voor deze laatste term bijvoorbeeld deze wiki..

Als je de rode kolommen in de tabellen van de XOR-operator en gelijkwaardigheids-operator met elkaar vergelijkt, dan zie je dat ze elkaars tegengestelde zijn. Als een bepaalde regel in de waarheidstabel van XOR FALSE is, is die van de gelijkwaardigheids-operator juist TRUE, en andersom. Met andere woorden, er geldt: de ontkenning van de

P	Q	P	≡	Q
T	T	T	<b>T</b>	T
T	F	T	<b>F</b>	F
F	T	F	<b>F</b>	T
F	F	F	<b>T</b>	F

ene is de waarde van de andere. Of in symbolen: NOT XOR is gelijk aan de GELIJKWAARDIGHEIDOperator. Dus:  $\sim \oplus \equiv$  of mooier:

$$\sim \oplus \leftrightarrow \equiv \quad (1)$$

Er zijn natuurlijk verschillende combinaties mogelijk. het gaat dan in feite om verschillende redenering die tot dezelfde conclusie leiden. Bijvoorbeeld de ontkenning van de P tesamen met een OF tussen P en Q geeft:

P	Q	<b>~</b> P	∨	Q
T	T	<b>F</b>	<b>T</b>	T
T	F	<b>F</b>	<b>F</b>	F
F	T	<b>T</b>	<b>F</b>	T
F	F	<b>T</b>	<b>F</b>	F

We gaan nu eerst aandacht geven aan een operator die de nodige uitleg behoeft en komen later uitvoeriger terug op de verschillende gelijkwaardige beweringen.

### 4.3 de logische operator IMPLICATOR

De implicator, aangeduid met het teken  $\rightarrow$ , vinden programmeurs terug in programmacode als IF-statement:

$$A \rightarrow B \quad \text{ofwel:} \quad \text{if } \{A\} \text{ then } \{B\} \quad (2)$$

Het gaat bij de logische implicator echter niet alleen om de waarheidswaarde van **antecedent** A maar ook die van gevolg (de consequent) B.

We lichten het toe aan de hand van een eenvoudige voorbeeld:

Als x is positief, dan x is even.

Hierin is A gelijk aan 'x is positief' en B is gelijk aan 'x is even'.

A noemen we de *antecedent* terwijl B de *consequent* wordt genoemd van de implicator. Zowel A als B en ook  $A \rightarrow B$  kan de waarde TRUE of FALSE aannemen. Het blijkt dat de waarde van de samengestelde uitspraak  $A \rightarrow B$  bepaald wordt door de waarde van A én door de waarde van B. Net als bij de eerder behandelde waarheidswaarden kunnen we ook hier een waarheidstabel opstellen. Deze luidt als volgt:

P	Q	P	→	Q
T	T	T	<b>T</b>	T
T	F	T	<b>F</b>	F
F	T	F	<b>T</b>	T
F	F	F	<b>T</b>	F

Bekijk deze tabelregels goed. Als de antecedent TRUE is (hier aangeduid met de P) terwijl de consequent Q FALSE is, dan is de implicator als geheel FALSE. In alle andere gevallen is de implicator TRUE. We lichten dit toe aan de hand van ons voorbeeld.

Stel: x = 6: dan geldt: A = TRUE, B = TRUE x = 7: dan geldt: A = TRUE, B = FALSE x = -2: dan geldt: A = FALSE, B = TRUE x = 0: dan geldt: A = FALSE, B = TRUE (!)

We zien dat de waarheidstabellen van  $A \rightarrow B$  en  $A \equiv B$  in één regel afwijken van elkaar. Dat is in deze tabellen de derde regel. Deze is voor  $A \rightarrow B$  gelijk aan TRUE terwijl deze voor  $A \equiv B$  gelijk is aan FALSE.

P	Q	P	→	Q
T	T	T	<b>T</b>	T
T	F	T	<b>F</b>	F
F	T	F	<b>F</b>	T
F	F	F	<b>T</b>	F

#### 4.4 de logische gelijkwaardigheid

**todo:** Voorbeeld geven van gebruik van logische gelijkheid:

#### 4.5 opdrachten

**Studietip:** zet de volgorde waarin je de kolommen onderzoekt aan de onderkant er bij. Zie in het voorbeeld in de onderstaande tabel de regel met 'step':

$p$	$q$	$(p \wedge q)$	$\vee$	$\neg(p \rightarrow q)$	
0	0	0	<b>0</b>	0	1
0	1	0	<b>0</b>	0	1
1	0	0	<b>1</b>	1	0
1	1	1	<b>1</b>	0	1
step	1	1	2	4	3 2

Figuur 5: De verschillende stappen die zijn gemaakt bij het invullen.

Onderzoek de volgende stellingen:

Stel: A en B zijn TRUE en X en Y zijn FALSE. Wat is dan de waarheidswaarde van de volgende uitspraken:

1. not (A or X)
2. (not A) or (not X)
3. (not B) and (not Y)
4. not (B and Y)
5. A or (X and Y)
6. (A or X) and Y
7. (A or X) and (B or Y)
8. not (A or X) and not (A or Y)
9. not (A  $\Rightarrow$  B)

#### 4.6 Programmacode:

### 5 Predicatenlogica: redeneermechanismes

#### 5.1 Equivalenties

Om je aan de symbolen te laten wennen, volgen hieronder enkele belangrijke eigenschappen. Probeer ze te begrijpen. Daarbij helpt het als je met pen en papier de regels op schrijft:



Zoals bij het optellen geldt  $1 + 2 = 2 + 1$ , zo geldt ook:

$$(P \wedge Q) \equiv (Q \wedge P) \quad (3)$$

Dit wordt de commutatieve eigenschap genoemd. In dit geval heeft die betrekking op een *conjunction*. Laat je dus niet afschrikken door de gebruikte symbolen en probeer te begrijpen wat er staat. Daarbij helpt het als je probeert ze te vergelijken met bekende rekenregels (zoals die we net noemden met  $1 + 2 = 2 + 1$ ) Hier volgen nog enkele (min of meer bekende of voor de handliggende) eigenschappen:

De ontkenning van een ontkenning, dus een dubbele ontkenning, is gelijk aan de oorspronkelijke conditie:

$$\neg\neg P \equiv P \quad (4)$$

Commutatieve eigenschap (voor alternatieven):

$$(P \vee Q) \equiv (Q \vee P) \quad (5)$$

Commutation (equivalences)

$$(P \leftrightarrow Q) \equiv (Q \leftrightarrow P) \quad (6)$$

Association (conjunctions):

$$P \wedge (Q \wedge R) \equiv (P \wedge Q) \wedge R \quad (7)$$

Association (alternatives):

$$P \vee (Q \vee R) \equiv (P \vee Q) \vee R \quad (8)$$

Distribution (1):

$$P \vee (Q \wedge R) \equiv (P \vee Q) \wedge (P \vee R) \quad (9)$$

Distribution (2):

$$P \wedge (Q \vee R) \equiv (P \wedge Q) \vee (P \wedge R) \quad (10)$$

## 5.2 De wetten van De Morgan

In sommige gevallen is het mogelijk om een combinatie van logische operatoren te vereenvoudigen. Bij het programmeren kan dit zeer handig zijn omdat daardoor het inzicht in de werking van de broncode beter wordt. De vereenvoudiging vereist natuurlijk dat de herschrijving tot precies hetzelfde resultaat leidt. Met andere woorden, de herschrijving is functioneel volledig gelijkwaardig aan de eerdere **kode**. Een bekend voorbeeld van zo'n vereenvoudiging zijn de twee 'wetten' die de Engelse wiskundige Augustus De Morgan in de 19e eeuw heeft ontdekt en daarom bekend zijn als 'de morgan's laws' of kortweg 'De Morgan'. Het aardige van deze 'wetten' of regels is dat ze zowel geformuleerd kunnen worden vanuit de propositielogica, verzamelingen-leer als electronica. Wij besteden kort aandacht aan de notatiewijzen in eerste twee vakgebieden. De twee wetten waar wij hier aandacht aan besteden luiden als volgt:

De Morgan's Law I (negation of alternatives = de ontkenning van alternatieven):

$$\neg(P \vee Q) \equiv \neg P \wedge \neg Q \quad (11)$$

Omdat een auteur soms wil benadrukken dat de NOT-operator (**inde** formule is dat:  $\neg$ ) een hogere prioriteit heeft dan een AND-operator (hier het symbool:  $\wedge$ ), met andere woorden de beide NOT-operators worden eerst uitgevoerd en daarna de AND-operator, kom je ook de volgende schrijfwijze tegen met haakjes. Strict genomen zijn ze echter redundant.

$$\neg(P \vee Q) \equiv (\neg P) \wedge (\neg Q) \quad (12)$$

De Morgan's Law II (negation of conjunction = de ontkenning van de samenvoeging):

$$\neg(P \wedge Q) \equiv \neg P \vee \neg Q \quad (13)$$

En ook hier zijn er eventueel twee extra haakjes te gebruiken:

$$\neg(P \wedge Q) \equiv (\neg P) \vee (\neg Q) \quad (14)$$

Zie voor een duidelijke schrijfwijze en meer achtergrondinformatie eventueel De Morgan op wikipedia.

### 5.3 opdrachten

OPDRACHT a Bewijs met behulp van waarheidstabellen dat de eerste wet inderdaad klopt!

OPDRACHT b Bewijs met behulp van waarheidstabellen dat de tweede wet ook klopt!

OPDRACHT c Onderzoek op dezelfde wijze of je de NOT-operator aan de rechterzijde buiten de haakjes mag brengen, dus of de volgende stellingen kloppen:

$$\neg(P \vee Q) \equiv \neg(P \wedge Q) \quad (15)$$

$$\neg(P \wedge Q) \equiv \neg(P \vee Q) \quad (16)$$

OPDRACHT d De antwoorden op opdracht c waren zonder de uitwerking via de waarheidstabellen ook te geven. Wat zou namelijk de eventuele gelijkheid van de linker- en rechterzijde impliceren?

Opdracht e

Bekijk nu enkele malen de volgende film op youtube: Arduino programmeer theorie het staat ook op: bbbb maar dan even naar beneden doorlopen.

### 5.4 een toepassing

Nu we de basis hebben besproken en begrijpen, zullen we kijken hoe we deze kennis kunnen gebruiken. We doen dat aan de hand van enkele (eenvoudige) stukken broncode. Hieronder een eenvoudige programma in de taal PHP. Het is geschreven door een eerstejaars AO student.

We raden je aan deze code ook daadwerkelijk uit te voeren. Dat geeft het meeste inzicht. Schrijf vervolgens een review over deze code en onderzoek met de hiervoor behandelde technieken de verschillende condities. Onderzoek met name de verschillende samengestelde condities op de volgende LoC (Lines of Code): 8, 12, 17-23.

```
1  $DeurOpen = false;  
2  $Tijd     = 9;  
3  $Pauze   = 10;  
4  $HetIsPauze = true;  
5  $NaarDeWC = true;  
6  $KoffieHalen = FALSE;  
7  
8  if (!$DeurOpen){  
9      echo("Ik_doe_de_deur_open.");  
10     $DeurOpen = true;  
11 }  
12 if ($Tijd < $Pauze){  
13     echo("Ik_moet_nog_" . ($Pauze - $Tijd) . "_uur_wachten.");  
14     $Tijd = $Pauze;  
15 }  
16  
17 if ( $DeurOpen  
18 &&  
19     ($Tijd >= $Pauze)  
20 &&  
21     ($HetIsPauze || $NaarDeWC || $KoffieHalen))  
22 {  
23     echo("Ik_loop_door_de_deur_naar_buiten.");  
24 }  
25 else{  
26     echo("De_deur_is_dicht.");  
27 }
```

## 5.5 toepassing bij zoekopdrachten: toepassing De Morgan in SQL

### 5.5.1 Zoeken in teksten

Een echte uitdaging voor een programmeur is aan te tonen met behulp van enkele tabellen in PHPmyAdmin bijvoorbeeld, dat het volgende inderdaad correct is c.q. onjuist is:

De Morgan's laws commonly apply to text searching using Boolean operators AND, OR, and NOT. Consider a set of documents containing the words "cars" and "trucks". De Morgan's laws hold that these two searches will return the same set of documents: Search A: NOT (cars OR trucks) Search B: (NOT cars) AND (NOT trucks)

The corpus of documents containing "cars" or "trucks" can be represented by four documents:

Document 1: Contains only the word "cars". Document 2: Contains only "trucks". Document 3: Contains both "cars" and "trucks". Document 4: Contains neither "cars" nor "trucks". To evaluate Search A, clearly the search "(cars OR trucks)" will hit on Documents 1, 2, and 3. So the negation of that search (which is Search A) will hit everything else, which is Document 4. Evaluating Search B, the search "(NOT cars)" will hit on documents that do not contain "cars", which is Documents 2 and 4. Similarly the search "(NOT trucks)" will hit on Documents 1 and 4. Applying the AND operator to these two searches (which is Search B) will hit on the documents that are common to these two searches, which is Document 4. A similar evaluation can be applied to show that the following two searches will return the same set of documents (Documents 1, 2, 4): Search C: NOT (cars AND trucks), Search D: (NOT cars) OR (NOT trucks).

Bovenstaande staat beschreven op wikipedia onder het kopje 'tekst searching'. Creeer enkele tabellen met een paar eenvoudige records (5 a 6 per tabel) (N.B. waar hierboven staat 'document' lees dat als tabel).

### 5.5.2 Zoeken in database

De wetten van De Morgan zijn bijvoorbeeld ook bruikbaar bij het zoeken van records in een sql-database. Je kunt dan een opvraging (query) op verschillende manieren formuleren terwijl ze dezelfde resultaten (lees: antwoord-records) opleveren. M.a.w. je kan dan (metaforisch gesproken) langs verschillende wegen van Groningen naar Maastricht reizen.

In de volgende opdracht word je gevraagd om twee verschillende SELECT-statements te maken die elk één van de zijden van de eerste stelling van De Morgan gebruiken.

**Opdracht:** Creëer een database met de naam db\_demorgan en run daarin onderstaande scripts:

```
1 CREATE TABLE 'auto' (  
2 'auto_id' int(11) NOT NULL,  
3 'merk' varchar(25) NOT NULL,  
4 'type' varchar(20) NOT NULL,  
5 'active' varchar(1) NOT NULL  
6 ) ENGINE=InnoDB DEFAULT CHARSET=latin1 COMMENT='de_morgan';  
7  
8  
9 ALTER TABLE 'auto'  
10 ADD PRIMARY KEY ('auto_id');  
11  
12 INSERT INTO 'auto' ('auto_id', 'merk', 'type', 'active') VALUES  
13 (1, 'Mercedes', 'truck', 'Y'),  
14 (2, 'Scania', 'truck', 'Y'),  
15 (3, 'VW', 'car', 'Y'),  
16 (4, 'Opel', 'car', 'Y'),  
17 (5, 'Honda', 'motor', 'Y'),  
18 (6, 'Vivaro', 'scooter', 'Y');  
19  
20 ALTER TABLE 'auto'  
21 MODIFY 'auto_id' int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=7;
```

Stel vervolgens twee query's op (een m.b.v. A en een m.b.v. B). Hierbij toon je aan dat met beide query's het resultaat gelijk is.

Je stelt dus twee verschillende SELECT-statements op en deze lever je per mail in bij de docent.

## 5.6 Waarheidstabellen met meer dan 2 variabelen

## 5.7 opdrachten met meer variabelen

Indien er meer dan 2 logische condities zijn, dan stijgt het aantal mogelijkheden snel. Dit komt tot uiting in het aantal regels in de relevante waarheidstabel. Stel: we duiden het aantal condities (dus A, B, C, etc) aan met de letter n. Bij  $n=2$  zijn er 4 combinaties, namelijk 2 tot de macht n. Zijn er 3 logische condities geformuleerd, dan zijn er al  $2^3 = 8$  combinaties denkbaar. Bij 4 zijn er  $2^4$  regels, dus 16 etc. Het wordt dan belangrijk dat je overzicht houdt door steeds dezelfde opzet te kiezen. Dat hebben wij bij de voorgaande voorbeelden ook gedaan. We volgen dan steeds dezelfde strategie: Bijvoorbeeld de AND functie met A, B en C geeft dan:

A	B	C	A & ( B & C )
T	T	T	T
T	T	F	F
T	F	T	F
T	F	F	F
F	T	T	F
F	T	F	F
F	F	T	F
F	F	F	F

Je kan deze tabel het beste per kolom opzetten. In de eerste kolom komen eerst 4x TRUE, dan 4x FALSE. We weten uit  $2^3$  dat er 8 regels mogelijk zijn bij drie operatoren. In de tweede kolom nemen we de helft van de 1e kolom aantal T's en in de 3e kolom daarvan weer de helft. Misschien heb jezelf een andere invulwijze gekozen. Die kan ook goed zijn maar doe het wel steeds op dezelfde wijze want dan kunnen de verschillen tussen de diverse waarheidstabellen gemakkelijk door onderlinge vergelijking worden vastgesteld.

P	Q	R	P & ( Q ∨ R )
T	T	T	T
T	T	F	T
T	F	T	T
T	F	F	F
F	T	T	F
F	T	F	F
F	F	T	F
F	F	F	F

Nu is het wel nodig dat we de prioriteitsregels goed kennen en toepassen! Vandaar dat we expliciet de ronde haakjes gebruiken zodat van binnenuit naar buiten kan worden gewerkt.

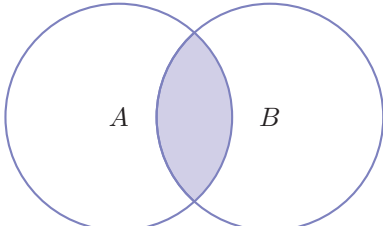
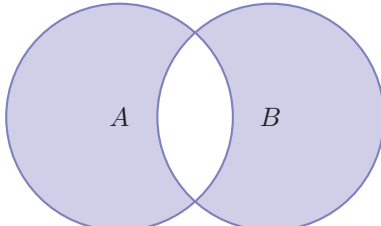
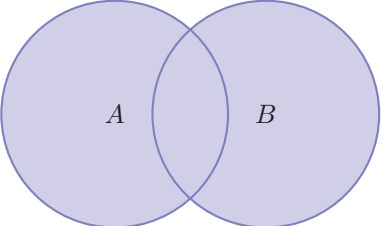
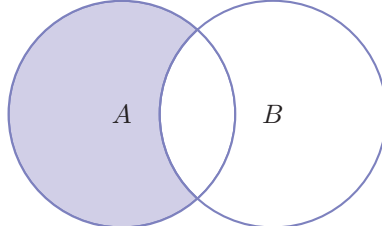
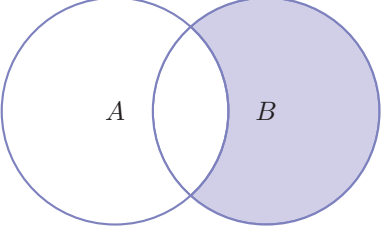
Onderzoek d.w.z. stel de waarheidstabellen op in de volgende gevallen (het teken  $\oplus$  staat voor de exclusieve OR: ook wel aangeduid met XOR):

1.  $A \wedge (B \vee C)$
2.  $A \wedge (B \oplus C)$
3.  $A \vee B \vee C$
4.  $A \oplus (B \wedge C)$

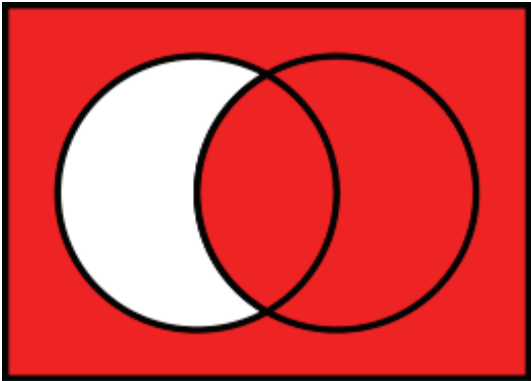
P	Q	R	P	∨	(	Q	∨	R	)
T	T	T	T	T	T	T	T	T	
T	T	F	T	T	T	T	F		
T	F	T	T	F	T	T			
T	F	F	T	F	F	F			
F	T	T	F	T	T	T			
F	T	F	F	T	T	F			
F	F	T	F	F	T	T			
F	F	F	F	F	F	F			

5.8 verzamelingenleer: venndiagrammen

Tabel 1: Venn diagrammen

regel	Naam	dag	uur
1	<div> <math>A \cap B</math>  </div> Set A and B	<div> <math>\overline{A \cap B}</math>  </div> Set A xor B	M
2	<div> <math>A \cup B</math>  </div> Set A or B	<div> <math>A - B</math>  </div> Set A but not B	M
3	<div> <math>B - A</math>  </div> Set B but not A		F

wikipedia



Figuur 6: De logische implicatie