

# **Logica en formele systemen**

---

**Prof. dr. Olga De Troyer**

Dit materiaal is bestemd voor het opleidingsonderdeel Logica en formele systemen en is een aanvulling op het leerboek “Logica voor Informatica, 3<sup>e</sup> editie, van Benthem et al., Pearson Education, ISBN 90-430-0722-6” (niet meer in de handel maar online beschikbaar: <http://resources illc.uva.nl/lvi/>).

Inhoud:

1. Het Grieks alfabet
2. Transparanten gebruikt tijdens de hoorcolleges
3. Studietekst Lambda Calculus
4. Oefeningen
5. Formularium

Voor een grondige beschrijving van de onderdelen propositielogica en predicaatlogica wordt verwezen naar het leerboek.

## Het Grieks alfabet

alfa	$\alpha$	
beta	$\beta$	
gamma	$\gamma$	$\Gamma$
delta	$\delta$	$\Delta$
epsilon	$\varepsilon$	
zeta	$\zeta$	
eta	$\eta$	
theta	$\theta$	$\Theta$
iota	$\iota$	
kappa	$\kappa$	
lambda	$\lambda$	$\Lambda$
mu	$\mu$	
nu	$\nu$	
xi	$\xi$	$\Xi$
omicron	$\circ$	
pi	$\pi$	$\Pi$
rho	$\rho$	
sigma	$\sigma$	$\Sigma$
tau	$\tau$	
upsilon	$\upsilon$	$\Upsilon$
phi	$\varphi$	$\Phi$
chi	$\chi$	
psi	$\psi$	$\Psi$
omega	$\omega$	$\Omega$



Vrije Universiteit Brussel

# Logica en formele systemen – Praktische Info

Prof. dr. Olga De Troyer

Grondslagen I



Vrije Universiteit Brussel

# Logica en formele systemen

## – Docent:

- Prof. dr. Olga De Troyer
- Vakgroep Computerwetenschappen
  - Onderzoeksgroep WISE, Pleinlaan 9 – 3<sup>de</sup> verdieping kamer 3.66
- Email: [Olga.DeTroyer@vub.be](mailto:Olga.DeTroyer@vub.be)

– Assistent: Kushal Soni [kushal.soni@vub.be](mailto:kushal.soni@vub.be)

– Communicatie en info: Leerplatform



## Studiemateriaal

- J.F.A.K. van Benthem, H.P. van Ditmarsch, J. Ketting, J.S. Lodder en W.P.M. Meyer-Viol,  
*Logica voor Informatica, 3e editie*  
Pearson Education Benelux (Addison-Wesley Nederland),  
ISBN 90-430-0722-6.  
**Niet meer in de handel beschikbaar – Online beschikbaar:**  
<http://resources.illc.uva.nl/lvi/>
  
- Dictaat: kopieën lesmateriaal, oefeningen-opgaven, Lambda Calculus  
Via de Standard Studenten shop (of te downloaden via het leerplatform)
- Het leerplatform voor dagdagelijkse info (oefeningen, uitwerkingen van oefeningen, deadlines, ...)



## Tentamen, taken en examen

- **Tentamen** in november (verplicht voor nieuwe generatiestudenten)
  - Vrijstelling voor tentamenstof vanaf 14/20 (vrijblijvend)
- **1 taak** gedurende het jaar.
  - Telt mee voor 5% in het examen van 1ste zit
- **Examen**
  - Schriftelijk
  - Gesloten boek
  - Theorie + oefeningen uit de 3 delen van de cursus
  - Formularium ter beschikking tijdens het examen



## Serious Game: TrueBiters

- Game om de waarheidstabellen van propositiologica in te oefenen

– Info met links naar laatste versie op het leerplatform





Vrije Universiteit Brussel

# Logica en formele systemen

Prof. dr. Olga De Troyer



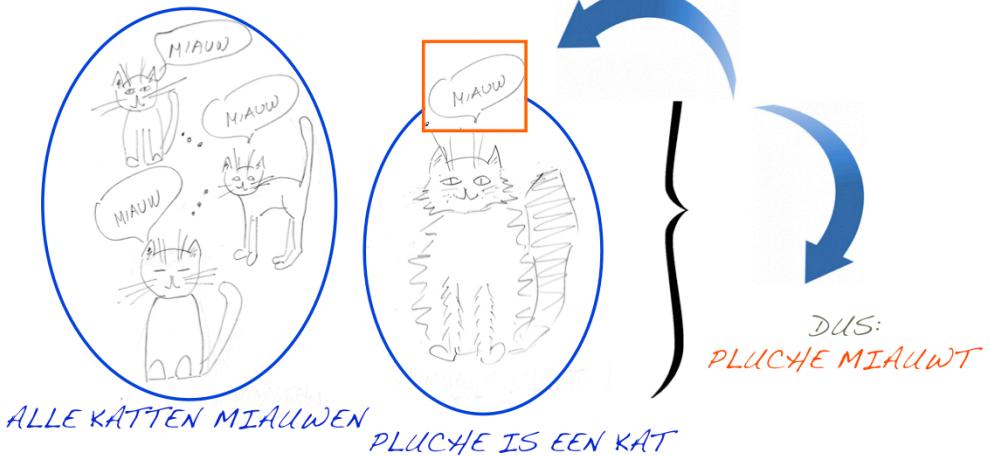
## Doel van de cursus

- Het begrijpen en verwerken van de **basisbegrippen** van de **mathematische logica**.
- Het leren om **correct** te **redeneren** en te **bewijzen**.
- **Toepassingen** van logica in de informatica.



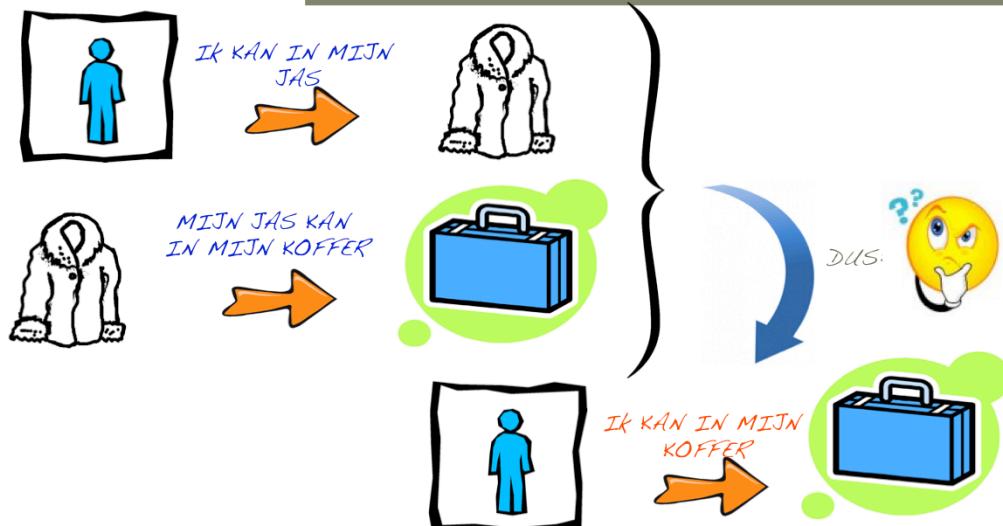
## Logica: wat? - Een voorbeeld

Menselijk redeneren gebeurt via *gevolgtrekkingen*





## Maar!



Logica & formele systemen -  
Inleiding

“kan in” heeft meedere betekenissen. Er worden hier twee verschillende betekenissen gemengd, nl “mijn jas past mij” en “ik kan mijn jas in mijn koffer steken”. Bij deze correcte formulering zijn we minder geneigd om de redeneerfout te maken.

Natuurlijke taal is vaak onnauwkeurig en dubbelzinnig en geeft op die manier vaak aanleiding tot foute redeneringen



## En ook!

### Mensen maken vaak fouten

A

3

2

B

Kaarten met aan de ene zijde een letter en aan de andere zijde een cijfer

Gegeven de bovenstaande 4 kaarten. Keer een zo klein mogelijk aantal kaarten om de volgende uitspraak te verifiëren of te falsificiëren:

*Als op de ene kant van een kaart een 'A' staat,  
dan staat op de andere kant een '2'.*

Experiment van Peter Wason (1972)  
slechts 10% gaf correct antwoord.

Toen Wason dit experiment uitvoerde gaf slechts 10% het juiste antwoord, nl dat het volstaat om “A” en de “3” om te draaien. Meestal zegt men dat men “A” en “2” moet omdraaien. Velen vinden het niet relevant om “3” om te draaien.

Nochtans, als deze kaart een “A” heeft op de achterkant dan is de uitspraak niet waar. Bovendien is het onbelangrijk om te weten wat er op de achterkant van “2” staat.



## En ook - vervolg!

Zelfde experiment maar met bestemmingen en vervoermiddelen

Manchester



London

Keer een zo klein mogelijk aantal kaarten om de volgende uitspraak te verifiëren of te falsificiëren:

*Als ik naar Manchester ga, neem ik de trein*

30% gaf nu correct antwoord.

Beide vraagstelling zijn dezelfde: Keer een zo klein mogelijk aantal kaarten om na te gaan of de uitspraak waar is. De vorm van de uitspraak is ook dezelfde: Als ... dan .... Toch beantwoorden meer mensen de tweede vraag correct.

Blijkbaar heeft de inhoud een effect op de correctheid van ons redenen. Terwijl we redenen stellen we ons de verschillende mogelijkheden voor die compatibel zijn met de premissen (= de veronderstellingen in de afleidingsregel). De moeilijkheid is dat we niet goed in staat zouden zijn om ons het onware voor te stellen; we hebben de neiging om ons enkel de ware gevallen voor te stellen.



## Dus

- ❖ Mensen maken vaak redeneerfouten
- ❖ Natuurlijke taal is vaak onnauwkeurig

Daarom:

- Wiskundig systeem: met als basis de patronen van het menselijk redenen
  - Maar: vereenvoudigingen nodig.



- Formeel systeem, bestaande uit Bijv. 
  - Formele taal: symbolische taal met een welbepaalde opbouw
  - +
    - Bewijssystemen: basisprincipes + combinatieregels laten complexe gevolgtrekkingen toe
      - Een redenering is *geldig* als ze te bewijzen valt via het bewijssysteem

≠	0	≡	⊕	‡	↔	III	0
2	3	0	111	111	111	0	1111
X	E	≡	S	r	↔	N	L
K	1	□	‡	目	↓	<	{
≤	§	×	⊕	‡	I	R	

Het herkennen van correcte en incorrecte redeneringen is belangrijke doelstelling van de logica.



# Logica: Geschiedenis

Doel: inzicht in het menselijk redeneren

Ontstaan moderne logica

Jaren 1960

19de eeuw:  
Boole, Frege,  
Hilbert



Ontstaan van computationele logica

11de eeuw:  
Middeleeuwen  
Scholastiek

4de eeuw vóór Chr:

Aristoteles



## Hoogtepunten:

### Reeds in de oudheid

Aristoteles, 4de eeuw v. Chr

Doel: inzicht in het menselijk redeneren

### Middeleeuwen

Scholastiek, +/- 1200

### Begin moderne tijden

Leibniz, +/- 1700

### Stroomversnelling in de 19de eeuw

George Boole, Gottlob Frege, David Hilbert

### Ontstaan van de moderne logica:

nieuwe begrippen en technieken

Stellingen over de bewijskracht en beperkingen van formele systemen

### Jaren 1960:

'Computationele' logica: logica voor informaticaprocessen  
(complexiteit van berekeningen, correctheid van programma's, ...)

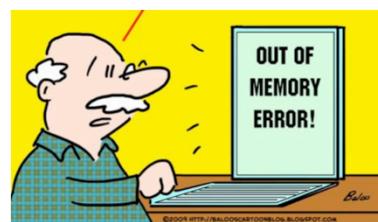


- Logica wordt gebruikt om de **correctheid** van (delen van) **programma's** te bewijzen.





- Logica kan gebruikt worden om de **complexiteit van een programma** (benodigde hoeveelheid tijd of geheugen) te bestuderen





- Logica is de basis voor **formele specificaties** van een programma
  - Sommige programmeertalen zijn zelfs gebaseerd op logica

```
APPLY.  
△ Agency  
p?: Person  
s?: Skills
```

```
p?notin dom cands  
vacS = vacS  
cands' = cands ∪ {p?mapsto s?}  
  
isCand(apply(a, p, s), p') = true if isCand(a, p) = false  
isCand(apply(a, p, s), p') = isCand(a, p') if isCand(a, p) = false,  
                                p ≠ p'  
candata(apply(a, p, s), p) = s if isCand(a, p) = false  
candata(apply(a, p, s), p') = candata(a, p') if isCand(a, p) = false,  
                                isCand(a, p') = true  
isVac(apply(a, p, s), n) = isVac(a, n) if isCand(a, p) = false  
vacData(apply(a, p, s), n) = vacData(a, n) if isCand(a, p) = false
```

```
file5.pro  
2:15 Insert Indent  
/*  
father("Bill", "John").  
father("Pam", "Bill").  
*/  
  
father(person("Bill", "male"), person("John", "male")).  
father(person("Pam", "female"), person("Bill", "male")).  
  
grandFather(Person, GrandFather):-  
    father(Father, GrandFather),  
    father(Person, Father).
```



## Logica & Informatica

- Logica vormt de basis om **verschillende wijzen van redeneren** vast te leggen: heel belangrijk voor heel veel kennistechnologietoepassingen (bijv. in Artificiële Intelligentie)





- Indirecte voordelen

- Logisch inzicht is belangrijk voor het schrijven van goede programma's.



- Logica helpt in **het formuleren van de “requirements”\*** van een te bouwen systeem.

\* “requirements”: de eisen waaraan het te bouwen systeem/programma moet voldoen



## Inhoud cursus

In *Logica en formele systemen* komen

- twee (standaard) logische systemen aan bod:
  - de **propositielogica**
    - boek “Logica voor Informatica” - hoofdstuk 2-5
  - de (eerste orde) **predikaatlogica**
    - boek “Logica voor Informatica” - hoofdstuk 6-11
- en het formele systeem:
  - **Lambda calculus** (zie dictaat).



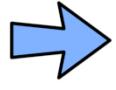
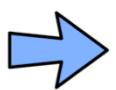
## Voorbeeld propositielogica

Meest eenvoudig logisch redeneersysteem

- Alleen ‘beweringen’ (proposities)

bv. ‘het regent’, ‘de zon schijnt’

Die **waar** of **onwaar** kunnen zijn





## Voorbeeld propositielogica

- ‘beweringen’ (proposities) kunnen verbonden worden met ‘logische connectieven’  
en, of, niet, als-dan, dan-en-slechts-dan-als  
tot samengestelde uitspraken:



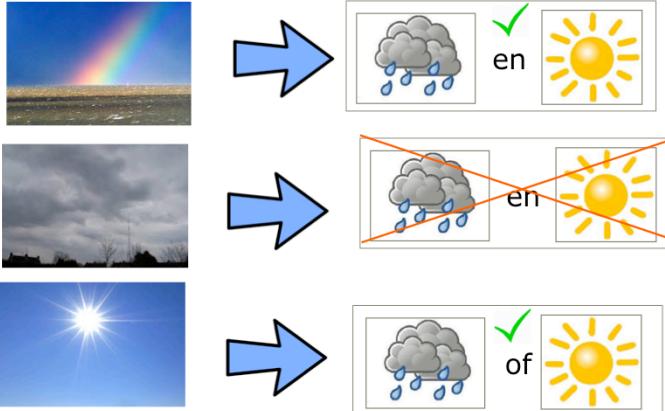
als      dan

dan en slechts dan als



## Voorbeeld propositielogica

- Deze samengestelde uitspraken kunnen op hun beurt waar of onwaar zijn





## Voorbeeld propositielogica

Hiermee gaan we dan redeneren, bv.:

aanname: als 'het regent' dan 'ik ben nat'

vaststelling: 'ik ben nat' is onwaar

af te leiden: 'het regent' is onwaar

aanname



vaststelling



Afleiding:





## Voorbeeld predikaatlogica

Predikaatlogica is uitbreiding van de propositielogica

- Laat toe om **eigenschappen van individuen** te beschrijven en **erover te redenen**.

Voorbeeld:

*ALLE KATTEN MIAUWEN*



DUS:  
*PLUCHE MIAUWT*

*PLUCHE IS EEN KAT*

Hiervoor is een rijkere taal nodig:

- ook variabelen (individuen) en kwantoren (alle, er bestaat)



## Logica: voor- en nadelen

– Voordeel/kracht:

- De formulering is **heel precies**

– Nadeel/zwakte:

- De formulering **kan heel ingewikkeld en lang worden**
- De formulering **kan cryptisch lijken**

Wanneer logica op een goede manier gebruikt wordt om iets te formuleren, is er geen discussie over de betekenis (zoals bij natuurlijke taal).



## Logica - Tips

Vergeet vooral niet om voldoende toelichting te geven bij het gebruik van logica.

Richtlijn:

- **Eerst in woorden** uitleggen waar het om gaat
- Daarna **vastleggen in logica** om de dubbelzinnigheden op te lossen
- Toevoegen van **tekst (commentaar)** om de logische formulering toe te lichten.



## Eindcompetenties voor het vak

- De studenten hebben een **basiskennis** van logica, meer bepaald **propositie- en predikaatlogica**, en zij kunnen deze kennis **aanwenden voor het formuleren en oplossen van problemen**.
- De studenten hebben een voldoende basis in logica opdat zij **andere soorten logica zouden kunnen leren**.
- De studenten hebben een **beeld** van het domein van de logica en zijn bewust van het bestaan **van diverse soorten logica en hun mogelijkheden**.
- De studenten begrijpen het **verband tussen formele systemen**, zoals logica en lambda calculus, en programmeren.

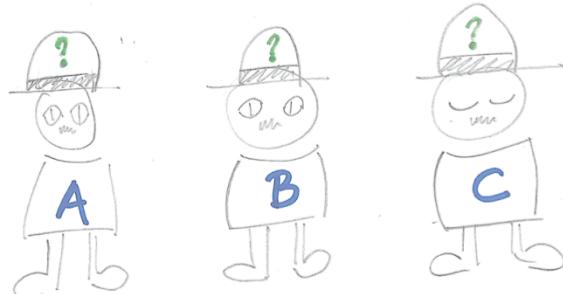


## Eindcompetenties voor het vak (2)

- **Kennis en het inzicht:**  
zie oo-fiche
- **Toepassing van de kennis en het inzicht:**  
Over de hierboven vermelde onderwerpen **eenvoudige oefeningen kunnen oplossen**; eenvoudige situaties en **problemen kunnen uitdrukken in termen van de propositie- of predikaatlogica** en kunnen oplossen.
- **Oordeelvorming:**  
Keuze kunnen maken over het te gebruiken formalisme voor eenvoudige oefeningen en problemen.
- **Communicatie:**  
Zich **duidelijk schriftelijk kunnen uitdrukken** over de hierboven opgesomde onderwerpen.
- **Leervaardigheden:**  
Vertrekend van de opgedane basiskennis, **andere logische systemen kunnen leren.**



# Een raadsel



blin WIT OF ZWART?  
GEEN 3 WITTE HOEDEN!

Drie logici A, B en C dragen hoeden waarvan zij niet weten of ze wit of zwart zijn, maar wel dat ze niet allemaal wit zijn.

A kan de hoeden van B en C zien, B kan de hoeden van A en C zien en C is blind.

Aan ieder wordt om beurt gevraagd of zij weten welke kleur de hoed op hun eigen hoofd heeft. De antwoorden zijn: A: "Nee", B: "Nee" en C: "Ja".

Wat is de kleur van de hoed op C's hoofd en hoe weet hij dat?



## Meer raadsels

- <http://logica.ugent.be/dirk/logica/puzzels.html>
- <http://www.folj.com/folj.com/puzzles/>
- <http://brainden.com/logic-puzzles.htm>



Vrije Universiteit Brussel

## EINDE INLEIDING



Vrije Universiteit Brussel

# Wiskundige bewijstechnieken

Prof. Dr. Olga De Troyer



## Stellingen van het type $p \rightarrow q$

- **Rechtstreeks**

- We trachten te bewijzen dat als  $p$  waar is dan ook  $q$  waar is
  - Hiervoor schakelen we het enige geval waarbij de implicatie onwaar is ( $p$  waar en  $q$  onwaar) uit

Bijv. door

$$p \rightarrow p_1$$

$$p_1 \rightarrow p_2$$

...

$$p_n \rightarrow q$$

Daaruit volgt  $p \rightarrow q$



## Stellingen van het type $p \rightarrow q$

- Onrechtstreeks
  - Bewijs door contrapositie



## Bewijs door contrapositie

$p \rightarrow q$  is logisch equivalent met  $\neg q \rightarrow \neg p$

Dus i.p.v.  $p \rightarrow q$  te bewijzen kunnen we

$\neg q \rightarrow \neg p$  bewijzen

Bijvoorbeeld: a en b positieve reële getallen

Als  $a^2 < b^2$  dan  $a < b$

Te bewijzen door contrapositie, nl

als  $\neg(a < b)$  dan  $\neg(a^2 < b^2)$

of nog TB: als  $a \geq b$  dan  $a^2 \geq b^2$

Bewijs: als  $a \geq b$  dan  $a.a \geq a.b$  en  $a.b \geq b.b$

Dus  $a.a \geq b.b$  of nog  $a^2 \geq b^2$



## Bewijs uit het ongerijmde

Ook genoemd: door **contradictie**  
Voor eender welk type van stelling!

Stel p is te bewijzen

We gaan nu  $\neg p$  aannemen en dan **hieruit** proberen een **eigenschap q af te leiden die in strijd is met axioma's of met gekende eigenschappen.**

Omdat  $\neg p \rightarrow q$  dan waar is  
En  $\neg p \rightarrow q$  onwaar  
Volgt hieruit dat  $\neg p$  onwaar moet zijn, of dus p waar.



## Bewijs uit het ongerijmde toegepast op stellingen van het type $p \rightarrow q$

We veronderstellen nu  $\neg(p \rightarrow q)$  waar

Dit is logisch equivalent met  $(p \wedge \neg q)$

En hieruit proberen we dan een onware bewering t af te leiden

Dus  $(p \wedge \neg q) \rightarrow t$  waar

En  $t$  onwaar

Hieruit volgt dat  $(p \wedge \neg q)$  onwaar moet zijn, en dus

$\neg(p \rightarrow q)$  onwaar, en dus  $(p \rightarrow q)$  waar.



## Stellingen van het type $p \leftrightarrow q$

Aangezien

$(p \leftrightarrow q)$  logisch equivalent is met  $(p \rightarrow q) \wedge (q \rightarrow p)$

kunnen we het bewijs van de equivalentie geven  
door elke implicatie afzonderlijk te bewijzen

Een veel voorkomend geval is:  $(p \rightarrow q)$  rechtstreeks  
bewijzen en  $(q \rightarrow p)$  via contrapositie, dus

$\neg p \rightarrow \neg q$  bewijzen



## Bewijs per inductie

- Te gebruiken wanneer we iets moeten bewijzen voor een oneindig aantal

Gebaseerd op de volgende **stelling**:

Zij  $\{p_n \mid n \in \mathbb{N}\}$  een verzameling uitspraken zodat

(a)  $p_0$  waar is

(b)  $\forall k \in \mathbb{N}: \text{als } p_k \text{ waar is, dan is } p_{k+1} \text{ waar}$

Dan is  $p_n$  waar voor alle  $n \in \mathbb{N}$



## Principe bewijs per inductie

Bewijs is gebaseerd op de volgende stelling

### Stelling

Zij  $S$  een deelverzameling van  $\mathbb{N}$  zodat

- (i)  $0 \in S$
- (ii)  $\forall k \in \mathbb{N}: \text{als } k \in S \text{ dan } k + 1 \in S$

Dan is  $S = \mathbb{N}$

(Bewijs van deze stelling uit het ongerijmde)



# Principe bewijs per inductie

## Stelling

Zij  $\{p_n \mid n \in \mathbb{N}\}$  een verzameling uitspraken zodat

- (a)  $p_0$  waar is
- (b)  $\forall k \in \mathbb{N}: p_k$  waar is, dan is  $p_{k+1}$  waar

Dan is  $p_n$  waar voor alle  $n \in \mathbb{N}$

## Bewijs

Veronderstel dat  $p_0, p_1, p_2, \dots$  voldoen aan (a) en (b)

Zij  $S = \{n \in \mathbb{N} \mid p_n \text{ is waar}\}$

Uit (a) volgt dat  $0 \in S$  en uit (b) volgt dat

$\forall k \in \mathbb{N}: k \in S \text{ dan } k + 1 \in S$

Dus is  $S = \mathbb{N}$  (vorige stelling), en d.w.z.:  $p_n$  waar voor alle  $n \in \mathbb{N}$



# Bewijs per inductie

## Voorbeeld Stelling

$$1 + 3 + 5 + \dots + (2n+1) = (n + 1)^2, \forall n \in \mathbb{N}$$

### Bewijs

Zij  $p_n$  de bewering  $1 + 3 + 5 + \dots + (2n+1) = (n + 1)^2$

Voldoet die aan (a) en (b) uit de vorige stelling?

(a)  $p_0$  is waar, nl.  $1 = 1^2$

(b) Zij  $k \in \mathbb{N}$  willekeurig. Onderstel dat  $p_k$  waar is, dus

$$1 + 3 + 5 + \dots + (2k+1) = (k + 1)^2$$

dan is  $1 + 3 + 5 + \dots + (2k+1) + (2(k+1)+1) =$

$$(k + 1)^2 + 2(k + 1) + 1 = ((k + 1) + 1)^2$$

Zo is bewezen dat  $\forall k \in \mathbb{N}$ : als  $p_k$  dan  $p_{k+1}$

Dus dan is  $p_n$  waar voor alle  $n \in \mathbb{N}$  (vorige stelling)



Vrije Universiteit Brussel

# Logica en formele systemen

## Deel I: Propositielogica

Prof. dr. Olga De Troyer



## Inhoud

- Syntaxis en semantiek
- Geldig gevolg
- Afleidingen
- Metatheorie



Vrije Universiteit Brussel

## Propositielogica: syntaxis en semantiek



# Syntaxis en semantiek

## – Inhoud

- syntaxis van een taal
- semantiek van een taal
- syntaxis van de propositielogica:
  - Alfabet, formule, in
  - cieve definitie, inductieprincipe, formuleschema, substitueren, constructieboom
- semantiek van de propositielogica:
  - Waarheidswaarden, waarheidstabbel, waardering, model van een formule(verzameling), modeleliminatie, tautologie, logisch equivalent, functioneel volledig, disjunctieve normaalvorm



## Voorbeeld

- **Correct redenering**

Aanname: De afstandsbediening is kapot of de TV werkt niet goed

Aanname: Maar de TV werkt wel goed

Conclusie: Dus is de afstandsbediening kapot

- **Foute redenering**

Aanname: Het schilderij hangt hier niet als het gestolen is

Aanname: Het schilderij hangt hier niet

Conclusie: Dus is het schilderij gestolen

Nogthans lijken beide sterk op elkaar

We laten ons vaak beïnvloeden door de tekst!



# Daarom

- Beweringen, **proposities** genoemd, zijn **atomair**
  - We willen ze niet verder analyseren
  - Daarom gaan we ze **voorstellen door symbolen**, nl. letters
  - Voorbeelden:
    - Ik ben ziek: **z**
    - Ik lust koffie: **k**
    - Mijn fiets is gestolen: **f**
    - Brussel is de hoofdstad van België: **b**
    - $5 < 2$ : **v**



# Inleiding

- Een propositie is een **uitspraak die, gegeven een situatie, waar of onwaar kan zijn**
  - Bv. 'Het regent' is in een gegeven situatie **of waar of onwaar**, maar **nooit zowel onwaar als waar**





## Inleiding

- Proposities kunnen verbonden worden met de logische connectieven **en**, **of**, **niet**, **als-dan**, **dan-en-slechts-dan-als** tot **formules**



als           dan     

     dan en slechts dan als



# Inleiding

- We gebruiken ook symbolen voor de logische connectieven:
  - niet :  $\neg$
  - en:  $\wedge$
  - of:  $\vee$
  - als-dan:  $\rightarrow$
  - dan-en-slechts-dan-als:  $\leftrightarrow$



# Inleiding

- Zinnen in natuurlijke taal kunnen zo vertaald worden naar formele specificaties

## Voorbeelden:

Aanname: **Als je ziek bent, dan lust je geen koffie**

Aanname: **Je lust koffie**

Conclusie: **Je bent niet ziek**

Aanname:  $(p \rightarrow \neg q)$

Aanname:  $q$

Conclusie:  $\neg p$

Aanname: **Als je fiets gestolen is, dan lust je geen koffie**

Aanname: **Je lust koffie**

Conclusie: **Je fiets is niet gestolen**

Aanname:  $(r \rightarrow \neg q)$

Aanname:  $q$

Conclusie:  $\neg r$



# Componenten van een formeel taal

- Bij een (formeel) taal zijn 3 aspecten belangrijk:

- Het **alfabet**

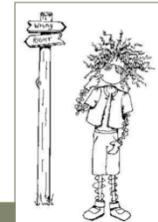
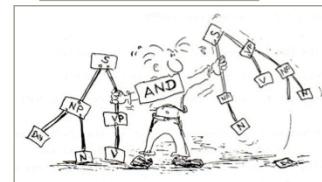
- Welke **symbolen** men mag gebruiken

- De **syntaxis** (grammatica)

- Geheel van **regels** die aangeeft op welke manier uitdrukkingen in de taal gevormd mogen worden

- De **semantiek**

- De **betekenis** van syntactisch correcte uitdrukkingen in een taal.





## Propositielogica:

- **propositie**: bewering of uitspraak, uitgedrukt in een zin
  - Vben: “Brussel is de hoofdstad van België”, “4 < 7”, “8 < 3”

Notatie: kleine letters.

Keuze: bijv.  $h$  voor “Jan huilt” (geheugensteuntje) of  $p, q, r, \dots$ , of  $p_1, p_2, p_3, \dots$  (abstracte notatie).

In een formele taal gebruiken we **propositieletters** om proposities voor te stellen omdat we die niet nader willen analyseren.



# Alfabet



## *Definitie*

Het **alfabet** van de propositielogica bestaat uit

- een verzameling **propositieletters**
- de logische symbolen:  $\neg$ ,  $\wedge$ ,  $\vee$ ,  $\rightarrow$  en  $\leftrightarrow$
- de hulpsymbolen: ) en (



- De symbolen uit het alfabet zijn te combineren via regels tot uitdrukkingen, **formules** genoemd.

### Definitie

De **formules** in de propositielogica zijn als volgt gedefinieerd:

1. elke propositieletter is een formule
2. als  $\varphi$  en  $\psi$  formules zijn dan zijn  $\neg\varphi$ ,  $(\varphi \wedge \psi)$ ,  $(\varphi \vee \psi)$ ,  $(\varphi \rightarrow \psi)$  en  $(\varphi \leftrightarrow \psi)$  ook formules
3. niets anders is een formule

- Om **abstracte (willekeurige) formules** weer te geven worden **kleine Griekse letters** gebruikt ( $\varphi$ ,  $\psi$ , ...), dit noemt ment **formulevariabelen**

Let op de haakjes!



# Syntax – Terminologie



## – Terminologie:

- Propositieletters heten ook **atomaire formules** of **atomen**.
- De samengestelde formules hebben een vaste **uitspraak** en **naam**:

**vorm**

$\neg \varphi$   
 $(\varphi \wedge \psi)$   
 $(\varphi \vee \psi)$   
 $(\varphi \rightarrow \psi)$   
 $(\varphi \leftrightarrow \psi)$

**uitspraak**

niet phi  
phi **en** psi  
phi **of** psi  
**als** phi **dan** psi  
phi **dan en slechts als** psi

**naam**

**negatie**  
**conjunctie**  
**disjunctie**  
**implicatie**  
**equivalentie**

Korte notatie voor **dan en slechts dan als**: **desda**

$(\varphi \rightarrow \psi)$ :  $\varphi$  wordt ook wel **het antecedent** genoemd en  $\psi$  **het consequent**

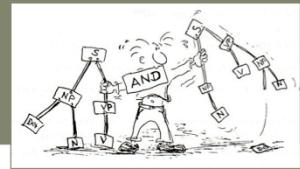


## Opgelet

- ‘en’ uit onze natuurlijk taal en logische ‘en’ ( $\wedge$ ) zijn niet 100% equivalent
  - “ze kwam binnen en deed het licht uit”
  - “ze deed het licht uit en kwam binnen”
  - In natuurlijke taal verschillend; in propositielogica gelijkwaardig
- Idem voor ‘of’
  - “Voor je verjaardag krijg je een racefiets of een computer”
  - In natuurlijke taal is bedoeling “óf ... óf ...”; in propositielogica kunnen beide waar zijn.



# Syntaxis – Voorbeelden



## – Voorbeelden

Welke zijn geldige formules?

p

$\neg\neg p$

$(\neg p)$

$\neg(p \rightarrow \neg q)$

$q\neg$

$((p \wedge q) \rightarrow r) \rightarrow (\neg p \wedge q))$

$(p \wedge q) \wedge r)$

$(p \vee q \vee r)$

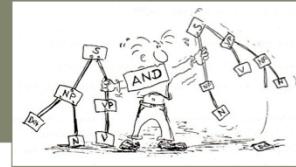


## Inductieve definitie

- De manier waarop de formules zijn gedefinieerd heet **inductief**.
- Een **inductieve definitie** bestaat uit
  1. Eén of meerdere **basisstappen** waarin bepaalde dingen meteen tot objecten van de gewenste soort worden verklaard
    - Bijv. elke propositiële letter is een formule
  2. Eén of meerdere **opbouwstappen** die de constructieprincipes geven om objecten te maken
    - Bijv. als  $\varphi$  en  $\psi$  formules zijn dan zijn  $\neg\varphi$ ,  $(\varphi \wedge \psi)$ ,  $(\varphi \vee \psi)$ ,  $(\varphi \rightarrow \psi)$  en  $(\varphi \leftrightarrow \psi)$  ook formules
  3. Een **afsluitende stap** die bepaalt dat alles wat niet in eindig veel stappen met behulp van 1 en 2 gevormd kan worden, geen toegestaan object is
    - Bijv. niets anders is een formule



# Syntaxis – Formuleschema

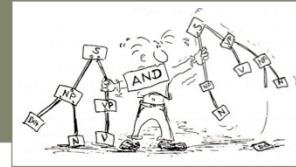


## – Formuleschema's en instanties

- een vorm zoals  $(\varphi \leftrightarrow \neg \psi)$  is **een abstracte vorm** van een formule: een **formuleschema** genoemd.
- Een **concrete formule** ontstaat als voor  $\varphi$  en  $\psi$  **concrete formules** worden ingevuld. Dit heet **een instantie van het formuleschema**.
- Instanties van  $(\varphi \leftrightarrow \neg \psi)$  zijn:  
 $(p \leftrightarrow \neg q)$   
 $(q \leftrightarrow \neg p)$   
 $((p \wedge q) \leftrightarrow \neg(r \leftrightarrow \neg(s \rightarrow q)))$



# Substitutie



Voorbeeld:  $((p \wedge q) \rightarrow p)$

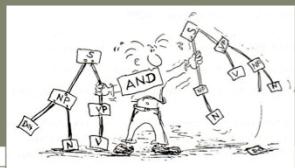
– Kan ik  $p$  vervangen door een andere letter?

$((s \wedge q) \rightarrow s)$

Laat  $\varphi$  een formule zijn waar (mogelijk) een propositieletter  $p$  in voorkomt. Door **elk voorkomen van  $p$  in  $\varphi$  te vervangen** door een formule  $\psi$ , ontstaat een nieuwe formule.

**Notatie uitspraak**

$[\psi/p]\varphi$  formule die resulteert door  $p$  in  $\varphi$  te  
**substitueren** (te vervangen) **door**  $\psi$



### Definitie

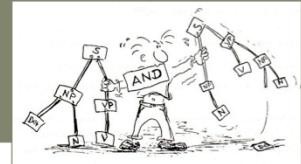
#### Substitutie

1.  $[\psi/p]\varphi = \psi$  als  $\varphi = p$   
 $[\psi/p]\varphi = \varphi$  als  $\varphi$  een propositieletter is verschillend van  $p$
2.  $[\psi/p]\neg\varphi = \neg[\psi/p]\varphi$   
 $[\psi/p](\varphi \wedge \chi) = ([\psi/p]\varphi \wedge [\psi/p]\chi)$   
 $[\psi/p](\varphi \vee \chi) = ([\psi/p]\varphi \vee [\psi/p]\chi)$   
 $[\psi/p](\varphi \rightarrow \chi) = ([\psi/p]\varphi \rightarrow [\psi/p]\chi)$   
 $[\psi/p](\varphi \leftrightarrow \chi) = ([\psi/p]\varphi \leftrightarrow [\psi/p]\chi)$

Merk op dat deze definitie gebaseerd is op de inductieve definitie van de formule



## Substitutie – voorbeelden



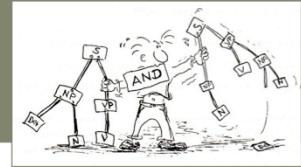
- Voorbeelden substitutie:

$$[(p \rightarrow q)/r](r \wedge s) =$$

$$[(p \vee \neg p)/q](q \rightarrow (s \rightarrow q)) =$$

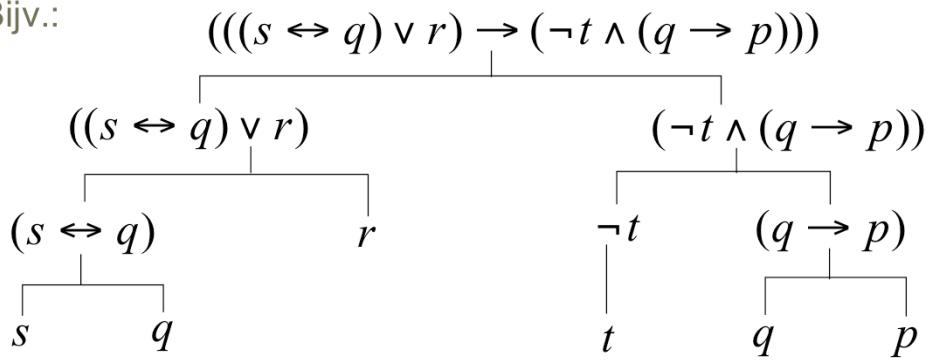


# Constructieboom



- De opbouw van een formule kan worden weergegeven door een **constructieboom**

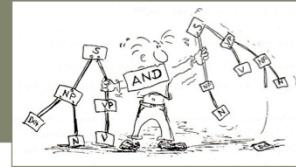
- Bijv.:



de boom wordt van onder naar boven gelezen



# Constructieboom



- Haakjes zijn belangrijk; ze geven het bereik van een connectief aan
- Ze leggen de constructie eenduidig vast
- Hoeveel constructiebomen bestaan er voor een formule?
  - In het algemeen is er **precies één constructieboom per formule**
  - Dit is **niet steeds zo in de natuurlijke taal**  
Bijv: “Als de baby niet huilt en trappelt, dan is hij gelukkig”.

2 mogelijkheden voor een gelukkige baby:

- De baby huilt niet en trappelt (dus enkel trappelen)
- De baby huilt en trappelt niet (dus noch trappelen, noch huilen)



# Semantiek



- We weten nu wat een geldige formule is maar **wat is de betekenis van een formule?**
  - Dit is **zijn waarheidswaarde**, nl. **waar** of **onwaar**
  - Hoe die bepalen?
    - Voorbeeld
      - De Nederlandse zin “het regent en de zon schijnt” kan **waar** zijn of **niet waar** zijn.  
Dit is gebaseerd op de waarheidswaarden van de onderdelen.



niet waar



waar



niet waar



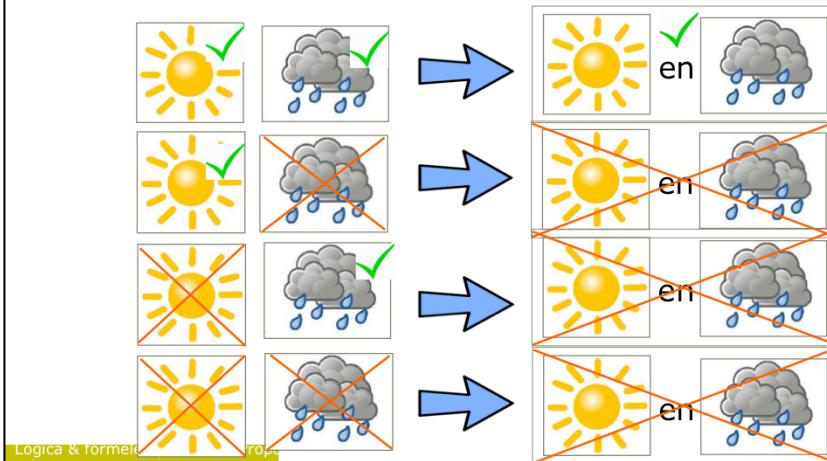
niet waar



# Semantiek



De waarheidswaarde van een formule wordt gegeven door de waarheidswaarde van de delen van de formule.



26

Zoals in natuurlijke taal is de waarheidswaarde van een formule gebaseerd op de waarheidswaarden van de onderdelen van de formule.



De begrippen ‘waar’ en ‘onwaar’ worden **waarheidswaarden** genoemd.

Vaak wordt **1** gebruikt voor **waar** en **0** voor **onwaar**

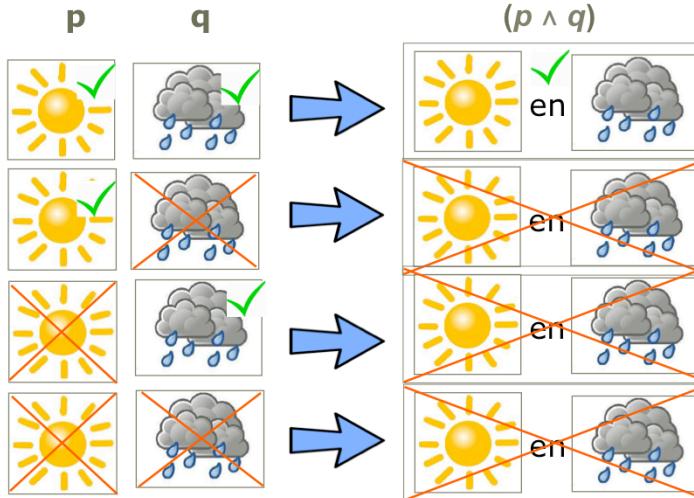
- Voor **atomen** (propositieletters) moet de **waarheidswaarde gegeven** (of verondersteld) worden
- De waarheidswaarde voor een samengestelde formule
  - is waar of onwaar
  - en volgt uit de waarheidswaarden van de samenstellende delen door middel van de **waarheidstabellen van de connectieven**.

Voor atomen (propositieletters) moet de **waarheidswaarde gegeven** (of verondersteld) worden: bijvoorbeeld er is gegeven dat p waar is, of we veronderstellen een keer dat p waar is en berekenen dan de waarheidswaarden voor de formules die p bevatten

en we veronderstellen een keer dat p onwaar en doen dan weer hetzelfde.



# Waarheidstabellen – Voorbeeld



- Waarheidstabel voor  $\wedge$  :

$p$	$q$	$(p \wedge q)$
1	1	1
1	0	0
0	1	0
0	0	0

Elke rij staat voor een bepaalde situatie,  
**waardering** genoemd  
(notatie  $V$ ).

Elke rij staat voor een bepaalde situatie, **waardering** genoemd (notatie  $V$ ). Ze zijn te beschouwen als functies van propositiële letters naar waarheidswaarden

$V(p) = 1$  wil zeggen:  
de waardering van  $p$  onder/bij de waardering  $V$  is 1.

$V((p \wedge q))$  wil zeggen:  
de waardering van  $(p \wedge q)$  onder/bij de waardering  $V$  is 1.



# Waarheidstabellen



- Er is een waarheidstabel voor elke connectief:

$\varphi$	$\neg\varphi$
1	0
0	1

$\varphi$	$\psi$	$(\varphi \wedge \psi)$
1	1	1
1	0	0
0	1	0
0	0	0

$\varphi$	$\psi$	$(\varphi \vee \psi)$
1	1	1
1	0	1
0	1	1
0	0	0

$\varphi$	$\psi$	$(\varphi \rightarrow \psi)$
1	1	1
1	0	0
0	1	1
0	0	1

$\varphi$	$\psi$	$(\varphi \leftrightarrow \psi)$
1	1	1
1	0	0
0	1	0
0	0	1



## Opmerking



- De semantiek van de implicatie ( $\varphi \rightarrow \psi$ ) vertoont overeenkomst met als-dan zinnen uit de natuurlijke taal

- Bijv. “als ik het gras afmaai dan krijg ik 5 Euro”
- Maar! “als de maan van groene kaas is dan ben ik rijk”
  - Waar of onwaar?
- Het geval waarin  $\varphi$  onwaar is komt onnatuurlijk over
  - In de natuurlijke taal gebruiken we meestal “als ...dan...” zinnen in een oorzaak-gevolg situatie  
**Als de oorzaak onwaar is** vinden we het onrealistisch om over het gevolg na te denken en vinden we de implicatie dus **intuïtief onwaar, maar in propositie logica is de implicatie dan waar!**



## Waarheidstabellen samengestelde formule



- De mogelijke waarderingen voor een willekeurige formule  $\varphi$  volgen uit de waarheidstabellen voor de connectieven. Deze worden ook weergegeven in een waarheidstable
- Waarheidstable voor een **samengestelde formule**:
  - tabel met waarheidswaarden voor alle mogelijke waarderingen (combinaties) van de voorkomende propositieletters en de deelformules.



# Voorbeeld



$((h \wedge s) \rightarrow \neg u)$ :

	$h$	$s$	$u$	$(h \wedge s)$	$\neg u$	$((h \wedge s) \rightarrow \neg u)$
$V_1$	1	1	1	1	0	0
$V_2$	1	1	0	1	1	1
$V_3$	1	0	1	0	0	1
$V_4$	1	0	0	0	1	1
$V_5$	0	1	1	0	0	1
$V_6$	0	1	0	0	1	1
$V_7$	0	0	1	0	0	1
$V_8$	0	0	0	0	1	1



# Voorbeeld



$((h \wedge s) \rightarrow \neg u)$ :

	$h$	$s$	$u$	$(h \wedge s)$	$\neg u$	$((h \wedge s) \rightarrow \neg u)$	$((h \wedge s) \rightarrow \neg u)$	$((h \wedge s) \rightarrow \neg u)$
$V_1$	1	1	1	1	0	0	1	0
$V_2$	1	1	0	1	1	1	1	1
$V_3$	1	0	1	0	0	1	0	1
$V_4$	1	0	0	0	1	1	0	1
$V_5$	0	1	1	0	0	1	0	1
$V_6$	0	1	0	0	1	1	0	1
$V_7$	0	0	1	0	0	1	0	1
$V_8$	0	0	0	0	1	1	0	1

Compactere notatie



## Complexiteit waarheidstabellen



- Wanneer er **n verschillende propositieletters** in een formule voorkomen dan heeft de waarheidstabel van de formule  **$2^n$  rijen**.

Dus bij veel propositieletters kunnen de waarheidstabellen van formules groot worden



# Semantiek - waardering

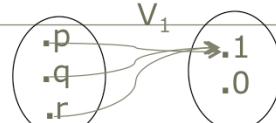


## Definitie

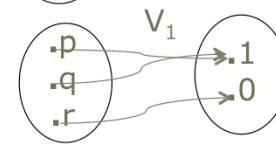
Een **waardering** is een functie van alle propositieletters naar de waarheidswaarden 'waar' (1) en 'onwaar' (0)

Voorbeelden waarderingen:

$$V_1 \quad p \rightarrow 1 ; q \rightarrow 1; r \rightarrow 1$$



$$V_2 \quad p \rightarrow 1 ; q \rightarrow 1; r \rightarrow 0$$



- Via een gegeven waardering kan men de waarheidswaarde van een willekeurige formule  $\varphi$  bepalen



## Definitie

Een waardering  $V$  heet een **model van een formule  $\varphi$**  als geldt dat  $V(\varphi) = 1$ .

- de **verzameling van alle modellen van  $\varphi$**  noteren we  
 $MOD(\varphi) = \{V \mid V(\varphi) = 1\}$

Dus, een model maakt de formule waar.



## Semantiek - model



- We kunnen ook spreken over de modellen van een verzameling formules

### Definitie

Een waardering  $V$  heet een **model van een formuleverzameling  $\Sigma$**  als  $V$  een model is van elke formule  $\varphi \in \Sigma$

De verzameling van modellen van  $\Sigma$  wordt genoteerd als  $\text{Mod}(\Sigma)$

Dus een model van een formuleverzameling maakt alle formules van de verzameling waar



## Voorbeeld

- Wat zijn de modellen van:  $\{(r \rightarrow s), \neg(r \wedge s)\}$ ?
- M.a.w. voor welke waarderingen van r en s zijn beide formules waar?

Merk op dat  $\text{Mod}(\Sigma \cup \{\varphi\}) \subseteq \text{Mod}(\Sigma)$

- Hoe meer modellen hoe minder informatieve inhoud;
- Hoe minder modellen hoe meer informatieve inhoud;
- Eén model betekent volledige informatie.



# Modeleliminatie



- Verschaft een aantal formules voldoende informatie om een vraag te kunnen beantwoorden?
  - Bijv.: Jan komt als Marie of Anne komt ( $\varphi$ ); Anne komt als Marie niet komt ( $\psi$ ); Jan komt niet als Anne komt ( $\chi$ )  
Wie komt er wel en wie niet?  
m.a.w. heeft  $\{\varphi, \psi, \chi\}$  een uniek model?
- Techniek: **Modeleliminatie**
  - gebaseerd op:  
als  $\Sigma_1 \subseteq \Sigma_2$  dan geldt  $MOD(\Sigma_2) \subseteq MOD(\Sigma_1)$
  - Bepaal eerst alle modellen van  $\varphi$ , vervolgens kijken we welke ook modellen zijn  $\psi$ , en tenslotte welke ook modellen zijn voor  $\chi$ .



## Modeleliminatie – voorbeeld



- Modeleliminatie: voorbeeld
  - Over het weer in Londen weten we het volgende:
    - Het waait of het regent.
    - Als het waait en regent, dan is het koud.
    - Als het regent, dan is het niet koud.
    - Als het niet waait, dan is het koud.
  - Welke conclusies kunnen we hieruit trekken?
    - Vertaling:
      - $w$  : het waait
      - $r$  : het regent
      - $k$  : het is koud
    - $\Sigma = \{ (w \vee r), ((w \wedge r) \rightarrow k), (r \rightarrow \neg k), (\neg w \rightarrow k) \}$



# Modeleliminatie – voorbeeld



$$\Sigma = \{ (w \vee r), ((w \wedge r) \rightarrow k), (r \rightarrow \neg k), (\neg w \rightarrow k) \}$$

	<b>w</b>	<b>r</b>	<b>k</b>	<b>(w ∨ r)</b>	<b>((w ∧ r) → k)</b>	<b>(r → ¬k)</b>	<b>(¬w → k)</b>	
$V_1$	1	1	1	1	1	0		
$V_2$	1	1	0	1	0			
→ $V_3$	1	0	1	1	1	1	1	
→ $V_4$	1	0	0	1	1	1	1	
$V_5$	0	1	1	1	1	0		
$V_6$	0	1	0	1	1	1	0	
$V_7$	0	0	1	0				
$V_8$	0	0	0	0				

Conclusie: Twee modellen blijven over:  $V_3$  en  $V_4$ .  
Voor beide geldt: het waait en het regent niet.

Modeleliminatie is ook de techniek die we toepaste voor de puzzel over de 3 logici en hun hoeden in de inleiding



# Tautologie en contradictie



## Definitie

Een formule  $\varphi$  heet een **tautologie** als elke waardering een model is van  $\varphi$ , m.a.w.  $\forall V: V(\varphi)=1$ .

- M.a.w. (intuitief) een tautologie is een formule die altijd waar is
- Voorbeelden van tautologieën:  
 $(p \rightarrow (q \rightarrow p))$ ,  
alle instanties van  $(\varphi \vee \neg\varphi)$  en  $\neg(\varphi \wedge \neg\varphi)$ .
- De tegenhanger van een tautologie heet een **contradictie**. Dus een contradictie is een formule die altijd onwaar is.



# Logisch equivalent



## Definitie

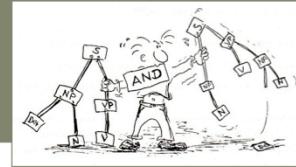
Twee formules  $\varphi$  en  $\psi$  heten **logisch equivalent**  
als de formule  $(\varphi \leftrightarrow \psi)$  een tautologie is

Vben logische equivalenties:

- $\varphi$  en  $\neg\neg\varphi$ ,
- $(\varphi \wedge (\psi \wedge \chi))$  en  $((\varphi \wedge \psi) \wedge \chi)$
- de wetten van De Morgan:
  - $(\neg(\varphi \wedge \psi) \leftrightarrow (\neg\varphi \vee \neg\psi))$
  - $(\neg(\varphi \vee \psi) \leftrightarrow (\neg\varphi \wedge \neg\psi))$
- de principes van distributiviteit:
  - $((\varphi \wedge (\psi \vee \chi)) \leftrightarrow ((\varphi \wedge \psi) \vee (\varphi \wedge \chi)))$
  - $((\varphi \vee (\psi \wedge \chi)) \leftrightarrow ((\varphi \vee \psi) \wedge (\varphi \vee \chi)))$



## Lossere notatie



- Logische equivalenties rechtvaardigen **een lossere notatie**:
  - Op grond van de logische equivalentie van  $((\varphi \wedge (\psi \wedge \chi)) \text{ en } ((\varphi \wedge \psi) \wedge \chi)$  laten we toe om haakjes weg te laten, dus  $(\varphi \wedge \psi \wedge \chi)$  i.p.v.  $((\varphi \wedge (\psi \wedge \chi))$  of  $((\varphi \wedge \psi) \wedge \chi)$
  - Dit noemt men de associativiteit van  $\wedge$
  - Ook ‘ $\vee$ ’ is associatief
  - Ook buitenste haakjes mogen weggelaten worden indien geen verwarring mogelijk
    - $\varphi \wedge \psi \wedge \chi$  i.p.v.  $(\varphi \wedge \psi \wedge \chi)$



# Functioneel volledig



## Definitie

Een verzameling van connectieven  $C$  heet **functioneel volledig** als elk formule  $\varphi$  logisch equivalent is met een formule  $\psi$  die enkel connectieven uit  $C$  bevat

- De verzameling connectieven  $\{\neg, \wedge, \vee\}$  is functioneel volledig (bewijs later).
- $\{\neg, \vee\}$  is dan ook functioneel volledig op grond van de tautologie  $((\varphi \wedge \psi) \leftrightarrow \neg(\neg\varphi \vee \neg\psi))$ . Hierdoor kunnen alle voorkomens van  $\wedge$  vervangen worden door  $\neg$  en  $\vee$ .



## Andere connectieven

- NOR connectief

- noch ... noch
- Waarheidstabel voor NOR:

$p$	$q$	$(p \text{ NOR } q)$
1	1	0
1	0	0
0	1	0
0	0	1
- {NOR} is functioneel volledig.



## Disjunctieve normaalvorm

### *Definitie*

Een formule is in **disjunctieve normaalvorm** wanneer deze de syntactische vorm heeft van een disjunctie van conjuncties, bestaande uit atomen of negaties van atomen:

$$(\varphi_1 \wedge \dots \wedge \varphi_{n1}) \vee \dots \vee (\chi_1 \wedge \dots \wedge \chi_{nk})$$

waarbij  $\varphi_1, \dots, \chi_{nk}$  atomen of negaties van atomen zijn

Algemeen geldt dat voor iedere formule  $\varphi$  een logisch equivalente formule  $\varphi^*$  bestaat die in disjunctieve normaalvorm is (zonder bewijs).



## Redeneren

Hoe kunnen we nu gaan redeneren?

Essentieel 2 manieren:

- Via de modellen: semantisch (geldig gevolg)
- Via afleidingsregels: syntactisch (natuurlijke deductie)



Vrije Universiteit Brussel

## Propositielogica: Geldig gevolg



# Geldig gevolg

## – Inhoud

- Geldig gevolg
- Sequent
- Tegenvoorbeeld van een sequent
- Semantisch tableau
- Reductieregels
- Open en gesloten tableau
- Semantisch consistent en inconsistent
- Adequaatheidsstelling



## Geldig gevolg - Voorbeeld

“Jan is een goede schaker en Karin is een goede dammer”

Hieruit kunnen we (intuitief) concluderen:

“Jan is een goede schaker”

Of nog: uit  $(p \wedge q)$  kunnen we  $p$  concluderen

p	q	$(p \wedge q)$	p
1	1	1	1
1	0	0	1
0	1	0	0
0	0	0	0

Het principe voor *geldig gevolg*:

Als het uitgangspunt waar, dan is ook de conclusie waar.



# Geldig gevolg

## Definitie

Een formule  $\psi$  heet **een geldig gevolg** van een verzameling formules  $\Sigma$  als elk model van  $\Sigma$  ook model is van  $\psi$

- Notatie:  $\Sigma \models \psi$  of ook nog  $\Sigma / \psi$ 
  - $\psi$  is een geldig gevolg van sigma
  - Als  $\Sigma = \{ \varphi_1, \dots, \varphi_n \}$  dan schrijven we  $\varphi_1, \dots, \varphi_n \models \psi$
- Voorbeelden
  - $p, p \rightarrow q \models q$
  - $p, p \rightarrow q, q \rightarrow r \models r$



# Geldig gevolg

–  $\Sigma \not\models \psi$

- D.w.z.  $\psi$  is geen geldig gevolg van  $\Sigma$
- Er bestaat dus een model van  $\Sigma$  dat geen model is van  $\psi$ . Dit model noemt men een *tegenoorbeeld* van de gevolgstrekkings  $\Sigma \models \psi$

Voorbeeld:

–  $q, p \rightarrow q \not\models p$

q waar en p onwaar dan  $(p \rightarrow q)$  waar , maar p is onwaar.



## Geldig gevolg - Opgelet

- Opgelet “geldig gevolg” en “waarheid” zijn verschillende begrippen!  
Vb:  $p \vee q, \neg q \wedge r \models p \wedge r$   
En toch maakt  $V(p) = V(q) = V(r) = 0$  alle formules onwaar
- Er is een nauw verband tussen de implicatie ( $\rightarrow$ ) en logisch gevolg ( $\models$ ) maar de twee zijn verschillende concepten!
  - $\rightarrow$  maakt deel uit van de syntaxis van de propositielogica;  $\models$  niet
  - Als  $\varphi \models \psi$ , dan is  $(\varphi \rightarrow \psi)$  waar,  
maar niet omgekeerd:  $(\varphi \rightarrow \psi)$  betekent niet noodzakelijk  $\varphi \models \psi$ 
    - Want  $(\varphi \rightarrow \psi)$  kan nl ook onwaar zijn (als  $\varphi$  waar en  $\psi$  onwaar) en dan is er geen geldig gevolg meer
  - Als  $(\varphi_1 \wedge \dots \wedge \varphi_n) \rightarrow \psi$  een tautologie is dan hebben we ook  $\varphi_1, \dots, \varphi_n \models \psi$  (en omgekeerd).



## Bekende geldige gevolgen

- Ex falso:  $\varphi, \neg\varphi \models \psi$   
Uit een contradictie volgt alles
- Contrapositie:  $\varphi \rightarrow \psi \models \neg\psi \rightarrow \neg\varphi$
- Hypothetisch syllogisme:  $\varphi \rightarrow \psi, \psi \rightarrow \chi \models \varphi \rightarrow \chi$
- Disjunctief syllogisme:  $\varphi \rightarrow \psi, \neg\varphi \rightarrow \psi \models \psi$

Special geval indien  $\Sigma = \emptyset$

$\models \psi$  betekent dat elke waardering een model is voor  $\psi$ ; m.a.w.  $\psi$  is een tautologie.



# Geldig gevolg

- Hoe weten we dat een gevolgtrekking geldig is?
  - Eerste mogelijkheid: stel waarheidstabel op

Voorbeeld:  $\neg q, p \rightarrow q / \neg p$

p	q	$\neg q$	$p \rightarrow q$	$\neg p$
1	1	0	1	0
1	0	1	0	0
0	1	0	1	1
0	0	1	1	1

→ Model voor  $\{\neg q, p \rightarrow q\}$  en voor  $\neg p$

- Maar de grootte van een waarheidstabel is exponentieel in het aantal proposities
- Het is efficiënter om naar **een tegenvoorbeeld te zoeken**
  - Door middel van een techniek die men **semantische tableaus** noemt.



# Geldig gevolg – semantisch tableau

## De nodige concepten:

- Een **sequent** is een rijtje van de vorm:  
 $\varphi_1, \dots, \varphi_n \circ \psi_1, \dots, \psi_m$   
met  $\varphi_1, \dots, \psi_m$  formules,  $n \geq 0, m \geq 0$ .
- Een waardering  $V$  heet een **tegenvoorbeeld** van een **sequent**  $\varphi_1, \dots, \varphi_n \circ \psi_1, \dots, \psi_m$  indien  
 $V(\varphi_1) = \dots = V(\varphi_n) = 1$  en  
 $V(\psi_1) = \dots = V(\psi_m) = 0$
- Merk op (belangrijk!): Een sequent  $\varphi_1, \dots, \varphi_n \circ \psi_1, \dots, \psi_m$  heeft geen tegenvoorbeeld als er een  $i$  en een  $j$  bestaan zodat  $\varphi_i = \psi_j$

Er kan geen tegenvoorbeeld zijn als er links en rechts dezelfde formule staat in een sequent.



## Geldig gevolg – semantisch tableau

- Om te bepalen of  $\varphi_1, \dots, \varphi_n \models \psi$  gaan we onderzoeken of er al dan niet een tegenvoorbeeld is voor het sequent  $\varphi_1, \dots, \varphi_n \circ \psi$ .
  - Dit doen we op een systematische manier
    - Semantisch tableau
- Een **semantisch tableau** is een schema waarin op systematische wijze het mogelijk bestaan van tegenvoorbeelden van een gegeven sequent teruggebracht wordt (*gereduceerd*) tot dat van één of meer eenvoudiger sequenten.



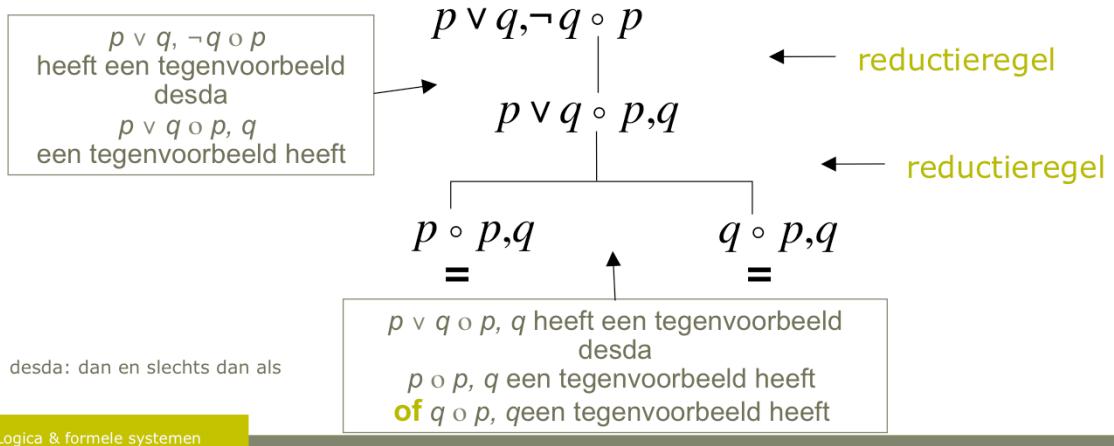
## Geldig gevolg – semantisch tableau

- Voorbeeld van een semantisch tableau:
  - Heeft  $p \vee q, \neg q / p$  een tegenvoorbeeld?
  - Wordt herleid tot: heeft  $p \vee q, \neg q \circ p$  een tegenvoorbeeld?  
 $\neg q$  waar maken is hetzelfde als  $q$  onwaar maken,  
dus m.a.w. heeft  $p \vee q \circ p, q$  een tegenvoorbeeld?  
Nu 2 gevallen ( $p \vee q$ ):
    - $p \circ p, q$  heeft tegenvoorbeeld of
    - $q \circ p, q$  heeft tegenvoorbeeldbeide kunnen geen tegenvoorbeeld hebben (de opmerking!)  
dus  $p \vee q, \neg q / p$  heeft geen tegenvoorbeeld,  
dus  $p \vee q, \neg q \nvDash p$ .



# Geldig gevolg

- Nu als semantisch tableau:  
Heeft  $p \vee q, \neg q / p$  een tegenvoorbeeld?





# Geldig gevolg – semantisch tableau

## Techniek van de semantische tableau

- Bij elke connectief hoort een linker en een rechter reductieregel:
  - Linkerregel: toepassen als de connectief links staat
  - Rechterregel: toepassen als de connectief aan de rechterkant staat
- De regels voor  $\neg$

$$\neg_L : \Phi, \neg \alpha \circ \Psi \quad \begin{array}{c} | \\ \Phi \circ \alpha \circ \Psi \end{array}$$

$$\neg_R : \Phi \circ \neg \alpha, \Psi \quad \begin{array}{c} | \\ \Phi, \alpha \circ \Psi \end{array}$$

De reductieregels zijn belangrijk voor de oefeningen! Te kennen



# Geldig gevolg – semantisch tableau

- Voor de  $\wedge$  en  $\vee$  connectieven:

$$\wedge_L : \Phi, \alpha \wedge \beta \circ \Psi$$
$$\Phi, \alpha, \beta \circ \Psi$$

$$\wedge_R : \Phi \circ \alpha \wedge \beta, \Psi$$
$$\Phi \circ \alpha, \Psi \quad \Phi \circ \beta, \Psi$$

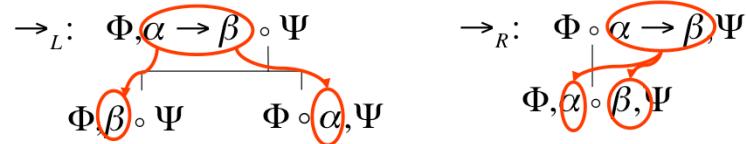
$$\vee_L : \Phi, \alpha \vee \beta \circ \Psi$$
$$\Phi, \alpha \circ \Psi \quad \Phi, \beta \circ \Psi$$

$$\vee_R : \Phi \circ \alpha \vee \beta, \Psi$$
$$\Phi \circ \alpha, \beta, \Psi$$



# Geldig gevolg – semantisch tableau

- Reductieregels voor de connectieven  $\rightarrow$  en  $\leftrightarrow$





## Geldig gevolg – semantisch tableau

- Door het telkens toepassen van een reductieregel op de sequenten krijgen we **een boom**
- Als er geen reductieregel meer kan worden toegepast spreken we van een semantisch tableau
  - **Elke tak** correspondeert met **een mogelijke route** om een **tegen voorbeeld** te vinden.



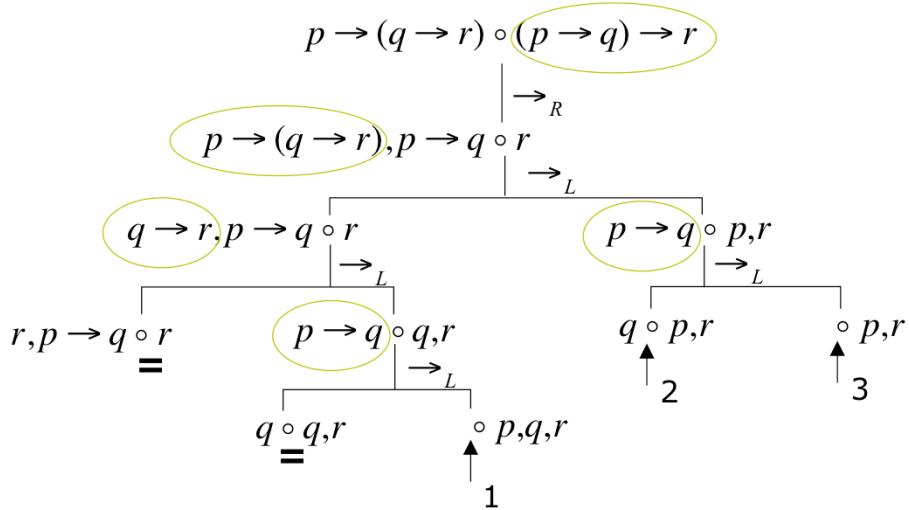
## Geldig gevolg – semantisch tableau

- Indien links en rechts in de tak (sequent) dezelfde formule optreedt is er geen tegenvoorbeeld te vinden. Men noemt de tak gesloten
- Een tak is open indien de tak niet gesloten is en er kunnen geen reductiestappen meer worden toegepast op deze tak
- Een tableau is gesloten als al zijn takken gesloten zijn:
  - Er is geen enkel tegenvoorbeeld te vinden, dus geldig gevolg!
- Een tableau is open als er tenminste één tak niet gesloten kan worden
  - Dus tenminste één tegenvoorbeeld en dus geen geldig gevolg.



# Geldig gevolg – semantisch tableau

Voorbeeld: Is  $p \rightarrow (q \rightarrow r) / (p \rightarrow q) \rightarrow r$  geldig?





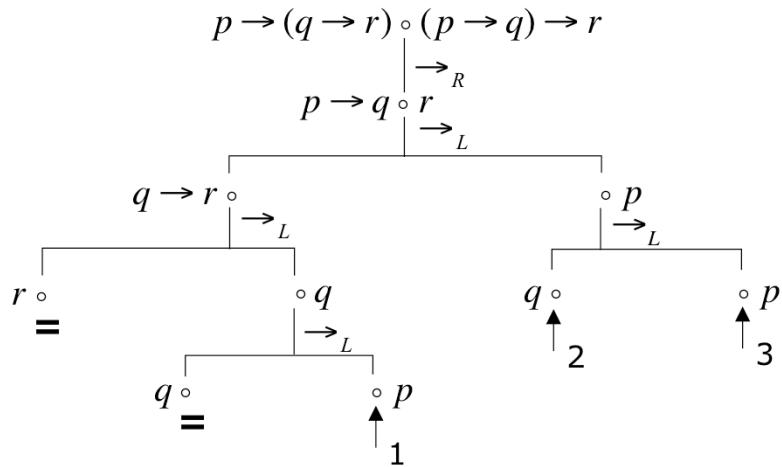
# Geldig gevolg – semantisch tableau

- Er worden 3 tegenvoorbeelden gevonden:
  1.  $V_1(p) = V_1(q) = V_1(r) = 0$
  2.  $V_2(q) = 1, V_2(p) = V_2(r) = 0$
  3.  $V_3(p) = V_3(r) = 0$
- Conclusie:  $p \rightarrow (q \rightarrow r)$  o  $(p \rightarrow q) \rightarrow r$  heeft tegenvoorbeelden, en dus:  
 $p \rightarrow (q \rightarrow r) \not\models (p \rightarrow q) \rightarrow r$
- Een semantisch tableau vindt **alle** tegenvoorbeelden!



# Geldig gevolg – semantisch tableau

- Is  $p \rightarrow (q \rightarrow r) / (p \rightarrow q) \rightarrow r$  geldig?  
Nogmaals met verkorte schrijfwijze: enkel wat wijzigt





# Semantisch Consistent

## Definitie

Een formuleverzameling  $\Sigma$  is **semantisch consistent** als  $\Sigma$  een model heeft

- Dus  $\Sigma = \{\varphi_1, \dots, \varphi_n\}$  is consistent als  $\varphi_1, \dots, \varphi_n$  o een tegenvoorbeeld heeft.

Het tegenvoorbeeld is nl de waardering die alle formules in  $\Sigma$  (links in het sequent) waar maakt (en rechts onwaar maakt – maar dat is leeg)

- Of nog  $\Sigma = \{\varphi_1, \dots, \varphi_n\}$  is consistent als  $\varphi_1, \dots, \varphi_n$  o een open tableau heeft

Een formuleverzameling zonder model heet *inconsistent*. Het corresponderende tableau is gesloten.

Semantisch consistent betekent dus dat er minstens 1 waardering is die alle formules in de verzameling waar maakt.



## Tautologie Testen

- Tautologie testen met behulp van een semantische tableau:
  - Herinner: Een formule  $\varphi$  is een *tautologie* als alle waarderingen model zijn van  $\varphi$
  - m.a.w. als er geen waardering te vinden is zodat  $V(\varphi) = 0$
  - m.a.w. er is geen tegenvoorbeeld voor  $\circ \varphi$
  - m.a.w. als het semantische tableau voor  $\circ \varphi$  gesloten is.



## Geldig gevolg

- Voorbeelden:

- is  $\{p \vee q, \neg(p \rightarrow \neg r), q \rightarrow r, \neg(r \wedge p)\}$  consistent of inconsistent?
- is  $\neg(p \wedge \neg p)$  een tautologie?



## Geldig gevolg - Adequaatheidstelling

- Semantische tableaus werden geïntroduceerd als een methode om de geldigheid van een gevolgtrekking te bepalen

### Adequaatheidstelling:

$$\varphi_1, \dots, \varphi_n \models \psi$$

desda

er bestaat een gesloten semantisch tableau voor  $\varphi_1, \dots, \varphi_n \circ \psi$

desda

alle semantische tableaus voor  $\varphi_1, \dots, \varphi_n \circ \psi$  zijn gesloten

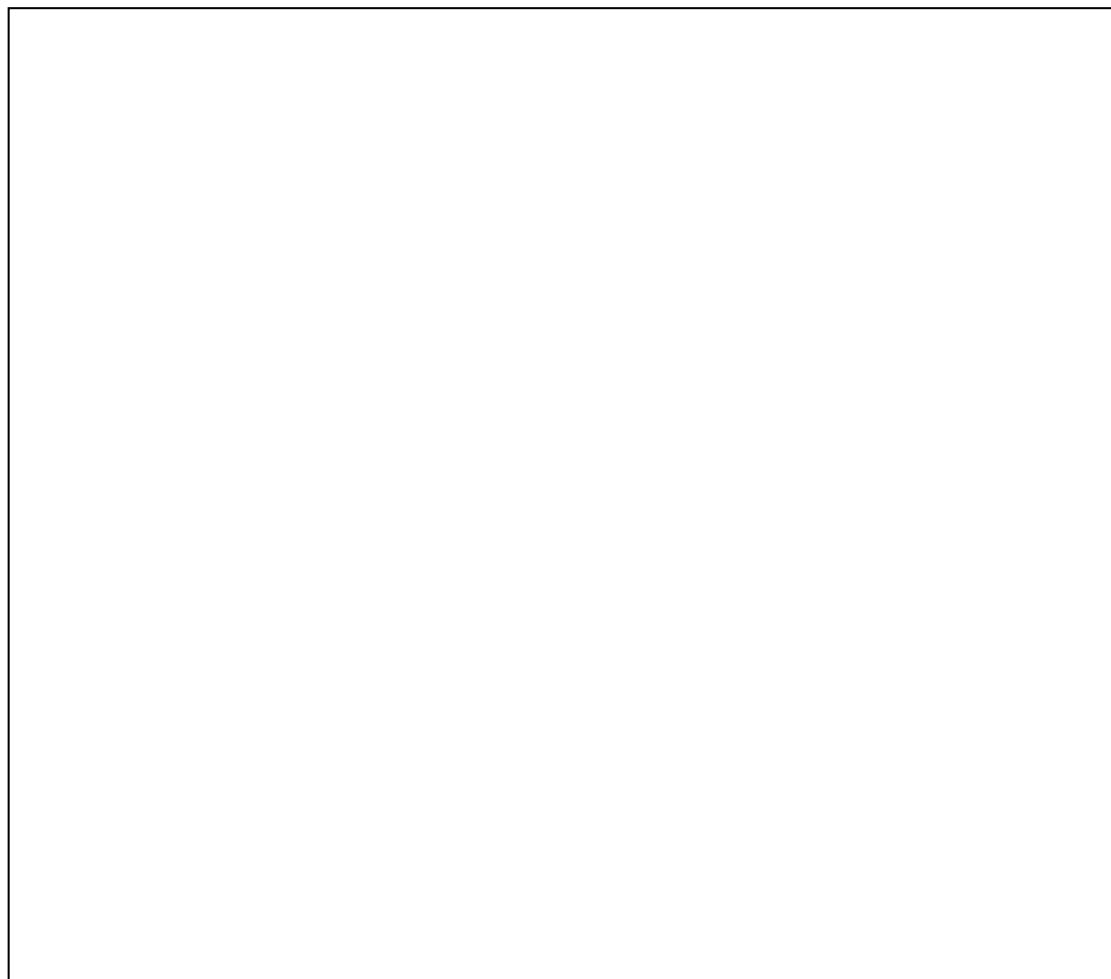
Bewijs later

Dus de adequaatheidstelling bewijst dat semantische tableaus een correcte methode is om geldig gevolg te bewijzen.



Vrije Universiteit Brussel

## Propositielogica: Afleidingen





# Propositielogica: afleidingen

## – Inhoud

- Afleiding: lineair, boomvorm
- Natuurlijke deductie
- Afleidingsregels: eliminatieregels, introductieregels
- (Hulp)aanname
- Afleidbaar
- Syntactisch consistent en inconsistent
- Volledigheidsstelling



# Afleidingen

Vorig hoofdstuk: semantiek uitgedrukt in termen van modellen

Nu: Hoe conclusies trekken uit aannames door middel van regels. Dit wordt **afleiden** genoemd

Voorbeeld regels:

als  $\varphi \wedge \psi$  dan  $\varphi$   
als  $\varphi$  en  $\psi$  dan  $\varphi \wedge \psi$

Voorbeeld afleiding:

1. Jan vertelt een verhaal en Piet leest de krant
2. Als Jan een verhaal vertelt, dan lacht Marie
3. Als Piet de krant leest, dan kijkt Wilma televisie

Stap 1: Uit 1. leiden we af: 1a. **Jan vertelt een verhaal**

Stap 2: Uit 2. en 1a. leiden we af: 2a. **Marie lacht**

Stap 3: Uit 1. leiden we af: 1b. **Piet leest de krant**

Stap 4: Uit 3. en 1b. leiden we af: 2b. **Wilma kijkt televisie**

Stap 5: Uit 2a. en 2b leiden we af: **Marie lacht en Wilma kijkt televisie**



# Afleidingsregels

- Afleidingsregels lineair voorgesteld:
  - voorbeeld: als  $\varphi \wedge \psi$  dan  $\varphi$
- Compactere voorstelling: boomvorm

$$\frac{\varphi_1 \dots \varphi_n}{\psi} \quad \begin{matrix} \xleftarrow{\hspace{1cm}} & \text{aannames} \\ & \xleftarrow{\hspace{1cm}} \end{matrix} \quad \frac{\varphi \wedge \psi}{\varphi}$$



# Afleidingen

- **Boomvorm voor het voorbeeld**

1. Jan vertelt een verhaal en Piet leest de krant
2. Als Jan een verhaal vertelt, dan lacht Marie
3. Als Piet de krant leest, dan kijkt Wilma televisie

j: Jan vertelt verhaal

p: Piet leest de krant

m: Marie lacht

w: Wilma kijkt televisie

$$\frac{\frac{j \wedge p}{\frac{j \quad j \rightarrow m}{m}} \quad \frac{j \wedge p}{\frac{p \quad p \rightarrow w}{w}}}{m \wedge w}$$



## Afleidingen - voorbeelden regels

- Voorbeelden van afleidingsregels voor het systeem van natuurlijke deductie:

$$\frac{\varphi \wedge \psi}{\varphi}$$

$\wedge$ -eliminatie

notatie:  $\wedge E$

$$\frac{\varphi \wedge \psi}{\psi}$$

$$\frac{\varphi \quad \psi}{\varphi \wedge \psi}$$

$\wedge$ -introductie

$\wedge I$

$$\frac{\varphi \quad \varphi \rightarrow \psi}{\psi}$$

$\rightarrow$ -eliminatie

$\rightarrow E$

- Naam van de toegepaste regel wordt naast de streep genoteerd

$$\frac{\varphi \wedge \psi}{\varphi} \quad \wedge E$$



# Afleidingen

Als we met de gegeven afleidingsregels een formule  $\psi$  kunnen afleiden uit een formule  $\varphi$ , dan schrijven we:

$$\varphi \vdash \psi$$

De formule  $\varphi$  noemt men de **aanname** in het bewijs

Vaak hebben we tijdens het afleiden **hulpaannames** nodig

- Daarom, om te zien op grond van welke aannames een formule geldt schrijven we:

$\varphi$  uit  $\Sigma$

of

$\varphi$  uit  $\psi_1, \dots, \psi_n$

- Notatie voor een aanname:  $\varphi$  uit  $\varphi$



## Afleidingen - definitie

### *Definitie*

Een formule  $\varphi$  heet **afleidbaar** uit een verzameling aannames  $\Sigma$  als er een afleiding van  $\varphi$  bestaat op basis van de gegeven afleidingsregels waarin aan het eind alleen nog aannames uit  $\Sigma$  van kracht zijn

Notatie:  $\Sigma \vdash \varphi$

- Als  $\varphi$  niet afleidbaar is noteren we dit  $\Sigma \not\vdash \varphi$



# Natuurlijke deductie - regels

De afleidingsregels voor natuurlijke deductie:

## $\wedge$ -Eliminatieregels

$$\frac{\varphi \wedge \psi \text{ uit } \Sigma}{\varphi \text{ uit } \Sigma} \wedge E \quad \frac{\varphi \wedge \psi \text{ uit } \Sigma}{\psi \text{ uit } \Sigma} \wedge E$$

## $\wedge$ -Introductieregel

$$\frac{\varphi \text{ uit } \Sigma \quad \psi \text{ uit } \Phi}{\varphi \wedge \psi \text{ uit } \Sigma \cup \Phi} \wedge I$$

## $\rightarrow$ -Eliminatieregels

$$\frac{\varphi \rightarrow \psi \text{ uit } \Sigma \quad \varphi \text{ uit } \Phi}{\psi \text{ uit } \Sigma \cup \Phi} \rightarrow E$$

## $\rightarrow$ -Introductieregel

$$\frac{\psi \text{ uit } \Sigma, \varphi}{\varphi \rightarrow \psi \text{ uit } \Sigma} \rightarrow I, [-\varphi]$$

De hulpaanname  $\varphi$  kan door deze regel terug worden ingetrokken

Geef aan dat de hulpaanname  $\varphi$  ingetrokken wordt

Te kennen! Belangrijk voor de oefeningen!



# Afleidingen - regels

## *v-Introductieregels*

$$\frac{\varphi \text{ uit } \Sigma}{\varphi \vee \psi \text{ uit } \Sigma} \vee I \quad \frac{\psi \text{ uit } \Sigma}{\varphi \vee \psi \text{ uit } \Sigma} \vee I$$

## *v-Eliminatieregel*

$$\frac{\varphi \vee \psi \text{ uit } \Sigma \quad \alpha \text{ uit } \Phi, \varphi \quad \alpha \text{ uit } \Psi, \psi}{\alpha \text{ uit } \Sigma \cup \Phi \cup \Psi} \vee E[\neg \varphi, \neg \psi]$$

De aanwezigheid van de disjunctie laat ons toe om de hulpaannames  $\varphi$  en  $\psi$  te schrappen

Geef aan dat de hulpaannames  $\varphi$  en  $\psi$  ingetrokken worden



# Afleidingen - regels

## $\neg$ -Eliminatieregels

$$\frac{\varphi \text{ uit } \Phi \quad \neg\varphi \text{ uit } \Psi}{\psi \text{ uit } \Phi \cup \Psi} \neg E$$

Uit een tegenspraak volgt eender wat

$$\frac{\varphi \text{ uit } \Phi, \neg\psi \quad \neg\varphi \text{ uit } \Psi, \neg\psi}{\psi \text{ uit } \Phi \cup \Psi} \neg E^* [\neg\neg\psi]$$

Bewijs uit het ongerijmde: iets bewijzen door het tegendeel te weerleggen

Merk op:  $\neg\psi$  is hulpaanname die ingetrokken wordt

## $\neg$ -Introductieregel

$$\frac{\varphi \text{ uit } \sum, \psi \quad \neg\varphi \text{ uit } \Phi, \psi}{\neg\psi \text{ uit } \sum \cup \Phi} \neg I [-\psi]$$

Ook hier: uit een tegenspraak volgt eender wat. Hier wordt  $\psi$  als hulpaanname gebruikt. Merk op dat de weerlegde aanname  $\psi$  ook terug wordt ingetrokken.



## Afleidingsregels - notatie

- Al de afleidingsregels hebben een equivalente lineaire vorm, b.v.  
 $(\wedge I)$  : Als  $\varphi$  uit  $\Phi$  en  $\psi$  uit  $\Psi$ , dan  $\varphi \wedge \psi$  uit  $\Phi \cup \Psi$ .
- Om de leesbaarheid te verhogen worden de namen van de toegepaste regels vaak weggelaten.



## Afleidingen – voorbeeld 1

– Afleiding: voorbeeld 4.1

in boomvorm:  $p \wedge (q \wedge r) \vdash (p \wedge q) \wedge r$

$$\varphi = p \wedge (q \wedge r)$$

$$\frac{p \wedge (q \wedge r) \text{ uit } \varphi}{\frac{p \text{ uit } \varphi}{p \wedge q \text{ uit } \varphi}} \wedge E \quad \frac{\frac{p \wedge (q \wedge r) \text{ uit } \varphi}{q \wedge r \text{ uit } \varphi} \wedge E}{\frac{q \text{ uit } \varphi}{(p \wedge q) \wedge r \text{ uit } \varphi}} \wedge I \quad \frac{\frac{p \wedge (q \wedge r) \text{ uit } \varphi}{r \text{ uit } \varphi} \wedge E}{\frac{q \wedge r \text{ uit } \varphi}{(p \wedge q) \wedge r \text{ uit } \varphi}} \wedge I$$



## Afleidingen – voorbeeld 2

– Afleiding: voorbeeld 4.2

in boomvorm:  $p \rightarrow (q \rightarrow r) \vdash (p \rightarrow q) \rightarrow (p \rightarrow r)$

$$\varphi = p \rightarrow (q \rightarrow r)$$

$$\frac{\frac{p \text{ uit } p \quad p \rightarrow q \text{ uit } p \rightarrow q}{q \text{ uit } p, p \rightarrow q} \rightarrow E \quad \frac{\frac{p \text{ uit } p \quad p \rightarrow (q \rightarrow r) \text{ uit } \varphi}{q \rightarrow r \text{ uit } p, \varphi} \rightarrow E}{\frac{r \text{ uit } p, p \rightarrow q, \varphi}{p \rightarrow r \text{ uit } p \rightarrow q, \varphi} \rightarrow I, [-1]} \rightarrow E}{(p \rightarrow q) \rightarrow (p \rightarrow r) \text{ uit } \varphi \rightarrow I, [-2]}$$



## Afleidingen – voorbeeld 3

– Afleiding: voorbeeld 4.3

in boomvorm:  $p \rightarrow (q \rightarrow r) \vdash (p \wedge q) \rightarrow r$

$$\varphi = p \rightarrow (q \rightarrow r)$$

$$\frac{\frac{\frac{p \wedge q \text{ uit } p \wedge q}{q \text{ uit } p \wedge q} \wedge E \quad \frac{\frac{p \wedge q \text{ uit } p \wedge q}{p \text{ uit } p \wedge q} \wedge E \quad \frac{p \rightarrow (q \rightarrow r) \text{ uit } \varphi}{q \rightarrow r \text{ uit } \varphi, p \wedge q}}{q \rightarrow r \text{ uit } \varphi, p \wedge q} \rightarrow E}{r \text{ uit } p \wedge q, \varphi} \rightarrow I, [-1]}{(p \wedge q) \rightarrow r \text{ uit } \varphi} \rightarrow E$$



## Afleidingen – voorbeeld 4

– Afleiding: voorbeeld 4.4

in boomvorm:  $(p \vee q) \rightarrow r \vdash (p \rightarrow r) \wedge (q \rightarrow r)$

$$\varphi = (p \vee q) \rightarrow r$$

$$\frac{\frac{p \text{ uit } p}{p \vee q \text{ uit } p} \vee I \quad \frac{(p \vee q) \rightarrow r \text{ uit } \varphi}{r \text{ uit } p, \phi} \rightarrow I[-1]}{p \rightarrow r \text{ uit } \phi} \rightarrow E \quad \frac{\frac{q \text{ uit } q}{p \vee q \text{ uit } q} \vee I \quad \frac{(p \vee q) \rightarrow r \text{ uit } \varphi}{q \rightarrow r \text{ uit } \phi} \rightarrow I[-2]}{q \rightarrow r \text{ uit } \phi} \rightarrow E}{(p \rightarrow r) \wedge (q \rightarrow r) \text{ uit } \phi} \wedge I$$



## Afleidingen – voorbeeld 5

– Afleiding: voorbeeld 4.5

$$(p \rightarrow r) \wedge (q \rightarrow r) \vdash (p \vee q) \rightarrow r$$
$$\varphi = (p \rightarrow r) \wedge (q \rightarrow r)$$



## Afleidingen – voorbeeld 6

– Afleiding: voorbeeld 4.6

$$p \rightarrow q \vdash \neg q \rightarrow \neg p$$

$$\varphi = p \rightarrow q$$



## Stelling - definitie

### *Definitie*

Als  $\varphi$  afleidbaar is zonder aannames, dan heet  $\varphi$  **een stelling** ( $\Sigma$  is leeg).

Notatie:  $\vdash \varphi$

- Als  $\varphi$  geen stelling is noteren we  $\not\vdash \varphi$



## Afleidingen en stellingen

Alle natuurlijke deducties leiden tot stellingen indien men alle aannames intrekt met de regel

$\rightarrow I$

(*zonder bewijs*)

– Voorbeeld:

$\varphi, \psi \vdash \zeta$  wordt  $\varphi \vdash \psi \rightarrow \zeta$   
en dit wordt

$$\vdash \varphi \rightarrow (\psi \rightarrow \zeta)$$



## Afleidingen – voorbeeld 7

– Afleiding: voorbeeld 4.7

1e wet van de Morgan:  $(\neg p \wedge \neg q) \Leftrightarrow \neg(p \vee q)$

Eerst:  $\vdash \neg(p \vee q) \rightarrow (\neg p \wedge \neg q)$  ( $\varphi = \neg(p \vee q)$ )

dan  $\vdash (\neg p \wedge \neg q) \rightarrow \neg(p \vee q)$  ( $\varphi = \neg p \wedge \neg q$ )



## Afleidingen – voorbeeld 8

– Afleiding: voorbeeld 4.8

$$\neg\neg p \leftrightarrow p$$

Eerst:  $\vdash \neg\neg p \rightarrow p$

- |                               |                  |                     |
|-------------------------------|------------------|---------------------|
| 1. $\neg\neg p$               | uit $\neg\neg p$ | aanname             |
| 2. $\neg p$                   | uit $\neg p$     | aanname             |
| 3. $p$                        | uit $\neg\neg p$ | $\neg E^*(-2)$      |
| 4. $\neg\neg p \rightarrow p$ | uit $\emptyset$  | $\rightarrow I(-1)$ |

Tweeds:  $\vdash p \rightarrow \neg\neg p$

- |                               |                 |                     |
|-------------------------------|-----------------|---------------------|
| 1. $p$                        | uit $p$         | aanname             |
| 2. $\neg p$                   | uit $\neg p$    | aanname             |
| 3. $\neg\neg p$               | uit $p$         | $\neg I(-2)$        |
| 4. $p \rightarrow \neg\neg p$ | uit $\emptyset$ | $\rightarrow I(-1)$ |



## Afleidingen – voorbeeld 9

Afleiding: voorbeeld 4.9

$$\vdash p \rightarrow (q \rightarrow p)$$

- |                                      |                 |                     |
|--------------------------------------|-----------------|---------------------|
| 1. $p$                               | uit $p$         | aanname             |
| 2. $q$                               | uit $q$         | aanname             |
| 3. $q \rightarrow p$                 | uit $p$         | $\rightarrow I(-2)$ |
| 4. $p \rightarrow (q \rightarrow p)$ | uit $\emptyset$ | $\rightarrow I(-1)$ |



# Syntactisch consistent - def

## Definitie

Een verzameling formules  $\Gamma$  heet **syntactisch consistent** wanneer er geen formule  $\varphi$  is waarvoor zowel  $\Gamma \vdash \varphi$  als  $\Gamma \vdash \neg\varphi$

- Een verzameling formules die niet (syntactisch) consistent is heet **syntactisch inconsistent**
- Voorbeelden:
  - $\{\neg p, p \rightarrow q, q\}$  is (syntactisch) consistent
  - $\Gamma = \{p, p \rightarrow q, \neg q\}$  is (syntactisch) inconsistent,  
nl.  $\Gamma \vdash \neg q$  en  $\Gamma \vdash q$

Intuitief: uit een syntactisch consistente verzameling van formules kunnen we geen contradictie afleiden



## Syntactisch consistent - bewering

Bewering:

Een formuleverzameling  $\Gamma$  is syntactisch consistent  $\Leftrightarrow$   
er bestaat een formule  $\varphi$  zodat  $\Gamma \vdash \varphi$

Bewijs

$\Rightarrow TB$ :  $\Gamma$  is consistent dan bestaat er een formule  $\varphi$  zodat  $\Gamma \vdash \varphi$

Als  $\Gamma$  consistent is, dan is er geen formule  $\varphi$  waarvoor  $\Gamma \vdash \varphi$  en  $\Gamma \vdash \neg \varphi$ .

Voor een willekeurige formule  $\xi$  geldt dan: ofwel  $\Gamma \vdash \xi$  ofwel  $\Gamma \vdash \neg \xi$ .

Als  $\Gamma \vdash \xi$  dan is de  $\xi$  gezochte formule, zoniet is  $\neg \xi$  de gezochte formule.

We hebben dus bewezen dat er een formule  $\varphi$  bestaat zodat  $\Gamma \vdash \varphi$

TB staat voor Te Bewijzen

Bewijs in 2 richtingen.



## Syntactisch consistent - bewering

Bewering:

Een formuleverzameling  $\Gamma$  is consistent  $\Leftrightarrow$   
er bestaat een formule  $\varphi$  zodat  $\Gamma \vdash \varphi$

### Bewijs (deel 2)

$\Leftarrow$  TB: Als er een formule  $\varphi$  zodat  $\Gamma \vdash \varphi$  dan is  $\Gamma$  consistent

Bewijs via contrapositie, m.a.w. we bewijzen:

Als  $\Gamma$  inconsistent is dan bestaat er geen formule  $\varphi$  zodat  $\Gamma \vdash \varphi$

Bewijs: stel  $\Gamma$  inconsistent, dan bestaat er een formule  $\xi$  zodat  $\Gamma \vdash \xi$  en  $\Gamma \vdash \neg \xi$ .

Maar dan is met de  $\neg E$  regel elke formule afleidbaar uit  $\Gamma$ , en dus kan er geen  
formule  $\varphi$  bestaat zodat  $\Gamma \vdash \varphi$



## Syntactisch consistent – bewering 2

Bewering:

$\Gamma \vdash \varphi \Leftrightarrow \Gamma \cup \{\neg\varphi\}$  is syntactisch consistent

Bewijs (Beide richtingen telkens via contrapositie)

$\Rightarrow$  TB:  $\Gamma \vdash \varphi$  dan  $\Gamma \cup \{\neg\varphi\}$  consistent, of nog:  $\Gamma \cup \{\neg\varphi\}$  inconsistent dan  $\Gamma \vdash \varphi$   
Stel dus  $\Gamma \cup \{\neg\varphi\}$  is inconsistent.

Dan is er een formule  $\xi$  waarvoor geldt  $\Gamma \cup \{\neg\varphi\} \vdash \xi$  en  $\Gamma \cup \{\neg\varphi\} \vdash \neg\xi$ .  
De  $\neg E^*$  regel geeft dan:  $\Gamma \vdash \varphi$  (elke formule is afleidbaar).

$\Leftarrow$  TB:  $\Gamma \cup \{\neg\varphi\}$  is consistent dan  $\Gamma \vdash \varphi$ , of nog:  $\Gamma \vdash \varphi$  dan  $\Gamma \cup \{\neg\varphi\}$  is inconsistent

Stel  $\Gamma \vdash \varphi$ , dan ook  $\Gamma \cup \{\neg\varphi\} \vdash \varphi$ . Omdat ook  $\Gamma \cup \{\neg\varphi\} \vdash \neg\varphi$ , is  $\Gamma \cup \{\neg\varphi\}$  inconsistent



## Intermezzo: Axiomatisch afleiden

- Axiomatisch afleiden
  - In **natuurlijke deductie** spelen de **afleidingsregels** de belangrijkste rol
  - In **axiomatisch afleiden** spelen **axioma's** de hoofdrol
  - Een axioma is een formule die op elk moment in een bewijs kan gebruikt worden
- Een **axiomatisch systeem** bestaat dan ook uit een **verzameling axioma's en afleidingsregels**.



## Axiomatisch afleiden - voorbeeld

- Voorbeeld

Axioma's (S)

$$\varphi \rightarrow (\psi \rightarrow \varphi)$$

$$(\varphi \rightarrow (\psi \rightarrow \chi)) \rightarrow ((\varphi \rightarrow \psi) \rightarrow (\varphi \rightarrow \chi))$$

$$(\neg \varphi \rightarrow \neg \psi) \rightarrow (\psi \rightarrow \varphi)$$

Afleidingsregel (Modus Ponens): Uit  $\varphi$  en  $(\varphi \rightarrow \psi)$  mogen we ook  $\psi$  afleiden

$\Sigma \vdash_S \psi$  :  $\psi$  afleidbaar uit  $\Sigma$  met behulp van S en de afleidingsregel

als  $\varphi$  een axioma is, dan  $\vdash_S \varphi$



# Volledigheidsstelling

- Verband tussen syntactische afleidbaarheid en semantische geldigheid

## Volledigheidsstelling

Voor de propositielogica geldt

Als  $\Sigma$  een formuleverzameling is, en  $\varphi$  een formule, dan geldt:

$$\Sigma \vdash \varphi \text{ desda } \Sigma \models \varphi$$

- Nog twee belangrijke begrippen:
  - **Correctheid**: afleidbare gevolgtrekkingen zijn semantisch geldig
  - **Volledigheid**: semantisch geldige gevolgtrekkingen zijn afleidbaar.

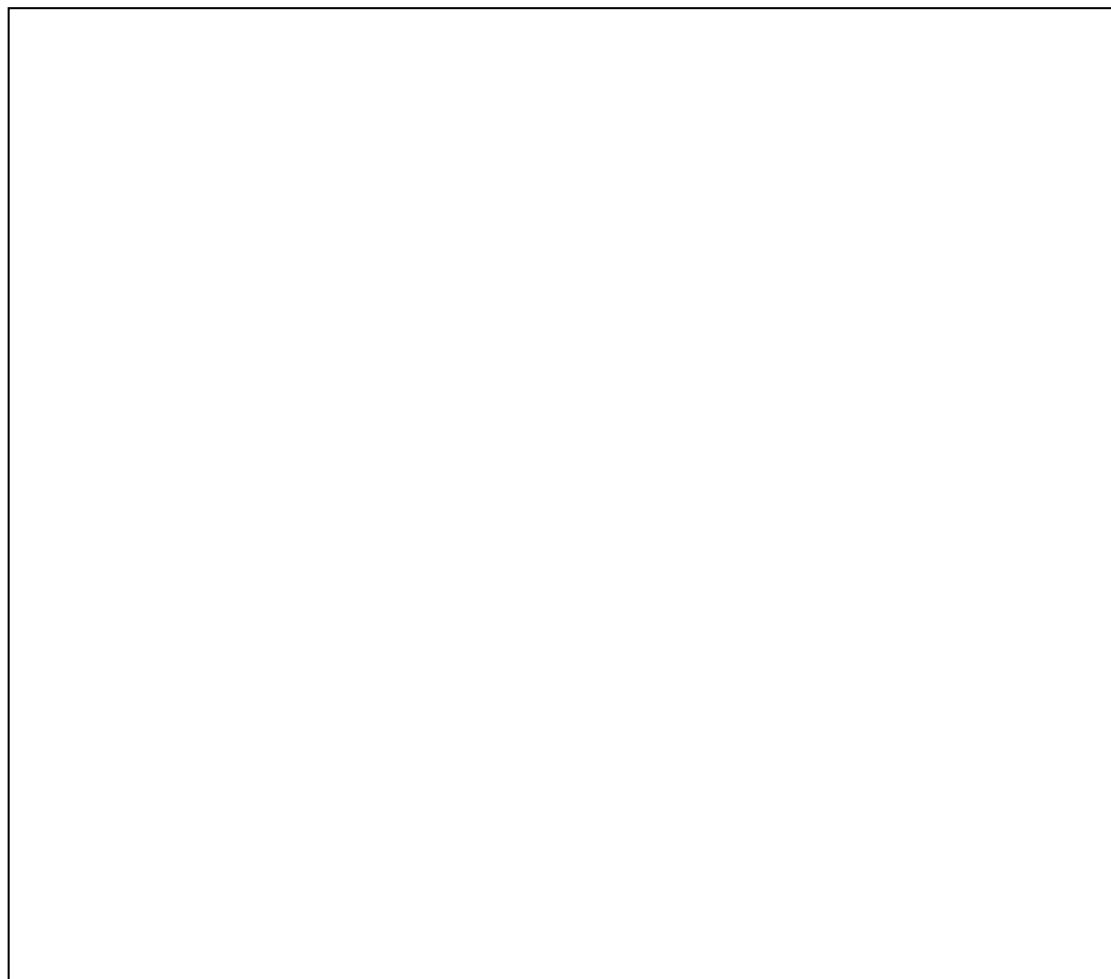
m.a.w. alle formules die geldig gevolg zijn van een bepaalde formuleverzameling zijn ook afleidbaar uit die formuleverzameling en omgekeerd alle formules die afleidbaar zijn uit een bepaalde formuleverzameling zijn ook geldig gevolg van deze formuleverzameling.

Hieruit volgt dus dat de 2 begrippen gelijkwaardig zijn.



Vrije Universiteit Brussel

## Propositielogica: Metatheorie





# Propositielogica: Metatheorie

## – Inhoud:

- Metatheorie
- Formule inductie
- Inductiehypothese
- Functionele volledigheid
- Dualiteit
- Adequaatheid van semantische tableaus
- Correctheid en Volledigheid



# Metatheorie

- **Metabeweringen:** beweringen **over** het systeem (in tegenstelling tot de beweringen uitgedrukt in het systeem)
  - Voorbeelden:
    - Functionele volledigheid van connectieven
    - Correctheid van bewijssystemen
- Welke bewijsvormen gebruiken we op dit metaniveau?
  - Gebeurt in principe **met dezelfde logica** als deze van ons logisch systeem
  - Specifieke meta redeneervorm: **formule-inductie.**



# Formule-inductie

## • Formule-inductie

- Aantonen dat een bewering  $B$  geldt voor alle formules uit de propositielogica
  - Gebaseerd op het principe dat formules zijn opgebouwd vanuit een basisverzameling van atomen via eindige combinatie met logische operatoren
  - Basisstap:  
Laat zien dat  $B$  opgaat voor de atomen (propositieletters)
  - Inductiestap:  
Toon aan dat, als  $B$  opgaat voor de formules  $\varphi$ ,  $\psi$ , dan ook voor hun combinaties  $\neg\varphi$ ,  $(\varphi \wedge \psi)$ ,  $(\varphi \vee \psi)$ ,  $(\varphi \rightarrow \psi)$  en  $(\varphi \leftrightarrow \psi)$
- De aannname in de inductiestap (dat  $B$  opgaat voor de formules  $\varphi$  en  $\psi$ ) heet de **inductiehypothese (IH)**.



## Formule-inductie - voorbeeld

– Voorbeeld formule-inductie:

- Definieer voor iedere formule  $\varphi$ 
  - $\text{subf}(\varphi)$  = het aantal subformules in  $\varphi$ , en
  - $\text{length}(\varphi)$  = het aantal symbolen in een formule  $\varphi$  met de hulpsymbolen ) en ( niet meegerekend.
  - Voorbeeld:  $\text{subf}((p \wedge \neg q) \rightarrow p) = 6$  en  $\text{length}((p \wedge \neg q) \rightarrow p) = 6$ .
- *Stelling:  $\text{subf}(\varphi) = \text{length}(\varphi)$  voor iedere formule  $\varphi$ .*



## Formule-inductie - voorbeeld

### Bewijs:

- **Basisstap:** Voor  $\varphi = p$ , een propositionele letter, geldt  
 $\text{subf}(p) = 1$  en  $\text{length}(p) = 1$ .  
Dus  $\text{subf}(p) = \text{length}(p)$ .
- **Inductiestap:** Stel dat de bewering geldt voor de formules  $\alpha$  en  $\beta$  (de inductiehypothese IH). Dan bewijzen we dat de bewering ook geldt voor:  
 $\varphi = \neg\alpha$ ,  $\varphi = (\alpha \wedge \beta)$ ,  $\varphi = (\alpha \vee \beta)$ ,  $\varphi = (\alpha \rightarrow \beta)$  en  $\varphi = (\alpha \leftrightarrow \beta)$ 
  - voor  $\varphi = \neg\alpha$  is  
 $\text{subf}(\neg\alpha) = \text{subf}(\alpha) + 1 =_{IH} \text{length}(\alpha) + 1 = \text{length}(\neg\alpha)$
  - voor  $\varphi = (\alpha \wedge \beta)$  is  
 $\text{subf}(\alpha \wedge \beta) = \text{subf}(\alpha) + \text{subf}(\beta) + 1 =_{IH} \text{length}(\alpha) + \text{length}(\beta) + 1 = \text{length}(\alpha \wedge \beta)$
  - idem voor de andere gevallen.



# Functioneel volledig

## Stelling

De verzameling connectieven  $\{\neg, \wedge\}$  is functioneel volledig, m.a.w. voor elke formule  $\varphi$  is er een logisch equivalente formule  $\varphi'$  die alleen de connectieven  $\neg$  en  $\wedge$  bevat.

### Bewijs:

– **Basisstap:** voor atomaire formules geldt dit automatisch want deze bevatten geen connectieven

– **Inductiestap:**

IH: stel  $\varphi$  en  $\psi$  herschreven kunnen worden als logisch equivalente formule  $\varphi'$  en  $\psi'$  en  $\varphi'$  en  $\psi'$  zijn met alleen  $\neg$  en  $\wedge$  geconstrueerd. ( $\varphi \equiv \varphi'$  en  $\psi \equiv \psi'$ ;  $\equiv$  herschreven)

TB: ook  $\neg\varphi$ ,  $\varphi \wedge \psi$ ,  $\varphi \vee \psi$ ,  $\varphi \rightarrow \psi$  en  $\varphi \leftrightarrow \psi$  zo te herschrijven

Bewijs:

$$(\neg\varphi)' \equiv \neg\varphi'$$

$$(\varphi \wedge \psi)' \equiv (\varphi' \wedge \psi')$$

$$(\varphi \vee \psi)' \equiv \neg(\neg\varphi' \wedge \neg\psi')$$

$$(\varphi \rightarrow \psi)' \equiv \neg(\varphi' \wedge \neg\psi')$$

$$(\varphi \leftrightarrow \psi)' \equiv (\varphi \rightarrow \psi)' \wedge (\psi \rightarrow \varphi)'$$



# Dualiteit

- Dualiteit:

Laat  $\varphi$  een formule zijn waarin alleen de connectieven  $\wedge$ ,  $\vee$  en  $\neg$  in voorkomen.

De **duale formule van  $\varphi$**  (notatie  $\varphi^d$ ) is de formule die ontstaat door elk voorkomen van  $\wedge$  in  $\varphi$  te vervangen door  $\vee$  en omgekeerd elk voorkomen van  $\vee$  door  $\wedge$ .

- Voorbeeld:

$$(p \wedge q)^d = (p \vee q)$$

$$(\neg p \vee (p \wedge q))^d = (\neg p \wedge (p \vee q))$$



# Stelling Dualiteit

## Stelling Dualiteit

Laat  $\varphi, \psi$  formules zijn waarin alleen de connectieven  $\wedge, \vee$  en  $\neg$  voorkomen, dan geldt:

$$\models \varphi \leftrightarrow \psi \Rightarrow \models \varphi^d \leftrightarrow \psi^d$$

## Bewijs

Laat voor een willekeurige formule  $\chi$  (waarin alleen de connectieven  $\wedge, \vee$  en  $\neg$  voorkomen),  $\chi^+$  de formule zijn die ontstaat door in  $\chi^d$  alle atomen te vervangen door hun negaties.

We bewijzen nu eerst:  $\models \neg\chi \leftrightarrow \chi^+$  (hulpstelling)



# Hulpstelling dualiteit

$$\models \neg\chi \leftrightarrow \chi^+$$

## Bewijs

Basisstap: Als  $\chi$  een atoom is dan is  $\chi^d = \chi$  en dus is  $\chi^+ = \neg\chi$

Inductiestap: laat de stelling gelden voor  $\alpha, \beta$  dus

$$\models \neg\alpha \leftrightarrow \alpha^+ \text{ en } \models \neg\beta \leftrightarrow \beta^+$$

negatie: te bewijzen:  $\models \neg\neg\alpha \leftrightarrow (\neg\alpha)^+$

$$\text{Bewijs: } (\neg\alpha)^+ = \neg(\alpha^+) \leftrightarrow \neg\neg\alpha$$

conjunctie:  $(\alpha \wedge \beta)^+ = (\alpha^+ \vee \beta^+)$  (definitie dualiteit)

$$(\alpha^+ \vee \beta^+) = (\neg\alpha \vee \neg\beta) \quad (\text{inductiehypothese})$$

$$(\neg\alpha \vee \neg\beta) = \neg(\alpha \wedge \beta) \quad (\text{2de wet van De Morgan})$$

$$\text{dus } \models (\alpha \wedge \beta)^+ \leftrightarrow \neg(\alpha \wedge \beta)$$

disjunctie: analoog



## Stelling dualiteit - bewijs

### Bewijs stelling

Stel  $\models \varphi \leftrightarrow \psi$ . Dan ook  $\models \neg\varphi \leftrightarrow \neg\psi$

Uit de vorige bewering volgt nu:  $\models \varphi^+ \leftrightarrow \psi^+$

Merk op: Als we de atomen van  $\chi^+$  vervangen door hun negatie dan krijgen we een formule  $\chi^+$  die logisch equivalent is met  $\chi^d$  (dubbele negatie)

Als we dit toepassen op  $\models \varphi^+ \leftrightarrow \psi^+$  (substitutie) krijgen we dan:  $\models \varphi^{+'} \leftrightarrow \psi^{+'}$  en dus  $\models \varphi^d \leftrightarrow \psi^d$ .



# Adequaatheidsstelling

## Bewijs van de Adequaatheidsstelling

Herinner: Adequaatheidsstelling

$$\varphi_1, \dots, \varphi_n \models \psi \text{ desda}$$

er bestaat een gesloten semantisch tableau voor

$$\varphi_1, \dots, \varphi_n \circ \psi \text{ desda}$$

alle semantische tableaus voor  $\varphi_1, \dots, \varphi_n \circ \psi$  zijn gesloten

Eerst 2 beweringen bewijzen



## Adequaatheidsstelling

Eerst veralgemening:

$\Phi \circ \Psi$  met  $\Phi = \varphi_1, \dots, \varphi_n$  en  $\Psi = \psi_1, \dots, \psi_m$ .

- Het tableau met  $\Phi \circ \Psi$  als topsequent is gesloten als

$\varphi_1 \wedge \dots \wedge \varphi_n \models \psi_1 \vee \dots \vee \psi_m$  of anders geschreven als  
 $\varphi_1, \dots, \varphi_n \models \psi_1, \dots, \psi_m$  of ook wel  $\Phi \models \Psi$

Let op het verschil tussen de conjuncties links en de disjuncties rechts! Waarom is dit zo?



# Adequaatheidsstelling – hulpstelling 1

## Bewering 1

Stel  $\Phi \circ \Psi$  is de topsequent van een gesloten tableau. Dan geldt  $\Phi \models \Psi$

## Bewijs

per inductie op het aantal knopen van de tableau

- Tableau heeft 1 knoop: omdat de tableau gesloten is betekent dit dat een zekere formule zowel links in  $\Phi$  als rechts in  $\Psi$  voorkomt. Dan kan er geen tegenvoorbeeld zijn en dus  $\Phi \models \Psi$

$$\Phi \circ \Psi$$

$$=$$



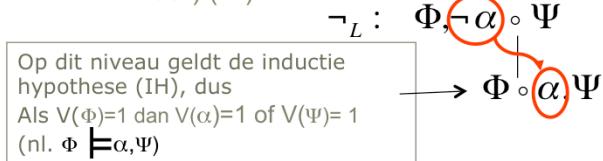
# Adequaatheidsstelling – hulpstelling 1

## Vervolg bewijs

Tableau heeft meerdere knopen: ontstaan door toepassen van een regel

– Geval I (reductie zonder splitsing): door toepassing van  $\neg_L$  regel

- Topsequent:  $\Phi, \neg\alpha \circ \Psi$
- Sequent er onder:  $\Phi \circ \alpha, \Psi$ . Hier geldt per inductie  $\Phi \models \alpha, \Psi$  (want 1 knoop minder) (IH)



TB: als  $V(\Phi)=1$  en  $V(\neg\alpha)=1$  dan  $V(\Psi)=1$  (nl.  $(\Phi, \neg\alpha \models \Psi)$ )

Stel dus een waardering  $V$  zodat  $V(\Phi)=1$  en  $V(\neg\alpha)=1$ . Uit IH volgt nu dat deze bewering dan  $\alpha$  of  $\Psi$  waar moet maken. Vermits  $V(\neg\alpha)=1$  en dus  $V(\alpha)=0$  moet  $V$  dus  $\Psi$  waar maken en hebben we dan bewezen:  $\Phi, \neg\alpha \models \Psi$



# Adequaatheidsstelling – hulpstelling 1

## Vervolg bewijs

Geval II (reductie met splitsing): : door toepassing van  $\wedge_R$  regel

- Topsequent:  $\Phi \circ \alpha \wedge \beta, \Psi$
- Sequenten er onder:  $\Phi \circ \alpha, \Psi$  en  $\Phi \circ \beta, \Psi$ . Hier geldt per inductie  $\Phi \models \alpha, \Psi$  en  $\Phi \models \beta, \Psi$  (want beide tableau's een knoop minder)

Op dit niveau geldt de inductie hypothese (IH):  
-  $V(\Phi)=1$  dan  $V(\alpha)=1$  of  $V(\Psi)=1$   
en  
-  $V(\Phi)=1$  dan  $V(\beta)=1$  of  $V(\Psi)=1$

$$\wedge_R : \Phi \circ \alpha \wedge \beta, \Psi \longrightarrow \Phi \circ \alpha, \Psi \quad \Phi \circ \beta, \Psi$$

TB:  $V(\Phi)=1$  dan  $V(\alpha \wedge \beta)=1$  of  $V(\Psi)=1$

Als  $V(\Phi)=1$  dan zal die waardering ofwel  $\Psi$  waar maken ofwel  $\alpha$  en  $\beta$  waar maken (door de IH) en dus ofwel  $\Psi$  ofwel  $\alpha \wedge \beta$  waar maken.

En dus  $\Phi \models \alpha \wedge \beta, \Psi$



## Adequaatheidsstelling – hulpstelling 1

### Vervolg Bewijs

andere gevallen gelijkaardig



# Adequaatheidsstelling – hulpstelling 2

## Bewering 2

Stel een open tableau heeft een tak t die niet sluit. Zij  $V$  een waardering die alle **propositieletters** links op de tak t waar maakt en al deze rechts onwaar maakt. Dan geldt voor **elke formule**  $\varphi$ : als  $\varphi$  links op tak t voorkomt dan  $V(\varphi) = 1$  en als  $\varphi$  rechts op de tak t voorkomt dan  $V(\varphi) = 0$

## Bewijs

per **formule-inductie op**  $\varphi$ , en in een open tableau is elke toepasbare regel toegepast

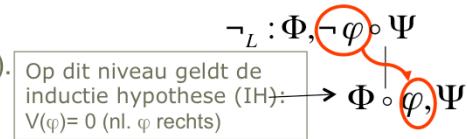
Basisstap.  $\varphi$  is atoom: per definitie van  $V$

Inductiestap. (bewijs enkel voor 2 gevallen).

- Bewijs voor  $\neg\varphi$

Als  $\neg\varphi$  links voorkomt dan moet er een keer de  $\neg_L$  regel zijn toegepast waar  $\varphi$  rechts voorkomt.

Dan is (door de inductiehypothese)  $V(\varphi) = 0$ , zodat  $V(\neg\varphi) = 1$





# Adequaatheidsstelling – hulpstelling 2

## Bewijs vervolg

Inductiestap. (bewijs 2de geval)

- Bewijs voor  $\varphi \wedge \psi$

Als  $\varphi \wedge \psi$  rechts voorkomt dan moet er een keer de  $\wedge_R$  regel zijn toegepast.

De tak t volgt in de splitsing de linker- of rechterkant.

Neem bijvoorbeeld de linkerkant met  $\varphi$  rechts.

Door IH is  $V(\varphi) = 0$  en dan is ook  $V(\varphi \wedge \psi) = 0$

Indien rechterkant: Door IH is  $V(\psi) = 0$  en dat is ook  $V(\varphi \wedge \psi) = 0$

Op dit niveau geldt de  
inductie hypothese (IH):  
 $V(\varphi) = 0$  (nl.  $\varphi$  rechts)  
of  
 $V(\psi) = 0$  (nl.  $\psi$  rechts)

$$\wedge_R : \Phi \circ \varphi \wedge \psi, \Psi \rightarrow \Phi \circ \varphi, \Psi \quad \Phi \circ \psi, \Psi$$



# Adequaatheidsstelling – bewijs deel 1

## Nu bewijs adequaatheidsstelling

$\varphi_1, \dots, \varphi_n \models \psi$  desda

er bestaat een gesloten semantisch tableau voor

$\varphi_1, \dots, \varphi_n \circ \psi$

## Bewijs

$\Leftarrow$  is een speciaal geval van bewering 1

$\Rightarrow$  stel er is geen gesloten tableau. Dan moet er een open tableau zijn.

Door bewering 2 levert elke open tak dan een tegenvoorbeeld zodat de geldigheid weerlegt wordt (en dit is een contradictie met het gegeven)

## En bijkomend:

Als één tableau voor een sequent sluit dan sluiten alle tableaus voor dit sequent



## Adequaatheidsstelling – bewijs deel 2

### En bijkomend:

Als één tableau voor een sequent sluit dan sluiten alle tableaus voor dit sequent

### Bewijs

Als één tableau sluit dan is de gevolgtrekking geldig.

Als er ook een open tableau zou zijn dan zou de gevolgtrekking niet geldig zijn.

Dit is een contradictie. Dus alle tableaus moeten sluiten

### Betekenis:

Het is **onbelangrijk in welke volgorde we de reductieregels toepassen om het tableau te construeren**, het resultaat is altijd hetzelfde.



# Volledigheidsstelling

## Volledigheidsstelling:

Voor de propositielogica geldt dat voor iedere formuleverzameling  $\Sigma$  en iedere formule  $\varphi$

$$\Sigma \vdash \varphi \text{ desda } \Sigma \models \varphi$$

- correctheid (“soundness”):

$$\Sigma \vdash \varphi \text{ impliceert } \Sigma \models \varphi$$

- Volledigheid (“completeness”):

$$\Sigma \models \varphi \text{ impliceert } \Sigma \vdash \varphi$$

Met een lege verzameling  $\Sigma$  zegt deze stelling dat de tautologieën van de propositielogica precies de stellingen van het bewijssysteem van natuurlijke deductie zijn.



## Volledigheidsstelling

Stelling (correctheid):

$$\Sigma \vdash \varphi \text{ impliceert } \Sigma \models \varphi$$

- Deze stelling wordt bewezen door van alle afleidingsregels aan te tonen dat hun resultaten geldig zijn (met inductie naar het aantal knopen in een bewijsboom).  
(Bewijs niet te kennen)



## Volledigheidsstelling

Stelling (volledigheid):

$$\Sigma \models \varphi \text{ impliceert } \Sigma \vdash \varphi$$

– Bewijs:

- Gebaseerd op maximaal consistente verzamelingen  
(een verzameling is maximaal consistent als het toevoegen van elke formule die nog niet in die verzameling zat tot een inconsistente verzameling leidt).  
(Bewijs niet te kennen)



# EINDE PROPOSITIELOGICA



Vrije Universiteit Brussel

## Logica en formele systemen Deel II: Predikaatlogica



# Predikaatlogica - Inhoud

## Inhoud

- Taal
- Semantiek
- Semantische Tableaus
- Afleidingen
- Een eenvoudige theorie
- Metatheorie



Vrije Universiteit Brussel



## Predikaatlogica: Taal



# Predikaatlogica: Taal

## – Inhoud:

- Alfabet
- Termen
- Formules
- Bereik van kwantoren
- Gebonden en vrije variabelen
- Gesloten en open Formules
- Substitutie
- Alfabetische variant



## Taal

- Predikaatlogica is een uitbreiding van de propositielogica
  - Nodig om **eigenschappen van individuen te beschrijven en erover te redeneren.**
- Voorbeeld:
  - Aanname 1: Alle mannen voetballen
  - Aanname 2: Piet is een man
  - Conclusie: Piet voetbalt
  - Niet weer te geven door propositielogica
  - Aanname 2 bevat eigenschap (**predikaat** genoemd) van een **individu** en er wordt over deze eigenschap **een conclusie gevormd**



# Taal



De taal van predikaatlogica heeft meer uitdrukkingskracht dan de taal van propositielogica en bevat daarom meer bouwstenen:

1. **Predikaatzinnen**, voorbeelden:

- Piet voetbalt
- Jan bemint Marie
- Stef is kind van Kees en Ada

zeggen telkens iets over **individu(en)**

Predikaten worden voorgesteld via **predikaatletters** (hoofdletters) **met vast aantal argumenten** (volgorde is belangrijk!)

Individuen worden voorgesteld via **constanten** (kleine letters)

Voorbeelden

1. V(p)
2. B(j,m)
3. K(s,k,a)

Alternatieve notaties

VOETBALT(PIET)	Vp
BEMINT(JAN, MARIE)	Bjm
KIND_VAN(STEF, KEESEN, ADA)	Kska



# Taal



2. **Functies:** voorgesteld door kleine letters (f, g, ...)

$$f(x,y) = 4x+3y^2$$

$m(j)$  de moeder van Jan

$m(j)$  nu te gebruiken:  $L(j, m(j))$

Jan vindt zijn moeder lief.

3. **Kwantoren:**

uitdrukkingen van hoeveelheid

$\forall$ : **universele kwantor:** 'alle'

$\exists$ : **existentiële kwantor:** 'er is minstens één'

- **Alle** mannen voetballen
- **Er is minstens één** vrouw die voetbalt.



#### 4. Variabelen:

Gebruik van kwantoren introduceert de nood aan variabelen (kleine letters  $x, y, z, \dots$ )

1.  $\forall x(Mx \rightarrow Vx)$
2.  $\exists x(Bx \wedge \neg Fx)$
3.  $\forall x(G(x) \rightarrow \exists y(G(y) \wedge (y > x))).$



- Meerdere kwantoren in één zin mogelijk
  - Voorbeelden
    - $\forall$  en  $\exists$ 
      - Voor elk getal bestaat er een groter getal
      - Iedereen heeft een moeder (voor iedereen bestaat er een moeder)
    - $\exists$  en  $\forall$ 
      - Er is een verzameling die bevat is in elke verzameling
      - Iemand is voorouder van iedereen (eva?).



### Definitie

Het **alfabet van een predikaat logische taal** bestaat uit:

- Een verzameling **C** van individuele constanten:  $a, b, c, \dots, a_1, a_2, a_3, \dots$
  - Een verzameling **P** van predikaatletters:  $P, Q, R, \dots, P_1, P_2, P_3, \dots$
  - Een verzameling **F** van functieletters:  $f, g, h, \dots, f_1, f_2, f_3, \dots$
  - De logische symbolen:  $\neg, \wedge, \vee, \rightarrow, \leftrightarrow, \forall, \exists$
  - Een aantal individuele variabelen:  $u, v, w, x, y, z, x_1, x_2, x_3, \dots$
  - De hulpsymbolen: ) en (
- **Plaatsigheid** (aantal argumenten) voor functieletters en predikaatletters ook wel via een index:  $f^3abc$  of  $f^3(a, b, c)$ :  $f$  is 3-plaatsig
- Merk op: propositionele letters kunnen beschouwd worden als 0-plaatsige predikaatletters.



# Syntaxis – definitie term



## – Termen

- duiden individuele objecten aan

### *Definitie*

De **termen** in de predikaatlogica zijn als volgt gedefinieerd (inductieve definitie):

- individuele variabelen en constanten zijn termen;
- als  $f$  een  $k$ -plaatsige functieletter is en  $t_1, \dots, t_k$  zijn termen, dan is  $f(t_1, \dots, t_k)$  ook een term;
- Niets anders is een term.

– Voorbeeld:  $f^3(g^2(x,h^1(y)),a,g^2(a,y))$



## Opmerking notatie



- Prefix versus infixnotatie
  - Prefixnotatie
    - Functiesymbool vóór de argumenten
    - voorbeelden:  $f(x,y)$ ,  $+(x,y)$ ,  $\cdot(x,y)$
  - Infixnotatie
    - Meestal bij 2-plaatsige functies
    - Functiesymbool tussen de argumenten
    - voorbeelden:  $x + y$  en  $x \cdot y$
- Meestal wordt in logica prefix notatie gebruikt

Geef boom (zie p 92)



# Syntaxis – definitie formule



## Definitie

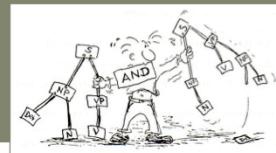
De **formules** in de predikaatlogica zijn als volgt gedefinieerd (inductieve definitie):

- als  $P$  een  $k$ -plaatsige predikaatletter is en  $t_1, \dots, t_k$  zijn termen dan is  $P(t_1, \dots, t_k)$  een formule;
- als  $\varphi$  en  $\psi$  formules zijn, dan zijn ook  $\neg\varphi$ ,  $(\varphi \wedge \psi)$ ,  $(\varphi \vee \psi)$ ,  $(\varphi \rightarrow \psi)$  en  $(\varphi \leftrightarrow \psi)$  formules;
- als  $\varphi$  een formule is en  $x$  een individuele variabele, dan zijn  $\forall x \varphi$  en  $\exists x \varphi$  ook formules;
- niets anders is een formule

Een formule van de vorm  $P(t_1, \dots, t_k)$  heet een **atomaire formule** of **atoom**.



# Syntaxis – voorbeelden



- Voorbeelden atomaire formules
  - $P^2(x, a)$
  - $P^2(f^2(x, y), g^1(a))$
- Voorbeelden van formules
  - $\forall x (A^1(a) \rightarrow \forall y (R^2(x, y) \rightarrow A^1(y)))$
  - $\forall x \exists y \forall z (R^2(z, y) \leftrightarrow \neg Q^2(y, x))$
- Zijn geen formules
  - $P(\neg x)$
  - $A(x)\forall x$
  - $\forall x \exists R^2(x, y)$
- We houden in het algemeen de notatie zo eenvoudig mogelijk
  - Zo weinig mogelijk boven indices
  - Zo weinig mogelijk haakjes.



## Syntaxis – voorbeelden



- Betekenisvolle voorbeelden
  - Niemand kent iedereen
    - $\neg \exists x \forall y Kxy$
  - Wie iemand kent, kent iedereen
    - $\forall x (\exists y Kxy \rightarrow \forall z Kxz)$
  - Wie alleen zichzelf kent, wordt door niemand gekend
    - $\forall x (\forall y (Kxy \leftrightarrow x=y) \rightarrow \neg \exists z Kzx)$



## Kracht van de taal

- Predikaatlogica is geschikt om wiskundige beweringen te formaliseren, bijv. uit rekenkunde, meetkunde, verzamelingenleer,  
...  
– Zie boek voor voorbeelden
- Predikaatlogica wordt ook gebruikt als basis voor programmeertalen  
– De voorbeeldtaal hiervan is PROLOG.

Voorbeelden uit de wiskunde niet te kennen

Wel weten dat PROLOG een programmeertaal gebaseerd op logica is.



# Bereik van kwantoren



- Zijn de volgende formules gelijkwaardig?

$$(\exists x Ax \wedge \forall y Bxy) \quad \text{en} \quad \exists x (Ax \wedge \forall y Bxy)$$

- Het **bereik van een kwantor** is die subformule waarvóór die kwantor is gevoegd tijdens de constructie van die formule.



# Bereik van kwantoren



$$\varphi = \forall x (Ax \rightarrow \exists y (Rxy \wedge \forall x Sxy))$$

$$\begin{array}{c} \forall x (Ax \rightarrow \exists y (Rxy \wedge \forall x Sxy)) \\ | \\ Ax \rightarrow \exists y (Rxy \wedge \forall x Sxy) \\ | \\ Ax \quad \exists y (Rxy \wedge \forall x Sxy) \\ | \\ Rxy \wedge \forall x Sxy \\ | \\ Rxy \quad \forall x Sxy \\ | \\ Sxy \end{array}$$



# Bereik van kwantoren



– Illustratie

$$\varphi = \forall x (Ax \rightarrow \exists y (Rxy \wedge \forall z Sxz))$$

(1)                          (2)                          (3)

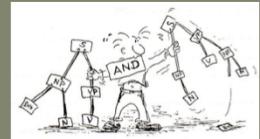
Bereik van  $\forall x$  (1)

Bereik van  $\exists y$  (2)

Bereik van  $\forall z$  (3)



## Vrije en gebonden variabelen



Def. Een voorkomen van een variabele  $x$  heet **vrij** als het niet in het bereik van een kwantor  $\forall x$  of  $\exists x$  ligt

Def. Een kwantor  $\forall x$  of  $\exists x$  **bindt** de voorkomens van variabelen  $x$  in zijn bereik, voor zover die nog vrij zijn. Deze voorkomens heten dan **gebonden**

Voorbeeld van gebonden variabelen:

$$\varphi = \forall x (Ax \rightarrow \exists y (Rxy \wedge \forall x Sxy))$$




# Vrije en gebonden variabelen



- De vrije variabelen van een formule kunnen beschouwd worden als parameters (zoals in programmeren)

Def. Een **zin** of een **gesloten formule** is een formule zonder vrije variabelen

– Voorbeelden:

- $Pa$
- $\exists x Px$
- $\forall x \exists y (Rxy \vee \exists z (Rxz \wedge Rzy))$

Def. Een formule met vrije variabelen heet een **open formule**

– Voorbeelden

- $Rax$
- $\exists x Sxy$



### Definitie

#### Substitutie

- Als  $t, t'$  termen zijn,  $x$  een variabele, dan is  $[t/x]t'$  de term die ontstaat door **elk voorkomen van  $x$**  in  $t'$  te vervangen door  $t$ .
- Als  $\varphi$  een formule is,  $t$  een term en  $x$  een variabele, dan is  $[t/x]\varphi$  de formule die ontstaat door **elk voorkomen van  $x$  als vrije variabele** in  $\varphi$  te vervangen door  $t$ .
  - $[t/x]\varphi$  wordt ook wel een instantie van  $\varphi$  genoemd
- Voorbeelden
  - $[y/x](\forall x(Rx \vee Sx) \vee Pxx) = \forall x(Rx \vee Sx) \vee Pyy$
  - $[f(a,b)/z]\exists x(Px \rightarrow Ry) = \exists x(Px \rightarrow Ryf(a,b))$
  - $[y/x](\exists y(y < x)) = \exists y(y < y)$

Dus enkel vrije variabelen worden vervangen!!



## “Veilige” substituties



- De substitutie van  $t$  voor  $x$  is wel altijd gedefinieerd maar levert soms ongewenste resultaten op (zie vorig voorbeeld)

### Definitie

Een term  $t$  heet **vrij voor  $x$  in  $\varphi$**   
als in  $[t/x]\varphi$  **geen** variabele van  $t$  gebonden wordt.

- Voorbeelden
  - $f^2zy$  is niet vrij voor  $x$  in  $\exists y Rxy$ , maar wel in  $Rxy$  of in  $\exists u Rxu$ .



## Alfabetische variant



- Een **alfabetische variant van een formule** ontstaat door gebonden variabelen door nieuwe variabelen te vervangen.
  - Intuïtief gezien verandert hierdoor niets
  - $\exists y (y < x)$  heeft als alfabetische variant:  $\exists z (z < x)$

Wordt gebruikt om substitutie veilig te kunnen uitvoeren:

Zie vorig voorbeeld:

$f^2 z y$  is niet vrij voor  $x$  in  $\exists y R x y$

Maar wel in  $\exists u R x u$  (gebonden  $y$  vervangen door  $u$ ).

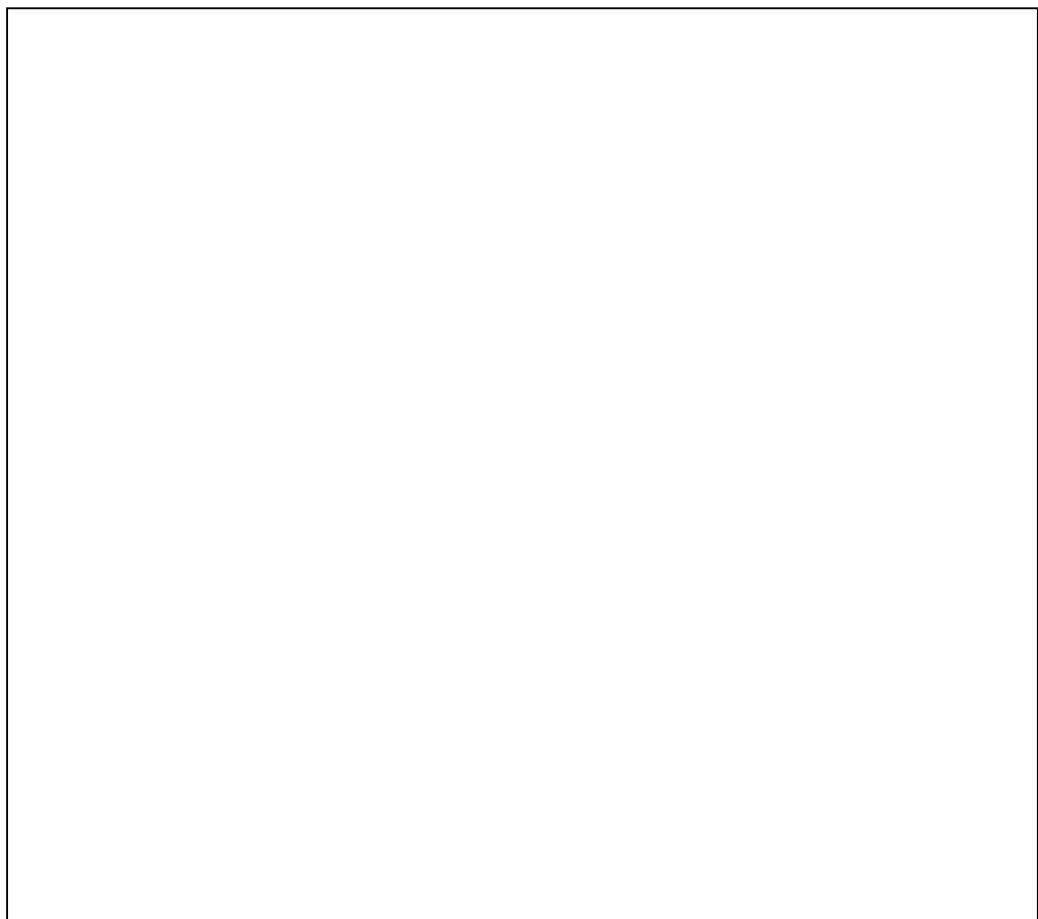
Enkel gebonden variabelen mogen vervangen worden, de vrije variabelen dus niet!



Vrije Universiteit Brussel



## Predikaatlogica: Semantiek





# Semantiek

## – Inhoud:

- Principe
- Situatie of structuur
- Interpretatiefunctie
- Model
- Bedeling
- Waardering
- Model van een formule
- Gelijkheid tussen termen, universeel geldig, logisch equivalent
- Geldig gevolg
- Theorie en Modelverzameling



# Semantiek - principe



- Drie aspecten zijn belangrijk bij de bestudering van de semantiek van predikaatlogica:



De interpretatiefunctie geeft betekenis aan de bouwstenen van de taal (predikaatletters, functieletters, ....).



## Semantiek - principe



- Alle 3 de aspecten zijn essentieel
  - vergelijking met natuurlijke taal
    - Stel tekst gekend en woordenboek (interpretatiefunctie), maar niet de situatie
      - We kunnen niet weten of wat gezegd wordt waar is
    - Stel tekst gekend en de situatie, maar geen woordenboek (interpretatiefunctie)
      - We kunnen niet weten of wat gezegd wordt waar is
    - Stel een woordenboek (de interpretatiefunctie) en de situatie, maar niet de tekst
      - We weten niet wat er gezegd werd.



## Semantiek: Structuur



- Structuur is te vergelijken met een stukje werkelijkheid
- Een **structuur** bestaat uit een **domein** waarop **relaties** en **operaties** gedefinieerd zijn.
  - **Voorbeeld:**  
domein: de natuurlijke getallen IN  
relaties: < , =  
operaties: + , \*
- **Informeel:**
  - Relaties tussen de objecten in het domein corresponderen met beweringen (vb.  $5 < 10$ )
  - Operaties op objecten leveren andere objecten op (vb.  $8 + 2$  is 10)



## Semantiek: Structuur - definitie



### *Definitie*

Een **structuur  $D$**  is een drietal  $\langle D, R, O \rangle$  bestaande uit een **niet-lege** verzameling  $D$  (het domein), een verzameling  $R$  van relaties **op  $D$**  en een verzameling  $O$  van operaties **op  $D$** .



## Intermezzo: Speciale structuren

Speciale structuren:

- **Relationele structuur:** enkel relaties, geen operaties  
vb.: lineaire ordeningen, bomen, grafs, ...

Voorbeeld:

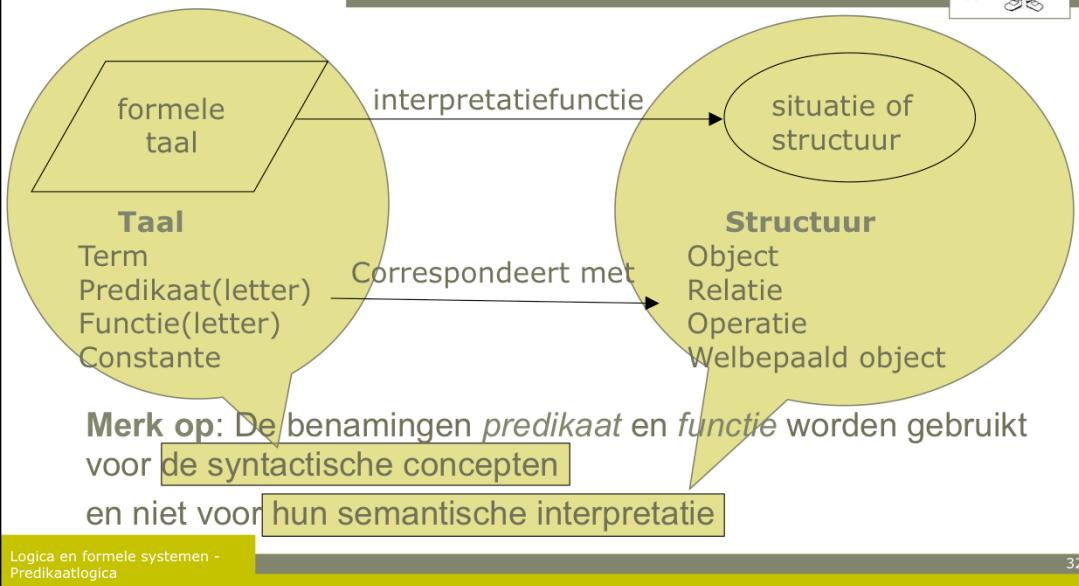
$D = \{1, 2, 3, 4, 5\}$  met de relatie  $R$  'kleiner dan'

$R = \{(1,2), (1,3), (1,4), (1,5), (2,3), (2,4), (2,5), (3,4), (3,5), (4,5)\}$

- **Operationele structuur:** domein met enkel operaties  
vb.: groepen, ringen (wiskunde).



## Semantiek: principe vervolg





# Semantiek: interpretatiefunctie



- Om een formule te kunnen interpreteren is er een interpretatie van variabelen, constanten, predikaatletters en functieletters nodig

## Definitie

Laat  $D = \langle D, R, O \rangle$  een structuur zijn.

Een **interpretatiefunctie** / kent aan

- elke individuele constante  $c$  uit de predikaat logische taal een welbepaald object uit  $D$  toe via een nul-plaatsige operatie,  $I(c) \in O$
- elke predikaatletter  $P$  een relatie uit  $R$  van dezelfde plaatsigheid toe,  $I(P) \in R$
- elke functieletter  $f$  een operatie uit  $O$  van dezelfde plaatsigheid toe,  $I(f) \in O()$

Interpreteren: betekenis geven

Een nul-plaatsige operatie geeft steeds een object uit het domein terug, zodoende wordt elke constante afgebeeld op een welbepaald object uit het domein

**Definitie**

Een paar  $(D, I)$  met  $D$  een structuur en  $I$  een interpretatiefunctie heet een **model**

Een (on)eindig model is een model met een (on)eindig domein.

– Voorbeeld:

$$D = \langle \mathbb{IN}, \{\geq\}, \{0\} \rangle$$

$$I(c_1) = 0$$

$$I(R) = \geq$$

$$I(\forall x R(x, c_1)) = \forall x (x \geq 0)$$

Dus in dit model is de zin  $\forall x R(x, c_1)$  waar

Let op: dit is verschillend van hoe een model gedefinieerd is in propositie logica



### Definitie

Een **bedeling**  $b$  is een functie die aan **elke variabele**  $x$  een object uit het domein toekent, dus  $b(x) \in D$ .

- Voorbeeld: Model  $M = (\langle D, \mathcal{R}, \mathcal{O} \rangle, I)$  met  
 $D = \mathbb{IN}$ ,  $\mathcal{R} = \{<\}$ ,  $\mathcal{O} = \{0, +, \cdot\}$   
 $I: I(P) = <$ ,  $I(f) = +$ ,  $I(g) = \cdot$ ,  $I(a) = 0$   
 $b: b(x_i) = i \quad (i = 1, 2, 3, \dots)$

$$Px_1x_2 \wedge Pf(a, x_9)g(x_5, x_9) \quad \xrightarrow{b, I} 1 < 2 \text{ en } 9 < 45$$

Opm: bedeling is gedefinieerd voor alle variabelen, niet enkel voor vrije variabelen!!



- Notatie:

$\langle \mathbf{IN}, \{<, =\}, \{0, 1, +, \cdot\} \rangle$  ook als  $\langle \mathbf{IN}, <, =, 0, 1, +, \cdot \rangle$  wanneer duidelijk is wat de relaties zijn en wat de operaties zijn.  
Zo ook:  $P$  i.p.v.  $I(P)$  en  $f$  i.p.v.  $I(f)$

$b[x \mapsto d]$  is de bedeling  $b$  waarbij  $d$  aan de variabele  $x$  wordt toebedeeld.

- Vben:  $b: b(x_i) = i \ (i = 1, 2, 3, \dots)$   
 $b[x_1 \mapsto 10](x_2) = 2$   
 $b[x_1 \mapsto 10](x_1) = 10$



### Definitie

Laat  $M = (D, I)$  een model zijn en  $b$  een bedeling.

Dan is de **semantische waardering**  $V_{M,b}$  van **termen** als volgt gedefinieerd:

- $V_{M,b}(x) = b(x)$  voor variabelen  $x$
- $V_{M,b}(a) = I(a)$  voor constanten  $a$
- $V_{M,b}(f(t_1, \dots, t_k)) = I(f)(V_{M,b}(t_1), \dots, V_{M,b}(t_k))$



### Voorbeeld

Laat  $M$  een model zijn met  $D = \langle \mathbb{IN}, 0, + \rangle$  en  $I(f) = '+'$ ,  $I(a) = 0$   
 $b$  een bedeling waarbij  $b(x) = 1$

Dan geldt:

$$\begin{aligned}V_{M,b}(f(a,x)) &= I(f)(V_{M,b}(a), V_{M,b}(x)) \\I(f)(V_{M,b}(a), V_{M,b}(x)) &= I(f)(I(a), b(x)) \\I(f)(I(a), b(x)) &= +(0,1) = 1\end{aligned}$$



Net als in propositielogica is de interpretatie (semantiek) van een formule een waarheidswaarde: waar (0) of onwaar(1)

### Definitie

Laat  $M = (D, I)$  een model zijn en  $b$  een bedeling.

De **waarheidswaarden van formules** zijn als volgt gedefinieerd:

- $V_{M,b}(P(t_1, \dots, t_m)) = 1$  desda  $I(P)(V_{M,b}(t_1), \dots, V_{M,b}(t_m))$  geldt
- $V_{M,b}(\neg\varphi) = 1$  desda  $V_{M,b}(\varphi) = 0$   
idem als in propositielogica voor  $\varphi \wedge \psi$ ,  $\varphi \vee \psi$ ,  $\varphi \rightarrow \psi$  en  $\varphi \leftrightarrow \psi$ .
- $V_{M,b}(\exists x \varphi) = 1$  desda er is een  $d \in D$  zodat  $V_{M,b[x \mapsto d]}(\varphi) = 1$
- $V_{M,b}(\forall x \varphi) = 1$  desda voor alle  $d \in D$  geldt  $V_{M,b[x \mapsto d]}(\varphi) = 1$



- Een **atomaire formule** is waar in een structuur, als het feit dat wordt uitgedrukt inderdaad **waar is in de structuur**
  - Voorbeeld:  $I(P) = <$ ,  $b(x)=2$  en  $b(y)=7$  dan is  $Pxy$  waar in  $\langle IN, < \rangle$   
nl,  $2 < 7$  geldt in **IN**
- Eén formule  $\varphi$  kan in verschillende structuren heel verschillende beweringen uitdrukken.  
  
Bij gegeven  $\varphi$  en één structuur **D** kunnen verschillende interpretatiefuncties aan  $\varphi$  een andere waarheidswaarde geven
  - Voorbeeld:  $\forall x \forall y (f(x,y) = f(y,x))$  is waar op **Q** en **IN** zowel met  $I(f) = +$  als met  $I(f) = \cdot$  maar onwaar met  $I(f) = -$



## Notatie

- |                                       |   |
|---------------------------------------|---|
| – $V_{M,b}(\varphi) = 1$              | $\varphi$ is waar in $M$ onder $b$                      |
| – $V(\varphi)$ ipv $V_{M,b}(\varphi)$ | als geen verwarring mogelijk                            |
| – $V_{M,b}(\varphi) = 1$              | $M,b \models \varphi$ of $M \models \varphi[b]$         |
| – $V_{M,b}(\varphi) = 0$              | $M,b \not\models \varphi$ of $M \not\models \varphi[b]$ |

### Definitie

Een paar  $M=(D,I)$  met structuur  $D$  en interpretatiefunctie  $I$  heet een model van een formule  $\varphi$  als voor iedere bedeling  $b$  geldt:

$$V_{M,b}(\varphi) = 1$$



## Semantiek - voorbeeld



### Voorbeeld

$M$  model met  $D = \langle Q, < \rangle$  en  $I(R) = '<'.$

$b$  een bedeling waarbij  $b(x_1) = 4$

Dan  $V_{M,b}(\forall y(Rx_1y \rightarrow \exists z(Rx_1z \wedge Rzy))) = 1$

desda voor alle  $q \in Q$ :  $V_{M,b[y \mapsto q]}(Rx_1y \rightarrow \exists z(Rx_1z \wedge Rzy)) = 1$

desda voor alle  $q \in Q$ : als  $V_{M,b[y \mapsto q]}(Rx_1y) = 1$  dan  $V_{M,b[y \mapsto q]}(\exists z(Rx_1z \wedge Rzy)) = 1$

desda voor alle  $q \in Q$ : als  $4 < q$ ,

dan is er een  $q' \in Q$ :  $V_{M,b[y \mapsto q, z \mapsto q']}(Rx_1z \wedge Rzy) = 1$

en  $V_{M,b[y \mapsto q, z \mapsto q']}(Rzy) = 1$

desda voor alle  $q \in Q$  als  $4 < q$ ,

dan is er een  $q' \in Q$  met  $4 < q'$  en  $q' < q$



- De **gelijkheidsrelatie** ('=') is niet gedefinieerd (als relatie) in de taal:

Indien nodig: explicet te definiëren; alsook de semantiek ervan.

Voorbeeld definitie gelijkheid tussen termen:

$$V_{M,b}(t_1 = t_2) = 1 \text{ desda } V_{M,b}(t_1) = V_{M,b}(t_2)$$

Gelijkheid hier is gebaseerd op de identiteit tussen objecten in het domein



# Semantiek – eigenschappen



## Eigenschap van de waarheidsfunctie:

Waarheidswaarde van een formule hangt af van de structuur  $D$ , de interpretatiefunctie  $I$  en van het effect van de bedeling  $b$  op de **vrije variabelen** in die formule (en **dus niet van de bedeling van de gebonden variabelen**).

Dit is gebaseerd op de volgende bewering:

### Bewering:

Als een formule  $\varphi$  vrije variabelen  $x_1, \dots, x_k$  bevat en er zijn 2 bedelingen  $b_1$  en  $b_2$  met  $b_1(x_i) = b_2(x_i)$ , voor  $i = 1, \dots, k$ , dan geldt  $V_{M,b1}(\varphi) = V_{M,b2}(\varphi)$



## Semantiek – eigenschappen



- Gevolg:
  - Voor zinnen (gesloten formules) zijn er geen vrije variabelen, dus doet de bedeling er niet toe.
    - Voor zinnen spreken we dus over waarheid en onwaarheid in een model.



## Substitutie – eigenschappen



### Bewering

Substitutie

Voor alle termen  $t, t'$  geldt:

$$V_{M,b}([t/x]t') = V_{M,b[x \mapsto V_{M,b}(t)]}(t')$$

Links: eerst substitueren en dan de waarde van  $t'$  bepalen

Rechts: waarde van  $t'$  berekenen en we bedelen de waardering van  $t$  aan  $x$

Dus wat betreft de waardering is substitutie in een term hetzelfde als substitutie in de bedeling.



## Substitutie – eigenschappen



$$V_{M,b}([t/x]t') = V_{M,b[x \mapsto V_{M,b}(t)]}(t')$$

Voorbeeld:

Als  $t' = f(x,y)$  en  $t = a$  dan

$$\begin{aligned} V_{M,b}([a/x]f(x,y)) &= V_{M,b}(f(a,y)) \\ &= I(f)(V_{M,b}(a), V_{M,b}(y)) \\ &= I(f)(I(a), b(y)) \end{aligned}$$

Anderzijds:

$$\begin{aligned} V_{M,b[x \mapsto V_{M,b}(a)]}(f(x,y)) &= I(f)(V_{M,b[x \mapsto V_{M,b}(a)]}(x), V_{M,b[x \mapsto V_{M,b}(a)]}(y)) \\ &= I(f)(V_{M,b}(a), b(y)) \\ &= I(f)(I(a), b(y)) \end{aligned}$$

Bewijs later.



# Geldig gevolg



## Definitie

### Geldig gevolg:

Laat  $\Sigma$  een verzameling formules zijn en  $\psi$  een formule,  
dan  $\psi$  volgt uit  $\Sigma$ ,  $\Sigma \models \psi$ , desda:

voor elk model  $M$  en elke bedeling  $b$  geldt:  
als voor elke  $\varphi \in \Sigma$  geldt dat  $V_{M,b}(\varphi) = 1$   
dan ook  $V_{M,b}(\psi) = 1$

- Opmerking: oneindig veel mogelijkheden voor  $M$  en  $b$  !!



**Def.** Een formule  $\psi$  heet **universeel geldig** als  $\models \psi$

- Intuïtief: Een universeel geldige formule is **waar in alle modellen  $M$  en onder iedere bedeling  $b$**

**Def.** Twee formules  $\varphi$  en  $\psi$  heten **logisch equivalent** als  
 $\models \varphi \leftrightarrow \psi$

Voorbeelden:

$$\begin{aligned} & \forall x (Rx \rightarrow Px), \exists x Rx \models \exists x Px \\ & \models Ta \rightarrow \exists x Tx \\ & \models \forall x Rx \leftrightarrow \neg \exists x \neg Rx \end{aligned}$$

**Definitie**

$\{\varphi \mid \varphi \text{ is een zin en } V_M(\varphi) = 1\}$  is een **theorie voor een model  $M$**   
**Notatie  $\text{Th}(M)$**

- Intuïtief: De verzameling van ware zinnen is een theorie voor  $M$
- We kunnen een theorie ook weergeven door axioma's

**Definitie****Axiomaverzameling voor een model  $M$** 

Een formuleverzameling  $\Sigma$  axiomatiseert een theorie  $\text{Th}(M)$  als voor alle zinnen  $\varphi$  geldt:  $\varphi \in \text{Th}(M)$  desda  $\Sigma \models \varphi$

- Een goede axiomatisering geeft de essentiële kenmerken van het model weer.



### Definitie

Modelverzameling voor de zin  $\varphi$ ,  $MOD(\varphi)$ , is

$$\{M \mid V_M(\varphi) = 1\}$$

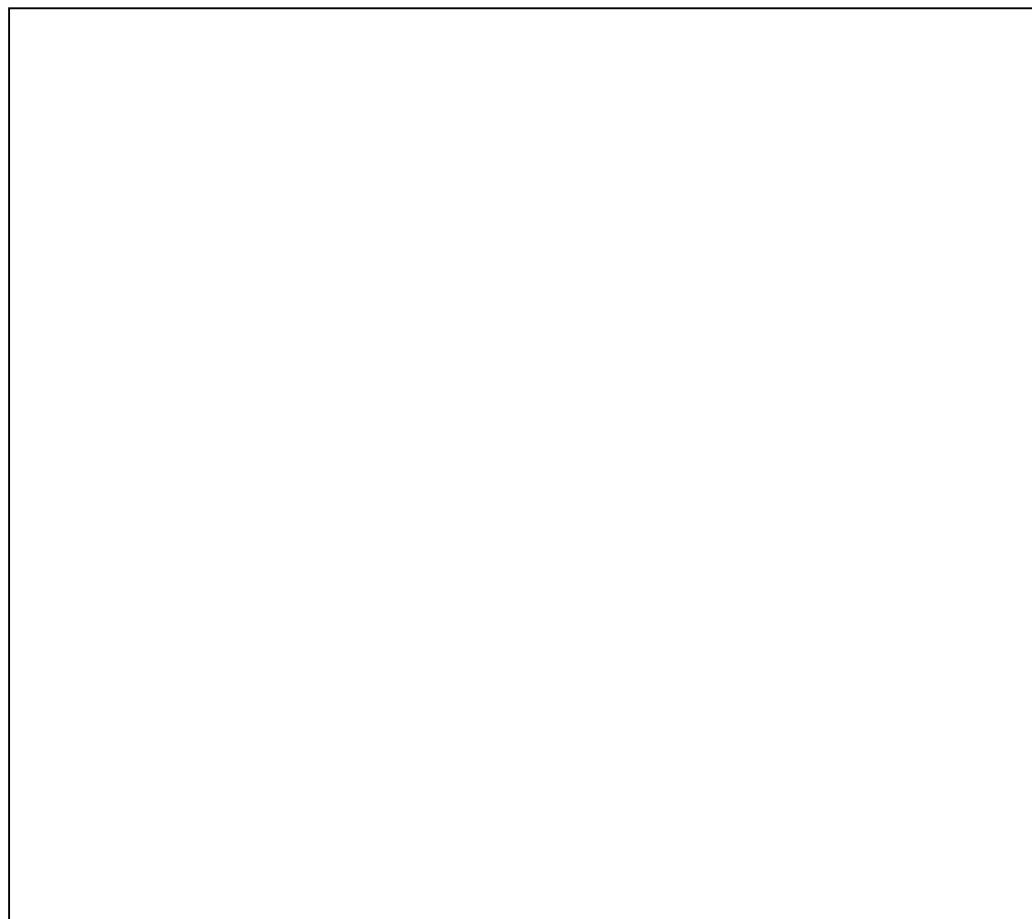
Modelverzameling voor de verzameling zinnen  $\Sigma$ ,  
 $MOD(\Sigma)$ , is  $\{M \mid V_M(\varphi) = 1 \text{ voor alle } \varphi \in \Sigma\}$

- MOD is verzameling van modellen die de zin, resp. een verzameling zinnen, waar maakt.



Vrije Universiteit Brussel

## Predikaatlogica: Semantische tableaus





## Semantische tableaus

- Zoals in de propositielogica kunnen we semantische tableaus gebruiken om de geldigheid van een gevolgtrekking te testen
- Hoofdidee:
  - zoeken van een tegenvoorbeeld voor  $\varphi_1, \dots, \varphi_n \circ \psi$
- Let op: tegenvoorbeeld in predikaatlogica is
  - Er bestaat een **model** en een **bedeling** die  $\varphi_1, \dots, \varphi_n$  waar maakt en  $\psi$  onwaar
    - Dus structuur (domain), interpretatiefunctie en bedeling nodig.



## Semantische tableaus

- Regels en techniek van de propositielogica zijn uitgebreid naar de predikaatlogica:
  - Bijkomende **reductieregels voor de kwantoren**  $\forall$  en  $\exists$
  - Gaandeweg **construeren van een domein**  $D$
  - Bijhouden van **de interpretatiefunctie**  $I$  en **de bedeling**  $b$
  - De reductieregels voor de connectieven blijven van kracht
- Beperking:
  - geen functieletters in de formules
  - Gevolgtrekkingen zonder constanten en zonder vrije variabelen (anders veel ingewikkelder).



# Semantische tableaus

- Voorbeeld

- Geldige gevolgtrekking:

$$\forall x(Ax \rightarrow Bx), \forall x(Bx \rightarrow Cx) / \forall x(Ax \rightarrow Cx)$$
$$\varphi_1 = Ax \rightarrow Bx, \varphi_2 = Bx \rightarrow Cx$$

Universele formules moeten waar zijn voor elk element in het domein, dus ook voor  $d_1$

Nodig: Minstens 1 element in het domein die de bewering onwaar maakt

$$\forall x\varphi_1, \forall x\varphi_2 \circ \forall x(Ax \rightarrow Cx)$$

$$\begin{array}{c} \forall x\varphi_1, \forall x\varphi_2 \circ Ad_1 \rightarrow Cd_1 \\ | \quad | \\ \forall x\varphi_1, \forall x\varphi_2, Ad_1 \circ Cd_1 \end{array} \quad (1) D = \{d_1\}$$

(1)  $D = \{d_1\}$

$$\begin{array}{c} \forall x\varphi_1, \forall x\varphi_2, Ad_1 \rightarrow Bd_1, Ad_1 \circ Cd_1 \\ | \\ \forall x\varphi_1, \forall x\varphi_2, Ad_1 \rightarrow Bd_1 \end{array} \quad (1) D = \{d_1\}$$

(2)  $\forall x\varphi_1 : \{d_1\}$

Constructie van het domein

Eerst de voor-alles rechts elimineren omdat er nog geen domein geconstrueerd is (of het domein is leeg). Voor-alles rechts is dus gemakkelijker



## Semantische tableaus

$$\begin{array}{c} \boxed{\forall x\varphi_1, \forall x\varphi_2, Ad_1 \rightarrow Bd_1, Ad_1 \circ Cd_1} \quad (1) D = \{d_1\} \\ \qquad \qquad \qquad \downarrow \rightarrow_L \quad (2) \forall x\varphi_1 : \{d_1\} \\ \boxed{\forall x\varphi_1, \forall x\varphi_2, Bd_1, Ad_1 \circ Cd_1} \quad (1) D = \{d_1\} \\ \qquad \qquad \qquad \downarrow \forall_L \quad (2) \forall x\varphi_1 : \{d_1\} \end{array} = \begin{array}{c} \boxed{\forall x\varphi_1, \forall x\varphi_2, Ad_1 \circ Ad_1, Cd_1} \quad (1) D = \{d_1\} \\ \qquad \qquad \qquad \downarrow (2) \forall x\varphi_1 : \{d_1\} \\ \boxed{\forall x\varphi_1, \forall x\varphi_2, Bd_1 \rightarrow Cd_1, Bd_1, Ad_1 \circ Cd_1} \quad (1) D = \{d_1\} \\ \qquad \qquad \qquad \downarrow \rightarrow_L \quad (2) \forall x\varphi_1 : \{d_1\} \\ \qquad \qquad \qquad \qquad \downarrow (3) \forall x\varphi_2 : \{d_1\} \\ \boxed{\forall x\varphi_1, \forall x\varphi_2, Cd_1, Bd_1, Ad_1 \circ Cd_1} \quad (1) D = \{d_1\} \\ \qquad \qquad \qquad \downarrow (2) \forall x\varphi_1 : \{d_1\} \\ \qquad \qquad \qquad \qquad \downarrow (3) \forall x\varphi_2 : \{d_1\} \end{array} = \begin{array}{c} \boxed{\forall x\varphi_1, \forall x\varphi_2, Bd_1, Ad_1 \circ Bd_1, Cd_1} \quad (1) D = \{d_1\} \\ \qquad \qquad \qquad \downarrow (2) \forall x\varphi_1 : \{d_1\} \\ \qquad \qquad \qquad \qquad \downarrow (3) \forall x\varphi_2 : \{d_1\} \end{array}$$

Tableau is gesloten!



- Extra reductieregels:

$$\begin{array}{ll} \forall_R : \Phi \circ \forall x \varphi, \Psi & (1) \quad D = \{d_1, \dots, d_k\} \\ | & (2) \quad \dots\dots \\ \Phi \circ [d_{k+1}/x] \varphi, \Psi & (1) \quad D = \{d_1, \dots, d_k, d_{k+1}\} \\ & (2) \quad \dots \text{idem} \dots \end{array}$$

Nieuw element

- Om  $\forall x \varphi$  onwaar te maken, moet **er minstens 1 object** in  $D$  zijn zodat  $[d/x] \varphi$  onwaar is. We voeren daarom  $d_{k+1}$  in
- Achtereenvolgens toepassen van  $\forall_R$  zorgt dat het domein langzamerhand wordt opgebouwd.

 $\forall_L :$ 

$$\Phi, \forall x \varphi \circ \Psi \quad \left| \begin{array}{l} (1) \quad \{d_1, \dots, d_k\} \\ (2) \quad \dots\dots \end{array} \right.$$
$$\Phi, \forall x \varphi, [d_1/x]\varphi, \dots, [d_k/x]\varphi \circ \Psi \quad \left| \begin{array}{l} (1) \quad \{d_1, \dots, d_k\} \\ (2) \quad \forall x \varphi : \{d_1, \dots, d_k\} \\ \dots \text{idem} \dots \end{array} \right.$$

- Om  $\forall x \varphi$  waar te maken, moet  $[d/x] \varphi$  waar zijn voor alle objecten die tijdens de constructie in het domein terecht komen.
  - Dit vereist het “invullen” voor elk object uit het geconstrueerde domein.
- Moet ook terug gebeuren als er later nog nieuwe elementen aan het domein worden toegevoegd !!!



# Semantische tableaus

$$\begin{array}{lll} \exists_L : & \Phi, \exists x \varphi \circ \Psi & (1) \quad \{d_1, \dots, d_k\} \\ & | & (2) \quad \dots\dots \\ & \Phi, [d_{k+1}/x] \varphi \circ \Psi & (1) \quad \{d_1, \dots, d_k, d_{k+1}\} \\ & & (2) \quad \dots \text{idem} \dots \end{array}$$

$$\begin{array}{lll} \exists_R : & \Phi \circ \exists x \varphi, \Psi & (1) \quad \{d_1, \dots, d_k\} \\ & | & (2) \quad \dots\dots \\ & \Phi \circ [d_1/x] \varphi, \dots, [d_k/x] \varphi, \exists x \varphi, \Psi & (1) \quad \{d_1, \dots, d_k\} \\ & & (2) \quad \exists x \varphi : \{d_1, \dots, d_k\} \\ & & \dots \text{idem} \dots \end{array}$$



## Semantische tableaus -Vb 9.2

$$\varphi \equiv Ax \rightarrow \forall y By$$

$$\Psi \equiv \forall x \forall y (Ax \rightarrow By)$$

$$\forall x \varphi \circ \forall x \forall y (Ax \rightarrow By)$$

$$\forall x \varphi \circ \forall y^R (Ad_1 \rightarrow By) \quad (1) D = \{d_1\}$$

$$\forall x \varphi \circ Ad_1^R \rightarrow Bd_2 \quad (1) D = \{d_1, d_2\}$$

$$\forall x \varphi, Ad_1 \circ Bd_2 \quad (1) D = \{d_1, d_2\}$$

\*:(1)  $D = \{d_1, d_2\}$

(2)  $\forall x \varphi : \{d_1, d_2\}$

$$\forall x \varphi, Ad_1 \rightarrow \forall y By, Ad_2 \rightarrow \forall y By, Ad_1 \circ Bd_2 \quad *$$

$$\forall x \varphi, \forall y By, Ad_2 \rightarrow \forall y By, Ad_1 \circ Bd_2 \quad *$$

$$\forall x \varphi, Ad_2 \rightarrow \forall y By, Ad_1 \circ Ad_1, Bd_2 \quad *$$

$$\forall x \varphi, \forall y By, Ad_1 \circ Bd_2 \quad * \quad \forall x \varphi, \forall y By, Ad_1 \circ Ad_2, Bd_2 \quad *$$

$$\forall x \varphi, \forall y By, Bd_1, Bd_2, Ad_1 \circ Bd_2 \quad ** \quad \forall x \varphi, \forall y By, Bd_1, Bd_2, Ad_1 \circ Ad_2, Bd_2 \quad **$$

\*\*:(1)  $D = \{d_1, d_2\}$   
(2)  $\forall x \varphi : \{d_1, d_2\}$   
 $\forall y By : \{d_1, d_2\}$

Dit tableau sluit.



$$\varphi_1 \equiv Ax \wedge Bx$$

$$\varphi_2 \equiv Bx \wedge Cx$$

## Semantische tableaus -Vb 9.4

\*:(1)  $D = \{d_1\}$   
(2)  $\exists x \varphi_1 : \{d_1\}$

$\neg \exists x(Ax \wedge Bx), \neg \exists x(Bx \wedge Cx) \circ \neg \exists x(Ax \wedge Cx)$	$  \neg_L, \neg_L, \neg_R$	
$\exists x(Ax \wedge Cx) \circ \exists x(Ax \wedge Bx), \exists x(Bx \wedge Cx)$		
$Ad_1 \wedge Cd_1 \circ \exists x(Ax \wedge Bx), \exists x(Bx \wedge Cx) \quad D = \{d_1\}$	$\wedge_L$	
$Ad_1, Cd_1 \circ \exists x(Ax \wedge Bx), \exists x(Bx \wedge Cx) \quad D = \{d_1\}$	$\exists_L$	
$Ad_1, Cd_1 \circ Ad_1 \wedge Bd_1, \exists x(Ax \wedge Bx), \exists x(Bx \wedge Cx) \quad *$	$\wedge_R$	
$Ad_1, Cd_1 \circ Ad_1, \exists x \varphi_1, \exists x \varphi_2 \quad *$		
$Ad_1, Cd_1 \circ Bd_1, \exists x \varphi_1, \exists x \varphi_2 \quad *$		
$Ad_1, Cd_1 \circ Bd_1, Bd_1 \wedge Cd_1, \exists x \varphi_1, \exists x \varphi_2 \quad **$	$\exists_R$	
$Ad_1, Cd_1 \circ Bd_1, \exists x \varphi_1, \exists x \varphi_2 \quad **$		
$Ad_1, Cd_1 \circ Cd_1, \dots \quad =$		

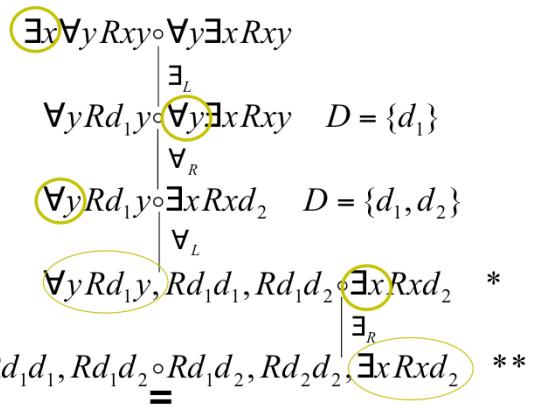
Tegen voorbeeld!



## Semantische tableaus -Vb 9.5

\*:(1)  $D = \{d_1, d_2\}$   
(2)  $\forall y R d_1 y : \{d_1, d_2\}$

\*\*:(1)  $D = \{d_1, d_2\}$   
(2)  $\forall y R d_1 y : \{d_1, d_2\}$   
 $\exists x R x d_2 : \{d_1, d_2\}$





Domein mag niet leeg zijn!

\* : (1)  $D = \{d_1\}$   
(2)  $\forall y \exists x Rxy : \{d_1\}$

\*\* : (1)  $D = \{d_1\}$   
(2)  $\forall y \exists x Rxy : \{d_1\}$   
 $\exists x \forall y Rxy : \{d_1\}$

\*\*\* : (1)  $D = \{d_1, d_2\}$   
(2)  $\forall y \exists x Rxy : \{d_1\}$   
 $\exists x \forall y Rxy : \{d_1\}$

\*\*\*\* : (1)  $D = \{d_1, d_2, d_3\}$   
(2)  $\forall y \exists x Rxy : \{d_1\}$   
 $\exists x \forall y Rxy : \{d_1\}$

## Semantische tableaus -Vb 9.6

$\forall y \exists x Rxy \circ \exists x \forall y Rxy \quad D = \{d_1\}$   
 $\forall y \exists x Rxy, \exists x Rxd_1 \circ \exists x \forall y Rxy \quad *$   
 $\forall y \exists x Rxy, \exists x Rxd_1 \circ \forall y Rd_1y, \exists x \forall y Rxy \quad **$   
 $\forall y \exists x Rxy, Rd_2d_1 \circ \forall y Rd_1y, \exists x \forall y Rxy \quad ***$   
 $\forall y \exists x Rxy, Rd_2d_1 \circ Rd_1d_3, \exists x \forall y Rxy \quad ****$

Dit wordt een oneindig diepe tak, want  
we moeten de formules terug invullen  
voor  $d_2$  en  $d_3$ !  
Dit tableau sluit niet.



## Semantische tableaus: Samenvatting

- Mogelijkheden:
  1. Het tableau **sluit**, gevolgtrekking is **geldig**.
  2. Er is een **niet-sluitende tak**. Deze kan:
    - 2.1 **eindig afbreken**, of
    - 2.2 **oneindig doorlopen**In beide gevallen: tegen voorbeeld; gevolgtrekking is **niet geldig**

Oneindige tak: tegen voorbeeld met een oneindig domein.



## Predikaatlogica & beslisbaarheid

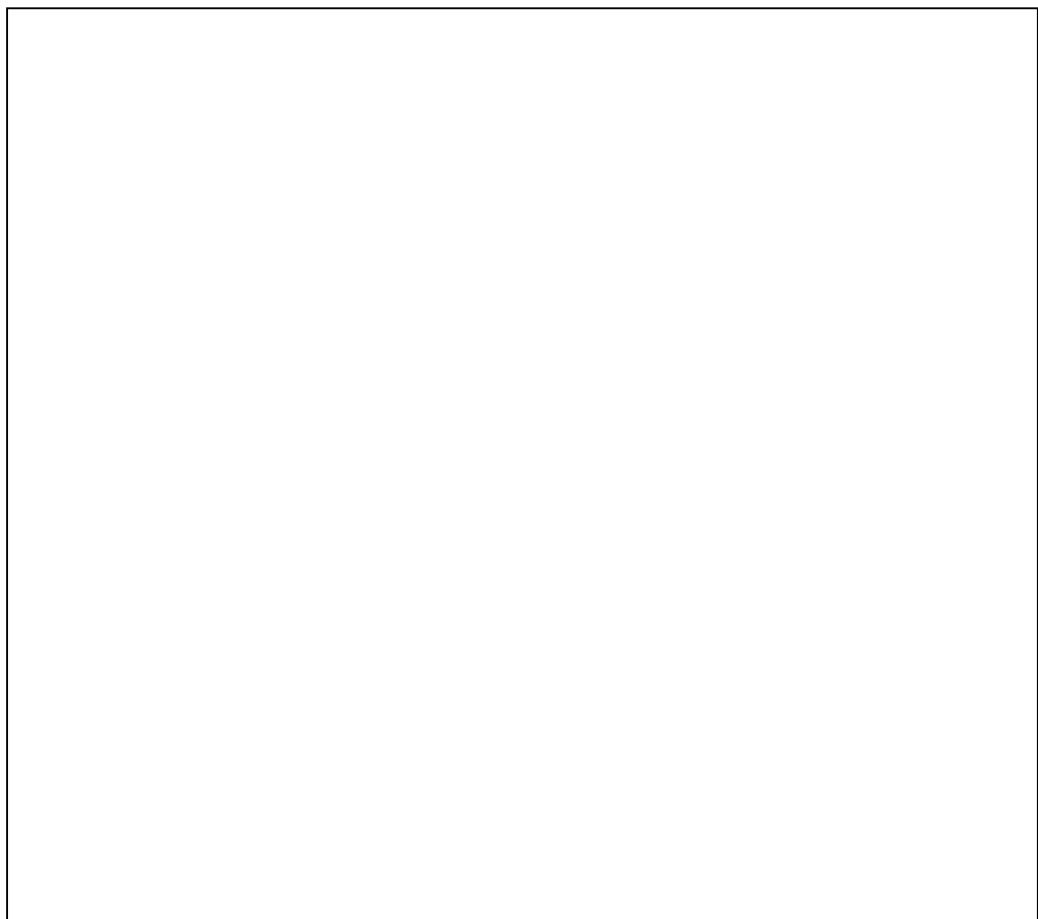
- De predikaatlogica is **niet beslisbaar** (stelling van Church 1936):  
Er bestaat geen algoritme om voor **iedere willekeurige** formule te bepalen of deze afleidbaar is of niet uit een formuleverzameling.
  - Sommige fragmenten van de predikaatlogica zijn wel beslisbaar
- Semantische tableaus zijn wel **adequaat** voor de predikaatlogica:  
Een sequent heeft een gesloten tableau dan en slechts dan als de corresponderende gevolgtrekking geldig is.
  - Zo'n gesloten tableau wordt gevonden wanneer men zorgt voor **fair scheduling** van de reductieregels (d.w.z. ze moeten allemaal aan de beurt komen).

Algemeen: Een theorie T is beslisbaar wanneer er een mechanische procedure bestaat om, bij een gegeven formule  $\phi$  van de taal, uit te maken of  $\phi$  afleidbaar uit T dan wel niet.



Vrije Universiteit Brussel

## Predikaatlogica: Afleidingen





We breiden het systeem van natuurlijke deductie uit de propositielogica uit tot predikaatlogica

- Afleidingsregels propositielogica blijven geldig
- Nodig: afleidingsregels voor  $\forall$  en  $\exists$ .
  - Beperking tot **formules zonder vrije variabelen**.



# Afleidingsregels

## – Extra afleidingsregels predikaatlogica:

Let op: t is een term zonder variabelen

d is een individuele constante

$$\frac{\forall x \varphi \text{ uit } \Sigma}{[t/x]\varphi \text{ uit } \Sigma} \forall E \quad \text{Ook wel de regel van instantiatie genoemd}$$

$$\frac{[d/x]\varphi \text{ uit } \Sigma}{\forall x \varphi \text{ uit } \Sigma} \forall I \quad \text{mits } d \text{ niet in } \forall x \varphi \text{ of } \Sigma \text{ voorkomt}$$

Ook wel de regel van generalisatie genoemd

Vb som vn hoeken diehoek is 180

Aantonen voor een willekeurige driehoek (d), zonder iets van de eigenschappen van die driehoek te gebruiken (d komt niet voor in ...)



# Afleidingsregels

$$\frac{[t/x]\varphi \text{ uit } \Sigma}{\exists x \varphi \text{ uit } \Sigma} \exists I$$

Mits t vrij is voor x in  $\varphi$

Als we een speciaal geval bewezen hebben  
dan kunnen we de existentiële kwantor invoeren

$$\frac{\exists x \varphi \text{ uit } \Phi \quad \psi \text{ uit } \Sigma, [d/x]\varphi}{\psi \text{ uit } \Phi \cup \Sigma} \exists E, [-[d/x]\varphi] \quad \text{mits } d \text{ niet in } \exists x \varphi \text{ of } \psi \text{ of } \Sigma \text{ voorkomt}$$

Een conclusie uit een existentiële bewering moet  
los staan van een specifiek voorbeeld van die bewering

Vb socrates sterfelijk -> er bestaat een mens die sterfelijk is



## Afleidingsregels - belangrijk

- Conditie “Mits t vrij is voor x in  $\varphi$ ” in  $\exists I$  is nodig om de volgende ongewenste afleiding te vermijden:

Uit  $\forall y Ryy$  mogen we niet afleiden:

$\exists x \forall y Ryx$  omdat y niet vrij voor x is in  $\forall y Ryx$

Nl:  $\varphi = \forall y Ryx$  ; t = y dan  $[y/x] \forall y Ryx = \forall y Ryy$

Dus y is nu gebonden! En dus was y niet vrij voor x in  $\forall y Ryx$ .



## Afleidingen - Opmerking

- Deze regels zijn eenvoudiger dan die in het boek, omdat zij beperkt zijn tot **formules zonder vrije variabelen**.
- De voorwaarden voor de toepassing van deze regels dienen zorgvuldig in acht genomen te worden en altijd bij de verantwoording van een toepassing van de regels worden vermeld.



– We bewijzen:  $\forall x \exists y Rxy \vdash \exists y Rdy$

1.  $\forall x \exists y Rxy$  uit  $\varphi$  aanname  $\varphi = \forall x \exists y Rxy$
2.  $\exists y Rdy$  uit  $\varphi$   $\forall E(1)$  ( $[t/x]$  waar  $t = d$ )



## Afleidingen: Vb 10.3

– We bewijzen:  $\forall x(Ax \rightarrow Bx) \vdash \forall x Ax \rightarrow \forall x Bx$

- |  |                     |  |
|--|---------------------|--|
| 1. $\forall x(Ax \rightarrow Bx)$          | uit $\varphi$       | aanname $\varphi = \forall x(Ax \rightarrow Bx)$         |
| 2. $\forall x Ax$                          | uit $\psi$          | aanname $\psi = \forall x Ax$                            |
| 3. $Ad$                                    | uit $\psi$          | $\forall E(2)$   |
| 4. $Ad \rightarrow Bd$                     | uit $\varphi$       | $\forall E(1)$   |
| 5. $Bd$                                    | uit $\varphi, \psi$ | $\rightarrow E(3,4)$                                     |
| 6. $\forall x Bx$                          | uit $\varphi, \psi$ | $\forall I(5) \ d$ niet in $\forall x Bx, \varphi, \psi$ |
| 7. $\forall x Ax \rightarrow \forall x Bx$ | uit $\varphi$       | $\rightarrow I(6)$                                       |



## Afleidingen: Vb 10.4

We bewijzen:

- |                                     |  |
|-------------------------------------|--|
| $\forall x Ax \vee \forall x Bx$    | $\vdash \forall x(Ax \vee Bx)$                                       |
| 1. $\forall x Ax \vee \forall x Bx$ | uit $\varphi$ aanname $\varphi = \forall x Ax \vee \forall x Bx$     |
| 2. $\forall x Ax$                   | uit $\psi$ aanname $\psi = \forall x Ax$                             |
| 3. $Ad$                             | uit $\psi$ $\forall E(2)$  |
| 4. $Ad \vee Bd$                     | uit $\psi$ $\vee I(3)$   |
| 5. $\forall x(Ax \vee Bx)$          | uit $\psi$ $\forall I(4) d$ niet in $\forall x(Ax \vee Bx)$ , $\psi$ |
| 6. $\forall x Bx$                   | uit $\chi$ aanname $\chi = \forall x Bx$                             |
| 7. $Bd$                             | uit $\chi$ $\forall E(6)$  |
| 8. $Ad \vee Bd$                     | uit $\chi$ $\vee I(7)$   |
| 9. $\forall x(Ax \vee Bx)$          | uit $\chi$ $\forall I(8) d$ niet in $\forall x(Ax \vee Bx)$ , $\chi$ |
| 10. $\forall x(Ax \vee Bx)$         | uit $\varphi$ $\vee E(1,5,9)$  |



## Afleidingen: Vb 10.5

We bewijzen:  $\exists x(Ax \wedge Bx) \vdash \exists x Ax \wedge \exists x Bx$

1.  $\exists x(Ax \wedge Bx)$  uit  $\chi$  aannname  $\chi = \exists x(Ax \wedge Bx)$
2.  $Ad \wedge Bd$  uit  $\varphi$  aanname  $\varphi = Ad \wedge Bd$
3.  $Ad$  uit  $\varphi$   $\wedge E(2)$
4.  $\exists x Ax$  uit  $\varphi$   $\exists I(3)$
5.  $Bd$  uit  $\varphi$   $\wedge E(2)$
6.  $\exists x Bx$  uit  $\varphi$   $\exists I(5)$
7.  $\exists x Ax \wedge \exists x Bx$  uit  $\varphi$   $\wedge I(4,6)$
8.  $\exists x Ax \wedge \exists x Bx$  uit  $\chi$   $\exists E(1,7)$  d niet in  $\exists x Ax \wedge \exists x Bx$  of  $\chi$



## Afleidingen: Vb 10.6

- We proberen te bewijzen dat  $\exists x Rxx \vdash \forall x \exists z Rxz$   
Deze afleiding is echter **niet correct!**
  1.  $\exists x Rxx$  uit  $\chi$  aanname  $\chi = \exists x Rxx$
  2.  $Rdd$  uit  $\varphi$  aanname  $\varphi = Rdd$
  3.  $\exists z Rdz$  uit  $\varphi$   $\exists I(2)$
  4.  $\exists z Rdz$  uit  $\chi$   $\exists E(1,3) -\varphi d$  niet in  $\exists z Rdz, \chi$  (Onjuist!)
  5.  $\forall x \exists z Rxz$  uit  $\chi$   $\forall I(4) d$  niet in  $\forall x \exists z Rxz, \chi$



## Afleidingen: Vb. 10.7

We laten zien dat  $\exists x(Ax \wedge Bx), \neg \exists x(Bx \wedge Cx) \vdash \neg \forall x(Ax \rightarrow Cx)$

1.  $\exists x(Ax \wedge Bx)$  uit  $\varphi$  aanname  $\varphi = \exists x(Ax \wedge Bx)$
2.  $\neg \exists x(Bx \wedge Cx)$  uit  $\psi$  aanname  $\psi = \neg \exists x(Bx \wedge Cx)$
3.  $Ad \wedge Bd$  uit  $\xi$  aanname  $\xi = Ad \wedge Bd$
4.  $\forall x(Ax \rightarrow Cx)$  uit  $\chi$  aanname  $\chi = \forall x(Ax \rightarrow Cx)$
5.  $Ad \rightarrow Cd$  uit  $\chi$   $\forall E(4)$
6.  $Ad$  uit  $\xi$   $\wedge E(3)$
7.  $Bd$  uit  $\xi$   $\wedge E(3)$
8.  $Cd$  uit  $\chi, \xi$   $\rightarrow E(6,5)$
9.  $Bd \wedge Cd$  uit  $\chi, \xi$   $\wedge I(7,8)$
10.  $\exists x(Bx \wedge Cx)$  uit  $\chi, \xi$   $\exists I(9)$
11.  $\neg \forall x(Ax \rightarrow Cx)$  uit  $\psi, \xi$   $\neg I(2,10)$  (intrekking van  $\chi$ )
12.  $\neg \forall x(Ax \rightarrow Cx)$  uit  $\varphi, \psi$   $\exists E(1,11)$  d niet in  $\varphi, \psi, \exists x(Ax \wedge Bx)$

Syllogisme: sommige A zijn B, geen B is C dus niet alle A zijn C



We laten zien dat  $\exists x \forall y Rxy \vdash \forall y \exists x Rxy$

1.  $\exists x \forall y Rxy$  uit  $\varphi$  aanname  $\varphi = \exists x \forall y Rxy$
2.  $\forall y Rdy$  uit  $\psi$  aanname  $\psi = \forall y Rdy$
3.  $Rde$  uit  $\psi$   $\forall E(2)$
4.  $\exists x Rxe$  uit  $\psi$   $\exists I(3)$
5.  $\exists x Rxe$  uit  $\varphi$   $\exists E(1,4)$   $d$  niet in  $\varphi$ ,  $\exists x Rxe$
6.  $\forall y \exists x Rxy$  uit  $\varphi$   $\forall I(5)$   $e$  niet in  $\forall y \exists x Rxy$ ,  $\varphi$



## Intermezzo: Axiomatisch bewijssysteem

- Zoals bij propositielogica bestaat er ook een alternatief bewijssysteem, nl axioma's en afleidingsregels



## Intermezzo: Axiomatisch bewijssysteem

- Voorbeeld hiervan uit de Wiskunde

- Peano-rekenkunde: theorie voor optellen en vermenigvuldigen van natuurlijke getallen

Constante: 0

Functieletters: +, ., S (opvolgfunctie)

Termen:  $0+x$ ,  $x.y$ ,  $S(x+Sy)$

0,  $S0$ ,  $SS0$ ,  $SSS0$ , ... komt overeen met 0, 1, 2, 3, ...

**Axioma's:**

$$\text{PA1: } \forall x \neg 0 = Sx$$

*0 is van geen enkel getal de opvolger*

$$\text{PA2: } \forall x \forall y (Sx = Sy \rightarrow x = y)$$

*opvolgerfunctie is injectief*

$$\text{PA3: } \forall x x + 0 = x$$

*recursieve definitie van +*

$$\forall x \forall y x + Sy = S(x + y)$$

*recursieve definitie van .*

$$\text{PA4: } \forall x x . 0 = 0$$

$$\forall x \forall y x . Sy = x . y + x$$

$$\text{PA5: } ([0/x] \varphi \wedge \forall x (\varphi \rightarrow [Sx/x] \varphi)) \rightarrow \forall x \varphi \quad \text{(voor elke formule } \varphi\text{)}$$

*principe van inductie (op S)*



Vrije Universiteit Brussel

## Predikaatlogica: Een eenvoudige theorie



## Eenvoudige theorie

- Inhoud
  - Substitutie
  - Prenexvormen
  - Fragmenten van Predikaatlogica



## Substitutie

- Herinner:

Bewering

Substitutie

Voor termen  $t, t'$  en variabele  $x$  geldt:

$$V_{M,b}([t/x]t') = V_{M,b[x \mapsto V_{M,b}(t)]}(t')$$

Bewijs: met inductie naar de opbouw van  $t'$ .



## Substitutie

$$V_{M,b}([t/x]t') = V_{M,b[x \mapsto V_{M,b}(t)]}(t')$$

Bewijs: met inductie naar de opbouw van  $t'$ .

- Als  $t' = x$  dan  $[t/x]t' = t$ , en dan  $V([t/x]t') = V(t)$   
Anderzijds  $V_{b[x \mapsto V(t)]}(t') = V_{b[x \mapsto V(t)]}(x) = V(t)$
- Als  $t' = y$  en  $y$  is een andere variabele of een constante,  
dan  $[t/x]t' = y$   
en  $V([t/x]t') = V(y)$   
Anderzijds  $V_{b[x \mapsto V(t)]}(t') = V_{b[x \mapsto V(t)]}(y) = V(y)$
- Inductiehypothese: stel bewering geldt voor termen  $t_i$ ,  $i = 1, \dots, n$   
Dan geldt voor  $t' = f(t_1, \dots, t_n)$  dat  $V([t/x]t') = I(f)(V([t/x]t_1), \dots, V([t/x]t_n)) =$   
 $I(f)(V_{b[x \mapsto V(t)]}(t_1), \dots, V_{b[x \mapsto V(t)]}(t_n))$   
Anderzijds:  
 $V_{b[x \mapsto V(t)]}(t') = V_{b[x \mapsto V(t)]}(f(t_1, \dots, t_n)) = I(f)(V_{b[x \mapsto V(t)]}(t_1), \dots, V_{b[x \mapsto V(t)]}(t_n))$



# Substitutie

In formules:

Probleem: door substitutie op een vrije variabele kan deze gebonden raken, daarom extra conditie!

## Bewering

Als  $\varphi$  een formule is,  $t$  een term,  $x$  een variabele en  $t$  is vrij voor  $x$  in  $\varphi$  dan geldt in elk model  $M$  en voor elke bedeling  $b$ :

$$V_{M,b}([t/x]\varphi) = V_{M,b[x \mapsto V_{M,b}(t)]}(\varphi)$$



## Substitutie

$$V_{M,b}([t/x]\varphi) = V_{M,b[x \mapsto V_{M,b}(t)]}(\varphi)$$

Bewijs: met inductie naar de opbouw van  $\varphi$ .

Schets

- Eerst voor  $\varphi = P(t_1, \dots, t_n)$
- Dan voor  $\varphi = \neg\psi$  en alle andere connectieven
- Dan voor  $\varphi = \forall z \Psi$ 
  - Hierbij 2 gevallen:  $x$  niet vrij en  $x$  vrij
- En  $\varphi = \exists z \Psi$  analoog



# Prenexvorm

Er bestaat een verband tussen  $\forall$  en  $\exists$

– Equivalenties met  $\forall$ ,  $\exists$  en  $\neg$ :

- $\forall x \neg \varphi$  is logisch equivalent met  $\neg \exists x \varphi$

NI.

$$V(\forall x \neg \varphi) = 1$$

desda voor alle  $d \in D$  geldt dat  $V_{b[x \mapsto d]}(\neg \varphi) = 1$

desda voor alle  $d \in D$  geldt dat  $V_{b[x \mapsto d]}(\varphi) = 0$

desda er is geen  $d \in D$  zodat  $V_{b[x \mapsto d]}(\varphi) = 1$

desda  $V(\exists x \varphi) = 0$

desda  $V(\neg \exists x \varphi) = 1$

- $\exists x \neg \varphi$  is logisch equivalent met  $\neg \forall x \varphi$

Voorbeeld:

$$\neg \neg \exists x \forall y \varphi \equiv \neg \exists x \forall y \varphi \equiv \forall x \neg \forall y \varphi \equiv \forall x \exists y \neg \varphi$$

$\equiv$  is logisch equivalent met



## Prenexvorm

- Equivalenties met  $\forall$ ,  $\exists$ ,  $\wedge$  en  $\vee$ :

Lemma:

Als  $\varphi$  en  $\psi$  formules zijn en  $x$  een variabele is die *niet vrij voorkomt in  $\psi$*  dan zijn de volgende formules logisch equivalent:

$(\exists x \varphi) \wedge \psi$ en $\exists x (\varphi \wedge \psi)$	$\psi \wedge (\exists x \varphi)$ en $\exists x (\psi \wedge \varphi)$
$(\forall x \varphi) \wedge \psi$ en $\forall x (\varphi \wedge \psi)$	$\psi \wedge (\forall x \varphi)$ en $\forall x (\psi \wedge \varphi)$
$(\exists x \varphi) \vee \psi$ en $\exists x (\varphi \vee \psi)$	$\psi \vee (\exists x \varphi)$ en $\exists x (\psi \vee \varphi)$
$(\forall x \varphi) \vee \psi$ en $\forall x (\varphi \vee \psi)$	$\psi \vee (\forall x \varphi)$ en $\forall x (\psi \vee \varphi)$

Als  $x$  toch vrij voorkomt in  $\psi$  dan kunnen we in  $\exists x \varphi$  of  $\forall x \varphi$  overgaan op een alfabetische variant



- Equivalenties met  $\forall$ ,  $\exists$  en  $\rightarrow$ :

**Lemma**

Als  $\varphi$  en  $\psi$  formules zijn en  $x$  een variabele is die *niet vrij voorkomt in  $\psi$* , dan zijn de volgende formules logisch equivalent:

$$\begin{array}{ll} (\forall x \varphi) \rightarrow \psi \text{ en } \exists x (\varphi \rightarrow \psi) & \psi \rightarrow (\forall x \varphi) \text{ en } \forall x (\psi \rightarrow \varphi) \\ (\exists x \varphi) \rightarrow \psi \text{ en } \forall x (\varphi \rightarrow \psi) & \psi \rightarrow (\exists x \varphi) \text{ en } \exists x (\psi \rightarrow \varphi) \end{array}$$

Namelijk (voorbeeld):  $(\exists x \varphi) \rightarrow \psi \equiv \neg(\exists x \varphi) \vee \psi \equiv$   
 $(\forall x \neg \varphi) \vee \psi \equiv \forall x (\neg \varphi \vee \psi) \equiv \forall x (\varphi \rightarrow \psi)$

$\equiv$  is logisch equivalent met



## Prenexvorm

- Equivalenties met  $\forall$ ,  $\exists$  en  $\leftrightarrow$ :

Lemma:

Als  $\varphi$  en  $\psi$  formules zijn en  $x$  is een variabele die *niet vrij voorkomt in  $\psi$*  en  $y$  is een variabele die *niet vrij voorkomt in  $\varphi$  en  $\psi$* , dan zijn de volgende formules logisch equivalent:

$$\begin{aligned} & (\exists x \varphi) \leftrightarrow \psi \text{ en} \\ & ((\exists x \varphi) \rightarrow \psi) \wedge (\psi \rightarrow (\exists x \varphi)) \text{ en} \\ & ((\forall x (\varphi \rightarrow \psi)) \wedge (\exists x (\psi \rightarrow \varphi))) \text{ en} \\ & ((\forall x (\varphi \rightarrow \psi)) \wedge (\exists y (\psi \rightarrow [y/x]\varphi))) \text{ en} \\ & \forall x ((\varphi \rightarrow \psi) \wedge (\exists y (\psi \rightarrow [y/x]\varphi))) \text{ en} \\ & \forall x \exists y ((\varphi \rightarrow \psi) \wedge (\psi \rightarrow [y/x]\varphi)) \end{aligned}$$



### Definitie

#### Prenexvorm

Een formule van de vorm  $Q_1x_1 \dots Q_nx_n \psi$ , waarbij

- $Q_i$  kwantoren ( $\exists$  of  $\forall$ ) zijn,
- $i = 1, \dots, n$  en
- $\psi$  een formule is waarin geen kwantoren meer voorkomen

heet een **prenexvorm** met  $Q_1x_1 \dots Q_nx_n$  als **prefix** en  $\psi$  als **matrix**.

Voorbeeld:

$$\exists x \forall y \forall x_1 (((Ryx \rightarrow Ay) \rightarrow Ax) \rightarrow Ax_1)$$



## Prenexstelling

### Prenexstelling:

Voor elke formule  $\varphi$  bestaat er een prenexformule  $\psi$  zodat  $\varphi$  en  $\psi$  logisch equivalent zijn, i.e ( $V(\varphi)=V(\psi)$ ).

Bewijs: met inductie naar  $\varphi$ .



- Voorbeeld:

$$\begin{aligned} & \forall x(\forall y(Ryx \rightarrow Ay) \rightarrow Ax) \rightarrow \forall x Ax \\ & \equiv \forall x(\forall y(Ryx \rightarrow Ay) \rightarrow Ax) \rightarrow \forall x_1 Ax_1 \text{ (alfabetische variant)} \\ & \equiv \forall x(\forall y(Ryx \rightarrow Ay) \rightarrow Ax) \rightarrow \forall x_1 Ax_1 \\ & \equiv \forall x(\exists y((Ryx \rightarrow Ay) \rightarrow Ax)) \rightarrow \forall x_1 Ax_1 \\ & \equiv \forall x(\exists y((Ryx \rightarrow Ay) \rightarrow Ax)) \rightarrow \forall x_1 Ax_1 \\ & \equiv \exists x(\exists y((Ryx \rightarrow Ay) \rightarrow Ax)) \rightarrow \forall x_1 Ax_1 \\ & \equiv \exists x(\exists y((Ryx \rightarrow Ay) \rightarrow Ax)) \rightarrow \forall x_1 Ax_1 \\ & \equiv \exists x\forall y(((Ryx \rightarrow Ay) \rightarrow Ax) \rightarrow \forall x_1 Ax_1) \\ & \equiv \exists x\forall y\forall x_1(((Ryx \rightarrow Ay) \rightarrow Ax) \rightarrow Ax_1) \end{aligned}$$

– NB De volgorde van de kwantoren vooraan hangt af van de volgorde waarin je ze naar voren haalt.



## Fragmenten van de Predikaatlogica

- Men beperkt zich vaak tot delen (fragmenten) van de predikaatlogica
  - **Monadische taal:** enkel één-plaatsige predikaatletters
    - Voldoende voor behandeling van syllogismen

Syllogismen: gevolgtrekkingen met 2 aannames en 1 conclusie van de vorm 'alle/geen/sommige A is/zijn B'  
voorbeeld: alle kaaimannen zijn reptielen  
geen reptiel kan fluiten  
dus geen kaaiman kan fluiten.



## Fragmenten van de Predikaatlogica

- Universele formules: alleen universele kwantoren in hun prefix
- Horn-zinnen

- zijn universele zinnen van de vorm

$\forall x_1 \dots \forall x_n (A_1 \wedge \dots \wedge A_k) \rightarrow B$   
waarbij  $A_1, \dots, A_k, B$  atomaire beweringen zijn

Worden gebruikt in de programmeertaal PROLOG

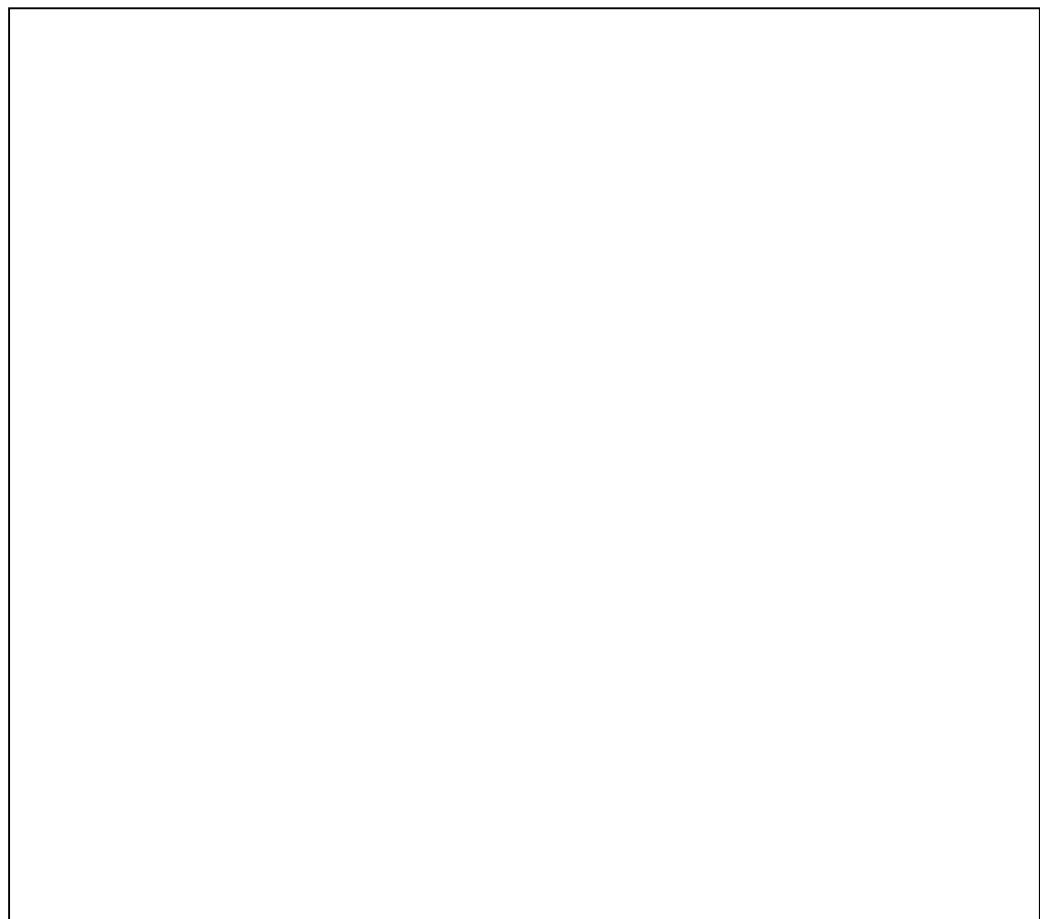
Voorbeeld:

```
sibling(X,Y) :-  
    parentS(F,M,X),  
    parentS(F,M,Y),  
    X \= Y.
```



Vrije Universiteit Brussel

## Predikaatlogica: Metatheorie





# Predikaatlogica: Metatheorie

## – Inhoud:

- Adequaatheid van semantische tableaus
- De volledigheidsstelling



## Adequaatheid van tableaus

- Semantische tableaus zijn een juiste methode om de geldigheid van een gevolgtrekking in de predikaatlogica te bewijzen of te weerleggen.
- Herinner:
  - Beperkt tot taal zonder functiesymbolen, zonder individuele constanten en zonder vrije variabelen

### Adequaatheidsstelling (zonder bewijs)

Als  $\Sigma$  een formuleverzameling is en  $\varphi$  een formule, dan geldt:

$\Sigma \models \varphi$  desda er bestaat een gesloten semantisch tableau voor  $\Sigma \cup \varphi$



# Volledigheidsstelling

## Volledigheidsstelling:

Voor de predikaatlogica geldt dat voor iedere formuleverzameling  $\Sigma$  en iedere formule  $\varphi$

- $\Sigma \vdash \varphi$  desda  $\Sigma \models \varphi$
- Correctheid (“soundness”):  
 $\Sigma \vdash \varphi$  impliceert  $\Sigma \models \varphi$
- Volledigheid (“completeness”):  
 $\Sigma \models \varphi$  impliceert  $\Sigma \vdash \varphi$



Vrije Universiteit Brussel

## EINDE PREDIKAATLOGICA



Vrije Universiteit Brussel

# Inleiding tot de $\lambda$ -calculus

Prof. dr. Olga De Troyer



## Overzicht

- Inleiding
- $\lambda$ -expressies
- Currying
- Vrije en gebonden variabelen; combinator
- Substitutie en  $\beta$ -gelijkheid
- $\lambda$ -expressies en rekenen
- Programmeerconstructies
- Recursiviteit en fixpunten



# $\lambda$ -calculus

- Geïntroduceerd door Alonzo Church en Stephen Kleene in de jaren 1930s om het begrip functie abstract te kunnen bestuderen
- Ook gebruikt om het begrip berekenbaarheid te formaliseren
  - Om na te gaan welke functies al dan niet berekenbaar zijn, of nog, welke (wiskundige) problemen oplosbaar zijn, en welke niet

(Een functie  $f$  is berekenbaar desda er bestaat een programma  $P$  dat de functie  $f$  berekent, m.a.w.  $P$  stop voor input  $a$  en geeft  $b$  als output als  $f$  gedefinieerd is voor  $a$  en anders stop  $P$  nooit ).



# $\lambda$ -calculus & Programmeertalen

- Hoewel ontwikkeld vóór het bestaan van computertalen is het formalisme de basis van **functionele programmeertalen** zoals Lisp en Scheme
  - Basisconcept van deze programmeertalen zijn nl functies
- Invloed op programmeertalen:
  - “Call by name” mechanism
    - Parameters worden pas geëvalueerd indien nodig
  - Hogere orde functies
    - functie met functies als parameters of
    - Output is een functie.

1938 Zuse bouwde eerste computer Z1 of de Zuses.

Computers werden pas in de 2<sup>de</sup> wereldoorlog gebruikt om codes te kraken

Call by name en hogere orde functies in programmeertalen zijn afkomstig van Lambda Calculus



# Functies

- Een numerieke functie is op twee manieren te definiëren
  - Extensioneel (verzameling theoretisch)
  - Intentioneel (functievoorschrift)
- Voorbeeld:
  - $f(x) = 2 + x$  functievoorschrift

verzameling theoretisch:

{..., (-1, 1), (0, 2), (1, 3), ... }

extensie



- $f(x) = 2 + x$

→ In  $\lambda$ -achtige notatie:

parameter  $\lambda x. (+2)x$  voorschrift

– Te lezen als:

Om de waarde van  $f(x)$  te berekenen voor de parameter  $x$ :

- Pas de functie  $+2$  toe op  $2$ : dit resulteert in de functie ' $+2$ '  
 $+2(x) = x + 2$

- Pas de functie ' $+2$ ' toe op de parameter  $x$

- Merk op: prefix notatie

– Merk op dat er geen naam nodig is voor de functie.



## $\lambda$ -calculus - basis

- 2 fundamentele bewerkingen:
  - **Abstractie**: het maken van een functievoorschrift
  - **Toepassen** (of aanroepen) van het functievoorschrift op een parameter.



# Syntax of $\lambda$ -calculus

## Definitie $\lambda$ -expressies

Laat  $V$  een verzameling variabelen zijn  $V = \{x, y, z, \dots\}$

De verzameling van  $\lambda$ -expressies wordt als volgt gedefinieerd:

1. alle variabelen (elementen van  $V$ ) zijn  $\lambda$ -expressies
2. Als  $M$  en  $N$   $\lambda$ -expressies zijn dan is  
 $(M)N$  een  $\lambda$ -expressie (toepassing)
3. Als  $x \in V$  en  $M$   $\lambda$ -expressie dan is  
 $\lambda x.M$  een  $\lambda$ -expressie (abstractie)

M.a.w.:

definitie van functie in  $\lambda$ -calculus:  $\lambda$ formele-parameter.functievoorschrift

Aanroep van een functie in  $\lambda$ -calculus: (functie) actuele-parameter

De verzameling van  $\lambda$ -expressies wordt genoteerd als  $\Lambda$ .



## Voorbeelden λ-expressies

$\lambda x.x$

$\lambda x.\lambda y.(y)x$

$(\lambda y.(x)y) \lambda x.(u)x$

$\lambda x. \lambda y.x$

$\lambda f. \lambda x.(f)x$

$\lambda f. \lambda x.(f)(f)x$



# Syntax of $\lambda$ -calculus

Toepassing van rechts naar links:

$(P)(Q)x$    komt overeen met  $P(Q(x))$   
                        (in klassieke functienotatie)

$((P)Q)x$    komt overeen met eerst P toepassen op  
                        Q en daarna het resultaat op x  
                        functie



# Currying

## Maar 1 parameter voor functies? Beperking?

- Functie met meerdere parameters steeds voor te stellen via functies met 1 parameter

F: domein → co-domein

Waarbij co-domein zelf weer functies kan bevatten

m.a.w:  $f: A \times B \rightarrow C$  wordt  
 $g: A \rightarrow (B \rightarrow C)$  functie  
 $g(a) = f_a$  en  $f_a(b) = f(a,b)$

Voorbeeld:

plus(2,3) wordt  $+_2 = +_2$  en  $+_2(3) = 5$



# Currying

In het algemeen:

$$A_1 \times \dots \times A_n \rightarrow B \text{ wordt } A_1 \rightarrow (A_2 \rightarrow \dots (A_n \rightarrow B))$$

Deze techniek wordt **currying** genoemd, genoemd naar een Amerikaanse wiskundige Curry.

- **Voordeel:**

- Theorie te beperken tot functies met 1 argument

- **Nadeel:**

- Leesbaarheid

Voorbeeld:  $\lambda x. \lambda y. (y)x$

is functie met 2 argumenten die zijn 2de argument toepast op zijn eerste argument of m.a.w.  $g(x,f) = f(x)$



## Currying in Scheme

(lambda (x y)  
(+ x y))

Currying →

(lambda (x)  
(lambda (y)  
(+ x y)))

((lambda (x y)  
(+ x y)) 3 4)

((lambda (x)  
lambda (y)  
(+ x y))) 3) 4)

geeft bij evaluatie (x gebonden  
aan 3):

((lambda (y)  
(+ 3 y)) 4)



## Binden van variabelen

Voor een  $\lambda$ -expressie van de vorm  $\lambda x.M$  zegt men:

- $\lambda x$  bindt  $x$  in de  $\lambda$ -expressie  $M$
- Het bereik van de binding is  $M$ , d.w.z. dat alle nog niet gebonden voorkomens van  $x$  in  $\lambda x.M$  gebonden zijn

Voorbeeld:

$$((\lambda x.\lambda z.((x)z)(\lambda x.x)z)\lambda y.y)\lambda x.x$$



# Vrije en gebonden variabelen

In een  $\lambda$ -expressie heten alle voorkomen van variabelen die niet gebonden zijn **vrij**

## Definitie

Laat  $M$  een  $\lambda$ -expressie zijn. Dan wordt de **verzameling  $VV(M)$  van vrije variabelen van  $M$**  als volgt gedefinieerd:

- Voor alle variabelen  $x \in V$ :  $VV(x) = \{x\}$
- $VV(\lambda x.M) = VV(M) \setminus \{x\}$
- $VV((M)N) = VV(M) \cup VV(N)$

## Definitie

Een  $\lambda$ -expressie zonder vrije variabelen heet **gesloten** of een **combinator**. De verzameling van combinatoren wordt genoteerd als  $\Lambda_0$



## Substitutie - intuïtief

- Uitwerking van  $(\lambda x.M)P$ 
  - Intuïtief: door elk voorkomen van de parameter  $x$  te vervangen door  $P$

Om dit formeel te definiëren moeten we eerst de substitutie definiëren.



## Substitutie - definitie

### Definitie

Zij  $P$  en  $M$  twee  $\lambda$ -expressies, en  $x$  een variabele  $x \in V$ .

De substitutie  $\{P/x\}M$  van  $P$  voor  $x$  in  $M$ , is als volgt gedefinieerd:

$$\{P/x\}x = P \quad (\text{S1})$$

$$\{P/x\}y = y \text{ als } y \in V \setminus \{x\} \quad (\text{S2})$$

$$\{P/x\}(F)Q = (\{P/x\}F)\{P/x\}Q \quad (\text{S3})$$

$$\{P/x\} \lambda x. M = \lambda x. M \quad (\text{S4})$$

$$\{P/x\} \lambda y. M = \lambda y. \{P/x\} M \text{ als } y \neq x \text{ en } y \notin VV(P) \quad (\text{S5})$$

$$\begin{aligned} \{P/x\} \lambda y. M &= \lambda z. \{P/x\} \{z/y\} M \\ \text{als } y \neq x \text{ en } z \notin VV(P) \text{ en } z \notin VV(M) &\quad (\text{S6}) \end{aligned}$$



## Substitutie - toelichting

Toelichting regel (S6):

$$\{P/x\} \lambda y. M = \lambda z. \{P/x\} \{z/y\} M$$

waarbij  $y$  vrije variabele in  $P$

Voorbeeld:  $\{\lambda u.(y)u/x\} \lambda y.(\textcolor{blue}{x})y$

$P$  staat voor  $\lambda u.(y)u$

$M$  staat voor  $(\textcolor{blue}{x})y$

**Vóór** substitutie:  $y$  is vrij in  $P$ :  $\lambda u.(\textcolor{red}{y})u$

Bij een directe substitutie zou  $y$  gebonden worden:

$$\{P/x\} \lambda y. M = \lambda y. (\lambda u.(\textcolor{red}{y})u)y$$

Daarom eerst  $y$  vervangen door (bijvoorbeeld)  $z$  in  $M$ :  $\lambda z.(\textcolor{blue}{x})z$

Dan  $x$  vervangen door  $P$ . Het correct antwoord:

$$\lambda z. (\lambda u. (y)u)z$$



## Voorbeeld substitutie

$\{\lambda u.(y)u/x\} \lambda y.(x)y$

$\{P/x\}M$

$P \equiv \lambda u.(y)u$ ,  $M \equiv \lambda y.(x)y$  en  $y \in VV(P)$  dus (S6):

$\lambda z.\{\lambda u.(y)u/x\}\{z/y\}(x)y$  dan (S3):

$\lambda z.\{\lambda u.(y)u/x\}(\{z/y\}x)\{z/y\}y$  dan (S2):

$\lambda z.\{\lambda u.(y)u/x\}(x)\{z/y\}y$  dan (S1):

$\lambda z.\{\lambda u.(y)u/x\}(x)z$  dan (S3):

$\lambda z.(\{\lambda u.(y)u/x\}x)\{\lambda u.(y)u/x\}z$  dan (S2):

$\lambda z.(\{\lambda u.(y)u/x\}x)z$  dan (S1):

$\lambda z.(\lambda u.(y)u)z$



## $\beta$ -gelijkheid - definitie

### Definitie

De  $\beta$ -gelijkheid relatie  $=_{\beta} \subseteq \Lambda \times \Lambda$  is gedefinieerd als volgt:

1.  $(\lambda x.M)P =_{\beta} \{P/x\}M$  (β)
2.  $\lambda x.M =_{\beta} \lambda z.\{z/x\}M$  als  $z \notin VV(M)$  (α)
3.  $M =_{\beta} M$  (reflexief)
4.  $M =_{\beta} N$  dan ook  $N =_{\beta} M$  (symmetrisch)
5.  $M =_{\beta} N$  en  $N =_{\beta} L$  dan ook  $M =_{\beta} L$  (transitief)
6.  $M =_{\beta} M'$  en  $P =_{\beta} P'$  dan  $(M)P =_{\beta} (M')P'$  (congruent)
7.  $M =_{\beta} M'$  dan  $\lambda x.M =_{\beta} \lambda x.M'$  (congruent)

Axioma's 1 en 2 worden de  $\beta$  - en  $\alpha$ -axioma's genoemd (historisch was  $\beta$  het 2de axioma)

Het  $\beta$ -axioma geeft de intuïtieve betekenis van het toepassen van een functie op een actuele parameter, nl. formele parameter vervangen door de actuele parameter

Het  $\alpha$ -axioma laat toe om een parameter te herbenoemen

Axioma's 3, 4 en 5 zeggen dat  $=_{\beta}$  een *equivalentie* relatie is

Axioma 6 & 7 zeggen dat het niet belangrijk is waar we beginnen met uitwerken, eerst de parameter  $P$  (nl.  $(M)P =_{\beta} (M')P'$ ) of eerst de functie  $M$  (nl.  $(M)P =_{\beta} (M')P$ )



## Rekenen in $\lambda$ -calculus

- $\lambda$ -calculus kent geen constanten
  - Toch kan men de natuurlijke getallen voorstellen d.m.v. de  $\lambda$ -calculus
  - We kunnen zelf een volledige programmeertaal maken met:
    - Constanten, rekenkundige functies
    - Boolean, if-then constructies
    - Recursie
    - ....

*Definitie*

Beschouw  $\lambda$ -expressies  $F$  en  $M$ . De expressie  $(F)^n M$  wordt inductief gedefinieerd als volgt:

$$(F)^0 M \equiv M$$

$$(F)^{1+n} M \equiv (F)(F)^n M$$



# Rekenen in $\lambda$ -calculus

## *Lemma (“+”)*

Beschouw  $\lambda$ -expressies  $F$  en  $M$ .

Voor alle  $n, m \in \mathbb{N}$  geldt dat

$$(F)^{n+m}M \equiv (F)^n(F)^mM$$

Bewijs: per inductie op  $n$

**Geval  $n = 0$**

$$(F)^{0+m}M \equiv (F)^mM$$

neem  $M' \equiv (F)^mM$

$$\equiv M' \equiv (F)^0M' \quad (def)$$

$$\equiv (F)^0(F)^mM$$



## Rekenen in $\lambda$ -calculus

Stel lemma geldt voor het **geval  $n = k$  (IH)**;  
Nu bewijzen voor **het geval  $n = k+1$  of  $1+k$** :

$$\begin{aligned} (F)^{1+k+m} M &= (F)^{1+(k+m)} M \\ &\equiv (F)(F)^{(k+m)} M \quad (\text{def}) \\ &\equiv (F)(F)^k (F)^m M \quad (IH: \text{inductiehypothese}) \\ &\equiv (F)(F)^k M' \quad (M' \equiv (F)^m M) \\ &\equiv (F)^{1+k} M' \quad (\text{def}) \\ &\equiv (F)^{1+k} (F)^m M \quad (M' \equiv (F)^m M). \end{aligned}$$



# Rekenen in $\lambda$ -calculus

## Church getallen

### Voorstelling natuurlijke getallen:

- 0 als  $\lambda f. \lambda x. x$  → Een functie en een parameter
- 1 als  $\lambda f. \lambda x. (f)x$  → Functie 1 keer toegepast op parameter
- 2 als  $\lambda f. \lambda x. (f)(f)x$  → Functie 2 keer toegepast op parameter
- enz.

### Definietie

De zogenaamde **Church getallen**  $c_n$  ( $n \in \mathbb{N}$ ) worden gedefinieerd als volgt:

$$c_n = \lambda f. \lambda x. (f)^n x$$

Functievoorschrift met 2 argumenten; het 1ste argument (een functie) wordt een aantal keer ( $n$ ) toegepast op het 2<sup>de</sup> argument (de parameter voor de functie).

Dus de natuurlijke getallen worden voorgesteld door welbepaalde lambda expressies.



# Rekenen in $\lambda$ -calculus

## $\lambda$ -definieerbaar

### Definitie

Een numerieke functie  $f: \mathbb{N}^p \rightarrow \mathbb{N}$  met  $p \in \mathbb{N}$  parameters is  **$\lambda$ -definieerbaar** als er een combinator  $F$  bestaat zodat

$$(((F)c_{n1})c_{n2}) \dots c_{np} =_{\beta} c_{f(n1, n2, \dots, np)}$$

m.a.w. er bestaat een combinator waarvan het effect op de Church getallen hetzelfde is als de operatie toegepast op de 'echte' getallen

Voorbeeld: de optelling is  $\lambda$ -definieerbaar als er een lambda expressie (combinator) *plus* bestaat zodat:

$$((\text{plus}) c_{n1}) c_{n2} =_{\beta} c_{+(n1, n2)}$$

$$\text{bv: } ((\text{plus}) c_2) c_3 =_{\beta} c_5$$



## Rekenen in $\lambda$ -calculus “successor”

**succ**(n) = n+1 is  $\lambda$ -definieerbaar

Bewijs:

Zoek **succ** zodat:  $(\text{succ})c_n =_{\beta} c_{n+1}$

We hebben dus nodig:

$$(\text{succ})c_0 \equiv (\text{succ}) \lambda f. \lambda x. x =_{\beta} \lambda f. \lambda x. (f)x \equiv c_1$$

$$(\text{succ})c_1 \equiv (\text{succ}) \lambda f. \lambda x. (f)x =_{\beta} \lambda f. \lambda x. (f)(f)x \equiv c_2$$

In het algemeen moet **(succ)** dus een extra “aanroep” van f toevoegen:

$$(\text{succ})c_n =_{\beta} \lambda f. \lambda x. (\textcolor{red}{f}) \text{ voorschrift}_n$$

Hoe vinden we nu *voorschrift*\_n?

≡ Wordt hier gebruikt om het gebruik van een definitie aan te geven



## Rekenen in $\lambda$ -calculus “successor”

Voorschrift<sub>n</sub> vinden:

$$(\mathbf{succ})c_n =_{\beta} \lambda f. \lambda x. (\textcolor{red}{f}) \text{ voorschrift}_n$$

Merk op dat voorschrift<sub>n</sub> staat voor  $(f)^n x$

en  $c_n \equiv \lambda f. \lambda x. (\textcolor{yellow}{(f)^n x})$

Dus als we de  $\lambda f. \lambda x.$  kunnen wegwerken hebben we het voorschrift:

$$\begin{aligned} ((c_n) f) x &\equiv ((\lambda f. \lambda x. (\textcolor{yellow}{(f)^n x})) f) x \\ &=_{\beta} (\lambda x. (\textcolor{yellow}{(f)^n x})) x \\ &=_{\beta} (\textcolor{green}{f})^n x \\ &\equiv \text{voorschrift}_n \end{aligned}$$

Dus  $(\mathbf{succ})c_n =_{\beta} \lambda f. \lambda x. (\textcolor{red}{f}) ((c_n) f) x$       of  
 $\mathbf{succ} \equiv \lambda n. \lambda f. \lambda x. (\textcolor{red}{f}) ((\textcolor{blue}{n}) f) x$



## Rekenen in $\lambda$ -calculus “successor”

Dus  $\mathbf{succ}(n) = n+1$  is  $\lambda$ -definieerbaar en  
 $\mathbf{succ} \equiv \lambda n. \lambda f. \lambda x. (f)((n)f)x$

Check (enkel voor  $c_0$ ):

$$\begin{aligned} (\mathbf{succ})c_0 &\equiv (\lambda n. \lambda f. \lambda x. (f)((n)f)x) c_0 && \text{(def } c_0\text{)} \\ &\equiv (\lambda n. \lambda f. \lambda x. (f)((n)f)x) \color{green}{\lambda f. \lambda x. x} \\ &= (\color{red}{\lambda n. \lambda f. \lambda x. (f)((n)f)x}) \color{red}{\lambda f. \lambda x. x} \\ &= \beta \lambda f. \lambda x. (f)((\color{red}{\lambda f. \lambda x. x})f)x && (\beta) \\ &= \lambda f. \lambda x. (f)(\color{teal}{\lambda f. \lambda x. x})x \\ &= \lambda f. \lambda x. (f)(\color{green}{\lambda x. x})x \\ &= \beta \lambda f. \lambda x. (f) \color{teal}{x} && (\beta) \\ &\equiv c_1 \end{aligned}$$

Bv examenvraag: bewijs dat  $(\mathbf{succ})c_n = c_{n+1}$



## Rekenen in $\lambda$ -calculus Optelling

Is de optelling  $\lambda$ -definieerbaar?

Ja en de combinator is:

$$\text{plus} = \lambda n. \lambda m. \lambda f. \lambda x. ((n)f)((m)f)x$$

Intuitief (gebaseerd op definitie van church getallen):

n keer toepassen van f op

m keer toepassen van f op x

=> in totaal n+m keer toepassen van f op x



# Rekenen in $\lambda$ -calculus

## Optelling

$$\text{plus} \equiv \lambda n. \lambda m. \lambda f. \lambda x. ((n)f)((m)f)x$$

Bewijs

$$\begin{aligned} ((\text{plus}) c_n) c_m &\equiv ((\lambda n. \lambda m. \lambda f. \lambda x. ((n)f)((m)f)x) c_n) c_m && (\text{def plus}) \\ &\equiv_{\beta} (\lambda m. \lambda f. \lambda x. ((c_n)f)((m)f)x) c_m && (\beta) \\ &\equiv (\lambda m. \lambda f. \lambda x. ((\lambda f. \lambda x. (f^n x)f)((m)f)x) c_m && (\text{def } c_n) \\ &\equiv (\lambda m. \lambda f. \lambda x. ((\lambda f. \lambda x. (f^n x)f)((m)f)x) c_m && (\beta) \\ &\equiv_{\beta} (\lambda m. \lambda f. \lambda x. (\lambda x. (f^n x)((m)f)x) c_m && (\beta) \\ &\equiv (\lambda m. \lambda f. \lambda x. (\lambda x. (f^n x)((m)f)x) c_m && (\beta) \\ &\equiv (\lambda m. \lambda f. \lambda x. (f^n((m)f)x) c_m && (\beta) \\ &\equiv (\lambda m. \lambda f. \lambda x. (f^n((m)f)x) c_m && (\beta) \\ &\equiv_{\beta} \lambda f. \lambda x. (f^n((c_m)f)x) && (\beta) \\ &\equiv \lambda f. \lambda x. (f^n((\lambda f. \lambda x. (f^m x)f)x) && (\text{def } c_m) \\ &\equiv \lambda f. \lambda x. (f^n((\lambda f. \lambda x. (f^m x)f)x) && (\beta) \\ &\equiv_{\beta} \lambda f. \lambda x. (f^n(\lambda x. (f^m x)x) && (\beta) \\ &\equiv \lambda f. \lambda x. (f^n(\lambda x. (f^m x)x) && (\beta) \\ &\equiv_{\beta} \lambda f. \lambda x. (f^n(f^m x) && (\beta) \\ &\equiv \lambda f. \lambda x. (f^{n+m}x) && (\text{Lemma "+"}) \\ &\equiv C \end{aligned}$$



# Rekenen in $\lambda$ -calculus

Volgend doel: vermenigvuldiging

Eerst lemma

*Lemma ("x")*

Voor alle  $n, m \in \mathbb{N}$  geldt dat

$$\underbrace{((c_n)f)^m}_{{m \text{ keer } h \text{ elkaar}}} y =_{\beta} (f)^{n \times m} y$$

Soort copierfunctie.



# Rekenen in $\lambda$ -calculus

TB:  $((c_n)f)^m y =_{\beta} (f)^{nxm} y$

Bewijs per inductie over m

1. Geval m = 0

$$\begin{aligned} ((c_n)f)^0 y &\equiv y && \text{(def)} \\ &\equiv (f)^0 y && \text{(def)} \\ &= (f)^{nx0} y \end{aligned}$$



## Rekenen in $\lambda$ -calculus

Vervolg bewijs  $((\mathbf{c}_n)f)^m y =_{\beta} (f)^{nx^m} y$

2. Stel bewezen voor geval  $m=k$  (IH),  
dan nu bewijs voor het geval  $m= k+1$

$$\begin{aligned} ((\mathbf{c}_n)f)^{k+1} y &\equiv ((\mathbf{c}_n)f) ((\mathbf{c}_n)f)^k y \\ &\equiv ((\mathbf{c}_n)f) (f)^{n \times k} y && \text{inductiehypothese} \\ &\equiv ((\lambda f. \lambda x. (f)^n x) f) (f)^{n \times k} y && (\text{def } \mathbf{c}_n) \\ &=_{\beta} (\lambda x. (f)^n x) (f)^{n \times k} y && (\beta) \\ &=_{\beta} (f)^n (f)^{n \times k} y && (\beta) \\ &\equiv (f)^{n+n \times k} y && (\text{lemma "+"}) \\ &= (f)^{nx(1+k)} y \end{aligned}$$



# Rekenen in $\lambda$ -calculus vermenigvuldiging

Is de vermenigvuldiging  $\lambda$ -definieerbaar?

Ja, en de combinator is

$$\mathbf{times} \equiv \lambda n. \lambda m. \lambda f. (n)(m)f$$

Bewijs

$$((\mathbf{times}) c_n) c_m$$

$$\begin{aligned} &\equiv ((\lambda n. \lambda m. \lambda f. (n)(m)f) \mathbf{c}_n) \mathbf{c}_m && (\text{def times}) \\ &=_{\beta} (\lambda m. \lambda f. (\mathbf{c}_n)(m)f) \mathbf{c}_m && (\beta) \\ &\equiv (\lambda m. \lambda f. (\lambda f. \lambda x. (f)^n x)(m)f) \mathbf{c}_m && (\text{def } c_n) \\ &= (\lambda m. \lambda f. (\lambda f. \lambda x. (f)^n x)(\mathbf{m})f) \mathbf{c}_m \\ &=_{\beta} (\lambda m. \lambda f. \lambda x. ((m)f)^n x) \mathbf{c}_m && (\beta) \\ &= (\lambda m. \lambda f. \lambda x. ((\mathbf{m})f)^n x) \mathbf{c}_m \\ &=_{\beta} \lambda f. \lambda x. ((\mathbf{c}_m)f)^n x && (\beta) \\ &= \lambda f. \lambda x. ((\mathbf{c}_m)f)^n x \\ &=_{\beta} \lambda f. \lambda x. (\mathbf{f})^{mn} x && (\text{Lemma "x"}) \\ &\equiv c_{nxm} \end{aligned}$$



## “Boolean” expressies in $\lambda$ -calculus

$$\mathbf{true} \equiv \lambda t. \lambda f. t$$

$$\mathbf{false} \equiv \lambda t. \lambda f. f$$

*true geeft 1ste argument terug;*

*false geeft 2de argument terug*

$$((\mathbf{true})A)B =_{\beta} A$$

$$((\mathbf{false})A)B =_{\beta} B$$

*Bewijs:*

$$((\lambda t. \lambda f. t) A) B =_{\beta} (\lambda f. A) B =_{\beta} A$$

$$((\lambda t. \lambda f. f) A) B =_{\beta} (\lambda f. f) B =_{\beta} B$$



## “If” statement in $\lambda$ -calculus

$$\text{if} \equiv \lambda c. \lambda d. \lambda e. ((c)d)e$$

Staat voor: if c then d else e

c (conditie) moet true of false geven

true  $\Rightarrow ((c)d)e$  geeft 1ste argument terug, zijnde d

false  $\Rightarrow ((c)d)e$  geeft 2de argument, zijnde e

$$(((\text{if})\text{true})A)B =_{\beta} A$$

Bewijs:

$$\begin{aligned} (((\lambda c. \lambda d. \lambda e. ((c)d)e) \text{true})A)B &=_{\beta} ((\lambda d. \lambda e. ((\text{true})d)e)A)B \\ &=_{\beta} (\lambda e. ((\text{true})A)e)B =_{\beta} ((\text{true})A)B =_{\beta} A \end{aligned}$$



## “iszero” in $\lambda$ -calculus

$$\text{iszero} \equiv \lambda n. ((n) \lambda x. \text{false}) \text{true}$$

$\lambda x. \text{false}$ : geeft steeds false terug

- $((n) \lambda x. \text{false}) \text{true}$  : als n een church getal, dan (betekenis church getal) n keer toepassen van de functie  $\lambda x. \text{false}$  op het argument true: geeft steeds false
  - maar bij  $n = c_0$  wordt de functie niet toegepast, maar wordt het argument (true) terug gegeven

Dus:

$$\begin{aligned} (\text{iszero})c_0 &=_{\beta} \text{true} \\ (\text{iszero})c_n &=_{\beta} \text{false} \quad \text{for } n > 0 \end{aligned}$$



## “is-zero” in $\lambda$ -calculus

$$(iszero)c_0 =_{\beta} \text{true}$$

Bewijs

$$\begin{aligned} (\lambda n.((n) \lambda x. \text{false}) \text{true}) c_0 &=_{\beta} ((c_0) \lambda x. \text{false}) \text{true} \\ &\equiv ((\lambda f. \lambda x. x) \lambda x. \text{false}) \text{true} && (\text{def } c_0) \\ &=_{\beta} (\lambda x. x) \text{true} \\ &=_{\beta} \text{true} \end{aligned}$$

$$(iszero)c_n =_{\beta} \text{false} \quad \text{for } n > 0$$

Bewijs

$$\begin{aligned} (\lambda n.((n) \lambda x. \text{false}) \text{true}) c_n &=_{\beta} ((c_n) \lambda x. \text{false}) \text{true} \\ &\equiv (\lambda x. \text{false})^n \text{true} && (\text{lemma "x": } ((c_n)f)^m y =_{\beta} (f)^{nxm} y) \\ &=_{\beta} \text{false} \end{aligned}$$



## “car en cdr”

$$\mathbf{car} \equiv \lambda a. \lambda d. a$$

$$\mathbf{cdr} \equiv \lambda a. \lambda d. d$$

*car geeft 1ste argument terug; cdr geeft 2de argument terug*

$$\mathbf{cons} \equiv \lambda a. \lambda d. \lambda z. ((z)a)d$$

*Intuïtief: geef een car en geef een cdr en er wordt een  $\lambda$ -expressie terug gegeven, die ofwel car ofwel cdr terug geeft*

$$((\mathbf{cons}) A) B \equiv ((\lambda a. \lambda d. \lambda z. ((z)a)d)A)B =_{\beta} \lambda z. ((z)A)B$$



## “cons”

$$(((\text{cons}) A) B) \text{ car} =_{\beta} A$$

$$(((\text{cons}) A) B) \text{ cdr} =_{\beta} B$$

Bewijs

$$\begin{aligned} (((\text{cons}) A) B) \text{ car} &=_{\beta} (\lambda z.((z)A)B) \text{ car} \\ &=_{\beta} ((\text{car})A)B \\ &\equiv ((\lambda a.\lambda d.a)A)B \\ &=_{\beta} (\lambda d.A)B \\ &=_{\beta} A \end{aligned}$$



## Andere numerieke functies als $\lambda$ -expressies

$\text{nextp} \equiv \lambda p. ((\text{cons})(\text{succ})(p)\text{car}) (p)\text{car}$

$p$  als cons

Voorbeeld  $p$  als  $(1,0)$

$(p)\text{car}$

geeft de car van de cons  $p$ , dus 1

$(\text{succ})(p)\text{car}$

geeft de successor van de car van de cons  $p$ ,  
dus 2

$((\text{cons})(\text{succ})(p)\text{car}) (p)\text{car}$

geeft nieuwe cons met 2 en 1, dus  $(2,1)$



## Andere numerieke functies als $\lambda$ -expressies

***pred***  $\equiv \lambda n. ((n)\text{nextp}) ((\text{cons})c_o)c_o) \text{ cdr}$

n keer nextp toepassen op (0,0)

Daarvan de cdr nemen

Vb: pred(3): 3 keer nextp van (0,0) = (3,2)

Dan cdr (3,2) is 2

***minus***  $\equiv \lambda m. \lambda n. ((m)\text{pred})n$

m keer pred toepassen op n,

m.a.w. m keer -1 toepassen op n, dus n - m



# Recursie

- We missen nu nog recursiviteit voor een volwaardige programmeertaal!

Eerst introductie van fixpunten



# Fixpunten

Voor een numerieke functie  $f: D \rightarrow D$  heet  $x \in D$  een **fixpunt** van  $f$  als en slechts als  $f(x) = x$

Bv. 0 is fixpunt voor  $*^2$ , nl  $0 *^2 = 0$

Voor een  $\lambda$ -expressie wordt dit:

**Definitie**

$X \in \Lambda$  is een **fixpunt** van  $F \in \Lambda$  als en slechts als

$$(F)X =_{\beta} X$$

*Stelling*

1. Iedere  $\lambda$ -expressie heeft een fixpunt:

$$\forall F \in \Lambda \exists X: (F)X =_{\beta} X$$

2. Er is een fixpunt combinator

$$Y = \lambda f. (\lambda x. (f)(x)x) \lambda x. (f)(x)x$$

waarvoor geldt dat

$$\forall F \in \Lambda : (F)(Y)F =_{\beta} (Y)F$$

m.a.w. deze  $Y$  laat toe om het fixpunt te berekenen voor een willekeurige  $F$ , nl. het fixpunt is  $(Y)F$ .



Bewijs deel 1

- Definieer  $W \equiv \lambda x.(F)(x)x$   
en  $X \equiv (W)W$
- Dan geldt  $X \equiv (W)W \equiv (\lambda x.(F)(x)x)W$   
 $\equiv_{\beta} (F)(W)W \equiv (F)X$

En dus is  $X$  een fixpunt van  $F$

Bewijs deel 2

- $(Y)F \equiv (\lambda f.(\lambda x.(f)(x)x) \lambda x.(f)(x)x)F$   
 $\equiv_{\beta} (\lambda x.(F)(x)x) \lambda x.(F)(x)x$   
 $\equiv (W)W$   
 $\equiv X$

en dus is  $(Y)F$  een fixpunt van  $F$ .

**Gevolg**

Beschouw de combinator  $F \equiv \lambda f.\text{body}_F$  en zijn fixpunt  $X_F \equiv (Y)F$

dan is het fixpunt:  $X_F =_{\beta} \{X_F / f\} \text{body}_F$

**Bewijs**

Omdat  $X_F$  een fixpunt is van  $F$  is  $(F)X_F =_{\beta} X_F$

Nu is  $(F)X_F \equiv (\lambda f.\text{body}_F)X_F$

$=_{\beta} \{X_F / f\} \text{body}_F$  ( $\beta$  regel).



## Recursiviteit - voorbeeld

Voorbeeld recursieve definitie in de wiskunde

$$\begin{aligned} n! &= 1 && \text{als } n = 0 \\ n! &= n \cdot (n-1)! && \text{als } n > 0 \end{aligned}$$

```
(define fac (lambda n)
  (if (zero? n) 1
      (* n (fac (pred n)))))
```

Letterlijk vertaling naar  $\lambda$ -calculus:

$\text{fac} \equiv \lambda n. (((\text{if}(\text{iszero})n)c, ((\text{times})n)(\text{fac})(\text{pred})n))$

Deze definitie is echter **circulair!**: **fac** is gedefinieerd in termen van **fac** zelf! Mag niet!



## Recursiviteit - voorbeeld

Oplossing:  $\equiv$  vervangen door  $=_{\beta}$

$$\begin{aligned} \mathbf{fac} =_{\beta} & \lambda n. (((\mathbf{if})(\mathbf{iszzero})n)\mathbf{c}_1)((\mathbf{times})n)(\mathbf{fac}) \\ & (\mathbf{pred})n \end{aligned}$$

We zoeken dus een combinator **fac** zodanig dat bovenstaand geldt



## Recursiviteit - voorbeeld

$$\mathbf{fac} =_{\beta} \lambda n. (((\mathbf{if})(\mathbf{iszero})n)\mathbf{c}_1)((\mathbf{times})n)(\mathbf{fac}) (\mathbf{pred})n$$

Introductie van  
een parameter

We kunnen dit herschrijven als volgt:

$$\mathbf{fac} =_{\beta} (\lambda f. \lambda n. (((\mathbf{if})(\mathbf{iszero})n)\mathbf{c}_1)((\mathbf{times})n)(f) (\mathbf{pred})n) \mathbf{fac}$$

Neem nu

$$\mathbf{FAC} = \lambda f. \lambda n. (((\mathbf{if})(\mathbf{iszero})n)\mathbf{c}_1)((\mathbf{times})n)(f) (\mathbf{pred})n$$

We hebben dus een combinator gevonden waarvoor geldt:

$$\mathbf{fac} =_{\beta} (\mathbf{FAC}) \mathbf{fac}$$

M.a.w. **fac** is het fixpunt voor **FAC** !!

We kunnen **fac** dus definieren als

$$\mathbf{fac} = (Y) \mathbf{FAC}$$



# Recursiviteit

Samenvatting

$$\text{fac} \equiv \lambda n. (((\text{if})(\text{iszero})n)c_1)((\text{times})n)(\text{fac}) (\text{pred})n$$

$$\text{FAC} \equiv \lambda f. \lambda n. (((\text{if})(\text{iszero})n)c_1)((\text{times})n)(f) (\text{pred})n$$

$$Y \equiv \lambda f. (\lambda x. (f)(x)x) \lambda x. (f)(x)x$$

$$\text{fac} \equiv (Y) \text{FAC}$$

Dit principe is te veralgemenen voor het definiëren van andere recursieve functies:

- Definieer de combinator in kleine letters
- Ga van de combinator in kleine letters naar combinator in grote letters (door introductie van een parameter)
- De combinator in kleine letters is fixpunt voor de combinator in grote letters.



# EINDE LAMBDA CALCULUS

# Inleiding tot de $\lambda$ calculus

D. Vermeir

B. Manderick

W. De Meuter

S. Casteleyn

## 1 Inleiding

Alonzo Church en Stephen Kleene ontwierpen de pure  $\lambda$ -calculus<sup>1</sup> in de jaren '30 als een systeem dat kan gebruikt worden om het begrip "functie", en meer-bepaald functie definitie, applicatie en recursie, abstract te bestuderen.

Hiermee wordt bedoeld dat de nadruk ligt op het functievoorschrift eerder dan een verzamelingtheoretische benadering, waar een functie bekijken wordt als een verzameling van geordende paren.

Bijvoorbeeld kan de functie

$$f(x) = 2 + x$$

over de gehele getallen bekijken worden d.m.v. haar voorschrift

$$2 + x$$

of als de verzameling

$$f = \{\dots, (-1, 1), (0, 2), (1, 3), \dots\}$$

waarbij het idee van het toepassen van het functievoorschrift naar de achtergrond wordt verbannen.

In  $\lambda$ -notatie<sup>2</sup> zou  $f$  geschreven worden als:

$$\lambda x.((+2)x) \tag{1}$$

Intuitief kan men dit lezen als: om de waarde van  $f(x)$  te berekenen voor een zekere parameter  $x$  (dit laatste wordt aangeduid door de  $\lambda x.$ ) gaat men als volgt te werk:

- Neem de functie  $+$  (optelling) en pas deze toe op het element 2. Dit levert een nieuwe functie (laten we ze  $+_2$  noemen) op van één enkele parameter die als volgt kan beschreven worden:

$$+_2(y) = y + 2$$

---

<sup>1</sup>Er bestaan ook zgn. getypeerde varianten waarover we het in deze tekst niet zullen hebben. Pure  $\lambda$ -calculus is net als Scheme niet getypeerd.

<sup>2</sup>Merk op dat we het hebben over  $\lambda$ -notatie niet  $\lambda$ -calculus: de hier gegeven expressie is immers, zoals we later zullen zien, geen geldige  $\lambda$ -calculus expressie, maar geeft wel een idee van de gebruikte notatie.

Deze nieuwe functie  $+_2$  heeft dus één parameter, en doet niets anders dan 2 optellen bij (de waarde van) deze parameter.

- In de uitdrukking (1) wordt  $+_2$  aldus verder toegepast op de waarde  $x$  wat tot het uiteindelijk resultaat leidt.

Uit (1) valt nog op te maken dat men in  $\lambda$ -calculus blijkbaar geen naam moet geven aan een functie ( $f$  komt helemaal niet voor in het voorbeeld) en dat functies slechts één argument hebben. Door echter functies te definieren waarvan de functiewaarde voor een gegeven parameter weer een functie ( $+_2$  in het voorbeeld) is, is het mogelijk om functies met meerdere parameter te simuleren ( $+$  in het voorbeeld)<sup>3</sup>.

Het blijkt dus al direct dat de benadering van  $\lambda$ -calculus intiem verbonden is met het idee van functieëvaluatie als *berekening*, en dus ook met functievoorschrift als *programma*. Het hoeft dan ook niet te verwonderen dat  $\lambda$ -calculus diverse toepassingen kent binnen de informatica. Het is bijvoorbeeld de basis voor zogenaamde *functionele programmeertalen*, talen die de nadruk leggen op functie evaluatie i.p.v. data en staat veranderingen, zoals gebruikelijk bij imperatieve programmeertalen. Lisp, de voorganger van Scheme, is rechtstreeks gebaseerd op  $\lambda$ -calculus en is een voorbeeld van een (niet-pure) functionele programmeertaal. Later werden heel wat nieuwere functionele programmeertalen ontwikkeld, zoals ML, Miranda, Haskell en Gofer, die eigenlijk allen gebaseerd zijn op en uitbreidingen zijn van de originele  $\lambda$ -calculus. Ook het zogenaamde *call-by-name* ('lazy evaluation') mechanisme dat in sommige programmeertalen voorhanden is komt uit de  $\lambda$ -calculus. Er dient ook opgemerkt te worden dat de beïnvloeding in twee richtingen gaat vermits vele recente resultaten in de  $\lambda$ -calculus rechtstreeks geïnspireerd werden door problemen in de informatica.

Naast bovengenoemde invloed op de ontwikkeling van programmeertalen was de  $\lambda$ -calculus eveneens van groot belang voor de theorie van de "berekenbaarheid". Berekenbaarheid is de tak van de wiskunde en (theoretische) informatica die uitspraken doet over welke functies al dan niet berekenbaar zijn, of anders gezegd, welke (wiskundige) problemen oplosbaar zijn, en welke niet. Historisch gezien is het beslissen of twee  $\lambda$ -expressies equivalent zijn het eerste probleem waarvan werd aangetoond dat het onbeslisbaar is: het is onmogelijk een programma te schrijven dat, gegeven twee willekeurige  $\lambda$ -expressies bepaalt of beiden  $\lambda$ -expressies equivalent zijn of niet.

## 2 Basisbegrippen

$\lambda$ -calculus gaat uit van twee fundamentele bewerkingen omtrent functies: aanroepen (applicatie) en aanmaken (abstractie). Met "aanroepen" bedoelen we hier het toepassen van het functievoorschrift op een parameter. Met "abstractie" maken we van een voorschrift een 'nieuwe' functie.

---

<sup>3</sup>Zie verder over "Currying".

## 2.1 Lambda Expressies

De volgende definitie definieert de verzameling der geldige uitdrukkingen ( $\lambda$ -expressies) in de  $\lambda$ -calculus.

**Definitie 1** Weze  $V$  een verzameling variabelen. De verzameling van  $\lambda$ -expressies  $\Lambda$  wordt als volgt gedefinieerd:

- $V \subseteq \Lambda$
- Als  $M \in \Lambda$  en  $N \in \Lambda$  dan is  $(M)N \in \Lambda$  (**applicatie of toepassing**).
- Als  $x \in V$  en  $M \in \Lambda$  dan is  $\lambda x.M \in \Lambda$  (**abstractie**).
- Niets anders zit in  $\Lambda$

Bovenstaande (constructieve) definitie definieert wat geldige  $\lambda$ -expressies zijn. Merk op dat hier nog geen enkele uitspraak wordt gedaan over de *betekenis* van de expressies; op dit moment maken we enkel uitspraken over wat syntactisch gezien geldige  $\lambda$ -expressies zijn. Voor een beter begrip geven we wel al intuitief aan wat de betekenis van de  $\lambda$ -expressies is; een formele definitie van wat we nu juist kunnen doen met deze  $\lambda$ -expressies, en dus wat de semantiek ervan is, komt later.

- De eerste regel zegt simpelweg dat een iedere variabele een  $\lambda$ -expressie is.
- De tweede regel zegt dat, gegeven twee geldige  $\lambda$ -expressies  $M$  en  $N$ , we een nieuwe geldige  $\lambda$ -expressie kunnen vormen door haakjes rond één te zetten en de andere erachter te plaatsen. Deze regel correspondeert met een functieaanroep:

*(functie) actuele parameter*

De actuele parameter (ook wel *argument*) van een functie is de parameter die een oproeper van de functie ‘aan de functie geeft’.

- De derde regel zegt dat, gegeven een variabele en een geldige  $\lambda$ -expressie, we een nieuwe geldige  $\lambda$ -expressie kunnen vormen door een  $\lambda$  voor de variabele te zetten, gevolgd door een punt en de geldige  $\lambda$ -expressie. Deze regel beschrijft de vorm van een “functie” in  $\lambda$ -calculus:

*$\lambda$ formele parameter variabele functie voorschrift*

De formele parameter van een functie is de parameter zoals die in de *definitie* van de functie voorkomt.

- De vierde regel zorgt ervoor dat al wat niet gevormd kan worden door (opeenvolgende) toepassing van de vorige drie regels geen geldige  $\lambda$ -expressie kan zijn.

Intuïtief moet b.v.  $(\lambda x.x)y$  gelezen worden als “pas de functie  $\lambda x.x$  toe op de parameter  $y$ ”. Aan de functie  $\lambda x.x$  kan men zien dat  $x$  de formele parameter is (de formele parameter is de variabele vóór het punt) terwijl de tweede  $x$  (in dit voorbeeld) het voorschrift geeft.

Merk op dat de pure  $\lambda$ -calculus geen constanten kent. We zullen later zien dat het toch mogelijk is om b.v. (gehele) getallen door middel van  $\lambda$ -expressies voor te stellen. Evenzo zullen we zien dat het mogelijk is om bepaalde arithmetische functies, programmeerconcepten en -constructies, zoals b.v. “+” en “-”, booleans, if-then expressies, arithmetische functies, lijsten (allen niet aanwezig in  $\lambda$ -calculus) voor de stellen door middel van  $\lambda$ -expressies.

Merk ook op dat we het in deze tekst steeds (verkeerdelijk) over “functies” in  $\lambda$ -calculus hebben. Eigenlijk bevat  $\lambda$ -calculus helemaal geen functies, ze is, zoals eerder reeds vermeld, slechts een verzameling van goed gevormde expressies (volgens 1) waaraan wij mensen een bepaalde interpretatie kunnen geven.

Weze  $x$  en  $y$  variabelen ( $x, y \in V$ )<sup>4</sup>, dan zijn volgende voorbeelden correcte  $\lambda$ -expressies.

### Voorbeeld 1

$$\begin{array}{c}
 x \\
 \lambda x.x \\
 \lambda x.\lambda y.(y)x \\
 (\lambda y.(x)y)\lambda x.(u)x \\
 \lambda x.\lambda y.x \\
 \lambda f.\lambda x.(f)x \\
 \lambda f.\lambda x.(f)(f)x
 \end{array}$$

Merk op dat uit definitie 1 volgt dat ‘functie-applicatie’ (waarover we het later nog uitvoerig gaan hebben) van rechts naar links gebeurt. Stel b.v. dat  $M$  en  $N$  geldige  $\lambda$ -expressies zijn, en  $x$  een variabele, dan verkrijgt men b.v. het resultaat van

$$(P)(Q)x$$

door eerst  $Q$  toe te passen op  $x$ , en daarna  $P$  toe te passen op het resultaat (van  $(Q)x$ ). M.a.w.,  $(P)(Q)x$  zou in “gewone” notatie geschreven worden als

$$P(Q(x))$$

Inderdaad kan men volgens de definitie  $(P)(Q)x$  slechts opbouwen door eerst  $(Q)x$  te vormen uit  $Q$  en  $x$  (regel 2 in Definitie 1), en dan weer een applicatie te vormen (opnieuw regel 2 in Definitie 1), waarbij  $(Q)x$  de (actuele) parameter is.

---

<sup>4</sup>In het vervolg zullen we niet steeds explicet vermelden wat de variabelen zijn; we gaan er in het vervolg vanuit dat letters gebruikt als variabelen in  $\lambda$ -expressies inderdaad als dusdanig gedefinieerd werden

De normale orde kan gewijzigd worden door de applicaties anders op te bouwen:  
in

$$((P)Q)x$$

staat een  $\lambda$ -expressie van de vorm  $(M)x$ , waarbij  $M = (P)Q$ . We moeten dus eerst  $P$  toepassen op  $Q$ , waarna de resulterende  $\lambda$ -expressie wordt toegepast op  $x$ . Tenslotte kan nog opgemerkt worden dat

$$((P)(Q))x$$

geen geldige  $\lambda$ -expressie is vermits de applicatie  $(Q)$  geen argument heeft.

## 2.2 Currying

Uit Definitie 1 blijkt dat een ‘functie’ in  $\lambda$ -calculus slechts één parameter kan hebben. Men zou zich kunnen afvragen of dit geen beperking vormt indien men functies van meerdere variabelen (b.v. de optelfunctie voor gehele getallen) wil voorstellen. Het blijkt dat dit geen probleem is omdat we b.v. een functie van 2 variabelen steeds kunnen beschouwen als een functie van 1 variabele die zelf een functie van 1 variabele als resultaat geeft (herinner het voorbeeld van  $+_2$  uit de inleiding; we gaan hier nu dieper op in).

Wanneer we een functie  $F$  voorstellen door een pijl

$$F : \text{domein} \rightarrow \text{codomein}$$

van het domein naar het codomein, waarbij het codomein zelf weer functies kan bevatten (m.a.w. bijvoorbeeld  $(E \rightarrow F)$  stelt een verzameling van functies van  $E$  naar  $F$  voor) komt dit neer op het vervangen van

$$f : A \times B \rightarrow C$$

door

$$g : A \rightarrow (B \rightarrow C)$$

waarbij voor alle  $a \in A, b \in B$   $g(a) = f_a$  zo gedefinieerd is dat  $f_a(b) = f(a, b)$ .

Toegepast op de optelling van twee (gehele) getallen komt dit neer op het definieren van een functie  $+$  die een willekeurig getal  $a$  omzet in een functie  $+_a$  die zelf weer gedefinieerd wordt door  $+_a(x) = a + x$ .

Het is duidelijk dat men op deze manier een functie van  $n$  variabelen kan omzetten naar een van een functie van 1 variabele die een functie van  $n - 1$  variabelen teruggeeft, waarbij we dus

$$A_1 \times \dots \times A_n \rightarrow B$$

vervangen door

$$A_1 \rightarrow (A_2 \times \dots \times A_n \rightarrow B)$$

Indien  $n > 2$ , dan kan men

$$A_2 \times \dots \times A_n \rightarrow B$$

weer vervangen door

$$A_2 \rightarrow (A_3 \times \dots A_n \rightarrow B)$$

enz., tot men uiteindelijk

$$A_1 \rightarrow (A_2 \rightarrow \dots (A_n \rightarrow B))$$

bekomt.

Dit mechanisme wordt “currying” genoemd naar Haskell B. Curry. We kunnen “currying” gebruiken in de  $\lambda$ -calculus omdat deze toelaat om functies van hogere orde, d.w.z. functies die b.v. een functie als functiewaarde teruggeven, te definiëren.

Het bovenstaande laat ons toe om b.v. de  $\lambda$ -expressie<sup>5</sup>

$$G \equiv \lambda x. \lambda f. (f)x$$

te beschouwen als een functie  $G$  van twee variabelen (in “gewone” notatie):

$$G(x, f) = f(x)$$

die zijn tweede parameter ( $f$ ) toepast op zijn eerste parameter ( $x$ ).

Currying is op zich interessant voor informatica vermits het toelaat om een gedeeltelijke evaluatie te doen wanneer b.v. niet alle parameters van een functie met verschillende parameters beschikbaar zijn. Het gebruik van zogenaamde “objecten” in de taal Scheme is hiervan een voorbeeld.

### 2.3 Vrije en gebonden variabelen

Het toepassen van een functie wordt in de  $\lambda$ -calculus gesimuleerd door het *herschrijven* van een  $\lambda$ -expressie door een andere (hopelijk eenvoudiger) expressie.

B.v. kan  $(\lambda x.x)y$  herschreven worden als  $y$  waarbij we dit bekomen door de declaratie van de formele parameter  $\lambda x$  te laten vallen en in het voorschrift “ $x$ ” de formele parameter variabele  $x$  te vervangen door de actuele parameter  $y$ . Formeel zullen we dit definieren door middel van substitutie (van formele door actuele parameters). Om bvb.

$$(\lambda x.(x)x)y$$

‘uit te rekenen’ dienen we alle voorkomens van  $x$  (de formele parameter) in  $(x)x$  (het voorschrift) te substitueren door  $y$  (de actuele parameter) zodat het resultaat  $(y)y$  wordt. We moeten dus een definitie opstellen die precies vastlegt hoe we alle voorkomens van een variabele in een  $\lambda$ -expressie kunnen vervangen door een andere  $\lambda$ -expressie. Men dient echter zeer zorgvuldig te zijn bij de precieze definitie van substitutie. Men kan b.v.

$$(\lambda x. \lambda y. x)y$$

---

<sup>5</sup>Het symbool  $\equiv$  wordt in deze tekst (en in vele andere wiskundige teksten) gebruikt om ‘is per definitie gelijk aan’ aan te geven.  $\equiv$  is als het ware de **define** van de wiskunde.

niet zomaar herschrijven als

$$\lambda y.y$$

waarbij in het voorschrift horend bij  $\lambda x$  (t.t.z.  $\lambda y.x$ )  $x$  vervangen werd door  $y$ . Inderdaad, intuïtief stelt  $(\lambda x.\lambda y.x)$  een functie van twee parameters voor (we gebruiken currying) die de eerste parameter als functiewaarde geeft. Echter, door in  $(\lambda x.\lambda y.x)y$   $x$  te vervangen door  $y$  verkrijgen we  $\lambda y.y$ , een functie die steeds de (oorspronkelijke) tweede parameter teruggeeft! Het probleem zit hem in het feit dat  $y$  een “vrije” variabele is<sup>6</sup> die ongewild in de substitutie gebonden wordt door de “binding”  $\lambda y$  in  $(\lambda x.\lambda y.x)y$ . Inderdaad is de tweede  $y$  in  $\lambda y.y$  niet vrij maar verwijst die gewoon naar de parameter van de functie.

Merk op dat het probleem zich niet voordoet als we de  $\lambda y$  in  $(\lambda x.\lambda y.x)y$  vervangen door een ongebruikte variabele, b.v.  $z$ :

$$(\lambda x.\lambda z.x)y$$

wordt dan, na substitutie van  $x$  door  $y$ :

$$\lambda z.y$$

wat een functie is die steeds  $y$  (de oorspronkelijke tweede parameter) als functiewaarde geeft.

De onderstaande definities zullen ons toelaten om substitutie zo te definiëren dat problemen zoals boven geschetst niet voorkomen.

### Definitie 2

- Voor een  $\lambda$ -expressie van de vorm

$$\lambda x.M$$

zegt men dat

- $\lambda x$  een **binding** is van  $x$  in  $M$
- het **bereik** van de binding is  $M$ , dit wil zeggen dat alle (nog niet gebonden) voorkomens van  $x$  in  $\lambda x.M$  gebonden zijn.

- In een  $\lambda$ -expressie heten alle voorkomens van variabelen die niet gebonden zijn **vrij**.

We definiëren nu de verzameling  $VV(M)$  die alle variabelen uit  $M$  bevat die minstens één keer vrij voorkomen in  $M$ .

**Definitie 3** Weze  $M \in \Lambda$ . De verzameling  $VV(M)$  van **vrije variabelen** van  $M$  wordt gedefinieerd door:

- $\forall x \in V : VV(x) = \{x\}$

---

<sup>6</sup>In informatica termen kan men een vrije variabele beschouwen als een globale variabele (of in elk geval een variabele die in een hogere bereik (“scope”) werd gedefinieerd).

- $VV(\lambda x.N) = VV(N) \setminus \{x\}$
- $VV((M)N) = VV(M) \cup VV(N)$

Een  $\lambda$ -expressie zonder vrije variabelen heet **gesloten** of een **combinator**. We noteren de verzameling van combinatoren als  $\Lambda_0 \subset \Lambda$ .

## 2.4 Substitutie

Intuïtief willen we een toepassing (functieaanroep)

$$(\lambda x.M)P$$

uitwerken door in het voorschrift  $M$  elk voorkomen van de formele parameter  $x$  die “hoort bij” de binding  $\lambda x$  te vervangen door de actuele parameter  $P$ . De voorkomens van  $x$  die “horen bij” de binding  $\lambda x$  zijn natuurlijk precies de voorkomens van  $x$  die *vrij* zijn in  $M$  (let op: vrij in  $M$ , niet in  $\lambda x.M$ ).

Formeel zullen we die substitutie noteren als

$$[P/x]M$$

Intuïtief lezen we dit als ”in  $M$  alle voorkomens van  $x$  vervangen door  $P$ ”. Echter, bij de precieze definitie van  $[P/x]M$  zullen we er moeten voor zorgen dat vrije voorkomens van variabelen in  $P$  niet ”per vergissing” (zoals in het voorbeeld aangehaald in de vorige sectie) gebonden worden (in de gesubstitueerde voorkomens van  $P$ ).

**Definitie 4** Beschouw twee  $\lambda$ -expressies  $P$  en  $M$ , en een variabele  $x \in V$ . De **substitutie**  $[P/x]M$  van  $P$  voor  $x$  in  $M$ , wordt als volgt gedefinieerd:

$$[P/x]x = P \tag{S1}$$

$$[P/x]y = y \quad \text{als } y \in V \setminus \{x\} \tag{S2}$$

$$[P/x](F)Q = ([P/x]F)[P/x]Q \tag{S3}$$

$$[P/x]\lambda x.M = \lambda x.M \tag{S4}$$

$$[P/x]\lambda y.M = \lambda y.[P/x]M \quad \text{als } y \neq x \text{ en } y \notin VV(P) \tag{S5}$$

$$[P/x]\lambda y.M = \lambda z.[P/x][z/y]M \quad \text{als } y \neq x \text{ en } z \notin VV(P) \text{ en } y \in VV(P) \tag{S6}$$

De regels in Definitie 4 dekken alle mogelijke gevallen van  $\lambda$ -expressies, en vertellen ons voor ieder geval hoe we de substitutie moeten doorvoeren. Regels (S1) en (S2) behandelen variabelen, regel (S3) behandelt applicaties, en regels (S4), (S5) en (S6) behandelen abstractie. Intuïtief moeten we Definitie 4 als volgt begrijpen.

- Regel 1 is triviaal, en zegt dat als we in  $x$ ,  $x$  moeten vervangen door  $P$ , we dan  $P$  bekomen.
- Regel 2 is eveneens triviaal, en zegt dat als we in  $y$ ,  $x$  moeten vervangen door  $P$ , we dan  $y$  bekomen (immers,  $x \neq y$  en dus is er helemaal geen  $x$  om te vervangen).

- Regel 3 zegt dat als we met een applicatie te maken hebben, we de substitutie moeten doorvoeren op beide deel- $\lambda$ -expressies van de applicatie.
- Regel 4 zegt dat als we in  $\lambda x.M$  (alle)  $x$ 'en moeten vervangen door  $P$ , we dan helemaal niets moeten doen (immers, de  $x$ 'en in  $\lambda x.M$  zijn gebonden door de  $\lambda x$ !)
- Regel 5 zegt dat we, in het algemeen, als we in  $\lambda y.M$  (alle)  $x$ 'en moeten vervangen door  $P$ , we de  $\lambda y$  moeten laten staan, en de substitutie doorvoeren in  $M$  (zie regel 6 voor de uitzondering op deze regel).
- Regel 6 behandelt de uitzonderingen op regel 5, namelijk wanneer we "per vergissing" een vrije variabele in  $M$  (verkeerdelyk) zouden kunnen binden. Dit is het geval wanneer  $y$  vrij voorkomt in  $P$ : bij klakkeloze substitutie zouden de oorspronkelijk vrij voorkomende  $y$ 's (in  $P$ ) immers plots (verkeerdelyk) gebonden worden door de  $\lambda.y$  van  $\lambda y.M$ ! Om ervoor te zorgen dat zo'n verkeerdelyke binding niet mogelijk is, gaan we in dit geval eerst een "hernoeming" doorvoeren: we herschrijven  $\lambda y.M$  tot de vorm  $\lambda z.M$ , waarbij we alle voorkomens  $y$  in  $M$  vervangen door  $z$  (m.a.w. we voeren de substitutie  $[z/y]M$  door). Hierbij mag  $z$  uiteraard niet vrij voorkomen in  $P$ , anders zitten we met hetzelfde probleem. Eenmaal deze hernoeming doorgevoerd, komen we in het geval van regel 5, en kunnen we de substitutie doorvoeren als in regel 5, m.a.w. houden we

$$\lambda z.[P/x]M'$$

over, waarbij we weten dat  $z \notin VV(P)$

We beschouwen nu enkele voorbeelden van substituties:

### Voorbeeld 2

$$\begin{aligned}
 & [u/x]\lambda u.x && (u \in VV(u) \text{ dus S6}) \\
 = & \lambda z.[u/x][z/u]x && (\text{S2}) \\
 = & \lambda z.[u/x]x && (\text{S1}) \\
 = & \lambda z.u
 \end{aligned}$$
  

$$\begin{aligned}
 & [u/x]\lambda u.u && (u \in VV(u) \text{ dus S6}) \\
 = & \lambda z.[u/x][z/u]u && (\text{S1}) \\
 = & \lambda z.[u/x]z && (\text{S2}) \\
 = & \lambda z.z
 \end{aligned}$$
  

$$\begin{aligned}
 & [u/x]\lambda y.x && (\text{S5}) \\
 = & \lambda y.[u/x]x && (\text{S1}) \\
 = & \lambda y.u
 \end{aligned}$$
  

$$\begin{aligned}
 & [u/x]\lambda y.u && (\text{S5}) \\
 = & \lambda y.[u/x]u && (\text{S2}) \\
 = & \lambda y.u
 \end{aligned}$$

### Voorbeeld 3

$$\begin{aligned}
 & [\lambda u.(y)u/x]\lambda y.(x)y && (y \in VV(P) \text{ dus S6}) \\
 = & \lambda z.[\lambda u.(y)u/x][z/y](x)y && (\text{S3}) \\
 = & \lambda z.[\lambda u.(y)u/x]([z/y]x)[z/y]y && (\text{S2}) \\
 = & \lambda z.[\lambda u.(y)u/x](x)[z/y]y && (\text{S1}) \\
 = & \lambda z.[\lambda u.(y)u/x](x)z && (\text{S3}) \\
 = & \lambda z.([\lambda u.(y)u/x]x)[\lambda u.(y)u/x]z && (\text{S1}) \\
 = & \lambda z.(\lambda u.(y)u)[\lambda u.(y)u/x]z && (\text{S2}) \\
 = & \lambda z.(\lambda u.(y)u)z
 \end{aligned}$$

## 2.5 Object symbolen en Meta symbolen

In de  $\lambda$ -calculus zijn we tot nu toe twee soorten symbolen tegengekomen. De eerste soort symbolen zijn de ‘lettertjes’ die deel uitmaken van de  $\lambda$ -calculus. Voorbeelden zijn variabelen, haakjes, de punt (.) en de lambda ( $\lambda$ ). Deze symbolen noemt men *object symbolen*. Ze zitten als het ware *in* het systeem van  $\lambda$ -calculus.

De tweede soort symbolen zijn ‘wiskundige symbolen die we gebruiken om  $\lambda$ -calculus te beschrijven’. Deze symbolen behoren tot ons wiskundig arsenaal en zitten niet echt *in* de  $\lambda$ -calculus. Ze worden gebruikt om *over* het  $\lambda$ -systeem te redeneren. Men noemt ze dan ook *meta symbolen*. Voorbeelden van meta symbolen zijn  $VV()$  en de substitutie-operator  $[./.]$ , die 2  $\lambda$ -expressies en een variabele (allen object symbolen) omzet in een nieuwe  $\lambda$ -expressie.

## 3 Lambda expressies en berekenbaarheid

### 3.1 Beta-gelijkheid

De  $\lambda$ -calculus kan beschouwd worden als een theorie over gelijkheid tussen  $\lambda$ -expressies. De onderstaande definitie definieert deze zogenaamde  $\beta$ -gelijkheid.

**Definitie 5** De relatie  $=_\beta \subset \Lambda \times \Lambda$  wordt gedefinieerd door de onderstaande axiomas:

$$\begin{aligned}
 (\lambda x.M)P &=_\beta [P/x]M && (\beta) \\
 \lambda x.M &=_\beta \lambda z.[z/x]M \text{ als } z \notin VV(M) && (\alpha) \\
 M &=_\beta M && (\text{reflexief}) \\
 M &=_\beta N \Rightarrow N =_\beta M && (\text{symmetrisch}) \\
 M &=_\beta N \wedge N =_\beta L \Rightarrow N =_\beta L && (\text{transitief}) \\
 M &=_\beta M' \wedge P =_\beta P' \Rightarrow (M)P =_\beta (M')P' && (\text{congruent}) \\
 M &=_\beta M' \Rightarrow \lambda x.M =_\beta \lambda x.M'
 \end{aligned}$$

Definitie 5 geeft ons regels die ons toelaten een  $\lambda$ -expressie te herschrijven in een equivalente (hopelijk simpelere)  $\lambda$ -expressie. Het laat ons dus als het ware toe  $\lambda$ -expressies *uit te werken* (te vergelijken met *evalutatie* van expressies in programmeertalen). Het  $\beta$ -axioma<sup>7</sup> geeft onze intuïtie weer over het toepassen van

---

<sup>7</sup>Historisch was  $(\beta)$  het tweede axioma, na  $(\alpha)$ , vanwaar de naam.

een functie op een actuele parameter: het volstaat om in het functievoorschrift de formele parameter te vervangen door de actuele parameter. Het  $\alpha$ -axioma zegt dat men de naam van een formele parameter in een functiedefinitie mag vervangen door een andere.

De volgende drie axoma's zijn nodig om de gelijkheid "netjes" te definieren: ze zorgen ervoor dat de  $\beta$ -gelijkheid zich gedraagt zoals we dat van een gelijkheid verwachten, nl. reflexief, symmetrisch en transitief (de  $\beta$ -gelijkheid bepaalt m.a.w. een equivalentie relatie over de set van  $\lambda$ -expressies).

De laatste twee expressies definiëren de congruentie van de  $\beta$ -gelijkheid. Intuïtief gezien zegt regel 6 (de eerste congruentie regel) dat het in een gegeven applicatie ( $M$ ) $P$  niet uitmaakt of we eerst  $M$  reduceren (m.a.w.  $\beta$ -gelijkheid regels toepassen op  $M$ ), of eerst  $P$  reduceren. De laatste regel (de tweede congruentie regel) zegt dat we, gegeven een  $\lambda$ -expressie van de vorm  $\lambda x.M$ , we de  $\lambda x$ . mogen laten staan, en de  $M$  reduceren.

Laten we een voorbeeld beschouwen waarin we de  $\beta$ -gelijkheidsregels toepassen:

**Voorbeeld 4** In de onderstaande berekening zijn de deelexpressies waarop een axioma wordt toegepast onderlijnd.

$$\begin{aligned}
 & ((\lambda x.\lambda y.\lambda z.((x)z)(y)z)\lambda x.x) \underline{\lambda x.x} \quad (\alpha) \\
 & =_{\beta} ((\lambda x.\lambda y.\lambda z.((x)z)(y)z)\underline{\lambda x.x})\lambda u.u \quad (\alpha) \\
 & =_{\beta} ((\lambda x.\lambda y.\lambda z.((x)z)(y)z)\lambda v.v) \underline{\lambda u.u} \quad (\beta) \\
 & =_{\beta} (\lambda y.\lambda z.(\underline{((\lambda v.v)z)}(y)z)\lambda u.u) \quad (\beta) \\
 & =_{\beta} (\lambda y.\lambda z.(\underline{z})(y)z) \underline{\lambda u.u} \quad (\beta) \\
 & =_{\beta} \lambda z.(\underline{z})(\lambda u.u)z \quad (\beta) \\
 & =_{\beta} \lambda z.(z)z
 \end{aligned}$$

### 3.2 Rekenen met Lambda Expressies

Hoewel de  $\lambda$ -calculus geen constanten kent blijkt het toch mogelijk te zijn om zowel de natuurlijke getallen als de gewone rekenkundige functies voor te stellen d.m.v.  $\lambda$ -expressies, waarbij  $\beta$ -gelijkheid kan gebruikt worden om het gewenste effect van die functies aan te tonen.

Eerst hebben we een hulpdefinitie nodig:

**Definitie 6** *Beschouw  $\lambda$ -expressies  $F$  en  $M$ . De expressie  $(F)^n M$  wordt inductief gedefinieerd door:*

$$\begin{aligned}
 (F)^0 M &\equiv M \\
 (F)^{1+n} M &\equiv (F)(F)^n M
 \end{aligned}$$

Merk op dat voor elke  $n$ ,  $(.)^n$ . een meta operatie is. Zij neemt 2  $\lambda$ -expressies  $F$  en  $M$  en levert de  $\lambda$ -expressie  $(F)(F)(F)\dots(F)M$  op. Deze meta operatie zit niet in de verzameling der  $\lambda$ -expressies. Ze laat ons enkel toe om bepaalde elementen van  $\Lambda$  veel korter op te schrijven. De volgende eigenschap zal later nuttig blijken.

**Lemma 1** *Beschouw  $\lambda$ -expressies  $F$  en  $M$ . Voor alle  $n, m \in \mathbf{N}$  geldt dat*

$$(F)^{n+m}M \equiv (F)^n(F)^mM \quad (2)$$

**Bewijs.** We gebruiken inductie op  $n$ .

1. Als  $n = 0$  dan geldt dat

$$\begin{aligned} (F)^{0+m}M &\equiv (F)^mM \\ &\equiv M' && \text{neem } M' \equiv (F)^mM \\ &\equiv (F)^0M' && \text{definitie 6} \\ &\equiv (F)^0(F)^mM && \text{definitie } M' \end{aligned}$$

2. Stel dat (1) geldt voor  $n = k$ . Als  $n = k + 1$  dan geldt dat

$$\begin{aligned} (F)^{1+k+m}M &\equiv (F)^{1+(k+m)}M \\ &\equiv (F)(F)^{k+m}M && \text{definitie 6} \\ &\equiv (F)(F)^k(F)^mM && \text{inductiehypothese} \\ &\equiv (F)(F)^kM' && \text{neem } M' \equiv (F)^mM \\ &\equiv (F)^{1+k}M' && \text{definitie 6} \\ &\equiv (F)^{1+k}(F)^mM && \text{definitie } M' \end{aligned}$$

**Q.E.D**

Het getal 0 zal nu voorgesteld worden door de  $\lambda$ -expressie  $\lambda f.\lambda x.x$ , het getal 1 door  $\lambda f.\lambda x.(f)x$ , 2 door  $\lambda f.\lambda x.(f)(f)x$ , enz. In het algemeen hebben we de volgende definitie:

**Definitie 7** *De zogenaamde “Church getallen”  $\mathbf{c}_n$  ( $n \in \mathbf{N}$ ) worden gedefinieerd door*

$$\mathbf{c}_n \equiv \lambda f.\lambda x.(f)^n x$$

Intuïtief kunnen we de definitie van een Church getal  $c_n$  lezen als: ”gegeven een functie ( $f$ ) en een parameter ( $x$ ) (aangeduid door de beide formele parameters), dan (het voorschrift volgt) wordt deze functie ( $f$ )  $n$  keer toegepast op  $x$ ”. Onthoud deze intuïtieve interpretatie van een Church getal, ze zal toelaten het vervolg beter te begrijpen. Nu we getallen kunnen voorstellen in  $\lambda$ -calculus kunnen we gaan denken over de voorstelling van operaties op getallen. We zullen dan zeggen dat een operatie op getallen in  $\lambda$ -calculus is voor te stellen ( $\lambda$ -definieerbaar), indien we een combinator kunnen vinden waarvan het effect op Church getallen hetzelfde is als de operatie toegepast op de ‘echte’ getallen die met deze Church getallen overeenkomen. Formeel:

**Definitie 8** *Een numerieke functie  $f : \mathbf{N}^p \rightarrow \mathbf{N}$  met  $p \in \mathbf{N}$  parameters is  $\lambda$ -definieerbaar als er een combinator  $F$  bestaat zodat*

$$(((F)\mathbf{c}_{n_1})\mathbf{c}_{n_2})\dots)\mathbf{c}_{n_p} =_{\beta} \mathbf{c}_{f(n_1, n_2, \dots, n_p)}$$

We zullen eerste aantonen dat de “opvolger” functie

$$\mathbf{succ}(n) = n + 1$$

$\lambda$ -definieerbaar is.

Een juiste definitie van **succ** moet zo zijn dat b.v.

$$(\mathbf{succ})\mathbf{c}_0 \equiv (\mathbf{succ})\lambda f.\lambda x.x =_{\beta} \lambda f.\lambda x.(f)x \equiv \mathbf{c}_1$$

en ook

$$(\mathbf{succ})\mathbf{c}_1 \equiv (\mathbf{succ})\lambda f.\lambda x.(f)x =_{\beta} \lambda f.\lambda x.(f)(f)x \equiv \mathbf{c}_2$$

In het algemeen willen we dus dat  $(\mathbf{succ})\mathbf{c}_n$  een extra “aanroep” van  $f$  toevoegt aan het voorschrift van  $\mathbf{c}_n$ :

$$(\mathbf{succ})\mathbf{c}_n =_{\beta} \lambda f.\lambda x.(f)\text{voorschrift}_n \quad (3)$$

waarbij *voorschrift* <sub>$n$</sub>  een afkorting is voor  $(f)^n x$ .

Maar uit de definitie van een Church getal (herinner u de (intuitieve) betekenis van een Church-getal; we hebben deze niet toevallig zo gedefinieerd):

$$\mathbf{c}_n \equiv \lambda f.\lambda x.(f)^n x$$

geldt dat

$$\begin{aligned} ((\mathbf{c}_n)f)x &\equiv ((\lambda f.\lambda x.(f)^n x)f)x \quad (\beta) \\ &=_{\beta} (\lambda x.(f)^n x)x \quad (\beta) \\ &=_{\beta} (f)^n x \\ &\equiv \text{voorschrift}_n \end{aligned}$$

M.a.w., wanneer we een Church getal nemen (wat uiteraard een  $\lambda$ -expressie is), en we passen die achtereenvolgens toe op  $f$  en  $x$ , dan bekomen we het voorschrift van het Church getal!

We kunnen nu deze kennis gebruiken om (3) te herschrijven als

$$(\mathbf{succ})\mathbf{c}_n =_{\beta} \lambda f.\lambda x.(f)((\mathbf{c}_n)f)x$$

waaruit we de gewenste definitie van **succ** afleiden (waarbij de  $n$  in  $\lambda n$ . de formele parameter is waarin het Church getal zal terechtkomen bij applicatie):

$$\mathbf{succ} \equiv \lambda n.\lambda f.\lambda x.(f)((n)f)x$$

Intuïtief kunnen we **succ** dus lezen als ”gegeven een Church getal ( $\lambda n.$ ), dan geven we een  $\lambda$ -expressie terug die de vorm heeft van een Church getal ( $\lambda f.\lambda x.$ ), waarbij het voorschrift van dit Church getal het oorspronkelijke voorschrift is  $((n)f)x$  met daarvoor een extra aanroep van  $f$ , nl.  $(f)$ ”.

Als voorbeeld berekenen  $(\mathbf{succ})\mathbf{c}_0 =_{\beta} \mathbf{c}_1$  en  $(\mathbf{succ})\mathbf{c}_1 =_{\beta} \mathbf{c}_2$ :

### Voorbeeld 5

$$\begin{aligned}
 (\mathbf{succ})\mathbf{c}_0 &\equiv (\lambda n.\lambda f.\lambda x.(f)((n)f)x)\lambda f.\lambda x.x & (\beta) \\
 &=_{\beta} \lambda f.\lambda x.(f)\underline{((\lambda f.\lambda x.x)f)}x & (\beta) \\
 &=_{\beta} \lambda f.\lambda x.(f)(\lambda x.x)x & (\beta) \\
 &=_{\beta} \lambda f.\lambda x.(f)x \\
 &\equiv \mathbf{c}_1
 \end{aligned}$$

$$\begin{aligned}
 (\mathbf{succ})\mathbf{c}_1 &\equiv (\lambda n.\lambda f.\lambda x.(f)((n)f)x)\lambda f.\lambda x.(f)x & (\beta) \\
 &=_{\beta} \lambda f.\lambda x.(f)\underline{((\lambda f.\lambda x.(f)x)f)}x & (\beta) \\
 &=_{\beta} \lambda f.\lambda x.(f)(\lambda x.(f)x)x & (\beta) \\
 &=_{\beta} \lambda f.\lambda x.(f)(f)x \\
 &\equiv \mathbf{c}_2
 \end{aligned}$$

Eigenlijk is **succ** een bijzonder geval van de meer algemene functie **plus** die de optelling van twee natuurlijke getallen simuleert:

**Stelling 1** *Definieer*

$$\mathbf{plus} \equiv \lambda n.\lambda m.\lambda f.\lambda x.((n)f)((m)f)x$$

dan geldt voor alle  $m, n \in \mathbf{N}$

$$((\mathbf{plus})\mathbf{c}_n)\mathbf{c}_m =_{\beta} \mathbf{c}_{n+m}$$

Opnieuw maken we gebruik van de definitie van een Church getal. Intuïtief lezen we **plus** als "gegeven twee Church getallen (aangeduid door  $\lambda n.\lambda m.$ ), dan geven we een  $\lambda$ -expressie terug die de vorm heeft van een Church getal ( $\lambda f.\lambda x.$ ), waarbij het voorschrift van dit Church getal het oorspronkelijk voorschrift van het tweede Church getal is, nl.  $((m)f)x$  (achteraan) met daarvoor het aantal functie-aanroepen van het eerste Church getal toegevoegd, nl.  $((n)f)$ ". Om in te zien waarom het aantal functie-aanroepen van het eerste Church getal inderdaad netjes voor het voorschrift van het tweede Church getal komt, beschouw volgende reductie:

$$\begin{aligned}
 ((c_n)f)\mathbf{voorschrift}_m &\equiv ((\lambda f.\lambda x.(f)^n x)f)\mathbf{voorschrift}_m & \text{definitie } c_n \\
 &=_{\beta} (\lambda x.(f)^n x)\mathbf{voorschrift}_m & (\beta) \\
 &=_{\beta} (f)^n \mathbf{voorschrift}_m & (\beta) \\
 &\equiv (f)^n (f)^m x & \text{definitie } \mathbf{voorschrift}_m
 \end{aligned}$$

Het formele bewijs is als volgt:

**Bewijs.**

$$\begin{aligned}
((\mathbf{plus})\mathbf{c}_n)\mathbf{c}_m &\equiv ((\lambda n.\lambda m.\lambda f.\lambda x.((n)f)((m)f)x)\mathbf{c}_n)\mathbf{c}_m && \text{definitie } \mathbf{plus} \\
&=_{\beta} (\lambda m.\lambda f.\lambda x.((\mathbf{c}_n)f)((m)f)x)\mathbf{c}_m && (\beta) \\
&\equiv (\lambda m.\lambda f.\lambda x.((\lambda f.\lambda x.(f)^n x)f)((m)f)x)\mathbf{c}_m && \text{definitie } \mathbf{c}_n \\
&=_{\beta} (\lambda m.\lambda f.\lambda x.(\lambda x.(f)^n x)((m)f)x)\mathbf{c}_m && (\beta) \\
&=_{\beta} (\lambda m.\lambda f.\lambda x.(f)^n((m)f)x)\mathbf{c}_m && (\beta) \\
&=_{\beta} \overline{\lambda f.\lambda x.(f)^n((\mathbf{c}_m)f)x} && (\beta) \\
&\equiv \lambda f.\lambda x.(f)^n((\lambda f.\lambda x.(f)^m x)f)x && \text{definitie } \mathbf{c}_m \\
&=_{\beta} \lambda f.\lambda x.(f)^n(\lambda x.(f)^m x)x && (\beta) \\
&=_{\beta} \lambda f.\lambda x.(f)^n(f)^m x && (\beta) \\
&\equiv \lambda f.\lambda x.(f)^{n+m} x && \text{lemma 1} \\
&\equiv \mathbf{c}_{n+m} && \text{definitie } \mathbf{c}_{n+m}
\end{aligned}$$

**Q.E.D**

Men kan ook de Church getallen vermenigvuldigen. Eerst hebben we een hulpstelling nodig.

**Lemma 2** Voor alle  $n, m \in N$  geldt dat

$$((\mathbf{c}_n)f)^m y =_{\beta} (f)^{n \times m} y \quad (4)$$

Intuïtief zegt dit lemma dat, als we een  $\lambda$ -expressie  $((c_n)f)$   $m$  keer herhalen en deze toepassen op  $y$ , dit overeenkomt met  $n * m$  keer de functie  $f$  toe te passen op  $y$ . Feitelijk zorgt de  $((c_n)f)^m$  ervoor dat we  $m$  keer "copieren". Om dit in te zien, beschouw de volgende deductie<sup>8</sup>:

$$\begin{aligned}
((c_n)f)^m y &\equiv ((\lambda f.\lambda x.(f)^n x)f)^m y && \text{definitie } c_n \\
&=_{\beta} (\lambda x.(f)^n x)^m y && (\beta) \\
&\equiv (\lambda x.(f)^n x) \dots (\lambda x.(f)^n x)(\lambda x.(f)^n x)y && \text{definitie } (f)^m \text{ (met } m \text{ herhalingen)} \\
&=_{\beta} (\lambda x.(f)^n x) \dots (\lambda x.(f)^n x)(f)^n y && (\beta) \\
&=_{\beta} (\lambda x.(f)^n x) \dots (f)^n (f)^n y && (\beta)
\end{aligned}$$

Het formele bewijs is als volgt:

**Bewijs.** We gebruiken inductie over  $m$ .

1. Als  $m = 0$  dan wordt (4):

$$\begin{aligned}
((\mathbf{c}_n)f)^m y &\equiv ((\mathbf{c}_n)f)^0 y \\
&\equiv y && \text{definitie 6} \\
&\equiv (f)^0 y && \text{definitie 6} \\
&\equiv (f)^{n \times 0} y
\end{aligned}$$

---

<sup>8</sup>Het gebruik van de  $\dots$  notatie is hieronder niet formeel correct te noemen, doch ze wordt hier ter illustratie van het principe gebruikt

2. Stel dat (4) geldt voor  $m = k$ :

$$\begin{aligned}
 ((\mathbf{c}_n)f)^{k+1}y &\equiv ((\mathbf{c}_n)f)((\mathbf{c}_n)f)^k y && \text{definitie 6} \\
 &\equiv ((\mathbf{c}_n)f)(f)^{n \times k}y && \text{inductiehypothese} \\
 &\equiv ((\lambda f. \lambda x. (f)^n x) f) (f)^{n \times k}y && \text{definitie } \mathbf{c}_n \\
 &= \beta \frac{(\lambda x. (f)^n x) (f)^{n \times k}y}{(f)^n (f)^{n \times k}y} && (\beta) \\
 &= \beta \frac{(f)^n (f)^{n \times k}y}{(f)^{n+n \times k}y} && (\beta) \\
 &\equiv (f)^{n+n \times k}y && \text{lemma 1} \\
 &\equiv (f)^{(n+1) \times k}y
 \end{aligned}$$

**Q.E.D**

We kunnen nu de vermenigvuldiging definieren:

**Stelling 2** Defineer

$$\mathbf{times} \equiv \lambda n. \lambda m. \lambda f. (n)(m)f$$

dan geldt voor alle  $m, n \in \mathbf{N}$

$$((\mathbf{times})\mathbf{c}_n)\mathbf{c}_m = \beta \mathbf{c}_{n \times m}$$

Intuïtief valt bovenstaande stelling makkelijk te begrijpen wanneer we terugdenken aan het "copieer-principe" van Lemma 2, immers (beschouw het voorschrift  $\lambda f. (n)(m)f$ ):

$$\begin{aligned}
 \lambda f. (c_n)(c_m)f &\equiv \lambda f. (\lambda f. \lambda x. (f)^n x) (c_m)f && \text{definitie } c_n \\
 &= \beta \lambda f. \lambda x. ((c_m)f)^n x && (\beta) \\
 &= \beta \lambda f. \lambda x. (f)^{n \times m}x && (\text{lemma 2})
 \end{aligned}$$

Het volledige formeel bewijs is als volgt:

**Bewijs.**

$$\begin{aligned}
 ((\mathbf{times})\mathbf{c}_n)\mathbf{c}_m &\equiv ((\lambda n. \lambda m. \lambda f. (n)(m)f)\mathbf{c}_n)\mathbf{c}_m && \text{definitie } \mathbf{times} \\
 &= \beta (\lambda m. \lambda f. (\mathbf{c}_n)(m)f)\mathbf{c}_m && (\beta) \\
 &\equiv (\lambda m. \lambda f. (\lambda f. \lambda x. (f)^n x)(m)f)\mathbf{c}_m && \text{definitie } \mathbf{c}_n \\
 &= \beta (\lambda m. \lambda f. \lambda x. ((m)f)^n x)\mathbf{c}_m && (\beta) \\
 &= \beta \lambda f. \lambda x. ((\mathbf{c}_m)f)^n x && (\beta) \\
 &= \beta \lambda f. \lambda x. (f)^{m \times n}x && \text{lemma 2} \\
 &\equiv \mathbf{c}_{n \times m} && \text{definitie } \mathbf{c}_{n \times m}
 \end{aligned}$$

**Q.E.D**

Het bedenken van een combinator voor de **minus**-functie is wat gecompliceerder.

We definiëren eerst een paar combinatoren die verder nog van pas zullen komen.

### Definitie 9

<b>true</b>	$\equiv \lambda t. \lambda f. t$
<b>false</b>	$\equiv \lambda t. \lambda f. f$
<b>if</b>	$\equiv \lambda c. \lambda d. \lambda e. ((c)d)e$
<b>iszero</b>	$\equiv \lambda n. ((n) \lambda x. \text{false}) \text{true}$
<b>cons</b>	$\equiv \lambda a. \lambda d. \lambda z. ((z)a)d$
<b>car</b>	$\equiv \lambda a. \lambda d. a$
<b>cdr</b>	$\equiv \lambda a. \lambda d. d$

Eerst en vooral valt op te merken dat bovenstaande combinatoren zo geconstrueerd zijn zodat ze goed met elkaar samenwerken. Zo is bijv. de if-combinator zo geconstrueerd zodat hij goed *true* en *false* samenwerkt. M.a.w., *if* verwacht als eerste argument een  $\lambda$ -expressie die reduceert naar de vorm *true* of *false* zoals hierboven gedefinieerd. Onder deze voorwaarde zal onze *if*-combinator functioneren zoals we dat van een *if* verwachten. Echter, als we als argument een  $\lambda$ -expressie van een andere vorm gebruiken (en er niets in de  $\lambda$ -calculus die ons dat verbiedt), dan zal onze *if*-combinator niet meer naar behoren werken, of beter gezegd, hij zal zich onvoorspelbaar gedragen, en mogelijk niet meer zoals we verwachten dat een *if*-combinator zou moeten werken. In het algemeen is het probleem dat de originele  $\lambda$ -calculus zoals wij die hier bestuderen niet getypeerd is: we kunnen voor een gegeven  $\lambda$ -expressie niet eisen dat hij enkel op welbepaalde  $\lambda$ -expressies toegepast mag worden. Dit heeft zware gevolgen: er werd voor de niet-getypeerde  $\lambda$ -calculus aangetoond dat men bij bepaalde reducties ongewenste resultaten kan bekomen! Verschillende oplossingen werden ontwikkeld om dit probleem op te lossen, waaronder getypeerde vormen van  $\lambda$ -calculus doch een discussie over de specifieke problemen met de niet-getypeerde  $\lambda$ -calculussen de oplossingen zou ons voor het doel van deze cursus veel te ver leiden. Je mag aldus aannemen dat, zolang we onze combinatoren "correct" gebruiken (m.a.w. we ze gebruiken zoals ze bedoeld zijn, en onze combinatoren dus niet toepassen op andere, ongewenste  $\lambda$ -expressies), we niet in de problemen zullen komen.

Untuift moeten we bovenstaande combinatoren als volgt begrijpen:

- **true**: gegeven twee argumenten ( $\lambda t.$  en  $\lambda f.$ ), dan (het voorschrift volgt) geeft **true** het eerste argument terug.
- **false**: gegeven twee argumenten ( $\lambda t.$  en  $\lambda f.$ ), dan (het voorschrift volgt) geeft **false** het tweede argument terug.
- **if**: gegeven een conditie ( $\lambda c.$ ), een *true* en een *else* take (respektievelijk  $\lambda d.$  en  $\lambda e.$ ), dan (het voorschrift  $((c)d)e$  volgt) passen we de conditie toe op (achtereenvolgend) beide argumenten *d* en *e*. Gezien **true** het eerste argument teruggeeft, en **false** het tweede, zal de **if** combinator inderdaad werken zoals we dat verwachten. Hier komt dus duidelijk tot uiting dat **true** en **false** zo geconstrueerd zijn zodat ze met **if** samenwerken.

- **iszzero:** gegeven een Church getal ( $\lambda n.$ ), dan (het voorschrift volgt) passen we het Church getal  $n$  toe op de argumenten  $\lambda x.\text{false}$  en **true**. Het eerste argument  $\lambda x.\text{false}$  is een  $\lambda$ -expressie die, om het even wat het argument is, steeds **false** teruggeeft. Herinner u nu de (intuitieve) definitie van een Church getal: gegeven een functie en een parameter, dan wordt die functie  $n$  keer toegepast op deze parameter. De functie is hier  $\lambda x.\text{false}$  (die steeds **false** teruggeeft!), het argument **true**. M.a.w., vanaf het moment dat  $\lambda x.\text{false}$  uitgevoerd wordt, dan zal het resultaat **false** zijn. Er is slecht één geval waarin deze functie *niet* zal uitgevoerd worden, nl. als het Church getal ( $n$ )  $c_0$  was (dit is immers de definitie van  $c_n$ , nl.  $\lambda f.\lambda x.x$ ): in dit geval krijgen we dus het tweede argument van  $n$  terug, nl. **true**.
- **cons:** gegeven twee argumenten ( $\lambda a.$ , de car, en  $\lambda d.$ , de cdr), dan (het voorschrift volgt) geeft **cons** een  $\lambda$ -expressie terug die één argument verwacht ( $\lambda z.$ ) en, wanneer toegepast, dit argument zal toepassen op op de beide (eerdere) argumenten car  $a$  en cdr  $d$ . Een cons cell in de  $\lambda$ -calculus is dus eigenlijk een dispatch-functie die, gegeven de juiste functie (nl. car of cdr), het eerste of tweede argument zal teruggeven (bekijk aandachtig voorbeeld 8 om dit volledig te begrijpen). **car** en **cdr** worden dus ook als dusdanig geconstrueerd (zie volgend).
- **car:** gegeven twee argumenten ( $\lambda a.$  en  $\lambda d.$ ), dan (het voorschrift volgt) geeft **car** het eerste argument (de **car**) terug.
- **cdr:** gegeven twee argumenten ( $\lambda a.$  en  $\lambda d.$ ), dan (het voorschrift volgt) geeft **cdr** het eerste argument (de **cdr**) terug.

Merk op dat **car** en **cdr** identiek gedefinieerd zijn als **true** en **false**. De combinatoren van definitie 9 gedragen zich zoals verwacht; zie hier enkele voorbeelden:

#### Voorbeeld 6

$$\begin{array}{ll}
 ((\mathbf{true})A)B & \equiv ((\lambda t.\lambda f.t)A)B \quad \text{voor willekeurige } A, B \in \Lambda \\
 & =_{\beta} (\lambda f.A)B \\
 & =_{\beta} A
 \end{array}$$
  

$$\begin{array}{ll}
 ((\mathbf{false})A)B & \equiv ((\lambda t.\lambda f.f)A)B \quad \text{voor willekeurige } A, B \in \Lambda \\
 & =_{\beta} (\lambda f.f)B \\
 & =_{\beta} B
 \end{array}$$
  

$$\begin{array}{ll}
 (\mathbf{if}\mathbf{true}) & \equiv (\lambda c.\lambda d.\lambda e.((c)d)e)\mathbf{true} \\
 & =_{\beta} \lambda d.\lambda e.((\mathbf{true})d)e \\
 & \equiv \lambda d.\lambda e.((\lambda t.\lambda f.t)d)e \\
 & =_{\beta} \lambda d.\lambda e.(\lambda f.d)e \\
 & =_{\beta} \lambda d.\lambda e.d \\
 & \equiv \mathbf{true}
 \end{array}$$

### Voorbeeld 7

$$\begin{aligned}
 (\text{true})\text{false} &\equiv \underline{(\lambda t.\lambda f.t)\text{false}} \\
 &=_{\beta} \underline{\lambda f.\text{false}} \\
 (\lambda f.\text{false})^n M &\equiv \underline{(\lambda f.\text{false})(\lambda f.\text{false})^{n-1} M} \quad \text{als } n > 0 \\
 &=_{\beta} \underline{\text{false}}
 \end{aligned}$$
  

$$\begin{aligned}
 (\text{iszero})\mathbf{c}_0 &\equiv \underline{(\lambda n.((n)\lambda.\text{false})\text{true})\mathbf{c}_0} \\
 &=_{\beta} \underline{((\mathbf{c}_0)\lambda.\text{false})\text{true}} \\
 &\equiv \underline{((\lambda f.\lambda x.x)\lambda.\text{false})\text{true}} \\
 &=_{\beta} \underline{(\lambda x.x)\text{true}} \\
 &=_{\beta} \underline{\text{true}}
 \end{aligned}$$
  

$$\begin{aligned}
 (\text{iszero})\mathbf{c}_n &\equiv \underline{(\lambda n.((n)\lambda.\text{false})\text{true})\mathbf{c}_n} \quad \text{stel } n > 0 \\
 &=_{\beta} \underline{((\mathbf{c}_n)\lambda f.\text{false})\text{true}} \\
 &\equiv \underline{((\lambda f.\lambda x.(f)^n x)\lambda f.\text{false})\text{true}} \\
 &=_{\beta} \underline{(\lambda x.(\lambda f.\text{false})^n x)\text{true}} \\
 &=_{\beta} \underline{(\lambda x.\text{false})\text{true}} \\
 &=_{\beta} \underline{\text{false}}
 \end{aligned}$$

De combinatoren **car** en **cdr** kunnen gebruikt worden om een component van een paar  $((\text{cons})A)B$  te selecteren:

### Voorbeeld 8

$$\begin{aligned}
 ((\text{cons})A)B &\equiv \underline{((\lambda a.\lambda d.\lambda z.((z)a)d)A)B} \\
 &=_{\beta} \underline{(\lambda d.\lambda z.((z)A)d)B} \\
 &=_{\beta} \underline{\lambda z.((z)A)B} \\
 \\ 
 (((\text{cons})A)B)\text{car} &=_{\beta} \underline{(\lambda z.((z)A)B)\text{car}} \\
 &=_{\beta} \underline{((\text{car})A)B} \\
 &\equiv \underline{((\lambda a.\lambda d.a)A)B} \\
 &=_{\beta} \underline{(\lambda d.A)B} \\
 &=_{\beta} \underline{A}
 \end{aligned}$$
  

$$\begin{aligned}
 (((\text{cons})A)B)\text{cdr} &=_{\beta} \underline{(\lambda z.((z)A)B)\text{cdr}} \\
 &=_{\beta} \underline{((\text{cdr})A)B} \\
 &\equiv \underline{((\lambda a.\lambda d.d)A)B} \\
 &=_{\beta} \underline{(\lambda d.d)B} \\
 &=_{\beta} \underline{B}
 \end{aligned}$$

Met behulp van de voorgaande definities kunnen we nu een “voorganger” functie **pred** definieren waarbij

$$\text{pred}(n) = n - 1 \text{ als } n > 0$$

Het idee is om paren van de vorm  $(c_n, c_{n-1})$  aan te maken:

$$((\text{cons})\mathbf{c}_n)\mathbf{c}_{n-1})$$

Als we erin slagen zo'n paren te vormen, dan kunnen we **pred** van  $c_n$  definieeren als de **cdr** van het paar  $(c_n, c_{n-1})$ :

$$(((\mathbf{cons})\mathbf{c}_n)\mathbf{c}_{n-1})\mathbf{cdr}$$

De uitdaging is dus om de paren  $(c_n, c_{n-1})$  te definieeren (of beter gezegd *construeren*). Een eerste stap hiertoe is een hulpfunctie **nextp** te definieeren die, gegeven een paar  $(c_n, c_{n-1})$ , ons het "volgende paar"  $(c_{n+1}, c_n)$  teruggeeft:

$$(\mathbf{nextp})((\mathbf{cons})\mathbf{c}_n)\mathbf{c}_{n-1} =_\beta ((\mathbf{cons})\mathbf{c}_{n+1})\mathbf{c}_n$$

De combinator **nextp** is eenvoudig te construeren: het volstaat de eerste component (m.a.w. de **car**) van het argument (een **cons** cell) naar de tweede component in het resultaat te brengen, en de opvolger (**succ**) van de eerste component (de **car**) van het argument (de **cons** cell) in de eerste component van het resultaat te plaatsen:

$$\begin{aligned}\mathbf{nextp} &\equiv \lambda p.((\mathbf{cons})(\mathbf{succ})(p)\mathbf{car})(p)\mathbf{car} \\ &\equiv \lambda p.\lambda z.((z)(\lambda n.\lambda f.\lambda x.(f)((n)f)x)(p)\mathbf{car})(p)\mathbf{car}\end{aligned}$$

Om nu een cons cell  $(c_n, c_{n-1})$  te construeren, volstaat het om, vertrekend vanuit  $(c_0, c_0)$ ,  $n$  keer **nextp** toe te passen. Zoals we reeds weten kunnen we een functie  $g$   $n$  keer itereren door gebruik te maken van (de definitie van) het Church getal  $\mathbf{c}_n$ , vermits b.v.

$$\begin{aligned}((\mathbf{c}_n)g)y &\equiv ((\lambda f.\lambda x.(f)^n x)g)y \\ &=_\beta (\lambda x.(g)^n x)y \\ &=_\beta (g)^n y\end{aligned}$$

We kunnen aldus  $(c_n, c_{n-1})$  genereren door:

$$((c_n)\mathbf{nextp})((\mathbf{cons})\mathbf{c}_0)\mathbf{c}_0$$

Uiteindelijk zijn we klaar om **pred** te definieeren, nl. als volgt

$$\mathbf{pred} \equiv \lambda n.(((n)\mathbf{nextp})((\mathbf{cons})\mathbf{c}_0)\mathbf{c}_0)\mathbf{cdr} \quad (5)$$

Indien we alle afkortingen in (5) uitschrijven bekomen we

$$\mathbf{pred} \equiv \lambda n.(\lambda t.\lambda f.f)((n)\lambda p.\lambda z.((z)(\lambda n.\lambda f.\lambda x.(f)((n)f)x) \\ (p)\lambda t.\lambda f.t)(p)\lambda t.\lambda f.t)\lambda z.((z)\lambda f.\lambda x.x)\lambda f.\lambda x.x$$

wat welliswaar niet erg leesbaar is.

Met het gebruik van **pred** wordt de definitie van **minus** eenvoudig: om  $n - m$  uit te rekenen volstaat het om **pred**  $m$  keer te itereren op  $\mathbf{c}_n$ :

$$\mathbf{minus} \equiv \lambda m.\lambda n.((m)\mathbf{pred})n$$

### 3.3 Fixpunten en recursie

We hebben in de vorige sectie reeds heel wat begrippen uit programmeertalen gedefinieerd, o.a. getallen, bepaalde arithmetische functies (v.b.  $+$ ,  $-$ ,  $*$ ), booleanse waarden (*true* en *false*), if-expressies, cons-cellen met bijhorende operaties (*car* en *cdr*). Toch missen we nog één fundamenteel concept aanwezig in praktisch alle programmeertalen: recursie. Alvorens we echter kunnen overgaan tot het definiëren van recursieve functies in  $\lambda$ -calculus, hebben we een ander stukje cruciale theorie nodig, nl. deze van fixpunten. Een fixpunt (of dekpunt) van een functie kennen we uit de wiskunde: het is een punt dat, gegeven als input voor een functie, zichzelf als resultaat oplevert. Zo is bijv. 0 een fixpunt voor de functie  $f(x) = 2x$ , gezien  $f(0) = 0$ . Een fixpunt kan als volgt gedefinieerd worden:

**Definitie 10** Voor een functie

$$f : D \rightarrow D$$

heet  $x \in D$  een **fixpunt** van  $f$  als en slechts als

$$f(x) = x$$

Voor  $\lambda$ -expressies wordt dit:

**Definitie 11**  $X \in \Lambda$  is een fixpunt van  $F \in \Lambda$  als en slechts als

$$(F)X =_{\beta} X$$

Men kan zich nu afvragen of er überhaupt  $\lambda$ -expressies zijn die een fixpunt hebben, en als dat zo is, welke dit dan zijn en wat hun fixpunt is. Het antwoord op deze vragen werd ontdekt door Haskell B. Curry, en kan enigzins verrassend lijken: niet alleen is het zo dat iedere  $\lambda$ -expressie een fixpunt heeft, er bestaat bovendien een combinator die, gegeven een willekeurige  $\lambda$ -expressie, voor ons een fixpunt berekent! Dit wordt weergegeven in de volgende stelling:

**Stelling 3**

1. Iedere  $\lambda$ -expressie heeft een fixpunt:

$$\forall F \in \Lambda \exists X : (F)X =_{\beta} X$$

2. Er is een **fixpunt combinator**

$$\mathbf{Y} \equiv \lambda f.(\lambda x.(f)(x)x)\lambda x.(f)(x)x$$

waarvoor geldt dat

$$\forall F \in \Lambda : (F)(\mathbf{Y})F =_{\beta} (\mathbf{Y})F$$

### Bewijs.

1. Definieer

$$W \equiv \lambda x.(F)(x)x$$

en

$$X \equiv (W)W$$

Dan geldt dat

$$\begin{aligned} X &\equiv (W)W \\ &\equiv (\lambda x.(F)(x)x)W \\ &=_{\beta} (F)(W)W \\ &=_{\beta} (F)X \end{aligned}$$

en dus is  $X$  een fixpunt van  $F$ .

2.

$$\begin{aligned} (\mathbf{Y})F &\equiv (\lambda f.(\lambda x.(f)(x)x)\lambda x.(f)(x)x)F \\ &=_{\beta} (\lambda x.(F)(x)x)\lambda x.(F)(x)x \\ &=_{\beta} (W)W \\ &=_{\beta} X \end{aligned}$$

Samen met punt (1) hierboven toont dit aan dat  $(\mathbf{Y})F$  een fixpunt is van  $F$ .

**Q.E.D**

Gegeven een willekeurige  $\lambda$ -expressie  $M$ , is het dus een koud kunstje om een fixpunt voor deze  $\lambda$ -expressie te berekenen: we passen simpelweg de  $\mathbf{Y}$  combinator toe op deze  $\lambda$ -expressie:  $(\mathbf{Y})M$ . We zijn nu klaar om recursie te definiëren. Beschouw het volgende Scheme fragment dat de faculteitsfunctie berekent d.m.v. een recursieve functie  $fac$ .

```
(define fac (lambda n)
  (if (= x 0)
      1
      (* x (fac (- x 1))))))
```

Dit kan zonder veel moeite omgezet worden naar een  $\lambda$ -expressie:

$$fac \equiv \lambda n.(((\mathbf{if})(\mathbf{iszzero})n)\mathbf{c}_1)((\mathbf{times})n)(\mathbf{fac})(\mathbf{pred})n \quad (6)$$

Nochthans is dit geen geldige definitie omdat ze circulair is: wat gedefinieerd wordt (**fac**) wordt ook gebruikt in de definitie. We zijn als het ware een oneindig lang woord aan het definiëren. De vraag is nu of we toch het effect van een recursieve definitie op een natuurlijke manier kunnen simuleren in de  $\lambda$ -calculus.

De oplossing bestaat er in om de identiteit ( $\equiv$ ) in (6) te vervangen door  $\beta$ -gelijkheid ( $=_{\beta}$ ):

$$fac =_{\beta} \lambda n.(((\mathbf{if})(\mathbf{iszzero})n)\mathbf{c}_1)((\mathbf{times})n)(\mathbf{fac})(\mathbf{pred})n \quad (7)$$

We zoeken dus een  $\lambda$ -expressie **fac** die zodanig is dat (7) geldt. Echter, als (7) geldt, en we maken gebruik van het feit dat voor een willekeurige  $\lambda$ -expressie  $M$  geldt dat  $\lambda x.(M)x =_{\beta} (\lambda f.\lambda x.(f)x)M$ , dan moet ook:

$$\mathbf{fac} =_{\beta} (\lambda f.\lambda n.(((\mathbf{if})(\mathbf{iszero})n)\mathbf{c}_1)((\mathbf{times})n)(f)(\mathbf{pred})n)\mathbf{fac} \quad (8)$$

gelden.

Mits we nu in (8) de volgende vervanging doorvoeren (en merk op dat (9) niet circulair is!):

$$\mathbf{FAC} \equiv \lambda f.\lambda n.(((\mathbf{if})(\mathbf{iszero})n)\mathbf{c}_1)((\mathbf{times})n)(f)(\mathbf{pred})n \quad (9)$$

dan kan (8) met behulp van (9) herschreven worden als:

$$\mathbf{fac} =_{\beta} (\mathbf{FAC})\mathbf{fac}$$

M.a.w. de gezochte  $\lambda$ -expressie **fac** is een fixpunt van de functie **FAC**!! Zoals eerder reeds vermeld hebben we een manier om, gegeven een  $\lambda$ -expressie (in dit geval, **FAC**), een fixpunt voor deze  $\lambda$ -expressie te berekenen, nl. door gebruik te maken van de **Y** combinator, nl.

$$\mathbf{fac} \equiv (\mathbf{Y})\mathbf{FAC}$$

We zijn er dus in geslaagd om **fac**, een recursieve functie, te definieren in de  $\lambda$ -calculus!

Het volgende voorbeeld illustreert hoe een recursieve functie die d.m.v. de **Y** operator werd gedefinieerd kan toegepast worden.

**Voorbeeld 9** We definieren **fac** en  $W_{\mathbf{FAC}}$  door

$$\begin{aligned} \mathbf{FAC} &\equiv \lambda f.\lambda n.(((\mathbf{if})(\mathbf{iszero})n)\mathbf{c}_1)((\mathbf{times})n)(f)(\mathbf{pred})n \\ \mathbf{Y} &\equiv \lambda f.(\lambda x.(f)(x)x)\lambda x.(f)(x)x \\ \mathbf{fac} &\equiv (\mathbf{Y})\mathbf{FAC} \\ W_{\mathbf{FAC}} &\equiv \lambda x.(\mathbf{FAC})(x)x \end{aligned}$$

Dan geldt

$$\begin{aligned}
(\mathbf{fac})\mathbf{c}_0 &\equiv ((\mathbf{Y})\mathbf{FAC})\mathbf{c}_0 \\
&\equiv ((\lambda f.(\lambda x.(f)(x)x)\lambda x.(f)(x)x)\mathbf{FAC})\mathbf{c}_0 \\
&=_{\beta} ((\lambda x.(\mathbf{FAC})(x)x)\lambda x.(\mathbf{FAC})(x)x)\mathbf{c}_0 \\
&\equiv ((\lambda x.(\mathbf{FAC})(x)x)W_{\mathbf{FAC}})\mathbf{c}_0 \\
&=_{\beta} ((\mathbf{FAC})(W_{\mathbf{FAC}})W_{\mathbf{FAC}})\mathbf{c}_0 \\
&\equiv \underline{((\lambda f.\lambda n.(((\mathbf{if})(\mathbf{iszero})n)\mathbf{c}_1)((\mathbf{times})n)(f)(\mathbf{pred})n)(W_{\mathbf{FAC}})W_{\mathbf{FAC}})}\mathbf{c}_0 \quad ll \\
&=_{\beta} \underline{((\lambda n.(((\mathbf{if})(\mathbf{iszero})n)\mathbf{c}_1)((\mathbf{times})n)((W_{\mathbf{FAC}})W_{\mathbf{FAC}})(\mathbf{pred})n)\mathbf{c}_0} \\
&=_{\beta} (((\mathbf{if})(\mathbf{iszero})\mathbf{c}_0)\mathbf{c}_1)((\mathbf{times})\mathbf{c}_0)((W_{\mathbf{FAC}})W_{\mathbf{FAC}})(\mathbf{pred})\mathbf{c}_0 \\
&=_{\beta} (((\mathbf{if})\mathbf{true})\mathbf{c}_1)((\mathbf{times})\mathbf{c}_0)((W_{\mathbf{FAC}})W_{\mathbf{FAC}})(\mathbf{pred})\mathbf{c}_0 \\
&=_{\beta} ((\mathbf{true})\mathbf{c}_1)((\mathbf{times})\mathbf{c}_0)((W_{\mathbf{FAC}})W_{\mathbf{FAC}})(\mathbf{pred})\mathbf{c}_0 \\
&=_{\beta} \underline{(\lambda f.\mathbf{c}_1)((\mathbf{times})\mathbf{c}_0)((W_{\mathbf{FAC}})W_{\mathbf{FAC}})(\mathbf{pred})\mathbf{c}_0} \\
&=_{\beta} \mathbf{c}_1
\end{aligned}$$

Merk op dat we bij het toepassen van  $(\beta)$  telkens de “juiste” deelexpressie kozen: b.v. hebben we in

$$(((\mathbf{if})(\mathbf{iszero})\mathbf{c}_0)\mathbf{c}_1)((\mathbf{times})\mathbf{c}_0)((W_{\mathbf{FAC}})W_{\mathbf{FAC}})(\mathbf{pred})\mathbf{c}_0$$

$(\beta)$  niet toegepast op  $(W_{\mathbf{FAC}})W_{\mathbf{FAC}}$ . Indien we dat wel gedaan hadden dan zou ons dit een nutteloze extra expansie van het voorschrift van **FAC** opgeleverd hebben.

Verder blijkt uit het voorbeeld dat we het fixpunt

$$X_{\mathbf{FAC}} \equiv (\mathbf{Y})\mathbf{FAC} \equiv (\mathbf{Y})\lambda f.\mathbf{voorschrift}_{\mathbf{FAC}}$$

kunnen gebruiken om het voorschrift  $\{X_{\mathbf{FAC}}/f\}\mathbf{voorschrift}_{\mathbf{FAC}}$  van **FAC** te realizeren waarbij het fixpunt  $X_{\mathbf{FAC}}$  zelf weer beschikbaar is in het voorschrift en eventueel verder kan geëxpandeerd worden ten behoeve van een recursieve toepassing.

Het bovenstaande kan veralgemeend worden.

**Gevolg 1** *Beschouw een combinator F van de vorm*

$$F \equiv \lambda f.\mathbf{voorschrift}_F$$

*en zijn fixpunt*

$$X_F \equiv (\mathbf{Y})F$$

*Dan geldt dat*

$$X_F =_{\beta} \{X_F/f\}\mathbf{voorschrift}_F$$

**Bewijs.** We weten dat  $X_F$  een fixpunt is van  $F$  en dus:

$$(F)X_F =_{\beta} X_F$$

Nu is

$$(F)X_F \equiv (\lambda f.\mathbf{voorschrift}_F)X_F =_{\beta} \{X_F/f\}\mathbf{voorschrift}_F$$

**Q.E.D**

In het volgende voorbeeld wordt dit toegepast.

**Voorbeeld 10** **FAC** en **fac** zijn gedefinieerd zoals in voorbeeld 9. Merk op dat

$$\text{voorschrift}_{\text{FAC}} \equiv \lambda n. (((\text{if})(\text{iszero})n)\mathbf{c}_1)((\text{times})n)(f)(\text{pred})n$$

Er geldt:

$$\begin{aligned}
 (\mathbf{fac})\mathbf{c}_1 &=_{\beta} (\lambda n. (((\text{if})(\text{iszero})n)\mathbf{c}_1)((\text{times})n)(\mathbf{fac})(\text{pred})n)\mathbf{c}_1 && \text{gevolg 1} \\
 &=_{\beta} (((\text{if})(\text{iszero})\mathbf{c}_1)\mathbf{c}_1)((\text{times})\mathbf{c}_1)(\mathbf{fac})(\text{pred})\mathbf{c}_1 \\
 &=_{\beta} (((\text{if})\text{false})\mathbf{c}_1)((\text{times})\mathbf{c}_1)(\mathbf{fac})(\text{pred})\mathbf{c}_1 \\
 &=_{\beta} ((\text{false})\mathbf{c}_1)((\text{times})\mathbf{c}_1)(\mathbf{fac})(\text{pred})\mathbf{c}_1 \\
 &=_{\beta} (\overline{\lambda f. f})((\text{times})\mathbf{c}_1)(\mathbf{fac})(\text{pred})\mathbf{c}_1 \\
 &=_{\beta} ((\text{times})\mathbf{c}_1)\underline{(\mathbf{fac})(\text{pred})\mathbf{c}_1} \\
 &=_{\beta} ((\text{times})\mathbf{c}_1)\underline{(\mathbf{fac})}\mathbf{c}_0 \\
 &=_{\beta} ((\text{times})\mathbf{c}_1)\overline{(\lambda n. (((\text{if})(\text{iszero})n)\mathbf{c}_1)((\text{times})n)(\mathbf{fac})(\text{pred})n)}\mathbf{c}_0 && \text{gevolg 1 } lll \\
 &=_{\beta} ((\text{times})\mathbf{c}_1)((\text{if})\text{true})\mathbf{c}_1)((\text{times})\mathbf{c}_0)(\mathbf{fac})(\text{pred})\mathbf{c}_0 \\
 &=_{\beta} ((\text{times})\mathbf{c}_1)((\text{true})\mathbf{c}_1)((\text{times})\mathbf{c}_0)(\mathbf{fac})(\text{pred})\mathbf{c}_0 \\
 &=_{\beta} ((\text{times})\mathbf{c}_1)\overline{(\lambda f. \mathbf{c}_1)}((\text{times})\mathbf{c}_0)(\mathbf{fac})(\text{pred})\mathbf{c}_0 \\
 &=_{\beta} ((\text{times})\mathbf{c}_1)\mathbf{c}_1 \\
 &=_{\beta} \mathbf{c}_1
 \end{aligned}$$

# Oefeningen

1. Een alfabet is een verzameling symbolen, bvb.  $\{a, b, c\}$  of  $\{1, 2, 3\}$  of  $\{\text{koe}, \text{paard}, \text{varken}\}$ . Als  $\Sigma$  een alfabet is, dan is  $\Sigma^i$  de verzameling van alle woorden over  $\Sigma$  van lengte  $i$ .

$$\Sigma^0 = \{\epsilon\}$$

$$\Sigma^1 = \Sigma$$

$$\Sigma^2 = \Sigma \times \Sigma$$

$$\Sigma^3 = \Sigma \times \Sigma \times \Sigma$$

Algemeen:

$$\Sigma^0 = \{\epsilon\}$$

$$\Sigma^i = \Sigma^{i-1} \times \Sigma$$

$$\Sigma^* = \bigcup_{i \geq 0} \Sigma^i$$

Gegeven  $\Sigma = \{a, b\}$ . Bereken  $\Sigma^0, \Sigma^1, \Sigma^2, \Sigma^3$ .

2. Op  $\Sigma^*$  is een ‘concatenatie’ operatie gedefineerd:

$$\cdot : \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$$

die 2 woorden over  $\Sigma^*$  aaneenplakt (concateneert). Bvb.,  $abc.de = abcde$ . Verder is, voor  $w \in \Sigma^*$ ,  $w^r$  de omgekeerde versie van  $w$ .

- (a) Defineer  $\cdot^r$ .
- (b) Bereken  $(abcd)^r$  volgens de definitie.
- (c) Toon aan dat  $(wa)^r = aw^r$ .
- (d) Toon aan dat  $w^{r^r} = w$ .

3. Schrijf het alfabet van de  $\lambda$ -calculus op.

4. Schrijf het alfabet van de propositielogica op.

5. Definieer  $d : \Sigma^* \times \Sigma \rightarrow \Sigma^*$  door

$$d(\epsilon, a) = \epsilon$$

$$d(a, a) = \epsilon$$

$$d(b, a) = b \text{ als } b \neq a$$

$$d(cx, a) = d(c, a)d(x, a)$$

en dit  $\forall a, b, c \in \Sigma$  en  $\forall x \in \Sigma^*$

- (a) Wat doet  $d$  informeel ?
- (b) Toon aan dat  $d$  en  $\cdot^r$  commuteren:  $(d(w, a))^r = d(w^r, a), \forall w \in \Sigma^*$  en  $\forall a \in \Sigma$ .

- (c) Toon aan dat  $d$  commutatief is in haar 2de argument:  $d(d(w, a), b) = d(d(w, b), a)$  voor alle  $w \in \Sigma^*$  en voor alle  $a, b \in \Sigma$ .
6. Bekijk de definitie van de propositiesformules  $PROP \subset \Sigma^*$ .
- Waarvoor dient de afsluitende stap ?
  - Zijn de volgende formules geldige formules? Zo ja, teken hun constructieboom. Zo neen, zeg wat er fout loopt.
    - $((p \rightarrow q) \vee \neg r)$
    - $(p \vee q \vee r)$
    - $((p \vee q) \wedge r) \rightarrow (r \vee \neg p)$
7. Geef de volgende zinnen weer in propositionele notatie:
- 'Als de bus niet komt, komen de tram en de trein'
  - 'Als de tram komt als de trein niet komt, dan komen de trein en de bus niet allebei'
8. Geef alle formules die met haakjes invoegen te maken zijn uit  $p \wedge \neg q \rightarrow r$ , met bijbehorende constructiebomen.
9. Kan je een efficiëntere representatie bedenken voor constructiebomen. Doe dit voor de formules uit oefening 6b.
10. (a) Definieer inductief rijen gebalanceerde haakjes.  
 (b) Definieer op een inductieve manier een syntax voor natuurlijke rekenkundige expressies. Maak gebruik van  $\text{IN}$  en van  $\{(, ), +, -, *, /\}$ . Teken bomen in de stijl van oefening 9 voor volgende expressies.
  - $(3 + 4) * 7$
  - $((8 - 5) + (6/2))$
11. Schrijf de volgende formules op in prefix en postfix notatie.
- $(1 + ((2 * 7) - 5))$
  - $((3 + 4) * (8/2))$
  - $((q \leftrightarrow r) \rightarrow p) \wedge \neg p$
12. Voer volgende substituties met de hand uit
- $[(q \leftrightarrow r)/p]\neg p$
  - $[(q \rightarrow s)/p](p \rightarrow q)$
  - $[((q \vee s) \rightarrow \neg p)/r](p \rightarrow p)$
13. Welke getallen  $x$  voldoen aan de volgende condities :
- $(\neg(p \wedge q) \wedge r)$
  - $((\neg p \wedge q) \wedge r)$
  - $\neg(p \wedge (q \wedge r))$
- waarbij  $p = "x \leq 1"$ ,  $q = "x \leq 3"$  en  $r = "x \geq 2"$ .

14. Zij  $v$  de waardering gedefinieerd door  $v(p) = 0$  en  $v(q) = 1$ .
- Wat is  $v(p \wedge q)$  en  $v(p \uparrow q)$  ?
  - Toon aan waar het boek te kort schiet in de definitie van semantiek.
  - Probeer de exacte definities te formuleren.
  - Geef exacte semantiek aan de standaardconnectieven.
  - Bereken  $v(((p \wedge q) \rightarrow r))$  als  $v(p) = 0$ ,  $v(q) = 1$  en  $v(r) = 0$ .
15. Geef waarheidstabellen op twee manieren voor
- $(\neg p \vee \neg q)$
  - $p \wedge (q \wedge p)$
  - $(\neg p \wedge (\neg q \wedge r)) \vee ((q \wedge r) \vee (p \wedge r))$
16. We voeren een nieuw connectief  $\Delta$  in waarvan de betekenis door de volgende waarheidstabel is gegeven:
- | $p$ | $q$ | $p\Delta q$ |
|-----|-----|-------------|
| 0   | 0   | 0           |
| 0   | 1   | 1           |
| 1   | 0   | 1           |
| 1   | 1   | 0           |
- Vind nu een propositie waarin alleen  $p, q$  en de connectieven  $\neg, \wedge, \vee$  voorkomen, die dezelfde eindkolom in een waarheidstabel oplevert.
  - Wat is de intuïtieve betekenis van  $\Delta$ ?
  - Druk ook  $\rightarrow$  en  $\leftrightarrow$  uit m.b.v.  $\neg, \wedge, \vee$ .
17. Rekenen binnen een computer gebeurt binair. Zij  $(p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8)$  en  $(q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8)$  twee binaire getallen (laten we afspreken dat  $p_1$  en  $q_1$  de minst-significante cijfers voorstellen). Net zoals gewoon decimaal optellen maakt binair optellen gebruik van twee mechanismen: één mechanisme om een reeks cijfers te combineren tot een nieuw cijfer (bvb voor  $5 + 8$  is het nieuwe cijfer 3) en een ander mechanisme om te weten wat de overdracht is (bvb  $5 + 8$  heeft als overdracht 1, t.t.z. “één onthouden”). Net zoals bij decimaal rekenen zowel het resulterend cijfer als de overdracht begrensd zijn door 9, is bij binair rekenen de uitkomst en de overdracht beperkt door 1.

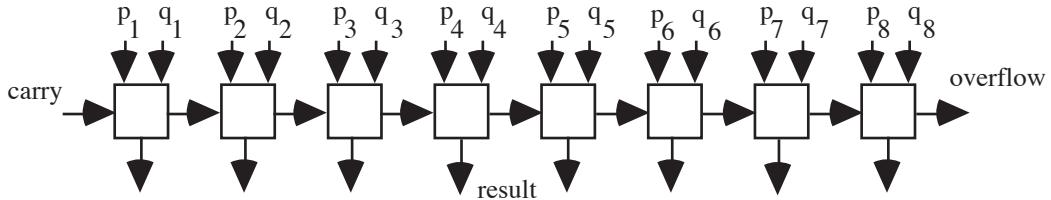


Figure 1: De full adder

- (a) Vind een logisch connectief om het cijfer bij de optelling van twee bits  $p$  en  $q$  te bepalen.
- (b) Vind een logisch connectief om de overdracht bij optelling van 2 bits te bepalen.
- (c) Bovenstaande schakeling (de “full adder”) rekent de som uit van 2 binaire getallen  $(p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8)$  en  $(q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8)$ . Geef voor iedere digit van het resultaat een logische formule en geef een logische formule voor de overflow. Er zal natuurlijk steeds gerekend worden met  $v(c_1) = 0$

18. Bepaal  $MOD(\varphi)$  voor volgende formules:

- (a)  $\varphi = ((\neg p \vee q) \leftrightarrow (p \rightarrow q))$
- (b)  $\varphi = (q \wedge (\neg p \vee p))$
- (c)  $\varphi = (q \wedge (\neg p \wedge p))$

19. Bepaal alle modellen van volgende formuleverzamelingen (t.t.z.  $MOD(\Sigma)$ ):

- (a)  $\Sigma = \{((p \leftrightarrow q) \leftrightarrow ((p \rightarrow q) \wedge (q \rightarrow p)))\}$
- (b)  $\Sigma = \{((p \leftrightarrow q) \leftrightarrow ((p \rightarrow q) \wedge (q \rightarrow p))), (p \vee q)\}$

Wat gebeurt er met  $MOD(\Sigma)$  als je  $\Sigma$  groter maakt ?

20. Gebruik modeleliminatie om de volgende problemen op te lossen:

- (a)
  - i. Als Formele Methoden veel blokwerk is, of een slechte assistent heeft, dan is het een moeilijk vak.
  - ii. Als Formele Methoden plezant is, heeft het geen slechte assistent.
  - iii. Als Formele Methoden veel blokwerk is, of als het geen slechte assistent heeft, dan is het een plezant vak.
  - iv. Formele Methoden is plezant enkel en alleen als er veel blokwerk aan is. Bovendien doet de assistent het fantastisch.

Wat zijn de eigenschappen van het vak ?

- (b)
  - i. Als je bist was je gebuisd.
  - ii. Als je gebuisd was, ging je niet veel naar TD.
  - iii. Als je bist of gebuisd was, ging je veel naar TD.
  - iv. Als je niet bist, ging je veel naar TD.

Is het aangeraden om van TD weg te blijven ?

21. Zijn volgende formules tautologieën ? Zijn het contradictions ?

- (a)  $((p \rightarrow q) \wedge (q \rightarrow p)) \leftrightarrow (p \leftrightarrow q)$
- (b)  $((p \wedge q) \vee (r \wedge s)) \wedge \neg r$
- (c)  $((p \wedge \neg p) \wedge r)$

22. Zijn volgende formules equivalent ?

- (a)  $(p \wedge q)$  en  $(p \rightarrow q)$
- (b)  $(p \rightarrow (q \rightarrow r))$  en  $(p \rightarrow q) \rightarrow (p \rightarrow r)$

(c)  $((p \vee \neg p) \rightarrow q) \rightarrow ((p \vee \neg p) \rightarrow r)$ ) en  $(q \rightarrow r)$

23. Het NOR connectief (def. pag. 29) wordt meestal met  $\downarrow$  aangeduid.

(a) Toon aan dat  $(p \downarrow q) \Leftrightarrow \neg(p \vee q)$ .

(b) Schrijf alle standaardconnectieven in functie van het  $\downarrow$  connectief.

Het NAND connectief  $\uparrow$  wordt gedefinieerd door  $(p \uparrow q) \Leftrightarrow \neg(p \wedge q)$ .

(a) Schrijf alle connectieven m.b.v. het  $\uparrow$  connectief.

(b) Toon aan dat je ook voor  $\uparrow$  en  $\downarrow$  wetten van de Morgan kan geven.

24. Geef een syntactische opsomming van de modellen van volgende formules.

(a)  $(p \rightarrow q)$

(b)  $((p \leftrightarrow q) \vee (p \wedge q))$

25. Zet volgende formule om naar disjunctieve normaalvorm.

$(\neg(\neg(p \wedge q) \vee r) \vee (\neg p \wedge q))$

26. (a) Toon aan dat je elke  $k$ -plaatsige functie ( $k \geq 0$ ) op  $\{0, 1\}$  m.b.v. de standaard connectieven kan maken.

(b) Zij  $h : \{0, 1\} \times \{0, 1\} \rightarrow \{0, 1\}$  een binaire functie waarmee je elke  $k$ -plaatsige functie op  $\{0, 1\}$  kan maken. Toon aan dat  $h = h_{\downarrow}$  of  $h = h_{\uparrow}$  waarbij

i.  $h_{\downarrow}(x, y) = 1$  desda  $x + y = 0$

ii.  $h_{\uparrow}(x, y) = 0$  desda  $x + y = 2$

27. Ga voor de volgende paren proposities telkens na of de tweede een geldig gevolg is van de eerste.

(a)  $\{p \wedge q\} \models p \rightarrow q$

(b)  $\{p \rightarrow (q \rightarrow r)\} \models (p \rightarrow q) \rightarrow (p \rightarrow r)$

(c)  $\{\neg p, q \rightarrow p\} \models q$

28. Onderzoek m.b.v. waarheidstabellen of volgende gevolgtrekkingen een tegenoorbeeld hebben

(a)  $\{p\} \models q \rightarrow p$

(b)  $\{p \rightarrow q, q\} \models p$

(c)  $\{p \rightarrow q, q \rightarrow s\} \models p \rightarrow s$

29. Toon volgende uitspraken aan:  $\{\varphi_1, \varphi_2\} \models \psi$  desda  $\{\varphi_1\} \models \varphi_2 \rightarrow \psi$  desda  $\{\varphi_1 \wedge \varphi_2\} \models \psi$

30. Bepaal alle tegenoorbeelden van volgende gevolgtrekkingen m.b.v. semantische tableaus.

(a)  $\{p \rightarrow q, q \rightarrow r\} \models q \rightarrow r$

(b)  $\{p \rightarrow q, q \rightarrow r, r \rightarrow s\} \models p \rightarrow \neg s$

(c)  $\{\neg p \rightarrow \neg q, \neg q \rightarrow \neg r\} \models \{r \rightarrow p, (r \wedge \neg p) \rightarrow (\neg q \wedge q)\}$

$$(d) \{s \vee p, b \wedge p, s \wedge p, \neg b\} \models \{\neg b \leftrightarrow a, c, d\}$$

31. Beschouw volgende nieuwe connectieven.

- (a)  $\uparrow$  (*NAND*)
- (b)  $\downarrow$  (*NOR*)
- (c)  $\underline{\vee}$  (*XOR*)

Bepaal L- en R-regels voor deze connectieven.

32. Als Superman in staat zou zijn en bereid zou zijn kwaad te voorkomen dan zou hij dat doen. Als Superman het kwaad niet zou kunnen voorkomen dan was hij machteloos en als hij het niet zou willen voorkomen dan was hij kwaadwillig. Superman voorkomt het kwaad niet. Als Superman bestaat dan is hij noch machteloos noch kwaadwillig.

Toon *formeel* aan dat Superman niet bestaat. Waarheidstabellen zijn verboden.

33. In ‘De Koopman van Venetië’ van Shakespeare bezit Portia 2 kistjes (een gouden en een zilveren) waarin ze haar portret verstopt heeft. Ieder kistje heeft een inscriptie. Ze belooft haar minnaar dat hij met haar kan trouwen indien hij op basis van de inscripties in staat is om het kistje te kiezen waarin haar portret zit.

Veronderstel dat de kistjes de volgende opschriften bevatten:

- Op het gouden kistje: Het portret zit hier niet in.
- Op het zilveren kistje: Precies één van de inscripties op de kistjes is waar.

Je mag ervan uitgaan dat het portret in precies één kistje zit.

- (a) Geef een gestructureerde (met genummerde stappen) redenering die de minnaar moet helpen het juiste kistje te kiezen.
- (b) Formaliseer het probleem m.b.v. propositielogica en toon d.m.v. een formele techniek aan dat je conclusie uit (a) correct is. Waarheidstabellen zijn verboden.

34. Bepaal m.b.v. semantische tableaus of volgende formuleverzamelingen semantisch consistent zijn.

- (a)  $\{p \rightarrow q, p \rightarrow r, r \rightarrow s, s \rightarrow \neg p\}$
- (b)  $\{p, \neg q, q \rightarrow p\}$
- (c)  $\{p, q, q \rightarrow \neg p\}$

35. Waarom zijn semantisch inconsistent verzamelingen niet zo interessant ?

36. Bepaal mb.v. semantische tableaus of volgende formules tautologieën zijn.

- (a)  $((p \rightarrow q) \wedge (q \rightarrow p)) \leftrightarrow (p \leftrightarrow q)$
- (b)  $((p \wedge q) \vee (r \wedge s)) \wedge \neg r$
- (c)  $(p \wedge \neg p) \wedge r$

Zoek, construeer dan minstens één tegenvoorbeeld.

37. Hoe zou je semantische tableaus gebruiken om aan te tonen of 2 formules  $\varphi$  en  $\psi$  equivalent zijn ?
38. Bewijs met natuurlijke deductie
- $\{p \rightarrow (q \rightarrow r)\} \vdash q \rightarrow (p \rightarrow r)$
  - $\{(p \wedge q) \rightarrow r\} \vdash p \rightarrow (q \rightarrow r)$
  - $\{p \rightarrow q, p \rightarrow r\} \vdash p \rightarrow (q \wedge r)$  en de twee beweringen die uit de omkering voortkomen.
39. Bewijs volgende wetten van de logica m.b.v. het natuurlijk deductiesysteem:
- $\{p \vee (q \wedge r)\} \vdash (p \vee q) \wedge (p \vee r)$  en omgekeerd
  - $\{p \vee (p \wedge q)\} \vdash p$  en omgekeerd
40. Bewijs met natuurlijke deductie:
- $\vdash p \rightarrow \neg\neg p$
  - $\{\neg p \vee q\} \vdash p \rightarrow q$
  - $\{\neg p \rightarrow \neg q\} \vdash q \rightarrow p$
41. Laat zien dat de volgende verzamelingen syntactisch inconsistent zijn:
- $\{\varphi, \psi, \varphi \rightarrow \neg\psi\}$
  - $\{\neg\rho \rightarrow \psi, \rho \rightarrow \sigma, \neg\psi, \neg\sigma\}$
42. Toon aan: een verzameling  $\Gamma$  is syntactisch consistent desda  $\Gamma$  is semantisch consistent.
43. Zijn de volgende verzamelingen consistent?
- $\{p \rightarrow q, p \rightarrow r, r \rightarrow s, s \rightarrow \neg p\}$
  - $\{p, \neg q, q \rightarrow p\}$
  - $\{p, q, q \rightarrow \neg p\}$
44. Pas de bewijstechniek van de formule-inductie toe op de volgende uitspraken.
- Elke formule bevat een even aantal haakjes.
  - Definieer de functie  $K$  van het alfabet naar  $\mathbb{N}$  door  $K(\wedge) = -1$ ;  $K(\vee) = -1$ ;  $K(\rightarrow) = -1$ ;  $K(\leftrightarrow) = -1$ ;  $K(\neg) = 0$ ;  $K(\lhd) = 0$ ;  $K(\lhd) = 0$  en  $K(p) = 1$  voor elke propositioneleletter  $p$ . We breiden  $K$  uit naar woorden door  $K(ab) = K(a) + K(b)$ . Toon nu aan dat voor iedere formule  $\varphi$  geldt:  $K(\varphi) = 1$ . Geldt het omgekeerde ook ?
  - Het aantal propositioneleletters gebruikt in een formule is altijd eindig.
45. Zij  $\Gamma$  een willekeurige collectie formules, toon nu aan:
- als  $\Gamma \vdash \varphi$  dan geldt voor alle  $\psi$   
 $\Gamma \vdash (\neg\varphi \rightarrow \psi)$  en  $\Gamma \vdash (\neg\psi \rightarrow \varphi)$

- (b) als  $\Gamma \vdash \neg(\varphi \rightarrow \neg\psi)$  dan  $\Gamma \vdash \varphi$  en  $\Gamma \vdash \psi$
- (c) omgekeerd: indien zowel  $\Gamma \vdash \varphi$  als  $\Gamma \vdash \psi$  dan  $\Gamma \vdash \neg(\varphi \rightarrow \neg\psi)$

46. Neem een predikaatlogisch alfabet met

$$C = \{nul\}$$

$$F = \{+1, -1, *2\}$$

Maak termen om volgende objecten uit  $\mathbb{N}$  voor te stellen. Neem +1 voor de opvolger, -1 voor de voorganger, *nul* voor 0 en \*2 voor het verdubbelen.

- (a) Het getal 4.
- (b) Het dubbele van 2.
- (c) Het getal 31

47. Beschrijf de verzameling termen  $X$  van volgende talen:

- (a) Een eerste orde taal met een alfabet waar geen functiesymbolen in zitten.
- (b) Een eerste orde taal met een alfabet met slechts één functiesymbool en zonder individuele constanten.

48. Welke van de volgende zijn formules in een taal met als alfabet  $C = \{a_1\}$ ,  $P = \{A_1^1, A_1^2, A_1^3\}$  en  $F = \{f_1^1, f_1^3, f_2^3\}$ ?

- (a)  $A_1^1(A_1^2(f_1^1(x_1), x_1))$
- (b)  $f_1^3(x_1, x_3, x_4)$
- (c)  $(A_1^1(x_2) \rightarrow A_1^3(x_3, a_1))$
- (d)  $\neg\forall x_2 A_1^2(x_1, x_2)$
- (e)  $\forall x_2 (A_1^1(x_2) \rightarrow \neg A_1^1(x_2))$
- (f)  $A_1^3(f_2^3(x_1, x_2, x_3))$

49. (a) Beschouw volgende beweringen over de natuurlijke getallen  $\mathbb{N}$ . De eenplaatsige functieletter  $S$  staat voor de successor- of opvolgerfunctie. De semantiek ervan is  $Sn = n + 1$ . Welke beweringen worden uitgedrukt door de volgende formules?

- i.  $\forall x \neg(Sx = 0)$
- ii.  $\forall x (x = 0 \vee \exists y (x = Sy))$
- iii.  $\forall x \exists y (x < y \wedge \neg \exists z (x < z \wedge z < y))$

(b) Schrijf de volgende beweringen met behulp van formules:

- i. Optellen van getallen is associatief.
- ii. De som van twee getallen is altijd kleiner dan hun produkt.
- iii. Elk even getal groter dan 2 is de som van twee priemgetallen ('Goldbach's Vermoeden').

(c) Definieer met behulp van formules de volgende begrippen:

- i.  $x$  is even
- ii.  $x$  is een deler van  $y$
- iii.  $x$  is een priemgetal.

50. Geef de gebonden en vrije voorkomens van  $x_1$  en  $x_2$  in de volgende formules.

- (a)  $\forall x_2(A_1^2(x_1, x_2) \rightarrow A_1^2(x_2, a_1))$
- (b)  $(A_1^1(x_3) \rightarrow (\neg\forall x_1 \forall x_2 A_1^3(x_1, x_2, a_1)))$
- (c)  $(\forall x_1 A_1^1(x_1) \rightarrow \forall x_2 A_1^2(x_1, x_2))$
- (d)  $\forall x_2(A_1^2(f_1^2(x_1, x_2), x_1) \rightarrow \forall x_1(A_2^2(x_3, f_2^2(x_1, x_2))))$

Is de term  $f_1^2(x_2, x_3)$  vrij voor  $x_1$  in enige van deze?

51. Zij  $t$  de term  $f_1^2(x_1, x_3)$ . Hieronder wordt telkens een formule  $\varphi$  gegeven. Bepaal steeds  $[t/x_1]\varphi$ .

- (a)  $(\forall x_2 A_1^2(x_2, f_1^2(x_1, x_2)) \rightarrow A_1^1(x_1))$
- (b)  $\forall x_1 \forall x_3 (A_1^1(x_3) \rightarrow A_1^1(x_1))$
- (c)  $\forall x_2 A_1^1(f_1^1(x_2)) \rightarrow \forall x_3 A_1^3(x_1, x_2, x_3)$
- (d)  $\forall x_2 A_1^3(x_1, f_1^1(x_1), x_2) \rightarrow \forall x_3 A_1^1(f_1^2(x_1, x_3))$

52. Bereken  $VV$  van de formules in oefening ???. Zijn het gesloten of open formules?

53. We bekijken het universum  $E_1$  bestaande uit de natuurlijke getallen en het universum  $E_2$  van alle kantoorgebouwen. We beschikken over de volgende predikaten:

$I(x, y)$	“x is gelijk aan y”
$K(x)$	“x is een kwadraat”
$G(x)$	“x is een gebouw”
$L(x, y)$	“x is groter dan y”

Druk nu onderstaande zinnen uit met behulp van kwantoren en deze predikaten en bepaal voor elk van de universa  $E_1$ ,  $E_2$ ,  $E_1 \cup E_2$  of ze waar zijn.

- (a) Er zijn twee verschillende objekten.
  - (b) Alle objekten zijn verschillend van kwadranten.
  - (c) Er zijn geen gebouwen.
  - (d) Er is geen grootste objekt.
54. Geef een minimaal model  $(\mathcal{ID}, I)$  om volgende formules te kunnen interpreteren.

- (a)  $\forall x_2(A_1^2(x_1, x_2) \rightarrow A_1^2(x_2, a_1))$
- (b)  $(\forall x_2(A_1^2(f_1^2(x_1, x_2), x_1) \rightarrow \forall x_1(A_2^2(x_3, f_2^2(x_1, x_2))))$

Volgend model zullen we een aantal keer tegenkomen. Als de taal  $L$  een constante, een predicaat en drie functiesymbolen bevat, worden die in het model  $\mathcal{IN} = ((\mathcal{IN}, \{\lambda x.x + 1, +, .\}, \{=\}), I)$  geïnterpreteerd als:

$$\begin{aligned} I(a_1) &= 0 \\ I(f_1^1) &= \lambda x.x + 1 \\ I(f_1^2) &= + \\ I(f_2^2) &= . \\ I(A_1^2) &= == \end{aligned}$$

55. Beschouw het model  $\mathbb{N}$  met de bedeling  $b$  zodanig dat  $b(x) = 4$ ,  $b(y) = 5$  en  $b(z) = 16$ . Waardeer nu telkens volgende termen en formules.
- $f_1^2(z, f_2^2(x, f_1^2(y, f_1^1(y))))$
  - $(A_1^1(x) \rightarrow \neg \forall x \forall y A_1^2(x, y))$
  - $\forall x (A_1^2(f_1^2(z, x), z) \rightarrow \forall x (A_1^2(y, f_2^2(x, z))))$
56. Vind in het model  $\mathbb{N}$ , zo mogelijk, bedelingen  $b$  en  $b'$  zodat voor volgende formules  $\varphi$  geldt  $V_{\mathbb{N}, b}(\varphi) = 1$  (t.t.z.  $\mathbb{N}, b \models \varphi$ ) en  $V_{\mathbb{N}, b'}(\varphi) = 0$  (t.t.z.  $\mathbb{N}, b' \not\models \varphi$ ).
- $A_1^2(f_1^2(x_1, x_1), f_2^2(x_2, x_3))$
  - $(A_1^2(f_1^2(x_1, a_1), x_2) \rightarrow A_1^2(f_1^2(x_1, x_2), x_3))$
  - $\forall x_1 A_1^2(f_2^2(x_1, x_2), x_3)$
  - $(\forall x_1 A_1^2(f_2^2(x_1, a_1), x_1) \rightarrow A_1^2(x_1, x_2))$
  - $\forall x_1 (A_1^2(x_1, a_1) \wedge A_1^2(x_1, x_2))$
57. Is  $\mathbb{M} = ((\{x|x \text{ is een prof}\}, \Phi, \{\text{geeft - meer - uren - als, geeft - het - moeilijkste - vak}\}), I)$  met  $I(a_1) = \text{Theo}$ ,  $I(A_1^2) = \text{geeft - meer - uren - als}$ , en  $I(A_1^1) = \text{geeft - het - moeilijkste - vak}$  een model voor
- $\forall x_1 (A_1^2(x_1, x_2) \wedge \neg A_1^2(x_1, x_2))$
  - $A_1^1(a_1) \vee \neg A_1^1(a_1)$
  - $\forall x_1 (A_1^2(x_1, x_2) \rightarrow A_1^1(x_1))$
58. Bereken  $V_{\mathbb{M}, b}([t/x]t')$  op 2 manieren als  
 $t = f_1^2(x, f_2^2(a_1, x))$   
 $t' = f_2^2(f_1^2(x, a_1), f_2^2(x, y))$   
en
- $$\mathbb{M} = ((\mathbb{N}, \{\lambda(x, y).x + y2, .\}, \Phi), I)$$
- met
- $$I(a1) = 3$$
- $$I(f_1^2) = .$$
- $$I(f_2^2) = \lambda(x, y).x + y^2$$
- en
- $$b(x) = 2$$
- $$b(y) = 1.$$
59. Zijn volgende formules universeel geldig?
- $\forall x R(x) \rightarrow \forall y \neg R(y)$
  - $\forall x \forall y (S(x, y) \rightarrow S(y, x))$
  - $\forall x (R(x) \vee \neg R(x))$
  - $(\forall x (R(x) \wedge A(x)) \rightarrow \neg \exists x \neg A(x))$

60. Zet volgende formule om in prenexvorm:

$$\forall x \exists y (R(x, y) \rightarrow \exists w R(w, y)) \rightarrow \exists y \forall x (S(x, y) \rightarrow \exists w S(y, w))$$

61. Bewijs met semantische tableaus.

- (a)  $\{\neg\exists x(A(x) \wedge B(x)), \exists x(B(x) \wedge C(x))\} \models \neg\forall(C(x) \rightarrow A(x))$
- (b)  $\{\forall x(A(x) \rightarrow B(x)) \vee \forall y(B(y) \rightarrow A(y))\} \models \forall x \forall y((A(x) \wedge B(y)) \rightarrow (B(x) \vee A(y)))$
- (c)  $\{\forall x \forall y((A(x) \wedge B(y)) \rightarrow (B(x) \vee A(y)))\} \models \forall x(A(x) \rightarrow B(x)) \vee \forall y(B(y) \rightarrow A(y))$
- (d)  $\models \neg\exists x \forall y(R(x, y) \leftrightarrow \neg R(y, x))$
- (e)  $\models \neg\exists x \forall y(R(x, y) \leftrightarrow \neg\exists z(R(y, z) \wedge R(z, y)))$

62. Geef met behulp van een semantisch tableau een tegenvoorbeeld voor volgende gevolgtrekking:

$$\{\forall x \forall y((A(x) \wedge B(x)) \rightarrow R(x, y)), \forall x \neg R(x, x), \forall x(A(x) \vee B(x))\} \models \forall x \forall y(R(x, y) \rightarrow A(x))$$

63. Test de volgende syllogismen op geldigheid en geef in geval van niet-geldigheid een tegenvoorbeeld:

- (a)  $\{\forall x(A(x) \rightarrow B(x)), \exists x(A(x) \wedge C(x))\} \models \exists x(C(x) \wedge B(x))$
- (b)  $\{\forall x(A(x) \rightarrow B(x)), \exists x(A(x) \wedge \neg C(x))\} \models \exists x(C(x) \wedge \neg B(x))$
- (c)  $\{\neg\exists x(A(x) \wedge B(x)), \forall x(B(x) \rightarrow C(x))\} \models \neg\exists x(C(x) \wedge A(x))$

64. Bewijs m.b.v. natuurlijke deductie:

- (a)  $\{\forall x(\varphi \wedge \psi)\} \vdash \forall x \varphi \wedge \forall x \psi$  en omgekeerd
- (b)  $\{\exists x(\varphi \vee \psi)\} \vdash \exists x \varphi \vee \exists x \psi$  en omgekeerd

65. Bewijs de volgende, bij prenexvormconstructies gebruikte, gevolgtrekkingen ( $x$  niet vrij in  $\psi$ ).

- (a)  $\{\exists x \varphi \rightarrow \psi\} \vdash \forall x(\varphi \rightarrow \psi)$  en omgekeerd
- (b)  $\{\forall x \varphi \rightarrow \psi\} \vdash \exists x(\varphi \rightarrow \psi)$  en omgekeerd

66. Bewijs het volgende syllogisme:

$$\{\exists x(A(x) \wedge B(x)), \neg\exists x(B(x) \wedge C(x))\} \vdash \neg\forall x(A(x) \rightarrow C(x))$$

67. Schrijf volgende functies in  $\lambda$ -notatie:

- (a)  $f(x) = 7x$
- (b)  $g(y) = y^2$
- (c)  $tripsam(f) = f \circ f \circ f$
- (d)

$$l(x) = \begin{cases} 1 & \text{als } x = 0 \\ x.l(x-1) & \text{als } x > 0 \end{cases}$$

68. Beschouw volgende ‘ $\mathcal{P}$ -calculus’: Syntaxregels:

- (a)  $|$  is een woord uit  $\mathcal{P}$ .
- (b) als  $w$  een woord is uit  $\mathcal{P}$ , is  $w|$  een woord uit  $\mathcal{P}$ .
- (c) als  $w$  en  $w'$  woorden zijn uit  $\mathcal{P}$  is  $(w\$)w'$  een woord uit  $\mathcal{P}$ .

Reductieregels:

- (a)  $(w\$)| =_{\mathcal{P}} w|$
- (b)  $(w\$)|w =_{\mathcal{P}} (w|\$)w$

Opgaven:

- (a) Reken de expressie  $((|||||\$)||\$)||||$  uit volgens  $=_{\mathcal{P}}$ .
- (b) Wat is de informele betekenis van  $\mathcal{P}$  ?
- (c) Geef een formele semantiek aan  $\mathcal{P}$ .
- (d) Bereken de semantiek van opgave en oplossing van (a).

69. Zijn volgende woorden elementen van  $\Lambda$  (t.t.z. geldige  $\Lambda$ -expressies) ?

- (a)  $(\lambda x.(\lambda y.((z)x)t)(\lambda x.y)z)(\lambda x.x)$
- (b)  $(\lambda x.x)(\lambda y.y)(\lambda z.z)$
- (c)  $(\lambda x.x)(\lambda y.y)\lambda z.z$
- (d)  $y$
- (e)  $\lambda x.t$

70. Voor iedere  $\lambda$ -expressie kan men als volgt ‘constructiebomen’ tekenen.

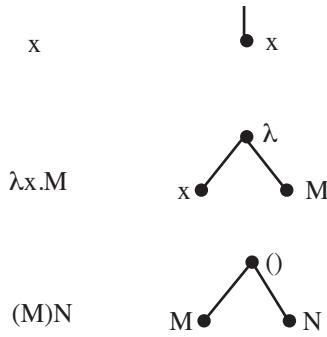


Figure 2: Constructiebomen voor  $\lambda$ -expressies

Teken de constructiebomen voor elk van de volgende  $\lambda$ -expressies.

- (a)  $(\lambda x.(\lambda y.((z)x)t)(\lambda x.y)z)(\lambda x.x)t$
- (b)  $(\lambda x.x)(\lambda y.y)(\lambda z.z)x$
- (c)  $(\lambda x.x)(\lambda y.y)\lambda z.z$
- (d)  $y$
- (e)  $\lambda x.t$

- (f)  $(y)(y)(y)x$   
 (g)  $((y)y)y)x$
71. Geef alle  $\lambda$ -expressies die je door invoeging van haakjes kan maken m.b.v.  $\lambda x.\lambda x.xxx$ . Teken ook hun constructiebomen.
72. Schrijf volgende functies in  $\lambda$  – *notatie ‘gecurried’*.
- $\lambda(x,y,z).(x^2 + y^2 = z^2)$
  - $\lambda(x,(y,z)).x - \parallel(y,z)\parallel$
73. Schrijf volgende functies in  $\lambda$  – *notatie ‘uncurried’*.
- $\lambda x\lambda y\lambda z.x + y - z$
  - $\lambda f\lambda xf(x)$
74. Welke (Scheme-special-form) zorgt er voor dat je in  $\lambda$ -calculus comfortabel kan programmeren ?
75. Geef in volgende  $\lambda$ -expressies voor ieder voorkomen van iedere variabele aan of ze vrij of gebonden voorkomt. Geef ook formeel van elke  $\lambda$ -expressie de verzameling  $VV$  aan. Zijn het combinatoren ?
- $\lambda f.\lambda x.((f)(x)x)(f)(x)x$
  - $\lambda f.(x)y$
  - $\lambda g.(\lambda y.(x)y)\lambda z.z$
76. Bepaal respectievelijk de substitutie van volgende  $\lambda$ -expressies in de  $\lambda$ -expressies van oefening ??.
- $\lambda x.x$  in  $x$
  - $\lambda x.(f)x$  in  $y$
  - $(y)(g)z$  in  $x$
77. Zijn volgende  $\lambda$ -expressies  $\alpha$ -gelijk ?
- $\lambda x.x$  en  $\lambda z.z$
  - $\lambda x.((f)x)x$  en  $\lambda y.((g)y)y$
  - $\lambda x.(x)x$  en  $\lambda z.(z)x$
  - $(\lambda x.x)x$  en  $(\lambda z.z)x$
  - $(\lambda x.x)y$  en  $y$
78. Zijn volgende  $\lambda$ -expressies  $\beta$ -gelijk ?
- $((\lambda n.\lambda m.\lambda f.\lambda x.((n)f)((m)f)x)\lambda f.\lambda x.(f)(f)(f)x)\lambda f.\lambda x.(f)x =_{\beta} \lambda g.\lambda y.(g)(g)(g)y$
  - $(\lambda n.\lambda f.\lambda x.(f)((n)f)x)\lambda f.\lambda x.(f)(f)x =_{\beta} \lambda g.\lambda y.(g)(g)y$
79. Reken uit:
- $(succ)c_4$
  - $((plus)c_2)c_2$

- (c)  $((cons)c_1)c_2)car$   
 (d)  $((times)c_3)c_2$
80. (a) Maak de booleaanse waarden samen met hun operaties *not*, *and* en *or*.  
 (b) Geef een  $\lambda$ -expressie voor de machtsverheffing voor  $n, m \geq 1$ .
81. (a) Maak het  $=$  en het  $<$  predicaat op Church-numerals. De andere predicaten volgen hier automatisch uit.  
 (b) Maak een  $\lambda$ -expressie *repeat* die een Church-numeral  $c_n$  en willekeurige  $\lambda$ -expressies  $F$  en  $A$  als invoer neemt, en die  $F$   $n$  keer op  $A$  toepast.
82. Modeer gehele getallen in  $\lambda$ -calculus.
83. Schrijf  $\lambda$ -termen voor *ggd* en *fib* op numerals.
84. (a) Maak een  $\lambda$ -expressie  $F$  zodanig dat voor elke  $\lambda$ -expressie  $M$  geldt:  

$$(F)M =_{\beta} (M)M.$$
- (b) Maak een  $\lambda$ -expressie  $F$  zodanig dat voor elke  $\lambda$ -expressie  $M$  geldt:  

$$(F)M =_{\beta} (M)F.$$
- (c) Maak  $F$  zodat voor elke  $M$ :  $(F)M =_{\beta} F$
- (d) Vind  $F$  zodat voor elke  $M$  en  $N$ :  $((F)M)N =_{\beta} (M(N)M)N$

# Grondslagen van de Informatica I

## Formularium

### Semantische Tableaus

$\neg L$	$\neg R$
$\neg\alpha$ 	$\alpha$ 
$\wedge L$	$\wedge R$
$\alpha \wedge \beta$ 	$\alpha \wedge \beta$ 
$\vee L$	$\vee R$
$\alpha \vee \beta$ 	$\alpha \vee \beta$ 
$\rightarrow L$	$\rightarrow R$
$\alpha \rightarrow \beta$ 	$\alpha \rightarrow \beta$ 
$\Leftrightarrow L$	$\Leftrightarrow R$
$\alpha \Leftrightarrow \beta$ 	$\alpha \Leftrightarrow \beta$ 
$\forall L$	$\forall R$
$\forall_x \varphi(x)$ voor alle d $\in D$ 	$\forall_x \varphi(x)$ $D' = D \cup \{d\}$ met d $\notin D$ $\varphi(d)$ 
$\exists L$	$\exists R$
$\exists_x \varphi(x)$ $D' = D \cup \{d\}$ met d $\notin D$ $\varphi(d)$ 	$\exists_x \varphi(x)$ $[d/x]\varphi(x)$ voor alle d $\in D$ 

### Afleidingen

$\wedge$	
$\frac{p \wedge q}{p} \wedge E$	$\frac{p \wedge q}{q} \wedge E$
$\frac{p \quad q}{p \wedge q} \wedge I$	
$\vee$	
$\frac{p}{p \vee q} \vee I$	$\frac{q}{p \vee q} \vee I$
$\frac{\begin{array}{c} (1) \\ p \\ \vdots \\ r \end{array}}{p \vee q} \quad \frac{\begin{array}{c} (2) \\ q \\ \vdots \\ r \end{array}}{p \vee q}$	$\frac{p \vee q}{r} \vee E[-1, -2]$
$\rightarrow$	
$\frac{\begin{array}{c} (1) \\ p \\ \vdots \\ q \end{array}}{p \rightarrow q} \rightarrow I[-1]$	$\frac{p \quad p \rightarrow q}{q} \rightarrow E$
$\neg$	
$\frac{p \quad \neg p}{q} \neg E$	
$\frac{\begin{array}{c} (1) \\ q \\ \vdots \\ p \end{array}}{\neg q} \neg I[-1]$	$\frac{\begin{array}{c} (1) \\ \neg q \\ \vdots \\ p \end{array}}{q} \neg E * [-1]$
$\forall$	
$\frac{\varphi(d)}{\forall_x \varphi(x)} \forall I$	$\frac{\forall_x \varphi(x)}{\varphi(d)} \forall E$
$\exists$	
$\frac{\begin{array}{c} (1) \\ \varphi(d) \\ \vdots \\ \varphi(d) \end{array}}{\exists_x \varphi(x)} \exists I$	$\frac{\exists_x \varphi(x)}{r} \exists E[-1]$