Propositielogica: syntaxis en semantiek

Inleiding

#aannames gevold door een conclusie
- Notatie: a_1, a_2, ... a_n / c met a = aannamen en c = conclusie

Vertalen van zinnen in natuurlijke taal in formele taal

- Het *alfabet* geeft aan welke symbolen gebruikt mogen worden om uitdrukkingen (zinnen bijvoorbeeld) in de taal te maken.
  3 soorten symbolen: propositieletters, logische symbolen & hulpsymbolen
- De *syntaxis* is de grammatica van een taal, dat wil zeggen, een geheel van regels dat aangeeft op welke manier uitdrukkingen van de taal gevormd mogen worden.
- De *semantiek* houdt zich bezig met de *betekenis* van uitdrukkingen:

Syntaxis van de propositielogica

Een propositie is een bewering of uitspraak, uitgedrukt in een zin.

In de formele taal gebruiken we *propositieletters* om proposities weer te geven die we niet nader wensen te analyseren. Notatie: kleine letters.

5 standaard logische symbolen

¬	niet
∧	en
∨	of
→	als ..., dan ...
↔	dan en slechts dan als

Logische voegwoorden/ connectieven:
- proposities verbinden en op die manier samengestelde uitdrukkingen vormen

Hulpsymbolen: '(' & ')' haakjes

Alfabet
Een alfabet van de propositielogica bestaat uit:
a een verzameling propositieletters;
b logische symbolen: ¬, ∧, ∨, →, ↔;
c hulpsymbolen:) en (.

Formule: Samengestelde uitdrukkingen in propositielogica met behulp van symbolen uit het alfabet en vaste regels
Notatie: kleine Griekse letters

Formule
De formules van de propositielogica worden als volgt gedefinieerd:
1 elke propositieletter is een formule;
2 als φ en ψ formules zijn, dan zijn $\neg\varphi$, $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $(\varphi \to \psi)$ en $(\varphi \leftrightarrow \psi)$ ook formules;
3 niets anders is een formule.

2 formules zijn logisch equivalent als formule_1 <=> formule_2

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:
$((\varphi \wedge (\psi \vee \chi)) \leftrightarrow ((\varphi \wedge \psi) \vee (\varphi \wedge \chi)))$
$((\varphi \vee (\psi \wedge \chi)) \leftrightarrow ((\varphi \vee \psi) \wedge (\varphi \vee \chi)))$

Een *inductieve definitie* heeft drie kenmerkende onderdelen:
1 een *basisstap* waarin bepaalde dingen meteen tot objecten van de gewenste soort verklaard worden;
2 één of meer *opbouwstappen* die verdere constructieprincipes geven om objecten te maken;
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.

Atomaire functies/ atomen/ propositieletters -> niet te ontleden

vorm	*uitspraak*	*vormnaam*
$\neg\varphi$	niet φ	negatie
$(\varphi \wedge \psi)$	φ en ψ	conjunctie
$(\varphi \vee \psi)$	φ of ψ	disjunctie
$(\varphi \to \psi)$	als φ, dan ψ	implicatie
$(\varphi \leftrightarrow \psi)$	φ dan en slechts dan als ψ	equivalentie

Formuleschema is een abstracte vorm van een formule
(Griekse letters dat formules voorstellen in een formule verwerkt)

Laat φ een formule zijn waarin (mogelijk) een propositieletter p voorkomt. Door elk voorkomen van p in φ te vervangen door een formule ψ, ontstaat een nieuwe formule. Het resultaat van dit *substitueren van ψ voor p in φ* wordt genoteerd als: $[\psi/p]\varphi$

Inductieve definitie van formules om substitutie in te voeren
- a) het begrip uitleggen voor de basisobjecten
- b) het begrip uitleggen voor complexe objecten

Substitutie
a $[\psi/p]\varphi = \psi$ als $\varphi = p$
$[\psi/p]\varphi = \varphi$ als φ een propositieletter is en $\varphi \neq p$
b $[\psi/p]\neg\varphi = \neg[\psi/p]\varphi$
$[\psi/p](\varphi \vee \chi) = ([\psi/p]\varphi \vee [\psi/p]\chi)$
$[\psi/p](\varphi \wedge \chi) = ([\psi/p]\varphi \wedge [\psi/p]\chi)$
$[\psi/p](\varphi \to \chi) = ([\psi/p]\varphi \to [\psi/p]\chi)$
$[\psi/p](\varphi \leftrightarrow \chi) = ([\psi/p]\varphi \leftrightarrow [\psi/p]\chi)$

Het opbouwen van een formule uit een aantal atomen met behulp van connectieven, kan worden weergegeven in een *constructieboom*. Zo'n constructieboom wordt van onder naar boven 'gelezen': we beginnen met de atomaire formules en bouwen daaruit de formule op.

Elke formule heeft exact 1 constructieboom.
Opgelet: in natuurlijke taal kan soms op meerde manieren grammaticaal gelezen worden
-> daarom vertalen we naar de propositielogica

De *haakjes* in een formule geven het *bereik* van een connectief aan. Het bereik van een voorkomen van een connectief is (zijn) de formule(s) die in de boom met dat connectief onmiddellijk verbonden wordt (worden).

Semantiek van de propositielogica

Betekenis van een zin/ formule wordt gegeven door te zeggen onder welke omstandigheden een zin waar is of niet waar is.

Waarheidswaarden: "waar = 1" en "niet waar = 0"

In een waarheidstabel staan alle combinaties van de waarheidswaarden van de propositieletters, met daarachter voor elk van deze gevallen de waarheidswaarde

φ	$\neg\varphi$
1	0
0	1

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

φ	ψ	$(\varphi \to \psi)$
1	1	1
1	0	0
0	1	1
0	0	1

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

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

Modellen

Waardering
Een waardering is een functie van alle propositieletters naar waarheidswaarden.

vb V(p) = 1 en V(a) = 0

Model MOD (Φ) = {V | V(Φ) = 1}
Een waardering V heet een *model* van een formule φ als geldt: $V(\varphi) = 1$.

Model van een formuleverzameling
Een waardering V heet een model van een formuleverzameling Σ als zij model is van elke $\varphi \in \Sigma$. MOD(Σ)

Modeleliminatie: $\Sigma_1 \subseteq \Sigma_2 \Rightarrow MOD(\Sigma_2) \subseteq MOD(\Sigma_1)$

Tautologie
Een formule φ heet een *tautologie* als elke waardering een model is van φ, ofwel als voor elke waardering V geldt: $V(\varphi) = 1$.

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

'en' en 'of' zijn associatief => haakjes mogen weggelaten worden

Opsomming van modellen
- connectieven: en, om & niet
- opsomming is een disjunctie van conjuncties van atomen

Functioneel volledig
Laat C een verzameling connectieven zijn (bijvoorbeeld $\{\neg, \vee\}$ of $\{\vee, \wedge, \leftrightarrow\}$). C heet *functioneel volledig* als elke formule φ logisch equivalent is met een formule ψ die slechts connectieven uit C bevat.

Waarheidsfunctie
Een functie van $\{0, 1\}^k$ naar $\{0, 1\}$ heet een *k*–plaatsige waarheidsfunctie.

Disjunctieve normaalvorm is een disjunctie van conjuncties, bestaande uit atomen of negaties van atomen $(\varphi_1 \wedge \ldots \wedge \varphi_{n_1}) \vee \ldots \vee (\chi_1 \wedge \ldots \wedge \chi_{n_k})$ waarbij $\varphi_1, \ldots, \chi_{n_k}$ atomen of negaties van atomen zijn.

Als φ een formule is, dan bestaat er een formule φ^* in disjunctieve normaalvorm, zodat φ en φ^* logisch equivalent zijn.

Propositielogica: Geldig gevolg

Geldig gevolg

Geldig gevolg
Een formule ψ heet een *geldig gevolg* van een verzameling formules Σ als elk model van Σ ook model is van ψ.

Tegenvoorbeeld van de gevolgtrekking Σ/ψ er bestaat een model van Σ dat geen model is van ψ

Semantische tableaus

Een sequent is een rijtje van de volgende vorm, waarbij $\varphi_1, \ldots, \varphi_n$ en $\psi_1, \ldots, \psi_m$ ≥ rijtjes formules zijn, die worden gescheiden door een nieuw symbool ∘: $\varphi_1, \ldots, \varphi_n \circ \psi_1, \ldots, \psi_m$

Een waardering V heet een *tegenvoorbeeld* van een sequent $\varphi_1, \ldots, \varphi_n \circ \psi_1, \ldots, \psi_m$ indien $V(\varphi_1) = \ldots = V(\varphi_n) = 1$ en $V(\psi_1) = \ldots = V(\psi_m) = 0$.

Als een formule aan beide kanten van ∘ voorkomt, kan deze formule geen tegenvoorbeeld hebben

Een *semantisch tableau* is een schema waarin op systematische wijze het mogelijk bestaan van tegenvoorbeelden van een gegeven sequent teruggebracht (*gereduceerd*) wordt tot dat van één of meer overzichtelijker sequenten. Uiteindelijk voert dit proces ons tot de eenvoudigste

Een sequent wordt gereduceerd met behulp van reductie-regels
Reductieregels mogen gebruikt worden op het examen!

Elke tak in een tableau correspondeert met een mogelijke route om een tegenvoorbeeld te vinden. Zo'n route kan op twee manieren eindigen:
a Er treedt eenzelfde formule links en rechts op in de sequent: we zeggen dan dat de tak van de boom *sluit*.
b Er treedt geen formule zowel links als rechts op, terwijl ook geen regel meer toepasbaar is. Dan noemen we de tak *open*.

We onderscheiden dan ook twee soorten tableaus:
1 Alle takken sluiten. Het tableau heet dan *gesloten*.
2 Minstens één tak blijft open. Het tableau heet dan *open*.

Consistentie

Semantisch consistent
Een formuleverzameling Σ is (*semantisch*) *consistent* als Σ een model heeft. We zeggen ook dat Σ *vervulbaar* is.

<> Semantisch inconsistent

Een verzameling is semantisch consistent als het tableau open is

$$\frac{}{\circ\, \varphi}$$

Als deze sequent geen tegenvoorbeeld heeft, dan is φ waar in elk model van de lege verzameling. Aangezien elke waardering model is van de lege verzameling, is elke waardering model van φ. Dus φ is een tautologie als een tableau van $\circ\, \varphi$ sluit.

Adequaatheid

Adequaatheidsstelling
$\varphi_1, \ldots, \varphi_n \models \psi \Leftrightarrow$ er bestaat een gesloten tableau voor $\varphi_1, \ldots, \varphi_n \circ \psi$.

De adequaatstelling zegt dat alle mogelijke tableaus sluiten als ook maar 1 van hen sluit.

Propositielogica: afleidingen

Structuur van redeneren

Afleidingen kunnen lineair en in boomvorm

$$\frac{\varphi_1 \quad \cdots \quad \varphi_n}{\psi}$$

Formules/ aannamens
Conclusie

Natuurlijke deductie

Formules in formularium!

Afleidbaar en consistent

Afleidbaar
Een formule φ heet *afleidbaar* uit een verzameling aannames Σ als er een afleiding van φ bestaat waarin aan het eind alleen nog aannames uit Σ van kracht zijn. Notatie: $\Sigma \vdash \varphi$.

Als een formule afleidbaar is zonder aannames, dan is deze formule een stelling.

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

Een formuleverzameling Γ is consistent desda er bestaat een formule φ zodat $\Gamma \nvdash \varphi$.

⇒ Als Γ consistent is, dan is er geen formule φ waarvoor $\Gamma \vdash \varphi$ en $\Gamma \vdash \neg\varphi$. Voor een willekeurige formule ξ geldt dan $\Gamma \nvdash \xi$ of $\Gamma \nvdash \neg\xi$. Als $\Gamma \nvdash \xi$, dan is ξ de gezochte formule, als $\Gamma \nvdash \neg\xi$, dan $\neg\xi$.
⇐ (Contrapositie) Als Γ inconsistent is, dan bestaat er een formule ξ zodat $\Gamma \vdash \xi$ en $\Gamma \vdash \neg\xi$. Maar dan is met $\neg$E elke formule afleidbaar uit Γ, ofwel er bestaat geen formule φ met $\Gamma \nvdash \varphi$. □

$\Gamma \nvdash \varphi \Leftrightarrow \Gamma \cup \{\neg\varphi\}$ is consistent.

We passen op zowel ⇒ als ⇐ contrapositie toe:
⇒ Stel $\Gamma \cup \{\neg\varphi\}$ is inconsistent. Dan is er een formule ξ zodat $\Gamma \cup \{\neg\varphi\} \vdash \xi$ en $\Gamma \cup \{\neg\varphi\} \vdash \neg\xi$. De eliminatieregel $\neg$E* geeft nu: $\Gamma \vdash \varphi$.
⇐ Als $\Gamma \vdash \varphi$, dan ook $\Gamma \cup \{\neg\varphi\} \vdash \varphi$. Omdat ook $\Gamma \cup \{\neg\varphi\} \vdash \neg\varphi$, is $\Gamma \cup \{\neg\varphi\}$ inconsistent. □

Volledigheid
Als Σ een formuleverzameling is en φ een formule, dan geldt: $\Sigma \vdash \varphi \Leftrightarrow \Sigma \models \varphi$.

correctheid: afleidbare gevolgtrekkingen zijn semantisch geldig
volledigheid: semantisch geldige gevolgtrekkingen zijn afleidbaar.

Propositielogica: metatheorie

Formule-inductie

Basisstap Stel B een bewering

Laat zien dat B opgaat voor de atomen.

Inductiestap

Laat zien dat, als B opgaat voor formules φ, ψ, dan ook voor hun combinaties $\neg\varphi, \varphi \wedge \psi, \varphi \vee \psi, \varphi \rightarrow \psi$ en $\varphi \leftrightarrow \psi$.

Een formule heet een *subformule* van een formule φ als die formule als een aaneengesloten deel in φ voorkomt.

Het aantal subformules in $\mathcal{G}$ als 'subf($\mathcal{G}$)

De *lengte* van een formule φ (*lengte*(φ)) is het aantal symbolen in φ, zonder de haakjes mee te tellen.

Voor elke formule φ is het aantal voorkomens van subformules in φ gelijk aan de lengte van φ.

Bewijs **Basisstap**

Voor een atomaire formule is het aantal subformules gelijk aan de lengte, want een atomaire formule heeft maar één voorkomen van een subformule, namelijk zichzelf, en ook de lengte van een atomaire formule is 1. **Inductiestap**

Stel dat de bewering reeds geldt voor twee willekeurige formules φ en ψ: dat wil zeggen, $subf(\varphi) = lengte(\varphi)$ en $subf(\psi) = lengte(\psi)$. We moeten bewijzen dat de bewering dan ook opgaat voor $\neg\varphi, \varphi \wedge \psi, \varphi \vee \psi, \varphi \rightarrow \psi$ en $\varphi \leftrightarrow \psi$.

Geval 1: negatie. We hebben dan de volgende berekening: $subf(\neg\varphi) = subf(\varphi) + 1 = lengte(\varphi) + 1 = lengte(\neg\varphi)$.

De eerste gelijkheid is juist omdat $\neg\varphi$ één subformule meer heeft dan φ, namelijk $\neg\varphi$. Bij de tweede gelijkheid passen we de inductiehypothese $subf(\varphi) = lengte(\varphi)$ toe. De laatste gelijkheid geldt omdat in $\neg\varphi$ één symbool meer voorkomt dan in φ, namelijk $\neg$.

Geval 2: binaire connectieven. Het voorbeeld van conjunctie illustreert het algemene geval: $subf(\varphi \wedge \psi) = subf(\varphi) + subf(\psi) + 1$ (de conjunctie heeft als subformules die van haar componenten plus zichzelf) = $lengte(\varphi) + lengte(\psi) + 1$ (wegens de inductiehypothese) = $lengte(\varphi \wedge \psi)$ ($\wedge$ telt nu ook mee). □

Functionele volledigheid

Het paar connectieven $\{\neg, \wedge\}$ is functioneel volledig.

Bewijs

Op een informele manier is dit reeds in hoofdstuk 2 'bewezen'. We herformuleren de bewering van de stelling als volgt:

Voor elke formule φ is er een logisch equivalente formule φ' die alleen met $\neg$ en $\wedge$ geconstrueerd is.

Basisstap

Voor atomaire formules geldt dit triviaal, want atomaire formules bevatten geen connectieven. **Inductiestap**

De inductiehypothese zegt dat we twee formules φ en ψ herschreven hebben tot logisch equivalente formules φ' respectievelijk ψ' die alleen met $\neg$ en $\wedge$ geconstrueerd zijn. We moeten laten zien dat dan ook $\neg\varphi$, $\varphi \wedge \psi, \varphi \vee \psi, \varphi \rightarrow \psi$ en $\varphi \leftrightarrow \psi$ aldus te herschrijven zijn. Voor negatie en conjunctie is dit direct duidelijk: $(\neg\varphi)' = \neg\varphi'$ en $(\varphi \wedge \psi)' = \varphi' \wedge \psi'$. Voor de andere connectieven gebruiken we tevens:

1 $(\varphi \vee \psi)' = \neg(\neg\varphi' \wedge \neg\psi')$
2 $(\varphi \rightarrow \psi)' = \neg(\varphi' \wedge \neg\psi')$
3 $(\varphi \leftrightarrow \psi)' = (\varphi \rightarrow \psi)' \wedge (\psi \rightarrow \varphi)'$ □

Adequaat van semantische tableaus

Gesloten tableau: voor rijtjes formules Φ en Ψ betekent $\Phi \models \Psi$ dat elke waardering die alle formules in Φ de waarde 1 geeft, minstens één formule in Ψ de waarde 1 geeft. Als het rijtje Ψ uit één formule ψ bestaat, reduceert dit tot semantische geldigheid in de gebruikelijke zin.

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

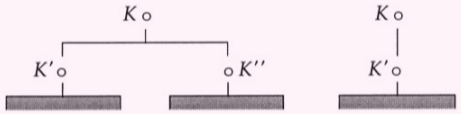
Bewijs

Hier is een gewone wiskundige inductie het meest aangewezen, en wel naar het aantal knopen in gesloten tableaus (dit zijn er altijd slechts eindig veel). **Basisstap**

Het tableau heeft één knoop. Sluiting in de bovenste knoop betekent dat een zekere formule zowel links in Φ als rechts in Ψ voorkwam. Dan is $\Phi \models \Psi$ een geldige gevolgtrekking. **Inductiestap**

Het tableau heeft meer knopen. Dan is het ontstaan door een van de regels voor connectieven toe te passen. Dit kan via een splitsende regel (type II) of een niet–splitsende regel (type I) zijn gebeurd:

II I



De gearceerde gebieden staan voor subbomen. Dat zijn zelf ook gesloten tableaus, maar met minder knopen, zodat de inductiehypothese op hun topknoop van toepassing is. Aan de hand van twee gevallen zullen we illustreren hoe nu de gewenste bewering wordt bereikt.

Type I: regel $\neg_L$. De topsequent heeft dan de vorm: $\Phi, \neg\alpha \circ \Psi$; en de sequent onmiddellijk daaronder: $\Phi \circ \Psi, \alpha$. Volgens de inductiehypothese geldt: $\Phi \models \Psi, \alpha$. Zij nu V een willekeurige waardering die alle formules in Φ en $\neg\alpha$ waar maakt. Omdat V Φ waar maakt, volgt uit $\Phi \models \Psi, \alpha$ dat een formule uit Ψ of α waar wordt gemaakt. Het laatste strijdt met de aanname dat V $\neg\alpha$ waar maakt en dus maakt V een formule uit Ψ waar, dus $\Phi, \neg\alpha \models \Psi$.

Type II: regel $\wedge$R. De topsequent heeft nu de vorm $\Phi \circ \alpha \wedge \beta, \Psi$. De twee onmiddellijke opvolgers zijn: $\Phi \circ \alpha, \Psi$ en $\Phi \circ \beta, \Psi$. Met de inductiehypothese geldt $\Phi \models \alpha, \Psi$ en $\Phi \models \beta, \Psi$. Maar dan maakt iedere waardering die alle formules uit Φ waar maakt, hetzij een formule uit Ψ waar, hetzij zowel α als β: en dus ook $\alpha \wedge \beta$. Dus $\Phi \models \alpha \wedge \beta, \Psi$. □

Een open tableau heeft een tak τ die niet sluit. Zij V de waardering die alle propositieletters die links op de tak τ voorkomen, waar maakt en alle propositieletters die rechts voorkomen onwaar. Dan geldt voor elke formule φ: als φ links op de tak voorkomt, dan $V(\varphi) = 1$, en als φ rechts op de tak voorkomt, dan $V(\varphi) = 0$. bewering 5.4

Bewijs

Deze keer is een formule–inductie de passende bewijsvorm. We kunnen de bewering aantonen met inductie naar de opbouw van formules, gebruikmakend van het feit dat in een niet–sluitend semantisch tableau elke toepasbare regel ook werkelijk is toegepast.

Basisstap

Atomen. De bewering geldt dan per definitie voor de valuatie V. **Inductiestap**

Stel dat de bewering reeds opgaat voor formules φ en ψ. Weer beschouwen we slechts een enkel illustratief geval.

Negatie links: Als $\neg\varphi$ links op de tak voorkomt, dan moet daaronder een keer de regel $\neg_L$ zijn toegepast, zodat φ rechts voorkomt. Volgens de inductiehypothese geldt dan $V(\varphi) = 0$, zodat $V(\neg\varphi) = 1$ volgens de waarheidstabel voor $\neg$.

Conjunctie rechts: Als $\varphi \wedge \psi$ rechts op τ voorkomt, dan moet daaronder een keer de regel $\wedge_R$ zijn toegepast, zodat een splitsing ontstond. De tak τ volgt hierin de linker– of rechterkant, bijvoorbeeld de linkerkant waarin we φ rechts in de sequent terugvinden. Volgens de inductiehypothese geldt dan $V(\varphi) = 0$, zodat tevens $V(\varphi \wedge \psi) = 0$. □

Adequaatheid

Als $\varphi_1, \ldots, \varphi_n$ en ψ formules zijn, dan geldt: $\varphi_1, \ldots, \varphi_n \models \psi$ desda er is een gesloten tableau voor $\varphi_1, \ldots, \varphi_n \circ \psi$.

⇐ Dit is een speciaal geval van bewering 5.3.

⇒ Stel er is geen gesloten tableau voor deze sequent. Omdat een tableau op zich altijd te construeren is, moet er dan een open tableau zijn. Op grond van bewering 5.4 levert elke open tak daarvan een tegenvoorbeeld, zodat geldigheid is weerlegd. □

Predikaatlogica: Inleiding

Complexe uitdrukkingen ontstaan nu door combinatie van basispatronen.
De eerste manier waarop dit kan, is die van de propositielogica: we nemen gewoon alle eerdere connectieven over.

Andere complexe uitdrukkingen zijn 'uitdrukkingen van hoeveelheid'. Die zeggen iets over hoeveel individuen al dan niet aan zekere predikaten voldoen. Dergelijke uitdrukkingen kunnen we logisch weergeven met behulp van de zogenaamde *kwantoren* ∀ en ∃.
∀ wordt de *universele* kwantor genoemd en is te lezen als 'voor alle'.
∃ wordt de de *existentiële* kwantor genoemd en is te lezen als 'er is een'.

Het gebruik van kwantoren introduceert nog een nieuwigheid. Om de individuen aan te geven waarover gekwantificeerd wordt, markeren we de relevante posities in de betreffende zin met zogenaamde *individuele variabelen*, die we noteren met kleine letters x, y, z, ...

Alfabet
Het alfabet van een predikaatlogische taal L bestaat uit:
a een verzameling $\boldsymbol{C}$ van individuele constanten;
b een verzameling $\boldsymbol{P}$ van predikaatletters;
c een verzameling $\boldsymbol{F}$ van functieletters;
d logische symbolen ¬, ∧, ∨, →, ↔, ∀ en ∃;
e individuele variabelen;
f hulpsymbolen) en (.

Voor individuele constanten gebruiken we meestal kleine letters a, b, c, ..., of ook wel a_1, a_2, a_3, ...
Voor predikaatletters nemen we in het algemeen hoofdletters P, Q, R, ..., of ook wel P_1, P_2, P_3, ...
Voor functieletters worden meestal f, g, h, ... genomen, of ook wel f_1, f_2, f_3, ... Ook hier geven we de plaatsigheid aan door een hooggeplaatste index.

Syntaxis

Termen
De termen van de predikaatlogica worden als volgt geconstrueerd:
a individuele variabelen en constanten zijn termen;
b als f een k–plaatsige functieletter is en t_1, ..., t_k zijn termen, dan is $f(t_1, ..., t_k)$ ook een term;
c niets is een term dan op grond van a en b.

Formules
De formules van de predikaatlogica worden als volgt gedefinieerd:
a als P een k–plaatsige predikaatletter is en t_1, ..., t_k zijn termen, dan is $P(t_1, ..., t_k)$ een formule;
b als φ en ψ formules zijn, dan ook $\neg\varphi$, $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $(\varphi \rightarrow \psi)$ en $(\varphi \leftrightarrow \psi)$;
c als φ een formule is en x een individuele variabele, dan zijn $\forall x\, \varphi$ en $\exists x\, \varphi$ ook formules;
d niets is een formule dan op grond van a, b en c.

Grammaticale verschijnselen

Een voorkomen van een variabele x heet *vrij* als het niet in het bereik van een kwantor $\forall x$ of $\exists x$ ligt. De kwantorvariabelen zelf tellen we hier niet mee. Een kwantor $\forall x$ of $\exists x$ *bindt* de voorkomens van variabelen x in zijn bereik, tenminste, voorzover die nog vrij zijn. Als in een formule φ de variabele x één of meer keer vrij voorkomt en we maken een nieuwe formule door $\forall x$ of $\exists x$ voor φ te zetten, dan raakt x *gebonden*.

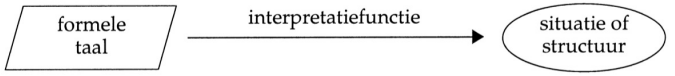
We vullen de terminologie nog wat aan: een *zin* of *gesloten formule* is een formule zonder vrije variabelen. Een formule waarin vrije variabelen voorkomen, heet een *open* formule.

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 φ 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 φ te vervangen door t.
  - $[t/x]\varphi$ wordt ook wel een instantie van φ genoemd

Een term t heet *vrij voor* x *in* φ, als in $[t/x]\varphi$ geen variabele van t in een gesubstitueerd voorkomen van t gebonden voorkomt.

In feite zijn substitutieproblemen altijd te ondervangen door over te gaan op zogenaamde *alfabetische varianten* van een formule, die ontstaan door gebonden variabelen door nieuwe te vervangen. Intuïtief gesproken tast zo'n vervanging de betekenis van een formule niet aan: gebonden variabelen zijn abstracte 'plaatsvullers' waarvan de precieze typografische aard irrelevant is.

Predikaatlogica: Semantiek



De formele taal is in het vorige hoofdstuk beschreven. Met deze taal willen we spreken over doorgaans niet–talige situaties of 'structuren'. Daartoe is echter een koppeling tussen beide nodig, in de vorm van een 'interpretatiefunctie' die betekenis geeft aan de onderdelen van de taal (predikaatletters, functieletters, eigennamen, enzovoort). Hiermee kan dan de betekenis van (termen en) formules bepaald worden, via hun *waarheidswaarde* in een structuur onder een interpretatie.

Structuren

Een structuur bestaat uit een *domein* waarop mogelijk *relaties* of *operaties* gedefinieerd zijn. Informeel kunnen we relaties tussen objecten uit het domein zien als beweringen die wel of niet het geval zijn, terwijl operaties op objecten andere objecten opleveren en dus geen beweringen uitdrukken.

Een *relationele* structuur bestaat uit een domein D van objecten met daarop één of meer *relaties*.

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

Een andere grote familie binnen de wiskunde wordt gevormd door structuren die uit een domein D bestaan met daarop één of meer *operaties* (functies). Deze worden *operationele* structuren genoemd. In

Waarheidsdefinitie

Interpretatiefunctie
Laat $\boldsymbol{D} = \langle D, \boldsymbol{R}, \boldsymbol{O}\rangle$ een structuur zijn. Een *interpretatiefunctie* I kent aan elke individuele constante c uit een predikaatlogische taal een speciaal object $I(c) \in \boldsymbol{O}$ toe (speciale objecten vinden we als nulplaatsige operaties terug in de verzameling $\boldsymbol{O}$), aan elke predikaatletter P een relatie $I(P) \in \boldsymbol{R}$ (van dezelfde plaatsigheid) en aan elke functieletter f een operatie $I(f) \in \boldsymbol{O}$ (van dezelfde plaatsigheid).

Een paar $(\boldsymbol{D}, I)$ heet een *model*.
Een (on)eindig model is een model met een (on)eindig domein.

Bedeling
Een *bedeling* b is een functie die aan elke variabele x een object $b(x) \in D$ toekent.

Een (concrete) structuur als bijvoorbeeld $\langle \mathbb{N}, \{<\}, \{+, \cdot\}\rangle$ schrijven we ook wel korter als $\langle \mathbb{N}, <, +, \cdot\rangle$, wanneer duidelijk is wat de relaties en wat de functies zijn. De bedeling $b[x \mapsto d]$ is de bedeling b waarbij d aan de variabele x wordt toebedeeld, ongeacht wat $b(x)$ voorheen ook was.

Waardering van termen
Laat $M = (\boldsymbol{D}, I)$ een model zijn en b een bedeling. De semantische waarden van termen zijn als volgt inductief gedefinieerd:
a $V_{M,b}(x) = b(x)$, voor variabelen x
(lees: de interpretatie – waarde, value – van variabele x in model M onder bedeling b is $b(x)$);
b $V_{M,b}(a) = I(a)$, voor individuele constanten a;
c $V_{M,b}(f(t_1, ..., t_k)) = I(f)(V_{M,b}(t_1), ..., V_{M,b}(t_k))$.

Waardering van formules
Laat $M = (\boldsymbol{D}, I)$ een model zijn en b een bedeling.
De waarheidswaarden van formules ontstaan als volgt:
a $M, b \vDash P(t_1, ..., t_m) \Leftrightarrow I(P)(V_{M,b}(t_1), ..., V_{M,b}(t_m))$
b $M, b \vDash \neg\varphi \quad \Leftrightarrow M, b \nvDash \varphi$
De overige connectieven gaan op dezelfde manier, met behulp van de eerdere waarheidstabellen. Voor de kwantoren is de uitleg als volgt:
c $M, b \vDash \exists x\ \varphi \quad \Leftrightarrow$ *er is een* $d \in D$ zodat $M, b[x \mapsto d] \vDash \varphi$
d $M, b \vDash \forall x\ \varphi \quad \Leftrightarrow$ *voor alle* $d \in D$ geldt $M, b[x \mapsto d] \vDash \varphi$.

Het paar (D, I) is een model van de formule φ als voor iedere bedeling b: $V_{(D,I),b}(\varphi) = 1$

Eindigheid
Laten $x_1, ..., x_k$ de vrije variabelen van φ zijn, b_1 en b_2 twee bedelingen met $b_1(x_i) = b_2(x_i)$, voor $i = 1, ..., k$. Dan geldt: $M, b_1 \vDash \varphi \Leftrightarrow M, b_2 \vDash \varphi$.

Substitutie
Voor alle termen t, t' geldt:

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

Geldig gevolg
Laat Σ een verzameling formules zijn en ψ een formule.

$\Sigma \vDash \psi$ (ψ *volgt uit* Σ) $\Leftrightarrow$ voor elk model M en elke bedeling b geldt: als $M, b \vDash \varphi$ voor elke $\varphi \in \Sigma$, dan $M, b \vDash \psi$.

Universeel geldige formules zijn dus waar in alle modellen M onder iedere bedeling b.
Het vorige in aanmerking genomen betekent de notatie $\vDash \Sigma$ dus: *een van de* formules in Σ is universeel geldig (niet noodzakelijk alle).
Als voor twee formules φ en ψ geldt dat $\vDash \varphi \leftrightarrow \psi$, dan heten φ en ψ *logisch equivalent*.

Modellen en axioma's

Theorie
$Th(M) = \{\varphi \mid \varphi$ is een zin en $M \vDash \varphi\}$

We kunnen theorieën weergeven door middel van axioma's.

Axiomaverzameling
Een formuleverzameling Σ *axiomatiseert* $Th(M)$ als voor alle zinnen φ geldt: $\varphi \in Th(M) \Leftrightarrow \Sigma \vDash \varphi$.

We zeggen ook wel dat Σ een *axiomatiek* of *axiomaverzameling* is voor $Th(M)$. Merk op dat $Th(M)$ zichzelf axiomatiseert. Uiteraard zijn we

Modelverzameling
a $MOD(\varphi) = \{M \mid M \vDash \varphi\}$

Evenzo voor een formuleverzameling Σ:

b $MOD(\Sigma) = \{M \mid M \vDash \varphi$ voor alle $\varphi \in \Sigma\}$

In het algemeen bestaat een model voor Σ uit een verzameling van een of meer structuren van de volgende vormen: een k–cykel ($k \geq 3$), een kopie van de gehele getallen, of een kopie van de negatieve gehele getallen, waarbij in elk punt nog kopieën van deze laatste vorm mogen aankomen.

Semantische tableaus

In een predikaatlogisch tableau kunnen zich twee situaties voordoen:
1 Het tableau sluit.
Het tableau is dan eindig en de gevolgtrekking die de sequent boven in het tableau uitdrukt is *geldig*.
2 Er is een niet–sluitende tak. Deze kan:
2.1 eindig afbreken, maar ook
2.2 oneindig doorlopen.
In beide gevallen beschrijft zo'n tak een tegenvoorbeeld en de gevolgtrekking die de sequent boven in het tableau uitdrukt, is *niet geldig*.

Predikaatlogica niet beslisbaar (stelling van Church 1936):
- geen algoritme om te bepalen of een formule afleidbaar is
- sommige fragmenten zijn wel beslisbaar

Adequaat voor semantische tableaus: een sequent heeft een gesloten tableau <=> de gevolgtrekking geldig is
Dit vinden men door fair scheduling van reductieregels: ze moeten allemaal voorkomen.

Afleidingen

Formules zie formularium

Uit $\forall y\ Ryy$ mogen we niet afleiden:
$\exists x\ \forall y\ Ryx$ omdat y niet vrij voor x is in $\forall y\ Ryx$

Een eenvoudige theorie

Substitutie in termen
Voor termen t en t' en variabele x geldt $V_{M,b}([t/x]t') = V_{M,b[x \mapsto V_{M,b}(t)]}(t')$.
Bewijs:
Aangezien substitutie in termen inductief definieerbaar is, kunnen we een bijpassend inductief bewijs leveren. Voor de overzichtelijkheid schrijven we in het volgende V in plaats van $V_{M,b}$. Basisstap
a Als $t' = x$, dan geldt $[t/x]t' = t$, en hebben we $V([t/x]t') = V(t)$. Anderzijds geldt ook $V_{b[x \mapsto V(t)]}(t') = V_{b[x \mapsto V(t)]}(x) = V(t)$. Dus de gewenste gelijkheid gaat op.
b Als $t' = y$, waarbij y een andere variabele is dan x of een constante, dan $[t/x]t' = y$, en hebben we $V([t/x]t') = V(y)$. Maar anderzijds geldt ook $V_{b[x \mapsto V(t)]}(t') = V_{b[x \mapsto V(t)]}(y) = V(y)$. Inductiestap
De inductiehypothese stelt dat het lemma reeds geldt voor een reeks termen t_i, $i = 1, ..., n$. Dan geldt voor $t' = f^n(t_1, ..., t_n)$ dat $V([t/x]t') = V(f^n([t/x]t_1, ..., [t/x]t_n)) = I(f^n)(V([t/x]t_1), ..., V([t/x]t_n))$ (waarheids-definitie), hetgeen volgens de inductiehypothese gelijk is aan $I(f^n)(V_{b[x \mapsto V(t)]}(t_1), ..., V_{b[x \mapsto V(t)]}(t_n))$. Anderzijds geldt ook $V_{b[x \mapsto V(t)]}(t') = V_{b[x \mapsto V(t)]}(f^n(t_1, ..., t_n)) = I(f^n)(V_{b[x \mapsto V(t)]}(t_1), ..., V_{b[x \mapsto V(t)]}(t_n))$. □

Substitutie in formules

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

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

Bewijs

Dit leveren we weer met formule–inductie naar φ. Voor de overzichtelijkheid schrijven we weer V in plaats van $V_{M,b}$.

Basisstap

a $\varphi = P(t_1, ..., t_n)$:

$V([t/x]P(t_1, ..., t_n)) = 1$
$\Leftrightarrow V(P([t/x]t_1, ..., [t/x]t_n)) = 1$
$\Leftrightarrow I(P)(V([t/x]t_1), ..., V([t/x]t_n))$ (waarheidsdefinitie)
$\Leftrightarrow I(P)(V_{b[x \mapsto V(t)]}(t_1), ..., V_{b[x \mapsto V(t)]}(t_n))$ (bewering 8.1)
$\Leftrightarrow V_{b[x \mapsto V(t)]}(P(t_1, ..., t_n)) = 1$ (waarheidsdefinitie)

Inductiestap

b $\varphi = \neg\psi$:

$V([t/x]\neg\psi) = 1$
$\Leftrightarrow V(\neg[t/x]\psi) = 1$
$\Leftrightarrow V([t/x]\psi) = 0$ (waarheidsdefinitie)
$\Leftrightarrow V_{M,b[x \mapsto V(t)]}(\psi) = 0$ (inductiehypothese)
$\Leftrightarrow V_{M,b[x \mapsto V(t)]}(\neg\psi) = 1$ (waarheidsdefinitie)

De overige connectieven zijn op eenzelfde manier te behandelen.

c $\varphi = \forall z\, \psi$:

We moeten twee gevallen nagaan.

1 x is niet vrij in φ.

Dan geldt $[t/x]\varphi = \varphi$ en dus $V([t/x]\varphi) = V(\varphi)$. Omdat x niet vrij voorkomt in φ, geldt volgens bewering 7.1 (eindigheid) $V_{b[x \mapsto V(t)]}(\varphi) = V(\varphi)$. Deze twee gelijkheden leveren inderdaad $V([t/x]\varphi) = V_{b[x \mapsto V(t)]}(\varphi)$.

2 x is vrij in φ.

Dan zal gelden $[t/x]\varphi = \forall z\, ([t/x]\psi)$ en dus $V([t/x]\varphi) = V(\forall z\, ([t/x]\psi))$. Volgens de waarheidsdefinitie geldt: $V(\forall z\, ([t/x]\psi)) = 1 \Leftrightarrow$ voor alle $d \in D$ $V_{b[z \mapsto d]}([t/x]\psi) = 1$.

De inductiehypothese vertelt ons hier dat voor ψ de bewering van het lemma reeds opgaat, voor willekeurige substituties van geschikte termen en willekeurige bedelingen. Derhalve geldt:

- $V_{b[z \mapsto d]}([t/x]\psi) = V_{b[z \mapsto d][x \mapsto V'(t)]}(\psi)$, waar $V' = V_{b[z \mapsto d]}$.

Omdat t vrij is voor x in φ, komt z niet in t voor en dus geldt $V'(t) = V_{b[z \mapsto d]}(t) = V(t)$. Bovendien, omdat x vrij is in φ, geldt zeker $x \neq z$. Daardoor geldt $b[z \mapsto d][x \mapsto V(t)] = b[x \mapsto V(t)][z \mapsto d]$. (Dit lijkt triviaal, maar voor $x = z$ hoeft dit niet waar te zijn!) De gelijkheid wordt nu

- $V_{b[z \mapsto d]}([t/x]\psi) = V_{b[x \mapsto V(t)][z \mapsto d]}(\psi)$.

Dit levert dan de volgende keten van equivalenties:

$V([t/x]\ \forall z\, \psi) = 1$
$\Leftrightarrow V(\forall z\, [t/x]\psi) = 1$ (definitie substitutie)
$\Leftrightarrow$ voor alle $d \in D$ $V_{b[z \mapsto d]}([t/x]\psi) = 1$ (waarheidsdefinitie)
$\Leftrightarrow$ voor alle $d \in D$ $V_{b[x \mapsto V(t)][z \mapsto d]}(\psi) = 1$ (volgens de zojuist gegeven analyse)
$\Leftrightarrow V_{b[x \mapsto V(t)]}(\forall z\, \psi) = 1$ (waarheidsdefinitie).

Het geval $\varphi = \exists z\, \psi$ gaat op eenzelfde manier. □

Prenexvormen

Er bestaat een logisch verband tussen de kwantoren $\forall$ en $\exists$. We kunnen met de waarheidsdefinitie laten zien dat een formule van de vorm $\forall x\, \neg\varphi$ dezelfde waarde heeft als de formule $\neg\exists x\, \varphi$ (voor elke M, b):

$V(\forall x\, \neg\varphi) = 1$
$\Leftrightarrow$ voor alle $d \in D$ geldt dat $V_{b[x \mapsto d]}(\neg\varphi) = 1$
$\Leftrightarrow$ voor alle $d \in D$ geldt dat $V_{b[x \mapsto d]}(\varphi) = 0$
$\Leftrightarrow$ er is geen $d \in D$ zodat $V_{b[x \mapsto d]}(\varphi) = 1$
$\Leftrightarrow V(\exists x\, \varphi) = 0$
$\Leftrightarrow V(\neg\exists x\, \varphi) = 1$

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

Lemma:

Als φ en ψ formules zijn en x een variabele is die *niet vrij voorkomt in* ψ 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 -> alfabetische variant

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

Lemma

Als φ en ψ formules zijn en x een variabele is die *niet vrij voorkomt in* ψ, dan zijn de volgende formules logisch equivalent:

$(\forall x\, \varphi) \rightarrow \psi$ en $\exists x\, (\varphi \rightarrow \psi)$	$\psi \rightarrow (\forall x\, \varphi)$ en $\forall x\, (\psi \rightarrow \varphi)$
$(\exists x\, \varphi) \rightarrow \psi$ en $\forall x\, (\varphi \rightarrow \psi)$	$\psi \rightarrow (\exists x\, \varphi)$ en $\exists x\, (\psi \rightarrow \varphi)$

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

Lemma:

Als φ en ψ formules zijn en x is een variabele die *niet vrij voorkomt in* ψ en y is een variabele die niet vrij voorkomt in φ en ψ, dan zijn de volgende formules logisch equivalent:

$(\exists x\, \varphi) \leftrightarrow \psi$ en
$((\exists x\, \varphi) \rightarrow \psi) \wedge (\psi \rightarrow (\exists x\, \varphi))$ en
$((\forall x\, (\varphi \rightarrow \psi)) \wedge (\exists x\, (\psi \rightarrow \varphi))$ en
$((\forall x\, (\varphi \rightarrow \psi)) \wedge (\exists y\, (\psi \rightarrow [y/x]\varphi))$ en
$\forall x\, ((\varphi \rightarrow \psi) \wedge (\exists y\, (\psi \rightarrow [y/x]\varphi)))$ en
$\forall x\, \exists y\, ((\varphi \rightarrow \psi) \wedge (\psi \rightarrow [y/x]\varphi))$

Prenexvorm

Een formule van de vorm $Q_1x_1 \ldots Q_nx_n\, \psi$, waarbij Q_i $(i = 1, ..., n)$ kwantoren zijn en ψ een formule is waarin geen kwantoren meer voorkomen, heet een *prenex*vorm met $Q_1x_1 \ldots Q_nx_n$ als *prefix* en ψ als *matrix*.

Prenexstelling

Voor elke formule φ bestaat er een prenexformule ψ zodat φ en ψ logisch equivalent zijn (dat wil zeggen $V(\varphi) = V(\psi)$).

Bewijs

Met inductie naar φ.

a $\varphi = P(t_1, ..., t_n)$: φ is reeds in prenexvorm.

b $\varphi = \neg\xi$: volgens de inductiehypothese is ξ te schrijven in prenexvorm, zeg $\xi = Q_1x_1 \ldots Q_nx_n\, \chi$, met χ kwantorvrij. Dan is φ equivalent met $\psi = Q_1'x_1 \ldots Q_n'x_n\, \neg\chi$, waarbij Q_j' de 'omgeklapte' Q_j is (dat wil zeggen $\forall$ wordt $\exists$ en andersom).

c $\varphi = \xi \wedge \chi$: volgens de inductiehypothese zijn ξ en χ te schrijven in prenexvorm, zeg $\xi = Q_1x_1 \ldots Q_nx_n\, \xi'$ en $\chi = K_1y_1 \ldots K_my_m\, \chi'$. Door geschikte alfabetische varianten te kiezen, kunnen we ervoor zorgen dat alle gekwantificeerde variabelen verschillend zijn. Dan is φ op grond van lemma 8.1 equivalent met $\psi = Q_1x_1 \ldots Q_nx_n\, K_1y_1 \ldots K_my_m\, (\xi' \wedge \chi')$. Het geval $\varphi = \xi \vee \chi$gaat analoog, en de gevallen voor $\xi \rightarrow \chi$ en $\xi \leftrightarrow \chi$ gaan als in de eerdere bespreking.

d $\varphi = \forall x\, \xi$: volgens de inductiehypothese is ξ in prenexvorm te schrijven, zeg $\xi = Q_1x_1 \ldots Q_nx_n\, \chi$, met χ kwantorvrij. Maar dan is φ equivalent met $\psi = \forall x\, Q_1x_1 \ldots Q_nx_n\, \chi$.

Het geval $\varphi = \exists x\, \xi$ gaat op eenzelfde manier. □

Fragmenten predikaatlogica

Een voorbeeld van zo'n fragment is de *monadische* taal. Hierin zijn alleen eenplaatsige predikaatletters toegestaan. De monadische predikaatlogica is voldoende om de zogenaamde 'syllogismen' uit de traditionele logica te behandelen.

Informeel kunnen we een *syllogisme* beschrijven als een gevolgtrekking bestaande uit twee aannames en één conclusie, alle drie van de vorm 'alle/geen/sommige A is/zijn B' (alle/geen/sommige objecten met eigenschap A heeft/hebben eigenschap B). In hoofdstuk 1 staat een voorbeeld van zo'n syllogisme, waarvan de predikaatlogische formalisering is:

$\forall x\, (Kx \rightarrow Rx)$	('alle kaaimannen zijn reptielen')
$\neg\exists x\, (Rx \wedge Fx)$	('geen reptiel kan fluiten')
dus	
$\neg\exists x\, (Kx \wedge Fx)$	('geen kaaiman kan fluiten')

Universele formules: alleen universele kwatoren in prefix

Horn–zin

Een *Horn–zin* is een universele zin van de vorm

$\forall x_1 \ldots \forall x_n\, ((A_1 \wedge \ldots \wedge A_k) \rightarrow B)$

waarbij $A_1, ..., A_k, B$ staan voor atomaire beweringen. Het is ook toegestaan dat het linker– of rechterlid van de implicatie leeg is.

Metatheorie

Adequaatheid Stelling

Als Σ een formuleverzameling is en φ een formule, dan geldt:

$\Sigma \models \varphi \Leftrightarrow$ er bestaat een gesloten tableau voor $\Sigma \circ \varphi$

Volledigheidsstelling:

Voor de predikaatlogica geldt dat voor iedere formuleverzameling Σ en iedere formule φ

$\Sigma \vdash \varphi$ desda $\Sigma \models \varphi$

Correctheid

Voor elke formuleverzameling Σ en formule φ geldt: $\Sigma \vdash \varphi \Rightarrow \Sigma \models \varphi$.

Volledigheid

Voor elke formuleverzameling Σ en formule φ geldt: $\Sigma \models \varphi \Rightarrow \Sigma \vdash \varphi$.

λ-calculus

De λ–calculus werd ontwikkeld terwille van een goede formalisering van het centrale begrip *functie* in de grondslagen van de wiskunde, een begrip dat eveneens centraal is gebleken in de informatica. Met functie samenhangende kernbegrippen zijn: *toepassing* of *applicatie* van een functie op een argument, alsmede het creëren van een functie door zogenaamde *abstractie* uit een voorschrift.

Ontstaan functionele programmeertalen

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

Numerieke functies
- Extensioneel (verzameling)
- Intentioneel (functievoorschrift)

Fundamentele bewerkingen
- abstractie: het maken van een functievoorschrift
- toepassen/ aanroepen van het functievoorschrift op parameter

Definitie 1 *Weze V een verzameling* **variabelen**. *De verzameling van λ-expressies Λ 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 Λ*

Functie in λ-calculus: λ formele-parameter.functievoorschrift
Aanroepen functie: (functie) actuele parameter
Verzameling van λ-expressies wordt genoteerd als Λ

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 x B → C wordt
g: A → (B → C) functie
g(a) = f_a en f_a (b) = f(a,b)

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

In het algemeen:
A_1 x … x A_n → B wordt A_1 → (A_2 → … (A_n → B)

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

- **Voordeel**:
  - Theorie te beperken tot functies met 1 argument
- **Nadeel**:
  - Leesbaarheid

Voorbeeld: λx. λy.(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)

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

Definitie 2

- *Voor een λ-expressie van de vorm*

$$\lambda x.M$$

zegt men dat
  - *λx 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 λ-expressie heten alle voorkomens van variabelen die niet gebonden zijn* **vrij**.

Definitie 3 *Weze $M \in \Lambda$. De verzameling $VV(M)$ van* **vrije variabelen** *van M wordt gedefinieerd door:*

- $\forall x \in V : VV(x) = \{x\}$
- $VV(\lambda x.N) = VV(N) \backslash \{x\}$
- $VV((M)N) = VV(M) \cup VV(N)$

Een λ-expressie zonder vrije variabelen heet **gesloten** *of een* **combinator**. *We noteren de verzameling van combinatoren als $\Lambda_0 \subset \Lambda$.*

Substitutie

formele parameter — actuele parameter

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

Definitie 4 *Beschouw twee λ-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:*

$[P/x]x = P$		*(S1)*
$[P/x]y = y$	*als* $y \in V \backslash \{x\}$	*(S2)*
$[P/x](F)Q = ([P/x]F)[P/x]Q$		*(S3)*
$[P/x]\lambda x.M = \lambda x.M$		*(S4)*
$[P/x]\lambda y.M = \lambda y.[P/x]M$	*als* $y \neq x$ *en* $y \notin VV(P)$	*(S5)*
$[P/x]\lambda y.M = \lambda z.[P/x][z/y]M$	*als* $y \neq x$ *en* $z \notin VV(P)$ *en* $y \in VV(P)$	*(S6)*

- 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-λ-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 λx!)
- Regel 5 zegt dat we, in het algemeen, als we in $\lambda y.M$ (alle) x'en moeten vervangen door P, we de λy 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 (verkeerdelijk) 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 (verkeerdelijk) gebonden worden door de $\lambda.y$ van $\lambda y.M$! Om ervoor te zorgen dat zo'n verkeerdelijke 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 $z \notin VV(P)$

β-gelijkheid

Definitie 5 *De relatie $=_\beta \subset \Lambda \times \Lambda$ wordt gedefinieerd door de onderstaande axiomas:*

$(\lambda x.M)P =_\beta [P/x]M$	*(β)*
$\lambda x.M =_\beta \lambda z.[z/x]M$ *als* $z \notin VV(M)$	*(α)*
$M =_\beta M$	*(reflexief)*
$M =_\beta N \Rightarrow N =_\beta M$	*(symmetrisch)*
$M =_\beta N \wedge N =_\beta L \Rightarrow N =_\beta L$	*(transitief)*
$M =_\beta M' \wedge P =_\beta P' \Rightarrow (M)P =_\beta (M')P'$	*(congruent)*
$M =_\beta M' \Rightarrow \lambda x.M =_\beta \lambda x.M'$	*(congruent)*

Axioma's 1 en 2 worden de β - en α-axioma's genoemd (historisch was β het 2de axioma)

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

Het α-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 λ-calculus

Hoewel de λ-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. λ-expressies, waarbij β-gelijkheid kan gebruikt worden om het gewenste effekt van die functies aan te tonen.

Definitie 6 *Beschouw λ-expressies F en M. De expressie $(F)^n M$ wordt inductief gedefinieerd door:*

$$\begin{aligned} (F)^0 M &\equiv M \\ (F)^{1+n} M &\equiv (F)(F)^n M \end{aligned}$$

Lemma 1 *Beschouw λ-expressies F en M. Voor alle $n, m \in \mathbf{N}$ geldt dat*

$$(F)^{n+m} M \equiv (F)^n (F)^m M \tag{2}$$

Bewijs. We gebruiken inductie op n.

1. Als $n = 0$ dan geldt dat

$$\begin{aligned} (F)^{0+m} M &\equiv (F)^m M \\ &\equiv M' && \text{neem } M' \equiv (F)^m M \\ &\equiv (F)^0 M' && \text{definitie 6} \\ &\equiv (F)^0 (F)^m M && \text{definitie } M' \end{aligned}$$

2. Stel dat (1) geldt voor $n = k$. Als $n = k + 1$ dan geldt dat

$$\begin{aligned} (F)^{1+k+m} M &\equiv (F)^{1+(k+m)} M \\ &\equiv (F)(F)^{k+m} M && \text{definitie 6} \\ &\equiv (F)(F)^k (F)^m M && \text{inductiehypothese} \\ &\equiv (F)(F)^k M' && \text{neem } M' \equiv (F)^m M \\ &\equiv (F)^{1+k} M' && \text{definitie 6} \\ &\equiv (F)^{1+k} (F)^m M && \text{definitie } M' \end{aligned}$$

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.

Definitie 7 *De zogenaamde "**Church getallen**" $\mathbf{c}_n$ ($n \in \mathbf{N}$) worden gedefinieerd door*

$$\mathbf{c}_n \equiv \lambda f.\lambda x.(f)^n x$$

Definitie 8 *Een numerieke functie $f : \mathbf{N}^p \to \mathbf{N}$ met $p \in N$ parameters is **λ-definieerbaar** als er een combinator F bestaat zodat*

$$((((F)\mathbf{c}_{n_1})\mathbf{c}_{n_2})\ldots)\mathbf{c}_{n_p} =_\beta \mathbf{c}_{f(n_1,n_2,\ldots,n_p)}$$

TB: $\mathbf{succ}(n) = n + 1$

λ-definieerbaar is. **Bewijs:**

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 $\mathbf{c}_n$:

$$(\mathbf{succ})\mathbf{c}_n =_\beta \lambda f.\lambda x.(f)voorschrift_n \tag{3}$$

waarbij $voorschrift_n$ een afkorting is voor $(f)^n x$.

Maar uit de definitie van een Church getal (herinner u de (intuïtieve) betekenis van een Church-getal; we hebben deze niet toevallig zo gedefiniëerd):

$$\mathbf{c}_n \equiv \lambda f.\lambda x.(f)^n x$$

geldt dat

$$\begin{aligned} ((\mathbf{c}_n)f)x &\equiv ((\lambda f.\lambda x.(f)^n x)f)x && (\beta) \\ &=_\beta (\lambda x.(f)^n x)x && (\beta) \\ &=_\beta (f)^n x \\ &\equiv voorschrift_n \end{aligned}$$

$(\mathbf{succ})\mathbf{c}_n =_\beta \lambda f.\lambda x.(f)((\mathbf{c}_n)f)x \qquad \mathbf{succ} \equiv \lambda n.\lambda f.\lambda x.(f)((n)f)x$

Vb ex vraag:

$$\begin{aligned} (\mathbf{succ})\mathbf{c}_0 &\equiv \underline{(\lambda n.\lambda f.\lambda x.(f)((n)f)x)\lambda f.\lambda x.x} && (\beta) \\ &=_\beta \lambda f.\lambda x.(f)(\underline{(\lambda f.\lambda x.x)f})x && (\beta) \\ &=_\beta \lambda f.\lambda x.(f)\underline{(\lambda x.x)x} && (\beta) \\ &=_\beta \lambda f.\lambda x.(f)x \\ &\equiv \mathbf{c}_1 \end{aligned}$$

$$\begin{aligned} (\mathbf{succ})\mathbf{c}_1 &\equiv \underline{(\lambda n.\lambda f.\lambda x.(f)((n)f)x)\lambda f.\lambda x.(f)x} && (\beta) \\ &=_\beta \lambda f.\lambda x.(f)(\underline{(\lambda f.\lambda x.(f)x)f})x && (\beta) \\ &=_\beta \lambda f.\lambda x.(f)\underline{(\lambda x.(f)x)x} && (\beta) \\ &=_\beta \lambda f.\lambda x.(f)(f)x \\ &\equiv \mathbf{c}_2 \end{aligned}$$

Stelling 1 *Definieer*

$$\textbf{plus} \equiv \lambda n.\lambda m.\lambda f.\lambda x.((n)f)((m)f)x$$

dan geldt voor alle $m, n \in \mathbf{N}$

$$((\textbf{plus})\mathbf{c}_n)\mathbf{c}_m =_\beta \mathbf{c}_{n+m}$$

Bewijs.

$$\begin{array}{lll}
((\textbf{plus})\mathbf{c}_n)\mathbf{c}_m & \equiv \underline{((\lambda n.\lambda m.\lambda f.\lambda x.((n)f)((m)f)x)\mathbf{c}_n)}\mathbf{c}_m & \text{definitie } \textbf{plus} \\
& =_\beta \underline{(\lambda m.\lambda f.\lambda x.((\mathbf{c}_n)f)((m)f)x)\mathbf{c}_m} & (\beta) \\
& \equiv (\lambda m.\lambda f.\lambda x.\underline{((\lambda f.\lambda x.(f)^n x)f)((m)f)x)\mathbf{c}_m} & \text{definitie } \mathbf{c}_n \\
& =_\beta \underline{(\lambda m.\lambda f.\lambda x.(\lambda x.(f)^n x)((m)f)x)\mathbf{c}_m} & (\beta) \\
& =_\beta \underline{(\lambda m.\lambda f.\lambda x.(f)^n((m)f)x)\mathbf{c}_m} & (\beta) \\
& =_\beta \underline{\lambda f.\lambda x.(f)^n((\mathbf{c}_m)f)x} & (\beta) \\
& \equiv \lambda f.\lambda x.(f)^n\underline{((\lambda f.\lambda x.(f)^m x)f)x} & \text{definitie } \mathbf{c}_m \\
& =_\beta \underline{\lambda f.\lambda x.(f)^n(\lambda x.(f)^m x)x} & (\beta) \\
& =_\beta \lambda f.\lambda x.(f)^n(f)^m x & (\beta) \\
& \equiv \lambda f.\lambda x.(f)^{n+m}x & \text{lemma 1} \\
& \equiv \mathbf{c}_{n+m} & \text{definitie } \mathbf{c}_{n+m}
\end{array}$$

Lemma 2 *Voor alle* $n, m \in N$ *geldt dat*

$$((\mathbf{c}_n)f)^m y =_\beta (f)^{n\times m} y \qquad (4)$$

Bewijs. We gebruiken inductie over m.

1. Als $m = 0$ dan wordt (4):

$$\begin{array}{lll}
((\mathbf{c}_n)f)^m y & \equiv ((\mathbf{c}_n)f)^0 y & \\
& \equiv y & \text{definitie 6} \\
& \equiv (f)^0 y & \text{definitie 6} \\
& \equiv (f)^{n\times 0} y &
\end{array}$$

2. Stel dat (4) geldt voor $m = k$:

$$\begin{array}{lll}
((\mathbf{c}_n)f)^{k+1}y & \equiv ((\mathbf{c}_n)f)((\mathbf{c}_n)f)^k y & \text{definitie 6} \\
& \equiv ((\mathbf{c}_n)f)(f)^{n\times k}y & \text{inductiehypothese} \\
& \equiv ((\lambda f.\lambda x.(f)^n x)f)(f)^{n\times k}y & \text{definitie } \mathbf{c}_n \\
& =_\beta \underline{(\lambda x.(f)^n x)(f)^{n\times k}y} & (\beta) \\
& =_\beta (f)^n(f)^{n\times k}y & (\beta) \\
& \equiv (f)^{n+n\times k}y & \text{lemma 1} \\
& \equiv (f)^{(n+1)\times k}y &
\end{array}$$

Stelling 2 *Definieer*

$$\textbf{times} \equiv \lambda n.\lambda m.\lambda f.(n)(m)f$$

dan geldt voor alle $m, n \in \mathbf{N}$

$$((\textbf{times})\mathbf{c}_n)\mathbf{c}_m =_\beta \mathbf{c}_{n\times m}$$

Bewijs.

$$\begin{array}{lll}
((\textbf{times})\mathbf{c}_n)\mathbf{c}_m & \equiv \underline{((\lambda n.\lambda m.\lambda f.(n)(m)f)\mathbf{c}_n)}\mathbf{c}_m & \text{definitie } \textbf{times} \\
& =_\beta \underline{(\lambda m.\lambda f.(\mathbf{c}_n)(m)f)\mathbf{c}_m} & (\beta) \\
& \equiv (\lambda m.\lambda f.\underline{(\lambda f.\lambda x.(f)^n x)(m)f)\mathbf{c}_m} & \text{definitie } \mathbf{c}_n \\
& =_\beta \underline{(\lambda m.\lambda f.\lambda x.((m)f)^n x)\mathbf{c}_m} & (\beta) \\
& =_\beta \lambda f.\lambda x.((\mathbf{c}_m)f)^n x & (\beta) \\
& =_\beta \lambda f.\lambda x.(f)^{m\times n}x & \text{lemma 2} \\
& \equiv \mathbf{c}_{n\times m} & \text{definitie } \mathbf{c}_{n\times m}
\end{array}$$

Definitie 9

$$\begin{array}{ll}
\textbf{true} & \equiv \lambda t.\lambda f.t \\
\textbf{false} & \equiv \lambda t.\lambda f.f \\
\textbf{if} & \equiv \lambda c.\lambda d.\lambda e.((c)d)e \\
\textbf{iszero} & \equiv \lambda n.((n)\lambda x.\textbf{false})\textbf{true} \\
\textbf{cons} & \equiv \lambda a.\lambda d.\lambda z.((z)a)d \\
\textbf{car} & \equiv \lambda a.\lambda d.a \\
\textbf{cdr} & \equiv \lambda a.\lambda d.d
\end{array}$$

- **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.

$$\begin{array}{lll}
((\textbf{true})A)B & \equiv \underline{((\lambda t.\lambda f.t)A)}B & \text{voor willekeurige } A, B \in \Lambda \\
& =_\beta \underline{(\lambda f.A)B} & \\
& =_\beta A &
\end{array}$$

$$\begin{array}{lll}
((\textbf{false})A)B & \equiv \underline{((\lambda t.\lambda f.f)A)}B & \text{voor willekeurige } A, B \in \Lambda \\
& =_\beta \underline{(\lambda f.f)B} & \\
& =_\beta B &
\end{array}$$

- **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.

$$\begin{array}{lll}
(\textbf{if})\textbf{true} & \equiv \underline{(\lambda c.\lambda d.\lambda e.((c)d)e)\textbf{true}} & \\
& =_\beta \lambda d.\lambda e.((\textbf{true})d)e & \\
& \equiv \lambda d.\lambda e.\underline{((\lambda t.\lambda f.t)d)e} & \\
& =_\beta \lambda d.\lambda e.\underline{(\lambda f.d)e} & \\
& =_\beta \lambda d.\lambda e.d & \\
& \equiv \textbf{true} &
\end{array}$$

- **iszero**: gegeven een Church getal ($\lambda n.$), dan (het voorschrift volgt) passen we het Church getal n toe op de argumenten $\lambda x.\textbf{false}$ en **true**. Het eerste argument $\lambda x.\textbf{false}$ is een λ-expressie die, om het even wat het argument is, steeds **false** teruggeeft. Herinner u nu de (intuïtieve) 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 $\lambda x.\textbf{false}$ (die steeds **false** teruggeeft!), het argument **true**. M.a.w., vanaf het moment dat $\lambda x.\textbf{false}$ 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**.

$$\begin{array}{lll}
(\textbf{iszero})\mathbf{c}_0 & \equiv \underline{(\lambda n.((n)\lambda.\textbf{false})\textbf{true})\mathbf{c}_0} & \\
& =_\beta \underline{((\mathbf{c}_0)\lambda.\textbf{false})\textbf{true}} & \\
& \equiv \underline{((\lambda f.\lambda x.x)\lambda.\textbf{false})\textbf{true}} & \\
& =_\beta \underline{(\lambda x.x)\textbf{true}} & \\
& =_\beta \textbf{true} &
\end{array}$$

$$\begin{array}{lll}
(\textbf{iszero})\mathbf{c}_n & \equiv \underline{(\lambda n.((n)\lambda.\textbf{false})\textbf{true})\mathbf{c}_n} & \text{stel } n > 0 \\
& =_\beta \underline{((\mathbf{c}_n)\lambda f.\textbf{false})\textbf{true}} & \\
& \equiv \underline{((\lambda f.\lambda x.(f)^n x)\lambda f.\textbf{false})\textbf{true}} & \\
& =_\beta \underline{(\lambda x.(\lambda f.\textbf{false})^n x)\textbf{true}} & \\
& =_\beta \underline{(\lambda x.\textbf{false})\textbf{true}} & \\
& =_\beta \textbf{false} &
\end{array}$$

- **cons**: gegeven twee argumenten ($\lambda a.$, de car, en $\lambda d.$, de cdr), dan (het voorschrift volgt) geeft **cons** een λ-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 λ-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 begijpen). **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 c**a**r) terug.
- **cdr**: gegeven twee argumenten ($\lambda a.$ en $\lambda d.$), dan (het voorschrift volgt) geeft **cdr** het eerste argument (de c**d**r) terug.

$$\begin{array}{lll}
((\textbf{cons})A)B & \equiv \underline{(\lambda a.\lambda d.\lambda z.((z)a)d)A)}B & \\
& =_\beta \underline{(\lambda d.\lambda z.((z)A)d)B} & \\
& =_\beta \lambda z.((z)A)B &
\end{array}$$

$$\begin{array}{lll}
((\textbf{cons})A)B)\textbf{car} & =_\beta \underline{(\lambda z.((z)A)B)\textbf{car}} & \\
& =_\beta ((\textbf{car})A)B & \\
& \equiv ((\lambda a.\lambda d.a)A)B & \\
& =_\beta (\lambda d.A)B & \\
& =_\beta A &
\end{array}$$

$$\begin{array}{lll}
(((\textbf{cons})A)B)\textbf{cdr} & =_\beta \underline{(\lambda z.((z)A)B)\textbf{cdr}} & \\
& =_\beta ((\textbf{cdr})A)B & \\
& \equiv ((\lambda a.\lambda d.d)A)B & \\
& =_\beta (\lambda d.d)B & \\
& =_\beta B &
\end{array}$$

p als cons

$$\begin{aligned}\mathbf{nextp} &\equiv \lambda p.((\mathbf{cons})(\mathbf{succ})(p)\mathbf{car})(p)\mathbf{car}\\ &\equiv \lambda p.\lambda z.((z)(\lambda n.\lambda f.\lambda x.(f)((n)f)x)(p)\mathbf{car})(p)\mathbf{car}\end{aligned}$$

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:

(p)car -> geeft de car van p weer
(succ)(p)car geeft de succesor van de car van de cons van p weer
((cons)(succ)(p)car) (p)car geeft nieuwe cons met als car de succesor van de car van p en de cdr de car van p

$$\mathbf{pred} \equiv \lambda n.(((n)\mathbf{nextp})((\mathbf{cons})\mathbf{c}_0)\mathbf{c}_0)\mathbf{cdr}$$

n keer nextp toepassen op de conns (0 . 0) daarvan de cdr nemen

pred(3): 3 keer nextp van (0.0) = (3.2) => cdr hiervan is 2

$$\mathbf{minus} \equiv \lambda m.\lambda n.((m)\mathbf{pred})n$$

m keer pred toepassen op n (m.a.w m keer -1 op n => n - m)

FIXPUNTEN EN RECURSIE

Definitie 10 *Voor een functie*

$$f : D \to D$$

heet $x \in D$ een **fixpunt** *van f als en slechts als*

$$f(x) = x$$

Voor λ-expressies wordt dit:

Definitie 11 *$X \in \Lambda$ is een fixpunt van $F \in \Lambda$ als en slechts als*

$$(F)X =_\beta X$$

Stelling 3

1. *Iedere λ-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.

Gevolg
Beschouw de combinator $F \equiv \lambda f.\text{body}_F$ en zijn fixpunt $X_F \equiv (\mathbf{Y})F$
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 $(F)X_F \equiv (\lambda f.\text{body}_F)X_F$
$=_\beta \{X_F / f\}\text{body}_F$ (β regel).

Definitie wiskunde: n! = 1 als n = 0 & n! = n (n-1)! als n > 0

```
(define fac (lambda n)
              (if (= x 0)
                  1
                  (* x (fac (- x 1)))))
```

Dit kan zonder veel moeite omgezet worden naar een λ-expressie:

$$\mathbf{fac} \equiv \lambda n.(((\mathbf{if})(\mathbf{iszero})n)\mathbf{c}_1)((\mathbf{times})n)(\mathbf{fac})(\mathbf{pred})n$$

Deze lambda-expresssie is niet geldig want het is circulair, fac is gedefineeird in de termen van fac zelf -> opl: beta-gelijkheid

$$\mathbf{fac} =_\beta \lambda n.(((\mathbf{if})(\mathbf{iszero})n)\mathbf{c}_1)((\mathbf{times})n)(\mathbf{fac})(\mathbf{pred})n$$

We zoeken dus een λ-expressie **fac** die zodanig is dat (7) geldt. Echter, als (7) geldt, en we maken gebruik van het feit dat voor een willekeurige λ-expressie M geldt dat $\lambda x.(M)x =_\beta (\lambda f.\lambda x.(f)x)M$, dan moet ook:

$$\mathbf{fac} =_\beta (\lambda f.\lambda n.(((\mathbf{if})(\mathbf{iszero})n)\mathbf{c}_1)((\mathbf{times})n)(f)(\mathbf{pred})n)\mathbf{fac} \qquad (8)$$

gelden.

Mits we nu in (8) de volgende vervanging doorvoeren (en merk op dat (9) niet circulair is!):

$$\mathbf{FAC} \equiv \lambda f.\lambda n.(((\mathbf{if})(\mathbf{iszero})n)\mathbf{c}_1)((\mathbf{times})n)(f)(\mathbf{pred})n \qquad (9)$$

dan kan (8) met behulp van (9) herschreven worden als:

$$\mathbf{fac} =_\beta (\mathbf{FAC})\mathbf{fac}$$

M.a.w. de gezochte λ-expressie **fac** is een fixpunt van de functie **FAC**!! Zoals eerder reeds vermeld hebben we een manier om, gegeven een λ-expressie (in dit geval, **FAC**), een fixpunt voor deze λ-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 definiëren in de λ-calculus!

Veralgemening:
- definieer combinator in kleine letters
- combinator kleine letters -> combinator in grote letters (door introductie parameter)
- de combinator in kleine letters is FIXPUNT voor combinator in grote letters