# LOGICA EXAM 2022-2023

## Prof M. Denecker

Liam Luys

*KU Leuven*

Veel succes met het examen!
Ik heb geprobeerd een samenvatting te maken van de cursustekst.
Geen garantie dat alles er in staat en/of juist is, doe zeker de boeken ook open.
Foutje gevonden? Laat het zeker weten, dan pas ik het aan!

# Samenvatting

vrijdag 19 mei 2023 22:50

***Liam Luys – r0934449 - 1e BA Informatica***
***Logica voor informatici***

## Oefenzittingen / LogicPalet

Zorg zeker dat de oefenzittingen duidelijk zijn en je oefeningen op LogicPalet kan begrijpen.
Dit hielp mij zeker bij een aantal examenvragen

Ik leg in deze samenvatting geen focus op de oefeningen, maar wees ervan bewust dat deze (bijna) even belangrijk zijn als de theorie!
Het examen legt in mijn oven veel meer focus op oefeningen en bewijzen, maar je hebt meestal deze theorie nodig om ze te begrijpen / kunnen maken.

## Bewijzen

Bewijzen in de cursus heb ik niet steeds overgebracht in de samenvatting.
Best in de cursus dus kijken naar bewijzen en uitleg daarrond.

## Samenvatting

Ik maakte deze samenvatting tijdens het studeren voor het examen.
Mij is het zo gelukt, maar weet dat niet alles hier volledig in staat.

## Oude Examens

Bekijk zeker vorige examens voor je start

# Hoofdstuk 1

vrijdag 30 september 2022 8:30

***Liam Luys – r0934449 - 1e BA Informatica***
***Logica voor informatici***

KU LEUVEN

## 1. Logica en zijn rol in informatica

### 1.1 De rol van Logica in de Informatica

**Betekenis Logica**
- Formele taal om informatie uit te drukken, wiskundige principes bij vastleggen van de syntax en betekenis.
- Kunst of wetenschap van correct redeneren

**Verband betekenissen?**
- Redeneren gaat altijd over informatie
- Wiskundige taal om voor te stellen
- De studie van bepaalde info

**Toepassingen van Logica**
- /
- Declaratief Problem Solving

### 1.2 Geschiedenis van Logica

*Niet echt kennen*

**Waarom zijn ze er mee begonnen?**
- Mensen maakte zich zorgen over structuur van wiskunde
- Nood aan grondslagen voor wiskunde, een formele taal om voor te stellen

**Predicatenlogica**
- "Predicatenlogica is wiskundig-formele logica waarin expliciet predicaten voorkomen, waarmee eigenschappen van en relaties tussen verzamelingen objecten worden beschreven."

- Eerste versie: Frege's Begriffschrift
    - Beperkte set van operatoren om complexe gedachten te construeren

**Rol in Computerwetenschappen**
- Belangrijke rol van logica in de computerwetenschappen
- Veel mensen in de computerwetenschappen waren logici en werkte mee aan apparaten en software

### 1.3 Toepassingen van Logica in de Informatica

**Booleaanse Algebra**
- Synoniem **propositielogica**
- Gebruikt in hardware en processoren
    - Transistoren --> 1 of 0
    - Circuits met "en" en "of" manieren door combinatie transistoren

**Relationele Databanken**
- Meest courante toepassing in de logica!
- Informaticasystemen
- Gebaseerd op logica

**Formele Specificaties**
- Beschrijving van wat de software moet doen
- Redeneersystemen en eigenschappen van het systeem in logische taal

**Programmaverificatie**
- Bewijzen van correctheid van programma's
- Correctheidscondities in logica geformuleerd

**Programmageneratie**
- Via interactief proces
- Arbeidsintensief en lange tijd nodig
- Voor kritische software die geen fouten mag bevatten
    - Raket lanceren
    - Kanker bestralen

**Declaratief Problem Solving**
- Formele specificaties
- Je hebt informatie, wetten en probleem
- Specificatie van de info van het domein

**Het Semantische Web**
- Internet als kennisbank
- Complexe vragen oplossen mbv logica

…

=> Logica is belangrijk in de informatica
- Vragen oplossen
- Query-talen

**Doelstellingen Cursus**
- Leren om info uit de drukken in logica
- Verschillende vormen van redeneren in logica en hun toepassingen in de informatica

## 1.4 Waarom vertrekken we van predicatenlogica?

- Duidelijke taal met beperkt aantal elementaire taalconstructies

**Taalconstructies**
- ∧ Conjunctie --> "en", de bewering is waar als beide beweringen waar zijn (Waarheidstabel)
- ∨ Disjunctie --> "of", de bewering is waar als één of beide beweringen waar zijn(inclusieve of)
- ¬ Ontkenning --> Waar als de bewering onwaar is
- ⇒ Implicatie --> "Als P waar is, dan is ook Q waar" (Niet perse oorzakelijk verband)
- ⇔ Equivalentie --> Als beide waar of onwaar zijn

## 1.5 Notaties en Basisbegrippen

*Niet zo belangrijk, zie cursus*

## 1.6 Verzamelingen en relaties

*Gekend van BRI, anders zie cursus*

## 1.7 Inductieve en Recursieve Definities, inductief Bewijs

Vaak eenduidige definities van verzamelingen

**Uitzonderingen**
- Verzamelingen die je construeert uit info die al in de verzameling zit
- Vb: reeks van Fibonnaci, verzameling natuurlijke getallen

**Inductieve verzameling definiëren**
- Als limiet van een inductie-proces
- Als kleinste verzameling die aan de eigenschappen voldoet
- Als doorsnede van verzamelingen die aan de eigenschappen voldoen

Bewijs is mogelijk via inductie

*Nagekeken en vervolledigd op 28-12-2022*

# 2. De Predicatenlogica

## 2.1 Inleiding tot Predicatenlogica

**Logische Symbolen in de Predicatenlogica**

| | |
|---|---|
| $\ldots \wedge \ldots$ | ...en ... |
| $\ldots \vee \ldots$ | ...of ... (of allebei) |
| $\neg \ldots$ | het is niet waar dat ... |
| $\ldots \Rightarrow \ldots$ | als ... dan ... |
| $\ldots \Leftrightarrow \ldots$ | ...als en slechts als ... |
| $\forall \ldots\ldots$ | voor alle ... geldt dat ... |
| $\exists \ldots\ldots$ | er bestaat ... zodat ... |
| $\ldots = \ldots$ | ...is gelijk aan ... |

**Databank**
- Verzameling van Tabellen (=Relaties)
- Bevat data
  - Simpele beweringen die waar zijn in de databank

**Relatie**
- Tabel in een databank
- Bevat **Atomen**
  - tupels die tot de relatie behoren

  Notatie atoom: R(t,s) waarbij (t,s) tot de relatie R behoort

**<u>Een databank geeft de waarheidswaarde van logische zinnen in de structuur</u>**

## 2.2 Syntax van Predicatenlogica

Logische formule = String bestaande uit symbolen uit reeksen van tekens

### Begrippen in de predicatenlogica

**Types van Symbolen**
- Hulptekens
  - Haakjes, komma's, dubbele punt --> Helpt om te formuleren, om inzicht te creëren

- Logische Symbolen *(Betekenis ligt vast in de logica)*
  - Logische Connectieven
  - Kwantoren
  - Gelijkheidssymbool

  *Zie bovenaan, de bekende logische symbolen in de logica*

- Niet-logische symbolen *(Betekenis ligt niet vast in de logica)*
  - Objectsymbolen / Constante symbolen (c/0:)
    - Variabelen x,y ; Ray; …
    - Is een functiesymbool met ariteit 0

- Predicaatsymbolen / Relatiesymbolen (P/n)
  - Vast aantal argumenten = Ariteit n
  - Propositioneel symbool = ariteit van 0 (P/0)
  - Slaagscore/2 ; Koppels/2; Trio's/3

- Functiesymbolen / Functoren (F/n:)
  - Vaste Ariteit n
  - Zal ook een "resultaat" in tupel hebben, niet meegeteld in ariteit!
  - Dubbele punt achter F/n om aan te geven dat het een functie is!
  - LeeftijdVan/1: ; PlusGetallen/2: ; AantalBoetesAuto/1:

**Vocabularium**
- Verzameling van alle niet-logische symbolen van een structuur
- Symbool Sigma

## Termen en Formules

**Term**
- String die (inductief) gedefinieerd wordt als volgt:
  - Een 0-voudig functiesymbool (=constante) c is een term
  - Als $t_1$, …., $t_n$ termen zijn en G een n-voudig functiesymbool, dan is G($t_1$, …, $t_n$) een term

- Alle termen zijn eindig, oneindig kan niet!

**Formule**
- String die (inductief) gedefinieerd wordt als volgt:
  - Als $t_1$, …., $t_n$ termen zijn en P/n een relatiesymbool, dan is P($t_1$, …, $t_n$) een formule
    - **Dit is een atoom!**

  - Als t, s termen zijn, dan is t = s een formule
    - Dit is het gelijkheidsatoom

  - Als A en B formules zijn, dan zijn operaties op A en B ook formules

  - Als x een variabele is en A een formule, dan zijn (∃x: A) en (∀x: A) ook formules

- Alle formules zijn eindig, oneindig kan niet!

**Abstracte Syntax-Boom**
- Visuele manier om structuur van een term of formule weer te geven

**Termen vs Atomen?**
- Termen --> Detoneren Objecten
  - Kan niet 'waar of onwaar' zijn!

- Atomen / Formules --> Detoneren Feiten
  - Kan waar of onwaar zijn

**Vrije Symbolen vs Gebonden Symbolen**
- Vrij symbool
  - Symbool dat zich in een formule niet bevindt onder een kwantor aan dat symbool
  - Heeft een vrij voorkomen

- Gebonden Symbool
  - Symbool dat niet vrij is

## Predikatentaal in een Vocabularium ∑

**Term over ∑**
- t is term over ∑ als elk vrij symbool van t behoort tot ∑

**Zin over ∑**
- Formule A is zin over ∑ als elk vrij symbool van A behoort tot ∑
- Indien er geen vrije symbolen staan die geen constantesymbolen zijn in ∑.
- Elk symbool moet gebonden zijn of een object/constantesymbool zijn

**Theorie over ∑**
- Verzameling van zinnen over ∑

**Predikatenlogicataal van ∑**
- Verzameling van alle zinnen over ∑

## Soorten Formules

**Atoom**
- Drukt uit dat een tupel tot een relatie behoort
- T(x,y)

**Gelijkheidsatoom**
- Drukt een gelijkheid uit met gelijkheidsrelatie =
- X = Y

**Negatie, Implicatie, Conjunctie, Disjunctie**

**Niet gelijk aan** --> bestaat niet als predikaat!

## Prefix / Infix

**t1 = t2**
- Infixnotatie van =(t1, t2)
- Gelijkheid staat in midden tussen argumenten

**R(t1,t2)**
- Prefixnotatie
- R staat voor de argumenten

## Haakjes

- Bieden vaak duidelijkheid in formule
- Maken het makkelijker leesbaar
- Kunnen worden weggelaten in sommige gevallen

**Voorrangsregels** predicaten
- Zoals in de wiskunde met berekeningen.
- Om haakjes weg te kunnen laten.
- Bij verschillende connectieven

  **Niet, en, of, implicatie, equivalentie, voor alle, er bestaat** *("niet" bindt dus het sterkst!)*

- Bij gelijke connectieven

  **Binding rechts is sterker dan binding links!**

## 2.3 Formele Semantiek van Predikatenlogica

**Studie van Betekenis**
- Semantiek --> Betekenis van een woord of zin
- Pragmatiek --> Betekenis van de maker/gebruiker van de zin

**Formele Semantiek** = Wiskundige theorie die de betekenis van alle logische formules beschrijft

Waarom is formele semantiek nodig?
- Oneindig veel formules over oneindig veel vocabularia
  - Niet mogelijk om allemaal met intuïtie te begrijpen

- Redeneersystemen ontwikkelen voor logica
  - Vereist een formele semantiek om correct te zijn zonder interpretatieverandering

**Structuren U** *(over vocabularium)*
- Wiskundige objecten, voorstellingen van "situaties", een "stand van zaken"
- Bepaalde formule A kan waar zijn binnen een structuur (waarheidsrelatie)
  - U |= A
- Structuur kent een waarde toe aan elk symbool in het vocabularium
- Structuur bepaald het domein $D_u$

Formule kan dus waar zijn in een bepaalde structuur
- Er kan een structuur bestaan waarin ze niet waar is

**Definitie 2.3.1.** Een **structuur** $\mathfrak{A}$ bestaat uit een niet-lege verzameling $D_{\mathfrak{A}}$, het **domein** of **universum** van $\mathfrak{A}$, en een toekenning van waarden $\tau^{\mathfrak{A}}$ aan niet-logische symbolen $\tau$:

- De waarde $c^{\mathfrak{A}}$ voor een objectsymbool $c$ is een element uit het domein $D_{\mathfrak{A}}$.
- De waarde $F^{\mathfrak{A}}$ voor een functie-symbool $F/n$ : is een functie $F^{\mathfrak{A}} : {D_{\mathfrak{A}}}^n \to D_{\mathfrak{A}}$. Dit is een functie die n-tallen $(a_1, \ldots a_n)$ uit het domein op elementen van het domein afbeeldt.
- De waarde $P^{\mathfrak{A}}$ voor een predikaatsymbool $P/n$ is een $n$-voudige relatie $P^{\mathfrak{A}}$ in $D_{\mathfrak{A}}$, dus $P^{\mathfrak{A}} \subseteq {D_{\mathfrak{A}}}^n$.

We noemen $\tau^{\mathfrak{A}}$ de **waarde** of de **interpretatie** van $\tau$ in $\mathfrak{A}$.

De verzameling van symbolen symbolen die een interpretatie hebben in $\mathfrak{A}$ wordt genoteerd als $\Sigma(\mathfrak{A})$. We zeggen dat $\mathfrak{A}$ een structuur is over $\Sigma$ indien $\Sigma = \Sigma_{\mathfrak{A}}$.

### Structuren van Propositionele Vocabularia

**Propositionele Vocabularium**
- Vocabularium bestaande uit
  - Propositionele symbolen
  - Relatiesymbolen van ariteit 0 *(Dit zijn geen constantesymbolen!)*
    - {} = f
    - {()} = t

- Waarde van symbool in structuur is altijd f of t

- {(Zon, f), (Regen,t)}
  - Structuur U over ∑ (= {zon, regen}) die aangeeft dat het regent en de zon niet schijnt.
  - Kan ook eenvoudiger weergegeven worden door enkel de symbolen die waar zijn in U weer te geven
    - {Regen}

*Zie hoofdstuk 3*

## Herbrand-structuren

Structuur is **Herbrand-structuur** als
- Het domein de verzameling van termen over het vocabularium is
- Voor elk objectsymbool $c \in \Sigma$ geldt $c^u = c$. *(de interpretatie in de structuur is het symbool zelf)*
- Voor elk functiesymbool $F \in \Sigma$ is $F^u$ de functie die termen $t_1, \ldots, t_n$ over $\Sigma$ afbeeldt op de term $F(t_1, \ldots, t_n)$.

**Eigenschappen**
- Waarde van de term is de term zelf
- Waarde van objectsymbool is zichzelf
- Waarde van functiesymbool is functie

## Databankstructuren

**Databankstructuur** DB
- Herbrandstructuur $U_{DB}$
  - Geen functiesymbolen van ariteit groter dan 0
  - Vocabularium bestaat uit alle tabelpredicaten en alle objectnamen
  - Waardes van predicaten komen overeen in de structuur als in het vocabularium

**Correcte databank**
- Formele databankstructuur isomorf met de ware stand van zaken onder de informele interpretatie van alle symbolen

## Waarde van Termen en Formules in een Structuur

Symbols(U) = Verzameling symbolen met een interpretatie in U
Vrij(U) = Verzameling symbolen zonder interpretatie in U

**Interpretatie structuur**
- Structuur U interpreteert een term t of logische formule A indien elk vrij symbool van t of A een interpretatie heeft in U
  - Indien Vrij(U) deelverzameling van Symbols(U)

**Interpretatie $t^U$ van t in U door inductie op aantal symbolen van t**
- Als t objectsymbool is, dan $t^U$ de exacte interpretatie van t in U
- Als t een term $F(t_1, \ldots, t_n)$ is, dan $t^U = F^U(t_1{}^U, \ldots t_n{}^U)$, de functie $F^U$ toegepast op bestaande interpretaties van $t_1, \ldots, t_n$

**Formule A waar in structuur U**
- U |= A
- A is waar in U
- A is voldaan in U
- U voldoet aan A

- U is een **model** van A

**U is een model van Theorie T (verzameling zinnen) als elke zin in T waar is in U**

## Waarheid van een Zin --> BELANGRIJK!

Definitie van de waarheid van een zin --> Inductieve definitie

"We definiëren U |= A door middel van inductie op het aantal symbolen in A"

- Bewijzen door inductieproces
  - Vertrekkend van een lege verzameling
  - Zie formularium

**Waarde is niet altijd bepaald!**

- Indien U geen interpretatie heeft voor de formule A, is de waarde van A onbepaald in U als A vrije symbolen heeft zonder interpretatie in U

**Waarheid van formule in U hangt enkel af van de vrije symbolen van de formule in U**

- Elke expressie (term of formule) heeft dezelfde (waarheids)waarde in alle structuren met identieke domein en identieke waarde voor de vrije symbolen van die expressie

## Interpretatie van n-voudige relatievoorschriften

**Hoe complexere vragen stellen aan databank?** Naast waarheidswaarde van een zin?

- Verzameling-expressies gebruiken!
- $\{(x_1, \ldots, x_n) : A\}$

**Voorbeeld 2.3.4.** Gevraagd: alle koppels (x,y) zodat student x geslaagd is voor vak y.

$$\{(x, y) : (\exists z : Grade(x, y, z) \wedge PassingGrade(z))\}$$

Wat is de interpretatie van zo'n uitdrukking in structuur $\mathfrak{A}$?

**Definitie 2.3.6.** De interpretatie van $\{(x_1, \ldots, x_n) : A\}$ in $\mathfrak{A}$ is de volgende relatie:

$$\{(a_1, ..., a_n) \in {D_{\mathfrak{A}}}^n \mid \mathfrak{A}[x_1 : a_1, \ldots, x_n : a_n] \models A\}$$

Notatie: $\{(x_1, \ldots, x_n) : A\}^{\mathfrak{A}}$.

Deze definitie legt vast welk antwoord de databank moet geven op een query

## Semantisch Afgeleide Concepten

**Logisch Consistent**

- A is logisch consistent als A waar is in minstens één structuur

**Logisch Inconsistent**

- A is logisch inconsistent als A onwaar is in elke structuur die A interpreteert

**Logisch Waar / Tautologie**

- A is logisch waar als A waar is in elke structuur die A interpreteert
- vb: 1+1=2

**Logisch Gevolg**

- A is logisch gevolg van B als voor elke structuur U die beide interpreteert geldt: als U voldoet aan B, dan voldoet U aan A

**Logisch Equivalent**

- A is logisch equivalent met B als A en B dezelfde waarheidswaarde hebben in elke structuur die beide interpreteert

## 2.4 De Informele Semantiek van Predicatenlogica

**Informele Interpretatie I van $\sum$**

- Een vocabularium maken met een logische voorstelling uit het toepassingsdomein waarmee je werkt
- Je stelt de situatie voor in een eigen vocabularium met predicaat- en functiesymbolen

Adhv logische symbolen kan je de betekenis van I opstellen

- Je legt verbanden zoals in je toepassingsdomein adhv logische formules

Abstracte betekenis van zin in structuur

- Je hoeft niet de betekenis te snappen om de zin te gebruiken

## 2.5 Berekenen en Bewijzen van waarheid in structuur

**Propositionele Structuren**

- Gebruik van waarheidstabel om de waarheidswaarde van een zin te vinden

**Structuren met een eindig domein**

- Definitie ontvouwen
- Definitie opvouwen
- Vereenvoudigen van de zin

- Van beneden naar boven werken (Bottom-up)

- Van boven naar beneden werken (Top-down)
    - Vaak moeilijker om te bewijzen

**Structuren met een oneindig domein**

*Zie voorbeelden cursus pagina 60-62*

## 2.6 Construeren van Modellen van een Zin

**Vaak nuttig om model te maken** *(structuur waarin de zin waar is)*

- Geeft inzicht in formules
- Vaak nuttig voor informaticatoepassingen om (een deel van een) model te vinden

*Zie boek pagina 63 voorbeelden*

**Bewijzen dat eigenschap niet geld in alle structuren**

- 1 Tegenvoorbeeld geven volstaat

**<u>Omwisselen van kwantoren bewaart niet (altijd) de equivalentie! Volgorde kwantoren belangrijk!</u>**

## 2.7 Geo-Werelden en Decawerelden

*Zie LogicPalet voor de oefeningen*

## 2.8 Definities

**Definitie van een concept**

- Drukt uit welke objecten behoren tot het concept en welke niet
- Definitie van concept "priemgetal"
  - Getal dat enkel door 1 en zichzelf deelbaar is
  - Elk object dat niet voldoet aan deze definitie is geen priemgetal

- Belangrijke vorm van kennis in wiskunde en wetenschappen

- Eigenschappen definitie
  - Mag geen beperkingen opleggen aan parameterconcepten
    - Een nieuw symbool definiëren mag niets wijzigen aan bestaande symbolen
  - Bepaalt unieke waarde van het gedefinieerde concept uit waarde van parameterconcepten

**Definitie 2.8.1.** Een expliciete definitie $\Delta$ van een predicaat $P/n$ is een logische zin van de vorm:

$$\forall x_1 : \cdots : \forall x_n : P(x_1, \ldots, x_n) \Leftrightarrow A_P[x_1, \ldots, x_n]$$

zodat $P$ niet voorkomt in $A_P$.

**Hierbij mag het predicaat P niet gebruikt worden in de definiërende formule $A_p$**

**Stelling 2.8.1.** *Zij $\Delta$ een expliciete definitie over $\Sigma$ voor predicaat $P$ zoals gegeven in Definitie 2.8.1. Voor elke structuur $\mathfrak{A}$ van $\Sigma \setminus \{P\}$ bestaat exact één $\Sigma$-structuur $\mathfrak{A}'$ die $\mathfrak{A}$ uitbreidt en een model is van $\Delta$, namelijk:*

$$\mathfrak{A}' = \mathfrak{A}[P : (\{(x_1, \ldots, x_n) | A_P\})^{\mathfrak{A}}]$$

**Inductieve Definities**

- Definities waarin inductie wordt gebruikt en het 'te definiëren' zelf voorkomt in de definitie.
- Is moeilijk in de Predikatenlogica

*Zie voorbeeld 2.8.2*

## 2.9 Pragmatiek van Predikatenlogica

**Pragmatiek** = Kunst om natuurlijke taal te interpreteren en om te zetten naar logica

**Omzet logica naar NT (Natuurlijke Taal)**

- Kan gemakkelijk dankzij de informele betekenis I(A) van een zin A

**Omzet NT naar logica**

- Moeilijk
  - Veel interpretatie mogelijk in NT
  - Nadruk op bepaalde woorden verschuift betekenis van zin
  - **Implicaturen** --> Extra gedachten die niet letterlijk worden gezegd maar wel bedoeld worden in een zin

**Mogelijkheid tot Subtiele Fouten in Logica**

- Functiesymbolen stellen totale functies voor
  - Functies in NT zijn meestal partiele functies *(de moeder van, de som van, de partner van, de kleur van, …)*
    - In logica zou je in partiele functie een som kunnen nemen van cursus en student, wat natuurlijk niet kan
  - In logica moeten we logische en totale functies voorstellen
    - Predikatenlogica veronderstelt dat alle n-voudige functiesymbolen f /n : totale functies zijn. Dat de waarde van f in structuur U een totale functie is van $D_U{}^n$ naar $D_U$.

- Kwantificeren over een klasse
  - In logica: Unitaire Kwantificatie
    - We kwantificeren over alle objecten in het domein
  - In NT: Binaire Kwantificatie
    - We kwantificeren over bepaalde klasse van objecten in het domein
  - Let op verschil met "Voor Alle" en "Er Bestaat", ook de implicatie en "en"-symbool verschilt in gebruik!

- Veralgemeende Kwantoren in Natuurlijke Taal
  - Verschil kwantoren in NT en Logica
    - In NT nooit kwantor over hele universum, steeds over een klasse
    - In NT zijn veel meer kwantoren dan de twee logische "voor alle" en "er bestaat"
      - Minstens, niet één, hoogstens, weinig, de meeste, behalve, .....
  - Verschil in betekenis bij lege verzamelingen
    - Let op of een uitspraak ook geldt voor lege verzameling(en)
  - Volgorde kwantoren
    - Bij gelijke kwantoren mag je wisselen, ongelijke niet!

- Implicaturen van Definities
  - Opletten met betekenis van zinnen in NT en logica, niet altijd dezelfde betekenis

- Implicaturen van Namen
  - In NT duiden namen verschillende objecten aan, in logica mag je hier niet van uit gaan en moet je expliciet aangeven dat deze objecten verschillend zijn
  - *vb Jan != Anja*

- Materiele implicatie
  - Synoniem voor implicatiesymbool
  - Is niet altijd de "als, dan" zoals in NT, eerder "niet a OF b"
    - "Als, dan..." is onduidelijk in NT om te interpreteren

## 2.10 Isomorfisme

Als je over eenzelfde vocabularium meerdere structuren hebt door enkel andere namen te geven aan domeinelementen, zijn de functies isomorf.

**Definitie 2.10.1.** Zijn $\mathfrak{A}, \mathfrak{B}$ twee structuren die hetzelfde vocabularium interpreteren. We noemen $b$ een **isomorfisme** van $\mathfrak{A}$ naar $\mathfrak{B}$ indien $b$ een bijectie is van $D_{\mathfrak{A}}$ naar $D_{\mathfrak{B}}$ zodat:

- als $C^{\mathfrak{A}} = d$ dan is $C^{\mathfrak{B}} = b(d)$, voor elke constante symbool $C \in \Sigma$;
- als $F^{\mathfrak{A}}(d_1, \ldots, d_n) = d$ dan is $F^{\mathfrak{B}}(b(d_1), \ldots, b(d_n)) = b(d)$, voor elk functie symbool $F \in \Sigma$;
- $(d_1, \ldots, d_n) \in P^{\mathfrak{A}}$ asa $(b(d_1), \ldots, b(d_n)) \in P^{\mathfrak{B}}$, voor elk predikaat symbool $P \in \Sigma$.

**Definitie 2.10.2.** Een structuur $\mathfrak{A}$ is isomorf met een structuur $\mathfrak{B}$ indien er een isomorfisme van $\mathfrak{A}$ naar $\mathfrak{B}$ bestaat. Notatie: $\mathfrak{A} \cong \mathfrak{B}$.

Andere manier om isomorfisme te definiëren:

**Definitie 2.10.3.** Een functie $b$ is een **isomorfisme** van $\mathfrak{A}$ naar $\mathfrak{B}$ indien $\mathfrak{A}, \mathfrak{B}$ dezelfde symbolen interpreteren, $b$ een bijectie is van $D_{\mathfrak{A}}$ naar $D_{\mathfrak{B}}$, en voor elk geïnterpreteerd symbool $\sigma$ geldt dat $\overline{b}(\tau^{\mathfrak{A}}) = \tau^{\mathfrak{B}}$.

**Stelling 2.10.1.** *In isomorfe structuren zijn dezelfde formules waar. Formeel, indien* $\mathfrak{A} \cong \mathfrak{B}$, *dan geldt voor elke geïnterpreteerd formule* $A$ *dat* $\mathfrak{A} \models A$ *asa* $\mathfrak{B} \models A$.

Bewijs pagina 80

**De isomorfie-relatie ~= is een voorbeeld van een equivalentieverzameling.**

**Definitie 2.10.4.** Een binaire relatie $\sim$ (gebruikt in infix-notatie) in domein $D$ is een equivalentierelatie als het voldoet aan 3 voorwaarden:

- reflexiviteit: $\forall x \in D : x \sim x$
- symmetrie: $\forall x \in D : \forall y \in D : x \sim y \Rightarrow y \sim x$
- transitiviteit: $\forall x, y, z \in D : x \sim y \wedge y \sim z \Rightarrow x \sim z$

De equivalentie-klasse van een element $d \in D$ is de verzameling $\{z \in D | z \sim d\}$. Notatie: $d_{\sim}$.

Equivalentieklassen van een structuur = Verzameling alle structureel gelijke structuren

## 2.11 Informatie-Kwantiteit

Theorie (= verzameling zinnen) heeft in algemeen oneindig veel structuren met verschillende domeinen

- Oplosbaar door eindig domein in te stellen

N aantal structuren en n aantal modellen op structuur

- Hoe groter verhouding N/n, hoe minder modellen verhoudingsgewijs, hoe groter de informatie-kwantiteit van de zin!

Informatie Kwantiteit = log(N/n)

- Zegt niets over betekenis zin, enkel over kwantitatieve maat van informatie

*Nagekeken en vervolledigd op 03-01-2023*

## 3. Logische gevolg en redeneren

### 3.1 Logisch Waar, Logisch Gevolg

**Logisch Consistent**
- A is logisch consistent als A waar is in minstens één structuur

**Logisch Inconsistent / Logisch Tegenstrijdig**
- A is logisch inconsistent als A onwaar is in elke structuur die A interpreteert

**Logisch Waar / Tautologie**
- A is logisch waar als A waar is in elke structuur die A interpreteert
- vb: 1+1=2
- |= A

**Logisch Gevolg**
- A is logisch gevolg van B als voor elke structuur U die beide interpreteert geldt: als U voldoet aan B, dan voldoet U aan A
- B |= A

**Betekenis van |= ?**
- U |= A
  - Formule A is waar in structuur U
- |= A
  - Formule A is logisch waar
- B |= A
  - Formule A is logisch gevolg van formule B

**Relaties tussen 2 zinnen**

**Propositie 3.1.2.**

*(a)* $B \models A$ *asa* $\models B \Rightarrow A$. *In woorden,* $A$ *is een logisch gevolg van* $B$ *asa de zin* $B \Rightarrow A$ *logisch waar is.*

*(b)* $A$ *en* $B$ *zijn logisch equivalent asa* $A \Leftrightarrow B$ *logisch waar is.*

**Definitie 3.1.2.**

- Een zin $A$ is **logisch inconsistent** of **logisch tegenstrijdig** of **logisch contradictorisch met** $B$ indien $A$ onwaar is in elk model van $B$ dat ook $A$ interpreteert.
- Een zin $A$ is **logisch consistent met** zin $B$ indien $A$ voldaan is in minstens één model van $B$.
- Een zin $A$ is **logisch equivalent met** zin $B$ indien voor elke $\mathfrak{A}$ die beide zinnen interpreteert, $A^{\mathfrak{A}} = B^{\mathfrak{A}}$ (ze hebben dezelfde waarheidswaarde, beide zijn waar of beide zijn onwaar).

A ≡ B --> A is logisch equivalent met B

**Omdat een theorie een conjunctie is van zinnen, kan je bovenstaande eigenschappen ook uitbreiden naar theorieën**

## 3.2 Propositielogica

**Propositielogica** = Fragment van de predicatenlogica beperkt tot propositionele symbolen

**Propositioneel Symbool** = 0-voudig predicaatsymbool / relatiesymbool

- Stelt een 0-voudige relatie voor --> Een verzameling van 0-voudig koppels
    - {()} --> Gezien als "True"/t
    - Ø --> Gezien als "False"/f

  Samen vormen ze de booleaanse verzameling B = {t,f}

**Propositioneel Vocabularium** = Vocabularium enkel bestaande uit propositionele symbolen

**Propositionele Formule** = Formule opgebouwd uit propositionele symbolen en de logische connectieven
- Bevat geen kwantoren of =
- Is dus eigenlijk een logische zin

**Definitie 3.2.2.** Een structuur van een propositioneel vocabularium $\Sigma$ is een booleaanse functie van $\Sigma$ naar $\mathbb{B} = \{\mathbf{t}, \mathbf{f}\}$. Voor $P \in \Sigma$ zullen we $\mathfrak{A}(P)$ blijven noteren als $P^{\mathfrak{A}}$.

**Vergelijking SOCS**
- Propositionele logica vergelijken met transitoren en een koppeling van meerdere poorten om een resultaat te krijgen.

**Waarheid van formule berekenen?**
- Waarheidstabel opstellen *(voor kleinere formules)*
- SAT-systemen *(voor grotere formules, gebruikt in LogicPalet)*

## 3.3 Wetten van het denken van Propositionele logica

Wat precies betekenen NT zinnen in de logica? Wat zijn de wetten van het denken?

**Wetten van de connectieven** *(Formularium)*
- Commutativiteit van V en Λ *(P v Q) <=> (Q v P)*
- Associativiteit van V en Λ *(P v Q) v R <=> (Q v R) v P*
- Idempotentie *(P v P) <=> P*

***Bewijzen adhv waarheidstabel***

|= A --> A is tautologie/Logisch Waar
|/= A --> A is niet logisch Waar
A ≡ B --> A is logisch equivalent met B

- Distributiviteit van Λ en V
- Basiswetten negatie
- Betekenis van => en <=>
    - Transitiviteit
    - Commutativiteit <=>

- Contrapositie

- Bewijsprincipes
    - Vormen schema over hoe je iets moet bewijzen

**Normaalvorm** = Verwachte vorm voor invoerformule van logische tools

**Definitie 3.3.1.**

Een **literal** is een formule van de vorm $P(t_1, \ldots, t_n)$ of $\neg P(t_1, \ldots, t_n)$.

Als $L_1, \ldots, L_n$ literals zijn, dan noemen we een formule $L_1 \wedge \cdots \wedge L_n$ een simpele conjunctie en $L_1 \vee \cdots \vee L_n$ een simpele disjunctie. Een simpele disjunctie wordt ook een **clause** genoemd.

Een formule is in **disjunctieve normaal vorm** (DNF) indien het een disjunctie is van simpele conjuncties.

Een formule is in **conjunctieve normaal vorm** (CNF) indien het een conjunctie is van simpele disjuncties. Het is dus een conjunctie van clauses.

Een formule is in **negatie-normaalvorm** als $\wedge, \vee$ en $\neg$ de enige connectieven zijn en als het negatiesymbool $\neg$ enkel in literals voorkomt.

*Zie cursus omzetting formule naar normaalvorm*

## 3.4 Wetten van het denken in de Predikatenlogica

**Generaliseren van logische waarheden**

- Als een zin logisch waar is, kan je de elementen vervangen en blijft het logisch waar?

  vb: ***(A v A) <=> A*** Als je hierin A vervangt door (B V C) krijg je ***((B V C) v (B V C)) <=> (B V C)*** Ook dat is logisch waar

**Propositie 3.4.1** (Generalisatiepropositie). *Zij $A$ een logisch ware propositionele zin. Kies voor elk symbool $P$ in $A$ een logische formule $B_P$ en beschouw de formule $A'$ bekomen uit $A$ door elk symbool $P$ te vervangen door $B_P$. Dan geldt dat $A'$ logisch waar is.*

**Hernoeming van gebonden variabelen**

- Variabelen aan kwantoren mag je hernoemen waarna dezelfde eigenschappen gelden
- De hernoeming is veilig als de nieuwe variabele-naam niet vrij of gebonden voorkomt in de formule

  *Zie formularium*

**Wetten van de kwantificatie**

*Zie formularium*

**Normaalvormen van predicatenlogica**

**Definitie 3.4.3.** Een formule $A$ is in **prenex-normaalvorm** als geen enkele kwantor voorkomt in een conjunctie, disjunctie, implicatie of equivalentie. Dus als elke kwantor boven/buiten elk connectief staat.

<u>Elke formule kan naar Prenex-normaalvorm gebracht worden</u>

- Door toepassen transformatiestappen op de formules
- Vrije symbolen NOOIT van naam veranderen!

Bew. Elke formule kan in prenex-normaalvorm gebracht worden met equivalentie bewarende transformatiestappen. Eerst vertalen we $\Rightarrow$ en $\Leftrightarrow$ in $\wedge, \vee$ en $\neg$. Vervolgens worden de kwantoren naar buiten geschoven. Daarvoor gebruiken we de volgende transformaties :

- $\neg\forall x\colon A$ wordt $\exists x\colon \neg A$ (Eig. 3.4.3)
- $\neg\exists x\colon A$ wordt $\forall x\colon \neg A$ (Eig. 3.4.3).
- $A \wedge \exists x\colon B[x]$ wordt $\exists y\colon A \wedge B[y]$
  Hierbij is $y$ een nieuwe variabele die niet voorkomt in de formule. We hernoemen $x$ door $y$ (Prop. 3.4.4) en schuiven vervolgens $\exists y\colon$ naar buiten (Eig. 3.4.4(3iii) is van toepassing aangezien $y$ niet voorkomt in $A$). Deze methode is van toepassing voor alle formules hieronder.
  Merk op dat als $x$ niet voorkomt in $A$, hernoeming overbodig is.
- $\exists x\colon A[x] \wedge B$ wordt $\exists y\colon A[y] \wedge B$
- $A \vee \exists x\colon B[x]$ wordt $\exists y\colon A \vee B[y]$
- $(\exists x\colon A[x]) \vee B$ wordt $\exists y\colon A[y] \vee B$
- $A \wedge \forall x\colon B[x]$ wordt $\forall y\colon A \wedge B[y]$
- $(\forall x\colon A[x]) \wedge B$ wordt $\forall y\colon A[y] \wedge B$
- $A \vee \forall x\colon B[x]$ wordt $\forall y\colon A \vee B[y]$
- $(\forall x\colon A[x]) \vee B$ wordt $\forall y\colon A[y] \vee B$.

Door herhaalde toepassing van deze equivalentie bewarende omzettingen toe te passen, kunnen we al de kwantoren naar voren schuiven. Q.E.D.

**Definitie 3.4.4.**

Een logische zin is in **prenex-conjunctieve normaalvorm** (pre-CNF) als het in prenex-normaalvorm is, en de kwantorvrije deelformule is in conjunctieve normaalvorm.

Een logische zin is in **prenex-disjunctieve normaalvorm** (pre-DNF) als het in prenex-normaalvorm is, en de kwantorvrije deelformule is in disjunctieve normaalvorm.

## 3.5 Een formele bewijsmethode voor predicatenlogica

In propositielogica --> Waarheidstabel waarin elke mogelijke structuur wordt voorgesteld

**Dit kan niet in predicatenlogica**

- Er zijn oneindig veel structuren in predicatenlogica mogelijk

Oplossing voor predicatenlogica = KE-BEWIJS

**KE-Bewijzen**

- Klassieke Eliminatie-bewijzen
- Om inconsistentie te bewijzen van een formule
- Vertrekt van een lege boom door uitbreidingsregels
  - Nieuwe zin toevoegen onderaan tak afkomstig uit T of door propagatie-regels op zinnen uit die tak
  - Gevalsonderscheid maken voor een zin A en niet A en dus de tak op te splitsen

**Definitie 3.5.1.** Een KE-redenering voor theorie $T$ wordt gedefinieerd door inductie:

- De lege KE-redenering is een KE redenering (met 1 lege tak).
- Gegeven een KE-redenering voor $T$ en een uitbreidingsregel die van toepassing is op een tak ervan, dan is het resultaat van toepassing van die uitbreidingsregel op die tak een KE-redenering.

**Uitbreidingsregels van KE-Redeneringen**

- Hypothesen-regel
  - Van toepassing op elke tak in de redenering
  - In begin toepassen om theorie uit te breiden

- Propositionele propagatie-regels
  - Logisch gevolg toevoegen onderaan een tak van bepaalde formules in die tak
  - Staan in formularium

- ERule
  - Een nieuwe constante invoegen die nog niet voorkomt in de tak

- ARule
  - Een nieuwe of bestaande constante invoegen in de tak
  - Vaak nuttig om bestaande constante te gebruiken

- Gelijkheidsregels
  - Onderaan de tak gelijkheid t=t toevoegen voor willekeurige term t

- Gevalsonderscheiding-regel
  - Tak opsplitsen en aan één zijde A toevoegen, aan de andere 'niet A'

**Tak is gesloten**

- Als er een contradictie in de tak voorkomt

**KE-Redenering is een Bewijs**

- Als elke tak in de redenering gesloten is

**Correctheid en Volledigheid**

**Propositie 3.5.1.** *Als een logische theorie $T$ een model $\mathfrak{A}$ heeft, dan geldt voor elke KE-redenering dat er minstens 1 tak bestaat en een uitbreiding $\mathfrak{A}_1$ van $\mathfrak{A}$ die alle nieuwe constanten $c$ in die tak interpreteert, zodat alle zinnen van de tak waar zijn in $\mathfrak{A}_1$.*

★ *Bewijs pagina 121*

**Strategie voor KE-Bewijzen**

- Strategieregels --> Regels die nuttig kunnen zijn om uit te voeren in het bewijs
  - Het is niet nuttig om dezelfde regel opnieuw te herhalen, ….
  - Het is een beperking opgelegd aan de toepassing van uitbreidingsregels

- Strategie
  - Verzameling van strategieregels om tot het bewijs te komen
  - Strategie is volledig als
    - Elke KE-Redenering voor een inconsistente theorie T kan uitgebreid worden tot een KE-Bewijs van inconsistentie van T met die strategie

**Bewijzen van consistentie met KE-Redeneringen**

**Volledige tak** = Tak die niet gesloten is, maar waarop geen uitbreidingsregels toegepast mag worden door de volledige strategie

**Stelling 3.5.3.** *Als een KE-redenering uit theorie $T$ werd gebouwd met een volledige strategie $S$ en deze redenering heeft een volledige maar niet gesloten tak volgens strategie $S$, dan is $T$ consistent.*

Bew. Als $T$ inconsistent zou zijn, dan zou volgens de definitie van volledige strategie, deze KE-redenering een uitbreiding moeten hebben tot een KE-bewijs van inconsistentie. Maar zo'n uitbreiding bestaat niet omwille van de volledige maar niet gesloten tak. Q.E.D.

**Inferentie** = Nieuwe info afleiden uit oudere bestaande informatie

**Inferentie-Probleem**
- Input: Logische formule en structuur
- Output: Waarheidswaarde logische zin in de structuur: waar/onwaar/onbepaald

**Deductieve inferentie**
- Nieuwe redeneermethodes uit hoofdstuk 3
- Synoniem voor deductie en deductief redeneren

Bereken of input $T$ inconsistent is:
Input: theorie $T$
Output: **t** als $T$ inconsistent is, anders **f**.

Bereken of $T \models A$:
Input: $T, A$
Output: **t** als $T \models A$, anders **f**.

Bereken of $A$ inconsistent is met theorie $T$:
Input: $T, A$
Output: **t** als $A$ inconsistent is met theorie $T$, anders **f**.

***Elk van deze taken kan herleid worden tot elkaar!***

## 3.6 Samenvatting

/

*Nagekeken en vervolledigd op 03-01-2023*

## 4: Modelleren en redeneren in informatica-toepassingen

***Hoe passen we logica toe in informatica? Welk nut heeft het en hoe wordt het gebruikt?***

- Beschrijving hoe de verschillende input omgezet worden in logische input
- Benoemen van de inferentie en definitie van de vorm van inferentie
- Uitleg hoe uit het antwoord van de inferentie-tool het antwoord op probleem gehaald kan worden

### 4.1 Modelleren in logica: Inleiding

Telkens eerst de werkelijkheid modelleren in een theorie
- Kiezen van vocabularium om 'objecten' of 'concepten' voor te stellen van het toepassingsgebied
- Uitdrukken van aantal informele eigenschappen over het toepassingsgebied in formele zinnen over het vocabularium

**Ontologie** = Vocabularium met gekende betekenis van symbolen in het probleemdomein

**Modelleren**
- Eigenschappen van toepassingsdomein omzetten naar formele zinnen in de structuur

- Zinnen moeten voldoen aan eigenschappen
  - Zin moet waar zijn in alle mogelijke toestanden van de wereld
  - De zin moet onwaar zijn in minstens 1 model van de huidige theorie die we als onmogelijk beschouwen (de zin sluit dus minstens 1 onmogelijke structuur uit en is geen tautologie)

### 4.2 Oplossen van informaticaproblemen m.b.v. inferentie

**Definitie 4.2.1.** Een inferentie-probleem is een probleem met een input en een gewenste output. De input bestaat uit één of meerdere logische objecten. Hiermee bedoelen we

- een symbool of vocabularium
- een waarde uit een domein
- een structuur: een toekenning van waardes aan symbolen
- een logische uitdrukking: een term, een verzamelingenuitdrukking, een formule of zin, een theorie.

De output bestaat uit één of meerdere logische objecten die aan een bepaalde logische voorwaarde in termen van de invoer voldoen.

***Zie voorbeelden van problemen op pagina 131***

### 4.3 Databanken Revisited

Na het maken van een vocabularium zijn er verschillende modelleertaken
- Een databank bevat data en is een modellering van een specifieke toestand van het toepassingsdomein
- Modelleren van integriteitsbeperkingen
  - Eigenschappen die geldig zijn in alle mogelijke toestanden van het toepassingsdomein (Verantwoordelijkheid databankmanager)

- Modelleren van query's
  - Relatievoorschrift dat precies de gezochte eigenschap of relatie van de gebruiker moet specifiëren (Verantwoordelijkheid voor opstellen ligt bij gebruiker van databank)

### 4.3.1 Databank als logische theorie

Tegenstrijdigheid in uitgangspunten?

- Logica als dé manier om informatie formeel uit te drukken
- Databank vol met informatie is een structuur, geen theorie

**Waarom kiezen we voor de databank een structuur ipv een theorie?**

Waarom zouden we die vraag stellen?

- Nuttig om te kijken welke info in databank zit
- Nuttig om te leren hoe info wordt voorgesteld
- Nuttig om te begrijpen wat beperkingen van databank zijn

- Een theorie is niet gelijk aan een structuur! Stelt vaak veel minder info voor.
  - In structuur zijn verschillende strings die verwijzen naar verschillende elementen van het domein, dat is niet uitgedrukt in de theorie
  - In theorie niet uitgedrukt dat er geen andere objecten in universum zijn dan degene die overeenkomen met het databankdomein
  - In theorie werd enkel aangegeven welke koppels wel in tabel zitten, niet welke koppels er niet in zitten

**Groot en belangrijk probleem in logica: Mensen beschikken over allerlei impliciete kennis die zo evident is voor ons dat we er niet eens aan denken om deze kennis te expliciteren. Logica weet niets, en heeft die impliciete kennis niet. ==> Theorie onvolledig!**

**Stel S = {$C_1, \ldots, C_n$} een willekeurige niet lege verzameling van constanten**

**Unique Name Axioma - UNA**

**Definitie 4.3.1.** De **Unique Name Axioma's** voor $S$, notatie $UNA(S)$, is de verzameling van alle formules $\neg C_i = C_j$ voor elk paar van verschillende constanten $C_i, C_j \in S$.

**Domain Closure Axioma - DCA**

**Definitie 4.3.2.** Het Domain Closure Axioma voor $S$, notatie $DCA(S)$, is de zin:

$$\forall x\colon (x = C_1 \vee \cdots \vee x = C_n)$$

**Propositie 4.3.3.** *Een structuur $\mathfrak{A}$ is een model van $UNA(S) \wedge DCA(S)$ asa een bijectie $b : S \to D_{\mathfrak{A}}$ bestaat zodat voor alle $C \in S$, $b(C) = C^{\mathfrak{A}}$. Voor elk paar van modellen $\mathfrak{A}, \mathfrak{B}$ van deze theorie bestaat er een bijectie $b : D_{\mathfrak{A}} \to D_{\mathfrak{B}}$ zodat voor elke $C \in C$, voor elke $d \in D_{\mathfrak{A}}$ geldt dat als $C^{\mathfrak{A}} = d$ dan is $C^{\mathfrak{B}} = b(d)$.*

**Definitie van Predicaatsymbool**

**Definitie 4.3.3.** Zij $P/k \in \Sigma$ een predikaatsymbool zodat $P^{\mathfrak{A}}$ bestaat uit $m$ koppels $(C_{1,1}, \ldots, C_{1,k}), \ldots, (C_{m,1}, \ldots, C_{m,k})$. De *definitie* van $P$, notatie $Def(P)$, is de logische zin:

$$\begin{aligned}\forall x_1\colon \ldots \forall x_k\colon (P(x_1, \ldots, x_k) \Leftrightarrow\ & (x_1 = C_{1,1} \wedge \cdots \wedge x_k = C_{1,k}) \\ & \vee \cdots \vee \\ & (x_1 = C_{m,1} \wedge \cdots \wedge x_k = C_{m,k}))\end{aligned}$$

**Voorbeeld 4.3.2.** $Def(Prerequ)$ is:

$$\begin{aligned}\forall x\colon \forall y\colon [Prerequ(x, y) \Leftrightarrow\ & (x = CS148 \wedge y = CS230) \vee \\ & (x = CS230 \wedge y = CS238) \vee \\ & (x = M100 \wedge y = M200)]\end{aligned}$$

### Definitie van Theorie van Structuur U, Theo(U) *(die wel volledig overeenkomt met de structuur)*

Theorie van Structuur U, Theo(U), is de verzameling van logische zinnen:
- UNA(S)
- DCA(S)
- Def(P) voor elk predikaatsymbool van structuur U

Theo(U) is de manier om informatie uit de databank te modelleren in logica.

Bevat deze theorie alle informatie uit U?
- JA, kunnen bewijzen door aan te tonen dat de structuur en alle isomorfe structuren ervan de enige modellen zijn van Theo(U)

> **Stelling 4.3.1.** *Zij gegeven een databankstructuur $\mathfrak{A}$ met vocabularium $\Sigma$.*
> *(a) Een structuur $\mathfrak{B}$ van $\Sigma$ is een model van $Theo(\mathfrak{A})$ asa $\mathfrak{B}$ en $\mathfrak{A}$ isomorf zijn.*
> *(b) Voor elke logische zin $A$ van $\Sigma$ geldt dat $Theo(\mathfrak{A}) \models A$ asa $\mathfrak{A} \models A$.*

A is een logisch gevolg van de theorie Theo(U) asa A waar is in de structuur U.
Dat betekent dat - in principe - een theoremprover uit Theo(U) dezelfde query's kan beantwoorden als een databanksysteem uit U.
Het betekent ook dat we een databanksysteem kunnen beschouwen als een super efficiënte theoremprover voor de theorie Theo(U)

***Bewijs op pagina 137***

Eigenschappen Theo(U)
- Is consistent, want heeft al zeker 1 model U
- Theorie weet of A waar is, of dat A onwaar is

**Volledige versus onvolledige informatie**
- Mogelijks hebben we (nog) niet alle gegevens in onze databank
- We willen er wel vanuit gaan dat er nog andere objecten kunnen bestaan buiten onze databank

Onvolledige info kan niet beschreven worden in een structuur, wel in een theorie.

Hierdoor weten we nooit of zinnen waar zijn of niet, oplossing?
- NULL-waardes
  - Betekent dat de waarde op die plaats niet gekend is
  - Is GEEN logische constante, want kan meerdere keren voorkomen en een ander object voorstellen
  - Maakt antwoorden op query's veel complexer, omdat er niet altijd een antwoord geweten is *("mogelijks" kan nu ook)*

    Hoe theorie met NULL-waardes voorstellen in een structuur?
    - Voor elke NULL een nieuw logische constante toevoegen aan het vocabularium
      *(Zie pagina 139 voorbeelden)*

- Open Domeinen

  Open Domein Assumptie
  - Databank legt in werkelijkheid zelf geen domein vast. Indien je een onbestaand domeinelement oproept in een relatie zal de query 'onwaar' teruggeven omdat die het element niet vind in de tabel.
    Toch nam de databank aan dat het een element was en ging er naar op zoek. De databank zal dus nooit teruggeven "element niet in domein".

★ **Verder uitzoeken, nog onduidelijk (pagina 140)**

### 4.3.2 Integriteitsbeperkingen

**Integriteitsbeperkingen IB** = Eigenschappen die voldaan moeten worden om een zinvolle stand van zaken voor te stellen

Databank is zinvol indien ze voldoet aan de integriteitsbeperkingen IB
- Kunnen we controleren met predicatenlogica

**Revisie-Inferentie**
- Databankstructuur die niet voldoet aan IB bevat fouten

### 4.3.3 Queries modelleren en oplossen

**Databank** = Abstracte voorstelling van huidige stavaza van het toepassingsdomein

Formuleren van query op databank
- Modelleerstap
  - Formule of verzamelingsuitdrukking $\{(x_1, \ldots, x_n) : A\}$
  - Expressie die bedoelde vraag goed omschrijven

- Databanksysteem opgeroepen
  - Formule oproepen in databank
    - Bereken of A voldaan is in U

> Bereken of $A$ voldaan is in een structuur $\mathfrak{A}$ met eindig domein:
> Input: $\mathfrak{A}, A$
> Output: $\mathbf{t}$ indien $\mathfrak{A} \models A$, anders $\mathbf{f}$.

    - Bereken de relatie voorgesteld door voorschrift in U

> Bereken de relatie voorgesteld door een relatievoorschrift in structuur $\mathfrak{A}$ met eindig domein:
> Input: $\mathfrak{A}, \{(x_1, \ldots, x_n) : A\}$
> Output: $\{(x_1, \ldots, x_n) : A\}^{\mathfrak{A}}$.

**Correcte vertaling Nederlands <-> logica?**
- Als de informele semantiek van de query dezelfde gedachte heeft als de Nederlandse query
- Garandeert dat antwoord op query correct is in elke toestand
- Antwoord correct in elke zinvolle databank

> **Definitie 4.3.6.** We zeggen dat twee formules $A$, $B$ equivalent zijn volgens $IB$ indien $IB \models A \Leftrightarrow B$.

**Expliciet en impliciet getypeerde query's**

Queries en natuurlijke taal = Steeds kwantificeren over een deelklasse / deeltype

> **Definitie 4.3.7.** Een *expliciet getypeerde logische formule* is van de vorm waarin elke gekwantificeerde deelformule eruit ziet als $\exists x\colon (T(x) \wedge A)$ of $\forall x\colon (T(x) \Rightarrow A)$. We noemen $T$ het type van de variabele $x$. Een *expliciet getypeerde n-voudige query* is een relatievoorschrift van de vorm $\{(x_1, \ldots, x_n) : T_1(x_1) \wedge \cdots \wedge T_n(x_n) \wedge A\}$, met $A$ een expliciet getypeerde formule.

Soms moeten we types toevoegen en soms niet
- Wanneer wel en wanneer niet?

**<u>Als je bijvoorbeeld Som(a,b) neemt, moet je dan nog vermelden dat getal(a) en getal(b)? Dat is de vraag!</u>**

**Het Type-insertie algoritme**
- Algoritme waarmee getest wordt of een formule al dan niet correct is

  *Zie algoritme in cursus pagina 145*

**Voorbeeld 4.3.8.** De query

$$\{x : \neg Prerequ(x, x)\}$$

wordt getransformeerd door type-insertie in

$$\{x : \neg(Course(x) \wedge Prerequ(x, x))\}$$

Hierin is $x$ niet expliciet getypeerd. Deze formule is zeker niet equivalent met de expliciet getypeerde query:

$$\{x : Course(x) \wedge \neg Prerequ(x, x)\}.$$

Om de query te corrigeren moet $x$ expliciet getypeerd worden.

**Definitie 4.3.8.** Een **impliciet getypeerde formule** is er een die equivalent is met zijn expliciet getypeerde formule volgens de type-integriteitsbeperkingen van $IB$.

★ **Wat zijn expliciet en impliciet gedefinieerde definities?**

- Expliciete logische formule
  - Elke gekwantificeerde deelformule in de vorm van ∃x: (T(x) ∧ A) of ∀x: (T(x) ⇒ A)
  - T is het type van variabele x

- Expliciete n-voudige query
  - Relatievoorschrift van de vorm $\{(x_1, \ldots, x_n) : T_1(x_1) \wedge \cdots \wedge T_n(x_n) \wedge A\}$
  - A een expliciet getypeerde formule

- Impliciet getypeerde formule
  - Formule equivalent met zijn expliciet getypeerde formule volgens de type-integriteitsbeperkingen van IB

### 4.3.4 Definities

**Views** = Definities van afgeleide concepten m.b.t. de databankstructuren
- Kunnen gebruikt worden als concepten in query's
- Creëren van nieuwe concepten in termen van bestaande

**Definities in predicatenlogica**
- Expliciete definities

  $\forall x_1 : \ldots \forall x_n : P(x_1, \ldots, x_n) \Leftrightarrow A_P$ *waarbij $A_P$ een formule is die het symbool P niet bevat*

  vb: ∀x: ∀v : [GeslaagdVoor(x, v) ⇔ ∃s: Grade(x, v, s) ∧ PassingGrade(s))]

  - **Hier mag gedefinieerde symbool NIET rechts in de definiërende formule of condities van de inductieve regels voorkomen**

- Inductieve Definities
  - Veel inductieve/recursieve definities in cursus
    - Definitie van een term, van een formule, van waarheidsrelatie, van interpretatie in structuur, ….
  - Uitleg in 1.7
  - **Hier mag gedefinieerde symbool ook rechts in de definiërende formule of condities van de inductieve regels voorkomen**

★ *Zie pagina 148 onderaan, verschil tussen definities en stel van implicaties, **OEFENING DAAROP KUNNEN P149**!*

Implicaties
- Als conclusie van implicatie voldaan is, is heel de implicatie voldaan

Definitie
- Elk element moet door minstens 1 regel geproduceerd zijn

### 4.3.5 SQL

*Niet voor examen*

## 4.4 De oudste logische theorie: de Peano Axioma's

**Peano**
- 19-eeuwse logicus en wiskundige
- Publiceerde studie over natuurlijke getallen

**Overladen Symbool**
- Meerdere betekenissen
- Beter vermijden
- Context nodig om juiste betekenis te weten

**Peano's Theorie**
- Wilde structuur van natuurlijke getallen axiomatiseren
- Theorie maken waarvan alle modellen isomorf zijn met de oorspronkelijke structuur

  *$S^n(0)$ is geen term, wel afkorting voor S(S(…S(0)…)) en staat voor "n keer S"*
  *S is hier een functiesymbool in de structuur*

- Theo(DatabankNatuurlijkeGetallen)

  - UNA *(Unique Name Axioma)*
    - *Elk paar van verschillende termen hebben verschillende waardes*
    - ¬($S^n$(0) = $S^m$(0)), en dit voor elk paar n != m.

      ***Peano stelde dit voor door:***

      ∀x: ¬(0 = S(x)) *(Peano1)*
      ∀x: ∀y : (S(x) = S(y) ⇒ x = y) *(Peano2)*

  - DCA *(Domain Closure Axioma)*
    - *Elk object in universum is waarde van minstens één van de termen in de structuur*
    - ∀x: (x = 0 ∨ x = S(0) ∨ x = S(S(0)) ∨ . . .)
      ***Dit is geen logische formule, er is een oneindigheid***

    - DCA kan niet in de predicatenlogica worden uitgedrukt
      - Peano vond wel een manier in de tweede-orde predicatenlogica
      - Uitbreiding dat je ook over verzamelingen kan kwantificeren
      - ★ ∀P : P(0) ∧ (∀x: P(x) ⇒ P(S(x)) ⇒ ∀x: P(x) *(Peano3)*

      Elke verzameling P die 0 bevat en als het n bevat dan ook n+1, bevat alle elementen van het domein.
      Dit noemt men het **<u>inductie-axioma</u>** *(Peano 3)*

  - Expliciete Definitie-Formules voor elk predikaatsymbool in structuur
    - Definities van + en x

      $\forall x\colon (0 + x = x)$ $(\text{Peano}_4)$
      $\forall x\colon \forall y\colon (S(x) + y = S(x + y))$ $(\text{Peano}_5)$
      $\forall x\colon (0 \times x = 0)$ $(\text{Peano}_6)$
      $\forall x\colon \forall y\colon (S(x) \times y = y + (x \times y))$ $(\text{Peano}_7)$

Peano's theorie is correct, maar door de inductie-axioma is het geen predicatenlogica

**Inductieschema** --> Benadering van het inductie-axioma in predicatenlogica

**Definitie 4.4.1.** Het **inductieschema** voor een vocabularium $\Sigma$ dat minstens $0, S/1$ bevat, bestaat uit alle zinnen van de predikatenlogicataal van $\Sigma$ van de volgende vorm:

$$\forall y_1 \colon \ldots \forall y_n \colon (\varphi[0] \wedge \forall x \colon (\varphi[x] \Rightarrow \varphi[S(x)])) \Rightarrow \forall x \colon \varphi[x])$$

waarbij $\varphi[x]$ een willekeurige formule is van predikatenlogica met vrije constante-symbolen $x$ en $y_1, \ldots, y_n$ die niet voorkomen in $\Sigma$.

**Peano-Rekenkunde**

- Oneindige theorie verkregen door in de Peano-Axioma's het inductie-axioma te vervangen door inductieschema
- Is logisch zwakker dan gewone axioma's
    - Heeft ook niet-standaard modellen, modellen die niet isomorf zijn met de structuur

***Ook zijn er eigenschappen die wel voldaan zijn in de structuur N, maar niet in Peano-Rekenkunde geïmpliceerd worden***

**Rol Peano in Informatica**

- Vele complexe problemen redenering over getallen
- Correctheidsbewijzen programma's

## 4.5 Zoekproblemen en Modelgeneratie

Informaticaproblemen, zoeken naar waardes aan bepaalde beperkingen

- Kleuring zoeken in graaf zodat ze langs elkaar steeds verschillend zijn
- Oplossingen voor logische puzzels
- Uurroosterproblemen
- .....

Natuurlijke oplossing?

- Vocabularium opstellen, theorie T opstellen met de beperkingen en een model zoeken van deze theorie

Probleem = Modellen kunnen oneindig zijn

**Definitie 4.5.1.** We noemen $\mathfrak{A}$ een expansie van $\mathfrak{A}_i$ indien $D_{\mathfrak{A}} = D_{\mathfrak{A}_i}$ en voor elk symbool $\sigma \in \Sigma_{\mathfrak{A}_i}$ geldt dat $\sigma^{\mathfrak{A}} = \sigma^{\mathfrak{A}_i}$.

**Constraint Problemen**

- Waardes die de oplossing vormen zijn de in de structuur geïnterpreteerde symbolen van Theorie T
- Wordt onderzocht in Constraint Programming wetenschap
- Zoeken naar modellen van logische zinnen (Theorie)

Vaak ook nood aan een perfecte oplossing

- Een optimalisatie aan gewone oplossingen
  Vb: DHL wil zo snel mogelijk alle pakjes leveren en hiervoor een schema maken

- **Definities van relevante vormen van inferentie kennen**
- **Principe kennen hoe dit type probleem herleid kan worden naar inferentieprobleem**
- **Oefeningen uit oefenzitting kunnen**

## 4.6 Automated Theoremproving (ATP)

**Deductieve Inferentie**
- Kan niet volledig opgelost worden door computers

**ATP-Systeem**
- Ontwikkelt automatisch Theoremprovers
- Methodes die gebruikt worden voor aantonen van
    - Logische inconsistentie
    - Logische waarheid
    - Logisch gevolg

- Hebben al dingen bewezen die de mens niet kon

**Toepassingen Theoremprovers**
- Deductieve Databanken
- Software-Verificatie
- Hardware-Verificatie
- Correctheidsbewijzen
- Programmeertalen gebaseerd op predicatenlogica (Prolog)
- Semantische Web

## 4.7 Formele Methoden in Software Engineering

Vaak meerdere mensen werken aan zelfde project --> Communicatie essentieel

Hoe communiceren? Nederlands is vaak onduidelijk en niet strikt in betekenis!

Kan problemen geven bij systemen waar bugs gevaarlijk zijn
- Vliegtuigsoftware
- Lancering ruimtetuigen
- Medische toepassingen

**Oplossing is een formele taal voor specificaties**

Beschrijving transacties op databank
- Pre-Conditie --> Wanneer mag transactie plaatsvinden?
    - Uitgedrukt in formule van vocabularium

- Post-Conditie --> Wat is het effect op de databank?
    - Beschrijft toestand voor en na het uitvoeren van transactie
    - Kan niet in logische formule naar 2 situaties wijzen
        - Nieuwe symbolen introduceren is oplossing ***(zie pagina 164)***

*Zie voorbeeld Bibsys Databank in cursus pagina 163-165*

**Controleren van correctheid Bibsys databank**
- Waarheidsevaluatie-inferentie
    - Input een structuur en integriteitsbeperking
    - Output true *(als U |= IB)* of false

- Validiteits-checking-inferentie
    - Input een IB, Post- en Preconditie
    - Output true (als $\forall x_1 : \ldots \forall x_n : (IB \wedge Pre_{Trans}[x_1, \ldots, x_n] \wedge Post_{Trans}[x_1, \ldots, x_n] \Rightarrow IB_N )$ of false

- Update-Inferentie
    - Transacties worden verricht door middel van logische inferentie
    - Input Oude databank, post- en preconditie en een transactie
    - Output een structuur, expansie van oude structuur en voldoet aan pre- en postconditie

    - Gebruikt waarheidsevaluatie en modelexpansie

    - Wordt niet echt gebruikt, er is geen programma die zo'n modellen kan maken

## 4.8 Kennisbanksysteem

**Waarom logica bestuderen in informatica?**

- Informatie is de grondstof om informaticaproblemen op te lossen
- Logica onderscheid van programmeertalen door principe:

  **Gebruik van dezelfde theorie om diverse types van problemen binnen hetzelfde toepassingsdomein op te lossen door middel van verschillende vormen van inferentie**

> **Definitie 4.8.1.** Een kennisbanksysteem voor een logica is een systeem dat een theorie $T$ in die logica bevat en verschillende vormen van inferentie op $T$ of delen van $T$ aanbiedt om verschillende problemen op te lossen. De theorie $T$ wordt de kennisbank genoemd.

## 4.9 Conclusie

**Overal in informatica is informatie**

- Welke soort info is beschikbaar?
- Welke formele talen hebben we nodig om de info uit te drukken?
- Welke soorten van redeneren hebben we nodig om problemen met die info op te lossen?

Indien we probleem kunnen herleiden naar inferentie-probleem met bijhorende solver => Geen programma meer nodig

**Query-Probleem**

- Eenvoudig Inferentie-probleem dat sterk is ontwikkelt in generische solvers
- SQL is de formele taal die wordt gebruikt

Veel problemen of inferentie-solvers nog lang niet zo ver ontwikkeld!

*Nagekeken en vervolledigd op 07-01-2023*

# Hoofdstuk 5

zaterdag 7 januari 2023 13:50



## 5. Algoritmes, Berekenbaarheid, Halting-Probleem, Onbeslisbaarheid en Onvolledigheidsstelling Gödel

*Niet kennen voor examen*

*Nagekeken en vervolledigd op 07-01-2023*