Logica voor informatica

Matt ter Steege
mattersteege@gmail.com
Universiteit Utrecht
Utrecht, Nederland

Contents

TERM	LATEX	LATEX COMMAND
1. there exists at least one	$\exists$	`\exists`
2. there exists one and only one	$\exists !$	`\exists !`
3. there is no	$\nexists$	`\nexists`
4. for all	$\forall$	`\forall`
5. not (logical not)	$\neg$	`\neg`
6. or (logical or)	$\lor$	`\lor`
7. division	$\div$	`\div`
8. and (logical and)	$\land$	`\land`
9. implies	$\implies$	`\implies`
10. right implication	$\Rightarrow$	`\Rightarrow`
11. is implied by (only if)	$\Longleftarrow$	`\Longleftarrow`
12. left implication	$\Leftarrow$	`\Leftarrow`
13. if and only if, iff	$\iff$	`\iff`
14. equivalence	$\Leftrightarrow$	`\Leftrightarrow`
15.Subset	$\subset$	`\subset`
16. Logical XOR (exclusive or)	$\oplus$	`\oplus`
17. Union of sets	$\cup$	`\cup`
18. Empty set	$\emptyset$	`\emptyset`
19. Intersection of sets	$\cap$	`\cap`
20. Union of sets	$\cup$	`\cup`

1 Propositielogica

Propositielogica is de studie van bewijzen. Een propositie is te beantwoorden met waar of niet waar Een propositie is $3 < 17$, dit kan je checken op waarheid. Een propositie is $x \implies y$

1.1 Proposities

We gebruiken in logica variabelen zoals P en Q, dit zijn *atomisch* proposities, deze zijn altijd geschreven in HOOFDLETTERS. Deze zijn niet verder op te delen, dus niet opgebouwd van kleinere delen zoals implicaties etc. deze waardes zijn Waar/True/1 of Onwaar/False/0.

Je hebt ook kleine letters p, q, …, dit zijn niet-atomische proposities, het zijn geen propositionele formules, maar eerder *metavariabelen.*

Elke propositionele formule bestaat uit:

- Atomitsche proposities (P, Q, R, …)
- true, (T, Waar, 1)
- false, (F, Onwaar, 0)

Deze kunnen ook kleiner opgedeeld zijn, dan zijn dit ook propositionele formules:

- $P \implies Q$ - **implicatie** als P, dan Q
- $P \wedge Q$ - **conjunctie** P én Q
- $P \vee Q$ - **disjunctie** P of Q
- $\neg P$ - **negatie** (niet P / P houd geen stand)

Zoals in de wiskunde ook is heeft propositielogica óók een volgorde, deze is:

(1) Haakjes ()
(2) Negatie $\neg$
(3) Conjunctie $\wedge$
(4) Disjunctie $\vee$
(5) Implicatie $\implies$

$\neg P \vee Q \implies Q \wedge P$ moet gelezen worden als $((\neg P) \vee Q) \implies (Q \wedge P)$

Om dit te onthouden kan je deze zij onthouden: "Hoe Navigeert Connie De Ijssel"

1.2 Semantiek van propositie operatoren

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \oplus Q$	$P \implies Q$	$P \equiv Q$
0	0	1	0	0	0	1	1
0	1		0	1	1	1	0
1	0	0	0	1	1	0	0
1	1		1	1	0	1	1

Tabel 1: Truthtable van basisoperaties

Maar stel, je wilt iets moeilijks bewijzen zoals $\neg P \vee Q \implies Q \wedge P$, dan kan je dat op de volgende manier doen:

P	Q	$\neg P$	$\neg P \vee Q$	$Q \wedge P$	$\neg P \vee Q \implies Q \wedge P$
0	0	1	1	0	0
0	1	1	1	0	0
1	0	0	0	0	1
1	1	0	1	1	1

Tabel 2: Truthtable van moeilijkere propositie formule

Je kan ook met verschillende kleuren pennen een kleine thruthtable schrijven, maar dit is een hele nette manier om het ook te doen.

1.3 Voorbij thruth tables

Je kan d.m.v een truth table bewijzen of een formule wel of niet houd, maar dat kan ook anders, bijvoorbeeld door het versimpelen van de formules.

	Expression	**Law**
	$(p \implies q) \vee (q \implies p)$	original
$\Leftrightarrow$	$(\neg p \vee q) \vee (\neg q \vee p)$	implication
$\Leftrightarrow$	$\neg p \vee ((q \vee \neg q) \vee p)$	associativity / rearrangement
$\Leftrightarrow$	$\neg p \vee (T \vee p)$	tertium non datur
$\Leftrightarrow$	$\neg p \vee T$	absorbing property of T
$\Leftrightarrow$	T	absorbing property of T

Deze afleiding kan je maken door de regels te gebruiken die hieronder stan geschreven (er zijn er nog meer).

Commutativity:

- $p \wedge q \Leftrightarrow q \wedge p$
- $p \vee q \Leftrightarrow q \vee p$

Associativity:

- $p \wedge (q \wedge r) \Leftrightarrow (p \wedge q) \wedge r$
- $p \vee (q \vee r) \Leftrightarrow (p \vee q) \vee r$

Tertium non datur:

- $p \vee \neg p \Leftrightarrow T$

Idempotence:

- $p \wedge p \Leftrightarrow p$
- $p \vee p \Leftrightarrow p$

De Morgan:

- $\neg(p \vee q) \Leftrightarrow \neg p \wedge \neg q$
- $\neg(p \wedge q) \Leftrightarrow \neg p \vee \neg q$

Double negation (niet niet is wel):

- $p \Leftrightarrow \neg(\neg p)$

Properties of T en F:

- $p \vee F \Leftrightarrow p$
- $p \wedge F \Leftrightarrow F$
- $q \vee T \Leftrightarrow T$
- $q \wedge T \Leftrightarrow q$

Implication:

- $(p \implies q) \Leftrightarrow (\neg p \vee q)$

Contraposition:

- $(p \implies q) \Leftrightarrow (\neg q \implies \neg p)$

2 sets

Propositielogica heeft ook een manier om over een groep dingen te redeneren, ***sets***. Deze schrijf tussen accolades met comma's tussen de items: $\{false, true\}$, $\{3, 7, 14\}$, $\{red, blue, yellow\}$. Dit kan voor kleine sets met de hand, maar als je grote sets wilt schrijven, dan kost dat heel veel tijd, daarom kan je ook: 1, 3, 5, ..., 99 om alle oneven cijfers van 1 t/m 99 als set te hebben. Of als 1, 2, 3, ... om alle cijfers vanaf 1 te hebben (oneindig). De lengte van een sets is op te schrijven door |A| of A, waar A een set is met een lengte, de **cardinality** hier tel je alleen het aantal unieke elementen. Als er maar 1 element in een set zit, dan heet dat een **singleton**.

Ook kan je de ***set-builder notation*** gebruiken, hierbij schrijf je het soort element dat je wilt en wat het element kan zijn: {x : x has the property P}

Voorbeeld **(Set-builder notation)**:

Een collectie van alle prime numbers: {x : x is een priemgetal}
Of {s : s is een strand in Nederland}.

Er zijn ook een paar basis sets, die al van tevoren zijn gedefinieerd:

- $\emptyset = \{\}$ (the empty set)
- $\mathbb{B} = \{0, 1\}$ (the binaire set)
- $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$ (De natuurlijke nummers)
- $\mathbb{Z} = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$ (De integers)
- $\mathbb{Q} = \{\frac{m}{n} : \text{m, n} \in \mathbb{Z}, \text{n} \neq 0\}$ (De rationele nummers)
- $\mathbb{R} = \{x : \text{x is a real number}\}$ (De real nummers)

∅ en { ∅ } zijn twee verschillende dingen, de een is namelijk: { }, terwijl de ander { { } } is!

2.1 leden en gelijkheid

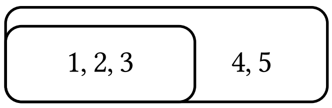
Je kan bekijken of een set een bepaalde member heeft door $7 \in \{3, 7, 14\}$ te doen. Hier is $\in$ te lezen als *"is element onderdeel van"* Je hebt ook de negatie vorm, $\notin$, wat *"is element niet onderdeel van"*, oftwel $6 \notin \{3, 7, 14\}$ betekend.

Ook is goed om te weten dat gelijkheid op basis van unieke element rust, dus $\{1, 2, 3, 2, 1\}\{1, 3, 2\}$. Ongelijkheid is dus te bepalen door te kijken of een van de twee sets een waarde heeft die niet in de ander voorkomt. Dus volgorde maakt niet uit, en hoeveelheid elementen die dan welniet hetzelfde zijn maakt ook niet uit.

2.2 subsets and supersets

Een ***subset*** is een set waarin minstens een deel van de waardes van de superset in voorkomen.

Dus $A = \{1, 2, 3\}$ is een subset van $B = \{1, 2, 3, 4\}$, hierin is B een ***superset*** van A en A een ***subset*** van B Dit schrijf je als volgt: $A \subseteq B$ *(A is included or contained in B)* en vice versa $B \supseteq A$ *(B includes or contains A)*.



2.3 Operaties op sets

Union $A \cup B = \{x : x \in A \vee x \in B\}$ Combineert A en B met elkaar
Intersection $A \cap B = \{x : x \in A \wedge x \in B\}$ Het deel wat zowel in A als in B zit
Complement $\overline{A} = \{x : x \notin A\}$, alles wat niet in A zit.
Difference $A = \{x : x \in A \wedge X \notin B\}$ Alles wat in A zit, maar niet in B

Zo heb je ook de *powerset P(A)*, dit is een set waarin alle mogelijke subsets van A zitten: $P(A) = \{X : X \subseteq A\}$. Deze heeft n^2 elementen, waar n het aantal elementen van set A is.

Voorbeeld **(powerset/matchsset)**:

Stel je hebt een computer scherm, van 1680 x 1050, je wilt elke mogelijke configuratie van zwart/witte pixels opschrijven. Een set geeft aan welke pixels wit zijn, dus je wilt elke set met elke mogelijke manier van pixelcoordinaten, dat kan je zo schrijven:
$W = \{0, 1, \ldots, 1697\}$ voor elke X coordinaat
$H = \{0, 1, \ldots, 1049\}$ Voor elke Y coordinaat
$W \times H = \{\{0, 0\}, \{0, 1\}, \{0, 2\}, \ldots, \{1, 0\}, \ldots\{1679, 1049\}\}$ dit is een volledig wit scherm, elke pixel is gerepresentateerd in deze set
$P(W \times H)$ = elke configuratie van witte pixels

2.4 Partities

Je kan een set A opdelen in meerdere sets, zoals A_1 en A_2, hier zijn de regels: $A_1 \cup A_2 = A$ en $A_1 \cap A_2 = \emptyset$ Zo kan je A = $\mathbb{N}$ opsplitsen in A_1 en A_2 waar A_1 de even getallen zijn en A_2 de oneven getallen zijn.

2.5 Naive set theory

3 Boolean algebra

korte notitie* in Bolean algebra is de + hetzelfde als de OR operator, · hetzelfde als de AND operator, $^{-1}$ hetzelfde als NOT

3.1 monoïde

Monoide is een verzameling A, is niet ledig en heeft minstens 1 element e, en een binaire operator ⊕ Een monoide heeft een nul-element, dit is een waarde die niks doet in de operatie. $x + 0 = x$, hier doet de 0 helemenaal niks, dus 0 is het 0-element in de "+" operatie. $x \times 1 = x$, hier is 1 het nul-element in de monoide × 1; het doet niks in de operatie. het is dus een neutrale waarde.

3.2 Bool's algebra

Boolean algebra heeft een paar regels:

- een set B
- Twee elementen $0 \in B$ en $1 \in B$, genaamd **zero** en **unit**
- Twee binaire operatoren + en · som/OR en product/AND respectievelijk.
- Een Unaire operator $^{-1}$ de inverse genoemd ofwel NOT (kan ook als x' zijn genoteerd).

Voorbeeld (**Bewijs dat $x + x = x$**):

We willen bewijzen dat voor elke $x, x + x = x$

$x + x$	$= (x + x) \cdot 1$	Ident
	$= (x + x) \cdot (x \cdot x^{-1})$	Complement
	$= x + (x \cdot x^{-1})$	Distrobution
	$= x + ()$	Complement
	$= x$	Ident

3.3 dualiteit

Je kan voor elke vergelijking die je opschrijft een soort duale tweede vergelijking maken door hetvoglende te doen:

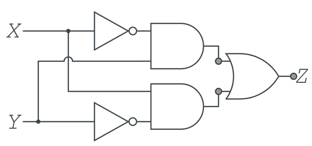
- 0 met 1 vervangen
- 1 met 0 te vervangen
- + met · te vervangen
- · met + te vervangen

3.4 waarheidstabel naar formule/circuit

Stel je krijgt een waarheidstabel zoals deze:

x	y	z
0	0	0
0	1	1
1	0	1
1	1	0

Als je dit omzet naar een propositie, dan moet je kijken naar alle rijen waar de uitkomst 1 is. Voor elke 0 waarde schrijf je de variabele (hier x of y), op als $\overline{x}$ of $\overline{y}$. Elke 1 waarde schrijf je alleen de variabele op, dus x of y. Als je dat hier zou doen dan krijg je hetvolgende antwoord: $(\overline{x} \land y) \lor (x \land \overline{y})$. Dit kan je dan vervolgens omzetten naar een circuit:



4 Predikatenlogica

4.1 Predicaten en vrije variabelen

We zagen eerder de *set-builder*-notatie

$$\{x \mid x \text{ heeft eigenschap } P\},$$

waarbij P een voorwaarde is. Een voorbeeld van zo'n voorwaarde is

$$P(x) = Prime(x) = \text{“}x \text{ is een priemgetal”}.$$

Het predicaat geeft voor elke waarde x waar of onwaar terug, en op basis daarvan bepalen we of x tot de verzameling behoort. De bijbehorende *truth set* bestaat uit alle waarden die voldoen aan het predicaat.

Voorbeeld **(Eendjes)**:

We hebben zeven eendjes:

$$Eendjes = \{Kwik, Kwek, Kwak, Dagobert, Donald, Katrien, Dumbella\}.$$

We definiëren het predicaat

$$Female(x) = \text{“}x \text{ is vrouwelijk”}.$$

- $Female(Katrien) \wedge Female(Dumbella)$ is waar.
- $Female(Kwik) \wedge Female(Kwek) \wedge \cdots \wedge Female(Donald)$ is niet waar.
- De truth set van $Female(x)$ is dus $\{Katrien, Dumbella\}$.

4.2 Kwantificatoren en gebonden variabelen

Stel dat we willen controleren of elke waarde in een lijst aan een voorwaarde voldoet. Het boek geeft het volgende voorbeeld:

Er zijn vier kinderen:

$$Kinderen = \{Joel, Felix, Oskar, Amanda\}.$$

Eén van hen moet voorin op de passagiersstoel zitten, omdat er achterin maar drie plekken zijn. Het predicaat $Front(x)$ betekent dat kind x voorin zit. We weten dus zeker dat voor één waarde van x het predicaat waar moet zijn.

De samengestelde propositie

$$Front(Joel) \vee Front(Felix) \vee Front(Oskar) \vee Front(Amanda)$$

moet waar zijn. Dat levert in principe een grote lijst aan mogelijke combinaties op, wat snel onoverzichtelijk wordt.

Als je een predikaat schrijft zoals $P(x) = \ldots x \ldots$, dan is x een gebonden variabele. Als je een predikaat schrijft zoals $P(x) = \ldots y \ldots$, dan is y een vrije variabele.
Bij gebonden variabelen kan je de x vervangen door een waarde, $P(x) = x > 1337$. $P(10000)$ wordt dan $10000 > 1337$

4.3 Universele kwantificator

Om dit compacter te schrijven gebruiken we de universele kwantor $\forall$. De notatie $\forall x\, P(x)$ betekent: "voor elke x geldt $P(x)$". De propositie is alleen waar wanneer alle waarden voldoen aan P.

Het voorbeeld

$$Happy(Joel) \wedge Happy(Felix) \wedge Happy(Oskar) \wedge Happy(Amanda)$$

kun je schrijven als

$$\forall x\, Happy(x),$$

waarbij de vraag is: zijn Joel, Felix, Oskar én Amanda allemaal blij?

4.4 Existentiële kwantificator

Als je wilt uitdrukken dat er minstens één waarde is waarvoor het predicaat waar is, gebruik je de existentiële kwantor $\exists$.

De notatie $\exists x\, P(x)$ betekent: "er bestaat een x waarvoor $P(x)$ geldt". Deze propositie is waar wanneer één of meer waarden voldoen aan P.

Het voorbeeld

$$Happy(Joel) \vee Happy(Felix) \vee Happy(Oskar) \vee Happy(Amanda)$$

kan worden geschreven als

$$\exists x\, Happy(x).$$

Daarnaast bestaat er de vorm $\exists! x\, P(x)$: "Er bestaat precies één x waarvoor $P(x)$ geldt."

4.5 Gebonden kwantificatoren

Je kunt kwantoren beperken tot een verzameling.

De universele, beperkte vorm:

$$\forall x \in A\ P(x) \quad \text{(alle waarden in } A \text{ voldoen aan } P).$$

De existentiële, beperkte vorm:

$$\exists x \in A\ P(x) \quad \text{(er bestaat minstens één waarde in } A \text{ die voldoet aan } P).$$

4.6 predikatenlogica lezen

- $\forall x\, P(x) \quad \rightarrow$ "Voor alle x geldt: $P(x)$". "Elke [type van x] heeft de eigenschap ...".
- $\exists x\, P(x) \quad \rightarrow$ "Er bestaat een x zodanig dat $P(x)$". "Er is ten minste één [type] waarvoor ...".
- $\exists! x\, P(x) \quad \rightarrow$ "Er bestaat precies één x zodanig dat $P(x)$". In goed Nederlands: "Er is precies één ... die/het ...".
- $P(x) \wedge Q(x) \quad \rightarrow$ "$P(x)$ en $Q(x)$".
- $P(x) \vee Q(x) \quad \rightarrow$ "$P(x)$ en/of $Q(x)$".
- $P(x) \rightarrow Q(x) \quad \rightarrow$ "Als $P(x)$, dan $Q(x)$".
- $\neg P(x) \quad \rightarrow$ "niet $P(x)$".
- $x = y \quad \rightarrow$ "x en y zijn hetzelfde object".
- Meerdere kwantoren: volg strikt de volgorde $\forall x \exists y \rightarrow$ "Voor elke x bestaat een y ...", $\exists y \forall x \rightarrow$ "Er bestaat een y die voor alle x ..." (volgorde verandert betekenis!).

Voorbeeld **(Een Predikatenlogica formule lezen)**:

$$\forall x\big(\mathit{Student}(x) \rightarrow \exists y(\mathit{Course}(y) \wedge \mathit{Enrolled}(x, y) \wedge \neg \mathit{Failed}(x, y))\big).$$

Stap 1: Kies domein = "personen en vakken": variabele x = persoon, y = vak.

Stap 2:

- Buitenste kwantor: $\forall x$ — "Voor alle personen x geldt: ..."
- Binnen: $\mathit{Student}(x) \rightarrow \exists y(\ldots)$ — een implicatie: "als x student is, dan bestaat er een y zodanig dat ..."
- Binnenste existentie: $\exists y(\mathit{Course}(y) \wedge \mathit{Enrolled}(x, y) \wedge \neg \mathit{Failed}(x, y))$.

Stap 3:

> Voor elke persoon x: als x een student is, dan bestaat er een vak y zodanig dat y een vak is, x staat ingeschreven voor y, en het is niet zo dat x voor y gezakt is.

Stap 4: Herschrijf de zin:

> Elke student is ingeschreven voor minstens één vak waarin die niet gezakt is.
