

Logica voor informatica

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

CONTENTS		TERM	LATEX	LATEX COMMAND
Contents	1	1. there exists at least one	\exists	<code>\exists</code>
1 Propositional logic	2	2. there exists one and only one	$\exists!$	<code>\exists !</code>
1.1 Propositions	2	3. there is no	\nexists	<code>\nexists</code>
1.2 Semantics of propositional operators	2	4. for all	\forall	<code>\forall</code>
1.3 Truth tables	2	5. not (logical not)	\neg	<code>\neg</code>
2 sets	3	6. or (logical or)	\vee	<code>\vee</code>
2.1 Membership and equality	3	7. division	\div	<code>\div</code>
2.2 Subsets and supersets	3	8. and (logical and)	\wedge	<code>\wedge</code>
2.3 Operations on sets	3	9. implies	\implies	<code>\implies</code>
2.4 Partitions	3	10. right implication	\Rightarrow	<code>\Rightarrow</code>
2.5 Naive set theory	4	11. is implied by (only if)	\Leftarrow	<code>\Leftarrow</code>
3 Boolean algebra	4	12. left implication	\Leftarrow	<code>\Leftarrow</code>
3.1 Monoids	4	13. if and only if, iff	\iff	<code>\iff</code>
3.2 Boolean algebra	4	14. equivalence	\Leftrightarrow	<code>\Leftrightarrow</code>
3.3 Duality	4	15. Subset	\subset	<code>\subset</code>
		16. Logical XOR (exclusive or)	\oplus	<code>\oplus</code>
		17. Union of sets	\cup	<code>\cup</code>
		18. Empty set	\emptyset	<code>\emptyset</code>
		19. Intersection of sets	\cap	<code>\cap</code>
		20. Union of sets	\cup	<code>\cup</code>

1 Propositielogica

Propositielogica is de studie van bewijzen. Een propositie is te beantwoorden met waar of niet waar

1.1 Propositions

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:

- Atomische 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 houdt 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	1	0	1	1	1	0
1	0	0	0	1	1	0	0
1	1	0	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:	- $q \vee T \Leftrightarrow T$	- $(p \Rightarrow q) \Leftrightarrow (\neg p \vee q)$
- $p \vee F \Leftrightarrow p$	- $q \wedge T \Leftrightarrow q$	Contraposition:
- $p \wedge F \Leftrightarrow F$	Implication:	- $(p \Rightarrow q) \Leftrightarrow (\neg q \Rightarrow \neg p)$

2 sets

Propositie logica 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 set 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: $\{x : x \text{ has the property } P\}$

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

Een collectie van alle prime numbers: $\{x : x \text{ is een priemgetal}\}$

Of $\{s : s \text{ is een strand in Nederland}\}$.

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

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

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

2.1 leden en gelijkheid

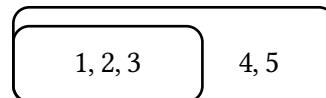
Je kan bekijken of een set een bepaalde member heeft door $7 \in \{3, 7, 14\}$ te doen. Hier is \in te lezen als "is element onderdeel van". Je hebt ook de negatie vorm, \notin , wat "is element niet onderdeel van", oftewel $6 \notin \{3, 7, 14\}$ betekend.

Ook is goed om te weten dat gelijkheid op basis van unieke element rust, dus $\{1, 2, 3, 2, 1\} = \{1, 2, 3\}$. 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 wel/niet 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 $\bar{A} = \{x : x \notin A\}$, 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 pixelcoördinaten, dat kan je zo schrijven:

$W = \{0, 1, \dots, 1697\}$ voor elke X coördinaat

$H = \{0, 1, \dots, 1049\}$ Voor elke Y coördinaat

$W \times H = \{\{0, 0\}, \{0, 1\}, \{0, 2\}, \dots, \{1, 0\}, \dots, \{1679, 1049\}\}$ dit is een volledig wit scherm, elke pixel is gerepresenteerd 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 Boolean algebra is de $+$ hetzelfde als de OR operator, \cdot hetzelfde als de AND operator, $^{-1}$ hetzelfde als NOT

3.1 monoïde

Monoïde is een verzameling A , is niet ledig en heeft minstens 1 element e , en een binaire operator \oplus . Een monoïde heeft een nul-element, dit is een waarde die niks doet in de operatie. $x + 0 = x$, hier doet de 0 helemaal niks, dus 0 is het 0-element in de "+" operatie. $x \times 1 = x$, hier is 1 het nul-element in de monoïde \times ; 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 \cdot 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
	$= x + (x \cdot x^{-1})$	Distributiviteit
	$= x + ()$	Complement
	$= x$	Ident

3.3 duality

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

- 0 met 1 vervangen
- 1 met 0 te vervangen
- $+$ met \cdot te vervangen
- \cdot met $+$ te vervangen