LOGIC AND SETS

Summary 2013-2014

Sets Theory for Computer Science — Sandjai Bhulai

Chapter 1 - 4 (not included: 5, 6)

Lecture slides Logic — Roel de Vrijer

Chapter 1 - 5 (probably not all chapters included too)

Vrije Universiteit

SET THEORY

CHAPTER 1 — SETS

Set (collection/family/class) = imaginary collection of objects
Elements (members) = these objects

Prototype = how to describe the elements of a set:
MultiplesOfTwo := { x : x is an even natural number }
"：=" = definition symbol

Example set A: A := { 1, 2, 3, 4 }
∈ = element-of symbol 4 ∈ A → *4 is an element of set A*
5 ∉ A → *5 is not an element of set A*
⊆ = inclusion symbol { 2, 3 } ⊆ A → *this is a subset of A*
→ "is contained in" { 2, 5} ⊈ A → *this is not a subset of A*

Two sets are equal if they contain exactly the same elements, despite repetitions.

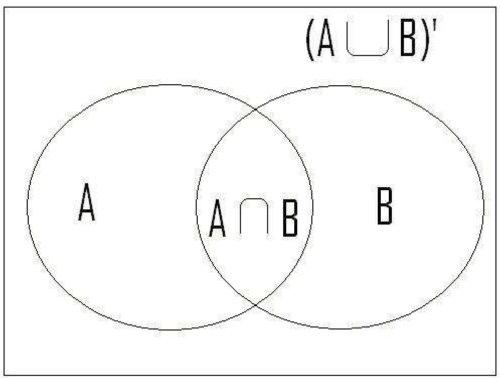
The empty set: ∅ = { }
Number of elements: # { 0, 1, 0, 1 } = 2
∅ = 0

Different sets:
N - natural numbers = { 0, 1, 2, 3, 4, … }
Z - integer numbers = { …, -3, -2, -1, 0, 1, 2, 3, … }
Q - rational numbers = { x : x is a rational number } → *the quotient of two real numbers*
R - real numbers = { x : x is a real number }

Operations:
*The **union** of two sets A and B contains elements that belong to set A or set B*. The result of the operation "union" is a set that we write as A ∪ B.
A ∪ B := { x : x ∈ A or x ∈ B }

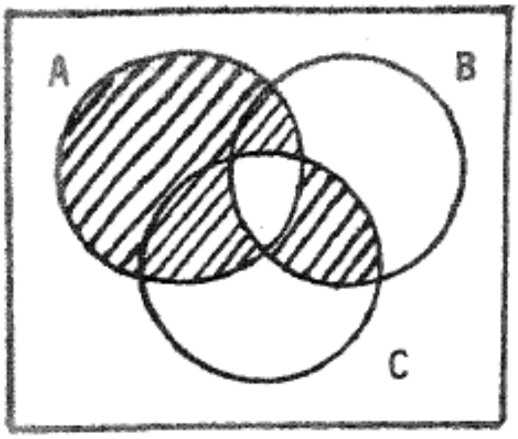
*The **intersection** of set A and set B consists of all elements that are member of both set A and B*. The result of the operation "intersection" is a set that is written as A ∩ B.
A ∩ B := { x : x ∈ A and x ∈ B }

*The **complement** of a set A = the set A' of all elements that are not a member of A*:

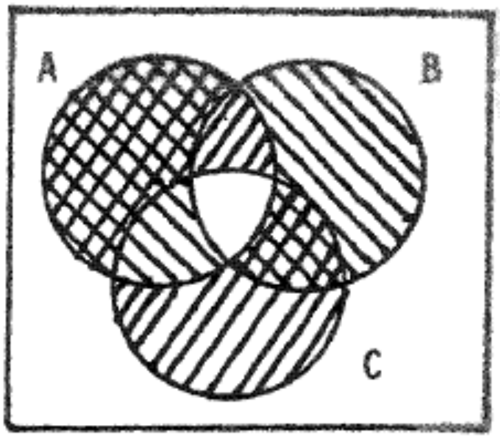


*The **difference** A \ B of two sets A and B is the set of elements that are element of A but not of B*: $A \setminus B := A \cap B'$

*The **symmetric difference** A Δ B are exactly those objects that belong to A or B, but not to both (exclusive membership)*: $A \Delta B := (A \setminus B) \cup (B \setminus A)$



$A\Delta(B\cap C)$



$(A\Delta B)\cap(A\Delta C)$

Venn diagrams are used for:
1. Checking formulas for equality
2. Counting of elements

Commutativity:

$A \cup B = B \cup A$

$A \cap B = B \cap A$

Associativity:

$A \cup (B \cup C) = (A \cup B) \cup C$

$A \cap (B \cap C) = (A \cap B) \cap C$

Distributivity:

$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$

$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$

Identity

$A \cup U = U$ and $A \cup \varnothing = A$

$A \cap U = A$ and $A \cap \varnothing = \varnothing$

Idempotence:

$A \cup A = A$

$A \cap A = A$

Complement:

$A \cup A' = U$

$A \cap A' = \varnothing$

DeMorgan's Law:

$(A \cup B)' = A' \cap B'$

$(A \cap B)' = A' \cup B'$

Involution

$(A')' = A$

Substitution rule: In a formula one can replace a part of a formula with an equal formula.

Laws of the algebra of sets:
Disjoint subsets = the subsets do not have any elements in common

Weekdays := { Monday, Tuesday, Wednesday, Thursday, Friday }
WeekendDays := { Saturday, Sunday }
Weekdays ∩ WeekendDays = ∅
So… Two sets A and B are disjoint when A ∩ B = ∅

Pairwise disjoint = when each element belongs to exactly one of the sets P_1, P_2, and P_3
Conjoint subsets = ?

A **partition** of a set V = the collection of non-empty subsets of V (the “parts” of the partition) such that each element of V belongs to exactly one of the parts

For example: there is a partition { {2, 4, 6, 8 }, { 0, 1, 9 }, { 3, 5, 7 } } of the set Digits into three parts. Every digit belongs to exactly one of these three parts

The summation formula for partitions:
$\#V = \#P_1 + \#P_2 + \#P_3 + \ldots + \#P_n$

CHAPTER 2 — RELATIONS

List = an enumeration of objects (elements) in a certain sequence

For example: DigitList := < 0, 1, 2, 3, …, 9 >
JustAList := < ‘h’, ‘e’, ‘l’, ‘l’, ‘o’ >

- A list of lists is also a valid construction
- The empty list: < >
- The elements have positions in the lists
- The length of the list = the total number of positions (0 - ∞)
  - A pair / tuple = a list of length two
  - A triple = a list of length three
  - n-pair = a list of length n (the n-th coordinate is the element at position n)

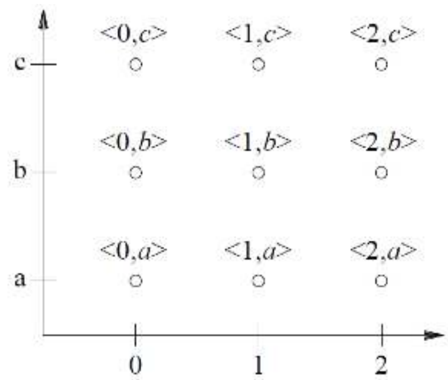
$L_1 = L_2$ if they have the same objects at the same position

$< 1, 1 > \neq < 1 >$ and $< 02, 10 > \neq < 10, 02 >$

A **Cartesian product** of two sets A and B = the set of all pairs $< a, b >$ with $a \in A$ and $b \in B$, and is denoted by $A \times B$

$A \times B := \{ < a, b > : a \in A \text{ and } b \in B \}$

The Cartesian product $\{ 0, 1, 2 \} \times \{ a, b, c \}$:



This can also be done with more sets:
$A_1 \times A_2 \times \ldots \times A_n := \{ < a_1, a_2, \ldots, a_n > : a_1 \in A_1, a_2 \in A_2, \ldots, a_n \in A_n \}$
When one takes n times the Cartesian product of the same set A, then one writes the Cartesian product as A^n (exponential notation)

$\mathbb{R}^2 := \mathbb{R} \times \mathbb{R}$ (the plane); a typical element is $< 3, -2 >$,
$\mathbb{R}^3 := \mathbb{R} \times \mathbb{R} \times \mathbb{R}$ (the space); a typical element is $< 3, -2, 5 >$.
$Alphabet^4$ (all 4-letter words); a typical element is <‘w’, ‘o’, ‘r’, ‘d’>.

The product formula:
$\#(A \times B) = \#A \cdot \#B$

Thus:
$\#(A_1 \times A_2 \times \ldots \times A_n) = \#A_1 \cdot \#A_2 \cdot \ldots \cdot \#A_n$
And also sometimes:
$\#(A^n) = (\#A)^n$

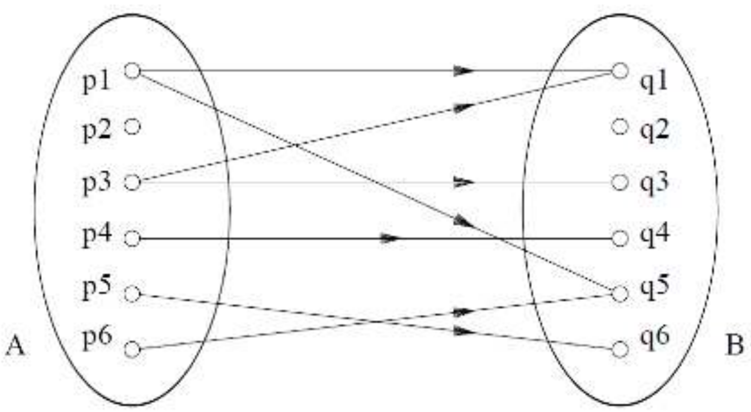
Relations (a set), described in… words, formulas or enumeration of elements
The arity of the relation R = the number n (length n of lists)

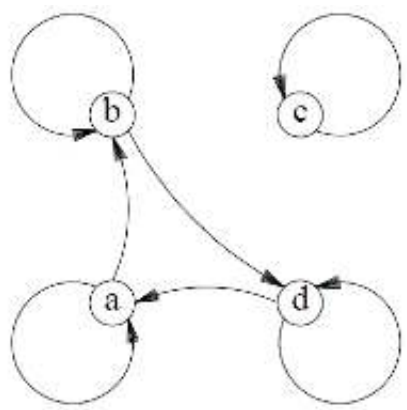
For an n-ary relation R, the formula $< a_1, \ldots, a_n > \in R$
Binary relations = relations with arity 2: $R \subseteq A \times B$ with R the set of the relation!

Description:
$\{ < x_1, \ldots x_n > :$ description (of the relation) for $x_1, \ldots x_n \}$
For example:
IsBrotherOf := { < x, y > : x is a brother of y }
After the introduction of the set People, one can define the relation IsBrotherOf to be of type People × People

Infix notation:
x R y instead of $< x, y > \in R$
In combination with a good choice for the name, the notation for the relation is suggestive:
x IsBrotherOf y instead of $< x, y > \in$ IsBrotherOf
x FollowsCourse y instead of $< x, y > \in$ FollowsCourse

Representation with a Venn diagram and a directed graph:
The relation of type A × B with $A := \{ p_1, p_2, \ldots, p_6 \}$ and $B := \{ q_1, q_2, \ldots, q_6 \}$,
and { < a, a >, < b, b >, < c, c >, < d, d >, < a, b >, < b, d >, < d, a > }:





Note that the relation of the directed graph is a relation of type $\{ a, b, c, d \}^2$

Matrices and relation tables of the binary relation R of type A × B:
A := { a, b, c, d } B := { 1, 2, 3, 4 }

Relation table:

R	1	2	3	4
a	0	1	0	1
b	0	0	0	0
c	1	1	1	1
d	0	0	1	1

Matrix representation:

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

R:= IsParentOf (this is a set!)
x R y → < x, y > ∈ ("an element of the set:") IsParentOf

Inverse relation S of relation R is constructed by reversing all pairs in relation R:

x R y	y S x
x IsParentOf y	y IsChildOf x

Thus: $(\text{IsParentOf})^{-1}$ = IsChildOf

In Venn diagrams/directed graphs, the inverse relation R^{-1} is given by reversing the arrows
With matrices, you change the rows into columns and vice verse

Composite relation = a serie of relations

The pairs < x,y > in S and < y, z > in R are chained to < x, z >

Notation: $R \circ S$ (read as: R after S)

An uncle of a friend → IsFriendOf ○ IsUncleOf

○ = composition operator („after")

Thus:
$R \circ S := \{ <x, z> : \text{there is a } y \text{ with } x\ S\ y \text{ and } y\ R\ z \}$

With multi-composite relations, a composition is associative, because:
$(R \circ S) \circ T = R \circ (S \circ T)$

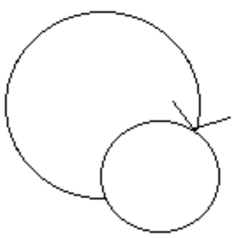
Inverse of composite relations:
$(R \circ S)^{-1} = S^{-1} \circ R^{-1}$

For example: a is a residence of b, who is a teacher of c → c is a student of b, and b lives in a

The type of the relation is the square of the set
→ if V is a set, then a relation of the type V × V is called a relation in V

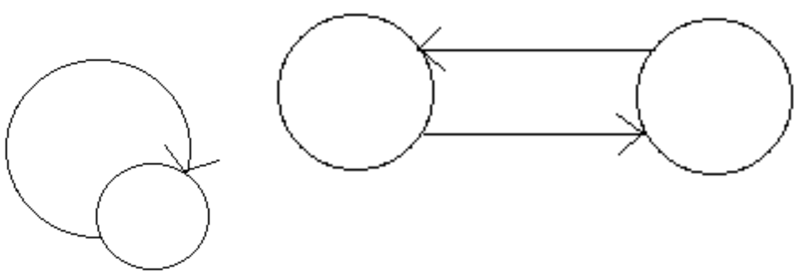
Note that V × W cannot be seen as a relation in a set

Reflexivity = every element of V is with itself in relation R



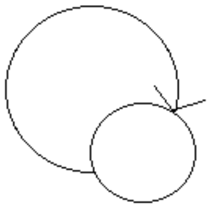
$\forall x \in V, x\ R\ x$

Symmetry = if an element of V is in relation R with a second element of V, then the second element is in relation R with the first element



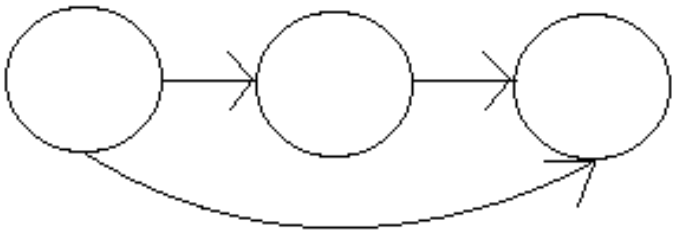
$\forall x, y \in V, (x\ R\ y \rightarrow y\ R\ x)$

Anti-symmetry = if an element of V is in relation R with a second element of V, and the second element is in relation R with the first element, then the two elements are the same



$\forall x, y \in V, (x \mathrel{R} y \land y \mathrel{R} x \rightarrow x = y)$

Transitivity = if an element of V is in relation R with a second element of V, and the second element is in relation R with a third element of V, then the first element is in relation R with the third element



$\forall x, y, z \in V, (x \mathrel{R} y \land y \mathrel{R} z \rightarrow x \mathrel{R} z)$

When not symmetric, nor transitive, it is also not anti-symmetric

CHAPTER 3 — RELATIONS: PARTIAL ORDER

Example
Let A = { 1, 2, 3 }, B = { 3, 4 } and C = { 4, 5, 6 }
Find $A \times (B \cap C)$:

  $B \times C = \{ 4 \}$
  So, $A \times (B \cap C) = \{ 1, 4 \}, \{ 2, 4 \}, \{ 3, 4 \}$

Find $(A \times B) \cap (A \times C)$:

  $A \times B = \{ 1, 3 \}, \{ 1, 4 \}, \{ 2, 3 \}, \{ 2, 4 \}, \{ 3, 3 \}, \{ 3, 4 \}$
  $A \times C = \{ 1, 4 \}, \{ 1, 5 \}, \{ 1, 6 \}, \{ 2, 4 \}, \{ 2, 5 \}, \{ 2, 6 \}, \{ 3, 4 \}, \{ 3, 5 \}, \{ 3, 6 \}$
  So, $(A \times B) \cap (A \times C) = \{ 1, 4 \}, \{ 2, 4 \}, \{ 3, 4 \}$

Recall: a relation R on a set A is a subset of the Cartesian product $A \times A$

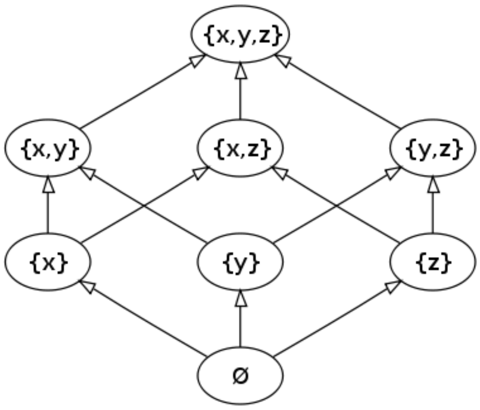
Partial order = when a relation R in a set V satisfies the following three properties:

- Reflexive ($x \mathrel{R} x$: $< a_1, a_2 > R < a_1, a_2 > \rightarrow a_1 + a_2 = a_1 + a_2$)
- Anti-symmetric ($x \mathrel{R} y \cap y \mathrel{R} x \rightarrow x = y$)
- Transitive ($x \mathrel{R} y \cap y \mathrel{R} z \rightarrow x \mathrel{R} z$)

Partially ordered set = set V + partial order R

  You describe partial order with the „smaller than/equal to" or the subset symbol
  When there is a diagram for the relation, it is partially ordered

Power set P (S) of any set S = the set of all subsets of S, including the empty set and S itself. Hasse diagram of power set of three elements:



Notation for power sets of limited cardinality:
- $P_{<K}$ (S)
- $P_{\geq 1}$ (S) : the set of non-empty subsets of S

For every point x:
1. Let Gx := { y : x < y }
2. For every y ∈ Gx: let Gy := { z : y < z } and Gx := Gx / Gy
3. For every y ∈ Gx: draw an arrow between x → y in the Hasse diagram

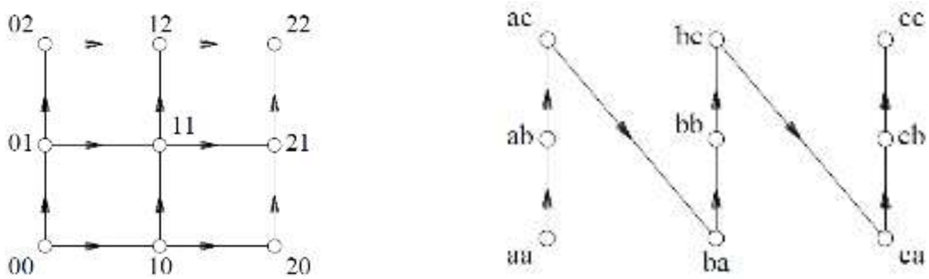
It is very practical when there is a lot of transitivity in the relation

Inclusion = the relationship of one set being a subset of another

Total ordering = $\forall x, \forall y \in V$ ($x \leq y$ OR $y \leq x$)
= partial order with all elements comparable to each other (relation-wise)

Strict ordering „<" = a partial order „smaller than": $x < y :\leftrightarrow x \leq y \wedge x \neq y$
With **lexicographic order**: $A \times B \neq B \times A$ *(=TOTAL)*

Constructions of relations for orderings:



Hasse diagram of { 0, 1, 2 }^{2} with Cartesian ordering (left) and Hasse diagram of { a, b, c }^{2} with lexicographic ordering (right)
If a set is not symmetric + not reflexive → it is not anti-symmetric

Maxima and minima:

#	Property	Definition
(1)	p is a largest element of A	$p \in A \land \forall a \in A, a \leq p$
(2)	p is a smallest element of A	$p \in A \land \forall a \in A, p \leq a$
(3)	p is a maximal element of A	$p \in A \land \forall a \in A, (p \leq a \rightarrow p = a)$
(4)	p is a minimal element of A	$p \in A \land \forall a \in A, (a \leq p \rightarrow a = p)$

It is possible that there are more than one „largest/smallest" elements
For maximal/minimal: each element of A is not strictly larger/smaller than p

CHAPTER 4 — RELATIONS: EQUIVALENCE

Equivalence relation = when a relation R in a set V satisfies the following 3 properties:
- *Reflexive* (x = x) (A relation R in a set A is called reflexive, if (a, a) ∈ R, for every a ∈ A)
  A formula has the same truth table as itself
- *Symmetric* (x = y, so y = x)
- *Transitive* (x = y, and y = z, so x = z)
  Example x ≡ y (they have the same truth table)

Equivalence class = set of elements that are equal to each other with respect to R
  Example { x : all x in this set are equivalent }
Logic equivalence is the same as equivalence relation!

For a positive integer n, two integers a and b are said to be congruent modulo n:
a ≡ b (mod n)
If a - b is an integer multiple of n (or n divides a - b), the number n is called the modulus of the congruence, for example:
  38 ≡ 14 (mod 12) 38 - 14 = 24, 24 / 2 = 12
  -8 ≡ 7 (mod 5) -8 - 7 = -15, -15 / 3 = -5 ~ 5
Thus, **congruence modulo** = x R y:↔ y - x is divisible by m
Think of a sequence of numbers, with elements that occur every x (=m) distance:
  Arithmetic with weekdays: weekdays constantly rujan in a cycle of 7. If it is Tuesday today, than 8 days later it is Wednesday, this is one day further in the cycle (m=7)

- *Reflexivity*: x ≡ x (mod m) because x - x = 0 is divisible by m
- *Symmetry*: Assume that x ≡ y (mod m). Then x - y is divisibly by m. But then also reversely, y - x is divisible by m. Hence, y ≡ x (mod m)
- *Transitivity*: Assume x ≡ y (mod m) and y ≡ z (mod m). Then x - y is divisible by m, and… y - z is divisible by m. But hen the sum of these numbers is divisible by m. This sum is:
  (x - y) + (y - z) = x - z
  Consequently, x ≡ z (mod m)

Quantifiers:
∀ = for all
∃ = there exists
∃! = there exists exactly one

Complete system of representatives = a set S (subset of V) that contains (randomly) one element from each R-equivalence class, so # equivalence classes = # representatives

LOGIC THEORY

CHAPTER 1 — PROPOSITIONAL LOGIC: SYNTAX AND TRUTH-TABLE SEMANTICS

Declarative sentence = a statement that is true or false

For example: grass is green

Propositional logic = a language in which we can express sentences in such a way that brings out their logical structure

Propositional variables — p, q, r (atoms)

Connectives

- ∧ — and (conjunction, only T and T = T)
- ∨ — or = at least one of them (disjunction, only F and F = F)
- ¬ — not = negation (unary connective)
- → — if… then… = implication
  p → q
  assumption → conclusion
- ↔ — if and only if = only T and T or F and F (the rest: binary connectives)

ϕ	ψ	$\phi \to \psi$
T	T	T
T	F	F
F	T	T
F	F	T

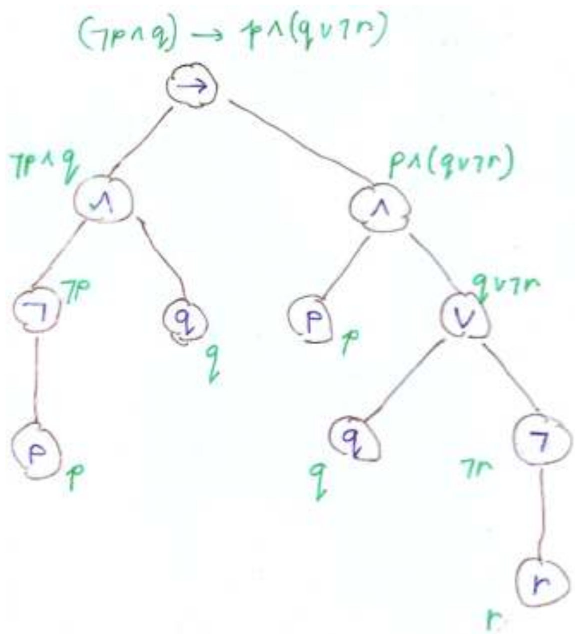
Every propositional variable is a formula

Exclusive ‘or’ = standard or, but false if both elements are true, notation = p ⊕ q

Priority scheme of formulas:

¬ : (parentheses around a negation
∧ ∨ may always be omitted)
→ ↔

Parsing a formula: the parse tree of (¬ p ∧ q) → p ∧ (q ∨ ¬ r):



Premises = a set of formulas

$\phi_1, \phi_2, \phi_3 \vdash \Psi$ → from the premises ϕ_1, ϕ_2, ϕ_3, we may conclude Ψ

$\phi_1, \phi_2, \phi_3 \vDash \Psi$ → if the premises are true, Ψ is true

$\vDash$ = the **semantic entailment** (or logical consequence) relation

Every valuation that makes all formulas $\phi_1 \dots \phi_n$ true, also makes Ψ true

Thus, $\phi_1 \dots \phi_n \vDash \Psi$ does not hold, if a valuation exists that makes all premises $\phi_1 \dots \phi_n$ true, but not the conclusion Ψ

→ this can be checked by making the truth table!

Special case: $\vDash \phi$, this means the conclusion is either way true → the formula ϕ is a tautology

Counterexample = a rule in the truth table where all premises are true, but the conclusion is not (→ a valuation is then a counterexample against the given semantic entailment)

Valuation = an assignment of truth values to propositional variables

Can be done with a parse tree or a truth table

Example of a truth table:

Truth table of $p \vee \neg q \to r$

p	q	r	$\neg q$	$p \vee \neg q$	$p \vee \neg q \to r$
T	T	T	F	T	T
T	T	F	F	T	F
T	F	T	T	T	T
T	F	F	T	T	F
F	T	T	F	F	T
F	T	F	F	F	T
F	F	T	T	T	T
F	F	F	T	T	F

If the truth values in the last columns of two different propositional formulas are identical, they are **logically equivalent** or **semantically equivalent**

$\Phi \equiv \Psi$ (a three-lined equivalent sign)

$\Phi \equiv \Psi$ holds precisely if both $\Phi \vDash \Psi$ and $\Psi \vDash \Phi$

Some important equivalences:

$\neg \neg p \equiv p$

$\neg (p \wedge q) \equiv \neg p \vee \neg q$

$\neg (p \vee q) \equiv \neg p \wedge \neg q$

$p \wedge p \equiv p$
$p \wedge q \equiv q \wedge p$
$p \rightarrow q \equiv \neg p \vee q$
$\neg (p \rightarrow q) \equiv p \wedge \neg q$ $\phi \vee \Psi \equiv \neg(\neg \phi \wedge \neg \Psi)$

Tautology = true for every valuation (a formula with a T on every line of its final column in a truth table)
Contradiction = it is false for every valuation
Contingency = if a propositional formula is not a tautology, nor a contradiction

CHAPTER 2 — SEMANTIC ENTAILMENT LOGIC PUZZLES

Recall: $\phi_1 \ldots \phi_n \models \Psi$
Every valuation that makes all formulas $\phi_1 \ldots \phi_n$ true, also makes Ψ true
Counterexample = If a valuation exists that makes all premises $\phi_1 \ldots \phi_n$ true, but not the conclusion Ψ, $\phi_1 \ldots \phi_n \models \Psi$ does not hold

Metalogic = questions where you need some overview from a higher standpoint
"If $\Phi \vee \Psi$ is a tautology, then what about Φ and Ψ?"
→ $\Phi \vee \Psi$ can be a tautology, without Φ or Ψ being so. Take $\Phi = p$ and $\Psi = \neg p$, for example. Indeed, $p \vee \neg p$ is a tautology, but p and $\neg p$ are contingent formulas

On the *Island of Liars and Truth speakers* every inhabitant has the peculiar property of: always lying, or always speaking the truth
W_x: x is a truth speaker
If islander x makes an assertion Φ, then Φ is true if x is a truth speaker, and false otherwise: $W_x \leftrightarrow \Phi$

"On the island you meet a and b. Islander a says: "we are both liars." What are a and b?'
→ a says: $\neg W_a \vee \neg W_b$, so $W_a \leftrightarrow (\neg W_a \vee \neg W_b)$ is T
→ Now suppose that W_a is T, then (because of the bi-implication), also $\neg W_a \vee \neg W_b$ is T. Then $\neg W_a$ is T as well, and therefore W_a is F. That contradicts our assumption, so W_a must be F
→ Since $W_a \leftrightarrow (\neg W_a \vee \neg W_b)$ is T, we must also have that $\neg W_a \vee \neg W_b$ is F (same truth values as W_a)
→ At least one of the conjuncts of $\neg W_a \vee \neg W_b$ must then be F, and, since we already know that $\neg W_a$ is T, this can only be $\neg W_b$
→ We are ready now, because from $\neg W_b$ is F it follows that W_b is T
→ The conclusion is: a is a liar and b is a truth speaker

Solution via truth table:
This proves that $W_a \leftrightarrow (\neg W_a \vee \neg W_b) \models \neg W_a \wedge \neg W_b$

W_a	W_b	$\neg W_a$	$\neg W_b$	$\neg W_a \wedge \neg W_b$	$W_a \leftrightarrow (\neg W_a \wedge \neg W_b)$	$\neg W_a \wedge W_b$
T	T	F	F	F	F	F
T	F	F	T	F	F	F
F	T	T	F	F	T	T
F	F	T	T	T	F	F

CHAPTER 3 — FUNCTIONAL COMPLETENESS, DNF AND CNF

When you have the results of the truth table but not the formula:

Disjunctive Normal Form (DNF) = formulas corresponding to a truth table, constructed in a special syntactic shape → *a sum of products!*

→ a disjunction of $\phi_1 \vee \ldots \vee \phi_n$, where the ϕ_i's are conjunctions of literals

Literal = a propositional variable or the negation of a propositional variable

For example: p, q, ¬ q, ¬ r, s, …

Every truth tables corresponds to a formula!

Boundary cases: 1) ϕ_x consists of just one single literal, 2) The four ϕ_i 's all a single literal, 3) One single conjunction of literals ϕ_i

Examples:

- $\underbrace{(p \wedge \neg q \wedge \neg r)}_{\psi_1} \vee \underbrace{(\neg p \wedge q \wedge r)}_{\psi_2} \vee \underbrace{(\neg p \wedge \neg q \wedge \neg r)}_{\psi_3}$
- $\underbrace{(p \wedge q)}_{\psi_1} \vee \underbrace{(p \wedge \neg q)}_{\psi_2}$
- $\underbrace{(\neg q \wedge r)}_{\psi_1} \vee \underbrace{(p \wedge q \wedge \neg s)}_{\psi_2} \vee \underbrace{\neg r}_{\psi_3}$

When the truth table is a contradiction, there is no DNF, but that is okay, because you can simply fill in the formula: p ∧ ¬ p

With the DNF-method you can make new formulas for known truth tables, this results in logical equivalences: for example:

The truth table of $p \to q$

p	q	?		
T	T	T	$\Leftarrow$	$p \wedge q$
T	F	F		
F	T	T	$\Leftarrow$	$\neg p \wedge q$
F	F	T	$\Leftarrow$	$\neg p \wedge \neg q$

Solution: $(p \wedge q) \vee (\neg p \wedge q) \vee (\neg p \wedge \neg q)$

So, apparently, we have the logical equivalence:

$p \to q \equiv (p \wedge q) \vee (\neg p \wedge q) \vee (\neg p \wedge \neg q)$

DNF-method:

1. Construct the truth table (if you already know the formula)
2. For the lines that are T, construct a formula ϕ_i
3. Fill these formulas in in $\phi_1 \vee \ldots \vee \phi_n$

Propositional logic is functional complete: every truth table (truth/Boolean function) can be represented by a propositional logic formula → note that you only need a few connectives to express a number of truth functions that is spectacularly large
The number of possible ways to assign a truth value to each line = $2^{(2^n)}$

{∧, ¬, ∨} = adequate system of connectives → every truth function can be expressed by it
Note that {∧, ¬} is already adequate, since: $\phi \vee \Psi \equiv \neg(\neg \phi \wedge \neg \Psi)$
Likewise, {¬, ∨} is adequate too

One single connective can already form an adequate system:
Sheffer stroke = I → At least one, but not both T
$p \wedge q \equiv (p \mid q) \mid (q \mid p)$ and $p \vee q \equiv (p \mid p) \mid (q \mid q)$

Conjunctive Normal Form (CNF) = a DNF, but with the roles of ∧ and ∨ reversed
→ *a product of sums:*
Not CNF: $a \vee (b \wedge c) \equiv (a \vee b) \wedge (a \vee c)$ *: CNF*
→ a conjunction of disjunctions of literals (= a conjunction of 'clauses')

CNF-method:
1. Construct the truth table (if you already know the formula)
2. For the lines that are F, construct a formula ϕ_i
3. Fill these formulas in in $\phi_1 \wedge \ldots \wedge \phi_n$

Example and solution:

p	q	r	ϕ		
T	T	T	T		
T	T	F	F	⇐	$\neg p \vee \neg q \vee r$
T	F	T	T		
T	F	F	F	⇐	$\neg p \vee q \vee r$
F	T	T	T		
F	T	F	T		
F	F	T	T		
F	F	F	F	⇐	$p \vee q \vee r$

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

CNF is a tautology when (meta-logical observations here):
1. The conjunction $\phi_1 \wedge \ldots \wedge \phi_n$ is a tautology if al conjuncts are tautologies
2. A disjunction $\phi_1 \vee \ldots \vee \phi_n$ (a 'clause') is a tautology, if some literal has both a positive and a negative occurrence in it. Example: ***p*** ∨ ¬ q ∨ **¬ *p*** ∨ r

a ↔ b	*equivalent to*	(a → b) ∧ (b → a)
a → b		¬ a ∨ b

Algorithm CNF:

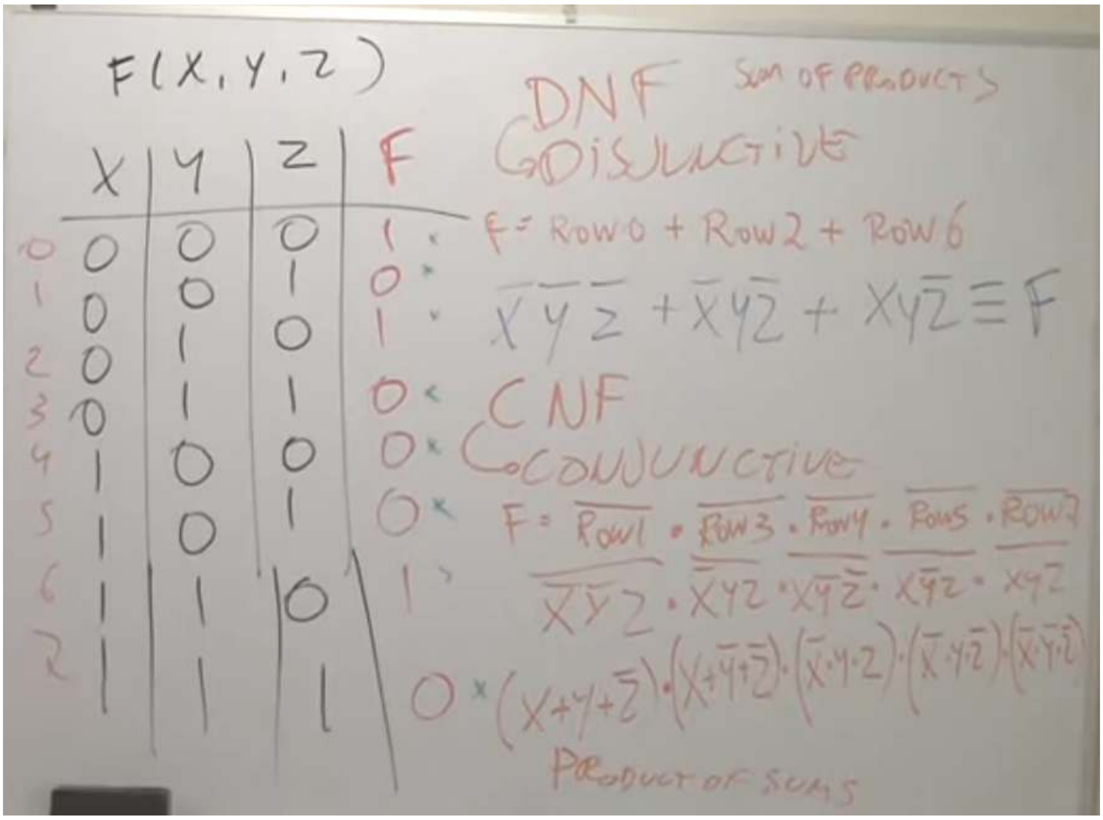
1. IMPL-FREE = remove implication
   $p \rightarrow q \equiv \neg p \vee q$
2. NNF = not normal form,; no „¬" in front of sub formulas
   $\neg (\phi_1 \vee \phi_2) \equiv \neg \phi_1 \wedge \neg \phi_2$ *(DeMorgan)*
3. CNF = divide in classes; solely conjunctions of literals

Example
Convert the following formula into CNF, using the algorithm.
$\neg (p \rightarrow (\neg (q \wedge (\neg p \rightarrow q))))$

1. $\neg (\neg p \vee (\neg (q \wedge (\neg \neg p \vee q))))$
2. ~~¬¬~~ $p \wedge$ ~~¬¬~~ $(q \wedge (\neg \neg p \vee q))))$
3. $p \vee q \wedge (p \vee q)$

DNF vs CNF
- In both, solely the connectives ∧, ¬, ∨ occur
- In both, negations only occur directly in front of a propositional variable
- In a DNF, the connective ∨ never occurs below ∧ in the parse tree
- In a CNF, the connective ∧ never occurs below ∨ in the parse tree

Example of CNF and DNF written in predicate logic:



CHAPTER 4 — LOGICAL EQUIVALENCE, BOOLEAN ALGEBRA, AND BOOLEAN FUNCTIONS, AND BINARY ADDITION

Equivalence relation = a relation that is reflexive, symmetric, and transitive

Within an equivalence class, all formulas are logically equivalent

One of the equivalence classes contains all tautologies, and one contains all contradictions

Examples of equivalence classes under $\equiv$:

$p \to (p \lor q)$	p	$p \to q$	$(p \to q) \to q$	$p \land \neg p$
$(p \land q) \to p$	$\neg\neg p$	$p \to (p \to q)$	$\neg q \to p$	$q \land \neg(p \to q)$
$p \to p$	$(p \to p) \to p$	$\neg(\neg q \land p)$	$p \lor q$	$\neg(p \to p)$
$p \lor \neg p$	$\neg p \to p$	$\neg q \to \neg p$	$q \lor p$	$\bot$
$p \to (p \to p)$	$p \land p$	$\ldots$	$(p \lor q) \land (p \to p)$	$\ldots$
$\top$	$\ldots$	$\ldots$	$\ldots$	$\ldots$
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$

Important logical equivalences:

Commutativity:

$$\phi \lor \psi \equiv \psi \lor \phi$$
$$\phi \land \psi \equiv \psi \land \phi$$

Idempotence:

$$\phi \lor \phi \equiv \phi$$
$$\phi \land \phi \equiv \phi$$

Associativity:

$$\phi \lor (\psi \lor \chi) \equiv (\phi \lor \psi) \lor \chi$$
$$\phi \land (\psi \land \chi) \equiv (\phi \land \psi) \land \chi$$

Complement:

$$\phi \lor \neg\phi \equiv \top$$
$$\phi \land \neg\phi \equiv \bot$$

Distributivity:

$$\phi \lor (\psi \land \chi) \equiv (\phi \lor \psi) \land (\phi \lor \chi)$$
$$\phi \land (\psi \lor \chi) \equiv (\phi \land \psi) \lor (\phi \land \chi)$$

DeMorgan's laws:

$$\neg(\phi \lor \psi) \equiv \neg\phi \land \neg\psi$$
$$\neg(\phi \land \psi) \equiv \neg\phi \lor \neg\psi$$

Identities:

$$\phi \lor \top \equiv \top \qquad \phi \lor \bot \equiv \phi$$
$$\phi \land \top \equiv \phi \qquad \phi \land \bot \equiv \bot$$

Involution:

$$\neg\neg\phi \equiv \phi$$

Axioms of Boolean algebra:

Commutativity:

$$x + y = y + x$$
$$x \cdot y = y \cdot x$$

Idempotence:

$$x + x = x$$
$$x \cdot x = x$$

Associativity:

$$x + (y + z) = (x + y) + z$$
$$x \cdot (y \cdot z) = (x \cdot y) \cdot z$$

Complement:

$$x + x' = 1$$
$$x \cdot x' = 0$$

Distributivity:

$$x + (y \cdot z) = (x + y) \cdot (x + z)$$
$$x \cdot (y + z) = (x \cdot y) + (x \cdot z)$$

DeMorgan's laws:

$$(x + y)' = x' \cdot y'$$
$$(x \cdot y)' = x' + y'$$

Identiteiten:

$$x + 1 = 1 \qquad x + 0 = x$$
$$x \cdot 1 = x \qquad x \cdot 0 = 0$$

Involution:

$$(x')' = x$$

Differences between algebra of sets and Boolean algebra:
- ∪, ∩ instead of +, ·
- ∅, *U* instead of 0, 1

Dual = equations that result from each other by the transformations

Between Boolean equations we have implications of the form: „If these equations hold, than also that one”. Example:

$$y \cdot x = z \cdot x \quad \& \quad y \cdot x' = z \cdot x' \quad \Rightarrow \quad y = z$$

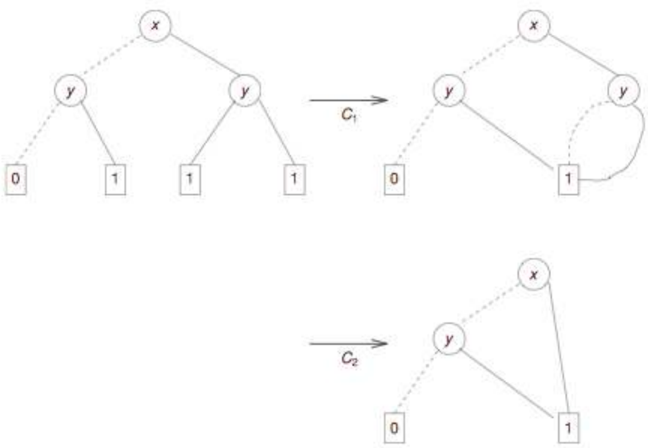
The proof is a simple calculation under the assumption that $y \cdot x = z \cdot x$ and $y \cdot x' = z \cdot x'$ do hold:

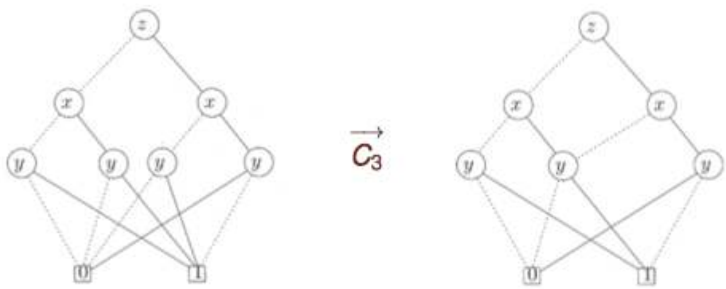
$$\begin{aligned} y = y \cdot 1 &= y \cdot (x + x') \\ &= y \cdot x + y \cdot x' && \text{(distributivity)} \\ &= z \cdot x + z \cdot x' && \text{(by assumptions)} \\ &= z \cdot (x + x') && \text{(distributivity)} \\ &= z \cdot 1 \\ &= z \end{aligned}$$

- Logically equivalent formulas express the same Boolean function
- Each Boolean function can be expressed by a formula (functional completeness)

CHAPTER 5 — BINARY DECISION DIAGRAMS (BDD) AND PREDICATE LOGIC

Reduction rule C1: share identical end nodes
Reduction rule C2: eliminate superfluous decision node
Reduction rule C3: sharing identical sub-BDDs





A BDD is **reduced** if no C-rule can be applied anymore (may correspond to same T/F)
Ordered BDD (**OBDD**) = a BDD from which the order of the variables is fixed, for example the order [x, y, z] from top to bottom

If reduced → a unique truth function is expressed