

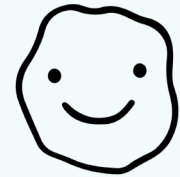
Set & Logic

Yue

Outline

- Set
 - Definitions
 - Ordered pair
- Logic
 - Boolean algebra
 - CNF DNF
 - Truth Tree

Anything marked with
is an exact topic



Maybe next time:

inference rule; induction(prove); functional language;
logic and type; Russel paradox

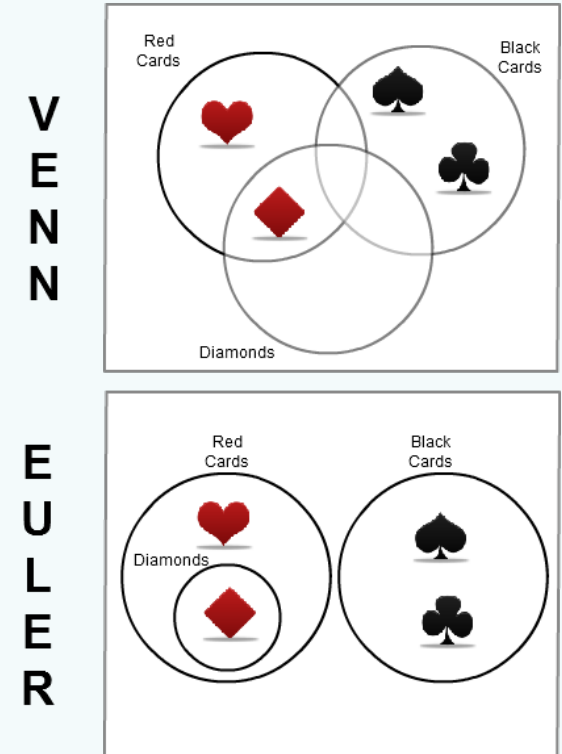
Set

- Distinct & unordered
- Symmetric difference: $A \Delta B = (A - B) \cup (B - A)$
- Venn Diagram vs Euler Diagram:

Difference: for empty set:

Venn diagram: shading it out.

Euler diagram: missing altogether.



Set Theory



- What is set theory?
“machine language” , “assembly language”
every other type of mathematical object can be “compiled” into sets.
- What is set?
 - Informally, a collection of objects.
 - Given a set A and some other object x , you are allowed to ask whether or not x is in the collection A , denoted $x \in A$.
 \Rightarrow completely determined by what all of its elements are.
- Axiom of Extensionality: For two sets A, B ,

$$A = B \iff \forall x (x \in A \iff x \in B)$$


(“axiom” means that this assertion is assumed, while “theorem” needs to be proved)

Naïve Set Theory



- Axiom of Extensionality: For two sets A, B , $A = B \iff \forall x(x \in A \iff x \in B)$
- Definition:

The word **set** is a synonym for “mathematical object” and is left undefined.

- There is a binary relation \in between sets, also undefined.
- The only thing we know about it is that the Axiom of Extensionality holds.
- Assertion \Rightarrow set? Axiom of comprehension: any assertion $\phi(x)$ depending on a variable x , exist unique set A that $\forall x(x \in A \iff \phi(x))$
the set A is denoted $A := \{x | \phi(x)\}$
- But... wait! Russell's paradox  (mentioned later)



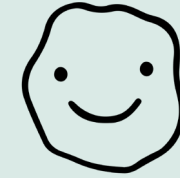
Subset, powerset, union, intersection

- For two sets A, B , $A \subseteq B \iff \forall x(x \in A \implies x \in B)$
- For a set X , its powerset:

$$P(X) := \{A \mid A \subseteq X\} = \{A \mid \forall x(x \in A \implies x \in X)\}$$

- Union: $\cup \mathcal{A} := \{x \mid \exists A \in \mathcal{A}(x \in A)\}$
- Intersection: $\cap \mathcal{A} := \{x \mid \forall A \in \mathcal{A}(x \in A)\}$
- Empty set: $\emptyset = \{x \mid false\}$
- $\cap \emptyset$ is undefined –why? Same definition would yield the entire universe (Vacuous true)

Ordered pair & Cartesian Product



- The process of “compiling” other types of commonly used mathematical objects into sets goes as follows:
Object \rightarrow axioms (capture everything we need to know when using this type of object in mathematical practice) \rightarrow “encode” into sets, \rightarrow prove the desired axioms from the set theory axioms.
might be many reasonable “encodings”.

Marr’s levels of explanation \rightarrow

For ordered pairs, we need to know two things about them in practice:

- For any mathematical objects x, y , there is another object called the pair (x, y) .
- The only feature of an ordered :

$$(x, y) = (a, b) \iff x = a \ \& \ y = b$$

1.2 Understanding Complex Information-Processing Systems

“encode” here

Computational theory	Representation and algorithm	Hardware implementation
What is the goal of the computation, why is it appropriate, and what is the logic of the strategy by which it can be carried out?	How can this computational theory be implemented? In particular, what is the representation for the input and output, and what is the algorithm for the transformation?	How can the representation and algorithm be realized physically?

Figure 1–4. The three levels at which any machine carrying out an information-processing task must be understood.



Ordered pair & Cartesian Product

- Kuratowski's definition: $(a, b) := \{\{a\}, \{a, b\}\}$
- Property: $(x, y) = (a, b) \iff x = a \ \& \ y = b$
- Valid encode:
 - Ordered triple $(a, b, c) := ((a, b), c)$.
 - n-tuple $(x_0, \dots, x_{n-1}) := (((x_0, x_1), x_2), \dots, x_{n-1})$.
- Cartesian Product

For two classes X, Y , their Cartesian product is

$$X \times Y := \{(x, y) \mid x \in X \ \& \ y \in Y\} = \{p \mid \exists x \in X \ \exists y \in Y (p = (x, y))\}$$

Exercise



- (1) Prove that the obvious way of generalizing the standard (Kuratowski) coding of ordered pairs to ordered triples, namely

$$(a, b, c) := \{\{a\}, \{a, b\}, \{a, b, c\}\}$$

fails to satisfy $(a, b, c) = (d, e, f) \iff a = d \ \& \ b = e \ \& \ c = f$

- (2) Prove that the following coding of ordered pairs also satisfies $(x, y) = (a, b) \iff x = a \ \& \ y = b$

$$(a, b) := \{\{0, a\}, \{1, b\}\}$$

(For the purposes of this problem, it does not matter how 0, 1 are coded into sets; all that matters is that **they are distinct from each other** (but possibly not from a, b).)

- (3) Does the following obvious generalization to triples satisfy “extensionality for triples”?

$$(a, b, c) := \{\{0, a\}, \{1, b\}, \{2, c\}\}$$

Haskell



Notation: $\binom{n}{k} = C_n^k$

```
1 module Main where
2
3 -- Define the `&&` operator
4 (&&) :: Bool -> Bool -> Bool → Type annotation
5 True && True = True
6 _ && _ = False } All possible patterns
7
8 -- Test the operator with some example inputs
9 main :: IO ()
10 main = do
11     putStrLn $ show $ True Main.&& True -- expected output: True
12     putStrLn $ show $ True Main.&& False -- expected output: False
13     putStrLn $ show $ False Main.&& True -- expected output: False
14     putStrLn $ show $ False Main.&& False -- expected output: False
15
```

[Haskell Playground](#)

Pattern matching

```
let get_number (n: int) : string =
  match n with
  | 1 -> "one"
  | 2 -> "two"
  | _ -> "other"

let get_pair (p: bool*int) : string =
  match p with
  | (true, 1) -> "one"
  | (false, 2) -> "two"
  | (true, _) -> "other"
  | (_, _) -> "other"
```

[OCaml Playground](#)



I will introduce more about functional programming language when we meet the topic of induction

Natural numbers

Five Peano axioms:

def1

- 0 is a natural number.
- Every natural number has a successor which is also a natural number.
- 0 is not the successor of any natural number.
- If the successor of x equals the successor of y, then x equals y.
- The axiom of induction: If a statement is true of 0, and if the truth of that statement for a number implies its truth for the successor of that number, then the statement is true for every natural number.

Definition 3.136. Let $\prec \subseteq X^2$ be a well-founded relation, with transitive closure $<$. The \prec -rank of $x \in X$ is its $<$ -Mostowski collapse

$$\rho(x) = \rho_{\prec}(x) := \downarrow_{\prec}(x) = \{\downarrow_{\prec}(y) \mid y \prec x\} = \{\rho(y) \mid y \prec x\};$$

by Corollary 3.114 and Exercise 3.15 as in the proof of Proposition 3.115, this is

$$\begin{aligned} &= \{\rho(y) \mid y \prec x \text{ or } \exists z \prec x (y \prec z)\} \\ &= \{\rho(y) \mid y \prec x\} \cup \bigcup_{y \prec x} \{\rho(y') \mid y' \prec y\} \\ &= \{\rho(y) \mid y \prec x\} \cup \bigcup_{y \prec x} \rho(y) \\ &= \bigcup_{y \prec x} (\rho(y) \cup \{\rho(y)\}). \end{aligned}$$

The rank of (X, \prec) is $\rho[X] = \downarrow_{\prec}[X]$.

Remark 3.137. As in Remark 3.95, this inductive definition is justified by Theorem 3.119.

Example 3.138. For $\prec =$ successor on \mathbb{N} , we compute

$$\begin{aligned} \rho(0) &= \emptyset, & \downarrow 0 &= \emptyset, \\ \rho(1) &= \rho(0) \cup \{\rho(0)\} = \{\rho(0)\} = \{\emptyset\}, & \downarrow 1 &= \{\emptyset\}, \\ \rho(2) &= \rho(1) \cup \{\rho(1)\} = \{\rho(0), \rho(1)\} = \{\emptyset, \{\emptyset\}\}, & \downarrow 2 &= \{\{\emptyset\}\}, \\ \rho(3) &= \rho(2) \cup \{\rho(2)\} = \{\rho(0), \rho(1), \rho(2)\} = \{\emptyset, \{\emptyset\}, \{\emptyset, \{\emptyset\}\}\}, & \downarrow 3 &= \{\{\{\emptyset\}\}\}. \end{aligned}$$

(Of course, we will soon define \mathbb{N} by declaring ρ here to be the identity; see Axiom 3.152.)

Corollary 3.139. For well-founded \prec_X, \prec_Y , we have

$$\begin{aligned} x \lesssim_{X,Y} y &\iff \rho(x) \subseteq \rho(y), \\ x (\lesssim \cap \gtrsim) y &\iff \rho(x) = \rho(y). \end{aligned}$$

def2

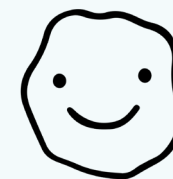
$$0 = \emptyset$$

$$1 = 0 \cup \{0\} = \{0\} = \{\emptyset\}$$

$$2 = 1 \cup \{1\} = \{0, 1\} = \{\emptyset, \{\emptyset\}\}$$

$$3 = 2 \cup \{2\} = \{0, 1, 2\}$$

$$n = n - 1 \cup \{n - 1\} = \{0, 1, \dots, n - 1\}$$



Logic



- Imply $p \rightarrow q \iff \neg p \vee q$

p	q	$p \rightarrow q$
0	0	1
0	1	1
1	0	0
1	1	1

- Exercise: $P \rightarrow (Q \rightarrow R) \iff (P \wedge Q) \rightarrow R$

- SAT function:

Saturation arithmetic is a version of arithmetic in which all operations, such as addition and multiplication, are limited to a fixed range between a minimum and maximum value.

Boolean algebra of all subsets of X (power set algebra)

$P(X), X, \emptyset$

Union, intersection, complementation

$$(1) \quad \emptyset' = X, \quad X' = \emptyset,$$

the laws for forming an intersection with the empty set and a union with the universal set,

$$(2) \quad P \cap \emptyset = \emptyset, \quad P \cup X = X,$$

the identity laws,

$$(3) \quad P \cap X = P, \quad P \cup \emptyset = P,$$

the complement laws,

$$(4) \quad P \cap P' = \emptyset, \quad P \cup P' = X,$$

the double complement law,

$$(5) \quad (P')' = P,$$

the idempotent law,

$$(6) \quad P \cap P = P, \quad P \cup P = P,$$

the De Morgan laws,

$$(7) \quad (P \cap Q)' = P' \cup Q', \quad (P \cup Q)' = P' \cap Q',$$

the commutative laws,

$$(8) \quad P \cap Q = Q \cap P, \quad P \cup Q = Q \cup P,$$

the associative laws,

$$(9) \quad P \cap (Q \cap R) = (P \cap Q) \cap R, \quad P \cup (Q \cup R) = (P \cup Q) \cup R,$$

and the distributive laws,

$$(10) \quad P \cap (Q \cup R) = (P \cap Q) \cup (P \cap R), \\ P \cup (Q \cap R) = (P \cup Q) \cap (P \cup R).$$

Boolean algebra

Non-empty set A , distinguished element 0 and 1

$\vee, \wedge, ' (\neg)$

$$(11) \quad 0' = 1, \quad 1' = 0,$$

$$(12) \quad p \wedge 0 = 0, \quad p \vee 1 = 1,$$

$$(13) \quad p \wedge 1 = p, \quad p \vee 0 = p,$$

$$(14) \quad p \wedge p' = 0, \quad p \vee p' = 1,$$

$$(15) \quad (p')' = p,$$

$$(16) \quad p \wedge p = p, \quad p \vee p = p,$$

$$(17) \quad (p \wedge q)' = p' \vee q', \quad (p \vee q)' = p' \wedge q',$$

$$(18) \quad p \wedge q = q \wedge p, \quad p \vee q = q \vee p,$$

$$(19) \quad p \wedge (q \wedge r) = (p \wedge q) \wedge r, \quad p \vee (q \vee r) = (p \vee q) \vee r,$$

$$(20) \quad p \wedge (q \vee r) = (p \wedge q) \vee (p \wedge r), \quad p \vee (q \wedge r) = (p \vee q) \wedge (p \vee r).$$

Introduction to Boolean Algebras

Paul Halmos , Steven Givant



$$|2^A| = |P(A)| = 2^{|A|}$$

$Y^X := \{f \mid f \text{ is a function } X \rightarrow Y\}.$

- Each subset P of X is naturally associated with a function p from X into 2 , namely the characteristic function of P , defined for each x in X by

$$p(x) = \begin{cases} 1 & \text{if } x \in P, \\ 0 & \text{if } x \notin P. \end{cases}$$

- The correspondence that maps each subset to its characteristic function is a bijection from $P(X)$ to 2^X . The inverse correspondence maps each function q in 2^X to the set consists of elements x in X for which $q(x) = 1$.

Example 2.80. For any set X , there is a bijection between subsets of X and their indicator (or characteristic) functions:

$$\begin{aligned} \mathcal{P}(X) &\cong 2^X \\ A &\mapsto \left(\begin{array}{l} \chi_A : X \rightarrow 2 = \{0, 1\} \\ x \mapsto \begin{cases} 1 & \text{if } x \in A, \\ 0 & \text{else} \end{cases} \end{array} \right) \\ f^{-1}[\{1\}] &\leftrightarrow f. \end{aligned}$$



$$p \vee 0 \Leftrightarrow p \quad (\max\{p, 0\} = p)$$

$$p \wedge 1 \Leftrightarrow p \quad (\min\{p, 1\} = p)$$

In slides,

$$p \wedge 0 \Leftrightarrow 0 \quad (\min\{p, 0\} = 0)$$

$$p \vee 1 \Leftrightarrow 1 \quad (\max\{p, 1\} = 1)$$

but wait, what is max/min here? What is the order? Note the 0 and 1 are just bool value but not numbers

- \cap \cup & max min
- For sets, either one of the equations $P \cap Q = P$ and $P \cup Q = Q$ is equivalent to the inclusion $P \subseteq Q$. This observation motivates the introduction of a binary relation \leq in every Boolean algebra; we write $p \leq q$ or $q \geq p$ in case $p \wedge q = p$, or, $p \vee q = q$.



CNF DNF

- For any proposition φ , there is a proposition φ_{cnf} over the same Boolean variables and in CNF such that $\varphi \Leftrightarrow \varphi_{cnf}$
- (also DNF)

AND of Ors

Example:

$$\varphi = p \rightarrow (q \wedge r)$$

p	q	r	$p \rightarrow (q \wedge r)$
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1



Truth Trees(with only logic operator)

The truth tree method tries to **systematically derive** a contradiction from the assumption that a certain set of statements is true.

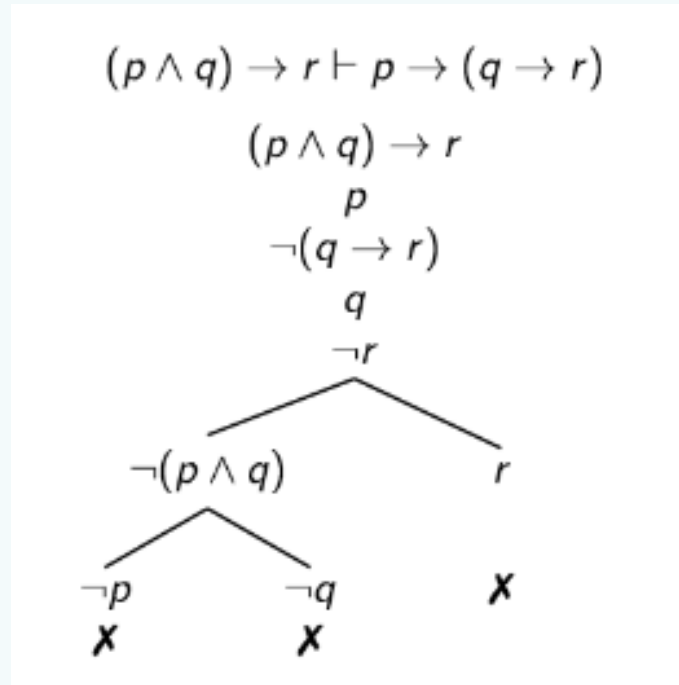
- Infers which statements are forced to be true under this assumption.
- When nothing is forced, then the tree branches into the possible options




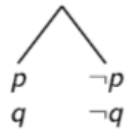
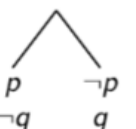
All branches close => the original statement cannot be false => tautology

- A sentence belongs to every branch below it.
- You can close a branch if a statement and its negation both belong to that branch
- A branch is finished if all of its decomposable statements have been decomposed

$p \wedge q$ p q	$\neg(p \vee q)$ $\neg p$ $\neg q$	$\neg(p \rightarrow q)$ p $\neg q$	$\neg\neg p$ p
$\neg(p \wedge q)$ $\neg p$ $\neg q$	$p \vee q$ p q	$p \rightarrow q$ $\neg p$ q	
$p \leftrightarrow q$ p $\neg p$ q $\neg q$	$\neg(p \leftrightarrow q)$ p $\neg p$ $\neg q$ q		

See the example in slides



$p \wedge q$ p q	$\neg(p \vee q)$ $\neg p$ $\neg q$	$\neg(p \rightarrow q)$ p $\neg q$	$\neg\neg p$ p
$\neg(p \wedge q)$ 	$p \vee q$ 	$p \rightarrow q$ 	
$p \leftrightarrow q$ 	$\neg(p \leftrightarrow q)$ 		

Procedure:

- At the 'root' of the tree(negation of what you think is tautology), write down all statements that you try
- Decompose according to the rules until you have a finished open branch or until all branches close.
 - If there is a finished open branch, then that means that it is possible for all statements at the root of the tree to be true.
 - If all branches close, then it is not possible for statements at the root of the tree to be true.

Exercise: prove $((P \wedge Q) \rightarrow R) \leftrightarrow (P \rightarrow (\neg Q \vee R))$

Proving methods



- Truth table
- Logic/set definitions
- Truth tree
- Inference rule (next time) 😊

$x \in (P \cap Q)'$	if and only if	$x \notin P \cap Q,$
	if and only if	$x \notin P$ or $x \notin Q,$
	if and only if	$x \in P'$ or $x \in Q',$
	if and only if	$x \in P' \cup Q'.$

End
~~QAQQ&A~~

Reference

- Umich MATH 582 notes and hw1
- *Introduction to Boolean Algebras*, Paul Halmos , Steven Givant
<https://link.springer.com/book/10.1007/978-0-387-68436-9>
- [Truth Trees \(rpi.edu\)](http://rpi.edu)