Discrete Mathematics in Computer Science

A1. Organizational Matters

Malte Helmert, Gabriele Röger

University of Basel

September 17, 2025

A1.1 Organizational Matters

People

Lecturers

Malte Helmert

- email: `malte.helmert@unibas.ch`
- office: room 06.004, Spiegelgasse 1

Gabi Röger

- email: `gabriele.roeger@unibas.ch`
- office: room 04.005, Spiegelgasse 1

Assistant

David Speck

- email: `davidjakob.speck@unibas.ch`
- office: room 04.003, Spiegelgasse 5

People

Tutors

- Maria Desteffani (maria.desteffani@unibas.ch)
- Pascal von Fellenberg (pascal.vonfellenberg@unibas.ch)
- Carina Schrenk (carina.schrenk@unibas.ch)
- Carina Fehr (carina.fehr@unibas.ch)

Target Audience

target audience:

- this is an introductory course on the Bachelor's level
- we cover mathematical foundations that are particularly useful for the computer science curriculum
- main target audience: B.Sc. Computer Science, 1st semester
- all other students welcome

Enrolment

- `https://services.unibas.ch/`
- official deadline: October 13
- better today, so that you get all relevant emails and access to the ADAM workspace

Discrete Mathematics Course on ADAM

ADAM

`https://adam.unibas.ch/`

- link to website with slides
- submission of exercise sheets
- model solutions for exercise sheets
- link to Discord server (for interaction among participants, but you also get answers from lecturers, assistant and tutors)
- additional material

Language

- The course is taught in English.
- All lecture material is in English.
- We (lecturers, assistant, tutors) speak German and English.
- You are also welcome to ask questions in German.
- Also exercise submissions can be in English or German.

Lectures

- Mon 16:15–18:00, Hörsaal U1.131, Biozentrum
  Wed 16:15–17:00, Hörsaal 1, Pharmazentrum
- first half of the course taught by Gabi Röger,
  second half by Malte Helmert
- on December 17: Q&A session for exam preparation

Exercises

Exercise sheets (homework assignments):

- mostly theoretical exercises
- exercise sheets on ADAM every Monday after the lecture
- must be solved in groups of two or three (not alone or in larger groups)
- due on the following Sunday (23:59) (upload to ADAM at `https://adam.unibas.ch/`)
- we only accept readable PDFs
  → with a bonus point per sheet created with LaTeX (template, cheat sheet and intro on ADAM)

Question: Who has experience with LaTeX?

Exercise Sessions With Tutors

Exercise Sessions (starting September 24/25/27)

Wed 17:15–18:00	Alte Universität, Seminarraum −201 with Carina S.
Wed 17:15–18:00	Spiegelgasse 1, Computer-Labor U1.001 with Pascal
Thu 17:15–18:00	Spiegelgasse 1, Seminarraum 00.003 with Maria
Fri 17:15–18:00	Pharmazentrum, Labor U1075 with Carina F.

- common mistakes/misconceptions (full model solutions on ADAM)
- questions about exercise sheets and the course
- as time permits, support while you solve the exercises

important: please fill in the survey on ADAM for the group allocation until Friday 12:00 (September 19).

Exam

- Written exam
- 6 ECTS credits
- Monday, January 19, 2026, 16:00-18:00
- Maurice E. Müller Saal, Biozentrum
- admission to exam: 50% of the exercise marks
- grade for course determined exclusively by the exam

Required Time

Official calculation

- 1 CP $\approx$ 30 hours
- The course has 6 CP.
- You need to invest about 180 hours.
- With 40 hours for exam preparation,
  this leaves 10–11 hours/week during the teaching period.

Alternative calculation

- A full-time student achieves 30 CP per semester.
- The course corresponds to $1/5$ of 30 CP.
- With a 42h week, this still corresponds to 8.4 hours/week.

Plagiarism

Plagiarism

Plagiarism is presenting someone else's work, ideas, or words as your own, without proper attribution.

For example:

- Using someone's text without citation
- Paraphrasing too closely
- Using information from a source without attribution
- Passing off AI-generated content as your own original work

Long-term impact:

- You undermine your own learning.
- You start to lose confidence in your ability to think, write, and solve problems independently.
- Damage to academic reputation and professional consequences in future careers

Plagiarism in Exercises

- You may discuss material from the course, including the exercise assignments, with your peers.
- But: You have to independently write down your exercise solutions (in your team).
- Help from an LLM is acceptable to the same extent as it is acceptable from someone who is not a member of your team.

Immediate consequences of plagiarism:
- 0 marks for the exercise sheet (first time)
- exclusion from exam (second time)

If in doubt: check with us what is (and isn't) OK before submitting

Exercises too difficult? We are happy to help!

Special Needs?

- We (and the university) strive for equality of students with disabilities or chronic illnesses.
- Contact the lecturers for small adaptations.
- Contact the Students Without Barriers (StoB) service point for general adaptations and disadvantage compensation.

A1.2 About this Course

Content: Discrete Mathematics in Computer Science

- mathematical thinking and proof techniques
- sets and relations
- group theory and permutations
- modular arithmetic
- graphs and trees
- formal logic

Learning Goals

- proficiency in abstract thinking
- ability to formalize mathematical ideas and arguments
- knowledge of common mathematical tools in computer science

Discrete Mathematics in Computer Science

A2. Sets: Foundations

Malte Helmert, Gabriele Röger

University of Basel

September 22, 2025

Discrete Mathematics in Computer Science
September 22, 2025 — A2. Sets: Foundations

A2.1 Sets

Important Building Blocks of Discrete Mathematics

- sets
- relations
- functions

These topics will mainly be the content of part B of the course.

We cover some foundations on sets already now because we will use them for illustrating proof techniques.

Sets

Definition

A set is an unordered collection of distinct objects.

- unorderd: no notion of a "first" or "second" object, e. g. $\{Alice, Bob, Charly\} = \{Charly, Bob, Alice\}$
- distinct: each object contained at most once, e. g. $\{Alice, Bob, Charly\} = \{Alice, Charly, Bob, Alice\}$

German: Menge

Notation

- Specification of sets
  - explicit, listing all elements, e. g. $A = \{1, 2, 3\}$
  - implicit with set-builder notation, specifying a property characterizing all elements, e. g. $A = \{x \mid x \in \mathbb{N}_0 \text{ and } 1 \leq x \leq 3\}$, $B = \{n^2 \mid n \in \mathbb{N}_0\}$
  - implicit, as a sequence with dots, e. g. $\mathbb{Z} = \{\dots, -2, -1, 0, 1, 2, \dots\}$
  - implicit with an inductive definition
- $e \in M$: e is in set M (an element of the set)
- $e \notin M$: e is not in set M
- empty set $\emptyset = \{\}$

Question: Is it true that $1 \in \{\{1, 2\}, 3\}$?

German: Element, leere Menge

Special Sets

- Natural numbers $\mathbb{N}_0 = \{0, 1, 2, \dots\}$
- Integers $\mathbb{Z} = \{\dots, -2, -1, 0, 1, 2, \dots\}$
- Positive integers $\mathbb{Z}_+ = \mathbb{N}_1 = \{1, 2, \dots\}$
- Rational numbers $\mathbb{Q} = \{n/d \mid n \in \mathbb{Z}, d \in \mathbb{N}_1\}$
- Real numbers $\mathbb{R} = (-\infty, \infty)$
  Why do we use interval notation?
  Why didn't we introduce it before?

German: Natürliche ($\mathbb{N}_0$), ganze ($\mathbb{Z}$), rationale ($\mathbb{Q}$), reelle ($\mathbb{R}$) Zahlen

A2.2 Russell's Paradox

Excursus: Barber Paradox

Barber Paradox

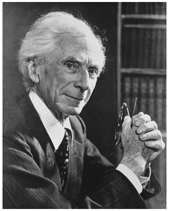
In a town there is only one barber, who is male.

The barber shaves all men in the town,
and only those, who do not shave themselves.

Who shaves the barber?

We can exploit the self-reference to derive a contradiction.

Russell's Paradox


Bertrand Russell

Question

Is the collection of all sets that do not contain themselves as a member a set?

Is $S = \{M \mid M \text{ is a set and } M \notin M\}$ a set?

Assume that S is a set.
If $S \notin S$ then $S \in S \leadsto$ Contradiction
If $S \in S$ then $S \notin S \leadsto$ Contradiction
Hence, there is no such set S.

$\rightarrow$ Not every property used in set-builder notation defines a set.

A2.3 Relations on Sets

Equality

Definition (Axiom of Extensionality)

Two sets A and B are equal (written $A = B$)
if every element of A is an element of B and vice versa.

Two sets are equal if they contain the same elements.

We write $A \neq B$ to indicate that A and B are not equal.

Subsets and Supersets

- $A \subseteq B$: A is a subset of B, i. e., every element of A is an element of B
- $A \subset B$: A is a strict subset of B, i. e., $A \subseteq B$ and $A \neq B$.
- $A \supseteq B$: A is a superset of B if $B \subseteq A$.
- $A \supset B$: A is a strict superset of B if $B \subset A$.

We write $A \not\subseteq B$ to indicate that A is not a subset of B.

Analogously: $\not\subset$, $\not\supseteq$, $\not\supset$

German: Teilmenge, echte Teilmenge, Obermenge, echte Obermenge

Power Set

Definition (Power Set)

The power set $\mathcal{P}(S)$ of a set S is the set of all subsets of S.
That is,

$$\mathcal{P}(S) = \{M \mid M \subseteq S\}.$$

Example: $\mathcal{P}(\{a, b\}) =$

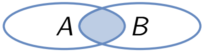
German: Potenzmenge

A2.4 Set Operations

Set Operations

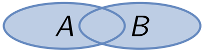
Set operations allow us to express sets in terms of other sets

- intersection $A \cap B = \{x \mid x \in A \text{ and } x \in B\}$

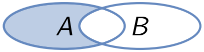


If $A \cap B = \emptyset$ then A and B are disjoint.

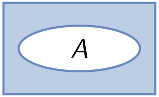
- union $A \cup B = \{x \mid x \in A \text{ or } x \in B\}$



- set difference $A \setminus B = \{x \mid x \in A \text{ and } x \notin B\}$



- complement $\overline{A} = B \setminus A$, where $A \subseteq B$ and B is the set of all considered objects (in a given context)



German: Schnitt, disjunkt, Vereinigung, Differenz, Komplement

Properties of Set Operations: Commutativity

Theorem (Commutativity of $\cup$ and $\cap$)

For all sets A and B it holds that

- $A \cup B = B \cup A$ *and*
- $A \cap B = B \cap A$.

Question: Is the set difference also commutative,
i. e. is $A \setminus B = B \setminus A$ for all sets A and B?

German: Kommutativität

Properties of Set Operations: Associativity

Theorem (Associativity of $\cup$ and $\cap$)

For all sets A, B and C it holds that

- $(A \cup B) \cup C = A \cup (B \cup C)$ *and*
- $(A \cap B) \cap C = A \cap (B \cap C)$.

German: Assoziativität

Properties of Set Operations: Distributivity

Theorem (Union distributes over intersection and vice versa)

For all sets A, B and C it holds that

- $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$ *and*
- $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$.

German: Distributivität

Properties of Set Operations: De Morgan's Law

Augustus De Morgan
British mathematician (1806-1871)

Theorem (De Morgan's Law)

For all sets A and B it holds that

- $\overline{A \cup B} = \overline{A} \cap \overline{B}$ *and*
- $\overline{A \cap B} = \overline{A} \cup \overline{B}$.

A2.5 Cardinality of Finite Sets

Cardinality of Sets

The cardinality $|S|$ measures the size of set S.

A set is finite if it has a finite number of elements.

Definition (Cardinality)
The cardinality of a finite set is the number of elements it contains.

- $|\emptyset| =$
- $|\{x \mid x \in \mathbb{N}_0 \text{ and } 2 \leq x < 5\}| =$
- $|\{3, 0, \{1, 3\}\}| =$
- $|\mathcal{P}(\{1, 2\})| =$

German: Kardinalität oder Mächtigkeit

Cardinality of the Union of Sets

Theorem

For finite sets A and B it holds that $|A \cup B| = |A| + |B| - |A \cap B|$.

Corollary

If finite sets A and B are disjoint then $|A \cup B| = |A| + |B|$.

Cardinality of the Power Set

Theorem

Let S be a finite set. Then $|\mathcal{P}(S)| = 2^{|S|}$.

Proof sketch.

We can construct a subset S' by iterating over all elements e of S and deciding whether e becomes a member of S' or not.

We make $|S|$ independent decisions, each between two options. Hence, there are $2^{|S|}$ possible outcomes.

Every subset of S can be constructed this way and different choices lead to different sets. Thus, $|\mathcal{P}(S)| = 2^{|S|}$. □

Summary

- Sets are unordered collections of distinct objects.
- Important set relations: equality ($=$), subset ($\subseteq$), superset ($\supseteq$) and strict variants ($\subset$ and $\supset$)
- The power set of a set S is the set of all subsets of S.
- Important set operations are intersection, union, set difference and complement.
  - Union and intersection are commutative and associative.
  - Union distributes over intersection and vice versa.
  - De Morgan's law for complement of union or intersection.
- The number of elements in a finite set is called its cardinality.

Discrete Mathematics in Computer Science

A3. Proofs: Introduction

Malte Helmert, Gabriele Röger

University of Basel

September 22, 2025

A3.1 What is a Proof?

A3.1 What is a Proof?

What is a Proof?

A mathematical proof is

- a sequence of logical steps
- starting with one set of statements
- that comes to the conlusion
  that some statement must be true.

What is a statement?

Mathematical Statements

Mathematical Statement

A mathematical statement is a declarative sentence that is either true or false (but not both).

Examples (some true, some false):

- Let $p \in \mathbb{N}_0$ be a prime number. Then p is odd.
- There exists an even prime number.
- The equation $a^k + b^k = c^k$ has infinitely many solutions with $a, b, c, k \in \mathbb{N}_1$ and $k \geq 2$.

German: Mathematische Aussage

Mathematical Statements: Quantification

Statements often use quantification.

- Universal quantification:
  "For all x in set S it holds that $\langle$sub-statement on $x\rangle$."
  This is true if the sub-statement is true for every x in S.

- Existential quantification:
  "There is an x in set S such that $\langle$sub-statement on $x\rangle$."
  This is true if there exists at least one x in S for which the sub-statement is true.

Examples (some true, some false):

- For all $x \in \mathbb{N}_1$ it holds that $x + 1$ is in $\mathbb{N}_1$.
- For all $x \in \mathbb{N}_1$ it holds that $x - 1$ is in $\mathbb{N}_1$.
- There is an $x \in \mathbb{N}_1$ such that $x = \sqrt{x}$.

Mathematical Statements: Preconditions and Conclusions

We can identify preconditions and conclusions.

"If ⟨preconditions⟩ then ⟨conclusions⟩."

The statement is true if the conclusions are true
whenever the preconditions are true.

Not every statement has preconditions. Preconditions are often used in universally quantified sub-statements.

Examples (some true, some false):

- If 4 is a prime number then $2 \cdot 3 = 4$.
- If n is a prime number with $n > 2$ then n is odd.
- For all $p \in \mathbb{N}_1$ it holds that if p is a prime number then p is odd.

Different Statements with the same Meaning

The following statements have the same meaning, we just move preconditions into the quantification, make some aspects implicit, and change the structure.

- For all $p \in \mathbb{N}_1$ it holds that if p is a prime number with $p > 2$ then p is odd.
- For all prime numbers p it holds that if $p > 2$ then p is odd.
- Let p be a natural number with $p > 2$.
  Then p is prime if p is odd.
- If p is a prime number with $p > 2$ then p is odd.
- All prime numbers $p > 2$ are odd.

A single mathematical statement can be expressed in different ways, as long as the meaning stays the same.

Like paraphrasing a sentence in everyday language.

On what Statements can we Build the Proof?

A mathematical proof is

- a sequence of logical steps
- starting with one set of statements
- that comes to the conlusion
  that some statement must be true.

We can use:

- axioms: statements that are assumed to always be true
  in the current context
- theorems and lemmas: statements that were already proven
  - lemma: an intermediate tool
  - theorem: itself a relevant result
- premises: assumptions we make
  to see what consequences they have

German: Axiom, Theorem/Satz, Lemma, Prämisse/Annahme

What is a Logical Step?

A mathematical proof is

- a sequence of logical steps
- starting with one set of statements
- that comes to the conlusion
  that some statement must be true.

Each step directly follows

- from the axioms,
- premises,
- previously proven statements and
- the preconditions of the statement we want to prove.

For a formal definition, we would need formal logics.

The Role of Definitions

Definition

A set is an unordered collection of distinct objects.

The objects in a set are called the elements of the set. A set is said to contain its elements.

We write $x \in S$ to indicate that x is an element of set S, and $x \notin S$ to indicate that S does not contain x.

The set that does not contain any objects is the *empty set* $\emptyset$.

- A definition introduces an abbreviation.
- Whenever we say "set", we could instead say "an unordered collection of distinct objects" and vice versa.
- Definitions can also introduce notation.

German: Definition

Disproofs

- A disproof (refutation) shows that a given mathematical statement is false by giving an example where the preconditions are true, but the conclusion is false.
- This requires deriving, in a sequence of proof steps, the opposite (negation) of the conclusion.

Example (False statement)

"If $p \in \mathbb{N}_0$ is a prime number then p is odd."

Refutation.

Consider natural number 2 as a counter example. It is prime because it has exactly 2 divisors, 1 and itself. It is not odd, because it is divisible by 2. □

German: Widerlegung

A Word on Style

A proof should help the reader to see why the result must be true.

- A proof should be easy to follow.
- Omit unnecessary information.
- Move self-contained parts into separate lemmas.
- In complicated proofs, reveal the overall structure in advance.
- Have a clear line of argument.

$\rightarrow$ Writing a proof is like writing an essay.

Recommended reading (ADAM additional ressources):

- "Some Remarks on Writing Mathematical Proofs" (John M. Lee)
- "§1. Minicourse on technical writing" of "Mathematical Writing" (Donald E. Knuth, Tracy Larrabee, and Paul M. Roberts)

Summary

A proof should convince the reader by logical steps of the truth of some mathematical statement.

Discrete Mathematics in Computer Science

A4. Proof Techniques I

Malte Helmert, Gabriele Röger

University of Basel

September 24, 2025

A4.1 Proof Strategies

Common Forms of Statements

Many statements have one of these forms:

1. "All $x \in S$ with the property P also have the property Q."
2. "A is a subset of B."
3. "For all $x \in S$: x has property P iff x has property Q."
   ("iff": "if and only if")
4. "$A = B$", where A and B are sets.

In the following, we will discuss some typical proof/disproof strategies for such statements.

Proof Strategies

1. "All $x \in S$ with the property P also have the property Q."
   "For all $x \in S$: if x has property P, then x has property Q."
   - To prove, assume you are given an arbitrary $x \in S$ that has the property P.
     Give a sequence of proof steps showing that x must have the property Q.
   - To disprove, find a counterexample, i. e., find an $x \in S$ that has property P but not Q and prove this.

Proof Strategies

2. "A is a subset of B."
   - To prove, assume you have an arbitrary element $x \in A$ and prove that $x \in B$.
   - To disprove, find an element in $x \in A \setminus B$ and prove that $x \in A \setminus B$.

Proof Strategies

3. “For all $x \in S$: x has property P iff x has property Q.”
   (“iff”: “if and only if”)
   - To prove, separately prove “if P then Q” and “if Q then P”.
   - To disprove, disprove “if P then Q” or disprove “if Q then P”.

Proof Strategies

4. "$A = B$", where A and B are sets.
   - To prove, separately prove "$A \subseteq B$" and "$B \subseteq A$".
   - To disprove, disprove "$A \subseteq B$" or disprove "$B \subseteq A$".

Proof Techniques

most common proof techniques:

- direct proof
- indirect proof (proof by contradiction)
- contrapositive
- mathematical induction
- structural induction

A4.2 Direct Proof

Direct Proof

Direct Proof

Direct derivation of the statement by deducing or rewriting.

German: Direkter Beweis

Direct Proof: Example

Theorem

For all sets A, B and C it holds that

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

Proof.

Let A, B and C be arbitrary sets.

We will show separately that

- $A \cap (B \cup C) \subseteq (A \cap B) \cup (A \cap C)$ and that
- $(A \cap B) \cup (A \cap C) \subseteq A \cap (B \cup C)$.

...

Direct Proof: Example cont.

Proof (continued).

We first show that $A \cap (B \cup C) \subseteq (A \cap B) \cup (A \cap C)$:

If $A \cap (B \cup C)$ is empty, the statement is trivially true. Otherwise consider an arbitrary $x \in A \cap (B \cup C)$. By the definition of the intersection it holds that $x \in A$ and that $x \in (B \cup C)$.
We make a case distinction between $x \in B$ and $x \notin B$:

Case 1 ($x \in B$): As $x \in A$ is true, it holds in this case that $x \in (A \cap B)$.

Case 2 ($x \notin B$): From $x \in (B \cup C)$ it follows for this case that $x \in C$. With $x \in A$ we conclude that $x \in (A \cap C)$.

In both cases it holds that $x \in A \cap B$ or $x \in A \cap C$, and we conclude that $x \in (A \cap B) \cup (A \cap C)$.

As x was chosen arbitrarily from $A \cap (B \cup C)$, we have shown that every element of $A \cap (B \cup C)$ is an element of $(A \cap B) \cup (A \cap C)$, so it holds that $A \cap (B \cup C) \subseteq (A \cap B) \cup (A \cap C)$. ...

Direct Proof: Example cont.

Proof (continued).

We will now show that $(A \cap B) \cup (A \cap C) \subseteq A \cap (B \cup C)$.

… **[Homework assignment]** …

Overall we have shown for arbitrary sets A, B and C that $A \cap (B \cup C) \subseteq (A \cap B) \cup (A \cap C)$ and that $(A \cap B) \cup (A \cap C) \subseteq A \cap (B \cup C)$, which concludes the proof of the theorem. □

A4.3 Indirect Proof

Indirect Proof

Indirect Proof (Proof by Contradiction)

- Make an assumption that the statement is false.
- Use the assumption to derive a contradiction.
- This shows that the assumption must be false and hence the original statement must be true.

German: Indirekter Beweis, Beweis durch Widerspruch

Indirect Proof: Example

Theorem

Let A and B be sets. If $A \setminus B = \emptyset$ then $A \subseteq B$.

Proof.

We prove the theorem by contradiction.

Assume that there are sets A and B with $A \setminus B = \emptyset$ and $A \not\subseteq B$.

Let A and B be such sets.

Since $A \not\subseteq B$ there is some $x \in A$ such that $x \notin B$.

For this x it holds that $x \in A \setminus B$.

This is a contradiction to $A \setminus B = \emptyset$.

We conclude that the assumption was false and thus the theorem is true. $\square$

A4.4 Proof by Contrapositive

Contrapositive

(Proof by) Contrapositive

Prove "If A, then B" by proving "If not B, then not A."

Examples:

- Prove "For all $n \in \mathbb{N}_0$: if n^2 is odd, then n is odd" by proving "For all $n \in \mathbb{N}_0$, if n is even, then n^2 is even."
- Prove "For all $n \in \mathbb{N}_0$: if n is not a square number, then $\sqrt{n}$ is irrational" by proving "For all $n \in \mathbb{N}_0$: if $\sqrt{n}$ is rational, then n is a square number."

German: Kontraposition

Contrapositive: Example

Theorem

For any sets A and B: If $A \subseteq B$ then $A \setminus B = \emptyset$.

Proof.

We prove the theorem by contrapositive, showing for any sets A and B that if $A \setminus B \neq \emptyset$ then $A \not\subseteq B$.

Let A and B be arbitrary sets with $A \setminus B \neq \emptyset$.

As the set difference is not empty, there is at least one x with $x \in A \setminus B$. By the definition of the set difference ($\setminus$), it holds for such x that $x \in A$ and $x \notin B$.

Hence, not all elements of A are elements of B, so it does not hold that $A \subseteq B$. □

Summary

- There are standard strategies for proving some common forms of statements, e.g. some property of all elements of a set.
- Direct proof: derive statement by deducing or rewriting.
- Indirect proof: derive contradiction from the assumption that the statement is false.
- Proof by contrapositive: Prove "If A, then B" by proving "If not B, then not A.".

Discrete Mathematics in Computer Science

A5. Proof Techniques II

Malte Helmert, Gabriele Röger

University of Basel

September 29, 2025

Discrete Mathematics in Computer Science

September 29, 2025 — A5. Proof Techniques II

A5.1 Mathematical Induction

Proof Techniques

most common proof techniques:

- direct proof
- indirect proof (proof by contradiction)
- contrapositive
- mathematical induction
- structural induction

Mathematical Induction

Concrete Mathematics by Graham, Knuth and Patashnik (p. 3)

Mathematical induction proves that

we can climb as high as we like on a ladder,

by proving that we can climb onto the bottom rung (the basis)

and that

from each rung we can climb up to the next one (the step).

Propositions

Consider a statement on all natural numbers n with $n \geq m$.

- E.g. "Every natural number $n \geq 2$ can be written as a product of prime numbers."
  - $P(2)$: "2 can be written as a product of prime numbers."
  - $P(3)$: "3 can be written as a product of prime numbers."
  - $P(4)$: "4 can be written as a product of prime numbers."
  - ...
  - $P(n)$: "n can be written as a product of prime numbers."
  - For every natural number $n \geq 2$ proposition $P(n)$ is true.

Proposition $P(n)$ is a mathematical statement that is defined in terms of natural number n.

Mathematical Induction

Mathematical Induction

Proof (of the truth) of proposition $P(n)$
for all natural numbers n with $n \geq m$:

- basis: proof of $P(m)$
- induction hypothesis (IH):
  suppose that $P(k)$ is true for all k with $m \leq k \leq n$
- inductive step: proof of $P(n+1)$
  using the induction hypothesis

German: Vollständige Induktion, Induktionsanfang,
Induktionsannahme oder Induktionsvoraussetzung,
Induktionsschritt

Mathematical Induction: Example I

Theorem

Every natural number $n \geq 2$ can be written as a product of prime numbers, i. e. $n = p_1 \cdot p_2 \cdot \ldots \cdot p_m$ with prime numbers $p_1, \ldots, p_m$.

Proof.

Mathematical Induction over n:

basis $n = 2$: trivially satisfied, since 2 is prime

IH: Every natural number k with $2 \leq k \leq n$
can be written as a product of prime numbers. ...

Mathematical Induction: Example I

Theorem

Every natural number $n \geq 2$ can be written as a product of prime numbers, i. e. $n = p_1 \cdot p_2 \cdot \ldots \cdot p_m$ with prime numbers $p_1, \ldots, p_m$.

Proof (continued).

inductive step $n \rightarrow n + 1$:

- Case 1: $n + 1$ is a prime number $\rightsquigarrow$ trivial
- Case 2: $n + 1$ is not a prime number.
  There are natural numbers $2 \leq q, r \leq n$ with $n + 1 = q \cdot r$.
  Using the IH shows that there are prime numbers
  $q_1, \ldots, q_s$ with $q = q_1 \cdot \ldots \cdot q_s$ and
  $r_1, \ldots, r_t$ with $r = r_1 \cdot \ldots \cdot r_t$.
  Together this means $n + 1 = q_1 \cdot \ldots \cdot q_s \cdot r_1 \cdot \ldots \cdot r_t$.

$\square$

Mathematical Induction: Example II

Theorem

Let S be a finite set. Then $|\mathcal{P}(S)| = 2^{|S|}$.

What proposition can we use to prove this
with mathematical induction?

Proof by Induction

Proof.

By induction over $|S|$.

Basis ($|S| = 0$): Then $S = \emptyset$ and $|\mathcal{P}(S)| = |\{\emptyset\}| = 1 = 2^0$.

IH: For all sets S with $|S| \leq n$, it holds that $|\mathcal{P}(S)| = 2^{|S|}$.

Inductive Step ($n \to n + 1$):

Let S' be an arbitrary set with $|S'| = n + 1$ and
let e be an arbitrary member of S'.

Let further $S = S' \setminus \{e\}$ and $X = \{S'' \cup \{e\} \mid S'' \in \mathcal{P}(S)\}$.

Then $\mathcal{P}(S') = \mathcal{P}(S) \cup X$. As $\mathcal{P}(S)$ and X are disjoint and $|X| = |\mathcal{P}(S)|$, it holds that $|\mathcal{P}(S')| = 2|\mathcal{P}(S)|$.

Since $|S| = n$, we can use the IH and get

$$|\mathcal{P}(S')| = 2 \cdot 2^{|S|} = 2 \cdot 2^n = 2^{n+1} = 2^{|S'|}.$$

$\square$

Weak vs. Strong Induction

- Weak induction: Induction hypothesis only supposes that $P(k)$ is true for $k = n$
- Strong induction: Induction hypothesis supposes that $P(k)$ is true for all $k \in \mathbb{N}_0$ with $m \leq k \leq n$
  - also: complete induction

Our previous definition corresponds to strong induction.

Which of the examples had also worked with weak induction?

Is Strong Induction More Powerful than Weak Induction?

Are there statements that we can prove with strong induction but not with weak induction?

We can always use a stronger proposition:

- “Every $n \in \mathbb{N}_0$ with $n \geq 2$ can be written as a product of prime numbers.”
- $P(n)$: “n can be written as a product of prime numbers.”
- $P'(n)$: “all $k \in \mathbb{N}_0$ with $2 \leq k \leq n$ can be written as a product of prime numbers.”

A5.2 Structural Induction

Inductively Defined Sets: Examples

Example (Natural Numbers)

The set $\mathbb{N}_0$ of natural numbers is inductively defined as follows:

- 0 is a natural number.
- If n is a natural number, then $n + 1$ is a natural number.

Example (Binary Tree)

The set $\mathcal{B}$ of binary trees is inductively defined as follows:

- $\Box$ is a binary tree (a leaf)
- If L and R are binary trees, then $\langle L, \bigcirc, R \rangle$ is a binary tree (with inner node $\bigcirc$).

Implicit statement: all elements of the set can be constructed by finite application of these rules

German: Binärbaum, Blatt, innerer Knoten

Inductive Definition of a Set

Inductive Definition

A set M can be defined inductively by specifying

- basic elements that are contained in M
- construction rules of the form "Given some elements of M, another element of M can be constructed like this."

German: Induktive Definition, Basiselemente, Konstruktionsregeln

Structural Induction

Structural Induction

Proof of statement for all elements of an inductively defined set

- basis: proof of the statement for the basic elements
- induction hypothesis (IH):
  suppose that the statement is true for some elements M
- inductive step: proof of the statement for elements
  constructed by applying a construction rule to M
  (one inductive step for each construction rule)

German: Strukturelle Induktion

Structural Induction: Example (1)

Definition (Leaves of a Binary Tree)

The number of leaves of a binary tree B, written $\mathit{leaves}(B)$, is defined as follows:

$$\mathit{leaves}(\square) = 1$$
$$\mathit{leaves}(\langle L, \bigcirc, R \rangle) = \mathit{leaves}(L) + \mathit{leaves}(R)$$

Definition (Inner Nodes of a Binary Tree)

The number of inner nodes of a binary tree B, written $\mathit{inner}(B)$, is defined as follows:

$$\mathit{inner}(\square) = 0$$
$$\mathit{inner}(\langle L, \bigcirc, R \rangle) = \mathit{inner}(L) + \mathit{inner}(R) + 1$$

Structural Induction: Example (2)

Theorem

For all binary trees B: $\mathit{inner}(B) = \mathit{leaves}(B) - 1$.

Proof.

induction basis:

$\mathit{inner}(\square) = 0 = 1 - 1 = \mathit{leaves}(\square) - 1$

$\leadsto$ statement is true for base case ...

Structural Induction: Example (3)

Proof (continued).

induction hypothesis:

to prove that the statement is true for a composite tree $\langle L, \bigcirc, R\rangle$, we may use that it is true for the subtrees L and R.

inductive step for $B = \langle L, \bigcirc, R\rangle$:

$$\begin{aligned} \mathit{inner}(B) &= \mathit{inner}(L) + \mathit{inner}(R) + 1 \\ &\overset{\text{IH}}{=} (\mathit{leaves}(L) - 1) + (\mathit{leaves}(R) - 1) + 1 \\ &= \mathit{leaves}(L) + \mathit{leaves}(R) - 1 = \mathit{leaves}(B) - 1 \end{aligned}$$

□

Example: Tarradiddles

Example (Tarradiddles)

The set of tarradiddles is inductively defined as follows:

- ✈ is a tarradiddle.
- ♥ is a tarradiddle.
- If x and y are tarradiddles, then x✿✿y is a tarradiddle.
- If x and y are tarradiddles, then ✿x✈y✿ is a tarradiddle.

How do you prove with structural induction that every tarradiddle contains an even number of flowers?

A5.3 Excursus: Computer-assisted Theorem Proving

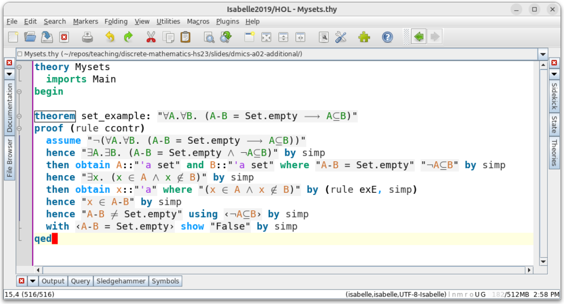
Computer-assisted Proofs

- Computers can help proving theorems.
- Computer-aided proofs have for example been used for proving theorems by exhaustion.
- Example: Four color theorem

Interactive Theorem Proving

- On the lowest abstraction level, rigorous mathematical proofs rely on formal logic.
- On this level, proofs can be automatically verified by computers.
- Nobody wants to write or read proofs on this level of detail.
- In Interactive Theorem Proving a human guides the proof and the computer tries to fill in the details.
- If it succeeds, we can be very confident that the proof is valid.
- Example theorem provers: Isabelle/HOL, Lean

Example


Isabelle2019/HOL - Mysets.thy
File Edit Search Markers Folding View Utilities Macros Plugins Help
Mysets.thy (~/repos/teaching/discrete-mathematics-hs23/slides/dmics-a02-additional/)
Documentation
File Browser
Sidekick
State
Theories
Output
Query
Sledgehammer
Symbols
15,4 (516/516)
(isabelle,isabelle,UTF-8-Isabelle) nmroUG 182/512MB 2:58 PM

```
theory Mysets
  imports Main
begin

theorem set_example: "∀A.∀B. (A-B = Set.empty ⟶ A⊆B)"
proof (rule ccontr)
  assume "¬(∀A.∀B. (A-B = Set.empty ⟶ A⊆B))"
  hence "∃A.∃B. (A-B = Set.empty ∧ ¬A⊆B)" by simp
  then obtain A::"'a set" and B::"'a set" where "A-B = Set.empty" "¬A⊆B" by simp
  hence "∃x. (x ∈ A ∧ x ∉ B)" by simp
  then obtain x::"'a" where "(x ∈ A ∧ x ∉ B)" by (rule exE, simp)
  hence "x ∈ A-B" by simp
  hence "A-B ≠ Set.empty" using ‹¬A⊆B› by simp
  with ‹A-B = Set.empty› show "False" by simp
qed
```

⇝ Demo

Summary

- Mathematical induction is used to prove a proposition P for all natural numbers $\geq m$.
  - Prove $P(m)$.
  - Make hypothesis that $P(k)$ is true for $m \leq k \leq n$.
  - Establish $P(n+1)$ using the hypothesis.
- Structural induction applies the same general concept to prove a proposition P for all elements of an inductively defined set.

Discrete Mathematics in Computer Science

B1. Tuples & Cartesian Product

Malte Helmert, Gabriele Röger

University of Basel

October 1, 2025

B1.1 Tuples and the Cartesian Product

B1.1 Tuples and the Cartesian Product

Motivation

- A set is an unordered collection of distinct objects.
- We often need a more structured way of representation.
  - A person is associated with a name, address, phone number.
  - A set of persons makes sense in many contexts.
  - Representing the associated data as a set rather not.
- We could for example want to
  - directly access the name of a person, or
  - have a separate billing and delivery address for some order, but in general, these can be the same.
- Tuples are mathematical building blocks that support this.

Sets vs. Tuples

- A set is an unordered collection of distinct objects.
- A tuple is an ordered sequence of objects.

Tuples

- k-tuple: ordered sequence of k objects ($k \in \mathbb{N}_0$)
- written $(o_1, \dots, o_k)$ or $\langle o_1, \dots, o_k \rangle$
- unlike sets, order matters ($\langle 1, 2 \rangle \neq \langle 2, 1 \rangle$)
- objects may occur multiple times in a tuple
- objects contained in tuples are called components
- terminology:
  - $k = 2$: (ordered) pair
  - $k = 3$: triple
  - more rarely: quadruple, quintuple, sextuple, septuple, ...
- if k is clear from context (or does not matter), often just called tuple

German: k-Tupel, Komponente, (geordnetes) Paar, Tripel, Quadrupel

Equality of Tuples

Definition (Equality of Tuples)

Two n-tuples $t = \langle o_1, \dots, o_n \rangle$ and $t' = \langle o'_1, \dots, o'_n \rangle$
are equal ($t = t'$) if for $i \in \{1, \dots, n\}$ it holds that $o_i = o'_i$.

Cartesian Product

Definition (Cartesian Product and Cartesian Power)

Let $S_1, \dots, S_n$ be sets. The Cartesian product $S_1 \times \dots \times S_n$ is the following set of n-tuples:

$$S_1 \times \dots \times S_n = \{\langle x_1, \dots, x_n \rangle \mid x_1 \in S_1, x_2 \in S_2, \dots, x_n \in S_n\}.$$

The k-ary Cartesian power of a set S (with $k \in \mathbb{N}_1$) is the set $S^k = \{\langle o_1, \dots, o_k \rangle \mid o_i \in S \text{ for all } i \in \{1, \dots, k\}\} = \underbrace{S \times \dots \times S}_{k \text{ times}}$.

René Descartes: French mathematician and philosopher (1596–1650)

Example: $A = \{a, b\}$, $B = \{1, 2, 3\}$

$$A \times B = \{(a, 1), (a, 2), (a, 3), (b, 1), (b, 2), (b, 3)\}$$
$$A^2 = \{(a, a), (a, b), (b, a), (b, b)\}$$

German: Kartesisches Produkt

(Non-)properties of the Cartesian Product

The Cartesian product is

- not commutative, in most cases $A \times B \neq B \times A$.
- not associative, in most cases $(A \times B) \times C \neq A \times (B \times C)$

Why? Exceptions?

Summary

- A k-tuple is an ordered sequence of k objects, called the components of the tuple.
- 2-tuples are also called pairs and 3-tuples triples.
- The Cartesian Product $S_1 \times \cdots \times S_n$ of set $S_1, \ldots, S_n$ is the set of all tuples $\langle o_1, \ldots, o_n \rangle$, where for all $i \in \{1, \ldots, n\}$ component o_i is an element of S_i.

Discrete Mathematics in Computer Science

B2. Relations

Malte Helmert, Gabriele Röger

University of Basel

October 6, 2025

Discrete Mathematics in Computer Science

October 6, 2025 — B2. Relations

B2.1 Relations

Relations: Informally

- Intuitively, a mathematical relation connects elements from several (possibly different) sets by specifying related groupings.
- We already know some relations, e. g.
  - $\subseteq$ relation for sets
  - $\leq$ relation for natural numbers
- These are examples of binary relations, considering pairs of objects.
- There are also relations of higher arity, e. g.
  - "$x + y = z$" for integers x, y, z.
  - "The name, address and office number belong to the same person."
- Relations are for example important for relational databases, semantic networks or knowledge representation and reasoning.

Relations

Definition (Relation)

Let $S_1, \dots, S_n$ be sets.

A relation over $S_1, \dots, S_n$ is a set $R \subseteq S_1 \times \dots \times S_n$.

The arity of R is n.

A relation of arity n is a set of n-tuples.

German: Relation, Stelligkeit

Relations: Examples

- $\subseteq \; = \{(S, S') \mid S$ and S' are sets and for every $x \in S$ it holds that $x \in S'\}$
- $\leq \; = \{(x, y) \mid x, y \in \mathbb{N}_0$ and $x < y$ or $x = y\}$
- $R = \{(x, y, z) \mid x, y, z \in \mathbb{Z}$ and $x + y = z\}$
- $R' = \{($Gabi Röger, Spiegelgasse 1, 04.005$)$, $($Malte Helmert, Spiegelgasse 1, 06.004$)$, $($David Speck, Spiegelgasse 5, 04.003$)\}$

B2.2 Properties of Binary Relations

Binary Relation

A binary relation is a relation of arity 2:

Definition (binary relation)

A binary relation is a relation over two sets A and B.

- Instead of $(x, y) \in R$, we also write xRy, e. g.
  $x \leq y$ instead of $(x, y) \in \ \leq$
- If the sets are equal, we say "R is a binary relation over A" instead of "R is a binary relation over A and A".
- Such a relation over a set is also called a homogeneous relation or an endorelation.

German: zweistellige Relation, homogene Relation

Reflexivity

A reflexive relation relates every object to itself.

Definition (reflexive)

A binary relation R over set A is reflexive if for all $a \in A$ it holds that $(a, a) \in R$.

Which of these relations are reflexive?

- $R = \{(a, a), (a, b), (a, c), (b, a), (b, c), (c, c)\}$ over $\{a, b, c\}$
- $R = \{(a, a), (a, b), (a, c), (b, b), (b, c), (c, c)\}$ over $\{a, b, c\}$
- equality relation $=$ on natural numbers
- less-than relation $\leq$ on natural numbers
- strictly-less-than relation $<$ on natural numbers

German: reflexiv

Irreflexivity

A irreflexive relation never relates an object to itself.

Definition (irreflexive)

A binary relation R over set A is irreflexive if for all $a \in A$ it holds that $(a, a) \notin R$.

Which of these relations are irreflexive?

- $R = \{(a, a), (a, b), (a, c), (b, a), (b, c), (c, c)\}$ over $\{a, b, c\}$
- $R = \{(a, a), (a, b), (a, c), (b, b), (b, c), (c, c)\}$ over $\{a, b, c\}$
- equality relation $=$ on natural numbers
- less-than relation $\leq$ on natural numbers
- strictly-less-than relation $<$ on natural numbers

German: irreflexiv

Symmetry

Definition (symmetric)

A binary relation R over set A is symmetric
if for all $a, b \in A$ it holds that $(a, b) \in R$ iff $(b, a) \in R$.

Which of these relations are symmetric?

- $R = \{(a, a), (a, b), (a, c), (b, a), (c, a), (c, c)\}$ over $\{a, b, c\}$
- $R = \{(a, a), (a, b), (a, c), (b, b), (b, c), (c, c)\}$ over $\{a, b, c\}$
- equality relation $=$ on natural numbers
- less-than relation $\leq$ on natural numbers
- strictly-less-than relation $<$ on natural numbers

German: symmetrisch

Asymmetry and Antisymmetry

How do these properties relate to irreflexivity?

Definition (asymmetric and antisymmetric)

Let R be a binary relation over set A.

Relation R is asymmetric if for all $a, b \in A$ it holds that if $(a, b) \in R$ then $(b, a) \notin R$.

Relation R is antisymmetric if for all $a, b \in A$ with $a \neq b$ it holds that if $(a, b) \in R$ then $(b, a) \notin R$.

Which of these relations are asymmetric/antisymmetric?

- $R = \{(a, a), (a, b), (a, c), (b, a), (c, a), (c, c)\}$ over $\{a, b, c\}$
- $R = \{(a, a), (a, b), (a, c), (b, b), (b, c), (c, c)\}$ over $\{a, b, c\}$
- equality relation $=$ on natural numbers
- less-than relation $\leq$ on natural numbers
- strictly-less-than relation $<$ on natural numbers

German: asymmetrisch, antisymmetrisch

Transitivity

Definition

A binary relation R over set A is transitive
if it holds for all $a, b, c \in A$ that
if $(a, b) \in R$ and $(b, c) \in R$ then $(a, c) \in R$.

Which of these relations are transitive?

- $R = \{(a, a), (a, b), (a, c), (b, a), (c, a), (c, c)\}$ over $\{a, b, c\}$
- $R = \{(a, a), (a, b), (a, c), (b, b), (b, c), (c, c)\}$ over $\{a, b, c\}$
- equality relation $=$ on natural numbers
- less-than relation $\leq$ on natural numbers
- strictly-less-than relation $<$ on natural numbers

German: transitiv

Summary

- A relation over sets $S_1, \dots, S_n$ is a set $R \subseteq S_1 \times \dots \times S_n$.
- A binary relation is a relation over two sets.
- A binary relation over set S is a relation $R \subseteq S \times S$ and also called a homogeneous relation.
- A binary relation R over A is
  - reflexive if $(a, a) \in R$ for all $a \in A$,
  - irreflexive if $(a, a) \notin R$ for all $a \in A$,
  - symmetric if for all $a, b \in A$ it holds that $(a, b) \in R$ iff $(b, a) \in R$,
  - asymmetric if for all $a, b \in A$ it holds that if $(a, b) \in R$ then $(b, a) \notin R$,
  - antisymmetric if for all $a, b \in A$ with $a \neq b$ it holds that if $(a, b) \in R$ then $(b, a) \notin R$,
  - transitive if for all $a, b, c \in A$ it holds that if $(a, b) \in R$ and $(b, c) \in R$ then $(a, c) \in R$.

Special Classes of Relations

- Some important classes of relations are defined in terms of these properties.
  - Equivalence relation: reflexive, symmetric, transitive
  - Partial order: reflexive, antisymmetric, transitive
  - Strict order: irreflexive, asymmetric, transitive
  - ...
- We will consider these and other classes in detail.

Discrete Mathematics in Computer Science

B3. Equivalence and Order Relations

Malte Helmert, Gabriele Röger

University of Basel

October 6/8, 2025

B3.1 Equivalence Relations

Motivation

- Think of any attribute that two objects can have in common, e. g. their color.
- We could place the objects into distinct "buckets", e. g. one bucket for each color.
- We also can define a relation $\sim$ such that $x \sim y$ iff x and y share the attribute, e. g.have the same color.
- Would this relation be
  - reflexive?
  - irreflexive?
  - symmetric?
  - asymmetric?
  - antisymmetric?
  - transitive?

Equivalence Relation

Definition (Equivalence Relation)

A binary relation $\sim$ over set S is an equivalence relation if $\sim$ is reflexive, symmetric and transitive.

Examples:

- $\{(x, y) \mid x$ and y have the same place of origin$\}$ over the set of all Swiss citizens
- $\{(x, y) \mid x$ and y have the same parity$\}$ over $\mathbb{N}_0$
- $\{(1,1),(1,4),(1,5),(4,1),(4,4),(4,5),(5,1),(5,4),(5,5),$ $(2,2),(2,3),(3,2),(3,3)\}$ over $\{1,2,\dots,5\}$

Is this definition indeed what we want?
Does it allow us to partition the objects into buckets
(e. g. one “bucket” for all objects that share a specific color)?

German: Äquivalenzrelation

Equivalence Classes

Definition (Equivalence Class)

Let $\sim$ be an equivalence relation over set S.

For any $x \in S$, the equivalence class of x is the set

$$[x]_\sim = \{y \in S \mid x \sim y\}.$$

Consider

$\sim = \{(1,1),(1,4),(1,5),(4,1),(4,4),(4,5),(5,1),(5,4),(5,5),$
$(2,2),(2,3),(3,2),(3,3)\}$

over set $\{1,2,\dots,5\}$.

$[4]_\sim =$

German: Äquivalenzklasse

Equivalence Classes: Properties

Let $\sim$ be an equivalence relation over set S and
$E = \{[x]_\sim \mid x \in S\}$ the set of all equivalence classes.

- Every element of S is in some equivalence class in E.
- Every element of S is in at most one equivalence class in E.
  $\rightsquigarrow$ homework assignment

$\Rightarrow$ Equivalence relations induce partitions
(not covered in this course).

B3.2 Order Relations

Order Relations

- We now consider other combinations of properties, that allow us to describe a consistent order of the objects.
- "Number x is not larger than number y."
  "Set S is a subset of set T."
  "Jerry runs at least as fast as Tom."
  "Pasta tastes better than Potatoes."

German: Ordnungsrelation

Partial Orders

- We begin with partial orders.
- Example partial order relations are $\leq$ over $\mathbb{N}_0$ or $\subseteq$ for sets.
- Are these relations
  - reflexive?
  - irreflexive?
  - symmetric?
  - asymmetric?
  - antisymmetric?
  - transitive?

Partial Orders – Definition

Definition (Partial order)

A binary relation $\preceq$ over set S is a partial order if $\preceq$ is reflexive, antisymmetric and transitive.

Which of these relations are partial orders?

- strict subset relation $\subset$ for sets
- not-less-than relation $\geq$ over $\mathbb{N}_0$
- $R = \{(a, a), (a, b), (b, b), (b, c), (c, c)\}$ over $\{a, b, c\}$

German: Halbordnung oder partielle Ordnung

Least and Greatest Element

Definition (Least and greatest element)

Let $\preceq$ be a partial order over set S.

An element $x \in S$ is the least element of S if for all $y \in S$ it holds that $x \preceq y$.

It is the greatest element of S if for all $y \in S$, $y \preceq x$.

- Is there a least/greatest element? Which one?
  - $S = \{1, 2, 3\}$ and $\preceq \; = \{(x, y) \mid x, y \in S \text{ and } x \leq y\}$
  - relation $\leq$ over $\mathbb{N}_0$
  - relation $\leq$ over $\mathbb{Z}$
- Why can we say the least element instead of a least element?

German: kleinstes/grösstes Element

Uniqueness of Least Element

Theorem

Let $\preceq$ be a partial order over set S.
If S contains a least element, it contains exactly one least element.

Proof.

By contradiction: Assume x, y are least elements of S with $x \neq y$.
Since x is a least element, $x \preceq y$ is true.
Since y is a least element, $y \preceq x$ is true.
As a partial order is antisymmetric, this implies that $x = y$. $\lightning$ $\square$

Analogously: If there is a greatest element then is unique.

Minimal and Maximal Elements

Definition (Minimal/Maximal element of a set)

Let $\preceq$ be a partial order over set S.
An element $x \in S$ is a minimal element of S
if there is no $y \in S$ with $y \preceq x$ and $x \neq y$.

An element $x \in S$ is a maximal element of S
if there is no $y \in S$ with $x \preceq y$ and $x \neq y$.

A set can have several minimal elements and no least element.
Example?

German: minimales/maximales Element

Total Orders

- Relations $\leq$ over $\mathbb{N}_0$ and $\subseteq$ for sets are partial orders.
- Can we compare every object against every object?
  - For all $x, y \in \mathbb{N}_0$ it holds that $x \leq y$ or that $y \leq x$ (or both).
  - $\{1, 2\} \not\subseteq \{2, 3\}$ and $\{2, 3\} \not\subseteq \{1, 2\}$
- Relation $\leq$ is a total order, relation $\subseteq$ is not.

Total Order – Definition

Definition (Total relation)

A binary relation R over set S is total
if for all $x, y \in S$ at least one of xRy or yRx is true.

Definition (Total order)

A binary relation is a total order if it is total and a partial order.

German: totale Relation, (schwache) Totalordnung oder totale Ordnung

Strict Orders

- A partial order is reflexive, antisymmetric and transitive.
- We now consider strict orders.
- Example strict order relations are $<$ over $\mathbb{N}_0$ or $\subset$ for sets.
- Are these relations
  - reflexive?
  - irreflexive?
  - symmetric?
  - asymmetric?
  - antisymmetric?
  - transitive?

Strict Orders – Definition

Definition (Strict (partial) order)

A binary relation $\prec$ over set S is a strict (partial) order if $\prec$ is irreflexive, asymmetric and transitive.

Which of these relations are strict orders?

- subset relation $\subseteq$ for sets
- strict superset relation $\supset$ for sets

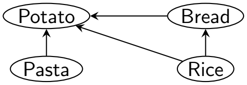
Can a relation be both, a partial order and a strict (partial) order?

We can omit irreflexivity or asymmetry from the definition (but not both). Why?

German: strenge (Halb-)ordnung

Strict Total Orders

- As partial orders, a strict order does not automatically allow us to rank arbitrary two objects against each other.
- Example 1 (personal preferences):
  - "Pasta tastes better than potato."
  - "Rice tastes better than bread."
  - "Bread tastes better than potato."
  - "Rice tastes better than potato."
  - This definition of "tastes better than" is a strict order.
  - No ranking of pasta against rice or of pasta against bread.



- Example 2: $\subset$ relation for sets
- It doesn't work to simply require that the strict order is total. Why?

Strict Total Orders – Definition

Definition (Trichotomy)

A binary relation R over set S is trichotomous if for all $x, y \in S$ exactly one of xRy, yRx or $x = y$ is true.

Definition (Strict total order)

A binary relation $\prec$ over S is a strict total order if $\prec$ is trichotomous and a strict order.

A strict total order completely ranks the elements of set S.

Example: $<$ relation over $\mathbb{N}_0$ gives the standard ordering $0, 1, 2, 3, \dots$ of natural numbers.

Attention: a non-empty strict total order is never a total order.

German: trichotom, strenge Totalordnung

Special Elements

Special elements are defined almost as for partial orders:

Definition (Least/greatest/minimal/maximal element of a set)

Let $\prec$ be a strict order over set S.

An element $x \in S$ is the least element of S
if for all $y \in S$ where $y \neq x$ it holds that $x \prec y$.

It is the greatest element of S if for all $y \in S$ where $y \neq x$, $y \prec x$.

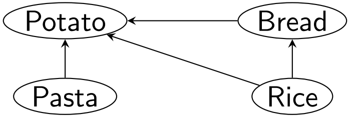
Element $x \in S$ is a minimal element of S
if there is no $y \in S$ with $y \prec x$.

It is a maximal element of S
if there is no $y \in S$ with $x \prec y$.

Special Elements – Example

Consider again the previous example:

$S = \{\text{Pasta}, \text{Potato}, \text{Bread}, \text{Rice}\}$
$\prec\ = \{(\text{Pasta}, \text{Potato}), (\text{Bread}, \text{Potato}),$
$(\text{Rice}, \text{Potato}), (\text{Rice}, \text{Bread})\}$



Is there a least and a greatest element?
Which elements are maximal or minimal?

Summary

- An equivalence relation is reflexive, symmetric and transitive.
- A partial order $x \preceq y$ is reflexive, antisymmetric and transitive.
  - If x is the greatest element of a set S, it is greater than every element: for all $y \in S$ it holds that $y \preceq x$.
  - If x is a maximal element of set S then it is not smaller than any other element y: there is no $y \in S$ with $x \preceq y$ and $x \neq y$.
  - A total order is a partial order without incomparable objects.
- A strict order is irreflexive, asymmetric and transitive.
  - Strict total orders and special elements are analogously defined as for partial orders but with a special treatment of equal elements.

Discrete Mathematics in Computer Science

B4. Operations on Relations

Malte Helmert, Gabriele Röger

University of Basel

October 13, 2025

B4.1 Operations on Relations

B4.1 Operations on Relations

Relations: Recap

- A relation over sets $S_1, \dots, S_n$ is a set $R \subseteq S_1 \times \dots \times S_n$.
- A binary relation is a relation over two sets.
- A homogeneous relation R over set S is a binary relation $R \subseteq S \times S$.

Set Operations

- Relations are sets of tuples, so we can build their union, intersection, complement, ...
- Let R be a relation over $S_1, \dots, S_n$ and R' a relation over $S'_1, \dots, S'_n$. Then $R \cup R'$ is a relation over $S_1 \cup S'_1, \dots, S_n \cup S'_n$.
  With the standard relations $<, =$ and $\leq$ for $\mathbb{N}_0$,
  relation $\leq$ corresponds to the union of relations $<$ and $=$.
- Let R and R' be relations over n sets.
  Then $R \cap R'$ is a relation.
  Over which sets?
  With the standard relations $\leq, =$ and $\geq$ for $\mathbb{N}_0$,
  relation $=$ corresponds to the intersection of $\leq$ and $\geq$.
- If R is a relation over $S_1, \dots, S_n$
  then so is the complementary relation $\bar{R} = (S_1 \times \dots \times S_n) \setminus R$.
  With the standard relations for $\mathbb{N}_0$, relation $=$ is the complementary relation of $\neq$ and $>$ the one of $\leq$.

Inverse of a Relation

Definition

Let $R \subseteq A \times B$ be a binary relation over A and B.

The inverse relation of R is the relation $R^{-1} \subseteq B \times A$ given by $R^{-1} = \{(b, a) \mid (a, b) \in R\}$.

- The inverse of the $<$ relation over $\mathbb{N}_0$ is the $>$ relation.
- Relation R with xRy iff person x has a key for y.
  Inverse: Q with aQb iff lock a can be openened by person b.

German: inverse Relation oder Umkehrrelation

Composition of Relations

Definition (Composition of relations)

Let R_1 be a relation over A and B and R_2 a relation over B and C.

The composition of R_1 and R_2 is the relation $R_2 \circ R_1$ over A and C with:

$$\begin{aligned} R_2 \circ R_1 = \{(a, c) \mid \ & \textit{there is a } b \in B \text{ with} \\ & (a, b) \in R_1 \text{ and } (b, c) \in R_2\} \end{aligned}$$

How can we illustrate this graphically?

German: Komposition oder Rückwärtsverkettung

Composition of Relations: Example

$S_1 = \{1, 2, 3, 4\}$
$S_2 = \{A, B, C, D, E\}$
$S_3 = \{a, b, c, d\}$

$R_1 = \{(1, A), (1, B), (3, B), (4, D)\}$ over S_1 and S_2

$R_2 = \{(B, a), (C, c), (D, a), (D, d)\}$ over S_2 and S_3

$R_2 \circ R_1 =$

Composition is Associative

Theorem (Associativity of composition)

Let $S_1, \dots, S_4$ be sets and R_1, R_2, R_3 relations with $R_i \subseteq S_i \times S_{i+1}$. Then

$$R_3 \circ (R_2 \circ R_1) = (R_3 \circ R_2) \circ R_1.$$

Proof.

It holds that $(x_1, x_4) \in R_3 \circ (R_2 \circ R_1)$ iff there is an x_3 with $(x_1, x_3) \in R_2 \circ R_1$ and $(x_3, x_4) \in R_3$.

As $(x_1, x_3) \in R_2 \circ R_1$ iff there is an x_2 with $(x_1, x_2) \in R_1$ and $(x_2, x_3) \in R_2$, we have overall that $(x_1, x_4) \in R_3 \circ (R_2 \circ R_1)$ iff there are x_2, x_3 with $(x_1, x_2) \in R_1$, $(x_2, x_3) \in R_2$ and $(x_3, x_4) \in R_3$.

This is the case iff there is an x_2 with $(x_1, x_2) \in R_1$ and $(x_2, x_4) \in R_3 \circ R_2$, which holds iff $(x_1, x_4) \in (R_3 \circ R_2) \circ R_1$. □

(Reflexive) Transitive Closure

Definition ((Reflexive) transitive closure)

Let R be a relation over set S.

The transitive closure R^+ of R is the smallest relation over S that is transitive and has R as a subset.

The reflexive transitive closure R^* of R is the smallest relation over S that is reflexive, transitive and has R as a subset.

The (reflexive) transitive closure always exists. Why?

Example: If aRb specifies that there is a direct flight from a to b, what do R^+ and R^* express?

German: (reflexive) transitive Hülle

Transitive Closure and n-fold Composition

Define the n-fold composition of a relation R over S as

$$\begin{aligned} R_0 &= \{(x,x) \mid x \in S\} && \text{and} \\ R_i &= R \circ R_{i-1} && \text{for } i \geq 1. \end{aligned}$$

Theorem

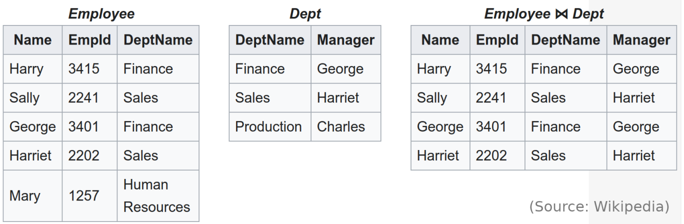
Let R be a relation over set S.
Then $R^+ = \bigcup_{i=1}^{\infty} R_i$ and $R^* = \bigcup_{i=0}^{\infty} R_i$.

Without proof.

German: n-fache Komposition

Other Operators

- There are many more operators, also for general relations.
- Highly relevant for queries over relational databases.
- For example, join operators combine relations based on common entries.
- Example for a natural join:



Employee

Name	EmpId	DeptName
Harry	3415	Finance
Sally	2241	Sales
George	3401	Finance
Harriet	2202	Sales
Mary	1257	Human Resources

Dept

DeptName	Manager
Finance	George
Sales	Harriet
Production	Charles

Employee ⋈ Dept

Name	EmpId	DeptName	Manager
Harry	3415	Finance	George
Sally	2241	Sales	Harriet
George	3401	Finance	George
Harriet	2202	Sales	Harriet

(Source: Wikipedia)

Summary

- Relations: general, binary, homogeneous
- Properties: reflexivity, symmetry, transitivity
  (and related properties)
- Special relations: equivalence relations, order relations
- Operations: inverse, composition, transitive closure

Discrete Mathematics in Computer Science

B5. Functions

Malte Helmert, Gabriele Röger

University of Basel

October 15/20, 2025

B5.1 Partial and Total Functions

Important Building Blocks of Discrete Mathematics

Important building blocks:

- sets
- relations
- functions

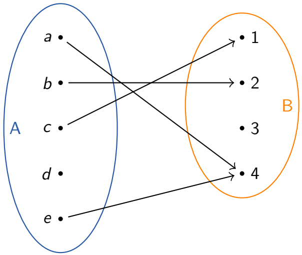
In principle, functions are just a special kind of relations:

- $f : \mathbb{N}_0 \to \mathbb{N}_0$ with $f(x) = x^2$
- relation R over $\mathbb{N}_0$ with $R = \{(x, x^2) \mid x \in \mathbb{N}_0\}$.

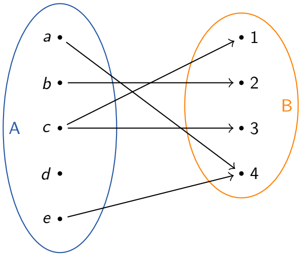
Functional Relations

Definition

A binary relation R over sets A and B is functional if for every $a \in A$ there is at most one $b \in B$ with $(a, b) \in R$.



functional



not functional

Functions – Examples

- $f : \mathbb{N}_0 \to \mathbb{N}_0$ with $f(x) = x^2 + 1$
- $abs : \mathbb{Z} \to \mathbb{N}_0$ with
$$abs(x) = \begin{cases} x & \text{if } x \geq 0 \\ -x & \text{otherwise} \end{cases}$$
- $distance : \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}$ with
$distance((x_1, y_1), (x_2, y_2)) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$

Partial Function – Example

Partial function $r : \mathbb{Z} \times \mathbb{Z} \nrightarrow \mathbb{Q}$ with

$$r(n, d) = \begin{cases} \frac{n}{d} & \text{if } d \neq 0 \\ \text{undefined} & \text{otherwise} \end{cases}$$

Partial Functions

Definition (Partial function)

A partial function f from set A to set B (written $f : A \nrightarrow B$) is given by a functional relation G over A and B.

Relation G is called the graph of f.

We write $f(x) = y$ for $(x, y) \in G$ and say y is the image of x under f.

If there is no $y \in B$ with $(x, y) \in G$, then $f(x)$ is undefined.

Partial function $r : \mathbb{Z} \times \mathbb{Z} \nrightarrow \mathbb{Q}$ with

$$r(n, d) = \begin{cases} \frac{n}{d} & \text{if } d \neq 0 \\ \text{undefined} & \text{otherwise} \end{cases}$$

has graph $\{((n, d), \frac{n}{d}) \mid n \in \mathbb{Z}, d \in \mathbb{Z} \setminus \{0\}\} \subseteq \mathbb{Z}^2 \times \mathbb{Q}$.

Domain (of Definition), Codomain, Image

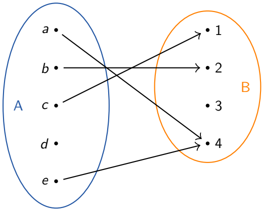
Definition (Domain of definition, codomain, image)

Let $f : A \nrightarrow B$ be a partial function.

Set A is called the domain of f, set B is its codomain.

The domain of definition of f is the set
$\mathrm{dom}(f) = \{x \in A \mid \text{there is a } y \in B \text{ with } f(x) = y\}$.

The image (or range) of f is the set
$\mathrm{img}(f) = \{y \mid \text{there is an } x \in A \text{ with } f(x) = y\}$.



$f : \{a, b, c, d, e\} \nrightarrow \{1, 2, 3, 4\}$
$f(a) = 4, f(b) = 2, f(c) = 1, f(e) = 4$
domain $\{a, b, c, d, e\}$
codomain $\{1, 2, 3, 4\}$
domain of definition $\mathrm{dom}(f) = \{a, b, c, e\}$
image $\mathrm{img}(f) = \{1, 2, 4\}$

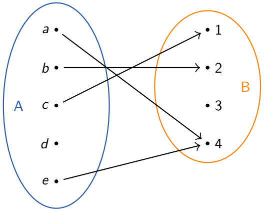
Preimage

The preimage contains all elements of the domain that are mapped to given elements of the codomain.

Definition (Preimage)

Let $f : A \nrightarrow B$ be a partial function and let $Y \subseteq B$.

The preimage of Y under f is the set $f^{-1}[Y] = \{x \in A \mid f(x) \in Y\}$.



$f^{-1}[\{1\}] =$

$f^{-1}[\{3\}] =$

$f^{-1}[\{4\}] =$

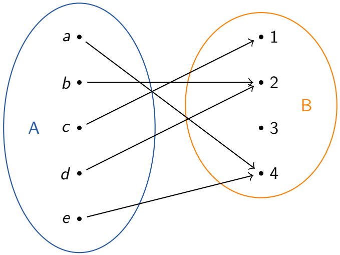
$f^{-1}[\{1, 2\}] =$

Total Functions

Definition (Total function)

A (total) function $f : A \to B$ from set A to set B is a partial function from A to B such that $f(x)$ is defined for all $x \in A$.

$\to$ no difference between the domain and the domain of definition



Specifying a Function

Some common ways of specifying a function:

- Listing the mapping explicitly, e. g.
  $f(a) = 4, f(b) = 2, f(c) = 1, f(e) = 4$ or
  $f = \{a \mapsto 4, b \mapsto 2, c \mapsto 1, e \mapsto 4\}$
- By a formula, e. g. $f(x) = x^2 + 1$
- By recurrence, e. g.
  $0! = 1$ and
  $n! = n(n-1)!$ for $n > 0$
- In terms of other functions, e. g. inverse, composition

Relationship to Functions in Programming

```
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n-1)
```

→ Relationship between recursion and recurrence

Relationship to Functions in Programming

```
def foo(n):
    value = ...
    while <some condition>:
        ...
        value = ...
    return value
```

→ Does possibly not terminate on all inputs.
→ Value is undefined for such inputs.
→ Theoretical computer science: partial function

Relationship to Functions in Programming

```
import random
counter = 0

def bar(n):
    print("Hi! I got input", n)
    global counter
    counter += 1
    return random.choice([1,2,n])
```

→ Functions in programming don't always compute mathematical functions (except *purely functional languages*).

→ In addition, not all mathematical functions are computable.

B5.2 Operations on Partial Functions

Restrictions and Extensions

Definition (Restriction and extension)

Let $f : A \nrightarrow B$ be a partial function and let $X \subseteq A$.
The restriction of f to X is the partial function $f|_X : X \nrightarrow B$ with $f|_X(x) = f(x)$ for all $x \in X$.

A function $f' : A' \nrightarrow B$ is called an extension of f if $A \subseteq A'$ and $f'|_A = f$.

The restriction of f to its domain of definition is a total function.

What's the graph of the restriction?

What's the restriction of f to its domain?

Function Composition

Definition (Composition of partial functions)

Let $f : A \nrightarrow B$ and $g : B \nrightarrow C$ be partial functions.

The composition of f and g is $g \circ f : A \nrightarrow C$ with

$$(g \circ f)(x) = \begin{cases} g(f(x)) & \text{if } f \text{ is defined for } x \text{ and} \\ & g \text{ is defined for } f(x) \\ \text{undefined} & \text{otherwise} \end{cases}$$

Corresponds to relation composition of the graphs.

If f and g are functions, their composition is a function.

Example:

$$f : \mathbb{N}_0 \to \mathbb{N}_0 \quad \text{with } f(x) = x^2$$
$$g : \mathbb{N}_0 \to \mathbb{N}_0 \quad \text{with } g(x) = x + 3$$
$$(g \circ f)(x) =$$

Properties of Function Composition

Function composition is

- not commutative:
  - $f : \mathbb{N}_0 \to \mathbb{N}_0$ with $f(x) = x^2$
  - $g : \mathbb{N}_0 \to \mathbb{N}_0$ with $g(x) = x + 3$
  - $(g \circ f)(x) = x^2 + 3$
  - $(f \circ g)(x) = (x + 3)^2$
- associative, i. e. $h \circ (g \circ f) = (h \circ g) \circ f$
  $\rightarrow$ analogous to associativity of relation composition

Function Composition in Programming

We implicitly compose functions all the time. . .

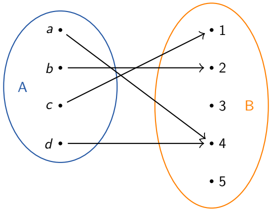
```
def foo(n):
    ...
    x = somefunction(n)
    y = someotherfunction(x)
    ...
```

Many languages also allow explicit composition of functions, e. g. in Haskell:

```
incr x = x + 1
square x = x * x
squareplusone = incr . square
```

B5.3 Properties of Functions

Properties of Functions



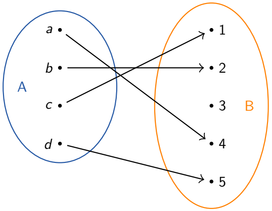
- Partial functions map every element of their domain to at most one element of their codomain, total functions map it to exactly one such value.
- Different elements of the domain can have the same image.
- There can be values of the codomain that aren’t the image of any element of the domain.
- We often want to exclude such cases
  $\rightarrow$ define additional properties to say this quickly

Injective Functions

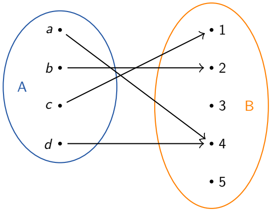
An injective function maps distinct elements of its domain to distinct elements of its co-domain.

Definition (Injective function)

A function $f : A \to B$ is injective (also one-to-one or an injection) if for all $x, y \in A$ with $x \neq y$ it holds that $f(x) \neq f(y)$.



injective



not injective

Injective Functions – Examples

Which of these functions are injective?

- $f : \mathbb{Z} \to \mathbb{N}_0$ with $f(x) = |x|$
- $g : \mathbb{N}_0 \to \mathbb{N}_0$ with $g(x) = x^2$
- $h : \mathbb{N}_0 \to \mathbb{N}_0$ with $h(x) = \begin{cases} x - 1 & \text{if } x \text{ is odd} \\ x + 1 & \text{if } x \text{ is even} \end{cases}$

Composition of Injective Functions

Theorem

If $f : A \to B$ and $g : B \to C$ are injective functions then also $g \circ f$ is injective.

Proof.

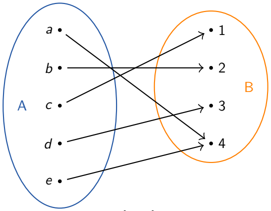
Consider arbitrary elements $x, y \in A$ with $x \neq y$.
Since f is injective, we know that $f(x) \neq f(y)$.
As g is injective, this implies that $g(f(x)) \neq g(f(y))$.
With the definition of $g \circ f$, we conclude that
$(g \circ f)(x) \neq (g \circ f)(y)$.
Overall, this shows that $g \circ f$ is injective. □

Surjective Functions

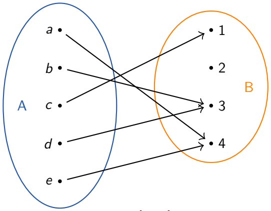
A surjective function maps at least one elements to every element of its co-domain.

Definition (Surjective function)

A function $f : A \to B$ is surjective (also onto or a surjection) if its image is equal to its codomain, i. e. for all $y \in B$ there is an $x \in A$ with $f(x) = y$.



surjective



not surjective

Surjective Functions – Examples

Which of these functions are surjective?

- $f : \mathbb{Z} \to \mathbb{N}_0$ with $f(x) = |x|$
- $g : \mathbb{N}_0 \to \mathbb{N}_0$ with $g(x) = x^2$
- $h : \mathbb{N}_0 \to \mathbb{N}_0$ with $h(x) = \begin{cases} x - 1 & \text{if } x \text{ is odd} \\ x + 1 & \text{if } x \text{ is even} \end{cases}$

Composition of Surjective Functions

Theorem

If $f : A \to B$ and $g : B \to C$ are surjective functions then also $g \circ f$ is surjective.

Proof.

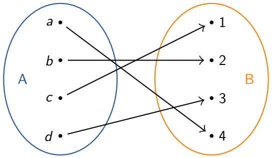
Consider an arbitary element $z \in C$.
Since g is surjective, there is a $y \in B$ with $g(y) = z$.
As f is surjective, for such a y there is an $x \in A$ with $f(x) = y$ and thus $g(f(x)) = z$.
Overall, for every $z \in C$ there is an $x \in A$ with $(g \circ f)(x) = g(f(x)) = z$, so $g \circ f$ is surjective. □

Bijective Functions

A bijective function pairs every element of its domain with exactly one element of its codomain and every element of the codomain is paired with exactly one element of the domain.

Definition (Bijective function)

A function is bijective (also a one-to-one correspondence or a bijection) if it is injective and surjective.



bijection

Corollary

The composition of two bijective functions is bijective.

Bijective Functions – Examples

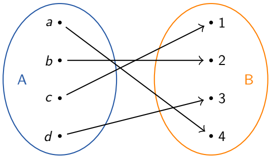
Which of these functions are bijective?

- $f : \mathbb{Z} \to \mathbb{N}_0$ with $f(x) = |x|$
- $g : \mathbb{N}_0 \to \mathbb{N}_0$ with $g(x) = x^2$
- $h : \mathbb{N}_0 \to \mathbb{N}_0$ with $h(x) = \begin{cases} x - 1 & \text{if } x \text{ is odd} \\ x + 1 & \text{if } x \text{ is even} \end{cases}$

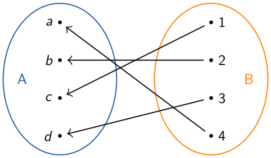
Inverse Function

Definition

Let $f : A \to B$ be a bijection.
The inverse function of f is the function $f^{-1} : B \to A$ with $f^{-1}(y) = x$ iff $f(x) = y$.



f



f^{-1}

Inverse Function and Composition

Theorem

Let $f : A \to B$ be a bijection.

1. *For all $x \in A$ it holds that $f^{-1}(f(x)) = x$.*
2. *For all $y \in B$ it holds that $f(f^{-1}(y)) = y$.*
3. *f^{-1} is a bijection from B to A.*
4. *$(f^{-1})^{-1} = f$*

Proof sketch.

1. For $x \in A$ let $y = f(x)$. Then $f^{-1}(f(x)) = f^{-1}(y) = x$
2. For $y \in B$ there is exactly one x with $y = f(x)$. With this x it holds that $f^{-1}(y) = x$ and overall $f(f^{-1}(y)) = f(x) = y$.
3. Surjective: for all $x \in A$, f^{-1} maps $f(x)$ to x (cf. (1)).
   Injective: if $f^{-1}(y) = f^{-1}(y')$ then $f(f^{-1}(y)) = f(f^{-1}(y'))$, so with (2) we have $y = y'$.
4. Def. of inverse: $(f^{-1})^{-1}(x) = y$ iff $f^{-1}(y) = x$ iff $f(x) = y$.

Inverse Function

Theorem

Let $f : A \to B$ and $g : B \to C$ be bijections.
Then $(g \circ f)^{-1} = f^{-1} \circ g^{-1}$.

Proof.

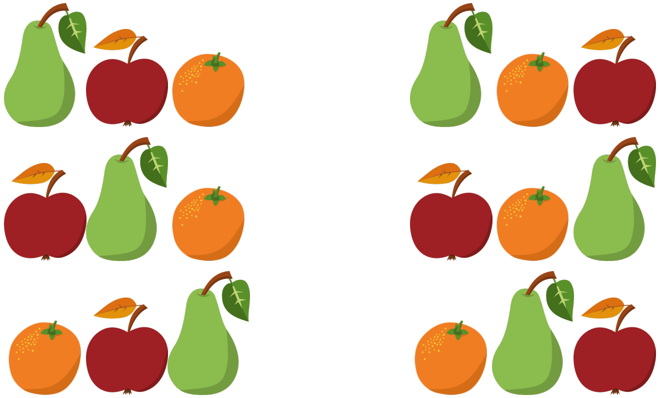
We need to show that for all $x \in C$ it holds that
$(g \circ f)^{-1}(x) = (f^{-1} \circ g^{-1})(x)$.

Consider an arbitrary $x \in C$ and let $y = (g \circ f)^{-1}(x)$.
By the definition of the inverse $(g \circ f)(y) = g(f(y)) = x$.

Let $z = f(y)$.
From $x = g(f(y))$, we know that $x = g(z)$ and thus $g^{-1}(x) = z$.
From $z = f(y)$ we get $f^{-1}(z) = y$.

This gives $(f^{-1} \circ g^{-1})(x) = f^{-1}(g^{-1}(x)) = f^{-1}(z) = y$. □

Permutations



Permutation – Definition

Definition (Permutation)

Let S be a set. A bijection $\pi : S \to S$ is called a permutation of S.

How many permutations are there for a finite set S?

Permutations of the same set S can be composed with function composition. The result is again a permutation of S. Why?

The inverse of a permutation is again a permutation.

Permutations as Functions on Positions

- A permutation can be used to describe the rearrangement of objects.
- Consider for example sequence o_2, o_1, o_3, o_4
- Let's rearrange the objects, e. g. to o_3, o_1, o_4, o_2.
  - The object at position 1 was moved to position 4,
  - the one from position 3 to position 1,
  - the one from position 4 to position 3 and
  - the one at position 2 stayed where it was.
- This corresponds to the permutation $\sigma : \{1, 2, 3, 4\} \to \{1, 2, 3, 4\}$ with $\sigma(1) = 4$, $\sigma(2) = 2$, $\sigma(3) = 1$, $\sigma(4) = 3$

Permutation: Example I

Determine the arrangement of some objects after applying a permutation that operates on the locations.

🍐🍎🍊 and π permutation of $\{1, 2, 3\}$.

Define f with $f($🍐$) = 1$, $f($🍎$) = 2$, $f($🍊$) = 3$
to describe the initial configuration.

Then $\pi \circ f$ describes the resulting configuration.

Permutation: Example II

Describe what fruit is moved to the place of what fruit, independent of the positions.

Swap the 🍐 and the 🍎 with permutation f of {🍐, 🍎, 🍊} with $f($🍐$) =$ 🍎, $f($🍎$) =$ 🍐, $f($🍊$) =$ 🍊.

If g maps locations to fruits then $f^{-1} \circ g$ describes the mapping from locations to fruits after the swap.

For example $g(1) =$ 🍐, $g(2) =$ 🍎, $g(3) =$ 🍊 for 🍐🍎🍊.

Then $(f^{-1} \circ g)(1) =$ 🍎, $(f^{-1} \circ g)(2) =$ 🍐, $(f^{-1} \circ g)(3) =$ 🍊 representing 🍎🍐🍊.

Permutation: Example III

Determine the permutation of locations that leads from one configuration to the other.

$$🍐🍎🍊 \Rightarrow 🍎🍐🍊.$$

Define f with $f(🍐) = 1$, $f(🍎) = 2$, $f(🍊) = 3$
to describe the initial configuration and
function g with $g(🍐) = 2$, $g(🍎) = 1$, $g(🍊) = 3$
for the final configuration.

Then $g \circ f^{-1}$ describes the permutation of locations.

Summary

- injective function: maps distinct elements of its domain to distinct elements of its co-domain.
- surjective function: maps at least one element to every element of its co-domain.
- bijective function: injective and surjective
  $\rightarrow$ one-to-one correspondence
- Bijective functions are invertible. The inverse function of f maps the image of x under f to x.
- Permutations are bijections from a set to itself. They can be used to describe rearrangements of objects.

Discrete Mathematics in Computer Science

B6. Sets: Comparing Cardinality and Hilbert's Hotel

Malte Helmert, Gabriele Röger

University of Basel

October 22, 2025

B6.1 Comparing Cardinality

Finite Sets Revisited

We already know:

- The cardinality $|S|$ measures the size of set S.
- A set is finite if it has a finite number of elements.
- The cardinality of a finite set
  is the number of elements it contains.

A set is infinite if it has an infinite number of elements.

Do all infinite sets have the same cardinality?

Comparing the Cardinality of Sets

- Consider $A = \{1, 2\}$ and $B = \{\text{dog}, \text{cat}, \text{mouse}\}$.
- We can map distinct elements of A to distinct elements of B, e.g.

$$1 \mapsto \text{dog}$$
$$2 \mapsto \text{cat}$$

- This is an injective function from A to B:
  - every element of A is mapped to an element of B;
  - different elements of A are mapped to different elements of B.

Comparing Cardinality

Definition (cardinality not larger)

Set A has cardinality less than or equal to the cardinality of set B ($|A| \leq |B|$), if there is an injective function from A to B.

Comparing the Cardinality of Sets

- $A = \{1, 2, 3\}$ and $B = \{\text{dog}, \text{cat}, \text{mouse}\}$ have cardinality 3.
- We can pair their elements by a bijection from A to B:

$$1 \leftrightarrow \text{dog}$$
$$2 \leftrightarrow \text{cat}$$
$$3 \leftrightarrow \text{mouse}$$

- This is a bijection from A to B.
  - Each element of A is paired with exactly one element of set B.
  - Each element of B is paired with exactly one element of A.
- If there is a bijection from A to B there is one from B to A (the inverse function).

Equinumerous Sets

We use the existence of a bijection also as criterion for infinite sets:

Definition (equinumerous sets)

Two sets A and B have the same cardinality ($|A| = |B|$) if there exists a bijection from A to B.

Such sets are called equinumerous.

Definition (strictly smaller cardinality)

Set A has cardinality strictly less than the cardinality of set B ($|A| < |B|$), if $|A| \leq |B|$ and $|A| \neq |B|$.

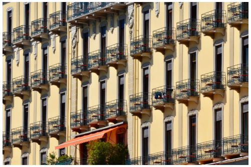
Consider set A and object $e \notin A$. Is $|A| < |A \cup \{e\}|$?

B6.2 Hilbert's Hotel

Hilbert's Hotel

Our intuition for finite sets does not always work for infinite sets.

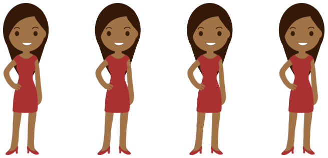
- If in a hotel all rooms are occupied then it cannot accomodate additional guests.
- But Hilbert's Grand Hotel has infinitely many rooms.
- All these rooms are occupied.



One More Guest Arrives

- Every guest moves from her current room n to room $n + 1$.
- Room 1 is then free.
- The new guest gets room 1.

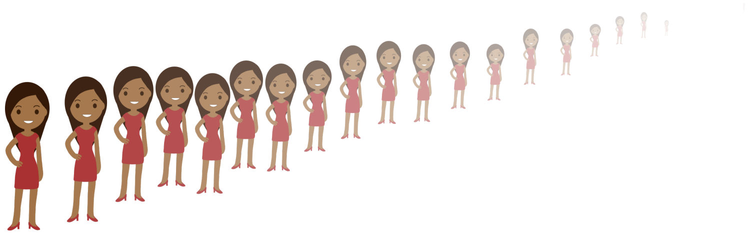
Four More Guests Arrive



- Every guest moves from her current room n to room $n + 4$.
- Rooms 1 to 4 are no longer occupied and can be used for the new guests.

$\rightarrow$ Works for any finite number of additional guests.

An Infinite Number of Guests Arrives



- Every guest moves from her current room n to room $2n$.
- The infinitely many rooms with odd numbers are now available.
- The new guests fit into these rooms.

Can we Go further?

What if ...

- infinitely many coaches, each with an infinite number of guests
- infinitely many ferries, each with an infinite number of coaches, each with infinitely many guests
- ...

... arrive?

There are strategies for all these situations
as long as with "infinite" we mean "countably infinite"
and there is a finite number of layers.

Summary

- Set A has cardinality less than or equal the cardinality of set B ($|A| \leq |B|$), if there is an injective function from A to B.
- Sets A and B have the same cardinality ($|A| = |B|$) if there exists a bijection from A to B.
- Our intuition for finite sets does not always work for infinite sets.

Discrete Mathematics in Computer Science

B7. Sets: Countability

Malte Helmert, Gabriele Röger

University of Basel

October 27, 2025

B7.1 Countable Sets

B7.1 Countable Sets

Comparing Cardinality

- Two sets A and B have the same cardinality if their elements can be paired (i.e. there is a bijection from A to B).
- Set A has a strictly smaller cardinality than set B if
  - we can map distinct elements of A to distinct elements of B (i.e. there is an injective function from A to B), and
  - $|A| \neq |B|$.

- This clearly makes sense for finite sets.
- What about infinite sets? Do they even have different cardinalities?

Countable and Countably Infinite Sets

Definition (countably infinite and countable)

A set A is countably infinite if $|A| = |\mathbb{N}_0|$.

A set A is countable if $|A| \leq |\mathbb{N}_0|$.

A set is countable if it is finite or countably infinite.

- We can count the elements of a countable set one at a time.
- The objects are "discrete" (in contrast to "continuous").
- Discrete mathematics deals with all kinds of countable sets.

Set of Even Numbers

- $\mathit{even} = \{n \mid n \in \mathbb{N}_0 \text{ and } n \text{ is even}\}$
- Obviously: $\mathit{even} \subset \mathbb{N}_0$
- Intuitively, there are twice as many natural numbers as even numbers — no?
- Is $|\mathit{even}| < |\mathbb{N}_0|$?

Set of Even Numbers

Theorem (set of even numbers is countably infinite)

The set of all even natural numbers is countably infinite,
i. e. $|\{n \mid n \in \mathbb{N}_0 \text{ and } n \text{ is even}\}| = |\mathbb{N}_0|$.

Proof Sketch.

We can pair every even number $2n$ with natural number n. □

Set of Perfect Squares

Theorem (set of perfect squares is countably infinite)

The set of all perfect squares is countably infinite,
i. e. $|\{n^2 \mid n \in \mathbb{N}_0\}| = |\mathbb{N}_0|$.

Proof Sketch.

We can pair every square number n^2 with natural number n. □

Subsets of Countable Sets are Countable

In general:

Theorem (subsets of countable sets are countable)

Let A be a countable set. Every set B with $B \subseteq A$ is countable.

Proof.

Since A is countable there is an injective function f from A to $\mathbb{N}_0$.
The restriction of f to B is an injective function from B to $\mathbb{N}_0$. □

Set of the Positive Rationals

Theorem (set of positive rationals is countably infinite)

Set $\mathbb{Q}_+ = \{n \mid n \in \mathbb{Q} \text{ and } n > 0\} = \{p/q \mid p, q \in \mathbb{N}_1\}$ *is countably infinite.*

Proof idea.

$$\begin{tikzcd}
\frac{1}{1}\ (0) \arrow[r] & \frac{1}{2}\ (1) \arrow[dl] & \frac{1}{3}\ (4) \arrow[r] & \frac{1}{4}\ (5) \arrow[dl] & \frac{1}{5}\ (10) \arrow[r] & {} \\
\frac{2}{1}\ (2) \arrow[d] & \frac{2}{2}\ (\cdot) \arrow[ur] & \frac{2}{3}\ (6) \arrow[dl] & \frac{2}{4}\ (\cdot) \arrow[ur] & \frac{2}{5} & \cdots \\
\frac{3}{1}\ (3) \arrow[ur] & \frac{3}{2}\ (7) \arrow[dl] & \frac{3}{3}\ (\cdot) \arrow[ur] & \frac{3}{4} & \frac{3}{5} & \cdots \\
\frac{4}{1}\ (8) \arrow[d] & \frac{4}{2}\ (\cdot) \arrow[ur] & \frac{4}{3} & \frac{4}{4} & \frac{4}{5} & \cdots \\
\frac{5}{1}\ (9) \arrow[ur] & \frac{5}{2} & \frac{5}{3} & \frac{5}{4} & \frac{5}{5} & \cdots \\
\vdots & \vdots & \vdots & \vdots & \vdots &
\end{tikzcd}$$

$\square$

Union of Two Countable Sets is Countable

Theorem (union of two countable sets countable)

Let A and B be countable sets. Then $A \cup B$ is countable.

Proof sketch.

As A and B are countable there is an injective function f_A from A to $\mathbb{N}_0$, analogously f_B from B to $\mathbb{N}_0$.

We define function $f_{A \cup B}$ from $A \cup B$ to $\mathbb{N}_0$ as

$$f_{A \cup B}(e) = \begin{cases} 2f_A(e) & \text{if } e \in A \\ 2f_B(e) + 1 & \text{otherwise} \end{cases}$$

This $f_{A \cup B}$ is an injective function from $A \cup B$ to $\mathbb{N}_0$. □

Integers and Rationals

Theorem (sets of integers and rationals are countably infinite)

The sets $\mathbb{Z}$ and $\mathbb{Q}$ are countably infinite.

Without proof ($\rightsquigarrow$ exercises)

Union of More than Two Sets

Definition (arbitrary unions)

Let M be a set of sets. The union $\bigcup_{S \in M} S$ is the set with

$$x \in \bigcup_{S \in M} S \text{ iff exists } S \in M \text{ with } x \in S.$$

Countable Union of Countable Sets

Theorem

Let M be a countable set of countable sets.

Then $\bigcup_{S \in M} S$ is countable.

Proof sketch.

With $M = \{S_1, S_2, S_3, \dots\}$ (possibly finite) and each $S_i = \{x_{i1}, x_{i2}, \dots\}$ (possibly finite), we can use an analogous idea as for the countability of $\mathbb{Q}_+$ (skipping duplicates):

$$\begin{tikzcd}
S_1 : & x_{11} \arrow[r] & x_{12} \arrow[dl] & x_{13} \arrow[r] & x_{14} \arrow[dl] & x_{15} \arrow[r] & {} \\
S_2 : & x_{21} \arrow[d] & x_{22} \arrow[ur] & x_{23} \arrow[dl] & x_{24} \arrow[ur] & x_{25} & \cdots \\
S_3 : & x_{31} \arrow[ur] & x_{32} \arrow[dl] & x_{33} \arrow[ur] & x_{34} & x_{35} & \cdots \\
S_4 : & x_{41} \arrow[d] & x_{42} \arrow[ur] & x_{43} & x_{44} & x_{45} & \cdots \\
S_5 : & x_{51} \arrow[ur] & x_{52} & x_{53} & x_{54} & x_{55} & \cdots \\
 & \vdots & \vdots & \vdots & \vdots & \vdots &
\end{tikzcd}$$

$\square$

Set of all Binary Trees is Countable

Theorem (set of all binary trees is countable)

The set $B = \{b \mid b \text{ is a binary tree}\}$ is countable.

Proof.

For $n \in \mathbb{N}_0$ the set B_n of all binary trees with n leaves is finite.

With $M = \{B_i \mid i \in \mathbb{N}_0\}$ the set of all binary trees is $B = \bigcup_{B' \in M} B'$.

Since M is a countable set of countable sets, B is countable. □

And Now?

We have seen several countably infinite sets.

What about our original questions?

- Do all infinite sets have the same cardinality?
- Are they all countably infinite?

Summary

- A set is countable if it has at most cardinality $|\mathbb{N}_0|$.
- If a set is countable and infinite, it is countably infinite.
- Sets $\mathbb{Z}$ and $\mathbb{Q}$ are countably infinite.
- Every subset of a countable set is countable.
- Every countable union of countable sets is countable, in particular, the union of two countable sets is countable.

Discrete Mathematics in Computer Science

B8. Cantor's Theorem

Malte Helmert, Gabriele Röger

University of Basel

October 29, 2025

Discrete Mathematics in Computer Science

October 29, 2025 — B8. Cantor's Theorem

Reminder: Cardinality of the Power Set

Theorem

Let S be a finite set. Then $|\mathcal{P}(S)| = 2^{|S|}$.

B8.1 Cantor's Theorem

Countable Sets

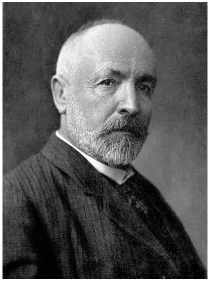
We already know:

- Sets with the same cardinality as $\mathbb{N}_0$ are called countably infinite.
- A countable set is finite or countably infinite.
- Every subset of a countable set is countable.
- The union of countably many countable sets is countable.

Open questions (to be resolved today):

- Do all infinite sets have the same cardinality?
- Does the power set of an infinite set S
  have the same cardinality as S?

Georg Cantor



- German mathematician (1845–1918)
- Proved that the rational numbers are countable.
- Proved that the real numbers are not countable.
- Cantor's Theorem: For every set S it holds that $|S| < |\mathcal{P}(S)|$.

Our Plan

- Understand Cantor's theorem
- Understand an important theoretical implication for computer science

Cantor's Diagonal Argument Illustrated on a Finite Set

$S = \{a, b, c\}$.

Consider an arbitrary function from S to $\mathcal{P}(S)$.
For example:

	a	b	c	
a	1	0	1	a mapped to $\{a, c\}$
b	1	1	0	b mapped to $\{a, b\}$
c	0	1	0	c mapped to $\{b\}$
	0	0	1	nothing was mapped to $\{c\}$.

We can identify an "unused" element of $\mathcal{P}(S)$.
Complement the entries on the main diagonal.

Works with every function from S to $\mathcal{P}(S)$.
$\rightarrow$ there cannot be a surjective function from S to $\mathcal{P}(S)$.
$\rightarrow$ there cannot be a bijection from S to $\mathcal{P}(S)$.

Cantor's Diagonal Argument on a Countably Infinite Set

$S = \mathbb{N}_0$.

Consider an arbitrary function from $\mathbb{N}_0$ to $\mathcal{P}(\mathbb{N}_0)$.
For example:

$$
\begin{array}{c|cccccc}
 & 0 & 1 & 2 & 3 & 4 & \dots \\
0 & 1 & 0 & 1 & 0 & 1 & \dots \\
1 & 1 & 1 & 0 & 1 & 0 & \dots \\
2 & 0 & 1 & 0 & 1 & 0 & \dots \\
3 & 1 & 1 & 0 & 0 & 0 & \dots \\
4 & 1 & 1 & 0 & 1 & 1 & \dots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \\
\hline
 & 0 & 0 & 1 & 1 & 0 & \dots
\end{array}
$$

Complementing the entries on the main diagonal
again results in an "unused" element of $\mathcal{P}(\mathbb{N}_0)$.

Cantor's Theorem

Theorem (Cantor's Theorem)

For every set S it holds that $|S| < |\mathcal{P}(S)|$.

Proof.

Consider an arbitrary set S. We need to show that

1. There is an injective function from S to $\mathcal{P}(S)$.
2. There is no bijection from S to $\mathcal{P}(S)$.

For 1, consider function $f : S \to \mathcal{P}(S)$ with $f(x) = \{x\}$.
It maps distinct elements of S to distinct elements of $\mathcal{P}(S)$. ...

Cantor's Theorem

Proof (continued).

We show 2 by contradiction.
Assume there is a bijection f from S to $\mathcal{P}(S)$.

Consider $M = \{x \mid x \in S, x \notin f(x)\}$ and note that $M \in \mathcal{P}(S)$.

Since f is bijective, it is surjective and there is an $y \in S$ with $f(y) = M$. Consider this y in a case distinction:

If $y \in M$ then $y \notin f(y)$ by the definition of M. Since $f(y) = M$ this implies $y \notin M$. $\leadsto$ contradiction

If $y \notin M$, we conclude from $f(y) = M$ that $y \notin f(y)$. Using the definition of M we get that $y \in M$. $\leadsto$ contradiction

Since all cases lead to a contradiction, there is no such y and thus f is not surjective and consequently not a bijection.

The assumption was false and we conclude that there is no bijection from S to $\mathcal{P}(S)$. $\square$

B8.2 Consequences of Cantor’s Theorem

Infinite Sets can Have Different Cardinalities

There are infinitely many different cardinalities of infinite sets:

- $|\mathbb{N}_0| < |\mathcal{P}(\mathbb{N}_0))| < |\mathcal{P}(\mathcal{P}(\mathbb{N}_0)))| < \dots$
- $|\mathbb{N}_0| = \aleph_0 = \beth_0$
- $|\mathcal{P}(\mathbb{N}_0)| = \beth_1 (= |\mathbb{R}|)$
- $|\mathcal{P}(\mathcal{P}(\mathbb{N}_0))| = \beth_2$
- $\dots$

Existence of Unsolvable Problems

There are more problems in computer science
than there are programs to solve them.

There are problems that cannot be solved by a computer program!

Why can we say so?

Decision Problems

"Intuitive Definition:" Decision Problem

A decision problem is a Yes-No question of the form
"Does the given input have a certain property?"

- "Does the given binary tree have more than three leaves?"
- "Is the given integer odd?"
- "Given a train schedule, is there a connection from Basel to Belinzona that takes at most 2.5 hours?"

- Input can be encoded as some finite string.
- Problem can also be represented as the (possibly infinite) set of all input strings where the answer is "yes".
- A computer program solves a decision problem if it terminates on every input and returns the correct answer.

More Problems than Programs I

- A computer program is given by a finite string.
- A decision problem corresponds to a set of strings.

More Problems than Programs II

- Consider an arbitrary finite set of symbols (an alphabet) Σ.
- You can think of $\Sigma = \{0, 1\}$ as internally computers operate on binary representation.
- Let S be the set of all finite strings made from symbols in Σ.
- There are at most $|S|$ computer programs with this alphabet.
- There are at least $|\mathcal{P}(S)|$ problems with this alphabet.
  - every subset of S corresponds to a separate decision problem
- By Cantor's theorem $|S| < |\mathcal{P}(S)|$, so there are more problems than programs.

B8.3 Sets: Summary

Summary

- Cantor's theorem: For all sets S it holds that $|S| < |\mathcal{P}(S)|$.
- There are problems that cannot be solved by a computer program.

B8.4 Outlook: Finite Sets and Computer Science

Enumerating all Subsets

Determine a one-to-one mapping between numbers $0, \dots, 2^{|S|} - 1$ and all subsets of finite set S:

- Consider the binary representation of numbers $0, \dots, 2^{|S|} - 1$.
- Associate every bit with a different element of S.
- Every number is mapped to the set that contains exactly the elements associated with the 1-bits.

$S = \{a, b, c\}$

decimal	binary abc	set
0	000	$\{\}$
1	001	$\{c\}$
2	010	$\{b\}$
3	011	$\{b, c\}$
4	100	$\{a\}$
5	101	$\{a, c\}$
6	110	$\{a, b\}$
7	111	$\{a, b, c\}$

Computer Representation as Bit String

Same representation as in enumeration of all subsets:

- Required: Fixed universe U of possible elements
- Represent sets as bitstrings of length $|U|$
- Associate every bit with one object from the universe
- Each bit is 1 iff the corresponding object is in the set

Example:

- $U = \{o_0, \dots, o_9\}$
- Associate the i-th bit (0-indexed, from left to right) with o_i
- $\{o_2, o_4, o_5, o_9\}$ is represented as:
  0010110001

How can the set operations be implemented?

Discrete Mathematics in Computer Science

B9. Divisibility & Modular Arithmetic

Malte Helmert, Gabriele Röger

University of Basel

November 3, 2025

B9.1 Divisibility

B9.2 Modular Arithmetic

B9.1 Divisibility

Divisibility

- Can we equally share n muffins among m persons without cutting a muffin?
- If yes then n is a multiple of m and m divides n.
- We consider a generalization of this concept to the integers.

Divisibility

Definition (divisor, multiple)

Let $m, n \in \mathbb{Z}$. If there exists a $k \in \mathbb{Z}$ such that $mk = n$, we say that m divides n, m is a divisor of n or n is a multiple of m and write this as $m \mid n$.

Which of the following are true?

- $2 \mid 4$
- $-2 \mid 4$
- $2 \mid -4$
- $4 \mid 2$
- $3 \mid 4$
- Every integer divides 0.

German: teilt, Teiler, Vielfaches

Divisibility and Linear Combinations

Theorem (Linear combinations)

Let a, b and d be integers. If $d \mid a$ and $d \mid b$ then for all integers x and y it holds that $d \mid xa + yb$.

Proof.

If $d \mid a$ and $d \mid b$ then there are $k, k' \in \mathbb{Z}$ such that $kd = a$ and $k'd = b$.
It holds for all $x, y \in \mathbb{Z}$ that $xa + yb = xkd + yk'd = (xk + yk')d$.
As x, y, k, k' are integers, $xk + yk'$ is integer, thus $d \mid xa + yb$. □

Some consequences:

- $d \mid a - b$ iff $d \mid b - a$
- If $d \mid a$ and $d \mid b$ then $d \mid a + b$ and $d \mid a - b$.
- If $d \mid a$ then $d \mid -8a$.

Multiplication and Exponentiation

Theorem

Let $a, b, c \in \mathbb{Z}$ and $n \in \mathbb{N}_1$.
If $a \mid b$ then $ac \mid bc$ and $a^n \mid b^n$.

Proof.

If $a \mid b$ there is a $k \in \mathbb{Z}$ such that $ak = b$.

Multiplying both sides with c, we get $cak = cb$ and thus $ca \mid cb$.

From $ak = b$, we also get $b^n = (ak)^n = a^n k^n$, so $a^n \mid b^n$. □

Partial Order

If we consider only the natural numbers,
divisibility is a partial order:

Theorem

Divisibility $\mid$ over $\mathbb{N}_0$ is a partial order.

Proof.

- reflexivity: For all $m \in \mathbb{N}_0$ it holds that $m \cdot 1 = m$, so $m \mid m$.
- transitivity: If $m \mid n$ and $n \mid o$ there are $k, k' \in \mathbb{Z}$ such that $mk = n$ and $nk' = o$.
  It holds that $o = nk' = mkk'$ and kk' is an integer, so we conclude $m \mid o$.

...

Partial Order

Proof (continued).

- antisymmetry: We show that if $m \mid n$ and $n \mid m$ then $m = n$.

  If $m = n = 0$, there is nothing to show.

  Otherwise, at least one of m and n is positive.

  Let this w.l.o.g. (without loss of generality) be m.
  If $m \mid n$ and $n \mid m$ then there are $k, k' \in \mathbb{Z}$
  such that $mk = n$ and $nk' = m$.

  Combining these, we get $m = nk' = mkk'$, which implies (with $m \neq 0$) that $kk' = 1$.

  Since k and k' are integers, this implies $k = k' = 1$ or $k = k' = -1$. As $mk = n$, m is positive and n is non-negative, we can conclude that $k = 1$ and $m = n$.

$\square$

B9.2 Modular Arithmetic

Halloween

- You have m sweets.
- There are k kids showing up for trick-or-treating.
- To keep everything fair, every kid gets the same amount of treats.
- You may enjoy the rest. :-)
- How much does every kid get, how much do you get?

Euclid's Division Lemma

Theorem (Euclid's division lemma)

For all integers a and b with $b \neq 0$ there are unique integers q and r with $a = qb + r$ and $0 \leq r < |b|$.

Number a is called the dividend, b the divisor, q is the quotient and r the remainder.

Without proof.

Examples:

- $a = 18, b = 5$
- $a = 5, b = 18$
- $a = -18, b = 5$
- $a = 18, b = -5$

German: Division mit Rest, Dividend, Divisor, Ganzzahlquotient, Rest

Modulo Operation

- With $a \bmod b$ we refer to the remainder of Euclidean division.
- Most programming languages have a built-in operator to compute $a \bmod b$ (for positive integers):

```
int mod = 34 % 7;
// result 6 because 4 * 7 + 6 = 34
```

- Common application: Determine whether a natural number n is even.

```
n % 2 == 0
```

- Languages behave differently with negative operands!

Halloween

```
def share_sweets(no_kids, no_sweets):
    print("Each kid gets",
          no_sweets // no_kids,
          "of the sweets.")
    print("You may keep",
          no_sweets % no_kids,
          "of the sweets.")
```

Congruence Modulo n

- We now are no longer interested in the value of the remainder but will consider numbers a and a' as equivalent if the remainder with division by a given number b is equal.
- Consider the clock:
  - It's now 3 o'clock
  - In 12 hours its 3 o'clock
  - Same in 24, 36, 48, … hours.
  - 15:00 and 3:00 are shown the same.
  - In the following, we will express this as $3 \equiv 15 \pmod{12}$

Congruence Modulo n – Definition

Definition (Congruence modulo n)

For integer $n > 1$, two integers a and b
are called congruent modulo n if $n \mid a - b$.

We write this as $a \equiv b \pmod{n}$.

Which of the following statements are true?

- $0 \equiv 5 \pmod{5}$
- $1 \equiv 6 \pmod{5}$
- $4 \equiv 14 \pmod{5}$
- $-8 \equiv 7 \pmod{5}$
- $2 \equiv -3 \pmod{5}$

Why is this the same concept as described in the clock example?!?

German: kongruent modulo n

Congruence Corresponds to Equal Remainders

Theorem

For integers a and b and integer $n > 1$ it holds that $a \equiv b \pmod{n}$ iff there are $q, q', r \in \mathbb{Z}$ with

$$a = qn + r$$
$$b = q'n + r.$$

Proof sketch.

"$\Rightarrow$": If $n \mid a - b$ then there is a $k \in \mathbb{Z}$ with $kn = a - b$.

As $n \neq 0$, by Euclid's lemma there are $q, q', r, r' \in \mathbb{Z}$ with $a = qn + r$ and $b = q'n + r'$, where $0 \leq r < |n|$ and $0 \leq r' < |n|$.

Together, we get that $kn = qn + r - (q'n + r')$, which is the case iff $kn + r' = (q - q')n + r$. By Euclid's lemma, quotients and remainders are unique, so in particular $r' = r$.

"$\Leftarrow$": If we subtract the equations, we get $a - b = (q - q')n$, so $n \mid a - b$ and $a \equiv b \pmod{n}$.

Congruence Modulo n is an Equivalence Relation

Theorem

Congruence modulo n is an equivalence relation.

Proof sketch.

Reflexive: $a \equiv a \pmod{n}$ because every integer divides 0.

Symmetric: $a \equiv b \pmod{n}$ iff $n \mid a - b$ iff $n \mid b - a$ iff $b \equiv a \pmod{n}$.

Transitive: If $a \equiv b \pmod{n}$ and $b \equiv c \pmod{n}$ then $n \mid a - b$ and $n \mid b - c$. Together, these imply that $n \mid a - b + b - c$. From $n \mid a - c$ we get $a \equiv c \pmod{n}$.

For modulus n, the equivalence class of a is
$\bar{a}_n = \{\dots, a - 2n, a - n, a, a + n, a + 2n, \dots\}$.
Set $\bar{a}_n$ is called the congruence class or residue of a modulo n.

German: Restklasse

Compatibility with Operations

Theorem

Congruence modulo n is compatible with addition, subtraction, multiplication, translation, scaling and exponentiation, i. e. if $a \equiv b \pmod{n}$ and $a' \equiv b' \pmod{n}$ then

- $a + a' \equiv b + b' \pmod{n}$,
- $a - a' \equiv b - b' \pmod{n}$,
- $aa' \equiv bb' \pmod{n}$,
- $a + k \equiv b + k \pmod{n}$ *for all* $k \in \mathbb{Z}$,
- $ak \equiv bk \pmod{n}$ *for all* $k \in \mathbb{Z}$, *and*
- $a^k \equiv b^k \pmod{n}$ *for all* $k \in \mathbb{N}_0$.

Congruence modulo n is a so-called congruence relation (= equivalence relation compatible with operations).

German: kompatibel mit Addition, Subtraktion, Multiplikation, Translation, Skalierung, Exponentiation; Kongurenzrelation

Summary

- m divides n (written $m \mid n$) if n is a multiple of m, i.e. there is an integer k with $n = mk$.
- Divisibility is compatible with multiplication and exponentiation.
- Divisibility over the natural numbers is a partial order.
- The modulo operation $a \bmod b$ corresponds to the remainder of Euclidean division.
- Congruence modulo n considers integers equivalent if they have with divisor n the same remainder.
- Congruence modulo n is an equivalence relation that is compatible with the arithmetic operations.

Discrete Mathematics in Computer Science

C1. Introduction to Graphs

Malte Helmert, Gabriele Röger

University of Basel

November 5, 2025

C1.1 Graphs and Directed Graphs

C1.2 Induced Graphs and Degree Lemma

C1.1 Graphs and Directed Graphs

Graphs

Graphs (of various kinds) are ubiquitous in Computer Science and its applications.

Some examples:

- Boolean circuits in hardware design
- control flow graphs in compilers
- pathfinding in video games
- computer networks
- neural networks
- social networks

Graph Theory

- Graph theory was founded in 1736 by Leonhard Euler's study of the Seven Bridges of Königsberg problem.
- It remains one of the main areas of discrete mathematics to this day.

More on Euler and the Seven Bridges of Königsberg:

The Seven Bridges of Königsberg - Numberphile
SOLVTIO PROBLEMATIS AD GEOMETRIAM SITVS PERTINENTIS. AVCTORE Leonh. Eulero.
1:41 / 14:41

- The Seven Bridges of Königsberg – Numberphile.
  `https://youtu.be/W18FDEA1jRQ`

Graphs and Directed Graphs – Definitions

Definition (Graph)

A graph (also: undirected graph) is a pair $G = (V, E)$, where

- V is a finite set called the set of vertices, and
- $E \subseteq \{\{u, v\} \subseteq V \mid u \neq v\}$ is called the set of edges.

German: Graph, ungerichteter Graph, Knoten, Kanten

Definition (Directed Graph)

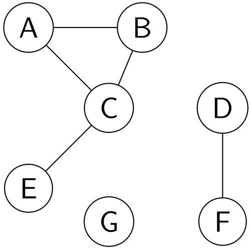
A directed graph (also: digraph) is a pair $G = (N, A)$, where

- N is a finite set called the set of nodes, and
- $A \subseteq N \times N$ is called the set of arcs.

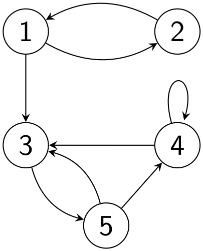
German: gerichteter Graph, Digraph, Knoten, Kanten/Pfeile

Graphs and Directed Graphs – Pictorially

often described pictorially:



graph (V, E)



directed graph (N, A)

- $V = \{\mathrm{A}, \mathrm{B}, \mathrm{C}, \mathrm{D}, \mathrm{E}, \mathrm{F}, \mathrm{G}\}$
- $E = \{\{\mathrm{A}, \mathrm{B}\}, \{\mathrm{A}, \mathrm{C}\}, \{\mathrm{B}, \mathrm{C}\}, \{\mathrm{C}, \mathrm{E}\}, \{\mathrm{D}, \mathrm{F}\}\}$

- $N = \{1, 2, 3, 4, 5\}$
- $A = \{(1, 2), (1, 3), (2, 1), (3, 5), (4, 3), (4, 4), (5, 3), (5, 4)\}$

Relationship to Relations

graphs vs. directed graphs:

- edges are sets of two elements, arcs are pairs
- arcs can be self-loops (v, v); edges cannot (why not?)

(di-)graphs vs. relations:

- A directed graph (N, A) is essentially identical to
  (= contains the same information as)
  an arbitrary relation R_A over the finite set N:
  $u \, R_A \, v$ iff $(u, v) \in A$
- A graph (V, E) is essentially identical to
  an irreflexive symmetric relation R_E over the finite set V:
  $u \, R_E \, v$ iff $\{u, v\} \in E$

Other Kinds of Graphs

many variations exist, for example:

- self-loops may be allowed in edges ("non-simple" graphs)
- labeled graphs: additional information associated with vertices and/or edges
- weighted graphs: numbers associated with edges
- multigraphs: multiple edges between same vertices allowed
- mixed graphs: both edges and arcs allowed
- hypergraphs: edges can involve more than 2 vertices
- infinite graphs: may have infinitely many vertices/edges

Graph Terminology

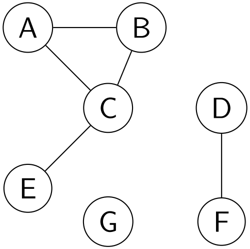
Definition (Graph Terminology)

Let (V, E) be a graph.

- u and v are the endpoints of the edge $\{u, v\} \in E$
- u and v are incident to the edge $\{u, v\} \in E$
- u and v are adjacent if $\{u, v\} \in E$
- the vertices adjacent with $v \in V$ are its neighbours $\text{neigh}(v)$: $\text{neigh}(v) = \{w \in V \mid \{v, w\} \in E\}$
- the number of neighbours of $v \in V$ is its degree $\deg(v)$: $\deg(v) = |\text{neigh}(v)|$

German: Endknoten, inzident, adjazent/benachbart, Nachbarn, Grad

Graph Terminology – Examples



endpoints, incident, adjacent, neighbours, degree

Directed Graph Terminology

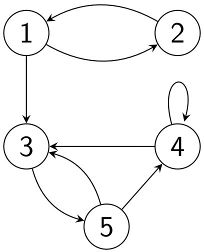
Definition (Directed Graph Terminology)

Let (N, A) be a directed graph.

- u is the tail and v is the head of the arc $(u, v) \in A$;
  we say (u, v) is an arc from u to v
- u and v are incident to the arc $(u, v) \in A$
- u is a predecessor of v and v is a successor of u if $(u, v) \in A$
- the predecessors and successor of v are written as
  $\mathrm{pred}(v) = \{u \in N \mid (u, v) \in A\}$ and
  $\mathrm{succ}(v) = \{w \in N \mid (v, w) \in A\}$
- the number of predecessors/successors of $v \in N$ is its
  indegree/outdegree: $\mathrm{indeg}(v) = |\mathrm{pred}(v)|$,
  $\mathrm{outdeg}(v) = |\mathrm{succ}(v)|$

German: Fuss, Kopf, inzident, Vorgänger, Nachfolger, Eingangs-/Ausgangsgrad

Directed Graph Terminology – Examples



head, tail, predecessors, successors, indegree, outdegree

C1.2 Induced Graphs and Degree Lemma

Induced Graph of a Directed Graph

Definition (undirected graph induced by a directed graph)

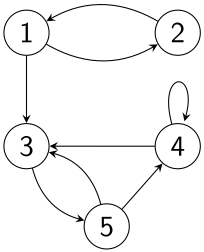
Let $G = (N, A)$ be a directed graph.
The (undirected) **graph induced by G** is the graph (N, E) with
$E = \{\{u, v\} \mid (u, v) \in A, u \neq v\}$.

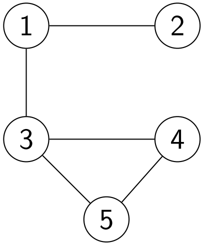
German: induziert

Questions:

- Why require $u \neq v$?
- If $|N| = n$ and $|A| = m$, how many vertices and edges does the induced graph have?
- How does the answer change if G has no self-loops?

Induced Graph of a Directed Graph – Example





- $N = \{1, 2, 3, 4, 5\}$
- $A = \{(1, 2), (1, 3), (2, 1), (3, 5),$
  $(4, 3), (4, 4), (5, 3), (5, 4)\}$

- $V = \{1, 2, 3, 4, 5\}$
- $E = \{\{1, 2\}, \{1, 3\}, \{3, 4\},$
  $\{3, 5\}, \{4, 5\}\}$

Degree Lemma

Lemma (degree lemma for directed graphs)

Let (N, A) be a directed graph.
Then $\sum_{v \in N} \mathrm{indeg}(v) = \sum_{v \in N} \mathrm{outdeg}(v) = |A|$.

Intuitively: every arc contributes 1 to the indegree of one node and 1 to the outdegree of one node.

Lemma (degree lemma for undirected graphs)

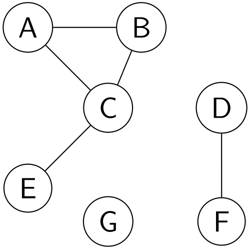
Let (V, E) be a graph.
Then $\sum_{v \in V} \deg(v) = 2|E|$.

Intuitively: every edge contributes 1 to the degree of two vertices.

Corollary

Every graph has an even number of vertices with odd degree.

Degree Lemma – Example



$$\begin{aligned}
&\sum_{v \in V} \deg(v) \\
&= \deg(\mathrm{A}) + \deg(\mathrm{B}) + \deg(\mathrm{C}) + \deg(\mathrm{D}) + \deg(\mathrm{E}) + \deg(\mathrm{F}) + \deg(\mathrm{G}) \\
&= 2 + 2 + 3 + 1 + 1 + 1 + 0 \\
&= 10 = 2 \cdot 5 = 2|E|
\end{aligned}$$

4 vertices with odd degree

Degree Lemma – Proof (1)

Proof of degree lemma for directed graphs.

$$\begin{aligned}
\sum_{v\in N} \mathrm{indeg}(v) &= \sum_{v\in N} |\mathrm{pred}(v)| \\
&= \sum_{v\in N} |\{u \mid u \in N, (u,v) \in A\}| \\
&= \sum_{v\in N} |\{(u,v) \mid u \in N, (u,v) \in A\}| \\
&= \left| \bigcup_{v\in N} \{(u,v) \mid u \in N, (u,v) \in A\} \right| \\
&= |\{(u,v) \mid u \in N, v \in N, (u,v) \in A\}| \\
&= |A|.
\end{aligned}$$

$\sum_{v\in N} \mathrm{outdeg}(v) = |A|$ is analogous. □

Degree Lemma – Proof (2)

We omit the proof for undirected graphs,
which can be conducted similarly.

One possible proof strategy that reuses the result we proved:

- Define directed graph (V, A) from the graph (V, E) by orienting each edge into an arc arbitrarily.
- Observe $\text{deg}(v) = \text{indeg}(v) + \text{outdeg}(v)$, where deg refers to the graph and indeg/outdeg to the directed graph.
- Use the degree lemma for directed graphs:
  $\sum_{v \in V} \text{deg}(v) = \sum_{v \in V}(\text{indeg}(v) + \text{outdeg}(v)) =$
  $\sum_{v \in V} \text{indeg}(v) + \sum_{v \in V} \text{outdeg}(v) = |A| + |A| = 2|A| = 2|E|$

Discrete Mathematics in Computer Science

C2. Paths and Connectivity

Malte Helmert, Gabriele Röger

University of Basel

November 10, 2025

C2.1 Walks, Paths, Tours and Cycles

Traversing Graphs

- When dealing with graphs, we are often not just interested in the neighbours, but also in the neighbours of neighbours, the neighbours of neighbours of neighbours, etc.
- Similarly, for digraphs we often want to follow longer chains of successors (or chains of predecessors).

Examples:

- circuits: follow predecessors of signals to identify possible causes of faulty signals
- pathfinding: follow edges/arcs to find paths
- control flow graphs: follow arcs to identify dead code
- computer networks: determine if part of the network is unreachable

Walks

Definition (Walk)

A walk of length n in a graph (V, E) is a tuple $\langle v_0, v_1, \dots, v_n \rangle \in V^{n+1}$ s.t. $\{v_i, v_{i+1}\} \in E$ for all $0 \leq i < n$.

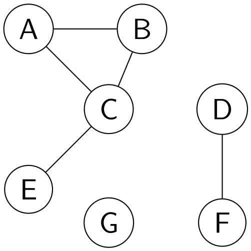
A walk of length n in a digraph (N, A) is a tuple $\langle v_0, v_1, \dots, v_n \rangle \in N^{n+1}$ s.t. $(v_i, v_{i+1}) \in A$ for all $0 \leq i < n$.

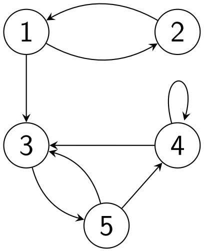
German: Wanderung

Notes:

- The length of the walk does not equal the length of the tuple!
- The case $n = 0$ is allowed.
- Vertices may repeat along a walk.

Walks – Example





examples of walks:

- $\langle B, C, A \rangle$
- $\langle B, C, A, B \rangle$
- $\langle D, F, D \rangle$
- $\langle B, A, B, C, E \rangle$
- $\langle B \rangle$

examples of walks:

- $\langle 4, 4, 4, 4 \rangle$
- $\langle 3, 5, 3, 5 \rangle$
- $\langle 2, 1, 3 \rangle$
- $\langle 4 \rangle$
- $\langle 4, 4 \rangle$

Walks – Terminology

Definition

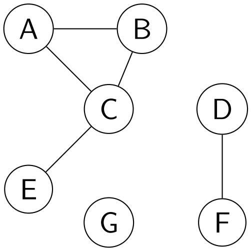
Let $\pi = \langle v_0, \dots, v_n \rangle$ be a walk in a graph or digraph G.

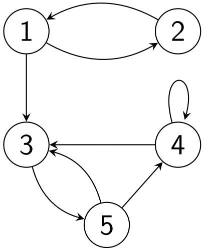
- We say π is a walk from v_0 to v_n.
- A walk with $v_i \neq v_j$ for all $0 \leq i < j \leq n$ is called a path.
- A walk of length 0 is called an empty walk/path.
- A walk with $v_0 = v_n$ is called a tour.
- A tour with $n \geq 1$ (digraphs) or $n \geq 3$ (graphs) and $v_i \neq v_j$ for all $1 \leq i < j \leq n$ is called a cycle.

German: von/nach, Pfad, leer, Tour, Zyklus

Note: Terminology is not very consistent in the literature.

Walks, Paths, Tours, Cycles – Example





Which walks are paths, tours, cycles?

- $\langle B, C, A \rangle$
- $\langle B, C, A, B \rangle$
- $\langle D, F, D \rangle$
- $\langle B, A, B, C, E \rangle$
- $\langle B \rangle$
- $\langle 4, 4, 4, 4 \rangle$
- $\langle 3, 5, 3, 5 \rangle$
- $\langle 2, 1, 3 \rangle$
- $\langle 4 \rangle$
- $\langle 4, 4 \rangle$

C2.2 Reachability

Reachability

Definition (successor and reachability)

Let G be a graph (digraph).
The successor relation S_G and reachability relation R_G are relations over the vertices/nodes of G defined as follows:

- $(u, v) \in \mathrm{S}_G$ iff $\{u, v\}$ is an edge ((u, v) is an arc) of G
- $(u, v) \in \mathrm{R}_G$ iff there exists a walk from u to v

If $(u, v) \in \mathrm{R}_G$, we say that v is reachable from u.

German: Nachfolger-/Erreichbarkeitsrelation, erreichbar

Reachability as Closure

Recall the n-fold composition R_n of a relation R over set S (Chapter B4):

- $R_0 = \{(x, x) \mid x \in S\}$
- $R_n = R \circ R_{n-1}$ for $n \geq 1$

Theorem

Let G be a graph or digraph. Then:
$(u, v) \in (\mathrm{S}_G)_n$ iff there exists a walk of length n from u to v.

Corollary

Let G be a graph or digraph. Then $\mathrm{R}_G = \bigcup_{n=0}^{\infty} (\mathrm{S}_G)_n$.

In other words, the reachability relation is the reflexive transitive closure of the successor relation.

Reachability as Closure – Proof (1)

Proof.

To simplify notation, we assume $G = (N, A)$ is a digraph.
Graphs are analogous.
Proof by induction over n.

induction base ($n = 0$):
By definition of the 0-fold composition, we have $(u, v) \in (\mathrm{S}_G)_0$ iff $u = v$, and a walk of length 0 from u to v exists iff $u = v$.
Hence, the two conditions are equivalent. . . .

Reachability as Closure – Proof (2)

Proof (continued).

induction step ($n \to n+1$):

($\Rightarrow$) : Let $(u, v) \in (\mathsf{S}_G)_{n+1}$.
By definition of S_{n+1}, we get $(u, v) \in \mathsf{S}_G \circ (\mathsf{S}_G)_n$.
By definition of $\circ$ there exists w with $(u, w) \in (\mathsf{S}_G)_n$ and $(w, v) \in \mathsf{S}_G$.
From the induction hypothesis, there exists a length-n walk $\langle x_0, \dots, x_n \rangle$ with $x_0 = u$ and $x_n = w$.
Then $\langle x_0, \dots, x_n, v \rangle$ is a length-$(n+1)$ walk from u to v.

($\Leftarrow$) : Let $\langle x_0, \dots, x_{n+1} \rangle$ be a length-$(n+1)$ walk from u to v ($x_0 = u$, $x_{n+1} = v$). Then $(x_n, x_{n+1}) = (x_n, v) \in A$.
Also, $\langle x_0, \dots, x_n \rangle$ is a length-n walk from x_0 to x_n.
From the IH we get $(u, x_n) = (x_0, x_n) \in (\mathsf{S}_G)_n$.
Together with $(x_n, v) \in \mathsf{S}_G$ this shows
$(u, v) \in \mathsf{S}_G \circ (\mathsf{S}_G)_n = (\mathsf{S}_G)_{n+1}$. □

C2.3 Connected Components

Overview

- In this section, we study reachability of graphs in more depth.
- We show that it makes no difference whether we define reachability in terms of walks or paths, and that reachability in graphs is an equivalence relation.
- This leads to the connected components of a graph.
- In digraphs, reachability is not always an equivalence relation.
- However, we can define two variants of reachability that give rise to weakly or strongly connected components.

Walks vs. Paths

Theorem

Let G be a graph or digraph.
There exists a path from u to v iff there exists a walk from u to v.

In other words, there is a path from u to v iff v is reachable from u.

Proof.

($\Rightarrow$): obvious because paths are special cases of walks

($\Leftarrow$): Proof by contradiction. Assume there exist u, v such that there exists a walk from u to v, but no path. Let $\pi = \langle w_0, \dots, w_n \rangle$ be such a counterexample walk of minimal length.
Because π is not a path, some vertex/node must repeat.
Select i and j with $i < j$ and $w_i = w_j$.
Then $\pi' = \langle w_0, \dots, w_i, w_{j+1}, \dots, w_n \rangle$ also is a walk from u to v.
If π' is a path, we have a contradiction.
If not, it is a shorter counterexample: also a contradiction. □

Reachability in Graphs is an Equivalence Relation

Theorem

For every graph G, the reachability relation R_G is an equivalence relation.

In directed graphs, this result does not hold (easy to see).

Proof.

We already know reachability is reflexive and transitive.
To prove symmetry:

$$\begin{aligned} & (u, v) \in \mathrm{R}_G \\ \Rightarrow\ & \text{there is a walk } \langle w_0, \dots, w_n \rangle \text{ from } u \text{ to } v \\ \Rightarrow\ & \langle w_n, \dots, w_0 \rangle \text{ is a walk from } v \text{ to } u \\ \Rightarrow\ & (v, u) \in \mathrm{R}_G \end{aligned}$$

□

Connected Components

Definition (connected components, connected)

In a graph G, the equivalence classes
of the reachability relation of G
are called the connected components of G.

A graph is called connected if it has at most 1
connected component.

German: Zusammenhangskomponenten, zusammenhängend

Remark: The graph $(\emptyset, \emptyset)$ has 0 connected components.
It is the only such graph.

Weakly Connected Components

Definition (weakly connected components, weakly connected)

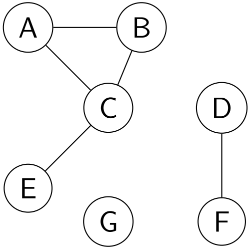
In a digraph G, the equivalence classes of the reachability relation of the induced graph of G are called the weakly connected components of G.

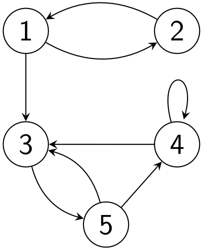
A digraph is called weakly connected if it has at most 1 weakly connected component.

German: schwache Zshk., schwach zusammenhängend

Remark: The digraph $(\emptyset, \emptyset)$ has 0 weakly connected components. It is the only such digraph.

(Weakly) Connected Components – Example





connected components:

- $\{A, B, C, E\}$
- $\{D, F\}$
- $\{G\}$

weakly connected components:

- $\{1, 2, 3, 4, 5\}$

Mutual Reachability

Definition (mutually reachable)

Let G be a graph or digraph.
Vertices/nodes u and v in G are called mutually reachable if v is reachable from u and u is reachable from v.
We write M_G for the mutual reachability relation of G

German: gegenseitig erreichbar

Note: In graphs, $\mathrm{M}_G = \mathrm{R}_G$. (Why?)

Mutual Reachability is an Equivalence Relation

Theorem

For every digraph G, the mutual reachability relation M_G is an equivalence relation.

Proof.

Note that $(u, v) \in \mathrm{M}_G$ iff $(u, v) \in \mathrm{R}_G$ and $(v, u) \in \mathrm{R}_G$.

- reflexivity: for all v, we have $(v, v) \in \mathrm{M}_G$ because $(v, v) \in \mathrm{R}_G$
- symmetry: Let $(u, v) \in \mathrm{M}_G$. Then $(v, u) \in \mathrm{M}_G$ is obvious.
- transitivity: Let $(u, v) \in \mathrm{M}_G$ and $(v, w) \in \mathrm{M}_G$.
  Then: $(u, v) \in \mathrm{R}_G$, $(v, u) \in \mathrm{R}_G$, $(v, w) \in \mathrm{R}_G$, $(w, v) \in \mathrm{R}_G$.
  Transitivity of R_G yields $(u, w) \in \mathrm{R}_G$ and $(w, u) \in \mathrm{R}_G$,
  and hence $(u, w) \in \mathrm{M}_G$.

$\square$

Strongly Connected Components

Definition (strongly connected components, strongly connected)

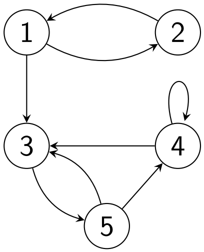
In a digraph G, the equivalence classes
of the mutual reachability relation
are called the strongly connected components of G.

A digraph is called strongly connected if it has at most 1
strongly connected component.

German: starke Zshk., stark zusammenhängend

Remark: The digraph $(\emptyset, \emptyset)$ has 0 strongly connected components.
It is the only such digraph.

Strongly Connected Components – Example



strongly connected components:

- $\{1, 2\}$
- $\{3, 4, 5\}$

Discrete Mathematics in Computer Science

C3. Acyclicity

Malte Helmert, Gabriele Röger

University of Basel

November 10/12, 2025

Discrete Mathematics in Computer Science

November 10/12, 2025 — C3. Acyclicity

C3.1 Acyclic (Di-) Graphs

Acyclic

Similarly to connectedness, the presence or absence of cycles is an important practical property for (di-) graphs.

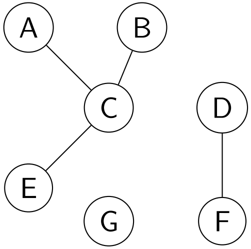
Definition (acyclic, forest, DAG)

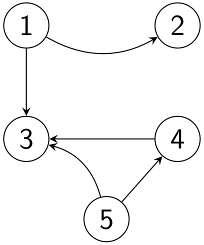
A graph or digraph G is called acyclic if there exists no cycle in G.

An acyclic graph is also called a forest.
An acyclic digraph is also called a DAG (directed acyclic graph).

German: azyklisch/kreisfrei, Wald, DAG

Acyclic (Di-) Graphs – Example





Trees

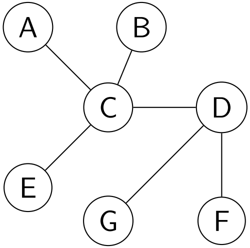
Definition (tree)

A connected forest is called a tree.

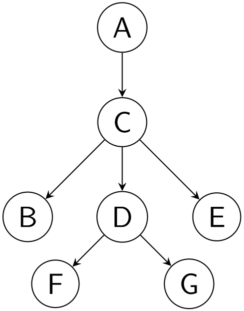
German: Baum

- Tree is also a word for a recursive data structure, which consists of either a leaf or a parent node with one or more children, which are themselves trees.
- This other kind of tree is also called a rooted tree to distinguish it from a tree as a graph.
- The two meanings of “tree” are distinct but closely related.

Tree Graphs vs. Rooted Trees – Example (1)

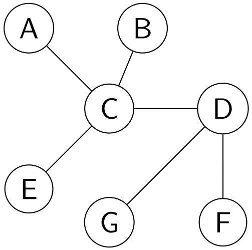


tree graph

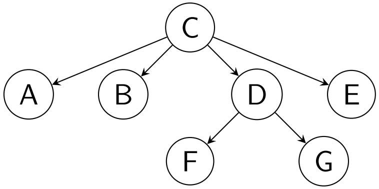


rooted tree with root A

Tree Graphs vs. Rooted Trees – Example (2)

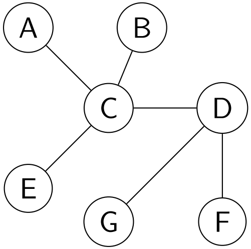


tree graph

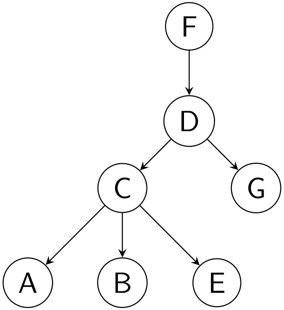


rooted tree with root C

Tree Graphs vs. Rooted Trees – Example (3)



tree graph



rooted tree with root F

From Tree Graphs to Rooted Trees

General procedure for converting tree graphs into rooted trees:

- Select any vertex v. Make v the root of the tree.
- Initially, v is the only pending vertex, and there are no processed vertices.
- As long as there are pending vertices:
  - Select any pending vertex u.
  - Make all neighbours v of u that are not yet processed children of u and mark them as pending.
  - Change u from pending to processed.

We do not prove that this procedure always works. A proof of correctness can be given based on the results we show next.

C3.2 Unique Paths in Trees

Unique Paths in Trees

Theorem

Let $G = (V, E)$ be a graph.
Then G is a tree iff there exists exactly one path from any vertex $u \in V$ to any vertex $v \in V$.

Unique Paths In Trees – Proof (1)

Proof.

(⇒): G is a tree. Let $u, v \in V$.
We must show that there exists exactly one path from u to v.

We know that at least one path exists because G is connected.

It remains to show that there cannot be two paths from u to v.
If $u = v$, there is only one path (the empty one).
(Any longer path would have to repeat a vertex.)

We assume that there exist two different paths from u to v ($u \neq v$) and derive a contradiction. ...

Unique Paths In Trees – Proof (2)

Proof (continued).

Let $\pi = \langle v_0, v_1, \dots, v_n \rangle$ and $\pi' = \langle v'_0, v'_1, \dots, v'_m \rangle$ be the two paths (with $v_0 = v'_0 = u$ and $v_n = v'_m = v$).
Let i be the smallest index with $v_i \neq v'_i$, which must exist because the two paths are different, and neither can be a prefix of the other (else v would be repeated in the longer path).
We have $i \geq 1$ because $v_0 = v'_0$.
Let $j \geq i$ be the smallest index such that $v_j = v'_k$ for some $k \geq i$.
Such an index must exist because $v_n = v'_m$.
Then $\langle v_{i-1}, \dots, v_{j-1}, v'_k, \dots, v'_{i-1} \rangle$ is a cycle,
which contradicts the requirement that G is a tree. ...

Unique Paths In Trees – Proof (3)

Proof (continued).

($\Leftarrow$): For all $u, v \in V$, there exists exactly one path from u to v.
We must show that G is a tree, i.e., is connected and acyclic.

Because there exist paths from all u to all v, G is connected.

Proof by contradiction: assume that there exists a cycle in G,
$\pi = \langle u, v_1, \dots, v_n, u \rangle$ with $n \geq 2$.
(Note that all cycles have length at least 3.)
From the definition of cycles, we have $v_1 \neq v_n$.

Then $\langle u, v_1 \rangle$ and $\langle u, v_n, \dots, v_1 \rangle$ are two different paths from u to v_1, contradicting that there exists exactly one path from every vertex to every vertex. Hence G must be acyclic. □

C3.3 Leaves and Edge Counts in Trees and Forests

Leaves in Trees

Definition

Let $G = (V, E)$ be a tree.
A leaf of G is a vertex $v \in V$ with $\deg(v) \leq 1$.

Note: The case $\deg(v) = 0$ only occurs in single-vertex trees ($|V| = 1$). In trees with at least two vertices, vertices with degree 0 cannot exist because this would make the graph unconnected.

Theorem

Let $G = (V, E)$ be a tree with $|V| \geq 2$.
Then G has at least two leaves.

Leaves in Trees – Proof

Proof.

Let $\pi = \langle v_0, \dots, v_n \rangle$ be path in G with maximal length among all paths in G.
Because $|V| \geq 2$, we have $n \geq 1$ (else G would not be connected).

We show that vertex v_n has degree 1: v_{n-1} is a neighbour in G.
Assume that it were not the only neighbour of v_n in G, so u is another neighbour of v_n. Then:

- If u is not on the path, then $\langle v_0, \dots, v_n, u \rangle$ is a longer path: contradiction.
- If u is on the path, then $u = v_i$ for some $i \neq n$ and $i \neq n - 1$. Then $\langle v_i, \dots, v_n, v_i \rangle$ is a cycle: contradiction.

By reversing π we can show $\deg(v_0) = 1$ in the same way. □

Edges in Trees

Theorem

Let $G = (V, E)$ be a tree with $V \neq \emptyset$.
Then $|E| = |V| - 1$.

Edges in Trees – Proof (1)

Proof.

Proof by induction over $n = |V|$.

Induction base ($n = 1$):
Then G has 1 vertex and 0 edges.
We get $|E| = 0 = 1 - 1 = |V| - 1$.

Induction step ($n \to n + 1$):
Let $G = (V, E)$ be a tree with $n + 1$ vertices ($n \geq 1$).
From the previous result, G has a leaf u.
Let v be the only neighbour of u.
Let $e = \{u, v\}$ be the connecting edge. ...

Edges in Trees – Proof (2)

Proof (continued).

Consider the graph $G' = (V', E')$
with $V' = V \setminus \{u\}$ and $E' = E \setminus \{e\}$.

- G' is acyclic: every cycle in G' would also be present in G (contradiction).
- G' is connected: for all vertices $w \neq u$ and $w' \neq u$,
  G has a path π from w to w' because G is connected.
  Path π cannot include u because u has only one neighbour, so traversing u requires repeating v. Hence π is also a path in G'.

Hence G' is a tree with n vertices, and we can apply
the induction hypothesis, which gives $|E'| = |V'| - 1$.
It follows that
$|E| = |E'| + 1 = (|V'| - 1) + 1 = (|V'| + 1) - 1 = |V| - 1$. □

Edges in Forests

Theorem

Let $G = (V, E)$ be a forest.
Let C be the set of connected components of G.
Then $|E| = |V| - |C|$.

This result generalizes the previous one.

Edges in Forests – Proof

Proof.

Let $C = \{C_1, \dots, C_k\}$.

For $1 \leq i \leq k$, let $G_i = (C_i, E_i)$ be G restricted to C_i, i.e., the graph whose vertices are C_i and whose edges are the edges $e \in E$ with $e \subseteq C_i$.

We have $|V| = \sum_{i=1}^{k} |C_i|$ because the connected components form a partition of V.

We have $|E| = \sum_{i=1}^{k} |E_i|$ because every edge belongs to exactly one connected component. (Note that there cannot be edges between different connected components.)

Every graph G_i is a tree with at least one vertex: it is connected because its vertices form a connected component, and it is acyclic because G is acyclic. This implies $|E_i| = |C_i| - 1$.

Putting this together, we get $|E| = \sum_{i=1}^{k} |E_i| = \sum_{i=1}^{k}(|C_i| - 1) = \sum_{i=1}^{k} |C_i| - k = |V| - |C|$. □

C3.4 Characterizations of Trees

Characterizations of Trees

Theorem

Let $G = (V, E)$ be a graph with $V \neq \emptyset$.
The following statements are equivalent:

1. *G is a tree.*
2. *G is acyclic and connected.*
3. *G is acyclic and $|E| = |V| - 1$.*
4. *G is connected and $|E| = |V| - 1$.*
5. *For all $u, v \in V$ there exists exactly one path from u to v.*

Characterizations of Trees – Proof (1)

Reminder:
(1) G is a tree.
(2) G is acyclic and connected.
(3) G is acyclic and $|E| = |V| - 1$.
(4) G is connected and $|E| = |V| - 1$.
(5) For all $u, v \in V$ there exists exactly one path from u to v.

Proof.
We know already:

- (1) and (2) are equivalent by definition of trees.
- We have shown that (1) and (5) are equivalent.
- We have shown that (1) implies (3) and (4).

We complete the proof by showing (3) $\Rightarrow$ (2) and (4) $\Rightarrow$ (2). . . .

Characterizations of Trees – Proof (2)

Reminder:
(2) G is acyclic and connected.
(3) G is acyclic and $|E| = |V| - 1$.

Proof (continued).
(3) $\Rightarrow$ (2):
Because G is acyclic, it is a forest.
From the previous result, we have $|E| = |V| - |C|$,
where C are the connected components of G.

But we also know $|E| = |V| - 1$. This implies $|C| = 1$.
Hence G is connected and therefore a tree. ...

Characterizations of Trees – Proof (3)

Reminder:

(2) G is acyclic and connected.

(4) G is connected and $|E| = |V| - 1$.

Proof (continued).

(4) $\Rightarrow$ (2):

In graphs that are not acyclic, we can remove an edge without changing the connected components: if $\langle v_0, \dots, v_n, v_0 \rangle$ ($n \geq 2$) is a cycle, remove the edge $\{v_0, v_1\}$ from the graph.
Every walk using this edge can substitute $\langle v_1, \dots, v_n, v_0 \rangle$ (or the reverse path) for it.

Iteratively remove edges from G in this way while preserving connectedness until this is no longer possible. The resulting graph (V, E') is acyclic and connected and therefore a tree.

This implies $|E'| = |V| - 1$, but we also have $|E| = |V| - 1$.
This yields $|E| = |E'|$ and hence $E' = E$: the number of edges removable in this way must be 0. Hence G is already acyclic. □

Discrete Mathematics in Computer Science

C4. Further Topics in Graph Theory

Malte Helmert, Gabriele Röger

University of Basel

November 17/19, 2025

Discrete Mathematics in Computer Science

November 17/19, 2025 — C4. Further Topics in Graph Theory

C4.1 Subgraphs

Overview

- We conclude our discussion of (di-) graphs by giving a brief tour of some further topics in graph theory that we do not have time to discuss in depth.
- In the interest of brevity (and hence wider coverage of topics), we do not give proofs for the results in this chapter.

Subgraphs

Definition (subgraph)

A subgraph of a graph (V, E) is a graph (V', E') with $V' \subseteq V$ and $E' \subseteq E$.

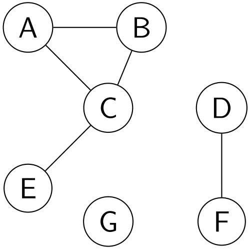
A subgraph of a digraph (N, A) is a digraph (N', A') with $N' \subseteq N$ and $A' \subseteq A$.

German: Teilgraph/Untergraph

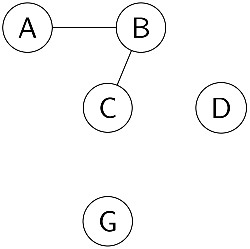
Question: Can we choose V' and E' arbitrarily?

The subgraph relationship defines a partial order on graphs (and on digraphs).

Subgraphs – Example



graph (V, E)



subgraph (V', E')

Induced Subgraphs (1)

Definition (induced subgraph)

Let $G = (V, E)$ be a graph, and let $V' \subseteq V$.
The subgraph of G induced by V' is the graph (V', E') with $E' = \{\{u, v\} \in E \mid u, v \in V'\}$.

We say that G' is an induced subgraph of $G = (V, E)$ if G' is the subgraph of G induced by V' for any set of vertices $V' \subseteq V$.

German: induzierter Teilgraph (eines Graphen)

Induced Subgraphs (2)

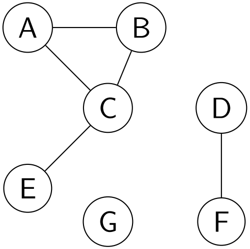
Definition (induced subgraph)

Let $G = (N, A)$ be a digraph, and let $N' \subseteq N$.
The subgraph of G induced by N' is the digraph (N', A') with $A' = \{(u, v) \in A \mid u, v \in N'\}$.

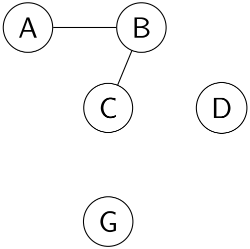
We say that G' is an induced subgraph of $G = (N, A)$ if G' is the subgraph of G induced by N' for any set of nodes $N' \subseteq N$.

German: induzierter Teilgraph (eines gerichteten Graphen)

Induced Subgraphs – Example

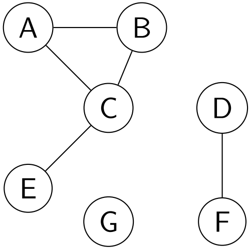


graph (V, E)

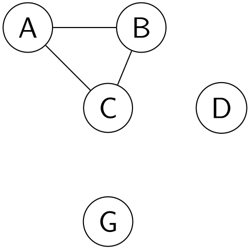


Is this an induced subgraph?

Induced Subgraphs – Example



graph (V, E)



This is an induced subgraph.

Induced Subgraphs – Discussion

- Induced subgraphs are subgraphs.
- They are the largest (in terms of the set of edges) subgraphs with any given set of vertices.
- A typical example is a subgraph induced by one connected component of a graph.
- The subgraphs induced by the connected components of a forest are trees.

Counting Subgraphs

- How many subgraphs does a graph (V, E) have?
- How many induced subgraph does a graph (V, E) have?

For the second question, the answer is $2^{|V|}$.

The first question is in general not easy to answer because vertices and edges of a subgraph cannot be chosen independently.

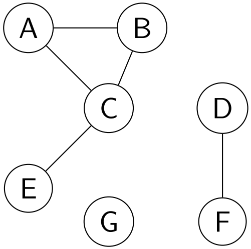
Example (subgraphs of a complete graph)

A complete graph with n vertices (i.e., with all possible $\binom{n}{2}$ edges) has $\sum_{k=0}^{n} \binom{n}{k} 2^{\binom{k}{2}}$ subgraphs. (Why?)

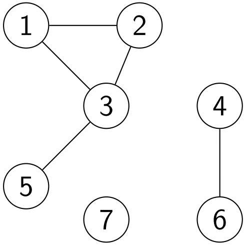
for $n = 10$: 1024 induced subgraphs, 35883905263781 subgraphs

C4.2 Isomorphism

Motivation



graph (V, E)



graph (V', E')

What is the difference between these graphs?

Isomorphism

- In many cases, the "names" of the vertices of a graph do not have any particular semantic meaning.
- Often, we care about the structure of the graph, i.e., the relationship between the vertices and edges, but not what we call the different vertices.
- This is captured by the concept of isomorphism.

Isomorphism – Definition

Definition (Isomorphism)

Let $G = (V, E)$ and $G' = (V', E')$ be graphs.

An isomorphism from G to G' is a bijective function $\sigma : V \to V'$ such that for all $u, v \in V$:

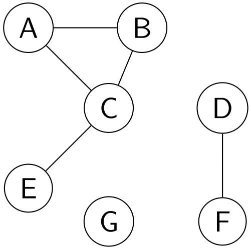
$$\{u, v\} \in E \quad \text{iff} \quad \{\sigma(u), \sigma(v)\} \in E'.$$

If there exists an isomorphism from G to G', we say that they are isomorphic, in symbols $G \cong G'$.

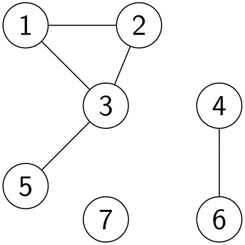
German: Isomorphismus, isomorph

- derives from Ancient Greek for "equally shaped/formed"
- analogous definition for digraphs omitted

Isomorphism – Example



graph (V, E)



graph (V', E')

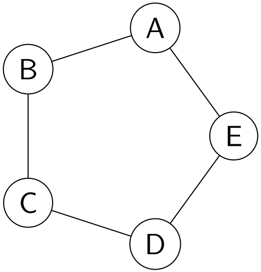
- $\sigma = \{\mathsf{A} \mapsto 1, B \mapsto 2, C \mapsto 3, D \mapsto 4, E \mapsto 5, F \mapsto 6, G \mapsto 7\}$
- for example: $\{\mathsf{A}, \mathsf{B}\} \in E$ and $\{\sigma(\mathsf{A}), \sigma(\mathsf{B})\} = \{1, 2\} \in E'$
- for example: $\{\mathsf{A}, \mathsf{D}\} \notin E$ and $\{\sigma(\mathsf{A}), \sigma(\mathsf{D})\} = \{1, 4\} \notin E'$

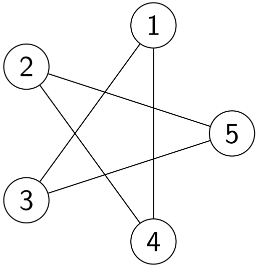
Isomorphism – Discussion

- The identity function is an isomorphism.
- The inverse of an isomorphism is an isomorphism.
- The composition of two isomorphisms is an isomorphism (when defined over matching sets of vertices)

It follows that being isomorphic is an equivalence relation.

Isomorphic or Not? (1)

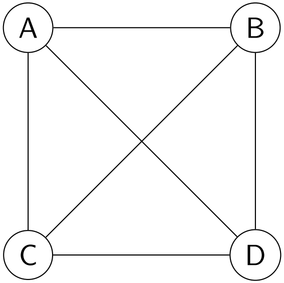


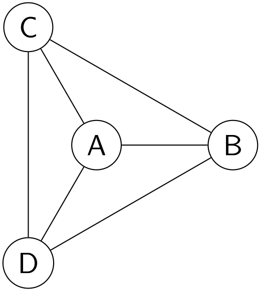


isomorphic

$$\sigma = \{\mathsf{A} \mapsto 1, \mathsf{B} \mapsto 3, \mathsf{C} \mapsto 5, \mathsf{D} \mapsto 2, \mathsf{E} \mapsto 4\}$$

Isomorphic or Not? (2)



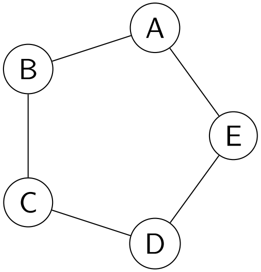


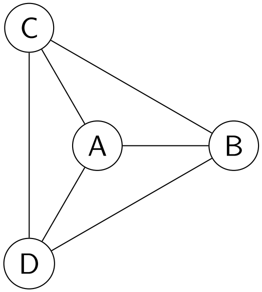
isomorphic

$\rightsquigarrow$ in fact, the same graph!

$\sigma = \{A \mapsto A, B \mapsto B, C \mapsto C, D \mapsto D\}$

Isomorphic or Not? (3)

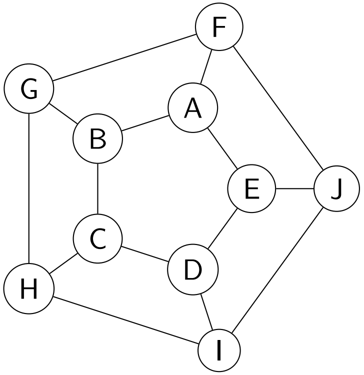


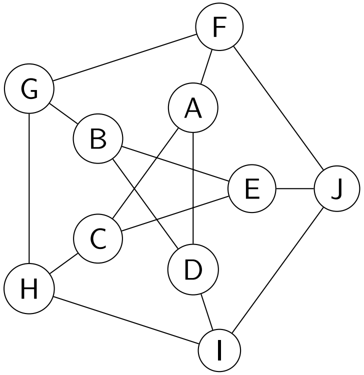


not isomorphic

There does not even exist a bijection between the vertices.

Isomorphic or Not? (4)



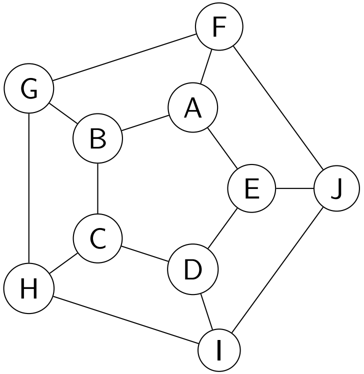


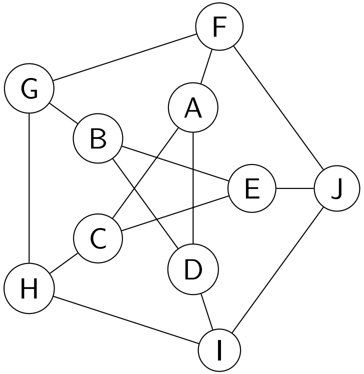
isomorphic or not?

Proving and Disproving Isomorphism

- To prove that two graphs are isomorphic, it suffices to state an isomorphism and verify that it has the required properties.
- To prove that two graphs are not isomorphic, we must rule out all possible bijections.
  - With n vertices, there are $n!$ bijections.
  - example $n = 10$: $10! = 3628800$
- A common disproof idea is to identify a graph invariant, i.e., a property of a graph that must be the same in isomorphic graphs, and show that it differs.
  - examples: number of vertices, number of edges, maximum/minimum degree, sorted sequence of all degrees, number of connected components

Isomorphic or Not? (5)





not isomorphic

- The left graph has cycles of length 4 (e.g., $\langle A, B, G, F, A\rangle$).
- The right graph does not.
- Having a cycle of a given length is an invariant.

Scientific Pop Culture

- Determining if two graphs are isomorphic is an algorithmic problem that has been famously resistant to studying its complexity.
- For more than 40 years, we have not known if polynomial algorithms exist, and we also do not know if it belongs to the famous class of NP-complete problems.
- In 2015, László Babai announced an algorithm with quasi-polynomial (worse than polynomial, better than exponential) runtime.

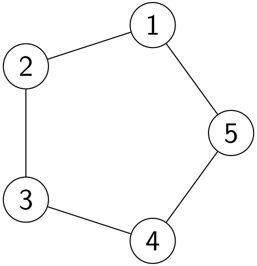
Further Reading

Martin Grohe, Pascal Schweitzer.
The Graph Isomorphism Problem.
Communications of the ACM 63(11):128–134, November 2020.
`https://dl.acm.org/doi/10.1145/3372123`

Symmetries, Automorphisms and Group Theory

- An isomorphism σ between a graph G and itself is called an automorphism or symmetry of G.
- For every graph, its symmetries are permutations of its vertices that form a mathematical structure called a group:
  - the identity function is a symmetry
  - the composition of two symmetries is a symmetry
  - the inverse of a symmetry is a symmetry

Automorphism Group of a Graph



What are the symmetries?

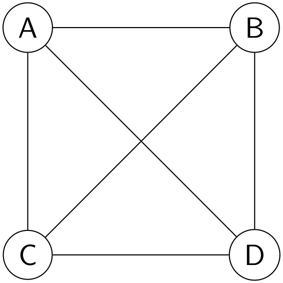
- one example is the rotation $\sigma_1 = \{1 \mapsto 2, 2 \mapsto 3, 3 \mapsto 4, 4 \mapsto 5, 5 \mapsto 1\}$
- another example is the reflection $\sigma_2 = \{1 \mapsto 5, 2 \mapsto 4, 3 \mapsto 3, 4 \mapsto 2, 5 \mapsto 1\}$
- There are 10 symmetries in total, and they are all generated by (= can be composed from) σ_1 and σ_2.

C4.3 Planarity and Minors

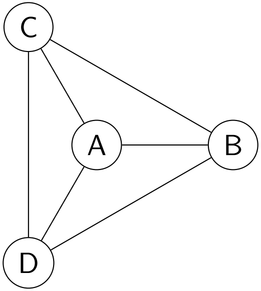
Planarity

- We often draw graphs as 2-dimensional pictures.
- When we do so, we usually try to draw them in such a way that different edges do not cross.
- This often makes the picture neater and the edges easier to visualize.
- A picture of a graph with no edge crossings is called a planar embedding.
- A graph for which a planar embedding exists is called planar.

Planar Embeddings – Example



not a planar embedding



planar embedding

The complete graph over 4 vertices is planar.

Planar Graphs

Definition (planar)

A graph $G = (V, E)$ is called planar if there exists a planar embedding of G, i.e., a picture of G in the Euclidean plane in which no two edges intersect.

German: planar

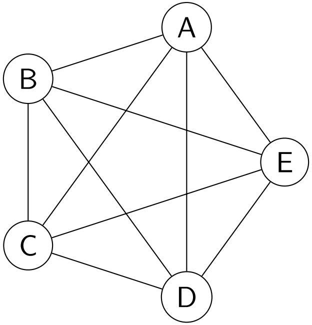
Notes:

- We do not formally define planar embeddings, as this is nontrivial and not necessary for our discussion.
- In general, we may draw edges as arbitrary curves.
- However, it is possible to show that a graph has a planar embedding iff it has a planar embedding where all edges are straight lines.

Planar Graphs – Discussion

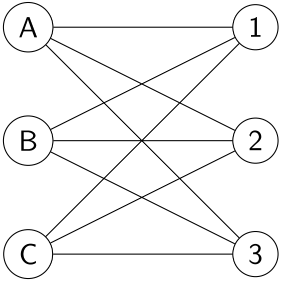
- Planar graphs arise in many practical applications.
- Many computational problems are easier for planar graphs.
  - For example, every planar graph can be coloured with at most 4 colours (i.e., we can assign one of four colours to each vertex such that two neighbours always have different colours).
- For this reason, planarity is of great practical interest.
- How can we recognize that a graph is planar?
- How can we prove that a graph is not planar?

Planar Graphs – Counterexample (1)



The complete graph K_5 over 5 vertices is not planar.
(We do not prove this result.)

Planar Graphs – Counterexample (2)



The complete bipartite graph $K_{3,3}$ over $3 + 3$ vertices is not planar.
(We do not prove this result.)

Non-Planarity in General

- The two non-planar graphs K_5 and $K_{3,3}$ are special: they are the smallest non-planar graphs.
- In fact, something much more powerful holds: a graph is planar iff it does not contain K_5 or $K_{3,3}$.
- The notion of containment we need here is related to the notion of subgraphs that we introduced, but a bit more complex. We will discuss it next.

Edge Contraction

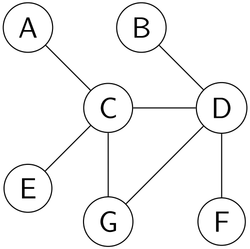
We say that $G' = (V', E')$ can be obtained from graph $G = (V, E)$ by contracting the edge $\{u, v\} \in E$ if

- $V' = (V \setminus \{u, v\}) \cup \{uv\}$, where $uv \notin V$ is a new vertex
- $E' = \{e \in E \mid e \cap \{u, v\} = \emptyset\} \cup$
  $\{\{uv, w\} \mid w \in V \setminus \{u, v\}, \{u, w\} \in E \text{ or } \{v, w\} \in E\}$.

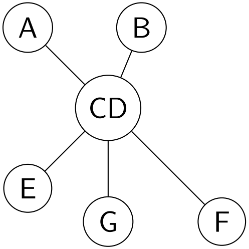
In words, we combine the vertices u and v
(which must be connected by an edge) into a single vertex uv.

The neighbours of uv are the union of the neighbours of u
and the neighbours of v (excluding u and v themselves).

Edge Contraction – Example



graph (V, E)



after contracting $\{\mathrm{C}, \mathrm{D}\}$

Minor

Definition (minor)

We say that a graph G' is a minor of a graph G if it can be obtained from G through a sequence of transformations of the following kind:

1. remove a vertex (of degree 0) from the graph
2. remove an edge from the graph
3. contract an edge in the graph

German: Minor (plural: Minoren)

Notes:

- If we only allowed the first two transformations, we would obtain the regular subgraph relationship.
- It follows that every subgraph is a minor, but the opposite is not true in general.

Wagner's Theorem

Theorem (Wagner's Theorem)

A graph is planar iff it does not contain K_5 or $K_{3,3}$ as a minor.

German: Satz von Wagner

Note: There exist linear algorithms for testing planarity.

Minor-Hereditary Properties

- Being planar is what is called a minor-hereditary property: if G is planar, then all its minors are also planar.
- There exist many other important such properties.
- One example is acyclicity.

How could one prove that a property is minor-hereditary?

The Graph Minor Theorem

Theorem (Graph minor theorem)

Let Π be a minor-hereditary property of graphs.

Then there exists a finite set of forbidden minors $F(\Pi)$ such that the following result holds:

A graph has property Π iff it does not have any graph from $F(\Pi)$ as a minor.

German: Minorentheorem

Examples:

- the forbidden minors for planarity are K_5 and $K_{3,3}$
- the (only) forbidden minor for acyclicity is K_3, the complete graph with 3 vertices (a.k.a. the 3-cycle graph)

Remarks on the Graph Minor Theorem (1)

- The graph minor theorem is also known as the Robertson-Seymour theorem.
- It was proved by Robertson and Seymour in a series of 20 papers between 1983–2004, totalling 500+ pages.
- It is one of the most important results in graph theory.

Remarks on the Graph Minor Theorem (2)

- In principle, for every fixed graph H, we can test if H is a minor of a graph G in polynomial time in the size of G.
- This implies that every minor-hereditary property can be tested in polynomial time.
- However, the constant factors involved in the known general algorithms for testing minors (which depend on $|H|$) are so astronomically huge as to make them infeasible in practice.

Discrete Mathematics in Computer Science

D1. Syntax and Semantics of Propositional Logic

Malte Helmert, Gabriele Röger

University of Basel

November 19/24, 2025

D1.1 Introduction to Formal Logic

Why Logic?

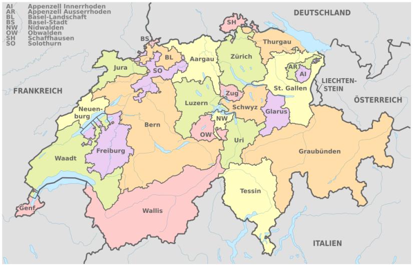
- formalizing mathematics
  - What is a true statement?
  - What is a valid proof?
  - What can and cannot be proved?
- basis of many tools in computer science
  - design of digital circuits
  - semantics of databases; query optimization
  - meaning of programming languages
  - verification of safety-critical hardware/software
  - knowledge representation in artificial intelligence
  - logic-based programming languages (e.g. Prolog)
  - ...

Application: Logic Programming I

Declarative approach: Describe what to accomplish,
not how to accomplish it.

Example (Map Coloring)

Color each region in a map with a limited number of colors
so that no two adjacent regions have the same color.



CC BY-SA 3.0 Wikimedia Commons (TUBS)

This is a hard problem!

Application: Logic Programming II

Prolog program

```
color(red). color(blue). color(green). color(yellow).

differentColor(ColorA, ColorB) :-
    color(ColorA), color(ColorB),
    ColorA \= ColorB.

switzerland(AG, AI, AR, BE, BL, BS, FR, GE, GL, GR,
            JU, LU, NE, NW, OW, SG, SH, SO, SZ, TG,
            TI, UR, VD, VS, ZG, ZH) :-
    differentColor(AG, BE), differentColor(AG, BL),
    ...
    differentColor(VD, VS), differentColor(ZH, ZG).
```

What Logic is About

General Question:

- Given some knowledge about the world (a knowledge base)
- what can we derive from it?
- And on what basis may we argue?

$\rightsquigarrow$ logic

Goal: "mechanical" proofs

- formal "game with letters"
- detached from a concrete meaning

Running Example

What's the secret of your long life?

I am on a strict diet: If I don't drink beer to a meal, then I always eat fish. Whenever I have fish and beer with the same meal, I abstain from ice cream. When I eat ice cream or don't drink beer, then I never touch fish.

Exercise from U. Schöning: Logik für Informatiker
Picture courtesy of graur razvan ionut / FreeDigitalPhotos.net

Propositional Logic

Propositional logic is a simple logic without numbers or objects.

Building blocks of propositional logic:

- propositions are statements that can be either true or false
- atomic propositions cannot be split into subpropositions
- logical connectives connect propositions to form new ones

German: Aussagenlogik, Aussage, atomare Aussage, Junktoren/logische Verknüpfungen

Examples for Building Blocks

If I don't drink beer to a meal, then I always eat fish. Whenever I have fish and beer with the same meal, I abstain from ice cream. When I eat ice cream or don't drink beer, then I never touch fish.

- Every sentence is a proposition that consists of subpropositions (e. g., "eat ice cream or don't drink beer").
- atomic propositions "drink beer", "eat fish", "eat ice cream"
- logical connectives "and", "or", negation, "if, then"

Exercise from U. Schöning: Logik für Informatiker
Picture courtesy of graur razvan ionut / FreeDigitalPhotos.net

Examples for Building Blocks

If I don't drink beer to a meal, then I always eat fish. Whenever I have fish and beer with the same meal, I abstain from ice cream. When I eat ice cream or don't drink beer, then I never touch fish.

- Every sentence is a proposition that consists of subpropositions (e. g., "eat ice cream or don't drink beer").
- atomic propositions "drink beer", "eat fish", "eat ice cream"
- logical connectives "and", "or", negation, "if, then"

Exercise from U. Schöning: Logik für Informatiker
Picture courtesy of graur razvan ionut / FreeDigitalPhotos.net

Challenges with Natural Language

If I don't drink beer to a meal, then I always eat fish.
Whenever I have fish and beer with the same meal, I abstain from ice cream.
When I eat ice cream or don't drink beer, then I never touch fish.

- "irrelevant" information

Exercise from U. Schöning: Logik für Informatiker
Picture courtesy of graur razvan ionut / FreeDigitalPhotos.net

Challenges with Natural Language

If I don’t drink beer, then I eat fish.
Whenever I have fish and beer, I abstain from ice cream.
When I eat ice cream or don’t drink beer, then I never touch fish.

- “irrelevant” information
- different formulations for the same connective/proposition

Exercise from U. Schöning: Logik für Informatiker
Picture courtesy of graur razvan ionut / FreeDigitalPhotos.net

Challenges with Natural Language

If not DrinkBeer, then EatFish.
If EatFish and DrinkBeer,
then not EatIceCream.
If EatIceCream or not DrinkBeer,
then not EatFish.

- “irrelevant” information
- different formulations for the same connective/proposition

Exercise from U. Schöning: Logik für Informatiker
Picture courtesy of graur razvan ionut / FreeDigitalPhotos.net

What is Next?

- What are meaningful (well-defined) sequences of atomic propositions and connectives?
  "if then EatIceCream not or DrinkBeer and" not meaningful
  $\rightarrow$ syntax
- What does it mean if we say that a statement is true?
  Is "DrinkBeer and EatFish" true?
  $\rightarrow$ semantics
- When does a statement logically follow from another?
  Does "EatFish" follow from "if DrinkBeer, then EatFish"?
  $\rightarrow$ logical entailment

German: Syntax, Semantik, logische Folgerung

D1.2 Syntax of Propositional Logic

Syntax of Propositional Logic

Definition (Syntax of Propositional Logic)

Let A be a set of atomic propositions. The set of propositional formulas (over A) is inductively defined as follows:

- Every atom $a \in A$ is a propositional formula over A.
- If φ is a propositional formula over A, then so is its negation $\neg\varphi$.
- If φ and ψ are propositional formulas over A, then so is the conjunction $(\varphi \land \psi)$.
- If φ and ψ are propositional formulas over A, then so is the disjunction $(\varphi \lor \psi)$.

The implication $(\varphi \rightarrow \psi)$ is an abbreviation for $(\neg\varphi \lor \psi)$.
The biconditional $(\varphi \leftrightarrow \psi)$ is an abbrev. for $((\varphi \rightarrow \psi) \land (\psi \rightarrow \varphi))$.
German: atomare Aussage, aussagenlogische Formel, Atom, Negation, Konjunktion, Disjunktion, Implikation, Bikonditional

Syntax: Examples

Which of the following sequences of symbols are propositional formulas over the set of all possible letter sequences? Which kinds of formula are they (atom, conjunction, ...)?

- $(\mathsf{A} \land (\mathsf{B} \lor \mathsf{C}))$
- $\lnot(\land \mathsf{Rain} \lor \mathsf{StreetWet})$
- $\lnot(\mathsf{Rain} \lor \mathsf{StreetWet})$
- $((\mathsf{EatFish} \land \mathsf{DrinkBeer}) \rightarrow \lnot\mathsf{EatIceCream})$
- $\mathsf{Rain} \land \lnot\mathsf{Rain}$
- $\lnot(\mathsf{A} = \mathsf{B})$
- $(\mathsf{A} \land \lnot(\mathsf{B} \leftrightarrow)\mathsf{C})$
- $((\mathsf{A} \leq \mathsf{B}) \land \mathsf{C})$
- $(\mathsf{A} \lor \lnot(\mathsf{B} \leftrightarrow \mathsf{C}))$
- $((\mathsf{A}_1 \land \mathsf{A}_2) \lor \lnot(\mathsf{A}_3 \leftrightarrow \mathsf{A}_2))$

D1.3 Semantics of Propositional Logic

Meaning of Propositional Formulas?

So far propositional formulas are only symbol sequences without any meaning.

For example, what does this mean:
$((\text{EatFish} \land \text{DrinkBeer}) \rightarrow \neg\text{EatIceCream})$?

▷ We need semantics!

Semantics of Propositional Logic

Definition (Semantics of Propositional Logic)

A truth assignment (or interpretation) for a set of atomic propositions A is a function $\mathcal{I} : A \to \{0, 1\}$.

A propositional formula φ (over A) holds under $\mathcal{I}$ (written as $\mathcal{I} \models \varphi$) according to the following definition:

$$\begin{array}{lll l}
\mathcal{I} \models a & \text{iff} & \mathcal{I}(a) = 1 & (\text{for } a \in A) \\
\mathcal{I} \models \neg\varphi & \text{iff} & \text{not } \mathcal{I} \models \varphi & \\
\mathcal{I} \models (\varphi \land \psi) & \text{iff} & \mathcal{I} \models \varphi \text{ and } \mathcal{I} \models \psi & \\
\mathcal{I} \models (\varphi \lor \psi) & \text{iff} & \mathcal{I} \models \varphi \text{ or } \mathcal{I} \models \psi &
\end{array}$$

Question: should we define semantics of $(\varphi \to \psi)$ and $(\varphi \leftrightarrow \psi)$?

German: Wahrheitsbelegung/Interpretation, φ gilt unter $\mathcal{I}$

Semantics of Propositional Logic: Terminology

- For $\mathcal{I} \models \varphi$ we also say $\mathcal{I}$ is a model of φ and that φ is true under $\mathcal{I}$.
- If φ does not hold under $\mathcal{I}$, we write this as $\mathcal{I} \not\models \varphi$ and say that $\mathcal{I}$ is no model of φ and that φ is false under $\mathcal{I}$.
- Note: $\models$ is not part of the formula but part of the meta language (speaking about a formula).

German: $\mathcal{I}$ ist ein/kein Modell von φ; φ ist wahr/falsch unter $\mathcal{I}$; Metasprache

Exercise

Consider the set $A = \{\mathsf{X}, \mathsf{Y}, \mathsf{Z}\}$ of atomic propositions and formula $\varphi = (\mathsf{X} \land \lnot\mathsf{Y})$.

Specify an interpretation $\mathcal{I}$ for A with $\mathcal{I} \models \varphi$.

Semantics: Example (1)

$$A = \{\mathsf{DrinkBeer}, \mathsf{EatFish}, \mathsf{EatIceCream}\}$$
$$\mathcal{I} = \{\mathsf{DrinkBeer} \mapsto 1, \mathsf{EatFish} \mapsto 0, \mathsf{EatIceCream} \mapsto 1\}$$
$$\varphi = (\neg\mathsf{DrinkBeer} \rightarrow \mathsf{EatFish})$$

Do we have $\mathcal{I} \models \varphi$?

Semantics: Example (2)

Goal: prove $\mathcal{I} \models \varphi$.

Let us use the definitions we have seen:

$$\begin{aligned}\mathcal{I} \models \varphi \text{ iff } & \mathcal{I} \models (\neg \mathsf{DrinkBeer} \rightarrow \mathsf{EatFish}) \\ \text{iff } & \mathcal{I} \models (\neg\neg \mathsf{DrinkBeer} \vee \mathsf{EatFish}) \\ \text{iff } & \mathcal{I} \models \neg\neg \mathsf{DrinkBeer} \text{ or } \mathcal{I} \models \mathsf{EatFish}\end{aligned}$$

This means that if we want to prove $\mathcal{I} \models \varphi$, it is sufficient to prove

$$\mathcal{I} \models \neg\neg \mathsf{DrinkBeer}$$

or to prove

$$\mathcal{I} \models \mathsf{EatFish}.$$

We attempt to prove the first of these statements.

Semantics: Example (3)

New goal: prove $\mathcal{I} \models \neg\neg\mathrm{DrinkBeer}$.

We again use the definitions:

$$\begin{aligned}\mathcal{I} \models \neg\neg\mathrm{DrinkBeer} &\text{ iff not } \mathcal{I} \models \neg\mathrm{DrinkBeer}\\ &\text{ iff not not } \mathcal{I} \models \mathrm{DrinkBeer}\\ &\text{ iff } \mathcal{I} \models \mathrm{DrinkBeer}\\ &\text{ iff } \mathcal{I}(\mathrm{DrinkBeer}) = 1\end{aligned}$$

The last statement is true for our interpretation $\mathcal{I}$.

To write this up as a proof of $\mathcal{I} \models \varphi$,
we can go through this line of reasoning back-to-front,
starting from our assumptions and ending with the conclusion
we want to show.

Semantics: Example (4)

Let $\mathcal{I} = \{\mathsf{DrinkBeer} \mapsto 1, \mathsf{EatFish} \mapsto 0, \mathsf{EatIceCream} \mapsto 1\}$.

Proof that $\mathcal{I} \models (\neg\mathsf{DrinkBeer} \rightarrow \mathsf{EatFish})$:

1. We have $\mathcal{I} \models \mathsf{DrinkBeer}$ (uses defn. of $\models$ for atomic props. and fact $\mathcal{I}(\mathsf{DrinkBeer}) = 1$).
2. From (1), we get $\mathcal{I} \not\models \neg\mathsf{DrinkBeer}$ (uses defn. of $\models$ for negations).
3. From (2), we get $\mathcal{I} \models \neg\neg\mathsf{DrinkBeer}$ (uses defn. of $\models$ for negations).
4. From (3), we get $\mathcal{I} \models (\neg\neg\mathsf{DrinkBeer} \lor \psi)$ for all formulas ψ, in particular $\mathcal{I} \models (\neg\neg\mathsf{DrinkBeer} \lor \mathsf{EatFish})$ (uses defn. of $\models$ for disjunctions).
5. From (4), we get $\mathcal{I} \models (\neg\mathsf{DrinkBeer} \rightarrow \mathsf{EatFish})$ (uses defn. of "$\rightarrow$"). □

Summary

- propositional logic based on atomic propositions
- syntax defines what well-formed formulas are
- semantics defines when a formula is true
- interpretations are the basis of semantics

Discrete Mathematics in Computer Science

D2. Properties of Formulas and Equivalences

Malte Helmert, Gabriele Röger

University of Basel

November 26, 2025

D2.1 Properties of Propositional Formulas

D2.2 Equivalences

D2.1 Properties of Propositional Formulas

The Story So Far

- propositional logic based on atomic propositions
- syntax: which formulas are well-formed?
- semantics: when is a formula true?
- interpretations: important basis of semantics

Reminder: Syntax of Propositional Logic

Definition (Syntax of Propositional Logic)

Let A be a set of atomic propositions. The set of propositional formulas (over A) is inductively defined as follows:

- Every atom $a \in A$ is a propositional formula over A.
- If φ is a propositional formula over A, then so is its negation $\neg\varphi$.
- If φ and ψ are propositional formulas over A, then so is the conjunction $(\varphi \wedge \psi)$.
- If φ and ψ are propositional formulas over A, then so is the disjunction $(\varphi \vee \psi)$.

The implication $(\varphi \rightarrow \psi)$ is an abbreviation for $(\neg\varphi \vee \psi)$.
The biconditional $(\varphi \leftrightarrow \psi)$ is an abbrev. for $((\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi))$.

Reminder: Semantics of Propositional Logic

Definition (Semantics of Propositional Logic)

A truth assignment (or interpretation) for a set of atomic propositions A is a function $\mathcal{I} : A \to \{0, 1\}$.

A propositional formula φ (over A) holds under $\mathcal{I}$ (written as $\mathcal{I} \models \varphi$) according to the following definition:

$$\begin{array}{lll l}
\mathcal{I} \models a & \text{iff} & \mathcal{I}(a) = 1 & (\text{for } a \in A) \\
\mathcal{I} \models \neg\varphi & \text{iff} & \text{not } \mathcal{I} \models \varphi & \\
\mathcal{I} \models (\varphi \land \psi) & \text{iff} & \mathcal{I} \models \varphi \text{ and } \mathcal{I} \models \psi & \\
\mathcal{I} \models (\varphi \lor \psi) & \text{iff} & \mathcal{I} \models \varphi \text{ or } \mathcal{I} \models \psi &
\end{array}$$

Properties of Propositional Formulas

A propositional formula φ is

- satisfiable if φ has at least one model
- unsatisfiable if φ is not satisfiable
- valid (or a tautology) if φ is true under every interpretation
- falsifiable if φ is no tautology

German: erfüllbar, unerfüllbar, allgemeingültig/eine Tautologie, falsifizierbar

Examples

How can we show that a formula has one of these properties?

- Show that $(A \land B)$ is satisfiable.
  $\mathcal{I} = \{A \mapsto 1, B \mapsto 1\}$ (+ simple proof that $\mathcal{I} \models (A \land B)$)
- Show that $(A \land B)$ is falsifiable.
  $\mathcal{I} = \{A \mapsto 0, B \mapsto 1\}$ (+ simple proof that $\mathcal{I} \not\models (A \land B)$)
- Show that $(A \land B)$ is not valid.
  Follows directly from falsifiability.
- Show that $(A \land B)$ is not unsatisfiable.
  Follows directly from satisfiability.

So far all proofs by specifying one interpretation.

How to prove that a given formula is valid/unsatisfiable/
not satisfiable/not falsifiable?

$\rightsquigarrow$ must consider all possible interpretations

Truth Tables

Evaluate for all possible interpretations
if they are models of the considered formula.

$\mathcal{I}(\mathsf{A})$	$\mathcal{I} \models \neg\mathsf{A}$
0	Yes
1	No

$\mathcal{I}(\mathsf{A})$	$\mathcal{I}(\mathsf{B})$	$\mathcal{I} \models (\mathsf{A} \land \mathsf{B})$
0	0	No
0	1	No
1	0	No
1	1	Yes

$\mathcal{I}(\mathsf{A})$	$\mathcal{I}(\mathsf{B})$	$\mathcal{I} \models (\mathsf{A} \lor \mathsf{B})$
0	0	No
0	1	Yes
1	0	Yes
1	1	Yes

Truth Tables in General

Similarly in the case where we consider a formula whose building blocks are themselves arbitrary unspecified formulas:

$\mathcal{I} \models \varphi$	$\mathcal{I} \models \psi$	$\mathcal{I} \models (\varphi \land \psi)$
No	No	No
No	Yes	No
Yes	No	No
Yes	Yes	Yes

Truth Tables for Properties of Formulas

Is $\varphi = ((A \rightarrow B) \lor (\neg B \rightarrow A))$ valid, unsatisfiable, ...?

$\mathcal{I}(A)$	$\mathcal{I}(B)$	$\mathcal{I} \models \neg B$	$\mathcal{I} \models (A \rightarrow B)$	$\mathcal{I} \models (\neg B \rightarrow A)$	$\mathcal{I} \models \varphi$
0	0	Yes	Yes	No	Yes
0	1	No	Yes	Yes	Yes
1	0	Yes	No	Yes	Yes
1	1	No	Yes	Yes	Yes

Connection Between Formula Properties and Truth Tables

A propositional formula φ is

- satisfiable if φ has at least one model
  $\rightsquigarrow$ result in at least one row is "Yes"
- unsatisfiable if φ is not satisfiable
  $\rightsquigarrow$ result in all rows is "No"
- valid (or a tautology) if φ is true under every interpretation
  $\rightsquigarrow$ result in all rows is "Yes"
- falsifiable if φ is no tautology
  $\rightsquigarrow$ result in at least one row is "No"

Main Disadvantage of Truth Tables

How big is a truth table with n atomic propositions?

1	2 interpretations (rows)
2	4 interpretations (rows)
3	8 interpretations (rows)
n	??? interpretations

Some examples: $2^{10} = 1024$, $2^{20} = 1048576$, $2^{30} = 1073741824$

$\rightsquigarrow$ not viable for larger formulas; we need a different solution

- more on difficulty of satisfiability etc.: Theory of Computer Science course
- practical algorithms: Foundations of AI course

D2.2 Equivalences

Equivalent Formulas

Definition (Equivalence of Propositional Formulas)

Two propositional formulas φ and ψ over A are (logically) equivalent ($\varphi \equiv \psi$) if for all interpretations $\mathcal{I}$ for A it is true that $\mathcal{I} \models \varphi$ if and only if $\mathcal{I} \models \psi$.

German: logisch äquivalent

Equivalent Formulas: Example

$$((\varphi \lor \psi) \lor \chi) \equiv (\varphi \lor (\psi \lor \chi))$$

$\mathcal{I} \models \varphi$	$\mathcal{I} \models \psi$	$\mathcal{I} \models \chi$	$\mathcal{I} \models (\varphi \lor \psi)$	$\mathcal{I} \models (\psi \lor \chi)$	$\mathcal{I} \models ((\varphi \lor \psi) \lor \chi)$	$\mathcal{I} \models (\varphi \lor (\psi \lor \chi))$
No	No	No	No	No	No	No
No	No	Yes	No	Yes	Yes	Yes
No	Yes	No	Yes	Yes	Yes	Yes
No	Yes	Yes	Yes	Yes	Yes	Yes
Yes	No	No	Yes	No	Yes	Yes
Yes	No	Yes	Yes	Yes	Yes	Yes
Yes	Yes	No	Yes	Yes	Yes	Yes
Yes	Yes	Yes	Yes	Yes	Yes	Yes

Some Equivalences (1)

$$(\varphi \land \varphi) \equiv \varphi$$
$$(\varphi \lor \varphi) \equiv \varphi \qquad \text{(idempotence)}$$

$$(\varphi \land \psi) \equiv (\psi \land \varphi)$$
$$(\varphi \lor \psi) \equiv (\psi \lor \varphi) \qquad \text{(commutativity)}$$

$$((\varphi \land \psi) \land \chi) \equiv (\varphi \land (\psi \land \chi))$$
$$((\varphi \lor \psi) \lor \chi) \equiv (\varphi \lor (\psi \lor \chi)) \qquad \text{(associativity)}$$

German: Idempotenz, Kommutativität, Assoziativität

Some Equivalences (2)

$$(\varphi \land (\varphi \lor \psi)) \equiv \varphi$$

$$(\varphi \lor (\varphi \land \psi)) \equiv \varphi \qquad \text{(absorption)}$$

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

$$(\varphi \lor (\psi \land \chi)) \equiv ((\varphi \lor \psi) \land (\varphi \lor \chi)) \qquad \text{(distributivity)}$$

German: Absorption, Distributivität

Some Equivalences (3)

$\neg\neg\varphi \equiv \varphi$	(double negation)
$\neg(\varphi \land \psi) \equiv (\neg\varphi \lor \neg\psi)$	
$\neg(\varphi \lor \psi) \equiv (\neg\varphi \land \neg\psi)$	(De Morgan's rules)
$(\varphi \lor \psi) \equiv \varphi$ if φ tautology	
$(\varphi \land \psi) \equiv \psi$ if φ tautology	(tautology rules)
$(\varphi \lor \psi) \equiv \psi$ if φ unsatisfiable	
$(\varphi \land \psi) \equiv \varphi$ if φ unsatisfiable	(unsatisfiability rules)

German: Doppelnegation, De Morgansche Regeln, Tautologieregeln, Unerfüllbarkeitsregeln

Substitution Theorem

Theorem (Substitution Theorem)

Let φ and φ' be equivalent propositional formulas over A.
Let ψ be a propositional formula with (at least) one occurrence of the subformula φ.

Then ψ is equivalent to ψ', where ψ' is constructed from ψ by replacing an occurrence of φ in ψ with φ'.

German: Ersetzbarkeitstheorem

(without proof)

Application of Equivalences: Example

$$
\begin{aligned}
(P \land (Q \lor \lnot P)) &\equiv ((P \land Q) \lor (P \land \lnot P)) && \text{(distributivity)} \\
&\equiv ((P \land \lnot P) \lor (P \land Q)) && \text{(commutativity)} \\
&\equiv (P \land Q) && \text{(unsatisfiability rule)}
\end{aligned}
$$

Discrete Mathematics in Computer Science

D3. Normal Forms and Logical Consequence

Malte Helmert, Gabriele Röger

University of Basel

December 1/3, 2025

D3.1 Simplified Notation

Parentheses

Associativity:

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

- Placement of parentheses for a conjunction of conjunctions does not influence whether an interpretation is a model.
- ditto for disjunctions of disjunctions

$\rightsquigarrow$ can omit parentheses and treat this as if parentheses placed arbitrarily

- Example: $(A_1 \land A_2 \land A_3 \land A_4)$ instead of $((A_1 \land (A_2 \land A_3)) \land A_4)$
- Example: $(\neg A \lor (B \land C) \lor D)$ instead of $((\neg A \lor (B \land C)) \lor D)$

Parentheses

Does this mean we can always omit all parentheses and assume an arbitrary placement? $\rightsquigarrow$ No!

$$((\varphi \land \psi) \lor \chi) \not\equiv (\varphi \land (\psi \lor \chi))$$

What should $\varphi \land \psi \lor \chi$ mean?

Placement of Parentheses by Convention

Often parentheses can be dropped in specific cases and an implicit placement is assumed:

- $\neg$ binds more strongly than $\wedge$
- $\wedge$ binds more strongly than $\vee$
- $\vee$ binds more strongly than $\rightarrow$ or $\leftrightarrow$

$\rightsquigarrow$ cf. PEMDAS/"Punkt vor Strich"

Example

$A \vee \neg C \wedge B \rightarrow A \vee \neg D$ stands for $((A \vee (\neg C \wedge B)) \rightarrow (A \vee \neg D))$

- often harder to read
- error-prone

$\rightsquigarrow$ not used in this course

Short Notations for Conjunctions and Disjunctions

Short notation for addition:

$$\sum_{i=1}^{n} x_i = x_1 + x_2 + \cdots + x_n$$
$$\sum_{x \in \{x_1,\dots,x_n\}} x = x_1 + x_2 + \cdots + x_n$$

Analogously:

$$\bigwedge_{i=1}^{n} \varphi_i = (\varphi_1 \land \varphi_2 \land \cdots \land \varphi_n)$$
$$\bigvee_{i=1}^{n} \varphi_i = (\varphi_1 \lor \varphi_2 \lor \cdots \lor \varphi_n)$$
$$\bigwedge_{\varphi \in X} \varphi = (\varphi_1 \land \varphi_2 \land \cdots \land \varphi_n)$$
$$\bigvee_{\varphi \in X} \varphi = (\varphi_1 \lor \varphi_2 \lor \cdots \lor \varphi_n)$$
$$\text{for } X = \{\varphi_1, \dots, \varphi_n\}$$

Short Notation: Corner Cases

Is $\mathcal{I} \models \psi$ true for

$$\psi = \bigwedge_{\varphi \in X} \varphi \text{ and } \psi = \bigvee_{\varphi \in X} \varphi$$

if $X = \emptyset$ or $X = \{\chi\}$?

convention:

- $\bigwedge_{\varphi \in \emptyset} \varphi$ is a tautology.
- $\bigvee_{\varphi \in \emptyset} \varphi$ is unsatisfiable.
- $\bigwedge_{\varphi \in \{\chi\}} \varphi = \bigvee_{\varphi \in \{\chi\}} \varphi = \chi$

$\leadsto$ Why?

Exercise

Express $\bigwedge_{i=1}^{2} \bigvee_{j=1}^{3} \varphi_{ij}$ without $\bigwedge$ and $\bigvee$.

D3.2 Normal Forms

Why Normal Forms?

- A normal form is a representation with certain syntactic restrictions.
- condition for reasonable normal form: every formula must have a logically equivalent formula in normal form
- advantages:
  - can restrict proofs to formulas in normal form
  - can define algorithms to work only for formulas in normal form

German: Normalform

Negation Normal Form

Definition (Negation Normal Form)

A formula is in negation normal form (NNF) if it does not contain the abbreviations $\rightarrow$ and $\leftrightarrow$ and if it contains no negation symbols except possibly directly in front of atomic propositions.

German: Negationsnormalform

Example

$((\neg P \lor (R \land Q)) \land (P \lor \neg S))$ is in NNF.
$(P \land \neg(Q \lor R))$ is not in NNF.

Construction of NNF

Algorithm to Construct NNF

1. Replace abbreviation $\leftrightarrow$ by its definition (($\leftrightarrow$)-elimination).
   $\rightsquigarrow$ formula structure: only $\neg$, $\lor$, $\land$, $\rightarrow$
2. Replace abbreviation $\rightarrow$ by its definition (($\rightarrow$)-elimination).
   $\rightsquigarrow$ formula structure: only $\neg$, $\lor$, $\land$
3. Repeatedly apply double negation and De Morgan rules until no rules match any more ("move negations inside"):
   - Replace $\neg\neg\varphi$ by φ.
   - Replace $\neg(\varphi \land \psi)$ by $(\neg\varphi \lor \neg\psi)$.
   - Replace $\neg(\varphi \lor \psi)$ by $(\neg\varphi \land \neg\psi)$.

   $\rightsquigarrow$ formula structure: only atoms, negated atoms, $\lor$, $\land$

Constructing NNF: Example

Construction of Negation Normal Form

Given: $\varphi = (((P \land \neg Q) \lor R) \rightarrow (P \lor \neg(S \lor T)))$

$$\begin{aligned}
\varphi &\equiv (\neg((P \land \neg Q) \lor R) \lor P \lor \neg(S \lor T)) && \text{[Step 2]}\\
&\equiv ((\neg(P \land \neg Q) \land \neg R) \lor P \lor \neg(S \lor T)) && \text{[Step 3]}\\
&\equiv (((\neg P \lor \neg\neg Q) \land \neg R) \lor P \lor \neg(S \lor T)) && \text{[Step 3]}\\
&\equiv (((\neg P \lor Q) \land \neg R) \lor P \lor \neg(S \lor T)) && \text{[Step 3]}\\
&\equiv (((\neg P \lor Q) \land \neg R) \lor P \lor (\neg S \land \neg T)) && \text{[Step 3]}
\end{aligned}$$

Literals, Clauses and Monomials

- A literal is an atomic proposition or the negation of an atomic proposition (e. g., A and $\neg A$).
- A clause is a disjunction of literals (e. g., $(Q \lor \neg P \lor \neg S \lor R)$).
- A monomial is a conjunction of literals (e. g., $(Q \land \neg P \land \neg S \land R)$).

The terms clause and monomial are also used for the corner case with only one literal.

German: Literal, Klausel, Monom

Terminology: Examples

Examples

- $(\neg Q \land R)$ is a monomial
- $(P \lor \neg Q)$ is a clause
- $((P \lor \neg Q) \land P)$ is neither literal nor clause nor monomial
- $\neg P$ is a literal, a clause and a monomial
- $(P \rightarrow Q)$ is neither literal nor clause nor monomial
  (but $(\neg P \lor Q)$ is a clause!)
- $(P \lor P)$ is a clause, but not a literal or monomial
- $\neg\neg P$ is neither literal nor clause nor monomial

Conjunctive Normal Form

Definition (Conjunctive Normal Form)

A formula is in conjunctive normal form (CNF) if it is a conjunction of clauses, i. e., if it has the form

$$\bigwedge_{i=1}^{n} \bigvee_{j=1}^{m_i} L_{ij}$$

with $n, m_i > 0$ (for $1 \leq i \leq n$), where the L_{ij} are literals.

German: konjunktive Normalform (KNF)

Example

$((\neg P \lor Q) \land R \land (P \lor \neg S))$ is in CNF.

Disjunctive Normal Form

Definition (Disjunctive Normal Form)

A formula is in disjunctive normal form (DNF) if it is a disjunction of monomials, i. e., if it has the form

$$\bigvee_{i=1}^{n} \bigwedge_{j=1}^{m_i} L_{ij}$$

with $n, m_i > 0$ (for $1 \leq i \leq n$), where the L_{ij} are literals.

German: disjunktive Normalform (DNF)

Example

$((\neg P \land Q) \lor R \lor (P \land \neg S))$ is in DNF.

NNF, CNF and DNF: Examples

Which of the following formulas are in NNF?
Which are in CNF? Which are in DNF?

- $((P \lor \neg Q) \land P)$ is in NNF and CNF
- $((R \lor Q) \land P \land (R \lor S))$ is in NNF and CNF
- $(P \lor (\neg Q \land R))$ is in NNF and DNF
- $(P \lor \neg\neg Q)$ is in none of the normal forms
- $(P \rightarrow \neg Q)$ is in none of the normal forms,
  but is in all three after expanding $\rightarrow$
- $((P \lor \neg Q) \rightarrow P)$ is in none of the normal forms
- P is in NNF, CNF and DNF

Construction of CNF (and DNF)

Algorithm to Construct CNF

First, convert to NNF (steps 1–3).

$\rightsquigarrow$ formula structure: only literals, $\lor$, $\land$

4. Repeatedly apply distributivity or commutativity + distributivity to distribute $\lor$ over $\land$:
   - Replace $(\varphi \lor (\psi \land \chi))$ by $((\varphi \lor \psi) \land (\varphi \lor \chi))$.
   - Replace $((\psi \land \chi) \lor \varphi)$ by $((\psi \lor \varphi) \land (\chi \lor \varphi))$.

   $\rightsquigarrow$ formula structure: CNF
5. optionally: Simplify the formula at the end or at intermediate steps (e. g., with idempotence).

Note: For DNF, swap the roles of $\land$ and $\lor$ in Step 4.

Constructing CNF: Example

Construction of Conjunctive Normal Form

Given: $\varphi = (((\mathsf{P} \land \neg \mathsf{Q}) \lor \mathsf{R}) \rightarrow (\mathsf{P} \lor \neg(\mathsf{S} \lor \mathsf{T})))$

$$\begin{aligned}
\varphi &\equiv (((\neg \mathsf{P} \lor \mathsf{Q}) \land \neg \mathsf{R}) \lor \mathsf{P} \lor (\neg \mathsf{S} \land \neg \mathsf{T})) && \text{[to NNF]} \\
&\equiv ((\neg \mathsf{P} \lor \mathsf{Q} \lor \mathsf{P} \lor (\neg \mathsf{S} \land \neg \mathsf{T})) \land \\
&\quad (\neg \mathsf{R} \lor \mathsf{P} \lor (\neg \mathsf{S} \land \neg \mathsf{T}))) && \text{[Step 4]} \\
&\equiv (\neg \mathsf{R} \lor \mathsf{P} \lor (\neg \mathsf{S} \land \neg \mathsf{T})) && \text{[Step 5]} \\
&\equiv ((\neg \mathsf{R} \lor \mathsf{P} \lor \neg \mathsf{S}) \land (\neg \mathsf{R} \lor \mathsf{P} \lor \neg \mathsf{T})) && \text{[Step 4]}
\end{aligned}$$

Construct DNF: Example

Construction of Disjunctive Normal Form

Given: $\varphi = (((P \land \neg Q) \lor R) \rightarrow (P \lor \neg(S \lor T)))$

$$\begin{aligned} \varphi &\equiv (((\neg P \lor Q) \land \neg R) \lor P \lor (\neg S \land \neg T)) && \text{[to NNF]} \\ &\equiv ((\neg P \land \neg R) \lor (Q \land \neg R) \lor P \lor (\neg S \land \neg T)) && \text{[Step 4]} \end{aligned}$$

Existence of an Equivalent Formula in Normal Form

Theorem

For every formula φ there is a logically equivalent formula in NNF, a logically equivalent formula in CNF and a logically equivalent formula in DNF.

- "There is a" always means "there is at least one". Otherwise we would write "there is exactly one".
- Intuition: algorithms to construct normal forms work with any given formula and only use equivalence rewriting.
- actual proof would use induction over structure of formula

Size of Normal Forms

- In the worst case, a logically equivalent formula in CNF or DNF can be exponentially larger than the original formula.
- Example: for $(x_1 \lor y_1) \land \cdots \land (x_n \lor y_n)$ there is no smaller logically equivalent formula in DNF than:
$$\bigvee_{S \in \mathcal{P}(\{1,\dots,n\})} \left(\bigwedge_{i \in S} x_i \land \bigwedge_{i \in \{1,\dots,n\} \setminus S} y_i \right)$$
- As a consequence, the construction of the CNF/DNF formula can take exponential time.
- For NNF, we can generate an equivalent formula in linear time if the original formula does not use $\leftrightarrow$.

More Theorems

Theorem

A formula in CNF is a tautology iff every clause is a tautology.

Theorem

A formula in DNF is satisfiable iff at least one of its monomials is satisfiable.

$\rightsquigarrow$ both proved easily with semantics of propositional logic

D3.3 Knowledge Bases

Knowledge Bases: Example

If not DrinkBeer, then EatFish.
If EatFish and DrinkBeer, then not EatIceCream.
If EatIceCream or not DrinkBeer, then not EatFish.

$$\begin{aligned} \text{KB} = \{&(\neg\text{DrinkBeer} \rightarrow \text{EatFish}),\\ &((\text{EatFish} \land \text{DrinkBeer}) \rightarrow \neg\text{EatIceCream}),\\ &((\text{EatIceCream} \lor \neg\text{DrinkBeer}) \rightarrow \neg\text{EatFish})\} \end{aligned}$$

Exercise from U. Schöning: Logik für Informatiker
Picture courtesy of graur razvan ionut / FreeDigitalPhotos.net

Models for Sets of Formulas

Definition (Model for Knowledge Base)

Let KB be a knowledge base over A,
i. e., a set of propositional formulas over A.

A truth assignment $\mathcal{I}$ for A is a model for KB (written: $\mathcal{I} \models \text{KB}$)
if $\mathcal{I}$ is a model for every formula $\varphi \in \text{KB}$.

German: Wissensbasis, Modell

Properties of Sets of Formulas

A knowledge base KB is

- satisfiable if KB has at least one model
- unsatisfiable if KB is not satisfiable
- valid (or a tautology) if every interpretation is a model for KB
- falsifiable if KB is no tautology

German: erfüllbar, unerfüllbar, gültig, gültig/eine Tautologie, falsifizierbar

Example I

Which of the properties does $\mathsf{KB} = \{(\mathsf{A} \land \neg\mathsf{B}), \neg(\mathsf{B} \lor \mathsf{A})\}$ have?

KB is unsatisfiable:
For every model $\mathcal{I}$ with $\mathcal{I} \models (\mathsf{A} \land \neg\mathsf{B})$ we have $\mathcal{I}(\mathsf{A}) = 1$.
This means $\mathcal{I} \models (\mathsf{B} \lor \mathsf{A})$ and thus $\mathcal{I} \not\models \neg(\mathsf{B} \lor \mathsf{A})$.

This directly implies that KB is falsifiable, not satisfiable and no tautology.

Example II

Which of the properties does

$$\begin{aligned}\mathrm{KB} = \{&(\neg \mathrm{DrinkBeer} \rightarrow \mathrm{EatFish}),\\ &((\mathrm{EatFish} \land \mathrm{DrinkBeer}) \rightarrow \neg \mathrm{EatIceCream}),\\ &((\mathrm{EatIceCream} \lor \neg \mathrm{DrinkBeer}) \rightarrow \neg \mathrm{EatFish})\} \text{ have?}\end{aligned}$$

- satisfiable, e. g. with $\mathcal{I} = \{\mathrm{EatFish} \mapsto 1, \mathrm{DrinkBeer} \mapsto 1, \mathrm{EatIceCream} \mapsto 0\}$
- thus not unsatisfiable
- falsifiable, e. g. with $\mathcal{I} = \{\mathrm{EatFish} \mapsto 0, \mathrm{DrinkBeer} \mapsto 0, \mathrm{EatIceCream} \mapsto 1\}$
- thus not valid

D3.4 Logical Consequences

Logical Consequences: Motivation

What's the secret of your long life?

I am on a strict diet: If I don't drink beer to a meal, then I always eat fish. Whenever I have fish and beer with the same meal, I abstain from ice cream. When I eat ice cream or don't drink beer, then I never touch fish.

Claim: the woman drinks beer to every meal.

How can we prove this?

Exercise from U. Schöning: Logik für Informatiker
Picture courtesy of graur razvan ionut/FreeDigitalPhotos.net

Logical Consequences

Definition (Logical Consequence)

Let KB be a set of formulas and φ a formula.

We say that KB logically implies φ (written as $\mathrm{KB} \models \varphi$) if all models of KB are also models of φ.

also: KB logically entails φ, φ logically follows from KB, φ is a logical consequence of KB

German: KB impliziert φ logisch, φ folgt logisch aus KB, φ ist logische Konsequenz von KB

Attention: the symbol $\models$ is "overloaded": $\mathrm{KB} \models \varphi$ vs. $\mathcal{I} \models \varphi$.

What if KB is unsatisfiable or the empty set?

Logical Consequences: Example

Let $\varphi = \text{DrinkBeer}$ and

$$\begin{aligned}\text{KB} = \{&(\neg\text{DrinkBeer} \rightarrow \text{EatFish}),\\ &((\text{EatFish} \land \text{DrinkBeer}) \rightarrow \neg\text{EatIceCream}),\\ &((\text{EatIceCream} \lor \neg\text{DrinkBeer}) \rightarrow \neg\text{EatFish})\}.\end{aligned}$$

Show: $\text{KB} \models \varphi$

Proof sketch.

Proof by contradiction: assume $\mathcal{I} \models \text{KB}$, but $\mathcal{I} \not\models \text{DrinkBeer}$.
Then it follows that $\mathcal{I} \models \neg\text{DrinkBeer}$.
Because $\mathcal{I}$ is a model of KB, we also have
$\mathcal{I} \models (\neg\text{DrinkBeer} \rightarrow \text{EatFish})$ and thus $\mathcal{I} \models \text{EatFish}$. (Why?)
With an analogous argumentation starting from
$\mathcal{I} \models ((\text{EatIceCream} \lor \neg\text{DrinkBeer}) \rightarrow \neg\text{EatFish})$
we get $\mathcal{I} \models \neg\text{EatFish}$ and thus $\mathcal{I} \not\models \text{EatFish}$. $\rightsquigarrow$ Contradiction!

Important Theorems about Logical Consequences

Theorem (Deduction Theorem)

$\mathsf{KB} \cup \{\varphi\} \models \psi$ *iff* $\mathsf{KB} \models (\varphi \rightarrow \psi)$

German: Deduktionssatz

Theorem (Contraposition Theorem)

$\mathsf{KB} \cup \{\varphi\} \models \neg\psi$ *iff* $\mathsf{KB} \cup \{\psi\} \models \neg\varphi$

German: Kontrapositionssatz

Theorem (Contradiction Theorem)

$\mathsf{KB} \cup \{\varphi\}$ *is unsatisfiable iff* $\mathsf{KB} \models \neg\varphi$

German: Widerlegungssatz

(without proof)

Discrete Mathematics in Computer Science

D4. Inference

Malte Helmert, Gabriele Röger

University of Basel

December 8, 2025

D4.1 Inference Rules and Calculi

Inference: Motivation

- up to now: proof of logical consequence with semantic arguments
- no general algorithm
- solution: produce formulas that are logical consequences of given formulas with syntactic inference rules
- advantage: mechanical method that can easily be implemented as an algorithm

Inference Rules

- Inference rules have the form

$$\frac{\varphi_1, \dots, \varphi_k}{\psi}.$$

- Meaning: "Every model of $\varphi_1, \dots, \varphi_k$ is a model of ψ."
- An axiom is an inference rule with $k = 0$.
- A set of inference rules is called a calculus or proof system.

German: Inferenzregel, Axiom, (der) Kalkül, Beweissystem

Some Inference Rules for Propositional Logic

Modus ponens $\dfrac{\varphi,\ (\varphi \rightarrow \psi)}{\psi}$

Modus tollens $\dfrac{\neg\psi,\ (\varphi \rightarrow \psi)}{\neg\varphi}$

$\wedge$-elimination $\dfrac{(\varphi \wedge \psi)}{\varphi}$ $\dfrac{(\varphi \wedge \psi)}{\psi}$

$\wedge$-introduction $\dfrac{\varphi,\ \psi}{(\varphi \wedge \psi)}$

$\vee$-introduction $\dfrac{\varphi}{(\varphi \vee \psi)}$

$\leftrightarrow$-elimination $\dfrac{(\varphi \leftrightarrow \psi)}{(\varphi \rightarrow \psi)}$ $\dfrac{(\varphi \leftrightarrow \psi)}{(\psi \rightarrow \varphi)}$

Derivation

Definition (Derivation)

A derivation or proof of a formula φ from a knowledge base KB is a sequence of formulas $\psi_1, \dots, \psi_k$ with

- $\psi_k = \varphi$ and
- for all $i \in \{1, \dots, k\}$:
  - $\psi_i \in \text{KB}$, or
  - ψ_i is the result of the application of an inference rule to elements from $\{\psi_1, \dots, \psi_{i-1}\}$.

German: Ableitung, Beweis

Derivation: Example

Example

Given: $\text{KB} = \{P, (P \rightarrow Q), (P \rightarrow R), ((Q \land R) \rightarrow S)\}$

Task: Find derivation of $(S \land R)$ from KB.

1. P (KB)
2. $(P \rightarrow Q)$ (KB)
3. Q (1, 2, Modus ponens)
4. $(P \rightarrow R)$ (KB)
5. R (1, 4, Modus ponens)
6. $(Q \land R)$ (3, 5, $\land$-introduction)
7. $((Q \land R) \rightarrow S)$ (KB)
8. S (6, 7, Modus ponens)
9. $(S \land R)$ (8, 5, $\land$-introduction)

Correctness and Completeness

Definition (Correctness and Completeness of a Calculus)

We write $\mathrm{KB} \vdash_C \varphi$ if there is a derivation of φ from KB in calculus C.
(If calculus C is clear from context, also only $\mathrm{KB} \vdash \varphi$.)

A calculus C is correct if for all KB and φ
$\mathrm{KB} \vdash_C \varphi$ implies $\mathrm{KB} \models \varphi$.

A calculus C is complete if for all KB and φ
$\mathrm{KB} \models \varphi$ implies $\mathrm{KB} \vdash_C \varphi$.

Consider calculus C, consisting of the derivation rules seen earlier.
Question: Is C correct?
Question: Is C complete?

German: korrekt, vollständig

D4.2 Summary

Summary (Consequence and Inference)

- knowledge base: set of formulas describing given information; satisfiable, valid etc. used like for individual formulas
- logical consequence KB $\models \varphi$ means that φ is true whenever (= in all models where) KB is true
- A logical consequence KB $\models \varphi$ allows to conclude that KB implies φ based on the semantics.
- A correct calculus supports such conclusions on the basis of purely syntactical derivations KB $\vdash \varphi$.

Further Topics

There are many aspects of propositional logic
that we do not cover in this course.

- resolution: a commonly used proof system for formulas in CNF
- other proof systems, for example tableaux proofs
- algorithms for model construction, such as the Davis-Putnam-Logemann-Loveland (DPLL) algorithm.

$\rightsquigarrow$ Foundations of AI course

Discrete Mathematics in Computer Science

D5. Syntax and Semantics of Predicate Logic

Malte Helmert, Gabriele Röger

University of Basel

December 8/10/15, 2025

Discrete Mathematics in Computer Science

December 8/10/15, 2025 — D5. Syntax and Semantics of Predicate Logic

D5.1 Syntax of Predicate Logic

Limits of Propositional Logic

Cannot be expressed well in propositional logic:

- "Everyone who does the exercises passes the exam."
- "If someone with administrator privileges presses 'delete', all data is gone."
- "Everyone has a mother."
- "If someone is the father of some person, the person is his child."

▷ need more expressive logic
$\rightsquigarrow$ predicate logic (a.k.a. first-order logic)

German: Prädikatenlogik (erster Stufe)

Syntax: Building Blocks

- Signatures define allowed symbols.
  analogy: atom set A in propositional logic
- Terms are associated with objects by the semantics.
  no analogy in propositional logic
- Formulas are associated with truth values (true or false) by the semantics.
  analogy: formulas in propositional logic

German: Signatur, Term, Formel

Signatures: Definition

Definition (Signature)

A signature (of predicate logic) is a 4-tuple $\mathcal{S} = \langle \mathcal{V}, \mathcal{C}, \mathcal{F}, \mathcal{P} \rangle$ consisting of the following four disjoint sets:

- a finite or countable set $\mathcal{V}$ of variable symbols
- a finite or countable set $\mathcal{C}$ of constant symbols
- a finite or countable set $\mathcal{F}$ of function symbols
- a finite or countable set $\mathcal{P}$ of predicate symbols (or relation symbols)

Every function symbol $\mathsf{f} \in \mathcal{F}$ and predicate symbol $\mathsf{P} \in \mathcal{P}$ has an associated arity $ar(\mathsf{f}), ar(\mathsf{P}) \in \mathbb{N}_1$ (number of arguments).

German: Variablen-, Konstanten-, Funktions-, Prädikat- und Relationssymbole; Stelligkeit

Signatures: Terminology and Conventions

terminology:

- k-ary (function or predicate) symbol: symbol s with arity $ar(\mathrm{s}) = k$.
- also: unary, binary, ternary

German: k-stellig, unär, binär, ternär

conventions (in this course):

- variable symbols written in *italics*, other symbols upright.
- predicate symbols begin with capital letter, other symbols with lower-case letters

Signatures: Examples

Example: Arithmetic

- $\mathcal{V} = \{x, y, z, x_1, x_2, x_3, \dots\}$
- $\mathcal{C} = \{\mathsf{zero}, \mathsf{one}\}$
- $\mathcal{F} = \{\mathsf{sum}, \mathsf{product}\}$
- $\mathcal{P} = \{\mathsf{Positive}, \mathsf{SquareNumber}\}$

$ar(\mathsf{sum}) = ar(\mathsf{product}) = 2$, $ar(\mathsf{Positive}) = ar(\mathsf{SquareNumber}) = 1$

Signatures: Examples

Example: Genealogy

- $\mathcal{V} = \{x, y, z, x_1, x_2, x_3, \dots\}$
- $\mathcal{C} = \{\text{roger-federer}, \text{lisa-simpson}\}$
- $\mathcal{F} = \emptyset$
- $\mathcal{P} = \{\text{Female}, \text{Male}, \text{Parent}\}$

$ar(\text{Female}) = ar(\text{Male}) = 1$, $ar(\text{Parent}) = 2$

Terms: Definition

Definition (Term)

Let $\mathcal{S} = \langle \mathcal{V}, \mathcal{C}, \mathcal{F}, \mathcal{P} \rangle$ be a signature.
A term (over $\mathcal{S}$) is inductively constructed according to the following rules:

- Every variable symbol $v \in \mathcal{V}$ is a term.
- Every constant symbol $\mathsf{c} \in \mathcal{C}$ is a term.
- If $t_1, \dots, t_k$ are terms and $\mathsf{f} \in \mathcal{F}$ is a function symbol with arity k, then $\mathsf{f}(t_1, \dots, t_k)$ is a term.

German: Term

examples:

- x_4
- lisa-simpson
- $\mathsf{sum}(x_3, \mathsf{product}(\mathsf{one}, x_5))$

Formulas: Definition

Definition (Formula)

For a signature $\mathcal{S} = \langle \mathcal{V}, \mathcal{C}, \mathcal{F}, \mathcal{P} \rangle$ the set of predicate logic formulas (over $\mathcal{S}$) is inductively defined as follows:

- If $t_1, \dots, t_k$ are terms (over $\mathcal{S}$) and $\mathsf{P} \in \mathcal{P}$ is a k-ary predicate symbol, then the atomic formula (or the atom) $\mathsf{P}(t_1, \dots, t_k)$ is a formula over $\mathcal{S}$.
- If t_1 and t_2 are terms (over $\mathcal{S}$), then the identity $(t_1 = t_2)$ is a formula over $\mathcal{S}$.
- If $x \in \mathcal{V}$ is a variable symbol and φ a formula over $\mathcal{S}$, then the universal quantification $\forall x\, \varphi$ and the existential quantification $\exists x\, \varphi$ are formulas over $\mathcal{S}$.

...

German: atomare Formel, Atom, Identität, Allquantifizierung, Existenzquantifizierung

Formulas: Definition

Definition (Formula)

For a signature $\mathcal{S} = \langle \mathcal{V}, \mathcal{C}, \mathcal{F}, \mathcal{P} \rangle$ the set of predicate logic formulas (over $\mathcal{S}$) is inductively defined as follows:

...

- If φ is a formula over $\mathcal{S}$, then so is its negation $\neg\varphi$.
- If φ and ψ are formulas over $\mathcal{S}$, then so are the conjunction $(\varphi \land \psi)$ and the disjunction $(\varphi \lor \psi)$.

German: Negation, Konjunktion, Disjunktion

Formulas: Examples

Examples: Arithmetic and Genealogy

- $\mathrm{Positive}(x_2)$
- $\forall x\,(\neg\mathrm{SquareNumber}(x) \lor \mathrm{Positive}(x))$
- $\exists x_3\,(\mathrm{SquareNumber}(x_3) \land \neg\mathrm{Positive}(x_3))$
- $\forall x\,(x = y)$
- $\forall x\,(\mathrm{sum}(x, x) = \mathrm{product}(x, \mathrm{one}))$
- $\forall x \exists y\,(\mathrm{sum}(x, y) = \mathrm{zero})$
- $\forall x \exists y\,(\mathrm{Parent}(y, x) \land \mathrm{Female}(y))$

Terminology: The symbols $\forall$ and $\exists$ are called quantifiers.

German: Quantoren

Abbreviations and Placement of Parentheses by Convention

abbreviations:

- $(\varphi \rightarrow \psi)$ is an abbreviation for $(\neg\varphi \lor \psi)$.
- $(\varphi \leftrightarrow \psi)$ is an abbreviation for $((\varphi \rightarrow \psi) \land (\psi \rightarrow \varphi))$.
- Sequences of the same quantifier can be abbreviated. For example:
  - $\forall x \forall y \forall z\, \varphi \rightsquigarrow \forall xyz\, \varphi$
  - $\exists x \exists y \exists z\, \varphi \rightsquigarrow \exists xyz\, \varphi$
  - $\forall w \exists x \exists y \forall z\, \varphi \rightsquigarrow \forall w \exists xy \forall z\, \varphi$

placement of parentheses by convention:

- analogous to propositional logic
- quantifiers $\forall$ and $\exists$ bind more strongly than anything else.
- example: $\forall x\, \mathsf{P}(x) \rightarrow \mathsf{Q}(x)$ corresponds to $(\forall x\, \mathsf{P}(x) \rightarrow \mathsf{Q}(x))$,
  not $\forall x\, (\mathsf{P}(x) \rightarrow \mathsf{Q}(x))$.

Exercise

$\mathcal{S} = \langle \{x, y, z\}, \{\mathsf{c}\}, \{\mathsf{f}, \mathsf{g}, \mathsf{h}\}, \{\mathsf{Q}, \mathsf{R}, \mathsf{S}\} \rangle$ with
$ar(\mathsf{f}) = 3, ar(\mathsf{g}) = ar(\mathsf{h}) = 1, ar(\mathsf{Q}) = 2, ar(\mathsf{R}) = ar(\mathsf{S}) = 1$

- $\mathsf{f}(x, y)$
- $(\mathsf{g}(x) = \mathsf{R}(y))$
- $(\mathsf{g}(x) = \mathsf{f}(y, \mathsf{c}, \mathsf{h}(x)))$
- $(\mathsf{R}(x) \land \forall x\, \mathsf{S}(x))$
- $\forall \mathsf{c}\, \mathsf{Q}(\mathsf{c}, x)$
- $(\forall x \exists y\, (\mathsf{g}(x) = y) \lor (\mathsf{h}(x) = \mathsf{c}))$

Which expressions are syntactically correct formulas or terms for $\mathcal{S}$?
What kind of term/formula?

D5.2 Semantics of Predicate Logic

Semantics: Motivation

- interpretations in propositional logic:
  truth assignments for the propositional variables
- There are no propositional variables in predicate logic.
- instead: interpretation determines meaning
  of the constant, function and predicate symbols.
- meaning of variable symbols not determined by interpretation
  but by separate variable assignment

Interpretations and Variable Assignments

Let $\mathcal{S} = \langle \mathcal{V}, \mathcal{C}, \mathcal{F}, \mathcal{P} \rangle$ be a signature.

Definition (Interpretation, Variable Assignment)

An interpretation (for $\mathcal{S}$) is a pair $\mathcal{I} = \langle U, \cdot^{\mathcal{I}} \rangle$ of:

- a non-empty set U called the universe and
- a function $\cdot^{\mathcal{I}}$ that assigns a meaning to the constant, function, and predicate symbols:
  - $\mathsf{c}^{\mathcal{I}} \in U$ for constant symbols $\mathsf{c} \in \mathcal{C}$
  - $\mathsf{f}^{\mathcal{I}} : U^k \to U$ for k-ary function symbols $\mathsf{f} \in \mathcal{F}$
  - $\mathsf{P}^{\mathcal{I}} \subseteq U^k$ for k-ary predicate symbols $\mathsf{P} \in \mathcal{P}$

A variable assignment (for $\mathcal{S}$ and universe U) is a function $\alpha : \mathcal{V} \to U$.

German: Interpretation, Universum (or Grundmenge), Variablenzuweisung

Interpretations and Variable Assignments: Example

Example

signature: $\mathcal{S} = \langle \mathcal{V}, \mathcal{C}, \mathcal{F}, \mathcal{P} \rangle$ with $\mathcal{V} = \{x, y, z\}$,
$\mathcal{C} = \{\text{zero}, \text{one}\}$, $\mathcal{F} = \{\text{sum}, \text{product}\}$, $\mathcal{P} = \{\text{SquareNumber}\}$
$ar(\text{sum}) = ar(\text{product}) = 2$, $ar(\text{SquareNumber}) = 1$

$\mathcal{I} = \langle U, \cdot^{\mathcal{I}} \rangle$ with

- $U = \{u_0, u_1, u_2, u_3, u_4, u_5, u_6\}$
- $\text{zero}^{\mathcal{I}} = u_0$
- $\text{one}^{\mathcal{I}} = u_1$
- $\text{sum}^{\mathcal{I}}(u_i, u_j) = u_{(i+j) \bmod 7}$ for all $i, j \in \{0, \dots, 6\}$
- $\text{product}^{\mathcal{I}}(u_i, u_j) = u_{(i \cdot j) \bmod 7}$ for all $i, j \in \{0, \dots, 6\}$
- $\text{SquareNumber}^{\mathcal{I}} = \{u_0, u_1, u_2, u_4\}$

$\alpha = \{x \mapsto u_5, y \mapsto u_5, z \mapsto u_0\}$

Semantics: Informally

Example: $(\forall x(\mathrm{Block}(x) \rightarrow \mathrm{Red}(x)) \land \mathrm{Block}(\mathrm{a}))$
"For all objects x: if x is a block, then x is red.
Also, the object called a is a block."

- Terms are interpreted as objects.
- Unary predicates denote properties of objects (to be a block, to be red, to be a square number, ...).
- General predicates denote relations between objects (to be someone's child, to have a common divisor, ...).
- Universally quantified formulas ("$\forall$") are true if they hold for every object in the universe.
- Existentially quantified formulas ("$\exists$") are true if they hold for at least one object in the universe.

Interpretations of Terms

Let $\mathcal{S} = \langle \mathcal{V}, \mathcal{C}, \mathcal{F}, \mathcal{P} \rangle$ be a signature.

Definition (Interpretation of a Term)

Let $\mathcal{I} = \langle U, \cdot^{\mathcal{I}} \rangle$ be an interpretation for $\mathcal{S}$,
and let α be a variable assignment for $\mathcal{S}$ and universe U.

Let t be a term over $\mathcal{S}$.
The interpretation of t under $\mathcal{I}$ and α, written as $t^{\mathcal{I},\alpha}$,
is the element of the universe U defined as follows:

- If $t = x$ with $x \in \mathcal{V}$ (t is a variable term):
  $x^{\mathcal{I},\alpha} = \alpha(x)$
- If $t = \mathsf{c}$ with $\mathsf{c} \in \mathcal{C}$ (t is a constant term):
  $\mathsf{c}^{\mathcal{I},\alpha} = \mathsf{c}^{\mathcal{I}}$
- If $t = \mathsf{f}(t_1, \dots, t_k)$ (t is a function term):
  $\mathsf{f}(t_1, \dots, t_k)^{\mathcal{I},\alpha} = \mathsf{f}^{\mathcal{I}}(t_1^{\mathcal{I},\alpha}, \dots, t_k^{\mathcal{I},\alpha})$

Interpretations of Terms: Example

Example

signature: $\mathcal{S} = \langle \mathcal{V}, \mathcal{C}, \mathcal{F}, \mathcal{P} \rangle$
with $\mathcal{V} = \{x, y, z\}$, $\mathcal{C} = \{\mathsf{zero}, \mathsf{one}\}$, $\mathcal{F} = \{\mathsf{sum}, \mathsf{product}\}$,
$ar(\mathsf{sum}) = ar(\mathsf{product}) = 2$

$\mathcal{I} = \langle U, \cdot^{\mathcal{I}} \rangle$ with

- $U = \{u_0, u_1, u_2, u_3, u_4, u_5, u_6\}$
- $\mathsf{zero}^{\mathcal{I}} = u_0$
- $\mathsf{one}^{\mathcal{I}} = u_1$
- $\mathsf{sum}^{\mathcal{I}}(u_i, u_j) = u_{(i+j) \bmod 7}$ for all $i, j \in \{0, \dots, 6\}$
- $\mathsf{product}^{\mathcal{I}}(u_i, u_j) = u_{(i \cdot j) \bmod 7}$ for all $i, j \in \{0, \dots, 6\}$

$\alpha = \{x \mapsto u_5, y \mapsto u_5, z \mapsto u_0\}$

Interpretations of Terms: Example (ctd.)

Example (ctd.)

- $\mathsf{zero}^{\mathcal{I},\alpha} =$
- $y^{\mathcal{I},\alpha} =$
- $\mathsf{sum}(x, y)^{\mathcal{I},\alpha} =$
- $\mathsf{product}(\mathsf{one}, \mathsf{sum}(x, \mathsf{zero}))^{\mathcal{I},\alpha} =$

Semantics of Predicate Logic Formulas

Let $\mathcal{S} = \langle \mathcal{V}, \mathcal{C}, \mathcal{F}, \mathcal{P} \rangle$ be a signature.

Definition (Formula is Satisfied or True)

Let $\mathcal{I} = \langle U, \cdot^{\mathcal{I}} \rangle$ be an interpretation for $\mathcal{S}$,
and let α be a variable assignment for $\mathcal{S}$ and universe U.
We say that $\mathcal{I}$ and α satisfy a predicate logic formula φ
(also: φ is true under $\mathcal{I}$ and α), written: $\mathcal{I}, \alpha \models \varphi$,
according to the following inductive rules:

$$\begin{aligned}
\mathcal{I}, \alpha &\models \mathsf{P}(t_1, \dots, t_k) && \text{iff } \langle t_1^{\mathcal{I},\alpha}, \dots, t_k^{\mathcal{I},\alpha} \rangle \in \mathsf{P}^{\mathcal{I}} \\
\mathcal{I}, \alpha &\models (t_1 = t_2) && \text{iff } t_1^{\mathcal{I},\alpha} = t_2^{\mathcal{I},\alpha} \\
\mathcal{I}, \alpha &\models \neg\varphi && \text{iff } \mathcal{I}, \alpha \not\models \varphi \\
\mathcal{I}, \alpha &\models (\varphi \land \psi) && \text{iff } \mathcal{I}, \alpha \models \varphi \text{ and } \mathcal{I}, \alpha \models \psi \\
\mathcal{I}, \alpha &\models (\varphi \lor \psi) && \text{iff } \mathcal{I}, \alpha \models \varphi \text{ or } \mathcal{I}, \alpha \models \psi \qquad \dots
\end{aligned}$$

German: $\mathcal{I}$ und α erfüllen φ (also: φ ist wahr unter $\mathcal{I}$ und α)

Semantics of Predicate Logic Formulas

Let $\mathcal{S} = \langle \mathcal{V}, \mathcal{C}, \mathcal{F}, \mathcal{P} \rangle$ be a signature.

Definition (Formula is Satisfied or True)

...

$$\begin{array}{ll} \mathcal{I}, \alpha \models \forall x \varphi & \text{iff } \mathcal{I}, \alpha[x := u] \models \varphi \text{ for all } u \in U \\ \mathcal{I}, \alpha \models \exists x \varphi & \text{iff } \mathcal{I}, \alpha[x := u] \models \varphi \text{ for at least one } u \in U \end{array}$$

where $\alpha[x := u]$ is the same variable assignment as α, except that it maps variable x to the value u.
Formally:

$$(\alpha[x := u])(z) = \begin{cases} u & \text{if } z = x \\ \alpha(z) & \text{if } z \neq x \end{cases}$$

Semantics: Example

Example

signature: $\mathcal{S} = \langle \mathcal{V}, \mathcal{C}, \mathcal{F}, \mathcal{P} \rangle$
with $\mathcal{V} = \{x, y, z\}$, $\mathcal{C} = \{\mathsf{a}, \mathsf{b}\}$, $\mathcal{F} = \emptyset$, $\mathcal{P} = \{\mathsf{Block}, \mathsf{Red}\}$,
$ar(\mathsf{Block}) = ar(\mathsf{Red}) = 1$.

$\mathcal{I} = \langle U, \cdot^{\mathcal{I}} \rangle$ with

- $U = \{u_1, u_2, u_3, u_4, u_5\}$
- $\mathsf{a}^{\mathcal{I}} = u_1$
- $\mathsf{b}^{\mathcal{I}} = u_3$
- $\mathsf{Block}^{\mathcal{I}} = \{u_1, u_2\}$
- $\mathsf{Red}^{\mathcal{I}} = \{u_1, u_2, u_3, u_5\}$

$\alpha = \{x \mapsto u_1, y \mapsto u_2, z \mapsto u_1\}$

Semantics: Example (ctd.)

Example (ctd.)

Questions:

- $\mathcal{I}, \alpha \models (\mathsf{Block}(\mathsf{b}) \lor \neg\mathsf{Block}(\mathsf{b}))$?
- $\mathcal{I}, \alpha \models (\mathsf{Block}(x) \rightarrow (\mathsf{Block}(x) \lor \neg\mathsf{Block}(y)))$?
- $\mathcal{I}, \alpha \models (\mathsf{Block}(\mathsf{a}) \land \mathsf{Block}(\mathsf{b}))$?
- $\mathcal{I}, \alpha \models \forall x (\mathsf{Block}(x) \rightarrow \mathsf{Red}(x))$?

Summary

- Predicate logic is more expressive than propositional logic and allows statements over objects and their properties.
- Objects are described by terms that are built from variable, constant and function symbols.
- Properties and relations are described by formulas that are built from predicates, quantifiers and the usual logical operators.

Discrete Mathematics in Computer Science

D6. Advanced Concepts in Predicate Logic and Outlook

Malte Helmert, Gabriele Röger

University of Basel

December 15, 2025

Discrete Mathematics in Computer Science

December 15, 2025 — D6. Advanced Concepts in Predicate Logic and Outlook

D6.1 Free and Bound Variables

Free and Bound Variables: Motivation

Question:

- Consider a signature with variable symbols $\{x_1, x_2, x_3, \dots\}$ and an interpretation $\mathcal{I}$.
- Which parts of the definition of α are relevant to decide whether $\mathcal{I}, \alpha \models (\forall x_4 (\mathsf{R}(x_4, x_2) \lor (\mathsf{f}(x_3) = x_4)) \lor \exists x_3 \mathsf{S}(x_3, x_2))$?
- $\alpha(x_1)$, $\alpha(x_5)$, $\alpha(x_6)$, $\alpha(x_7)$, $\dots$ are irrelevant since those variable symbols occur in no formula.
- $\alpha(x_4)$ also is irrelevant: the variable occurs in the formula, but all occurrences are bound by a surrounding quantifier.
- $\rightsquigarrow$ only assignments for free variables x_2 and x_3 relevant

German: gebundene und freie Variablen

Variables of a Term

Definition (Variables of a Term)

Let t be a term. The set of variables that occur in t, written as $\mathit{var}(t)$, is defined as follows:

- $\mathit{var}(x) = \{x\}$
  for variable symbols x
- $\mathit{var}(\mathrm{c}) = \emptyset$
  for constant symbols c
- $\mathit{var}(\mathrm{f}(t_1, \dots, t_k)) = \mathit{var}(t_1) \cup \dots \cup \mathit{var}(t_k)$
  for function terms

terminology: A term t with $\mathit{var}(t) = \emptyset$ is called ground term.
German: Grundterm

example: $\mathit{var}(\mathrm{product}(x, \mathrm{sum}(\mathrm{k}, y))) =$

Free and Bound Variables of a Formula

Definition (Free Variables)

Let φ be a predicate logic formula. The set of free variables of φ, written as $\mathit{free}(\varphi)$, is defined as follows:

- $\mathit{free}(\mathsf{P}(t_1, \dots, t_k)) = \mathit{var}(t_1) \cup \dots \cup \mathit{var}(t_k)$
- $\mathit{free}((t_1 = t_2)) = \mathit{var}(t_1) \cup \mathit{var}(t_2)$
- $\mathit{free}(\neg\varphi) = \mathit{free}(\varphi)$
- $\mathit{free}((\varphi \land \psi)) = \mathit{free}((\varphi \lor \psi)) = \mathit{free}(\varphi) \cup \mathit{free}(\psi)$
- $\mathit{free}(\forall x\, \varphi) = \mathit{free}(\exists x\, \varphi) = \mathit{free}(\varphi) \setminus \{x\}$

Example: $\mathit{free}((\forall x_4(\mathsf{R}(x_4, x_2) \lor (\mathsf{f}(x_3) = x_4)) \lor \exists x_3 \mathsf{S}(x_3, x_2)))$
$=$

Closed Formulas/Sentences

Note: Let φ be a formula and let α and β variable assignments with $\alpha(x) = \beta(x)$ for all free variables x of φ.
Then $\mathcal{I}, \alpha \models \varphi$ iff $\mathcal{I}, \beta \models \varphi$.

In particular, α is completely irrelevant if $\mathit{free}(\varphi) = \emptyset$.

Definition (Closed Formulas/Sentences)

A formula φ without free variables (i. e., $\mathit{free}(\varphi) = \emptyset$) is called closed formula or sentence.

If φ is a sentence, then we often write $\mathcal{I} \models \varphi$ instead of $\mathcal{I}, \alpha \models \varphi$, since the definition of α does not influence whether φ is true under $\mathcal{I}$ and α or not.

Formulas with at least one free variable are called open.

Closed formulas with no quantifiers are called ground formulas.

German: geschlossene Formel/Satz, offene Formel,
Grundformel/variablenfreie Formel

Closed Formulas/Sentences: Examples

Question: Which of the following formulas are sentences?

- $(\mathsf{Block}(\mathsf{b}) \vee \neg \mathsf{Block}(\mathsf{b}))$
- $(\mathsf{Block}(x) \rightarrow (\mathsf{Block}(x) \vee \neg \mathsf{Block}(y)))$
- $(\mathsf{Block}(\mathsf{a}) \wedge \mathsf{Block}(\mathsf{b}))$
- $\forall x(\mathsf{Block}(x) \rightarrow \mathsf{Red}(x))$

D6.2 Reasoning in Predicate Logic

Terminology for Formulas

The terminology we introduced for propositional logic equally applies to predicate logic:

- Interpretation $\mathcal{I}$ and variable assignment α form a model of the formula φ if $\mathcal{I}, \alpha \models \varphi$.
- Formula φ is satisfiable if $\mathcal{I}, \alpha \models \varphi$ for at least one $\mathcal{I}, \alpha$.
- Formula φ is falsifiable if $\mathcal{I}, \alpha \not\models \varphi$. for at least one $\mathcal{I}, \alpha$
- Formula φ is valid if $\mathcal{I}, \alpha \models \varphi$ for all $\mathcal{I}, \alpha$.
- Formula φ is unsatisfiable if $\mathcal{I}, \alpha \not\models \varphi$ for all $\mathcal{I}, \alpha$.

German: Modell, erfüllbar, falsifizierbar, gültig, unerfüllbar

All concepts can be used for the special case of sentences.
In this case we usually omit α. Examples:

- Interpretation $\mathcal{I}$ is a model of a sentence φ if $\mathcal{I} \models \varphi$.
- Sentence φ is unsatisfiable if $\mathcal{I} \not\models \varphi$ for all $\mathcal{I}$.

Sets of Formulas: Semantics

Definition (Satisfied/True Sets of Formulas)

Let $\mathcal{S}$ be a signature, Φ a set of formulas over $\mathcal{S}$, $\mathcal{I}$ an interpretation for $\mathcal{S}$ and α a variable assignment for $\mathcal{S}$ and the universe of $\mathcal{I}$.

We say that $\mathcal{I}$ and α satisfy the formulas Φ (also: Φ is true under $\mathcal{I}$ and α), written as: $\mathcal{I}, \alpha \models \Phi$, if $\mathcal{I}, \alpha \models \varphi$ for all $\varphi \in \Phi$.

German: $\mathcal{I}$ und α erfüllen Φ, Φ ist wahr unter $\mathcal{I}$ und α

We may again write $\mathcal{I} \models \Phi$ if all formulas in Φ are sentences.

Logical Equivalence and Logical Consequences

We again we use the same concepts and notations
as in propositional logic.

- A set of formulas Φ logically entails/implies formula ψ, written as $\Phi \models \psi$, if all models of Φ are models of ψ.
- For a single formula φ, we may write $\varphi \models \psi$ for $\{\varphi\} \models \psi$.
- Formulas φ and ψ are logically equivalent, written as $\varphi \equiv \psi$, if they have the same models.
  - Note that $\varphi \equiv \psi$ iff $\varphi \models \psi$ and $\psi \models \varphi$.

Important Theorems about Logical Consequences

Theorem (Deduction Theorem)

$\mathrm{KB} \cup \{\varphi\} \models \psi$ *iff* $\mathrm{KB} \models (\varphi \rightarrow \psi)$

German: Deduktionssatz

Theorem (Contraposition Theorem)

$\mathrm{KB} \cup \{\varphi\} \models \neg\psi$ *iff* $\mathrm{KB} \cup \{\psi\} \models \neg\varphi$

German: Kontrapositionssatz

Theorem (Contradiction Theorem)

$\mathrm{KB} \cup \{\varphi\}$ *is unsatisfiable iff* $\mathrm{KB} \models \neg\varphi$

German: Widerlegungssatz

These can be proved exactly the same way as in propositional logic.

Logical Equivalences

- All logical equivalences of propositional logic also hold in predicate logic (e. g., $(\varphi \lor \psi) \equiv (\psi \lor \varphi)$). (Why?)
- Additionally the following equivalences and implications hold:

$$\begin{aligned}
(\forall x\varphi \land \forall x\psi) &\equiv \forall x(\varphi \land \psi) && \\
(\forall x\varphi \lor \forall x\psi) &\models \forall x(\varphi \lor \psi) && \text{but not the converse} \\
(\forall x\varphi \land \psi) &\equiv \forall x(\varphi \land \psi) && \text{if } x \notin \mathit{free}(\psi) \\
(\forall x\varphi \lor \psi) &\equiv \forall x(\varphi \lor \psi) && \text{if } x \notin \mathit{free}(\psi) \\
\neg\forall x\varphi &\equiv \exists x\neg\varphi && \\
\exists x(\varphi \lor \psi) &\equiv (\exists x\varphi \lor \exists x\psi) && \\
\exists x(\varphi \land \psi) &\models (\exists x\varphi \land \exists x\psi) && \text{but not the converse} \\
(\exists x\varphi \lor \psi) &\equiv \exists x(\varphi \lor \psi) && \text{if } x \notin \mathit{free}(\psi) \\
(\exists x\varphi \land \psi) &\equiv \exists x(\varphi \land \psi) && \text{if } x \notin \mathit{free}(\psi) \\
\neg\exists x\varphi &\equiv \forall x\neg\varphi &&
\end{aligned}$$

Normal Forms (1)

Analogously to DNF and CNF for propositional logic
there are several normal forms for predicate logic, such as

- negation normal form (NNF):
  negation symbols ($\neg$) are only allowed in front of atoms or identities
- prenex normal form:
  quantifiers must form the outermost part of the formula
- Skolem normal form:
  prenex normal form without existential quantifiers

German: Negationsnormalform, Pränexnormalform, Skolemnormalform

Normal Forms (2)

Efficient methods transform formula φ

- into an equivalent formula in negation normal form,
- into an equivalent formula in prenex normal form, or
- into an equisatisfiable formula in Skolem normal form.

German: erfüllbarkeitsäquivalent

Inference Rules and Calculi

There exist correct and complete proof systems (calculi) for predicate logic.

- An example is the natural deduction calculus.
- This is (essentially) Gödel's Completeness Theorem (1929).
- However, one can show that correct and complete algorithms that prove that a given formula does not follow from a given set of formulas cannot exist.
- How are these statements reconcilable?

D6.3 Summary and Outlook

Summary

- Predicate logic is more expressive than propositional logic and allows statements over objects and their properties.
- Objects are described by terms that are built from variable, constant and function symbols.
- Properties and relations are described by formulas that are built from predicates, quantifiers and the usual logical operators.
- Bound vs. free variables: to decide if $\mathcal{I}, \alpha \models \varphi$, only free variables in α matter
- Sentences (closed formulas): formulas without free variables

Summary

Once the basic definitions are in place, predicate logic can be developed in the same way as propositional logic:

- logical consequence
- deduction theorem etc.
- logical equivalences
- normal forms
- inference rules, proof systems, resolution

Other Logics (1)

- We considered first-order predicate logic.
- Second-order predicate logic allows quantifying over predicate symbols.
- There are intermediate steps, e. g., monadic second-order logic (all quantified predicates are unary) and description logics (foundation of the semantic web).

Second-Order Logic Example

Second-order logic example:

- "T is the transitive closure of R"
- conjunction of
  - $\forall x \forall y (R(x, y) \rightarrow T(x, y))$
    "T is a superset of R"
  - $\forall x \forall y \forall z ((T(x, y) \land T(y, z)) \rightarrow T(x, z))$
    "T is transitive"
  - $$\begin{aligned}\forall Q((&\forall x \forall y (R(x, y) \rightarrow Q(x, y)) \land \\ &\forall x \forall y \forall z ((Q(x, y) \land Q(y, z)) \rightarrow Q(x, z))) \\ &\rightarrow \forall x \forall y (T(x, y) \rightarrow Q(x, y))))\end{aligned}$$
    "All supersets Q of R that are transitive are supersets of T"
- impossible to express in first-order logic

Other Logics (2)

- Modal logics have new operators $\Box$ and $\Diamond$.
  - classical meaning: $\Box\varphi$ for "φ is necessary",
    $\Diamond\varphi$ for "φ is possible".
  - temporal logic: $\Box\varphi$ for "φ is always true in the future",
    $\Diamond\varphi$ for "φ is true at some point in the future"
  - epistemic logic: $\Box\varphi$ for "φ is known",
    $\Diamond\varphi$ for "φ is possible"
  - doxastic logic: $\Box\varphi$ for "φ is believed",
    $\Diamond\varphi$ for "φ is considered possible"
  - deontic logic: $\Box\varphi$ for "φ is obligatory",
    $\Diamond\varphi$ for "φ is permitted"
  - ...
- very important in computer-aided verification

Other Logics (3)

- In fuzzy logic, formulas are not true or false but have values between 0 and 1.
- Intuitionist logic is "constructive" and excludes indirect proof methods such as the principle of the excluded third.
- Non-monotonic logics have rules with exceptions (e.g., default logic, cumulative logic).
- . . . and there is a lot more