Discrete Structures

Prof. Stefan Wolf
Arne Hansen

October 5, 2015

Contents

What is discrete mathematics all about?

Before venturing into formal details, let's briefly clearify what we refer to if we speak about *discrete mathematics*.

What is *mathematics*?

The pillars of mathematics are *logics* and *set theory*. While logics determines how to reason within mathematics (what is considered a proof, ...), set theory describes the objects we deal with.

Example (Set theory). It is actually possible to define all natural numbers using merely the emptyset:

$$\begin{aligned} 0 &:= \emptyset \\ 1 &:= \{\emptyset\} \\ 2 &:= \{\{\emptyset\}, \emptyset\} \\ &\vdots \end{aligned}$$

This exemplifies how set theory serves to define basic mathematical entities.

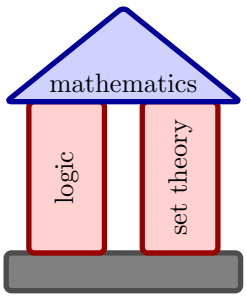


Figure 1: The pillars of mathematics.

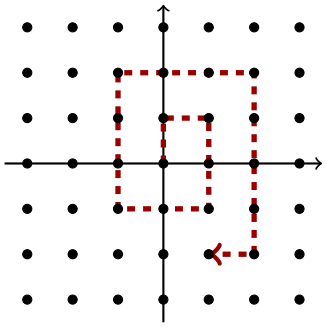


Figure 2: Matching the rational and the natural numbers

What does *discrete* mean?

With *discrete* we usually refer to *finite or countably infinite sets*. While it is intuitively clear what is meant by *finite* (a set of elements we can label with natural numbers $1, 2, 3, \ldots, N$) *countably infinite* needs slightly more explanation. A set S is called *countably infinite* if one can match each element in S with a unique natural number. More formally such a matching would be a function and we get the following definition.

Definition 1 (Countable set)**.** A set S is called *countable* if there exists a one-to-one or injective[1] function $f : S \to \mathbb{N}$. If f is also surjective[2] then S is called countably infinite.

To better understand the concept of countability, we'll consider two common examples.

Example (The rational numbers are countable). Even though there seem to be a lot more rational numbers than natural numbers, we can construct a matching showing that the set is countably infinite. The rational numbers can be associated with dots in a two-dimensional space, with the numerator on the x-axis and the denominator on the y-axis or vice versa. We can then number the dots in a spiral as shown in figure 2.

Example (The real numbers are not). We cannot find such a matching for the real numbers as one can prove by contradiction. Let's assume that there is a matching for all real numbers in the interval $[0, 1]$ with a binary representation

$$\begin{array}{ll} 1 & 0.0101011100110\ldots \\ 2 & 0.1001100101001\ldots \\ 3 & 0.0001011001010\ldots \\ \vdots & \vdots \end{array}$$

But one can construct a real number that is not contained in the list: the i^{th} digit of the binary representation is just the flipped value of the i^{th} digit of the i^{th} real number.

[1] This is to say, each element gets assigned a unique natural number, or in other words, no natural number is matched to two different elements in S

[2] That is, every natural number is matched to an element in S

1	$0.\mathbf{0}1010111001100\ldots$
2	$0.1\mathbf{0}0110010100 1\ldots$
3	$0.00\mathbf{0}1011001010\ldots$
$\vdots$	$\vdots$
constructed real number	$0.111\ldots$

The number we constructed differs from every element in the list and is therefore not contained in the list. This yields a contradiction with the initial assumption that there existed a complete matching. Thus the size of the set of real numbers is strictly larger than the size of the natural number (in the sense that there is no bijective map inbetween the two.).

$$|\mathbb{R}| > |\mathbb{Q}| = |\mathbb{N}| \tag{1}$$

Why bother?

What is the relevance of discrete mathematics? On the one hand logic is discrete as the basic set is $\{\texttt{TRUE}, \texttt{FALSE}\}$. On the other hand, computer science is not only closely related to logics, but as computers have only finite states informatics is in a sense discrete.

Chapter 1

Motivation

After having clearified the meaning of 'discrete mathematics' we will look into interesting examples of the field.

1.1 Swapping knights

Consider a reduced chess board with two knights of each color as shown in figure 1.1. Is it possible to transform the configuration on the left into the configuration on the right using regular knight moves?

The knights never reach the middle field. Further if one moves along the line of moves one finally returns back to the initial position. This can be represented nicely in a graph. Each possible field is then a *node* in the graph. The possible moves are then the *edges* of the graph. Rearranging the nodes yields one closed loop with 8 nodes. The knights can merely move around the loop but not change their order. Thus the answer is, that the transformation is impossible.

Lessons learnt from the example above:

Modelling Find the right model to represent the problem.

Graph are often a good structure to model discrete problems

Abstraction Concentrate on the relevant and get rid of the irrelevant.

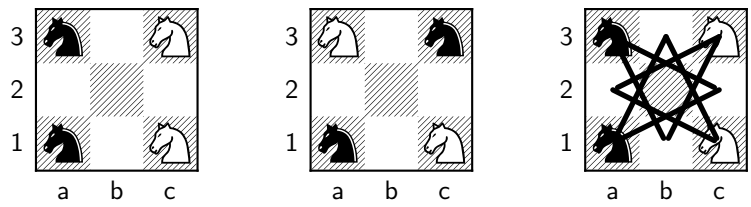


Figure 1.1: Reduced chessboard with knights

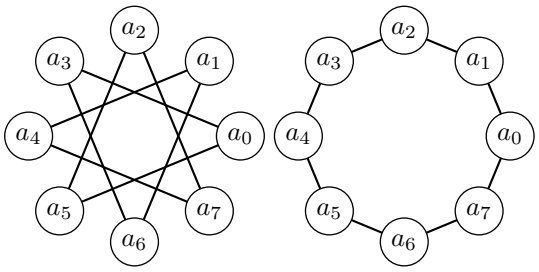


Figure 1.2: Graphs corresponding to the knights swapping problem

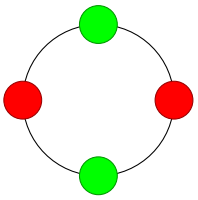


Figure 1.3: Necklace with further symmetries, but p not being prime.

1.2 Combinatorics

Imagine you want to create a necklace with p pearls, with p being prime. There are pearls of a different colors available. How many different necklaces can you make? If we can distinguish the pearls, there would be a^p different possible necklaces. But the necklace has a rotational symmetry. That is two necklaces that are made up from the same sequence of pearls except that they are rotated by an angle are indistinguishable. Just diving by p does not yield the right solution though, as the one-colored necklaces just appear once. As p is a prime number these are they only exceptions. Configuration like in 1.3 cannot occur. Therefore the right answer is

$$N = \frac{a^p - a}{p} + a \tag{1.1}$$

More generally we obtain for a prime number p, $a^p - a$ is divisable by p. This is called Fermat's small theorem and plays a role in cryptography, namely the public-key protocol RSA.

1.3 Connections without crossings

Imagine a settlement with three houses and three plants (power, waste water, ...) as shown in figure 1.4. The question is, whether it is possible to connect the three houses to each of the plants without crossing other connections.

This problem can again be turned into a graph, namely $K_{3,3}$ as shown in figure 1.5. The question is then: can the associated graph be drawn that edges merely meet in nodes? If so the graph is called *planar*. An example of a planar graph is K_4 shown in figure 1.6.

So we ultimately want to show, that $K_{3,3}$ is not planar. To do so we first try to better understand the properties of planar graphs.

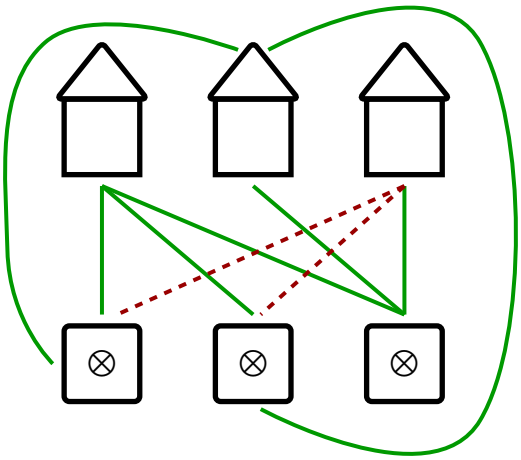


Figure 1.4: Houses with connections. The third house cannot be connected anymore without intersecting other connections.

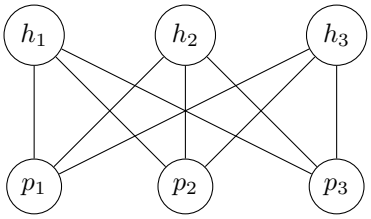


Figure 1.5: The "houses connection problem" can be associated to the graph $K_{3,3}$

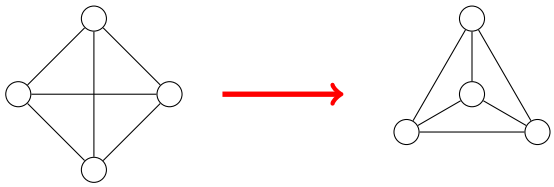


Figure 1.6: The graph K_4 is an example of a planar graph

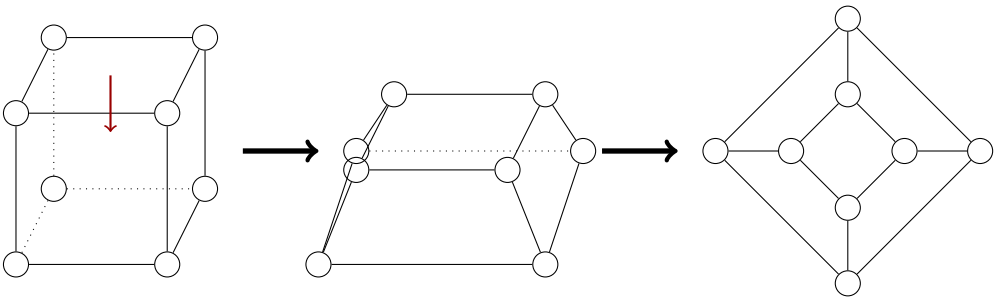


Figure 1.7: A cube can be transformed into a graph by removing the bottom (turns into outer region) and pressing the top down while spreading out the lower nodes.

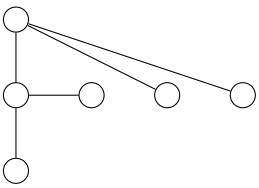


Figure 1.8: A simple tree

Graphs and polyhedrons There is an interesting analogy between graphs and polyhedrons. A polyhedron can be associated with a graph: Imagine the polyhedron was made out of elastic rubber. One could now remove one face and press the remaining part on a piece of paper. This would yield a planar graph. Edges of the Polyhedron become edges of the graph, corners turn into nodes of the graph. Note that the face that was initially removed corresponds now to the infinite region outside the graph (which as always considered as a proper region). Figure 1.7 shows the graph corresponding to a cube.

Looking at different polyhedra (or at planar graphs) it becomes apparant that the number of vertices (nodes) n, the number of edges e and the number of faces f (regions) are closely reelated. Indeed *Euler's polyhedron formula* states, for any polyhedron (and equivalently for planar graphs) it holds

$$n - e + f = 2 \tag{1.2}$$

Before looking at a sketch of the proof we can apply it to $K_{4,4}$ to answer our initial question. Indeed $n - e + f = 6 - 9 + 13 = 10 \neq 2$.

Proof sketch The proof consists of two parts: First we will show the proposition for trees, which form a simpler class of graphs. Then we will generalize this to arbitrary planar paths.

Euler for trees A tree is a path without any cycles. As it contains only one region, namely the outer one, it is always planar. We can now show by induction that Euler's law is always satisfied for trees.

If there is just one node, then the formula is satisfied as $n = 1$, $e = 0$ and $f = 1$. For the *induction step* we now assume that the law holds for a tree with

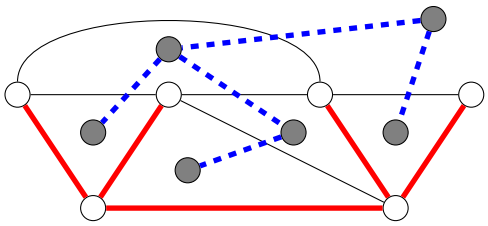


Figure 1.9:

n nodes and follow that it will hold for a tree with $n+1$ nodes. Adding node requires to add an edge as well. So the new tree will have $e+1$ edges and $n+1$ nodes and therefore

$$\underbrace{\tilde{n}}_{=n+1} - \underbrace{\tilde{e}}_{=e+1} + \underbrace{\tilde{f}}_{=1} = n - e + f = 2 \tag{1.3}$$

where the second equality holds due to the induction assumption.

Generalization A general planar graph can now be characterized by a spanning tree and the dual graph. A spanning tree is a subgraph, connecting all nodes, that is at the same time a tree. The dual graph to a given spanning tree is the constructed by placing a node in each region and connecting them without crossing the spanning tree.

The spanning tree is has the same number of nodes as the original graph $n_d = n$ and the number of edges just one less $e_s = n_s - 1$. The number of nodes in the dual tree are just the same as the number of regions in the graph, $n_d = f$. Further each edge of the graph is either part of the spanning tree or crossed by one of the edges of the dual tree. Putting all this together we obtain

$$e = e_s + e_d = (n_s - 1) + (n_d - 1) = n - 1 + f - 1 = n + f - 2 \tag{1.4}$$

which is just Euler's formula. The subsequent example will employ the formula as well.

1.4 Coloring maps

Imagine we were given a map showing countries and their borders[1]. Every two countries sharing a border are considered neighbors. So how many colors are needed to color the map in a way that no neighbors have the same color? Three colors are not enough as can be seen in figure 1.10.

Again the problem can be translated to planar graphs without multi-nodes[2]. Each country is replaced by a node and neighbouring countries are connected by edges. So can the nodes be colored with 4 colors requiring that neighbors are always colored differently? The answer is yes as a computer-aided proof showed. As the proof can never be verified directly by human being there has

[1] For simplicity we'll assume that there are no exclaves like Campione d'Italia.
[2] Multi-nodes are nodes with more than one edge connecting the two.

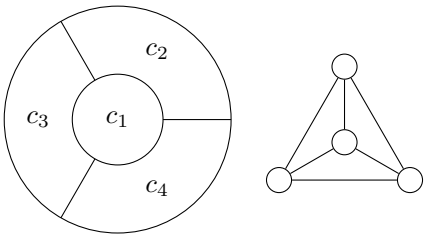


Figure 1.10: The figure on the left shows, that it is possible to arrange four countries such that they are all neighbors to one another. Thus we need at least four colors. The corresponding graph is shown on the right.

been some suspision towards the validity of the proof. A shorter proof shows that 5 colors are sufficient. In the following we sketch a proof for 6 colors.

Proposition 1.4.1 (6-Coloring). Six colors suffice to color a map (or equivalently the nodes of a planar, non-multi graph).

Proof. As there is a finite number of nodes (countries) we will prove the proposition by induction.

Base case For 6 or less nodes the proposition holds, as there are at least as many colors as nodes.

Induction hypothesis We will assume that the proposition holds for k nodes.

Induction step The major part of the proof is now to show that, given the induction hypothesis, the proposition hold also for $k+1$ nodes. The idea to do that is as follows

- Elimininate a node v
- By the induction hypothesis there exists a coloring for the reduced graph.
- One needs to find a coloring for the node v such that the proposition still holds. The goal choose the node v such that it has 5 or less neighbors and we can still choose one color.

While the first two points are clear, we still need to prove the last one — essentially the following lemma.

Lemma 1.4.1. *In every planar graph without multi-nodes there exists at least one node v with 5 or less neighbors.*

Lemma. We will prove the lemma by contradiction. We will assume that the lemma was actually false, i.e. that each (!) node has at least 6 neighbors. Based on this assumption we will construct a contradiction with Euler's formula.

As every node has at least 6 neighbors and each edge connects two nodes the edges and the nodes are related as follows

$$2e \geq 6n \tag{1.5}$$

Similarly the regions and the edges are related. Every region is bounded by at 3 edges (NB: two edges only suffice for multi-graphs). So we obtain

$$2e \geq 3f \tag{1.6}$$

Putting all this together we obtain

$$n + f \leq \frac{e}{3} + \frac{2}{3}e = e \quad \Rightarrow \quad n + f - e \leq 0 \tag{1.7}$$

This is in contradiction with Euler's formula and therefore completes the proof of the lemma. □

The proof of the lemma is the last missing piece. We now know that we can always find a node v with 5 or less neighbors to reduce a graph with $k+1$ nodes to a graph with k nodes. This completes the induction step and thus the proof of the proposition.

□

Chapter 2

Propositional Logic

In the introduction we mentioned logic as one of the pillars of mathematics. Now we will formaly introduce *propositional logic* (PL), a branch of logic concerned with propositions and how they can be constructed from basic proposition, so-called atomic propositions.

2.1 Definitions

Logic is ultimately about statements and the question whether they are true or false. This serves to formulate the following definition.

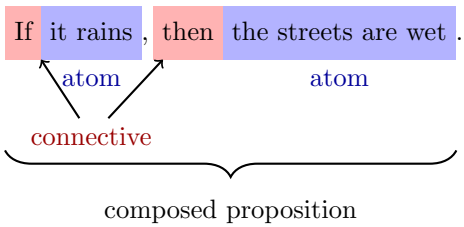
Definition 2 (Proposition)**.** A proposition is a sentence or an expression which is either true or false.

Note that we don't ask whether the statement is physically meaningful. The only criterion is, that a proposition is clearly either true or false. This property is called *truth-definiteness*.

Definition 3 (Atom)**.** An atom or atomic proposition is a basic proposition that is not composed from other propositions.

Definition 4 (Connective)**.** A connective relates to propositions logically.

Example 2.1.1. This examples shows how *atoms* (or atomic proposition) can be connected by connectives to form composed proposition.



(2.1)

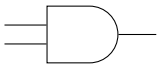


Figure 2.1: Logaical AND gate

The following sentence connects three atoms A, B, C.

A

When the rooster crows on the dungheap ,

then the weather changes or remains the same .

B C

The conditioning atom is without effect as the second proposition is always true. The atom C is just the negation of the atom B: $C = \neg B$. Therefore $C \vee B = (\neg B) \vee B =$ `true`. Such a proposition that is always true due to its form is called a *tautology*.

Propositions can be confusing when they refer to themselves. The following negation is false

This is not a proposition .

while the next one does not seem to be proper proposition at all as one cannot decide whether it is false or true.

This proposition is false .

2.1.1 Connectives as truth functions

A connective can be considered as a function of propositions, yielding yet another proposition. They can be characterized by so-called truth tables (see below). As propositions represent two values, just as bits in a computer, there is a close relation to gates (i.e. the abstract building blocks of a processor). Subsequently we will characterize some connectives.

Conjunction or AND gate is the connective that returns true if and only if both arguments are true.

A	B	$A \wedge B$
`true`	`true`	`true`
`true`	`false`	`false`
`false`	`true`	`false`
`false`	`false`	`false`

(2.2)

Negation or NOT gate returns the opposite of the input truth value.

A	$\neg A$
`true`	`false`
`false`	`true`

(2.3)

Together the AND and the NOT gate suffice to realize any other gate. We say these gates are *universal*. Interestingly just one gate — namely the NAND — is universal in itself.

NAND gate The two gates above can be combined to a new gate, the negated AND gate, short NAND gate

$$A \mid B := \neg(A \wedge B) \tag{2.4}$$

Consequently the truth table is just the inverted AND table

A	B	$A \mid B$
`true`	`true`	`false`
`true`	`false`	`true`
`false`	`true`	`true`
`false`	`false`	`true`

(2.5)

To get an idea of how the NAND gate can serve to simulate all other gates, one obtains the NOT gate by using A for both inputs: $\neg A$ ' = ' $A \mid A$. One can now use this NOT gate and the NAND to obtain an AND gate and so on and to forth.

Inclusive disjunction or OR gate returns true whenever at least one of the two inputs is true.

A	B	$A \vee B$
`true`	`true`	`true`
`true`	`false`	`true`
`false`	`true`	`true`
`false`	`false`	`false`

(2.6)

Note though that this is usually not what we refer to by "or". In common language "or" means "either ... or ...", corresponding to the following gate.

Exclusive disjunction or exclusive OR or XOR gate returns true if and only if one of the two arguments is true.

A	B	$A \oplus B$
`true`	`true`	`false`
`true`	`false`	`true`
`false`	`true`	`true`
`false`	`false`	`false`

A	B	$A \oplus B$
1	1	0
1	0	1
0	1	1
0	0	0

(2.7)

The XOR symbol resembles a plus for a reason: if we replace `false` by zero and `true` by 1 then the XOR is just addition modulo 2 (as we will see later in the course). This replacement by 0 and 1 simplies matters. We will therefore use from now on digits.

Logical equivalence is the inverse gate of the XOR, returning true if and only if the arguments have the same truth value

A	B	$A \leftrightarrow B$
0	0	1
0	1	0
1	0	0
1	1	1

(2.8)

So we get the that $A \leftrightarrow B$ ' = ' $\neg(A \oplus B)$ even though it is not yet clear what we mean by equality here. But we will return to the issue soon.

Implication connects two atoms A and B in the way, that if A is true also B is true. A is called the premise, B the conclusion.

A	B	$A \rightarrow B$
0	0	1
0	1	1
1	0	0
1	1	1

(2.9)

Conversely if B is true A is not necessarily true: $A \rightarrow B$ ' $\neq$ ' $B \rightarrow A$. Thus the connective is not symmetric. Also the 'negative' version is not equivalent: $A \rightarrow B$ ' $\neq$ ' $(\neg A) \rightarrow (\neg B)$. But the contraposition is: $A \rightarrow B$ ' = ' $(\neg B) \rightarrow (\neg A)$.

This asymmetry requires that one is particularly careful how to prove a statement. The line of reasoing cannot simply be inverted.

Nonetheless if both implications hold, then we obtain equivalence: $(A \rightarrow B) \wedge (B \rightarrow A)$ ' = ' $A \leftrightarrow B$. It is therefore possible to prove statements of equivalence by separately showing both implications.

There is still the issue of the '='. The expressions on the right and on the left were not the same, so on the level of strings there is no equality. It is necessary to distingiush the syntax (the sequence of symbols) and the semantic, the meaning, of a statement. The '=' indicated that the meaning of the two expressions was the same. We say they are *equivalent* and write $\equiv$. In the following we look at syntax and seminantics of propositional logic in greater detail.

2.2 Syntax of propositional logic

The following definition specifies what entities we can use in propositional logic. That is what are syntactically correct expressions, called *formulas*. The definition states how to construct formulas from *atomic formulas*.

Definition 5 (Syntax)**.** Syntactically correct formulas $\mathcal{E}_\mathcal{D}$ with a set of atoms $\mathcal{D} := \{A, B, C, \ldots\}$ are

- atmic forumlas in $\mathcal{D}$
- if f and g are syntactically correct formulas then also $(\neg f)$, $(f \wedge g)$ and $(f \vee g)$ are syntactically correct

These are all syntactically correct formulas.

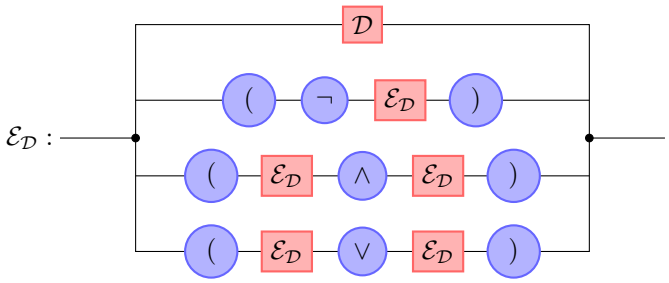


Figure 2.2: The syntax diagram for propositional logic

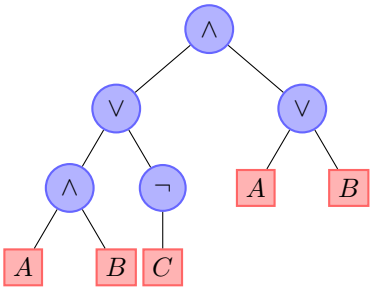


Figure 2.3: Tree of the formula F

In short, the only permitted connectives to related either atoms or other formulas are $\neg, \wedge$ and $\vee$. Remark that we listed more connectives above, that are a priori not part of the syntax. As we will see later, one can construct those from the ones in the definition.

We can describe this in a syntax diagram as shown in figure 2.2.

Example 2.2.1. The following are syntactically correct formulas. Note that the brackets cannot be left away so far.

- A
- $(\neg A)$
- $(A \wedge B)$
- $(A \vee B$
- $(\neg(A \wedge B))$
- $F := (((A \wedge B) \vee (\neg C)) \wedge (A \vee B))$

The last formula, F, can be visualized in a graph (figure 2.3). Subtrees correspond to partial formulas. The partial formulas of F are

$$\{F, A, B, C, (A \wedge B), (\neg C), ((A \wedge B) \vee (\neg C)), (A \vee B)\}$$

The following formulas are not correct as the brackets are missing

- $A \wedge B \wedge C$

- $A \wedge B \vee C$

The first can either be understood as $((A \wedge B) \wedge C)$ or $(A \wedge (B \wedge C))$. The meaning (i.e. the semantics) is the same. This does not hold for the second which can be interpreted as either $(A \wedge (B \vee C))$ or $((A \wedge B) \vee C)$.

2.3 Symantics of propositional logic

Definition 6 (Assignment)**.** A truth assignment $\mathcal{A} : \mathcal{D} \to \{0, 1\}$ is a function that assigns a truth value to all atoms. The function can naturally be extended to all formulas by simply evaluating the formula, given the truth values for its atoms.

Example 2.3.1 (Assignment). Let's consider a set of three atoms A, B, C. An assignment is for instance[1]

$$\begin{aligned} \mathcal{A} : A &\mapsto 0 \\ B &\mapsto 1 \\ C &\mapsto 0 \end{aligned}$$

The function can be extended to formulas. For $F = ((A \wedge (\neg B)) \vee C)$ the assignment yields $\mathcal{A}(F) = 0$.

Definition 7 (Semantic)**.** The semantic of a proposition is the set of truth values for all possible assignments.

In the examples below each row corresponds to one of the possible assignments and the semantic of the propositions is given by the vector highlighted in red.

Example 2.3.2. For the formula F in the example above one could evaluate the tree bottom-up. Ultimately we require a full list of the results for all possible inputs. It is more efficient to evaluate the subformulas and write the result below the connective as done below

$(((A$	$\wedge$	$B)$	$\vee$	$(\neg$	$C))$	$\wedge$	$(A$	$\vee$	$B))$
0	0	0	1	1	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0
0	0	1	1	1	0	1	0	1	1
0	0	1	0	0	1	0	0	1	1
1	0	0	1	1	0	1	1	1	0
1	0	0	0	0	1	0	1	1	0
1	1	1	1	1	0	1	1	1	1
1	1	1	1	0	1	1	1	1	1

(2.10)

Consequently we can define two formulas to be *equivalent* if they yield the same output on the same input.

Definition 8 (Semantically equivalent)**.** Two formulas are semantically equivalent if they have the same truth value for all assignments of their atomic formulas. We write then $F \equiv G$ or $F \Leftrightarrow G$.

[1] The symbol $\mapsto$ stands for "maps to". So for a function f the expression $f : x \mapsto y$ means "f maps x to y" and is just another way of writing $f(x) = y$.

Example 2.3.3 (Equivalent formulas). The following two formulas are equivalent, they yield the same red columns.

$$\begin{array}{ccccc} (A & \vee & B) \\ 0 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{array} \qquad \begin{array}{ccccccc} (& \neg & ((\neg & A) & \wedge & (\neg & B))) \\ & 0 & 1 & 0 & 1 & 1 & 0 \\ & 1 & 1 & 0 & 0 & 0 & 1 \\ & 1 & 0 & 1 & 0 & 1 & 0 \\ & 1 & 0 & 1 & 0 & 0 & 1 \end{array} \tag{2.11}$$

Thus we obtain

$$A \vee B \equiv \neg((\neg A) \wedge (\neg B)) \tag{2.12}$$

We have been using the values 0 and 1 for quite a while without introducing them rigorously. They can be defined as being equivalent to formulas using merely basic syntax elements.

Definition 9. We define the formulas

$$0 :\equiv (A \wedge (\neg A)) \tag{2.13}$$
$$1 :\equiv (A \vee (\neg A)) \tag{2.14}$$

The formulas equivalent to 0 and to 1 have own names.

Definition 10 (Tautology, Unsatisfiability)**.** If a formula F is semantically equivalent to 0, $F \equiv 0$, then F is called *unstaisfiable*.
If a formula F is semantically equivalent to 1, $F \equiv 1$, then F is called a *tautology*.

Simarly we can now formally introduce all other connectives such as the XOR

$$A \oplus B :\equiv ((A \wedge (\neg B)) \vee ((\neg A) \wedge B)) \tag{2.15}$$

or the equivalence connective

$$A \leftrightarrow B \;\Leftrightarrow\; (F \wedge G) \vee ((\neg F) \wedge (\neg G)) \tag{2.16}$$

Semantical equivalence is an *equivalence relation*, that is a mathematical relation with particular properties as we will see later in the course. For now it is enough to know that it groups formulas into disjoint subsets[2] of equivalent elements, so called *equivalence classes*. It structures the set of all syntactically correct formulas as visualized in 2.4.

The formulas $(B \vee (\neg B)), (\neg(\neg A) \vee (\neg(\neg(\neg A))))$ are all equivalent to 1 and therefore in the same equivalence class. They are all tautologies.

Number of equivalence classes The set of all syntactically correct formulas $\mathcal{E}_\mathcal{D}$ is infinite. The number of equivalence classes though is finite if the number of atomic formulas is finite. Given, for instance, that there are 26 atomic formulas $\mathcal{D} = \{A, B, C, \ldots, Z\}$, there are 2^{26} different input configurations (or lines if we wrote it like in 2.10). As each line can yield a 0 or 1, there are $2^{(2^{26})}$ different equivalence classes.

We will now related the equivalence connective and semantic equivalence:

[2] *Disjoint* means that every formula is contained in exactly one of those subsets.

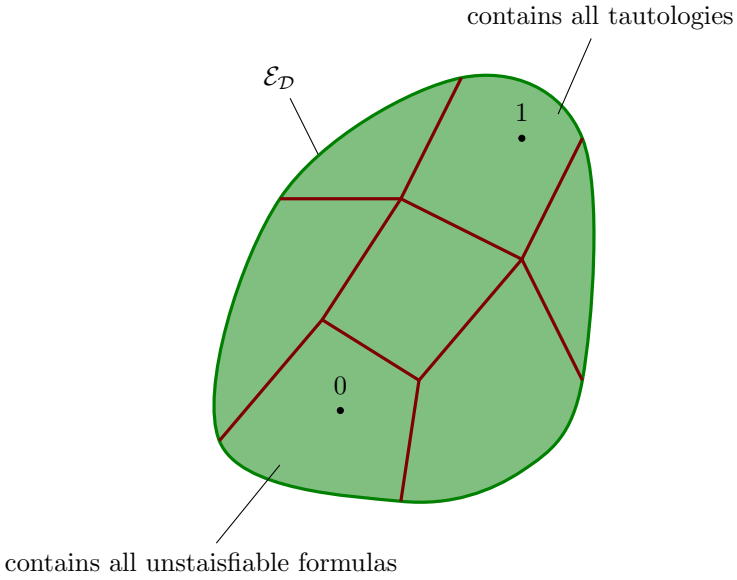


Figure 2.4: Equivalence classes in the set of all syntactically correct formulas $\mathcal{E}_\mathcal{D}$.

Theorem 2.3.1. *Two formulas F and G are semanically equivalent, i.e. $F \Leftrightarrow G$, if and only if (iff) the formula $F \leftrightarrow G$ is a tautology.*

Note that '$\leftrightarrow$' connects F and G syntactically whereas '$\Leftrightarrow$' connects the two semantically.

Proof. The formula $A \leftrightarrow B \equiv (F \wedge G) \vee ((\neg F) \wedge (\neg G))$ is a tautology iff F and G have the same truth values for all assignments. Thus it is a tautology iff F and G are semantically equivalent. □

Example 2.3.4 (Logical Laws). The following list contains important examples of semantical equivalences.

- Idempotence:
$$(F \wedge F) = F \qquad (F \vee F) \equiv F$$
- $(F \oplus F) \equiv 0$
- $A \leftrightarrow A \equiv 1$
- Symmetry
$$(F \wedge G) \equiv (G \wedge F) \qquad (F \vee G) \equiv (G \vee F)$$
- Associativity
$$(A \wedge (B \wedge C)) \equiv ((A \wedge B) \wedge C) \qquad (A \vee (B \vee C)) \equiv ((A \vee B) \vee C$$
- Absorption
$$((F \wedge G) \vee F) \equiv F \qquad ((F \vee G) \wedge F) \equiv F$$

- Distributivity

$$(F \wedge (G \vee H)) \equiv ((F \wedge G) \vee (F \wedge H)) \qquad (F \vee (G \wedge H)) \equiv ((F \vee G) \wedge (F \vee H)$$

- de Morgan

$$(\neg(F \wedge G)) \equiv ((\neg F) \vee (\neg G)) \qquad (\neg(F \vee G)) \equiv ((\neg F) \wedge (\neg G))$$

- double negation

$$(\neg(\neg F)) \equiv F$$

Note that the associativity for AND and XOR is not both ways but just for

$$A \wedge (B \oplus C) \equiv (A \wedge B) \oplus (A \wedge C) \tag{2.17}$$

Just as the XOR corresponds to addition, the AND corresponds to multiplication, i.e. $A \wedge B \equiv A \cdot B$. Associativity of addition and multiplication just extends to the logical gates.

Simplication of Notation So far we have been sticking closely to the syntax permitted by the definition 5. We will now introduce some simplifications, motivated by the equivalences above.

- We will allow all connectives introduced in the first part ($\oplus$, $\leftarrow$, $\leftrightarrow$) as they are equivalent to formulas with basic connectives.
- If brackets do not change the semantics they can be dropped as for instance in associative formulas (e.g. $A \wedge B \wedge C$). Based on this the following formulas are valid as well

$$\bigwedge_{i=1}^{n} A_i \equiv A_1 \wedge A_2 \wedge \ldots \wedge A_n \qquad \bigvee_{i=1}^{n} A_i \equiv A_1 \vee A_2 \vee \ldots \vee A_n \tag{2.18}$$

- We will introduce the following priority rules (or operator precedence): $\neg, (\wedge, \vee), (\oplus, \leftarrow, \rightarrow, \leftrightarrow)$. Again brackets can be dropped as long as the formula is in accordance with these priority rules.

2.4 Normal forms

Given a syntactically correct formula one obtains the semantics by writing down the truth table. Is it conversely possible to construct a syntactically correct formula that reproduces a given truth vector? The following example shows how this can be done.

Example 2.4.1. Let us consider the truth table with the atoms A, B and C. We would like to find a composed formula F that produces the truth vector in the

last column.

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

(2.19)

A first way to construct a formula F with the desired semantics is, to consider all the rows that have to be true. One has to be in row 3, row 4, row 5, row 6 *or* row 8. To check the validity of a single row one connects the atoms or their negation with ANDs. Thus we obtain the formula

$$\begin{aligned} F &= (\text{row } 3) \vee (\text{row } 4) \vee (\text{row } 5) \vee (\text{row } 6 \vee (\text{row } 8) \\ &= (\neg A \wedge B \wedge \neg C) \vee (\neg A \wedge B \wedge C) \\ &\quad \vee (A \wedge \neg B \wedge \neg C) \vee (A \wedge \neg B \wedge C) \vee (A \wedge B \wedge C) \end{aligned}$$

We will call this disjunction of conjunctions a *Disjunctive Normal Form* (DNF). Applying the formulas above, we can reduce the formula to a simpler semantically equivalent one. Employing associativity we obtain

$$(\neg A \wedge B \wedge \neg C) \vee (\neg A \wedge B \wedge C) \equiv (\neg A \wedge B) \vee \underbrace{(\neg C \wedge C)}_{\equiv 0} \equiv \neg A \wedge B$$

So we obtain

$$\begin{aligned} &\underbrace{(\neg A \wedge B \wedge \neg C) \vee (\neg A \wedge B \wedge C)}_{\equiv \neg A \wedge B} \vee \underbrace{(A \wedge \neg B \wedge \neg C) \vee (A \wedge \neg B \wedge C)}_{\equiv A \wedge \neg B} \vee (A \wedge B \wedge C) \\ &\equiv \underbrace{(\neg A \wedge B) \vee (A \wedge \neg B)}_{\equiv A \oplus B} \vee (A \wedge B \wedge C) \equiv A \oplus B \vee A \cdot B \cdot C \end{aligned}$$

using multiplication as an abbreviation of the AND.

A second approach to finding a formula F is to consider the false rows. Row 1 *and* row 2 *and* row 7 have to be false. Each of these rows is false if *any* of the subforrmulas is false. Thus we connect the atoms (or their negations) by ORs.

$$\begin{aligned} F &= \neg(\text{row } 1) \wedge \neg(\text{row } 2) \wedge \neg(\text{row } 7) \\ &= (A \vee B \vee C) \wedge (A \vee B \vee \neg C) \wedge (\neg A \vee \neg B \vee C) \end{aligned}$$

This conjunction of disjunctions is called a *Conjunctive Normal Form* (CNF). Using associativity again we can simplify

$$(A \vee B \vee C) \wedge (A \vee B \vee \neg C) \equiv (A \vee B) \wedge \underbrace{(C \vee \neg C)}_{\equiv 1}$$

to obtain

$$F \equiv (A \vee B) \wedge (\neg A \vee \neg A \vee C)$$

In the following definition the normal forms mentioned above will be introduced formally precise.

Definition 11 (Literal)**.** If $A \in \mathcal{D}$ is an atom then A and $\neg A$ are called literals. In other words, a literal is an atom or a negated atom.

Definition 12 (Conjunctive Normal Form)**.** A formula F is in Conjunctive Normal Form if there exist literals $L_{i,j}$ such that

$$\begin{aligned} F &= \bigwedge_{i=1}^{n} \left(\bigvee_{j=1}^{m_i} L_{i,j} \right) \\ &= (L_{1,1} \vee L_{1,2} \vee \ldots \vee L_{1,m_1}) \\ &\quad \wedge (L_{2,1} \vee \ldots \vee L_{2,m_2}) \\ &\quad \wedge \ldots \\ &\quad \wedge (L_{n,1} \vee \ldots \vee L_{n,m_n}) \end{aligned}$$

Definition 13 (Disjunctive Normal Form)**.** A formula F is in Disjunctive Normal Form if there exist literals $L_{i,j}$ such that

$$F = \bigvee_{i=1}^{n} \left(\bigwedge_{j=1}^{m_i} L_{i,j} \right)$$

Generally both methods we used in the example above can be applied to any truth vector. Thus any formula is semantically equivalent to a formula in CNF and to a formula in DNF.

2.5 Models and semantic conclusion

While for instance in physics is about which basic proposition are true the inverse questions are usually asked in logic. So instead of *how does the world work fundamentally?* one rather asks *which worlds are compatible with a given formula?*[3]. More practical: What assignments render a formula true?

A *possible world* is called a *model* as specified in the following definition.

Definition 14 (Model)**.** Let F be a formula and $\mathcal{A}$ an assignment which renders F true. Then $\mathcal{A}$ is a *model* of F with the notation

$$\mathcal{A} \vDash F \tag{2.20}$$

Example 2.5.1. Let us consider the following formula with its truth table

$F = (A$	$\wedge$	$B)$	$\rightarrow$	C
0	0	0	1	0
0	0	0	1	1
0	0	1	1	0
0	0	1	1	1
1	0	0	1	0
1	0	0	1	1
1	1	1	0	0
1	1	1	1	1

Only the assignment in the seventh row is *not* a model of F.

[3] 'World' is to be understood more philisophically as *reality*.

Some of the properties of models are the following

- Two formulas F and G are semantically equivalent, i.e. $F \equiv G$, if and only if[4]
$$\mathcal{A} \vDash F \quad \text{iff} \quad \mathcal{A} \vDash G \tag{2.21}$$
That is if F and G have the same models.
- A formula F is a tautology iff any assignment $\mathcal{A}$ is a model, i.e. F is true in any world.
- A formula F is unsatisfiable iff there exists no model, i.e. F is true in no world.

Having introduced the notion of a model we can now define a *semantic conclusion*.

Definition 15 (Semantic conclusion)**.** G is a semantic conclusion of $F_1, \ldots, F_n$ if any common model $\mathcal{A}$ for all F_i — that is $\mathcal{A} \vDash F_1, \ldots, \mathcal{A} \vDash F_n$ — is also a model of G — $\mathcal{A} \vDash G$. One also says, $F_1, \ldots, F_n$ semantically entail G. The semantic conclusion is denoted $\{F_a, \ldots, F_n\} \vDash G$ or $\{F_a, \ldots, F_n\} \Rightarrow G$

Remark 1 (Relation to semantic equivalence). The semantic conclusion is the one-sided variant of the semantic equivalence, similarly to $\rightarrow$ and $\leftrightarrow$ on the syntactical level. Therefore two formulas F and G are semantically equivalent iff F is the semantic conclusion of G and vice versa.

Remark 2 (Conflicting notation). Note that the symbol $\vDash$ has two meanings. It indicates semantic conclusion as well as models.

Example 2.5.2. Let's have a closer look at the following set of formulas $\{A, A \rightarrow B, B \rightarrow C\}$. The only model of the first formula is the assignment $\mathcal{A}_0(A) = 1$. The models of the second are (corresponding to the rows 1,2 and 4 in 2.9)

$$\mathcal{A}_1 : A \mapsto 0, B \mapsto 0$$
$$\mathcal{A}_2 : A \mapsto 0, B \mapsto 1$$
$$\mathcal{A}_3 : A \mapsto 1, B \mapsto 1$$

and similarly for the last formula

$$\mathcal{A}_4 : B \mapsto 0, C \mapsto 0$$
$$\mathcal{A}_5 : B \mapsto 0, C \mapsto 1$$
$$\mathcal{A}_6 : B \mapsto 1, C \mapsto 1$$

So merely the assignment

$$\tilde{\mathcal{A}} : A \mapsto 1, B \mapsto 1, C \mapsto 1$$

is a model that renders all formulas true, i.e. a common model for all formulas. As this is also a model for the atomic formula C we obtain

$$\{A, A \rightarrow B, B \rightarrow C\} \vDash C$$

[4] *if and only if* is usually abbreviated *iff*.

The semantic conclusion is related to the (syntactical) implication analogue to equivalence in the theorem (2.3.1).

Theorem 2.5.1. *A set of formulas $\{F_1, \ldots, F_2\}$ semantically entails a formula G — $\{F_1, \ldots, F_n\} \vDash G$ — iff the formula $\bigwedge_{i=1}^{n} F_i \to G$ is a tautology.*

One has to carefully distinguish the syntactical and the semantic level. The expression $\{F_1, \ldots, F_n\} \vDash G$ relates the set of formulas $\{F_1, \ldots, F_n\}$ *semantically* with the formula G. The expression $\bigwedge_{i=1}^{n} F_i \to G$ connects the two syntactically and yields another syntactically correct formula. Only by demanding this formula to be a tautology there emerges a semantic criterion.

Proof. The implication $A \to B$ is equivalent to $\neg A \vee B$. Therefore we obtain

$$\begin{aligned} \bigwedge_{i=1}^{n} F_i \to G &\equiv \neg(F_1 \wedge F_2 \wedge \ldots \wedge F_n) \vee G \\ &\equiv \neg F_1 \vee \neg F_2 \vee \ldots \vee \neg F_n \vee G \end{aligned}$$

Let's now consider models of this formula. If an assignment renders all F_i true — $\mathcal{A}(F_i) = 1 \ \forall i$ — then the formula is true iff the assignment also renders G true. This is just the definition of a semantic conclusion. □

Remark 3. A formula F is a tautology iff F is a conclusion of 1, i.e. $1 \vDash F$. So F is a tautology iff the implication $1 \to F$ is a tautology.

A formula F is unsatisfiable iff 0 is a conclusion of F, i.e. $F \vDash 0$. So F is unsatisfiable if the implication $F \to 0$ is a tautology.

2.6 Proof theory of propositional logic

The objective is to develop a calculus to decide semantic questions such as

- Is F a tautology?
- Is F unsatisfiable?
- Does $\{F_1, \ldots, F_n\} \vDash G$ hold?

In the following examples we will consider the "tautology" question for CNFs and DNFs.

Example 2.6.1 (Tautology problem of CNF). Given a formula F in CNF, can we decide whether it is a tautology? The formula can be rewritten as a conjunction of subformulas F_i

$$\begin{aligned} F &= \underbrace{(L_{1,1} \vee \ldots L_{1,m_1})}_{=:F_1} \wedge \ldots \underbrace{(L_{n,1} \vee \ldots L_{n,m_n})}_{=:F_n} \\ &= F_1 \wedge \ldots \wedge F_n \end{aligned}$$

So F is a tautology iff all F_i are tautologies. The F_i's in turn are tautologies if at least one atom reappears negated. That is F_i contains an atom $\tilde{A}$ as well as $\neg\tilde{A}$.

Example 2.6.2 (Tautology problem of DNF). Can we derive a similar criteria for the subformulas of a DNF

$$F = \bigvee_{i=1}^{n} \left(\bigwedge_{j=1}^{m_i} L_{i,j} \right) = F_1 \vee \ldots \vee F_n$$

to decide whether F is a tautology or not? Unfortunately the problem does not *localize* in the same manner. For a CNF to be a tautology, *all* subformulas have to be true for any assignment, and therefore have to be tautologies themselves. Whereas for a DNF just *one* subformula has to be true for any assignment. This could be different subformulas for different assignments. The question whether F is a tautology can only be decided considering the entire formula and not by looking at the single subformulas independently.

There seems to be no other way than to write down the entire truth table. It contains 2^l entries for a formula containing l atoms. Thus the table grows very fast with the number of atoms.

Relating DNF and CNF For a formula F in CNF we immediately obtain a formula in DNF that is semantically equivalent to $\neg F$. Employing de Morgan's law yields

$$\begin{aligned} \neg F = \neg \bigvee_{i=1}^{n} \left(\bigwedge_{j=1}^{m_i} L_{i,j} \right) &\equiv \bigwedge_{i=1}^{n} \left(\neg \bigwedge_{j=1}^{m_i} L_{i,j} \right) \\ &\equiv \bigwedge_{i=1}^{n} \left(\bigvee_{j=1}^{m_i} \neg L_{i,j} \right) \end{aligned}$$

This is a DNF as any negated literal is just another literal. The argument also works for a F being a DNF. Then one obtains a CNF semantically equivalent to $\neg F$.

As F being a tautology is equivalent to $\neg F$ being unsatisfiable, the satisfiability problem of a DNF can be solved analogue to the tautology problem of a CNF. Conversely the satisfiability problem of a CNF — analogue to the tautology problem of a DNF — can merely be decided by writing down the entire truth table.

Comparison of computational hardness How hard is it to decide the tautology problem for a CNF and for a DNF? Consider a formula F of length l (length being the number of literals). If F is in CNF one merely has to look at all pairs of literals in every subformula, following the criteria in the example above. So one has to perform at most $c \cdot l^2$ steps (c being some finite constant) to decide whether F is a tautology.
In the case of F being a DNF one has to write down the entire truth table, containing 2^l rows. The number of computational steps is thus exponential in l, growing much faster[5] than quadratically. In fact the two problems belong to different so-called *complexity classes*.

[5]Consider for instance 10 literals $2^{10} = 1024 > 10^2 = 100$ or 20 literals $2^{20} = 1048576 > 20^2 = 400$

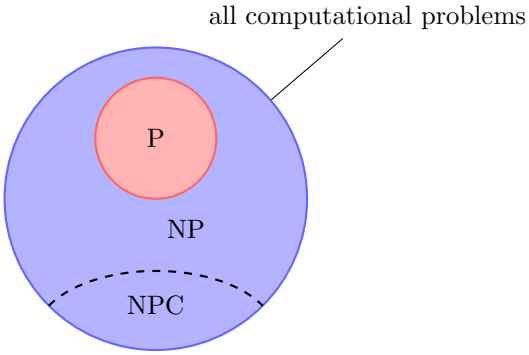


Figure 2.5: The set of computational problems is divided into classes of different complexity.

2.6.1 An Excursion into Complexity Theory

Complexity theory is about classifying computational problems by their computational difficulty. The set P contains all problems that can be solved in polynomial time, i.e. the number of computational steps is at most a polynomial of the input size. P is a subset of NP, containing all problems for which one can verify a given solution in polynomial time, whereas the solution may not be found in polynomial time.

A third class is NP-complete (NPC). Problems in NPC are as hard as any other problem in NP. That means, any problem in NP can be reduced to a problem in NPC in polynomial time. Consequently any problem in NPC can be reduced to another problem in NPC in polynomial time. Thus NPC is subset of NP and disjoint with P.

Example 2.6.3 (Elements in P). P contains

- the tautology problem for CNF
- the (un)satisfiability problem for DNF.

Example 2.6.4 (Elements in NPC). NPC contains

- the tautology problem for DNF
- the satisfiability problem for CNF
- the problem to find a semantically equivalent DNF for a given CNF and vice versa.

The last problem follows from the first two: If we could find a G in CNF for any F in DNF with $F \equiv G$ in polynomial time we could solve the tautology problem for a DNF in polynomial by first turning it into a CNF and then applying the criteria above.

Real life problems? Are problems in real life hard? Yes, they are, as one can see in the following example.

Example 2.6.5 (Sudoku). Sudoku is an NPC problem as we can reduce it to the satisfiability of a CNF. As one has to find a solution satisfying *all* conditions on the rows, columns and subboxes, it is a problem of the form

$$(\text{condition 1}) \wedge (\text{condition 2}) \wedge \dots$$

Therefore it is in CNF. To find a solution is just the same as proving satisfiability. The satisfiability problem for a CNF is in NPC, and so is the Sudoku problem.

Appendices

Appendix A

Proof techniques

In this appendix we will briefly review common proof techniques used throughout the course.

A.1 Proof by contradiction

In order to prove a proposition to be true, one can show that the assumption of the proposition being false leads to a contradiction. So the usual approach is to negate the proposition and then construct a contradiction.

Example A.1.1 ($\sqrt{2}$ is irrational).

Proposition A.1.1. The squareroot of 2 is irrational.

Proof. The negative of the proposition is "$\sqrt{2}$ is rational".

Given the negative proposition we will now construct a contradiction. If $\sqrt{2}$ is rational there is an irreducible fraction

$$\frac{a}{b} = \sqrt{2} \quad a, b \in \mathbb{N} \quad \Rightarrow \quad a^2 = 2b^2 \tag{A.1}$$

As a^2 is even, a has to be even. Thus $a^2 = 4a'^2$ for some other natural number $a' \in \mathbb{N}$. Therefore $b^2 = 2a'^2$ and consequently b is even as well. This contradicts a/b being an irreducible fraction. □

A.2 Proof by induction

If the proposition is a statement for all natural numbers, *induction* is usually a convenient approach.

One shows first that the proposition holds for a first natural number (the base case). Then one proves that assuming the proposition holds for a general natural number n it also holds for $n + 1$. The second part of the proof is called the *induction step*. Applying the induction step over and over again starting from 1 one obtains that the proposition holds for 2,3,4,... and eventually all natural numbers.