Formal Logic

Winter term 2018/19

PD Dr Dirk Frettlöh
Technische Fakultät
Universität Bielefeld

February 19, 2019

Contents

To the reader: These notes were written for the course "Formal Logic" in winter terms 2017/18 and 2018/19. They contain probably flaws, major and minor ones. Please let me know if you find one.

Text in grey was not covered in 2018/19.

1 Propositional Logic

1.1 Basics

16. Oct.

Logic is about truth or falseness of statements, such as

1. The university building is beautiful
2. $2+2=5$
3. If Ada Lovelace is a philosopher, she is human
4. If Ada Lovelace is human, she is a philosopher
5. Please do not smoke
6. Hello Kitty

The statements 5 and 6 are out of our scope, since they are neither "true" or "false".

Some statements cannot be divided into smaller statements (like 1 and 2), other are combinations of smaller statements (like 3 and 4). For instance, 3 is of the form "If A (is true) then B (is true)".

A statement that cannot be divided into smaller pieces is called **atomic formula**. More complex statements are build from atomic formulas as follows.

Definition 1.1. (and notation) Atomic formulas are abbreviated by $A,B,C,\ldots$, or by $A_1,A_2,\ldots$.

A **formula** (in propositional logic) is defined inductively as follows:

- Each atomic formula is a formula.
- If F and G are formulas, then $F \vee G$, $F \wedge G$ and $\neg F$ are formulas.

Everything that can be build in this way is a formula.

Example 1.1. $\neg((A \wedge B) \vee C)$ is a formula. It's partial formulas are $A,B,C,A \wedge B,(A \wedge B) \vee C$, and $\neg((A \wedge B) \vee C)$. But $(A\wedge$, or $\vee C$, or (are not formulas.

Notation

- $A,B,C,\ldots$, or $A_1,A_2,\ldots$ denote atomic formulas.
- $F,G,H,\ldots$, or $F_1,F_2,\ldots$ denote formulas.
- $F \Rightarrow G$ is short for $(\neg F) \vee G$. $F \Leftrightarrow G$ is short for $(F \wedge G) \vee (\neg F \wedge \neg G)$,

Up to here this is entirely abstract, the symbols are just symbols (**syntax**: symbols and rules). We want to give them some meaning (**semantics**):

Definition 1.2. The elements of $\{0,1\}$ are **truth values**. (Intuition: 0 = false, 1 = true). A **valuation** (or truth assignment) of a set $M = \{A,B,C,\ldots\}$ of atomic formulas is a map

$$\mathcal{A} : \{A,B,C,\ldots\} \to \{0,1\}$$

$\mathcal{A}$ extends to all formulas by

1. $\mathcal{A}(F \wedge G) = \begin{cases} 1 & \text{if } \mathcal{A}(F) = \mathcal{A}(G) = 1 \\ 0 & \text{else} \end{cases}$

2. $\mathcal{A}(F \vee G) = \begin{cases} 0 & \text{if } \mathcal{A}(F) = \mathcal{A}(G) = 0 \\ 1 & \text{else} \end{cases}$

3. $\mathcal{A}(\neg F) = \begin{cases} 0 & \text{if } \mathcal{A}(F) = 1 \\ 1 & \text{else} \end{cases}$

Example 1.2. Let $\mathcal{A}(A) = 1, \mathcal{A}(B) = 1, \mathcal{A}(C) = 0$. What is $\mathcal{A}(\neg((A \wedge B) \vee C))$?

$$\begin{aligned} \mathcal{A}(\neg((A \wedge B) \vee C)) = 1 - \mathcal{A}((A \wedge B) \vee C) &= 1 - \begin{cases} 0 & \text{if } \mathcal{A}(A \wedge B) = \mathcal{A}(C) = 0 \\ 1 & \text{else} \end{cases} \\ &= 1 - \begin{cases} 0 & \text{if } \mathcal{A}(A \wedge B) = 0 \\ 1 & \text{else} \end{cases} = 1 - 1 = 0 \quad (\text{since } \mathcal{A}(A) = \mathcal{A}(B) = 1) \end{aligned}$$

A more efficient method is to use **truth tables**. A formula with n atomic formulas has 2^n possible distinct valuations. Hence we might define $\neg, \vee$ and $\wedge$ by truth tables, and we may construct the truth tables also for $\Rightarrow$ and $\Leftrightarrow$ by considering all possibilities:

$\mathcal{A}(F)$	$\mathcal{A}(G)$	$\mathcal{A}(\neg(F))$	$\mathcal{A}(F \wedge G)$	$\mathcal{A}(F \vee G)$	$\mathcal{A}(F \Rightarrow G)$ $= \mathcal{A}(\neg F \vee G)$	$\mathcal{A}(F \Leftrightarrow G)$ $= \mathcal{A}((F \wedge G) \vee (\neg F \wedge \neg G))$
0	0	1	0	0	1	1
0	1		0	1	1	0
1	0	0	0	1	0	0
1	1		1	1	1	1

Interpretations of these rules are that $\vee$ means "or", $\wedge$ means "and", $\neg$ means "not". It can be discussed whether "or" really is a good interpretation: the statement "I will have beef or chicken" usually means that I take one of them, not both. Nevertheless, this is just an interpretation. In the sequel we want to formalize, hence do not care about the interpretation. A further operator, covering the other meaning of "or", is "exclusive or": $\mathcal{A}(A \oplus B) = 1$ if and only if $\mathcal{A}(A) \neq \mathcal{A}(B)$.

An interpretation of $\Rightarrow$ is "implies", an interpretation of $\Leftrightarrow$ is "if and only if". This is in accordance with the use in mathematics, but can be discussed with respect to natural language. Nevertheless, in order to formalize we ignore ambiguities in the language.

In order to spare brackets we introduce an order of operations:

$\neg$ before $\wedge$ and $\vee$ before $\Rightarrow$ before $\Leftrightarrow$.

Hence we may write for instance $\neg A \vee B \Rightarrow C$ rather than $((\neg A) \vee B) \Rightarrow C$.

Main questions/aims: 1. Which formulas have the same meaning? I.e., for which formulas F, G we have that $\mathcal{A}(F) = \mathcal{A}(G)$ holds for all valuations?

2. Which formulas have at least one valuation making it true? I.e., for which formulas F we have that there is at least one valuation such that $\mathcal{A}(F) = 1$ holds?

The focus will be on the algorithmic treatment of this questions. Now we need some terminology. If we write $\mathcal{A}(F)$ in the sequel it is always implicitly assumed that $\mathcal{A}$ is defined for all atomic formulas contained in F.

Definition 1.3. A valuation $\mathcal{A}$ **satisfies** a formula F if $\mathcal{A}(F) = 1$. Notation: $\mathcal{A} \models F$.

Otherwise ($\mathcal{A}(F) = 0$) we say $\mathcal{A}$ does not satisfy F. Notation: $\mathcal{A} \not\models F$.

F is **satisfiable** if there is a valuation $\mathcal{A}$ such that $\mathcal{A}(F) = 1$. Otherwise F is unsatisfiable.

F is a **tautology** if for all valuations $\mathcal{A}$ holds: $\mathcal{A}(F) = 1$.

F is **equivalent** to G if for all valuations holds: $\mathcal{A}(F) = \mathcal{A}(G)$. Notation: $F \equiv G$.

Example 1.3. $A \vee \neg A$ and $\neg A \Rightarrow (A \Rightarrow B)$ are tautologies. $A \wedge \neg A$ is unsatisfiable (truth tables!)

Remark 1.1. F and G may contain different atomic formulas and nevertheless $F \equiv G$ may hold (e.g. if both formulas are tautologies).

1.2 Calculation rules

Rules like $F \wedge G \equiv G \wedge F$ are obvious (are they?), but there are more sophisticated ones:

Theorem 1.1. *Let F, G, H are formulas. Then the following equivalences hold:*

$F \wedge F \equiv F$, $F \vee F \equiv F$	*(Idempotence)*
$F \wedge G \equiv G \wedge F$, $F \vee G \equiv G \vee F$	*(Commutativity)*
$(F \wedge G) \wedge H \equiv F \wedge (G \wedge H)$, $(F \vee G) \vee H \equiv F \vee (G \vee H)$	*(Associativity)*
$F \wedge (F \vee G) \equiv F$, $F \vee (F \wedge G) \equiv F$	*(Absorption)*
$F \wedge (G \vee H) \equiv (F \wedge G) \vee (F \wedge H)$, $F \vee (G \wedge H) \equiv (F \vee G) \wedge (F \vee H)$	*(Distributivity)*
$\neg\neg F \equiv F$	
$\neg(F \wedge G) \equiv \neg F \vee \neg G$, $\neg(F \vee G) \equiv \neg F \wedge \neg G$	*(de Morgan's laws)*
If F is a tautology, then $F \vee G \equiv F$, $F \wedge G \equiv G$	
If F is unsatisfiable, then $F \vee G \equiv G$, $F \wedge G \equiv F$	

All identities can be proven by truth tables, compare Exercise 5 on Exercise Sheet 2. 23. Oct.

Example 1.4. One can use truth tables to check whether the two formulas $(A \vee (B \vee C)) \wedge (C \vee \neg A)$ and $(\neg A \wedge B) \vee C$ are equivalent. But one can use the equivalences above as well:

$$(A \vee (B \vee C)) \wedge (C \vee \neg A) \overset{(Ass.,Comm.)}{\equiv} (C \vee (A \vee B)) \wedge (C \vee \neg A) \overset{(Distr.)}{\equiv} C \vee ((A \vee B) \wedge \neg A)$$
$$\overset{(Ass.,Comm.,Distr)}{\equiv} C \vee ((\neg A \wedge A) \vee (\neg A \wedge B)) \overset{(unsat.)}{\equiv} C \vee (\neg A \wedge B) \overset{(Comm.)}{\equiv} (\neg A \wedge B) \vee C.$$

In order to show some very formal aspect, namely, some proof in formal logic, we state and prove the following theorem.

Theorem 1.2 (Replacement theorem)**.** *Let F, G be formulas with $F \equiv G$. Let F be a partial formula of H. Let H' be the formula arising from H by replacing F with G. Then $H \equiv H'$.*

Proof. (By induction on the inductive construction of a formula)

Base of induction: Let H be an atomic formula. Then $H = F$, hence $H' = G$, hence $H' \equiv H$.

Induction step: Let F be some formula. Let the claim be true for all formulas smaller than F (in particular all partial formulas of F except F itself).

Case 0: $H = F$, then exactly as above.

Case 1: $H = \neg H_1$. By induction hypothesis $H_1 \equiv H_1'$, hence $H = \neg H_1 \equiv \neg H_1' = H'$.

Case 2: $H = H_1 \vee H_2$. Let us assume (without loss of generality) that F is a partial formula of H_1. By the induction hypothesis holds $H_1 \equiv H_1'$, hence $H = H_1 \vee H_2 \equiv H_1' \vee H_2 = H'$.

Case 3: $H = H_1 \wedge H_2$ (completely analogous to Case 2). □

This proof illustrates a general phenomenon: most proofs in formal logic are technical, not elegant, and yield no deeper insight. Because of this we will omit most proofs in the sequel.

1.3 Normal forms

Truth tables yield an algorithmic method to answer for a given formula F whether F is satisfiable, or whether F is equivalent to some further given formula G. But this is not efficient: If F contains n atomic formulas then the number of rows in the truth table is 2^n. In the sequel we want to introduce one or two efficient decision procedures to answer the question "is F satisfiable?" efficiently, at least for certain classes of formulas. The methods require the formulas to have some normal form. This is to be expected, since it is known that in general the question "is F satisfiable" is NP-complete.

Notation: $\bigwedge_{i=1}^{n} F_i = F_1 \wedge F_2 \wedge \cdots \wedge F_n$; $\bigvee_{i=1}^{n} F_i = F_1 \vee F_2 \vee \cdots \vee F_n$.

Definition 1.4. A **literal** is a formula of the form A or $\neg A$, where A is some atomic formula.

A formula F is in **disjunctive normal form (DNF)** if

$$F = \bigvee_{i=1}^{n} \Big(\bigwedge_{j=1}^{m_i} L_{ij} \Big) \quad \text{where } L_{ij} \text{ are literals.}$$

A formula of the form $L_1 \wedge L_1 \wedge \cdots \wedge L_n$ (where L_i are literals) is called *conjunctive clause*.

A formula F is in **conjunctive normal form (CNF)** if

$$F = \bigwedge_{i=1}^{n} \Big(\bigvee_{j=1}^{m_i} L_{ij} \Big) \quad \text{where } L_{ij} \text{ are literals.}$$

A formula of the form $L_1 \vee L_1 \vee \cdots \vee L_n$ (where L_i are literals) is called *disjunctive clause*.

Example 1.5. All of the following formulas are in DNF:

$$(A \wedge \neg B \wedge \neg C) \vee (\neg D \wedge E \wedge F), \quad (A \wedge B) \vee C, \quad A \wedge B, \quad A.$$

However, the following formulate are not in DNF:

$\neg(A \vee B)$ (since an OR is nested within a NOT), $A \vee (B \wedge (C \vee D))$ since an OR is nested within an AND.

Theorem 1.3. *Each formula has some equivalent formula in DNF, and also some equivalent formula in CNF.*

The proof of this result is long and technical and uses induction on the construction of the formula. It essentially consists of showing that the following algorithm terminates and yields a CNF for F:

Algorithm 1.1. First convert all partial formulas of F of the form $G \Rightarrow H$ into $\neg G \vee H$, and all partial formulas of the form $G \Leftrightarrow H$ into $(G \wedge H) \vee (\neg G \wedge \neg H)$.

1. Replace in F each
   - $\neg\neg G$ by G
   - $\neg(G \wedge H)$ by $\neg G \vee \neg H$
   - $\neg(G \vee H)$ by $\neg G \wedge \neg H$
2. Replace in F each
   - $G \vee (H \wedge J)$ by $(G \vee H) \wedge (G \vee J)$
   - $(G \wedge H) \vee J$ by $(G \vee J) \wedge (H \vee J)$
3. Remove repetitions of clauses, if necessary.

The corresponding algorithm to construct the DNF of a formula F is obtained by replacing step 2. above by

2. Replace in F each

- $G \wedge (H \vee J)$ by $(G \wedge H) \vee (G \wedge J)$
- $(G \vee H) \wedge J$ by $(G \wedge J) \vee (H \wedge J)$

Example 1.6. Let $F = (A \wedge B \wedge C) \vee (D \wedge E)$. This has DNF. Let us apply the algorithm to transform F into CNF:

$$\begin{aligned}(A \wedge B \wedge C) \vee (D \wedge E) &= \big((A \wedge B \wedge C) \vee D\big) \wedge \big((A \wedge B \wedge C) \vee E)\big)\\ &= (A \vee D) \wedge (B \vee D) \wedge (C \vee D) \wedge (A \vee E) \wedge (B \vee E) \wedge (C \vee E)\end{aligned}$$

Remark 1.2. By using the distributivity law in particular examples (see exercises) one realizes that the distributivity law generalizes as in the example above, respectively in the most general form to

$$(\bigvee_{i=1}^{n} F_i) \wedge (\bigvee_{j=1}^{m} G_j) \equiv \bigvee_{i=1}^{n} (\bigvee_{j=1}^{m} F_i \wedge G_j) \quad \text{resp.} \quad (\bigwedge_{i=1}^{n} F_i) \vee (\bigwedge_{j=1}^{m} G_j) \equiv \bigwedge_{i=1}^{n} (\bigwedge_{j=1}^{m} F_i \vee G_j)$$

There is an alternative algorithm to produce CNF or DNF using truth tables. Assume that we would have already constructed the truth table of some formula F, containing atomic formulas $A_1, \dots, A_n$.

Algorithm 1.2. In order to construct the DNF of F:

- Each row where $\mathcal{A}(F) = 1$ yields a part $(L_1 \wedge \cdots \wedge L_n)$. If $\mathcal{A}(A_i) = 1$ in this row then set $L_i = A_i$, else set $L_i = \neg A_i$ $(i = 1, \dots, n)$
- Connect the parts by $\vee$

In order to construct the CNF of F:

- Each row where $\mathcal{A}(F) = 0$ yields a part $(L_1 \vee \cdots \vee L_n)$. If $\mathcal{A}(A_i) = 0$ in this row then set $L_i = A_i$, else set $L_i = \neg A_i$ $(i = 1, \dots, n)$
- Connect the parts by $\wedge$

Example 1.7. Consider $F = \neg(A \Rightarrow B) \vee \neg(A \vee B \vee C)$. The truth table is

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

We read off for the DNF

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

We read off for the CNF

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

Theorem 1.4. *For each formula F there is a formula G in CNF where each disjunctive clause has at most three literals, such that F is satisfiable if and only if G is satisfiable.*

30. Okt

Proof. By Theorem 1.3 F has a CNF. We just need to show how to replace each disjunctive clause in the CNF that has more than three literals by one or more disjunctive clauses with three literals.

Assume there is a clause H with more than three literals: $H = A \vee B \vee G$. We replace H in the CNF by $H' = (A' \Leftrightarrow A \vee B) \wedge (A' \vee G)$, introducing a new atomic formula A', that does not occur in F. If H is satisfiable, any valuation $\mathcal{A}$ for H with $\mathcal{A}(H) = 1$ can be extended to H' by setting $\mathcal{A}(A') := \mathcal{A}(A \vee B)$. Then $\mathcal{A}(A' \Leftrightarrow A \vee B) = 1$ implies $A' \equiv A \vee B$. Hence

$$\mathcal{A}(A \vee B \vee F) = \mathcal{A}(A' \vee F) = \mathcal{A}\big((A' \Leftrightarrow A \vee B) \wedge (A' \vee F)\big).$$

Thus H' is satisfiable if and only H is. Because of

$$\begin{aligned} A' \Leftrightarrow A \vee B &\equiv (A' \Rightarrow A \vee B) \wedge (A \vee B \Rightarrow A') \equiv (\neg A' \vee A \vee B) \wedge (\neg(A \vee B) \vee A') \\ &\equiv (\neg A' \vee A \vee B) \wedge ((\neg A \wedge \neg B) \vee A') \equiv (\neg A' \vee A \vee B) \wedge (\neg A \vee A') \wedge (\neg B \vee A'), \end{aligned}$$

the new formula has CNF, too. This completes the proof. □

Let us also count how the number of clauses grows during this replacement process: replacing H by H' means replacing a single clause with $m \geq 4$ literals by one clause with $m-1$ literals plus one more clause with three literals plus two clauses with two literals. This can be applied until all clauses have three literals or less. Hence if our general formula F in CNF has n clauses with at most m literals each, our counting shows that we can reduce each clause in $m-3$ steps to several small clauses with at most three literals. How many clauses? In each step we trade one clause for four clauses, so in each step the number grows by three. So each long clause (of length m) is replaced by at most $3(m-3)$ of length three (or less). Altogether we end up with at most $3n(m-3)$ clauses. This is polynomial in n and m, resp. in $\max\{n,m\}$.

Remark 1.3. The question whether a formula G is satisfiable is NP-complete in general (Cook's Theorem 1971). He showed that the question of the satisfiability of any formula in propositional logic can be reduced in polynomial time to the question about satisfiability of formulas in CNF (sometimes called CNFSAT). We just showed that, the more general problem CNFSAT can be reduced (in polynomial time) to the question whether a given formula in CNF where each clause has at most three literals is satisfiable. This latter problem is called 3SAT ("three-satisfiability"). Hence 3SAT is also NP-complete (Karp 1972). In contrast to this we will see in the next section a special case where the question for satisfiability is efficiently computable Btw: the corresponding problem 2SAT — satisfiablity of a formula in CNF where each clause has only two literals — is decidable in polynomial time.

Remark 1.4. The corresponding problem of solving a formula in DNF is decidable in poyniomial time. Hence we cannot expect to have a general way to transform any formula in CNF into an equivalent formula in DNF with polynomial costs. In fact already the length of the formula can grow exponentially in general. The worst case is achieved for instance by

$$F = \bigwedge_{i=1}^{n} A_i \vee B_i = (A_1 \vee B_1) \wedge (A_2 \vee B_2) \wedge \cdots \wedge (A_{n-1} \vee B_{n-1}) \wedge (A_n \vee B_n)$$

(this CNF having essentially length $2n$). The DNF of F is then

$$\begin{array}{l}(A_1 \wedge A_2 \wedge \cdots \wedge A_{n-1} \wedge A_n) \vee \\ (A_1 \wedge A_2 \wedge \cdots \wedge A_{n-1} \wedge B_n) \vee \\ (A_1 \wedge A_2 \wedge \cdots \wedge A_{n-2} \wedge B_{n-1} \wedge A_n) \vee \\ \cdots \vee (B_1 \wedge B_2 \wedge \cdots \wedge B_{n-1} \wedge A_n) \\ \vee (B_1 \wedge B_2 \wedge \cdots \wedge B_{n-1} \wedge B_n)\end{array}$$

of length 2^n. (The analogue holds if one exchanges all $\vee$ with $\wedge$ and vice versa.)

However, there are some classes of formulas where there is an efficient decision procedure for satisfiability.

1.4 Horn formulas

Definition 1.5. A formula F is called a *Horn formula*, if it is in CNF and if each disjunctive clause has at most one positive literal (i.e. one literal not containing $\neg$)

For example, $F = (\neg A \vee \neg B \vee C) \wedge (A \vee \neg D) \wedge (\neg A \vee \neg B \vee \neg C) \wedge D \wedge \neg E$ is a Horn formula, whereas $G = (A \vee B \vee \neg C) \wedge (A \vee \neg B \vee \neg C)$ is not.

Each Horn formula can be written more intuitively using implications. E.g., the formula F above can be written as

$$(A \wedge B \Rightarrow C) \wedge (D \Rightarrow A) \wedge (A \wedge B \wedge C \Rightarrow 0) \wedge (1 \Rightarrow D) \wedge (E \Rightarrow 0),$$

where 1 stands for some arbitrary tautology and 0 stand for some arbitrary unsatisfiable formula.

There is an efficient algorithm checking for satisfiability of some Horn formula.

Algorithm 1.3. Input: some Horn formula F.

1. If F contains some partial formula $(1 \Rightarrow A)$, then mark all (!) literals containing the atomic formula A in F

2. **while** F contains some partial formula G with $G = (A_1 \wedge \cdots \wedge A_n \Rightarrow B)$ (with $n \geq 1$) where all A_i are marked (and B is not) **do**

   if $B \neq 0$ **then** mark all (!) literals containing B in F,

   else $(B = 0)$ return "F is unsatisfiable" STOP

3. Return "F is satisfiable" STOP

A valuation satisfying F is then given by setting $\mathcal{A}(A_i) = 1$ for all marked atomic formulas A_i and $\mathcal{A}(A_j) = 0$ for all unmarked atomic formulas A_j.

Theorem 1.5. *The algorithm above is correct for all Horn formulas and stops after at most k marking steps, where k denotes the number of atomic formulas contained in F.*
Moreover, the algorithm yields a minimal *satisfying valuation $\mathcal{A}$ for F. That is, if we denote the set of all i with $\mathcal{A}(A_i) = 1$ by I, then for all valuations $\mathcal{A}'$ with $\mathcal{A}' \models F$ holds that $i \in I$ implies $\mathcal{A}'(A_i) = 1$.*

The last statement can be seen as follows: when the algorithm starts all atomic formulas A_i are unmarked, i.e. their value is 0. All markings are forced, i.e. all marked A_i necessarily need to have value 1. If F is satisfiable the algorithm stops with a satisfying valuation, in which all atomic formulas with value 1 are forced to have value 1.

1.5 Finiteness Theorem

As well as a single formula we can consider finite (or infinite) sets of formulas, and ask for simultaneous satisfiability of the formulas. That is, if $M = \{F_1, F_2, \ldots, F_n\}$ is a set of formulas, we want to find a valuation $\mathcal{A}$ such that $\mathcal{A}(F_i) = 1$ for all $i = 1, \ldots, n$.

Remark 1.5. A set of formulas $\{F_1, F_2, \ldots, F_n\}$ is satisfiable if and only if $F = \bigwedge_{i=1}^{n} F_i$ is satisfiable.

For later purposes we want to consider also *infinite* sets of formulas. In this context the next result will be important in the sequel.

Theorem 1.6 (compactness theorem)**.** *An infinite set M of formulas is satisfiable if and only if each finite subset of M is satisfiable.*

The proof is long, technical, ugly, not illuminating and is contained for instance in UWE SCHÖNING: LOGIC FOR COMPUTER SCIENTISTS. Nevertheless it contains some important and interesting fact.

Theorem 1.7 (König's Lemma)**.** *Let T be a tree with infinitely many nodes such that each vertex has only finitely many neighbours. Then T contains an infinite path. (That is, a path $v_1 v_2 \cdots$ with infinitely many nodes such that $v_i \neq v_j$ for $i \neq j$.)*

Proof. (Entirely from wikipedia:) Let v_i be the set of vertices. Since T is an infinite tree we know that this vertex set is infinite and the graph is connected (i.e., for any two vertices v_i, v_j there is a path in T from v_i to v_j).

Start with any vertex v_1. Every one of the infinitely many vertices of G can be reached from v_1 with a (simple) path, and each such path must start with one of the finitely many vertices adjacent to v_1. There must be one of those adjacent vertices through which infinitely many vertices can be reached without going through v_1. If there were not, then the entire graph would be the union of finitely many finite sets, and thus finite, contradicting the assumption that the graph is infinite. We may thus pick one of these vertices and call it v_2.

Now infinitely many vertices of G can be reached from v_2 with a simple path which does not include the vertex v_1. Each such path must start with one of the finitely many vertices adjacent to v_2. So an argument similar to the one above shows that there must be one of those adjacent vertices through which infinitely many vertices can be reached; pick one and call it v_3. Continue. $\square$

As an application consider the problem whether there are infinitely many prime numbers $p_1, p_2, p_3, \ldots$ such that p_i is a prefix of p_{i+1} for all $i \in \mathbb{N}$. (An example might (or might nto) be: (1 not), 13, 131, 1319, (1319? not, 13195? not), 1319951,...) By König's lemma there is such a sequence:

Build a prefix tree for all natural numbers. I.e., arrange all natural numbers in a tree T. Let 0 be the root, let $1, 2, \ldots, 9$ the children of 0. For $n \in \mathbb{N}$ let $n0, n1, n2, \ldots, n9$ be the children of n. (E.g., the children of 13 are 130, 131, 132,...139, and all descendants of 13 are decimal number starting where the first digit is 1 and the second digit is 3). This tree has several infinite paths. Each node but 0 has 10 children.

Now construct a subtree T' of T by deleting all nodes that are not prime numbers and that have no prime numbers as descendants. Since there are infinitely many prime numbers, and all of them are nodes in T', the conditions of König's Lemma are fulfilled. Hence T' contains an infinite path. The nodes on this path are prefixes of their descendants. The path contains infinitely many prime numbers, since otherwise it would have been deleted from some point on.

Note that the proof does not tell us which numbers will do: it shows only the existence of such sequence, not the construction.

1.6 Consequences I

Definition 1.6. A formula G is a **consequence** of $F_1, \ldots, F_n$, if if whenever $\mathcal{A}(F_1) = \cdots = \mathcal{A}(F_n) = 1$ then $\mathcal{A}(G) = 1$. In this case we write shortly $\{F_1, \ldots, F_n\} \vdash G$. 6. Nov.

The difference between "implies" ("$\Rightarrow$") and "is consequence of" ("$\vdash$") is subtle. Implies is on a syntactic level. $F \Rightarrow G$ just means $\neg F \vee G$. Consequence is on a more concrete level: If F holds then G holds for each valuation/world/... The definition says that $F \vdash G$ means that $\mathcal{A} \models F$ implies $\mathcal{A} \models G$ for all $\mathcal{A}$. Exercise 11 on Problem Sheet 3 may give you another hint on the difference between the two concepts. Moreover the following remark (and its proof) may help.

Lemma 1.8. *The following are equivalent:*

1. $\{F_1, \ldots, F_n\} \vdash G$,
2. $F_1 \wedge \cdots \wedge F_n \Rightarrow G$ *is a tautology,*
3. $F_1 \wedge \cdots \wedge F_n \wedge \neg G$ *is unsatisfiable.*

Proof. It would be tempting to use truth tables. But this does not really work here.

For the equivalence of 1 and 2: Let $\mathcal{A}(F_1) = \cdots = \mathcal{A}(F_n) = 1$. Since G is a consequence of F this implies $\mathcal{A}(G) = 1$. In this case $F_1 \wedge \cdots \wedge F_n \Rightarrow G$ is true.

Let not all $\mathcal{A}(F_i)$ equal one. Then $\mathcal{A}(F_1 \wedge \cdots \wedge F_n) = 0$, hence $F_1 \wedge \cdots \wedge F_n \Rightarrow G$ is true. In both cases $F_1 \wedge \cdots \wedge F_n \Rightarrow G$ is true, hence it is a tautology.

Vice versa: Let $H = F_1 \wedge \cdots \wedge F_n \Rightarrow G$ be a tautology. This means that either not all $\mathcal{A}(F_i)$ equal one (then H is true anyway, the value of $\mathcal{A}(G)$ does not matter for $F \vdash G$), or $\mathcal{A}(F_1) = \cdots = \mathcal{A}(F_n) = 1$ and — since H is true — $\mathcal{A}(G) = 1$. Thus $\{F_1, \ldots, F_n\} \vdash G$.

For the equivalence of 2 and 3 we need to compare whether $F_1 \wedge \cdots \wedge F_n \Rightarrow G$ is a tautology with whether $F_1 \wedge \cdots \wedge F_n \wedge \neg G$ is unsatisfiable.

Let $F_1 \wedge \cdots \wedge F_n \Rightarrow G$ be a tautology. This means that $\neg(F_1 \wedge \cdots \wedge F_n \Rightarrow G)$ is unsatisfiable. Because of

$$\neg(F_1 \wedge \cdots \wedge F_n \Rightarrow G) \equiv \neg(\neg(F_1 \wedge \cdots \wedge F_n) \vee G) \equiv (F_1 \wedge \cdots \wedge F_n) \wedge \neg G$$

the latter term is unsatisfiable, too. $\square$

If we would insist on applying truth tables: we would see that the case $\mathcal{A}(F_1) = \cdots = \mathcal{A}(F_n) = 1$ and $\mathcal{A}(G) = 0$ can not happen (!); in one direction since G is a consequence of $\{F_1, \ldots, F_n\}$, in the other direction since $F_1 \wedge \cdots \wedge F_n \Rightarrow G$ is a tautology.

The notion of consequence plays a central rule in several calculi. Another notation for $\{F_1, \ldots, F_n\} \vdash G$ is

$$\begin{array}{c} F_1 \\ F_2 \\ \vdots \\ F_n \\ \hline G \end{array} \qquad \text{or} \qquad \frac{F_1, F_2, \ldots, F_n}{G}$$

One example of a consequence ist the *modus ponens*. Using the notation above we may write it as $\{F, F \Rightarrow G\} \vdash G$, respectively

$$\frac{F, F \Rightarrow G}{G}$$

This means: whenever F and $F \Rightarrow G$ are true (i.e., whenever $\mathcal{A} \models F$ and $\mathcal{A} \models F \Rightarrow G$) then G is also true under $\mathcal{A}$ (i.e., $\mathcal{A} \models G$). Why is this consequence rule true? We need to show for instance that $F \wedge (F \Rightarrow G) \Rightarrow G$ is a tautology:

$$\begin{aligned} F \wedge (F \Rightarrow G) \Rightarrow G &\equiv \neg(F \wedge (\neg F \vee G)) \vee G \equiv \neg F \vee \neg(\neg F \vee G) \vee G \equiv \neg F \vee (F \wedge \neg G) \vee G \\ &\equiv \big((\neg F \vee F) \wedge (\neg F \vee \neg G)\big) \vee G \equiv (\neg F \vee \neg G) \vee G \equiv \neg F \vee \neg G \vee G \end{aligned}$$

The last expression is obviously a tautology. Hence $F \wedge (F \Rightarrow G) \vdash G$.

1.7 Resolution

A **calculus** is typically a collection of transformation rules, or more general: any algorithmic rule to decide some question on logical formulas (e.g., $F \equiv G$, or F satisfiable). Truth tables are an example of a calculus, but an untypical one: it does not use transformation rules.

Remark 1.6. A calculus is particularly useful if it is **sound** and **complete**. In the context of this section ("Is F unsatisfiable?") this means:

- The calculus is sound, if it does not return "unsatisfiable" for any satisfiable F.
- The calculus is complete, if it returns "unsatisfiable" for any unsatisfiable F.

This is a subtle point: a calculus that is both sound and complete for "Is F unsatisfiable?" might not be complete for the question "Is F satisfiable?" (it may return nothing, resp. run forever, for some satisfiable F).

The resolution calculus uses just a single transformation rule to decide whether a formula F is satisfiable or not. (Later we will use the resolution calculus to decide whether a given formula in first order logic is *un*satisfiable.)

Remark 1.7. A test for unsatisfiability of F answers several further questions, e.g.

- Is F a tautology? (if and only if $\neg F$ is unsatisfiable)
- Is G a consequence of $F_1, \dots, F_n$? (if and only if $F_1 \wedge \cdots \wedge F_n \wedge \neg G$ is unsatisfiable)

The first point is obvious: $\mathcal{A}(F) = 1$ for all $\mathcal{A}$ implies $\mathcal{A}(\neg F) = 0$ for all $\mathcal{A}$. The second point is Lemma 1.8.

In order to describe the resolution calculus for formulas in propositional logic we need some notation. Given some formula F in CNF:

$$F = (L_{1,1} \vee L_{1,2} \vee \cdots \vee L_{1,n_1}) \wedge (L_{2,1} \vee L_{2,2} \vee \cdots \vee L_{2,n_2}) \wedge \cdots \wedge (L_{k,1} \vee L_{k,2} \vee \cdots \vee L_{k,n_k})$$

where the L_{ij} are literals, we write F in **clause set notation** as

$$F = \big\{ \{L_{1,1}, L_{1,2}, \dots, L_{1,n_1}\}, \{L_{2,1}, L_{2,2}, \cdots, L_{2,n_2}\}, \cdots, \{L_{k,1}, L_{k,2}, \cdots, L_{k,n_k}\} \big\}.$$

Different formulas can have the same form in this notation. In fact the set notation reduces ambiguities. For instance the formulas

$$(A_1 \vee \neg A_2) \wedge A_3 \wedge A_3, \quad A_3 \wedge (\neg A_2 \vee A_1), \quad \text{and } A_3 \wedge (\neg A_2 \vee A_1 \vee \neg A_2)$$

are all represented in set notation by $\big\{ \{A_3\}, \{A_1, \neg A_2\} \big\}$.

Definition 1.7. Let K_1, K_2 be clauses, L a literal such that $L \in K_1$, $\neg L \in K_2$. Then the **resolvent** (of K_1 and K_2) is

$$R = (K_1 \setminus \{L\}) \cup (K_2 \setminus \{\neg L\})$$

It may happen that $R = \{\}$ (if $K_1 = \{L\}$ and $K_2 = \{\neg L\}$). Since this means "unsatisfiable" in the sequel we write $R = \Box$ in this case (rather than $R = \{\}$ or $R = \varnothing$, which could be confused with the empty clause).

In a diagram we will write a resolvent R of K_1 and K_2 as

$$\begin{tikzcd}[column sep=small] K_1 \arrow[dr, no head] & & K_2 \arrow[dl, no head] \\ & R & \end{tikzcd}$$

Example 1.8.
$$\begin{tikzcd}[column sep=tiny]
\{A_1,A_3,\neg A_4\} \arrow[dr, no head] & & \{\neg A_1,A_4\} \arrow[dl, no head] \\
& \{A_1,\neg A_1,A_3\} &
\end{tikzcd}$$
(using $L = A_4$) or
$$\begin{tikzcd}[column sep=tiny]
\{A_1,A_3,\neg A_4\} \arrow[dr, no head] & & \{\neg A_1,A_4\} \arrow[dl, no head] \\
& \{A_3,A_4,\neg A_4\} &
\end{tikzcd}$$
(using $L = A_1$).

Lemma 1.9. *Let F be a formula in clause set notation, let R be a resolvent of two clauses in F. Then $F \equiv F \cup \{R\}$.*
In the language of consequences: the resolution $F \vee G$ is a consequence of $F \vee L$ and $G \vee \neg L$. In short: $\{F \vee L, G \vee \neg L\} \vdash F \vee G$.

Hence the general idea of the resolution calculus is to determine all possible resolvents iteratively. F is unsatisfiable if and only if $\square$ appears at some point.

Notation: Let F be a formula in clause set notation.

- $\mathrm{Res}(F) = F \cup \{R \mid R \text{ resolvent of two clauses in } F\}$
- $\mathrm{Res}^0(F) = F$
- $\mathrm{Res}^{n+1}(F) = \mathrm{Res}(\mathrm{Res}^n(F))$
- $\mathrm{Res}^*(F) = \bigcup_{n \geq 0} \mathrm{Res}^n(F)$

Even though in general the questions “Is F unsatisfiable?” and “Is F satisfiable?” might be of different nature, in propositional logic they are the same with respect to resolution:

Theorem 1.10. *F is unsatisfiable if and only if $\square \in \mathrm{Res}^*(F)$. Since the procedure always terminates (see below) it follows that the resolution calculus is sound and complete.*

The corresponding algorithm is clear now: compute iteratively $\mathrm{Res}^*(F)$. Stop if $\mathrm{Res}^k(F) = \mathrm{Res}^{k+1}(F)$ for some k (then $\mathrm{Res}^k(F) = \mathrm{Res}^*(F)$, see Exercise 18, Sheet 5). Return “F is unsatisfiable” if $\square \in \mathrm{Res}^*(F)$, return “$F$ is satisfiable” else.

The algorithm always terminates since we are dealing with finite formulas only. (In the next section we need to apply this algorithm to infinitely many formulas, hence it might not terminate if F is satisfiable.) We describe the algorithm by an example:

Example 1.9. Given $F = (A \vee B \vee \neg C) \wedge (A \vee B \vee C) \wedge (A \vee \neg B) \wedge \neg A$. We write F in clause set notation:
$$F = \big\{\{A,B,\neg C\},\{A,B,C\},\{A,\neg B\},\{\neg A\}\big\}.$$

$$\begin{tikzcd}[column sep=small]
\{A,B,\neg C\} \arrow[dr, no head] & & \{A,B,C\} \arrow[dl, no head] & \{A,\neg B\} \arrow[ddl, no head] & \{\neg A\} \arrow[dddl, no head] \\
& \{A,B\} \arrow[dr, no head] & & & \\
& & \{A\} \arrow[dr, no head] & & \\
& & & \square &
\end{tikzcd}$$

Hence F is unsatisfiable.

In general, the diagram may not be a tree: sometimes one needs to use the same clause more than once, hence creating some loop.

The resolutions used in the example were chosen in some optimal way. The list of all resolvents is:

$$\text{Res}^1(F) = F \cup \big\{\{A,B\},\{A,C\},\{A,\neg C\},\{B,\neg C\},\{B,C\},\{\neg B\}\big\}$$

$$\text{Res}^2(F) = \text{Res}^1(F) \cup \big\{\{A\},\{B\},\{C\},\{\neg C\}\big\}$$

A lot of work has been done on improving the algorithm by choosing resolvent in some optimal way. 13. Nov.
Nevertheless:

Theorem 1.11 (Haken 1984)**.** *There are formulas where each derivation of $\square$ needs exponentially many steps in n, where n is the number of atomic formulas.*

Moreover, the CNF of some given formula F can be exponentially longer than F, see remark 1.4.

Remark 1.8. The resolution calculus is efficient for Horn formulas: consider only those resolvents using clauses K_1 and K_2 where either K_1 has only one element, or K_2 has only one element. This can be seen by comparing the algorithms: This version of resolution simulates the Horn formula algorithm 1.3: Using clauses with one element covers all clauses in the Horn formula of the form $1 \Rightarrow A_i$. Marking means setting $\mathcal{A}(A_i) = 1$, hence implicitly adding a clause $\{A_i\}$. Marking all copies of A_i in $(A_i \wedge A_j \wedge \cdots \wedge A_m \Rightarrow B)$ means marking A_i in $(\neg A_i \vee \neg A_j \vee \cdots \vee \neg A_m \vee B)$ This corresponds to the clause $\{\neg A_i, \neg A_j, \cdots, \neg A_m, B\}$. The resolvent of the latter clause with $\{A_i\}$ is $\{\neg A_j, \cdots, \neg A_m, B\}$, and so on.

The resolution calculus is also efficient for formulas (in CNF) where each clause has at most two elements (see Exercise 18, Sheet 4).

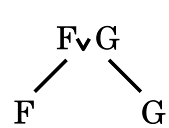
Of course some classes of formulas are even simpler to solve: for instance, if each clause contains at least one negative literal, then F is satisfiable, compare Exercise 10. The same is true for "at least one positive literal".

There are several implementations of the resolution algorithm and of its refinements (Davis-Putnam-algorithm, Davis-Putnam-Logemann-Loveland algorithm, SAT-solvers...)

1.8 Tableau calculus

For later purposes we need a further formalism deciding the satisfiability of some formula in propositional logic. It is assumed that the reader is familiar with the notion of a **binary tree** (root, node, child, leave, path...) In the sequel "path" means a path ending in a leave.

The idea is to start constructing a tree of all possibilities how a formula F may be true. The root will be F. A formula of the form $G \vee H$ leads to a branch



meaning that either F or G (or both) must be true. A formula of the form $G \wedge H$ leads to a branch

$$\begin{array}{c} F\wedge G \\ | \\ F \\ | \\ G \end{array}$$

meaning that both F or G must be true. A formula is satisfiable if there is a branch that is satisfiable. The general method goes as follows.

Algorithm 1.4. . Input: some formula F in propositional logic.

1. Start with F as the root.
2. Choose a node G that is not a literal. Mark G. Apply the following rules until all possibilities are exhausted.
   - If G is of the form $\neg\neg H$ then add a single node H to each path starting in G.
   - If G is of the form $H_1 \vee H_2$ then add to each path starting in G this:

     $$\begin{array}{cc} / & \backslash \\ H_2 & H_1 \end{array}$$

   - If G is of the form $H_1 \wedge H_2$ then add to each path starting in G this:

     $$\begin{array}{c} | \\ H_1 \\ | \\ H_2 \end{array}$$

   - If G is of the form $\neg(H_1 \vee H_2)$ then add to each path starting in G this:

     $$\begin{array}{c} | \\ \neg H_1 \\ | \\ \neg H_2 \end{array}$$

   - If G is of the form $\neg(H_1 \wedge H_2)$ then add to each path starting in G this:

     $$\begin{array}{cc} / & \backslash \\ \neg H_2 & \neg H_1 \end{array}$$

3. If there is no further node to mark then return the tableau, STOP.

A path is **closed** if it contains A_i and $\neg A_i$ for some i. In this case we mark the leave of this path by $\otimes$. A tableau is **closed** if all leaves are marked by $\otimes$. In this case F is unsatisfiable. Else F is satisfiable, and each path with unmarked leave yields a satisfying valuation $\mathcal{A}$: for each occurrence of A_i set $\mathcal{A}(A_i) = 1$, for each occurrence of $\neg A_i$ set $\mathcal{A}(A_i) = 0$.

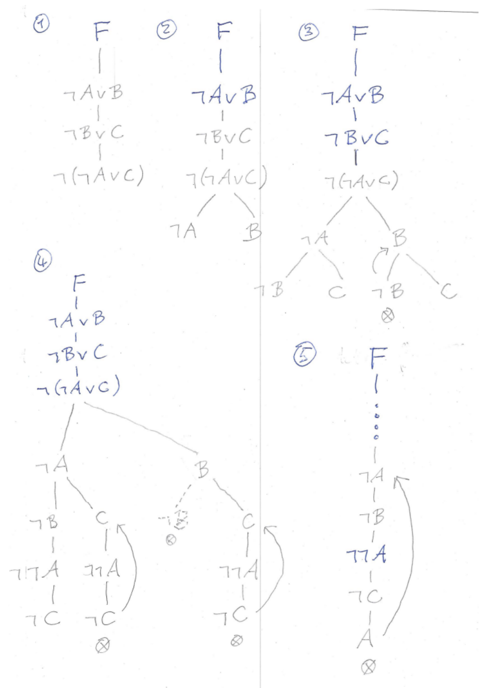


Figure 1: An example of the Tableau algorithm. For each new marking (in blue) there is a new diagram ((1)-(5)).

There are some obvious simplifications. For instance, a formula of the form $A \vee B \vee C$ can be realized in one step with three branches rather than two (the tree is not longer binary in this case).

There are also several refinements of this algorithm in order to improve the runtime. For instance, pathes can be marked closed as soon as they are detected to be closed.

Theorem 1.12. *The Tableau algorithm of propositional logic is sound and complete.*

Example 1.10. We show that $A \Rightarrow B$ and $B \Rightarrow C$ implies $A \Rightarrow C$. This means (see Remark 1.7) that

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

is unsatisfiable. The resulting tableau is shown in Figure 1.

Interlude: Relations

In the next section predicates will play a central role. Predicates are a generalisation of relations. (Predicates can have one, two, three... inputs, whereas relations have always two inputs.)

A relation is explained on some set W. Think of W as all students in the Technical Department, or all integers, or all 0-1-words. In plain words a relation is some property that any pair of elements in W may have or not have. For instance, on $W = \mathbb{Z}$ a relation may be $<$: $2 < 5$ is true, so the relation is true for the pair $(2,5)$. The relation is not true for the pair $(5,2)$, or the pair $(3,3)$.

Another relation on $W = \mathbb{Z}$ is "$a-b$ is odd". So the relation is true for $(2,5)$ and $(5,2)$, but not for $(3,3)$

Recall that for two sets V,W the **Cartesian product** is

$$V \times W := \{(a,b) \mid a \in V, b \in W\}.$$

A convenient way to define a relation is to view it as a subset R of $W \times W$. For the $<$-example we obtain

$$R = \{(a,b) \mid a,b \in \mathbb{Z}, a < b\}$$

So $(2,5) \in R$, $(5,2) \notin R$, $(3,3) \notin R$. For the "$a-b$ is odd"-example we obtain

$$R' = \{(a,b) \mid a-b \text{ is odd.}\}$$

So $(2,5) \in R$, $(5,2) \in R$, $(3,3) \notin R$. A particular nice class of relations are equivalence relations.

Definition 1.8. Let R be a relation. $R \subseteq W \times W$ is called an **equivalence relation**, if for all $a,b,c \in W$ holds:

1. $(a,a) \in R$, *(reflexive)*
2. $(a,b) \in R$ if and only if $(b,a) \in R$, and *(symmetric)*
3. $(a,b) \in R$ and $(b,c) \in R$ implies $(a,c) \in R$. *(transitive)*

An equivalence relation partitions the set W into **equivalence classes**: By $[a]$ we denote the set of all $b \in W$ such that $(a,b) \in R$. A simple example of an equivalence relation is $=$ in $\mathbb{Z}$. Here each equivalence class $[a]$ consists of one element only, namely, a.

2 First-order logic

In propositional logic we cannot split statements like "x is a philosopher" or "for all $n \in \mathbb{N}$ holds: $n \geq 0$" into smaller statements (thus "atomic" formula). Extending the language of logic we introduce now **variables** (like x). A statement "is a philosopher" then is a **predicate**, its value being 0 or 1, depending on the value of X. Furthermore adding **quantifiers** $\forall$ and $\exists$ and **functions** to the language of propositional logic yields the language of first-order logic. This allows for the formal logical treatment of statements like

$$\forall \varepsilon\, \exists \delta\, \forall x\, (|x-a| < \delta \Rightarrow |f(x) - f(a)| < \varepsilon)$$

Here $\forall$ and $\exists$ are quantifiers, f and $|\cdot|$ and $-$ are functions, ε, δ, x and a are variables, and $<$ is a predicate.

Like in Section 1 the general structure of this section is 1. formal definition of syntax and semantics, 2. normal forms, 3. algorithms to decide (un-)satisfiability of formulas. Unlike in Section 1 we will see here some problems that are undecidable.

2.1 Syntax and semantics of first-order logic

As in Section 1 we first declare what are valid formulas in a purely abstract way (syntax). Later we see how to fill this formulas with concrete meaning (semantics).

Definition 2.1 (syntax of first-order logic)**.** The building blocks of formulas are

- **Variables** denoted by $x_1, x_2, \dots$ or u, v, w, x, y, z.
- **Predicate**-symbols denoted by $P_1, P_2, \dots$ or P, Q, R.
- **Function**-symbols denoted by $f_1, f_2, \dots$ or f, g, h.
- **Terms** are denoted $t_1, t_2, \dots$ and are defined inductively:
  - Each variable is a term.
  - If f is a function (with k inputs) and $t_1, \dots, t_k$ are terms, then $f(t_1, \dots, t_k)$ is a term.
- **Formulas** are defined inductively:
  - If P is a predicate (with k inputs) and $t_1, \dots, t_k$ are terms, then $P(t_1, \dots, t_k)$ is a formula. (Sometimes these are called *atomic* formulas)
  - If F is a formula, then $\neg F$ is.
  - If F and G are formula, then $F \vee G$ and $F \wedge G$ are.
  - If x is a variable and F is a formula then $\forall x F$ and $\exists x F$ are formulas.

All intermediate steps of this inductive construction are called **partial formulas**. An occurrence of some variable x in some formula F is **bound** if F contains a partial formula $\forall x G$ or $\exists x G$ such that G contains x; otherwise this occurrence of x is called **free**. Note that a variable can be both bound and free in the same formula, see the example below. A formula is **closed** if all occurrences of all variables are bound. The **matrix** of F is obtained by deleting all $\forall x$ and $\exists x$. Functions with no inputs are allowed. Such functions are also called **constants**.

Remark 2.1. Because "being equal" is such a natural and frequently occurring concept, it is sometimes useful to allow a further symbol: $=$. The definition of the syntax has to be adjusted appropriately: If t_1 and t_2 are terms, then $t_1 = t_2$ is a formula.

Example 2.1. $F = \exists x P(x, f_1(y)) \vee \neg \forall y Q(y, f_7(f_2, f_3(z)))$ is a formula. All partial formulas of F are

$$F,\ \exists x P(x, f_1(y)),\ P(x, f_1(y)),\ \neg \forall y Q(y, f_7(f_2, f_3(z))),\ \forall y Q(y, f_7(f_2, f_3(z))),\ Q(y, f_7(f_2, f_3(z))).$$

All terms in F are $x, y, f_1(y), f_7((f_2, f_3(z)), f_2, f_3(z), z$. All occurrences of x in F are bound. The first occurrence of y in F is free, the second is bound. The occurrence of z in F is free. The matrix of F is

$$P(x, f_1(y)) \vee \neg Q(y, f_7(f_2, f_3(z))).$$

Remark 2.2. In order to use less brackets we agree on the rule that $\forall x$ and $\exists x$ binds stronger than $\vee$ and $\wedge$. Hence $\forall x\, P(x) \vee \forall y\, P(y)$ means $(\forall x\, P(x)) \vee (\forall y\, P(y))$, and not $\forall x\, (P(x) \vee \forall y\, P(y))$.

In order to interpret formulas in first-order logic as "true" or "false" one needs to give the symbols a meaning. More precisely, one needs to define a basic set in which the variables and functions take their values (the universe), and an interpretation of any predicate-symbol as a predicate (relation) in this universe, and so on. More precisely: Nov 20

Definition 2.2. A **structure** (aka world) for a formula F is a pair $\mathcal{A} = (U_{\mathcal{A}}, I_{\mathcal{A}})$, where

- $U_{\mathcal{A}}$ is a nonempty set (the **universe**), and
- $I_{\mathcal{A}}$ is a map (the **interpretation**) such that
  - $I_{\mathcal{A}}(x)$ is an element of $U_{\mathcal{A}}$ for all free variable-symbols x in F,
  - $I_{\mathcal{A}}(f)$ is a function over $U_{\mathcal{A}}$ for all function-symbols f in F,
  - $I_{\mathcal{A}}(P)$ is a predicate over $U_{\mathcal{A}}$ for all predicate-symbols P in F.

Here "element" has the usual meaning: some element of $U_{\mathcal{A}}$. "Function" means a function from $(U_{\mathcal{A}})^k \to U_{\mathcal{A}}$, where f has k inputs. "Predicate" is to be thought of as some property. E.g., for a predicate P with one input $P(x)$ may mean "x is odd", or "x is a prime number" (if $U_{\mathcal{A}} = \mathbb{N}$), or $P(x)$ may mean "x is a philosopher" if $U_{\mathcal{A}}$ is the set of all humans. For a predicate P with two inputs $P(x,y)$ can mean $x < y$, or x divides y (if $U_{\mathcal{A}} = \mathbb{N}$), or $P(x,y)$ can mean "x knows y" if $U_{\mathcal{A}}$ is the set of all humans.

Remark 2.3. . More formally a predicate P is defined as a subset of $U_{\mathcal{A}}$ (if P has one input), or a subset of $U_{\mathcal{A}} \times U_{\mathcal{A}}$ (if P has two inputs), or a subset of $U_{\mathcal{A}} \times U_{\mathcal{A}} \times \cdots \times U_{\mathcal{A}}$ (n times) if P has n inputs. Then P is for instance

$$\{n \in \mathbb{N} \mid x \text{ is odd }\}, \quad \{n \in \mathbb{N} \mid x \text{ is prime }\},$$

respectively

$$\{(n,m) \in \mathbb{N} \times \mathbb{N} \mid n < m\}, \quad \{(n,m) \in \mathbb{N} \times \mathbb{N} \mid n \text{ divides } m\}$$

for the examples mentioned above.

In the sequel we will write shortly $x^{\mathcal{A}}$, $f^{\mathcal{A}}$, $P^{\mathcal{A}}$ rather than $I_{\mathcal{A}}(x)$, $I_{\mathcal{A}}(f)$, $I_{\mathcal{A}}(P)$.

Example 2.2. Consider the formula $F' = (\forall x P(x, f(x))) \wedge Q((g(h,x))$. Here x is both a bound variable and a free variable, f is a function with one input, g is a function with two inputs, and h is a function with no input; hence h is a constant. P is a predicate with two inputs, Q is a predicate with one input.

In order to avoid confusion let us consider $F = (\forall x P(x, f(x))) \wedge Q(g(h,z))$. A possible structure for F is

$$\begin{aligned}
U_{\mathcal{A}} &= \{0,1,2,\ldots\} = \mathbb{N}_0 \\
P^{\mathcal{A}} &= \{(n,m) \in \mathbb{N}_0 \times \mathbb{N}_0 \mid n < m\} \\
Q^{\mathcal{A}} &= \{n \in \mathbb{N}_0 \mid x \text{ is prime }\} \\
f^{\mathcal{A}} &: \mathbb{N}_0 \to \mathbb{N}_0,\ f(n) = n+1 \\
g^{\mathcal{A}} &: \mathbb{N}_0 \times \mathbb{N}_0 \to \mathbb{N}_0,\ g(n,m) = n+m \\
h^{\mathcal{A}} &= 2 \quad \text{and} \quad z^{\mathcal{A}} = 3.
\end{aligned}$$

This structure does not only match F, but it makes F true! Indeed, in this universe F means

$$(\forall x \in \mathbb{N}\, x < x+1) \wedge 2+3 \text{ is prime.}$$

Since both statements are true, F is true in the structure $\mathcal{A}$. We will write shortly $\mathcal{A} \models F$.

If we change the structure above into $\mathcal{A}'$ by changing $z^{\mathcal{A}} = 3$ to $z^{\mathcal{A}'} = 4$ then F is false in the new structure $\mathcal{A}'$ (since $2+4=6$ is not prime).

In the sequel we will focus on the question "Given some formula F, is there some structure making F true?" (satisfiability); respectively "Given some formula F, is F true for all structures (matching F)?" (compare tautology in Section 1, here this will be denoted as "valid") In the spirit of Def. 1.2 we will define what it means that $\mathcal{A}$ makes F true.

Definition 2.3. Let F be a formula and $\mathcal{A}$ be a structure for F. $\mathcal{A}(F)$ is defined inductively by Definition 1.2 together with the following points

1. If $F = P(t_1, \ldots, t_k)$ then
$$\mathcal{A}(F) = \begin{cases} 1 & \text{if } (t_1^{\mathcal{A}}, \ldots, t_k^{\mathcal{A}}) \in P^{\mathcal{A}}, \\ 0 & \text{else} \end{cases}$$
2. If $F = \forall x G$ then
$$\mathcal{A}(F) = \begin{cases} 1 & \text{if for all } x^{\mathcal{A}} \in U_{\mathcal{A}} \text{ holds } \mathcal{A}(G) = 1 \\ 0 & \text{else} \end{cases}$$
3. If $F = \exists x G$ then
$$\mathcal{A}(F) = \begin{cases} 1 & \text{if there is } x^{\mathcal{A}} \in U_{\mathcal{A}} \text{ holds } \mathcal{A}(G) = 1 \\ 0 & \text{else} \end{cases}$$

Given a formula F and a structure $\mathcal{A}$ for F.

- If $\mathcal{A}(F) = 1$ then $\mathcal{A}$ **satisfies** F, short: $\mathcal{A} \models F$. (also $\mathcal{A}$ is **model** for F)
- If for all $\mathcal{A}$ for F holds $\mathcal{A} \models F$ then F is **valid** (this is the analogue concept of tautology in propositional logic).
- If there is $\mathcal{A}$ for F such that $\mathcal{A} \models F$ then F is **satisfiable**.

Example 2.3. Let $F = \forall x \exists y\, P(x,y)$. A structure $\mathcal{A}$ with $\mathcal{A} \models F$ is

$$U_{\mathcal{A}} = \mathbb{N}, \quad P^{\mathcal{A}} = \{(x,y) \mid x < y, x \in \mathbb{N}, y \in \mathbb{N}\}.$$

("For all $x \in \mathbb{N}$ exists $y \in \mathbb{N}$ such that $x < y$", bingo)

A structure $\mathcal{A}$ with $\mathcal{A} \not\models F$ is for instance

$$U_{\mathcal{A}} = \{0,1,2,\ldots,9\} \quad P^{\mathcal{A}} = \{(x,y) \mid x = 2y, x, y \in U_{\mathcal{A}}\}.$$

In this case F is false for $x = 1$.

In order to illustrate that a structure is not necessarily a mathematical object: Let $U_{\mathcal{A}}$ be the set of all humans, and let $P^{\mathcal{A}}(x,y)$ mean "x loves y". Consider all combinations of quantifier-variable-quantifier-variable $P(x,y)$. Then

- $\forall x\, \exists y\, P(x,y)$ is “everybody loves someone”,
- $\forall y\, \exists x\, P(x,y)$ is “everybody is loved by someone”,
- $\exists x\, \forall y\, P(x,y)$ is “there is someone loving everybody” (Jesus?),
- $\exists y\, \forall x\, P(x,y)$ is “there is someone loved by everybody” (Elvis??),
- $\forall x\, \forall y\, P(x,y)$ is “everybody loves everybody” (Woodstock???),
- $\exists x\, \exists y\, P(x,y)$ is “somebody loves someone”.
- $\exists x\, P(x,x)$ is “somebody loves himself”.

The truth of each formula depends on $\mathcal{A}$, but all these formulas are satisfiable (in a universe where everybody loves everybody), and probably none of these formulas is valid (= true in all possible universes).

Remark 2.4. Again, for first-order logic with identity, the definition of the semantics needs to be adjusted. This is done by giving the predicate $=$ the usual meaning: being equal as elements in $U_{\mathcal{A}}$.

Remark 2.5. In analogy to Remark 1.7 one can show here that

- F is valid if and only if $\neg F$ is unsatisfiable.
- G is a consequence of $F_1 \wedge \cdots \wedge F_n$ if and only if $F_1 \wedge \cdots \wedge F_n \neg G$ is unsatisfiable.

Remark 2.6. First-order logic does indeed contain propositional logic as a special case. In fact each formula in first order logic without quantifiers can be interpreted as a formula in propositional logic: all variables are free, hence treated as constants. All predicates contain only constants. Hence variables as such vanish. For instance,

$$F = (P(a) \vee \neg Q(f(b),c)) \wedge R(a,b)$$

corresponds in each interpretation to constants and predicates of constants. The latter are the atomic formulas, so F becomes

$$(A \vee \neg B) \wedge C.$$

The latter is satisfiable (resp. a tautology) if and only if the former is satisfiable (resp. valid).

Remark 2.7. Not every statement can be expressed in first-order logic. Up to now we need to deal with one universe: we cannot express “for all students there is an integer” (at least not easily). Moreover, more we cannot express statements like “for all functions”, or “it exists a predicate”. Allowing for these possibilities yields *second-order logic*. The latter is beyond the scope of this course.

2.2 Normal forms

27. Nov. All laws of calculation in propositional logic (see Theorem 1.1) do also apply to first-order logic. We need some further laws for treating quantifiers. These laws tell us when two formulas are equivalent. Being equivalent here means essentially the same as in Section 1: $F \equiv G$ means that for all $\mathcal{A}$ holds that $\mathcal{A}(F) = \mathcal{A}(G)$. The definition is the same only the meaning of $\mathcal{A}$ has changed.

Theorem 2.1. *Let F and G be formulas. Then the following rules hold.*

1. $\neg\forall x\, F \equiv \exists x\, \neg F$
   $\neg\exists x\, F \equiv \forall x\, \neg F$
2. *If x does not occur freely in G, then*
   $(\forall x\, F) \wedge G \equiv \forall x\, (F \wedge G)$
   $(\forall x\, F) \vee G \equiv \forall x\, (F \vee G)$
   $(\exists x\, F) \wedge G \equiv \exists x\, (F \wedge G)$
   $(\exists x\, F) \vee G \equiv \exists x\, (F \vee G)$
3. $(\forall x\, F) \wedge (\forall x\, G) \equiv \forall x\, (F \wedge G)$
   $(\exists x\, F) \vee (\exists x\, G) \equiv \exists x\, (F \vee G)$
4. $\forall x\, \forall y\, F \equiv \forall y\, \forall x F$
   $\exists x\, \exists y\, F \equiv \exists y\, \exists x F$

Proof. The proofs are standard, technical and do not yield further insight. So we show only the proof for the first formula. We need the notation $\mathcal{A}_{[x/u]}(F)$ which stands for "replace each x in F by u". Using Definition 2.3 we get:

$$\begin{aligned}
\mathcal{A}((\forall x\, F) \wedge G) = 1 &\Leftrightarrow \mathcal{A}(\forall x\, F) = 1 \text{ and } \mathcal{A}(G) = 1 \\
&\Leftrightarrow (\text{ for all } x \in U_{\mathcal{A}} :\ \mathcal{A}(F) = 1) \text{ and } \mathcal{A}(G) = 1 \\
&\Leftrightarrow \text{ for all } x \in U_{\mathcal{A}} :\ (\mathcal{A}(F) = 1 \text{ and } \mathcal{A}(G) = 1) \\
&\qquad \text{(since u does not appear in } G \text{ freely)} \\
&\Leftrightarrow \text{ for all } x \in U_{\mathcal{A}} :\ \mathcal{A}(F \wedge G) = 1 \\
&\Leftrightarrow \mathcal{A}(\forall (F \wedge G)) = 1.
\end{aligned}$$

All rules above can be proved in this fashion. □

In order to construct normal forms we will use the rules above to push quantifiers "to the front".

Example 2.4.

$$\begin{aligned}
\neg(\exists x\, P(x,y) \vee \forall z\, Q(y)) \wedge \exists w\, Q(w) =& \neg\exists x\, P(x,y) \wedge \neg\forall z\, Q(y) \wedge \exists w\, Q(w) \\
=& \forall x\, \neg P(x,y) \wedge \exists z\, \neg Q(y) \wedge \exists w\, Q(w) \\
=& \forall x\, (\neg P(x,y) \wedge \exists z\, \neg Q(y)) \wedge \exists w\, Q(w) \\
=& \forall x\, (\exists z(\neg P(x,y) \wedge\ \neg Q(y))) \wedge \exists w\, Q(w) \\
=& \exists w\, \forall x\, \exists z\, (\neg P(x,y) \wedge \neg Q(y) \wedge Q(w))
\end{aligned}$$

Note that the order of the quantifiers above depends on the transformations used. In this case the quantifiers can be arranged to any order, but this is not always the case.

A problem occurs if we want to use rule 2: $\forall x\, F \wedge G \equiv \forall x\, (F \wedge G)$ and if a variable x is free in G, but bound in F.

Lemma 2.2. *Let $F[x/y]$ denote the formula obtained by replacing each occurrence of x in F by y.*

Let G be some formula not containing y. Then

$$\forall x\, G \equiv \forall y\, G[x/y] \quad \textit{and} \quad \exists x\, G \equiv \exists y\, G[x/y]$$

Using this lemma we may transform each formula into one where no two quantifiers have the same variable, and where no variable occurs both bound and free. This is the requirement in order to construct the following normal form.

Definition 2.4. A formula F has **prenex normal form** (PNF) if

$$F = Q_1 x_1 Q_2 x_2 \cdots Q_n x_n G,$$

where the Q_i are quantifiers, and G contains no further quantifiers.

In other words, all variables in G (!) are free, respectively G is the matrix of F. For instance

$$F = \exists x \, \forall y \, \exists z \, ((P(f(x)) \wedge P(y)) \vee Q(z))$$

has PNF, whereas

$$F = \exists x \, \exists y \, (P(f(x)) \wedge P(y)) \vee \forall z \, Q(z)$$

has not.

Theorem 2.3. *Each formula in first-order logic has an equivalent formula in PNF.*

Again, this theorem is proven using the inductive definition of a formula. For the algorithmic treatment of (satisfiability of) formulas we need a more special normal form. Maybe the exercises gave an idea already on the fact that it makes no big difference whether a variable is free, or is bound by an $\exists$. So the next step is to delete all $\exists$ in some appropriate manner.

Definition 2.5. Given a formula in PNF its corresponding **Skolem normal form** (SNF) is the output of the following algorithm:

1. while F contains $\exists$ do
   - Is $F = \forall y_1 \, \forall y_2 \, \cdots \forall y_n \, \exists x \, G$
     then choose a function f not contained in G and let $F := \forall y_1 \, \forall y_2 \, \cdots \forall y_n \;\, G[x/f(y_1, \ldots, y_n)]$.
2. STOP

The case $n = 0$ is possible, that is, F is of the form $\exists x G$.

Theorem 2.4. *For each formula F in first-order logic there is a formula G in SNF, such that F is satisfiable if and only if G is.*

The following list summarizes the steps needed to transform some arbitrary formula into the normal form we need later.

Algorithm 2.1 (Skolem normal form).

0. Replace all partial formulas $F \Rightarrow G$ by $\neg F \vee G$, and all $F \Leftrightarrow G$ by $(F \wedge G) \vee (\neg F \wedge \neg G)$.
1. Rename bound variables in F until no two quantifiers have the same variable, and where no variable occurs both bound and free.
2. Let $y_1, \ldots, y_n$ be all free variables in F. Replace each y_i by a constant a_i.
3. Transform F into PNF (by pushing quantifiers to the front).
4. Delete all $\exists$ by transforming F into Skolem normal form.
5. Transform the matrix of F into CNF.

Note that only steps 0, 1, 3 and 5 preserve equivalence of formulas. In general the output will not be some equivalent formula but only an equisatisfiable one.

Remark 2.8. Obviously, constants play a particular role here. Up to now we used function symbols $f,g,h...$ for constants (where $f,g,h...$ had no inputs). For the sake of clarity, from now on we will use $a,b,c...$ or $a_1,a_2,...$ for constants.

2.3 Resolution calculus of first-order logic

We will see in Section 3 that we cannot hope for anything better than a test for unsatisfiability of a formula F that returns "Yes" whenever F is unsatisfiable, and may return nothing (runs forever) if F is satisfiable. Moreover, we may wait for a positive answer arbitrarily long: if there would be some bound on the waiting time this test could be turned into some test on satisfiability ("wait long enough, if algorithm did not terminate, return 'satisfiable"').

The problem is essentially that we need to test infinitely many structures (compare Theorem 3.2 in Section 3). The solution is to use a certain standard structure.

Definition 2.6. The **Herbrand universe** $H(F)$ of some formula F is the set of all variable-free terms that can be constructed out of partial formulas of F. That is, $H(F)$ consists of

1. All constants $a,b,\dots$ occurring in F are in $H(F)$.
2. If $t_1,t_2,\dots t_n \in H(F)$ and f occurs in F then $f(t_1,t_2,\dots,t_n) \in H(F)$.

If F contains no constant then add one constant a to $H(F)$ and proceed as above.

Note the twist here: the elements of the universe *are* the symbols in F. This means we define the semantics to be the syntax. Consequently, any two different symbols are different elements of $H(F)$.

Example 2.5. Let $F = \forall x\, \forall y\, \forall z\, P(x, f(y), g(z,x))$. F contains no constant, so we add a to $H(F)$ and get

$$H(F) = \{a, f(a), g(a,a), f(f(a)), f(g(a,a)), g(a, f(a)), g(f(a),a), g(f(a), f(a)), f(g(a, f(a))), \dots\}$$

Let $G = \forall x\, \forall y\, Q(a, f(z), h(y,b))$. G contains two constants a,b, and we get 4. Dec.

$$H(G) = \{a, b, f(a), f(b), h(a,b), f(f(a)), f(f(b)), f(h(a,b)), h(f(a),b), h(h(a,b),b), \dots\}$$

Note in tha latter example the different treatment of variables (x,y) and constants (a,b). In order to obtain a structure $\mathcal{A}$ we need an interpretation. All terms are already defined (e.g. $a^{\mathcal{A}} = a, f^{\mathcal{A}}(a) = f(a)$ and so on). In principle we still need to define interpretation of the predicates. But in the sequel we proceed by considering all possible interpretations of some formula F. Explicitly this means that we will iterate over all instances of (the matrix of) F — for instance

$$P^{\mathcal{A}}(a, f(a), g(a,a)), P^{\mathcal{A}}(b, f(a), g(a,a)), P^{\mathcal{A}}(a, f(b), g(a,a)), P^{\mathcal{A}}(a, f(a), g(b,a)), \dots$$

in the example above — and consider all these instances as atomic formulas, until we obtain a contradiction. Any structure $\mathcal{A} = (H(F), I_{\mathcal{A}})$ with $\mathcal{A} \models F$ is called a **Herbrand model** for F.

Theorem 2.5. *Let F be in Skolem normal form. Then F is satisfiable if and only if F has a Herbrand model.*

A proof is contained in Schöning.

Definition 2.7. Let $F = \forall y_1 \cdots \forall y_n F^*$ be a formula in Skolem normal form, and let F^* be the matrix of F. Let $E(F)$ be the set of all instances of F, that is:

$$E(F) = \{F^*[y_1/t_1][y_2/t_2]\cdots[y_n/t_n] \mid t_1, t_2, \ldots, t_n \in H(F)\}.$$

$E(F)$ is called **Herbrand expansion**.

Theorem 2.6. *For each formula F in Skolem normal form holds: F is satisfiable if and only if $E(F)$ is satisfiable (in the sense of propositional logic).*

Note that $E(F)$ is infinite, so this is the point where the Compactness Theorem (Thm. 1.6) is needed. Now we are able to provide a test that answers whether a given formula is unsatisfiable.

Algorithm 2.2. Input: a formula F in Skolem normal form such that the matrix of F has CNF. Let $E(F) = \{F_1, F_2, F_3, \ldots\}$ be an enumeration of the Herbrand expansion of F. Let $n = 0$, $M := \{\}$.

while $\square \notin M$ do

- $n := n + 1$
- $M := M \cup \{F_n\}$
- $M := \mathrm{Res}^*(M)$.

Return "unsatisfiable".

Note that this test runs forever if F is satisfiable.

Because $\neg F$ is unsatisfiable if and only if F is valid, this yields also a (semi-)decision procedure for validity of some formula.

Example 2.6. Consider $F = \forall x\,(P(x) \wedge \neg P(f(x)))$. The matrix of F in clause set notation is $\{\{P(x)\}, \{\neg P(f(x))\}\}$. The first four elements of $E(F)$ are

$$\{P(a)\}, \{\neg P(f(a))\}, \{P(f(a))\}, \{\neg P(f(f(a)))\}.$$

We obtain

$$\begin{array}{cccc} \{P(a)\} & \{\neg P(f(a))\} & \{P(f(a))\} & \{\neg P(f(f(a)))\} \\ & \searrow & \swarrow & \\ & \multicolumn{2}{c}{\square} & \end{array}$$

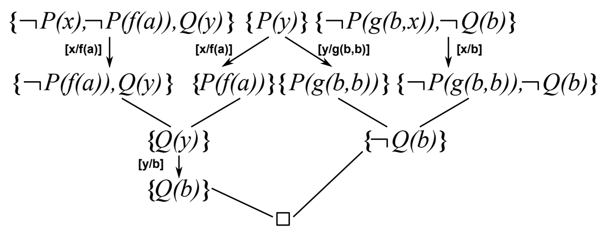
This shows that F is unsatisfiable.

In the example above we needed only four instances, and only one resolution step. Still, two of the clauses generated are superfluous. In bigger examples it is profitable to produce only clauses that are assumed to lead to $\square$ pretty quickly.

Example 2.7. Consider $F = \forall x\,\forall y\,\big((\neg P(x) \vee \neg P(f(a)) \vee Q(y)) \wedge P(y) \wedge (\neg P(g(b,x)) \vee \neg Q(b))\big)$. The matrix of F in clause set notation is

$$\{\{(\neg P(x), \neg P(f(a)), Q(y)\}, \{P(y)\}, \{\neg P(g(b,x)), \neg Q(b)\}\}$$

We substitute in a prudent manner:



Hence F is unsatisfiable.

Further questions now may be: 11. Dec

1. How can we refine this algorithm in order to make it more efficient? E.g. using prudent substitutions, or delayed substitutions (like delaying $[y/b]$ in the example above)

2. Which classes of formulas are efficiently decidable? (compare Horn formulas)

3. Can we apply the resolution calculus to prove mathematical theorems automatically?

Remark 2.9. In order to illustrate the last one consider this formula:

$$F = \forall x\, \forall y\, \forall z\, f(x, f(y,z)) = f(f(x,y),z) \wedge \forall x\, f(x,e) = x \wedge \forall x\, \exists y\, f(x,y) = e$$

Each model for F is a group (see Maths I, or wikipedia). Hence every statement derived from F is a (true) statement about groups. If there is a machine (or algorithm) who prints all true statement about groups a lot of mathematicians (namely, group theorists) would be unemployed. This is undergoing work, but it is not clear whether it will suffice. Right now the assumption is ”no”. One problem is that a machine cannot detect which result is ”important” and which one is not. The tendency goes in direction ”computer assisted proofs” and ”computer aided proofs”, see e.g. the very interesting survey Formal Proof by Thomas Hales. (Please note also the advertisement at the end of the paper.)

Interlude: Infinite cardinalities

For a finite set M it is clear ho to define the number of its elements. This number is called the **cardinality** of M, denoted by $|M|$ (sometimes also $\#M$ or $\mathrm{card}(M)$). If M has infinitely many elements this becomes more tricky. Then the following concept becomes helpful:

Definition 2.8. Two sets M and N have the same **cardinality** if there is a bijective map from M to N.

A map $f : M \to N$ is called *bijective* if for each $m \in M$ there is exactly one n in N such that $f(m) = n$. Think of a mass wedding: M consists of a bunch of men, N consists of a bunch of women. Then a bijective f assigns to each man exactly one woman and vice versa. For finite sets M and N this is clearly possible only if they have the same number of elements.

We can now extend the notion of cardinality to infinite sets like $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$..., or to their power sets.

Definition 2.9. The **power set** of some set M is the set of all its subsets, including M and $\varnothing$. It is denoted by $\mathcal{P}(M)$.

For instance, the power set of $\{1,2,3\}$ is

$$\mathcal{P}(\{1,2,3\}) = \big\{\varnothing, \{1\}, \{2\}, \{3\}, \{1,2\}, \{1,3\}, \{2,3\}, \{1,2,3\}\big\}$$

The first maybe surprising fact is the following. (Surprising, since for finite sets it is not possible that a proper subset of M has the same cardinality as M.)

Theorem 2.7 (Cantor 18??)**.** $|\mathbb{N}| = |\mathbb{Q}|$.

Proof. We need to construct a bijective map f from $\mathbb{Q}$ to $\mathbb{N}$. Write all elements of $\mathbb{Q}$ in an infinite two-dimensional array:

$$\begin{array}{cccccc}
\frac{1}{1} & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \cdots \\[2ex]
\frac{2}{1} & \frac{2}{2} & \frac{2}{3} & \frac{2}{4} & \frac{2}{5} & \cdots \\[2ex]
\frac{3}{1} & \frac{3}{2} & \frac{3}{3} & \frac{3}{4} & \frac{3}{5} & \cdots \\[2ex]
\frac{4}{1} & \frac{4}{2} & \frac{4}{3} & \frac{4}{4} & \frac{4}{5} & \cdots \\[2ex]
\frac{5}{1} & \frac{5}{2} & \frac{5}{3} & \frac{5}{4} & \frac{5}{5} & \cdots \\
\vdots & \vdots & \vdots & \vdots & \vdots &
\end{array}$$

It is clear that this array contains each positive element of $\mathbb{Q}$. Some of them occur in multiple copies (e.g. $\frac{1}{1} = \frac{2}{2} = \frac{3}{3} = \cdots$) Now walk through this array in the way indicated here in the following diagram, numbering each element of $\mathbb{Q}$ with consecutive integer numbers that is not numbered yet (e.g., don't mark $\frac{2}{2}, \frac{3}{3}, \dots$ since $\frac{1}{1}$ is numbered already).

$$\begin{array}{ccccccccccc}
\frac{1}{1}\ (1) & \rightarrow & \frac{1}{2}\ (2) & & \frac{1}{3}\ (5) & \rightarrow & \frac{1}{4}\ (6) & & \frac{1}{5}\ (11) & \rightarrow & \\
 & \swarrow & & \nearrow & & \swarrow & & \nearrow & & & \\
\frac{2}{1}\ (3) & & \frac{2}{2}\ (\cdot) & & \frac{2}{3}\ (7) & & \frac{2}{4}\ (\cdot) & & \frac{2}{5} & & \cdots \\
\downarrow & \nearrow & & \swarrow & & \nearrow & & & & & \\
\frac{3}{1}\ (4) & & \frac{3}{2}\ (8) & & \frac{3}{3}\ (\cdot) & & \frac{3}{4} & & \frac{3}{5} & & \cdots \\
 & \swarrow & & \nearrow & & & & & & & \\
\frac{4}{1}\ (9) & & \frac{4}{2}\ (\cdot) & & \frac{4}{3} & & \frac{4}{4} & & \frac{4}{5} & & \cdots \\
\downarrow & \nearrow & & & & & & & & & \\
\frac{5}{1}\ (10) & & \frac{5}{2} & & \frac{5}{3} & & \frac{5}{4} & & \frac{5}{5} & & \cdots \\
\vdots & & \vdots & & \vdots & & \vdots & & \vdots & &
\end{array}$$

It is clear that each number in the array will be marked: it will be reached after finitely many steps. This yields a bijection between $\mathbb{N}$ and the positive rationals $\mathbb{Q}^+$:

$$\begin{array}{cccccccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & \cdots \\
\downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
1 & \frac{1}{2} & 2 & 3 & \frac{1}{3} & \frac{1}{4} & \frac{2}{3} & \frac{3}{2} & 4 & 5 & \frac{1}{5} & \cdots
\end{array}$$

The rest is easy: Before the 1 add a 0, after each number its negative:

$$\begin{array}{cccccccccccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & \cdots \\
\downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
0 & 1 & -1 & \frac{1}{2} & -\frac{1}{2} & 2 & -2 & 3 & -3 & \frac{1}{3} & -\frac{1}{3} & \frac{1}{4} & -\frac{1}{4} & \frac{2}{3} & -\frac{2}{3} & \cdots
\end{array}.$$

This is the desired bijective map, hence $|\mathbb{N}| = |\mathbb{Q}|$. $\square$

This proof shows a general trick: in order to show $M| = |\mathbb{N}|$ it suffices to label all elements of M by $1,2,3,\ldots$ such that each has exactly one label. This leads to the following terminology.

Definition 2.10. A set M with $|M| = |\mathbb{N}|$ is called **countable** set. A set M with $|M| > |\mathbb{N}|$ is called **uncountable** set.

The second surprising fact is:

Theorem 2.8 (Cantor 1874)**.** $\mathbb{R}$ *is uncountable. I.e.* $|\mathbb{R}| > |\mathbb{N}|$.

The following proof is not the original one, but one found by Cantor in 1891.

Proof. by contradiction. We show that there is no bijection from $\mathbb{N}$ into the half-open interval $[0;1[$ (i.e., all numbers x such that $0 \le x < 1$). If we succeed this shows that that there is no bijection between $\mathbb{R}$ and $\mathbb{N}$, since $\mathbb{R}$ is even larger than $[0,1[$.

So let us assume there is a bijection from $\mathbb{N}$ into $[0;1[$. Then we can write the elements $a_1, a_2, \ldots$ of $[0;1[$ in a list. We use the decimal representation $0.a_{i,1}a_{i,2}a_{i,3}\ldots$.

$$\begin{array}{llllllll}
a_1 = 0. & a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} & a_{1,5} & \ldots \\
a_2 = 0. & a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} & a_{2,5} & \ldots \\
a_3 = 0. & a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} & a_{3,5} & \ldots \\
a_4 = 0. & a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} & a_{4,5} & \ldots \\
a_5 = 0. & a_{5,1} & a_{5,2} & a_{5,3} & a_{5,4} & a_{5,5} & \ldots \\
 & & & \cdots & & &
\end{array}$$

If there is an ambiguity (like $0.09999\cdots = 0.1$) we choose the finite version, ending in ...0000..., and omit the infinite version, edning in ...99999...

By our assumption this list contains all elements of $[0,1[$. We will now find a contradiction by constructing an element that is not in the list: define $s = 0.s_1s_2s_3\cdots$ by

$$s_i = \begin{cases} 7 & \text{if } a_{ii} = 3 \\ 3 & \text{if } a_{ii} \neq 3 \end{cases}$$

This number s is clearly in $[0,1[$, and it does not end in ...99999... (since its digits are all either 3 or 7). The first digit of s after the period differs from the first digit of a_1 (by construction: if the first digit of a_1 is 3, then the first digit of s is 7; and if the first digit of a_1 is not 3, then the first digit of s is 3). In a similar fashion, the ith digit of s after the period differs from the ith digit of a_i.

Hence s is not equal to any number a_i in the list. $\square$

Now that we know that there are different levels of ”infinitely many”, can we describe this hierarchy? Again Georg Cantor can help.

Theorem 2.9. *Let M be a set. Then* $|\mathcal{P}(M)| > |M|$.

For a finite set $M = \{m_1, m_2, \dots,\}$ this is a simple observation: $\mathcal{P}(M)$ contains at least $\{m_1\}, \{m_2\}, \dots$, but also $\{m_1, m_2\}$, or M. Note that $|\varnothing| = 0$, but

$$|\mathcal{P}(\varnothing)| = |\{\varnothing\}| = 1.$$

In particular the theorem implies $|\mathcal{P}(\mathbb{N})| > |\mathbb{N}|$, $|\mathcal{P}(\mathcal{P}(\mathbb{N}))| > |\mathcal{P}(\mathbb{N})|$ etc.

Definition 2.11. Let $\beth_0 := |\mathbb{N}|$. If $|M| = \beth_n$ for $n \geq 0$, let $\beth_{n+1} := |\mathcal{P}(M)|$.

The $\beth$ ("beth") is the second letter in the Hebrew alphabet.

3 Undecidability

In Theoretical Computer Science there are several levels of complexity of a (class of) problem(s): P, NP,... The question whether a formula in first-order logic is satisfiable is harder than any of them. Let us briefly sketch this hierarchy. A problem is here a problem that can have infinitely many, and arbitrary large, inputs, like an integer, or a graph, or a tree, or a list of integers. The size of the input is measured by some reasonable measure, like the number of the digits of an integer, or the number of nodes of a graph or tree, or the number of the entries of some list.

P A problem is in P if there is an algorithm that solves the problem for any input of size n in polynomial time; i.e., the number of the required steps is in $O(p(n))$ where p is some polynomial (or in general, something that is smaller than some polynomial). E.g. sorting a list of n numbers can be done in $O(n^2)$ steps (quicksort), or even in $O(n \log n)$ (mergesort). Finding the minimal number in a list of integers can be done trivially in time $O(n)$. The problem PRIME, i.e., testing whether a number is a prime number requires naively $O(2^{n/2})$ steps (this would not be in P), but there are algorithms known that need $O(n^{12})$ steps. Hence PRIME is in P.

NP A problem is in NP if each correct solution (aka "witness", or "certificate") of it can be checked in polynomial time (wrt the size n of the input). For instance, the question "does a given graph G with n nodes have a Hamiltonian cycle" is in NP. (A graph has a Hamiltonian cycle if it contains a cycle that contains each node of G exactly once.) In this example a solution is such a Hamiltonian cycle, and it is easily checked whether a given solution is correct. On the other hand, finding a Hamiltonian cycle can be pretty hard in general.

Similarly, the prime factorization of integers with n digits is in NP: it might be hard to tell what the prime factors of 1003 are, or of 1007 (are they prime numbers or composite numbers?) But a witness for the factorization of 1007 is $(19, 53)$, since $19 \cdot 53 = 1007$. (And 1003 = $17 \cdot 59$.)

It is a big open problem whether P=NP. It is generally assumed that NP is a bigger class than P. A problem is called **NP-hard** if a solution of this problem can be translated into a solution for any problem in NP in polynomial time. A problem that is both in NP and NP-hard is called **NP-complete**. Figure 2 illustrates the situation (for both cases, P=NP and P$\neq$NP). It is known that the Hamiltonian cycle problem is NP-complete, but prime factorization of integers is not. Usually such results are shown by reducing a problem to SAT, the question whether a formula in propositional logic in CNF is satisfiable, compare Remark 1.3. SAT was the first problem known to be NP-complete.

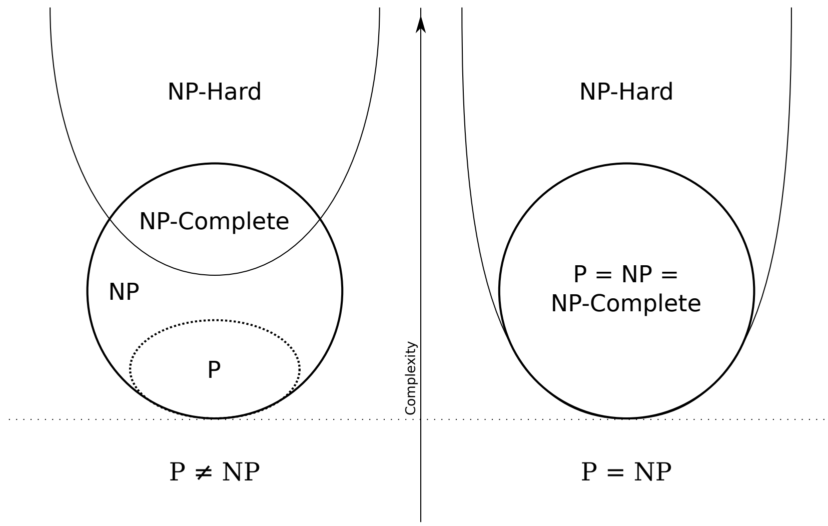


Figure 2: The situation if P$\neq$NP (left) and if P=NP (right).

Beyond NP Are there problems that are even harder than problems in NP? Yes. Later we will see undecidable problems. As a side remark, a class of problems that is harder than NP problems is the class NEXP. This is the class of problems having instances where all witnesses are of exponential size. It is known that NP$\neq$NEXP (more precisely, NP is a proper subset of NEXP). The examples for this look somehow artificial. For instance, the question whether a non-deterministic Turing machine will stop after n steps is in NEXP, since each witness is already of size $O(2^n)$. In a similar manner, it is known that certain graphs with 2^n nodes can be encoded into logical circuits with $O(n)$ logic gates ("succinct circuits"). If one asks for a Hamiltonian cycle in such a graph this problem is in NEXP, since each witness is of size $O(2^n)$. In the sequel we will see that there are even harder problems, namely, undecidable problems. One is the question whether a given formula in first-order logic is satisfiable.

3.1 Undecidable problems

We start with a simple consequence of Theorem 2.5.

Corollary 3.1. *Each satisfiable formula has a model with a countable universe.*

This result follows immediately since we know that any (non-finite) Herbrand universe is countable.

Recall that $\mathbb{N}$ has countably many elements, but $\mathbb{R}$ has uncountably many elements. (In fact, the Löwenheim-Skolem theorem states that each satisfiable formula has a model for each infinite cardinal number $\beth_n$).

In contrast to propositional logic, where the satisfiability of any given formula can be checked in finite time (e.g. by considering all possible valuation (finitely many!)) this is not true for formulas in first-order logic. As a first result we need

Theorem 3.2. *The are satisfiable formulas in first-order logic possessing only infinite models; that is, models $\mathcal{A} = (U_\mathcal{A}, I_\mathcal{A})$ such that $U_\mathcal{A}$ is an infinite set.*

Proof. (Exercise 34) □

This is of course not yet a proof for undecidability, it is only a necessary ingredient. It might still be true at this stage that there *is* a decision procedure. Anyway, first we need to define what "undecidable" means. Here it is helpful to use Turing machines. It is assumed that the reader is familiar with Turing machines. If not think of a Turing machine as an algorithm or a computer program.

Definition 3.1. A **decision problem** is a yes-no question with infinitely many possible inputs. Formally it can be modeled as a pair (I,M) where I is the set of all possible inputs, and $M \subset I$ is the set of inputs with answer "yes".

A problem is **decidable** (aka computable, berechenbar, entscheidbar) if there is a Turing machine that answers correctly "yes" or "no" on any input. A problem is **semi-decidable** (aka semi-computable) if there is a Turing machine that answers correctly "yes" for every input with answer "yes", and does not answer "yes" for any input with answer "no".

Usually the inputs can be enumerated, hence in several cases one has $I = \mathbb{N}$ or $I = \mathbb{N}^k$ and $M \subset \mathbb{N}$. Then one can state the same question using a function $f : \mathbb{N}^k \to \{0,1\}$, with $f(n) = 1$ if $n \in M$, $f(n) = 0$ else. The corresponding names for f are then **recursive** (corr. to decidable=computable) and **recursively enumerable** (corr. to semi-decidable).

A famous problem that is undecidable is the question "does Turing machine number i ever stop on input j?" This question is known as the halting problem. Since there are countably infinitely many Turing machines we may enumerate all Turing machines by numbers in $\mathbb{N}$. All inputs (finite starting configurations on the tape) can be enumerated as well (by some appropriate scheme).

Theorem 3.3 (Turing 1937)**.** *The halting problem is undecidable.*

Proof. Phrased as a function the problem asks now for a *computable* $f : \mathbb{N}\times\mathbb{N} \to \{0,1\}$ where $f(i,j) = 1$ if Turing machine i stops on input j, $f(i,j) = 0$ else. Let f be computable. Consider

$$g : \mathbb{N} \to \{0,1\}, \quad g(i) = \begin{cases} 0 & \text{if } f(i,i) = 0 \\ \text{undefined} & \text{else} \end{cases}$$

For instance, as a program g can be realized as

If $f(i,i) = 0$ then return 0 else run forever

Clearly g is computable if f is. Hence there is a Turing machine t that computes g. What is $f(t,t)$?

If $f(t,t) = 0$ then $g(t) = 0$. In particular (since the Turing machine t computes g) the t stops on input t, and $f(t,t) = 1$. Contradiction.

If $f(t,t) = 1$ then $g(t)$ is not defined, hence t does not stop on input t, thus $f(t,t) = 0$. Contradiction. The assumption that f is computable leads to a contradiction, hence f is not computable. □

Theorem 3.4 (Turing 1937)**.** *The halting problem is semi-decidable. I.e. there is a Turing machine that returns* 1 *on input* (i,j) *if Turing machine i stops on input j, and returns nothing if not.*

Proof. The rough idea is: just let a given Turing machine i run on input j. If it stops return 1. As usually we rely on the results in Theoretische Informatik and formulate just an algorithm without implementing it as an actual Turing machine.

We can build one universal Turing machine t that checks the kth step for Turing machine i on input j in the following order:

$$(i,j,k) = (1,1,1),(1,1,2),(1,2,1),(2,1,1),(1,1,3),(1,2,2),(1,3,1),(2,1,2),(2,2,1),(3,1,1),(1,1,4),\ldots$$

In plain words: list all pairs (i,j,k) with $i,j,k \in \mathbb{N}\setminus\{0\}$ in ascending order wrt $i+j+k$ and then wrt lexicographical order. It is clear that all possibilities in $\mathbb{N}^3$ are covered. Our Turing machine takes input i and returns 1 if Turing machine i eventually stops. $\square$

Some undecidable problems The following problems are undecidable (usually, semi-decidable)

1. Is a given formula in first-order logic valid? (Church 1936, Turing 1937) See below for details.

2. The **Post correspondence problem**: The input of the problem consists of two finite lists $u_1,\ldots,u_n$ and $v_1,\ldots,v_n$ of words over the alphabet $\{0,1\}$. A solution to this problem is a sequence of indices $i_1,i_2,\ldots,i_k$ with $k \geq 1$ such that

$$u_{i_1}\ldots u_{i_k} = v_{i_1}\ldots v_{i_k}.$$

The decision problem then is to decide whether such a solution exists or not.

A nice illustration of this problem is: assume we have finitely many distinct domino tiles, with some 0-1-words on top and on bottom. With an arbitrary large amount of copies of these, can we place them in a row such that the top row shows the same string as the bottom row?

For instance assume that we have $u_1 = 1, u_2 = 10, u_3 = 011$, and $v_1 = 101, v_2 = 00, v_3 = 11$. The tiles thus look as follows:

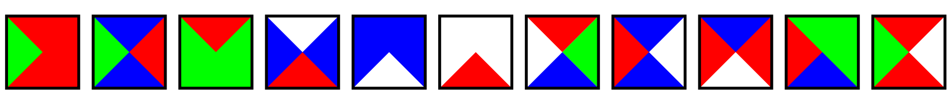
$$\left[\begin{smallmatrix}1\\101\end{smallmatrix}\right], \quad \left[\begin{smallmatrix}10\\00\end{smallmatrix}\right], \quad \left[\begin{smallmatrix}011\\11\end{smallmatrix}\right].$$

A solution is $(1,3,2,3)$, that is,

$$\left[\begin{smallmatrix}1\\101\end{smallmatrix}\right]\left[\begin{smallmatrix}011\\11\end{smallmatrix}\right]\left[\begin{smallmatrix}10\\00\end{smallmatrix}\right]\left[\begin{smallmatrix}011\\11\end{smallmatrix}\right].$$

The word in the top row, as well as in the bottom row, is 101110011.

3. The **Wang tile problem**: Given a set of squares such that the edges are coloured with distinct colours ("Wang tiles"). Can the plane be tiled by copies of those such that adjacent edges have the same colour? Tiling the plane means: placing the tiles such that they do not overlap and leave no gaps. Tiles should lie vertex-to-vertex. It is not allowed to rotate or reflect tiles. For instance, the set of 11 tiles below can tile the plane (but it is veeeeery tricky to see how)



No subset of 10 tiles can tile the plane. In general this problem is undecidable for more than three colours and more than 10 tiles.

4. **Mortal matrix problem**: Given some matrices $A_1,\ldots,A_n$ in $\mathbb{R}^{3\times 3}$. Is there any combination of them (say, $(i_1,i_2,\ldots,i_m)$ with $1 \leq i_j \leq n$) such that

$$A_{i_1}\cdot A_{i_2}\cdots A_{i_m} = \begin{pmatrix} 0 & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & 0\end{pmatrix}\ ?$$

This problem is undecidable for $n \geq 6$.

5. **Conway's Game of Life** (see wikipedia): given some start pattern and another pattern, can the latter ever be the result of the first one?

6. **Diophantine equations:** Given a polynomial with several variables, but integer coefficients, does it have an integer solution? (Diophantine means we are only interested in integer solutions). For example, the Diophantine equation $3x^2 - 2xy - y^2z - 7 = 0$ has an integer solution: $x = 1,\ y = 2,\ z = -2$. By contrast, the Diophantine equation $x^2 + 4y^2 - 3 = 0$ has no such solution (only non-integer ones, like $x = 1, y = \frac{\sqrt{2}}{2}$, or $x = \sqrt{2}, y = \frac{1}{2}$).

In a similar manner as the halting problem all problems above can be shown to be semi-decidable. We will now take a closer look at 1.

3.2 Undecidability of first-order logic

Usually the undecidability of some problem is proven by relating it to the halting problem. We will illustrate this with the following result.

Theorem 3.5 (Church 1936, Turing 1937)**.** *The problem whether a formula in first-order logic is satisfiable is undecidable.*
Since F satisfiable if and only $\neg F$ is valid, the problem whether a formula in first-order logic is valid is undecidable as well.

One strategy of the proof is the following (the original proof was different):

1. Show that: If the problem above is decidable then the Post correspondence problem is decidable.
2. Show that: If the Post correspondence problem is decidable then the halting problem is decidable.

This yields a contradiction to Theorem 3.3. Hence the problem above must be undecidable. It remains to show 1 and 2. In the sequel we outline the proof of 1, and omit the proof of 2.

Proof. (Of 1.) The general idea is to translate an *arbitrary* Post correspondence problem PCP into some formula F in first-order logic such that PCP has a solution if and only if F is valid.

Let PCP be given: $(u_1, v_1), \ldots, (u_n, v_n)$. The corresponding F uses a constant a, two functions f_0, f_1 and one predicate P with two inputs. We need the notation $f_{j_1 j_2 \cdots j_{\ell-1} j_\ell}(x) = f_{j_\ell}(f_{j_{\ell-1}}(\cdots f_{j_2}(f_{j_1}(x))\cdots))$. Then F is of the form $F = F_1 \wedge F_2 \Rightarrow F_3$, where

$$F_1 = \bigwedge_{i=1}^{n} P(f_{u_i}(a), f_{v_i}(a))$$

$$F_2 = \forall x\, \forall y\, \left(P(x,y) \Rightarrow \bigwedge_{i=1}^{n} P(f_{u_i}(x), f_{v_i}(y))\right)$$

$$F_3 = \exists z\, P(z,z)$$

We need to show: F is valid if and only if PCP has a solution. F being valid means that *each* (matching) structure $\mathcal{A}$ satisfies F. Hence also the following structure:

$$U_{\mathcal{A}} = \{0,1\}^* \quad \text{(the set of all finite 0-1-words)}$$
$$a^{\mathcal{A}} = \varepsilon \quad \text{(the empty word with no letters)}$$
$$f_0^{\mathcal{A}}(x) = x0, \quad f_1^{\mathcal{A}}(x) = x1 \quad \text{(e.g. } f_0^{\mathcal{A}}(11) = 110)$$
$$P^{\mathcal{A}} = \{(x,y) \mid x,y \in \{0,1\}^* \mid x,y \in \{0,1\}^* \setminus \{\varepsilon\}, \text{ and there are } i_1,\ldots,i_k \text{ such that } x = u_{i_1}\cdots u_{i_k};\ y = v_{i_1}\cdots v_{i_k}\}$$

$P(x,y)$ means that x and y are build from the same domino pieces (and $x \neq \varepsilon \neq y$). F_1 means here: P is true for all single pieces (with number i): $f_{u_i}(\varepsilon) = u_i$ and $f_{v_i}(\varepsilon) = v_i$. Hence $\mathcal{A} \models F_1$.

F_2 means: if x and y are build from the same pieces, then $f_{u_i}(x) = xu_i$ and $f_{v_i}(y) = yv_i$ are build from the same pieces. Hence $\mathcal{A} \models F_2$.

"$\Rightarrow$" Let us assume F is valid. I.e. $\mathcal{A}(F) = 1$. Because of $\mathcal{A} \models F_1$ and $\mathcal{A} \models F_2$ it follows $\mathcal{A} \models F_3$ (otherwise F would be false). This means there is some word z in $\{0,1\}^*$ such that $P(z,z) = 1$, that is, z is a solution of PCP.

"$\Leftarrow$" Let us assume PCP has some solution z, with indices $i_1,\ldots i_n$, hence $u_{i_1}\cdots u_{i_n} = v_{i_1}\cdots v_{i_n}$. Let $\mathcal{A}'$ be an arbitrary structure for F. We need to show that $\mathcal{A}' \models F$. Because of $F = F_1 \wedge F_2 \Rightarrow F_3$ it follows that $\mathcal{A}' \models F$ whenever $\mathcal{A}' \not\models F_1$ or $\mathcal{A}' \not\models F_2$. Hence it suffices to consider the case $\mathcal{A}' \models (F_1 \wedge F_2)$.

Embed our first model $\mathcal{A}$ into $\mathcal{A}'$ by the map $\mu : \{0,1\}^* \to U_{\mathcal{A}}$ with

$$\mu(\varepsilon) = a^{\mathcal{A}'},\ \mu(x0) = f_0^{\mathcal{A}'}(\mu(x)),\ \mu(x1) = f_1^{\mathcal{A}'}(\mu(x)).$$

Because of $\mathcal{A}' \models F_1$ for all initial pieces i holds $(\mu(u_i),\mu(v_i)) \in P^{\mathcal{A}'}$. Because of $\mathcal{A}' \models F_2$ holds that whenever $(\mu(x),\mu(y)) \in P^{\mathcal{A}'}$ it follows that $(\mu(xu_i),\mu(yv_i)) \in P^{\mathcal{A}'}$. Inductively this means that all legal piece combinations (wrt PCP) are in $P^{\mathcal{A}'}$. In particular the legal piece combination

$$(\mu(u_{i_1},\ldots,u_{i_n}),\mu(v_{i_1},\ldots,v_{i_n})) \in P^{\mathcal{A}^*}.$$

In other words, for the solution $z = \mu(u_{i_1},\ldots,u_{i_n}) = \mu(v_{i_1},\ldots,v_{i_n})$ holds $P(z,z)$. Hence $\mathcal{A}' \models \exists z\, P(z,z)$, hence $\mathcal{A}' \models F_3$, and altogether $\mathcal{A}' \models F$. $\square$

Point 2 above is proven by constructing for each Post correspondence problem PCP a Turing machine that stops if and only if PCP has a solution. The construction is tricky, lengthy and technical, so we omit it here. A detailed proof is contained in M. Sipser: Introduction to the theory of computation.

Other undecidable problems are reduced to the halting problem in a similar manner. E.g., regarding the Wang tiles, one shows that one can translate any Turing machine into Wang tiles. More precisely: for each Turing machine there is a set of Wang tiles such that the rows in a possible simulate the state of the tape of the Turing machine over time (letters on the tape and position of the head, row i is time step i). For details see e.g. Grünbaum and Shephard: Tilings and Patterns.

3.3 Computable numbers

The paper in which Turing proved Theorem 3.5 also introduced Turing machines. It was titled "On computable numbers with an application to the Entscheidungsproblem". Turing defined a number to be computable as follows: all integers are computable. A number $x \notin \mathbb{Z}$ is computable if there is a Turing machine that computes the n-th digit of its fractional part ("die n-te Nachkommastelle") on input n. This is in essence equivalent to the modern definition:

Definition 3.2. A number $x \in \mathbb{R}$ is **computable** if for each $\varepsilon > 0$ there is a computable function $f : \mathbb{Q} \to \mathbb{Q}$ such that $|x - f(\varepsilon)| < \varepsilon$.
The set of all computable numbers is denoted by $\mathbb{B}$.

The modern definition solves the problem of a Turing machine approximating the number 2, for instance, by computing n digits of $1,9999999\ldots$. Turing already realized that not all numbers are computable. The argument is simple: There are only countably many Turing machines (because each Turing machine can be encoded as some finite 0-1-word) hence there are at most countably many computable numbers. Since $\mathbb{R}$ is uncountable, almost all numbers in $\mathbb{R}$ are not computable. Nevertheless, Turing already showed that

1. All rational numbers are computable. In other words/symbols: $\mathbb{Q} \subset \mathbb{B}$
2. All algebraic numbers are computable (this is, the roots of some polynomial with integer coefficients).
3. π and e are computable.
4. The limit of a computable sequence that is computably convergent is computable.

In order to explain point 4: It is not true that the limit of any convergent sequence of any computable number is computable. We need to make sure that: (a) the sequence itself is computable, and (b) the fact that it converges is computable.

Definition 3.3. A sequence a_n is called **computably convergent** is there is a computable function $N : \mathbb{B} \to \mathbb{N}$ such that

$$\forall \varepsilon > 0 \, \forall m > N(\varepsilon), n > N(\varepsilon) : \quad |a_n - a_m| < \varepsilon$$

Compare this definition with the definition of a Cauchy sequence.

Now point 4 ensures that

$$\pi = 4(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \cdots) \quad \text{and} \quad e = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \frac{1}{4!} + \cdots$$

are computable. Moreover, point 4 ensures that all algebraic numbers are computable, because there are efficient methods to approximate roots of polynomials. In fact, almost all numbers we can describe in words are computable. Exceptions are constructed as follows:

A non-computable number Consider the halting problem for Turing machines. Let us assume we did agree already on an enumeration of all Turing machines. Let $0 < a < 1$ such that the nth digit of a is 0 if Turing machine number n stops on input 0, and 1 else. It is clear that $a \in \mathbb{R}$. If a would be computable we would have a solution to the halting problem. This contradicts Theorem 3.4.

3.4 Consequences II

22. Jan (for 8 Jan. see Sec. 4 below)

The definition of a consequence in Section 1 was a preliminary one that is OK for propositional logic. In general one needs to distinguish two different flavours of consequence.

Definition 3.4. A formula G is a (semantic) **consequence** of F, if for all $\mathcal{A}$ we have that whenever $\mathcal{A} \models F$ then $\mathcal{A} \models G$. In this case we write shortly $F \models G$.

A formula G is a (syntactic) **consequence** of F, if there is a proof (in the underlying formal system) of $\mathcal{A} \models G$, if we assume $\mathcal{A} \models F$. In this case we write shortly $F \vdash G$.

Right now I translate for myself syntactic consequence as "true" and semantic consequence as "provable". For propositional logic there is no difference between these two: everything that is true in propositional logic can be proven in propositional logic. Moreover, everything that can be proven in propositional logic is true in propositional logic. (That is why the notation in Def. 1.6. is mixed up: we *defined* semantic consequence there and *wrote* it $\vdash$, i.e., syntactic consequence.)

3.5 Gödel's incompleteness theorems

A celebrated result in the area of undecidability is due to the legendary (and tragic) figure of Kurt Gödel. There are two distinct senses of the word "undecidable" in the context of logic, or computability. The first of these is used in relation to computability theory and applies not to statements but to decision problems, which are countably infinite sets of questions each requiring a yes or no answer, as in Subsection 3.2. I.e., such a problem is said to be undecidable if there is no computable function (Turing machine) that correctly answers every question in the problem set, see Subsection 3.1.

The second sense of this term is the sense used in relation to Gödel's theorems, that of a statement being neither provable nor refutable in a specified deductive system. Let us call this concept **independent**. In order to illustrate this context let us briefly describe how to extend our terminology above in order to understand Gödel's famous incompleteness theorems.

Let us define a *theory* as a set of formulas (for instance, in first-order logic) that is closed under syntactic consequence. More precisely:

Definition 3.5. A set $\mathcal{F}$ of formulas is a **theory** if for all $F_1, \ldots, F_n \in \mathcal{F}$ and G a formula holds: if $F_1 \wedge \ldots \wedge F_n \vdash G$ then $G \in \mathcal{F}$.

Recall that $F \vdash G$ is short for "G is a syntactic consequence of F". Whenever we use $F \vdash G$ we assume that there is some *efficient* way to deduce G from F.

Since this is a formal definition it may happen that we get contradiction. For instance, if we define $\mathcal{F}$ to consist of *all* formulas in first-order logic, then $\mathcal{F}$ contains unsatisfiable formulas.

Definition 3.6. A theory $\mathcal{F}$ is **consistent** if there is no G such that both $\mathcal{F} \models G$ and $\mathcal{F} \models \neg G$.
A theory $\mathcal{F}$ is **complete** if for all G holds: either $G \in \mathcal{F}$ or $\neg G \in \mathcal{F}$.

Here are two approaches how to define a theory:

Axiomatic method Define a (finite or infinite) set $\mathcal{F}$ of formulas, and let

$$\mathrm{Cons}(\mathcal{F}) = \{G \mid \text{ there are } F_1, \ldots, F_n \in \mathcal{F} \text{ such that } F_1 \wedge \cdots \wedge F_n \vdash G\}$$

In the context of decidability $F \vdash G$ is to be understood as: it should be possible to determine in finite time whether G is a consequence of F, or not.

Example 3.1. What if $\mathcal{F}$ is empty? That is, what is $\mathrm{Cons}(\varnothing)$? This is the set of all formulas that are consequences of nothing; hence true under any condition (in all worlds). Hence

$$\mathrm{Cons}(\varnothing) = \{F \mid F \text{ is valid}\}$$

We have seen in Subsection 3.2 that this theory is not decidable in first-order logic, since the problem whether F is *not* valid cannot be answered in finite time in general.

Example 3.2. Let us consider the theory of groups, compare Remark 2.9. That is, consider $\mathrm{Cons}(\{F\})$ for

$$F = \forall x\, \forall y\, \forall z\, f(x, f(y,z)) = f(f(x,y),z) \wedge \forall x\, f(x,e) = x \wedge \forall x\, \exists y\, f(x,y) = e$$

Again this theory is not decidable.

Model theoretic approach In this method a structure $\mathcal{A}$ is given, and the theory $\mathrm{Th}(\mathcal{A})$ of $\mathcal{A}$ is the set of all formulas satisfied under $\mathcal{A}$. Formally

$$\mathrm{Th}(\mathcal{A}) = \{F \mid \mathcal{A} \models F\}$$

It is clear that $\mathrm{Th}(\mathcal{A})$ is a theory (i.e closed under $\vdash$): If $F \vdash G$ then $\mathcal{A} \models F$ implies $\mathcal{A} \models G$, hence G is in $\mathrm{Th}(\mathcal{A})$. It is also clear that $\mathrm{Th}(\mathcal{A})$ is always consistent: if $\mathcal{A} \models F$ then $\mathcal{A} \not\models \neg F$. Now we can formulate Gödel's results precisely.

Theorem 3.6 (Gödel's first incompleteness theorem, 1931)**.** $\mathrm{Th}(\mathbb{N},+,\cdot)$ *is not complete.*

The far-reaching consequence of this result is: any consistent formal system $\mathcal{F}$ within which a certain amount of elementary arithmetic can be carried out is incomplete. Sad.

Again the proof of this result uses a kind of diagonalization argument. All statements are numbered, and a statement "this theorem is not provable" can be formulated as "statement number i is not provable". The statement generalizes to all theories containing $\mathrm{Th}(\mathbb{N},+,\cdot)$, for instance, $\mathrm{Th}(\mathbb{R},+,\cdot)$.

Theorem 3.7 (Gödel's second incompleteness theorem, 1931)**.** *Let G be the statement* "$\mathrm{Th}(\mathbb{N},+,\cdot)$ *is consistent". Then* $G \notin \mathrm{Th}(\mathbb{N},+,\cdot)$*. I.e. G cannot be deduced from the axioms of* $\mathrm{Th}(\mathbb{N},+,\cdot)$.

Let us mention that there are several ways to define natural numbers by formulas in first-order logic (e.g. **Peano axioms**). But usually this requires the usage of infinitely many formulas. One possibility uses the identity, one constant (0), one function with one argument (S, "successor", i.e. $S(x) = x+1$) and two functions with two arguments: $f(x,y)$ (to be understood as +) and $g(x,y)$ (to be understood as $x \cdot y$). There are six particular formulas

1. $\forall x\, 0 \neq S(x)$ (read: $0 \neq x+1$)
2. $\forall x \forall y\, S(x) = S(y) \Rightarrow x = y$ (read: $x+1 = y+1 \Rightarrow x = y$)
3. $\forall x\, f(x,0) = x$ (read: $x+0 = x$)
4. $\forall x \forall y\, f(x,S(y)) = S(f(x,y))$ (read: $x+(y+1) = (x+y)+1$)
5. $\forall x\, g(x,0) = 0$ (read: $x \cdot 0 = 0$)
6. $\forall x \forall y\, g(x,S(y)) = f(g(x,y),x)$ (read: $x \cdot (y+1) = x \cdot y + x$)

and in addition countably infinitely many formulas:

$$\forall y_1,\ldots,y_k\, (P(0,y_1,\ldots,y_k) \wedge \forall x\, (P(x,y_1,\ldots,y_k) \Rightarrow P(S(x),y_1,\ldots,y_k)) \Rightarrow \forall x\, P(x,y_1,\ldots,y_k))$$

These latter formulas realize the **principle of induction**.

In this form there are several different models for these formulas. One needs e.g. second-order logic in order to state formulas such that $(\mathbb{N},+,\cdot)$ is essentially the only model for them.

Finally note that there are also complete (and consistent) theories, for instance the **Presburger arithmetic** $\mathrm{Th}(\mathbb{N},+)$, or the first-order theory of Euclidean geometry, or monadic first-order logic.

3.6 Gödel's completeness theorem

I imagined that I understood the incompleteness theorems above until I thought about this other result by Gödel:

Theorem 3.8. *Let $\mathcal{F}$ be a set of formulas in first order logic, and G be a formula.*

$$\mathcal{F} \models G \text{ implies } \mathcal{F} \vdash G$$

Very naively one might be confused by "everything can be proven" (completeness theorem) versus "not everything can be proven" (incompleteness theorems). A more careful formulation is this one: "All true sentences can be proven" (completeness theorem) versus "There are sentences that are neither true nor false" (incompleteness theorem). This is a clumsy interpretation, but one might get some better idea about the difference.

Indeed, by Definition 3.4 $\mathcal{F} \models G$ means: whenever $\mathcal{A} \models \mathcal{F}$ then $\mathcal{A} \models G$ (G is a semantic consequence.) And $\mathcal{F} \vdash G$ means that we can deduce G from $\mathcal{F}$ in the formal system.

So as well as in propositional logic and in modal logic, also for first order logic holds:

Everything that is true in first order logic can be proven in first order logic. But:

Not every statement in first order logic is either true or false (viewed from within the formal system)

4 Modal logic

In some contexts it is desirable to generalize the notions of "true" and "false". E.g. "I am hungry" or
8. Jan "I wear a blue shirt" depends on the circumstances, e.g. on the time of the day or week. With respect to chronological order we may distinguish

- A is always true vs
- A is sometimes true

Or with respect to conseqeunce we may distinguish

- A is necessarily true vs
- A is possibly true

For this purpose we introduce two new operators: $\Box$ and $\diamond$.

4.1 Syntax and semantics

The modal logic we discuss here is an extension of propositional logic. There is also a first-order modal logic, but it is not part of this course.

Definition 4.1 (Syntax)**.** A formula in modal logic is defined inductively as follows:

1. Each formula in propositional logic is a formula in modal logic.
2. If F and G are formulas in modal logic, then $\neg F$, $F \vee G$, $F \wedge G$, $\Box F$ ("necessary F") and $\diamond F$ ("possibly F") are formulas in modal logic.

For instance $\diamond A \Rightarrow \neg\Box(B \wedge \diamond\neg C)$ is a formula (in modal logic; in the remainder of this section "formula" means always "formula in modal logic" if not explicitly stated that not).

In order not to use too many brackets we agree that $\diamond$ and $\Box$ are stronger than $\wedge$ and $\vee$ (hence stronger than $\Rightarrow$ and $\Leftrightarrow$).

Example 4.1. Three football experts make three statements:

1. "If Schalke wins the championship ever again then I will eat my hat."
2. "At some time your statement will become true."
3. "Statement 2 being correct is the same as saying that Schalke will always be champion from now on."

If A="Schalke will be champion" and B="expert 1 will eat his hat" then 1. can be stated as $\diamond A \Rightarrow B$, 2. becomes $\diamond(\diamond A \Rightarrow B)$, and 3. becomes $\diamond(\diamond A \Rightarrow B) \Leftrightarrow \Box A$.

In order to define the semantics of modal logic we need to specify a collection W of different states ("worlds") in which a formula holds, or does not hold. For each state in W there are further points in W that can be reached from this state ("possible futures"), others not (e.g. "the past", or "impossible futures"). This is realised as follows.

Definition 4.2 (Semantics)**.** A **structure** (in modal logic) is a triple $\mathcal{A} = (W, R, \alpha)$, where

- (W,R) is pair such that W is a nonemtpy set and $R \subset W \times W$. (I.e., R is a relation on W). The pair (W,R) is called the **frame**.
- Let M be a set of atomic formulas. A map $\alpha : M \times W \to \{0,1\}$ is a **valuation** of M. In plain words: α assigns to each pair (A,s) true or false.

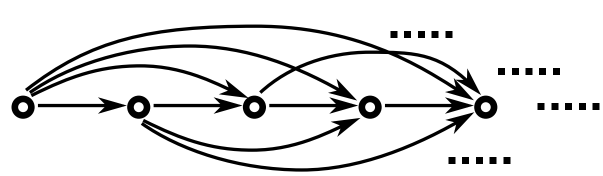
The pair (W,R) is called a **frame** for F. (W,R) can be viewed as a *directed graph*, see below.

Now we can define truth values for a formula F inductively as follows.

1. Is F an atomic formula A then $\mathcal{A}(F,s) = \alpha(A,s)$.
2. If F is a formula then $\mathcal{A}(\neg F,s) = \begin{cases} 1 & \text{if } \alpha(F,s) = 0 \\ 0 & \text{else} \end{cases}$
3. If F and G are formulas then $\mathcal{A}(F \wedge G,s) = \begin{cases} 1 & \text{if } \alpha(F,s) = \alpha(G,s) = 1 \\ 0 & \text{else} \end{cases}$
4. If F and G are formulas then $\mathcal{A}(F \vee G,s) = \begin{cases} 1 & \text{if } \alpha(F,s) = 1 \text{ or } \alpha(G,s) = 1 \\ 0 & \text{else} \end{cases}$
5. Is F a formula then $\mathcal{A}(\Box F,s) = \begin{cases} 1 & \text{if } \mathcal{A}(F,t) = 1 \text{ for all } t \in W \text{ with } (s,t) \in R \\ 0 & \text{else} \end{cases}$
6. Is F a formula then $\mathcal{A}(\diamond F,s) = \begin{cases} 1 & \text{if } \mathcal{A}(F,t) = 1 \text{ for some } t \in W \text{ with } (s,t) \in R \\ 0 & \text{else} \end{cases}$

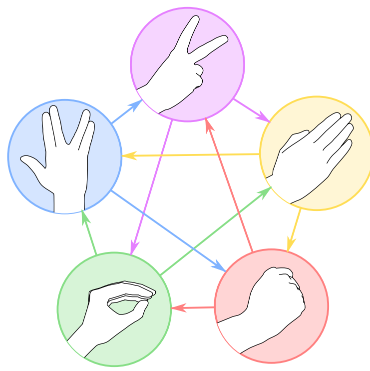
As always we implicitly assume that $\mathcal{A}$, resp. M, contains all atomic formulas of F.

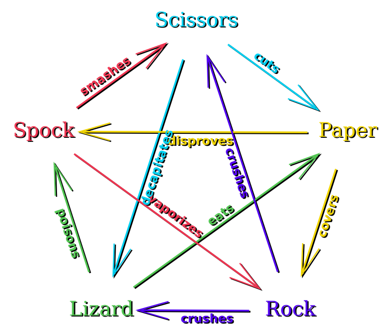
Example 4.2. A frame (W,R) can be visualized as a **directed graph**. The elements of W are the nodes of the graph, and an element $(n,m) \in R$ yields a directed edge from n to m. For instance the graph for the relation in the example above, i.e. for $(W,R) = (\mathbb{N}_0, <)$ looks in part as follows:



In particular the graph above is an infinite graph. Moreover, from each node infinitely many edges are leaving. A simpler example are the days of the week $\{1,2,\ldots,7\}$, ordered with respect to $<$.

Another simple example is the game stone-scissors-rock-spock-lizard. Here $W = \{$ stone, scissors, rock, lizard, spock $\}$, and $R = \{(n,m) \mid n,m \in W,\ n \text{ beats } m\}$. The corresponding graph is (two versions)





Example 4.3. (Ex. 4.1 cont.) A possible structure for $F = \diamond(\diamond A \Rightarrow B) \Leftrightarrow \Box A$ is $\mathcal{A} = (W, R, \alpha)$, where $W = \mathbb{N}_0$ (0 stands for the current championship 2018/19, 1 for the next championship and so on), the accessibility relation R is $<$ (that is, $R = \{(n,m) \mid n,m \in \mathbb{N}_0, n < m)\}$). A valuation α that makes F true is

$$\alpha(A,s) = 1 \text{ for all } s \in \mathbb{N}_0, \quad \alpha(B,s) = 1 \text{ for } s \text{ odd}, \quad \alpha(B,s) = 0 \text{ else.}$$

("Schalke will always be champion from now on", "expert 1 eats his hat in season 2018/19, 2020/21, 2022/23,...") Another structure for F — using the same frame — is $\mathcal{A}' = (\mathbb{N}_0, <, \beta)$ is for instance given by

$$\beta(A,s) = 1 \text{ for } s = 2, \beta(A,s) = 0 \text{ else}, \quad \beta(B,s) = 1 \text{ for } s = 3, \quad \beta(B,s) = 0 \text{ else.}$$

In this case $\mathcal{A}'(\diamond(\diamond A \Rightarrow B), s) = \mathcal{A}'(\diamond(\neg\diamond A \vee B), s) = 1$ for all $s \geq 2$ (since $\mathcal{A}'(\diamond A, s) = 0$ for $s \geq 2$, hence $\mathcal{A}'(\neg\diamond A, s) = 1$ for $s \geq 2$, hence $\mathcal{A}'(\diamond(\neg\diamond A \vee B), t) = 1$ for all t). On the other hand we have $\mathcal{A}'(\Box A, s) = 0$ for all s. Hence $\mathcal{A}'(F) = 0$ in this structure.

In particular the truth of a formula F depends on the point s in which we evaluate F. Therefore we will adjust the terms "model" and "valid" now to the context of modal logic.

The option to choose different frames for a given formula F, as well as different valuations for each frame, leads to a diversity of options for F "being true". The following definition lists all possibilities. In the sequel we will concentrate on the usual question "Is F unsatisfiable?" (or not!)

Definition 4.3. Let F be a formula in modal logic and $\mathcal{A} = (W, R, \alpha)$ a structure for F.

- If $\mathcal{A}(F,s) = 1$, then F **holds** in s. (One writes short $s \Vdash_{\mathcal{A}} F$, or $s \Vdash F$ if $\mathcal{A}$ is clear from the context)
- If $\mathcal{A}(F,s) = 1$ for all $s \in W$ we call $\mathcal{A}$ a **model** for F, short: $\mathcal{A} \models F$.
- Let a frame (W,R) be given. If for all structures $\mathcal{A} = (W,R,\alpha)$ holds $\mathcal{A} \models F$ then F is called **valid** in (W,R). In this case we write shortly $(W,R) \models F$.
- F is a **tautology** if F is valid in each frame for F.

With respect to Example 4.3 above: with $\mathcal{A} = (\mathbb{N}_0, <, \alpha)$ we have that $\alpha(F, s) = 1$ for all $s \in \mathbb{N}_0$, hence $\mathcal{A}$ is a model for F, short: $\mathcal{A} \models F$.

With $\mathcal{A}' = (\mathbb{N}_0, <, \beta)$ we saw that $\mathcal{A}'(F, s) = 0$ for all s. Hence $\mathcal{A}' \not\models F$ (but $\mathcal{A}' \models \neg F$). Therefore we also see $(N_0, <) \not\models F$, that is, F is not valid in $(\mathbb{N}_0, <)$. Because of this, F is in particular no tautology.

4.2 Calculation rules (and normal forms)

As in Sections 1 and 2 we also have some rules of calculation for modal logic. In order to state these we need to define equivalence and consequence.

Definition 4.4.

G is a **consequence** of $\mathcal{F}$ (short: $\mathcal{F} \vdash G$), if for all $\mathcal{A}$ with $\mathcal{A} \models \mathcal{F}$ holds: $\mathcal{A} \models G$.

F is **equivalent** to G if for all $\mathcal{A}$ holds: $\mathcal{A} \models F$ if and only if $\mathcal{A} \models G$. In this case we write shortly $F \equiv G$.

Moreover, F is **satisfiable** if there is $\mathcal{A}$ such that $\mathcal{A} \models F$.

Remark 4.1. Like in Lemma 1.8 we have that $F \vdash G$ if and only if $F \Rightarrow G$ is a tautology if and only if $F \wedge \neg G$ is unsatisfiable.

Theorem 4.1. *Let F, G be two formulas. In addition to all calculation rules for formulas in propositional logic (see Theorem 1.1) the following rules apply:*

1. $\neg \Box F \equiv \diamond \neg F$
2. $\neg \diamond F \equiv \Box \neg F$
3. $\Box F \Rightarrow \Box G$ *is a consequence of* $\Box(F \Rightarrow G)$
4. $\diamond F \Rightarrow \diamond G$ *is a consequence of* $\Box(F \Rightarrow G)$
5. $\diamond(F \Rightarrow G) \equiv \Box F \Rightarrow \diamond G$
6. $\Box(F \wedge G) \equiv \Box F \wedge \Box G$
7. $\diamond(F \vee G) \equiv \diamond F \vee \diamond G$
8. *If F is valid then $\Box F$ is valid.*

All identities can be proven by using the inductive definition of truth values. For instance, Rule 1 can be seen as follows: 15. Jan

$$\begin{aligned}
\mathcal{A}(\neg \Box F, t) = 1 &\equiv \neg(\mathcal{A}(\Box F, t) = 1) \\
&\equiv \neg \forall s \text{ with } (t, s) \in R:\ \mathcal{A}(F, s) = 1 \\
&\equiv \exists s \text{ with } (t, s) \in R:\ \neg(\mathcal{A}(F, s) = 1) \\
&\equiv \exists s \text{ with } (t, s) \in R:\ \mathcal{A}(\neg F, s) = 1 \\
&\equiv \mathcal{A}(\diamond \neg F, t) = 1.
\end{aligned}$$

More on this on Exercise Sheet 11.

Recall: because of the definitions of $\diamond$ and $\Box$, the truth value of a formula F depends on

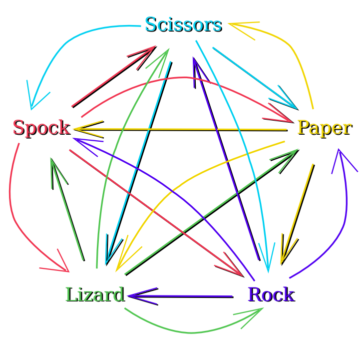
- the frame (W,R) in which we consider F, and
- the particular reference point $s \in W$ in which we evaluate $\mathcal{A}(F,s)$,
- as well as on all points that are accessible from s.

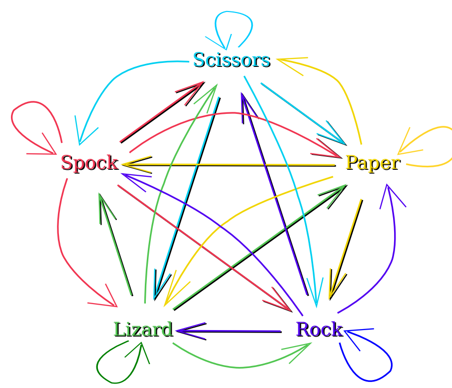
Assume for now we have fixed some frame (W,R). The following result answers how far we need to look from s on. The following definitions allows us to measure "far" in W and compare this with F.

Definition 4.5. Let F be a formula, and (W,R) a frame for F.
A point $t \in W$ is **accessible from s in n steps**, if there are $t_0, t_1, \dots, t_n \in W$ such that $s = t_0$, $t = t_n$, and $(t_i, t_{i+1}) \in W$ for $i = 0, 1, \dots, n-1$. The n-th iterate of R is

$$R^n := \{(s,t) \in W \times W \mid t \text{ is accessible from } s \text{ in } \leq n \text{ steps.}\}$$

Example 4.4. (Example 4.2 cont.) The graphs of (W,R^2) and (W,R^3) look as follows (they are almost identical: all distinct points are accessible in two steps from each other, only a path from a node x back to x requires 3 steps):





Note that the graphs for (W,R^n) in the first example are all equal to the graph of (W,R) (since the relation $<$ is *transitive*). More examples are found on Exercise Sheet 11, Exercise 43.

15 Jan **Definition 4.6.** Let F be a formula. The **modal rank** MR(F) of F is defined inductively as follows.

- If F is an atomic formula then MR(F)=0.
- If $F = \neg G$ then MR(F)=MR(G).
- If $F = G \wedge H$ or $F = G \vee H$ then MR(F)=max$\{$MR(G),MR(H)$\}$.
- If $F = \Box G$ or $F = \Diamond G$ then MR(F)=MR(G)+1.

For example, the modal rank of $F = \Diamond(\Diamond A \Rightarrow B) \Leftrightarrow \Box A$ is $\text{MR}(F) = 2$.

Theorem 4.2 (Coincidence Lemma)**.** *Let F be a formula with $MR(F) = m$, let (W,R) be a frame for F, and let $s \in W$. Furthermore, let $\mathcal{A} = (W,R,\alpha)$ and $\mathcal{A}' = (W,R,\beta)$ be two structures for F that are identical on all t that are accessible from s in at most m steps. Then $\mathcal{A}(F,s) = 1$ if and only if $\mathcal{A}'(F,s) = 1$.*

In plain words this means: Whether a formula F is true under $\mathcal{A}$ in the point s depends only on the value of $\mathcal{A}$ in those points $t \in W$ that are accessible from s in at most $\mathrm{MR}(F)$ steps.

With respect to Example 4.3 above this means: if we want to determine the value of $\mathcal{A}(A,s)$, i.e. $\alpha(A,s)$, we need to consider all points in the graph that are accessible in at most two steps. Unfortunately this means here: all years in the future, since each year in the future is accessible in only one step: $(n,m) \in R$ whenever $n < m$.

Normal forms For the sake of completeness we mention that there are normal forms for formulas in modal logic. We do not need them in the sequel since we will use the Tableau calculus, which requires no normal form.

Definition 4.7. A formula in modal logic is in **quasi-disjunctive normal form** (qDNF) if it is if the form

$$F = (L_{1,1} \wedge \cdots \wedge L_{1,n_1}) \vee (L_{2,1} \wedge \cdots \wedge L_{2,n_2}) \vee \cdots \vee (L_{m,1} \wedge \cdots \wedge L_{m,n_m}),$$

where the $L_{i,j}$ are either literals (as in propositional logic) or of the form $\Box G$ or $\Diamond G$ (where G is some formula in modal logic).

A formula in modal logic is in **quasi-conjunctive normal form** (qCNF) if it is if the form

$$F = (L_{1,1} \vee \cdots \vee L_{1,n_1}) \wedge (L_{2,1} \vee \cdots \vee L_{2,n_2}) \wedge \cdots \wedge (L_{m,1} \vee \cdots \vee L_{m,n_m}),$$

with the $L_{i,j}$ as above.

The qDNF, respectively qCNF, can be established as follows:

- Use Theorems 1.1 (de Morgan) and 4.1 (1,2) to move all $\neg$ directly before atomic formulas.
- Use Theorem 4.1 (5,6,7) to move all $\Box$ and all diamond $\Diamond$ directly before formulas that are as small as possible.
- Besides literals in propositional logic, consider also formulas of the form $\Box G$ and $\Diamond G$ as literals and apply Algorithm 1.2.

Example 4.5. Let us establish the qDNF of the formula $F = \Diamond(\Diamond A \Rightarrow B) \Leftrightarrow \Box A$ from the examples above. Because of the definitions of $\Leftrightarrow$ and $\Rightarrow$ we have

$$F \equiv \big(\Diamond(\neg\Diamond A \vee B) \wedge \Box A\big) \vee \big(\neg\Diamond(\neg\Diamond A \vee B) \wedge \neg\Box A\big)$$

hence we obtain

$$\begin{aligned}
&\big(\Diamond(\neg\Diamond A \vee B) \wedge \Box A\big) \vee \big(\neg\Diamond(\neg\Diamond A \vee B) \wedge \neg\Box A\big)\\
\equiv&\big(\Diamond(\Box\neg A \vee B) \wedge \Box A\big) \vee \big(\Box\neg(\Box\neg A \vee B) \wedge \Diamond\neg A\big)\\
\equiv&\big((\Diamond\Box\neg A \vee \Diamond B) \wedge \Box A\big) \vee \big(\Box(\neg\Box\neg A \wedge \neg B) \wedge \Diamond\neg A\big)\\
\equiv&(\Diamond\Box\neg A \wedge \Box A) \vee (\Diamond B \wedge \Box A) \vee ((\Box\Diamond\neg\neg A \wedge \Box\neg B) \wedge \Diamond\neg A)\\
\equiv&(\Diamond\Box\neg A \wedge \Box A) \vee (\Diamond B \wedge \Box A) \vee (\Box\Diamond A \wedge \Box\neg B \wedge \Diamond\neg A)
\end{aligned}$$

4.3 Tableau calculus for modal logic

In the sequel we will extend the Tableau calculus of Section 1.8 to modal logic. Before that we state the main result of this Section. In plain words it states that the satisfiability of formulas in modal logic is decidable.

Theorem 4.3. *A formula F in modal logic is unsatisfiable if and only if the complete tableau is closed. The number of nodes in any tableau for F is at most $O(n)$, where n denotes the number of partial formulas of F.*

For a proof see Kreuzer-Kühling.

Note that the number of partial formulas of a formula of length n (n symbols of any kind, $\neg$, $\Box$, A, $\vee$...) is $O(2^n)$ in the worst case.

The following algorithm is an extension of 1.4. For the sake of completeness we list also the instructions from the previous version. This is the point where the notation $s \Vdash F$ is convenient, which is short for $\mathcal{A}(F,s) = 1$.

Algorithm 4.1 (Tableau calculus for modal logic) Input: some formula F in modal logic.

1. Start with $s \Vdash F$ as the root.
2. Choose a node $u \Vdash G$ where G is not a literal (in the sense of propositional logic). Mark $u \Vdash G$. Apply the following rules until all possibilities are exhausted.
   - If G is of the form $u \Vdash \neg\neg H$ then add a single node $u \Vdash H$ to each path starting in $u \Vdash G$.
   - If G is of the form $H_1 \vee H_2$ then add $\begin{array}{cc} / & \backslash \\ u \Vdash H_2 & u \Vdash H_1 \end{array}$ to each path starting in $u \Vdash G$.
   - If G is of the form $H_1 \wedge H_2$ then add $\begin{array}{c} | \\ u \Vdash H_1 \\ | \\ u \Vdash H_2 \end{array}$ to each path starting in $u \Vdash G$.
   - If G is of the form $\neg(H_1 \vee H_2)$ then add $\begin{array}{c} | \\ u \Vdash \neg H_1 \\ | \\ u \Vdash \neg H_2 \end{array}$ to each path starting in $u \Vdash G$.
   - If G is of the form $\neg(H_1 \wedge H_2)$ then add $\begin{array}{cc} / & \backslash \\ u \Vdash \neg H_2 & u \Vdash \neg H_1 \end{array}$ to each path starting in $u \Vdash G$.
   - If G is of the form $\Diamond H$ then add $\begin{array}{c} | \\ (u,t) \in R \\ | \\ t \Vdash H \end{array}$ to each path starting in $u \Vdash G$, where $t \in W$ is a point not occurring in the tableau yet. For every node $u \Vdash \Box H'$ add $\begin{array}{c} | \\ t \Vdash H' \end{array}$ at each path starting in $u \Vdash H'$ that contains $u \Vdash G$.

- If G is of the form $\Box H$ then add $\begin{array}{c} | \\ t_1 \Vdash H \\ | \\ t_2 \Vdash H \\ \vdots \\ t_k \Vdash H \end{array}$ to each path starting in $u \Vdash G$, where $t_1, \ldots, t_k$ are the points that occur in the form $(u, t_i) \in R$ contained in this path.

- If G is of the form $\neg \diamond H$ then add $\begin{array}{c} | \\ u \Vdash \Box \neg H \end{array}$ to each path starting in $u \Vdash G$.

- If G is of the form $\neg \Box H$ then add $\begin{array}{c} | \\ u \Vdash \diamond \neg H \end{array}$ to each path starting in $u \Vdash G$.

3. If there is no further node to mark then return the tableau, STOP.

A path is **closed** if it contains $u \Vdash A_i$ and $u \Vdash \neg A_i$ for some i. In this case we mark the leave of this path by $\otimes$. A tableau is **closed** if all leaves are marked by $\otimes$. In this case F is unsatisfiable. Else F is satisfiable.

The following two pages show two examples: First we prove Rule 3 of Theorem 4.1 by showing that $\Box(F \Rightarrow G) \Rightarrow (\Box F \Rightarrow \Box G)$ is a tautology; hence by showing that $\neg\big(\Box(F \Rightarrow G) \Rightarrow (\Box F \Rightarrow \Box G)\big)$ is unsatisfiable. Then we show that our master example $\diamond(\diamond A \Rightarrow B) \Leftrightarrow \Box A$ is satisfiable.

Bsp 1: Beweise Regel $\Box(A \Rightarrow B) \Rightarrow (\Box A \Rightarrow \Box B)$ (4)

Also zeige, dass $\neg(\neg(\Box(\neg A \vee B)) \vee (\neg\Box A \vee \Box B))$ unerfüllbar.

$s \Vdash F$ //

$s \Vdash \neg\neg\Box(\neg A \vee B)$

$s \Vdash \neg(\neg\Box A \vee \Box B)$

$s \Vdash \Box(\neg A \vee B)$

$s \Vdash \neg\neg\Box A$

$s \Vdash \neg\Box B$

$s \Vdash \Box A$

$s \Vdash \Diamond\neg B$

$(s,t) \in R$

$t \Vdash \neg B$

$t \Vdash A$ (!)

$t \Vdash \neg A \vee B$ (!)

$t \Vdash \neg A$ ↯ | $t \Vdash B$ ↯

1. $\neg(A \vee B)$-Regel: $\neg A$, $\neg B$
2. $\neg\neg$ Regel
3. $\neg(A \vee B)$-Regel
4. $\neg\neg$ Regel
5. $\neg\Box$ Regel
6. $\Diamond$ Regel ($\Box$-Regel versagt: liefert nix neues)

$\neg F$ unerfüllbar, also F gültig (Tautologie)

(5)

<u>Bsp</u> (Schalke) $F \equiv \Diamond(\Diamond A \Rightarrow B) \Leftrightarrow \Box A$

$\equiv (\Diamond(\neg\Diamond A \vee B) \wedge \Box A) \vee (\neg\Diamond(\neg\Diamond A \vee B) \wedge \neg\Box A)$

$s \Vdash F$ — 1. $\vee, \wedge$ Regel kombiniert

Linker Zweig:

- $s \Vdash \Diamond(\neg\Diamond A \vee B)$
- $s \Vdash \Box A$
- $(s,t) \in R$ — 2. $\Diamond$ Regel
- $t \Vdash \neg\Diamond A \vee B$
- $t \Vdash A$
- 5. $\vee$ Regel:
  - $t \Vdash \neg\Diamond A$
    - $t \Vdash \Box\neg A$
    - ⋮
    - (?) egal
  - $t \Vdash B$

Rechter Zweig:

- $s \Vdash \neg\Diamond(\neg\Diamond A \vee B)$
- $s \Vdash \neg\Box A$ — 2. $\neg\Diamond$ Regel
- $s \Vdash \Box\neg(\neg\Diamond A \vee B)$ — 3. $\neg\Box$ Regel
- $s \Vdash \Diamond\neg A$ — 4. $\Diamond$ Regel dafür
- $(s,t') \in R$
- $t' \Vdash \neg A$
- $t' \Vdash \neg(\neg\Diamond A \vee B) = \Diamond A \wedge \neg B$ — 5. $\wedge$ Regel
- $t' \Vdash \Diamond A$
- $t' \Vdash \neg B$
- $(t',u) \in R$ — 6. $\Diamond$ Regel
- $u \Vdash A$

F erfüllbar! Wg $t \Vdash A$ bzw $u \Vdash A$ muss dazu Schalke mal Meister werden. Denn ein Szenario: Schalke ab jetzt immer Meister, er frisst immer einen Besen.

$\boxed{\text{Fakt:}}$ Modallog-Tableaukalkül ist <u>korrekt</u> und <u>vollständig</u>

4.4 Different flavours of modal logic

In general frames can have pretty general forms, see above or the exercises for examples. A lot of
22 Jan work has been dedicated to study modal logic under certain restrictions for frames. Three important properties that frames may or may not have are (compare also Exercise 19):

- $\forall s \in W: \ (s,s) \in R$ **(reflexive)**
- $\forall s,t \in W: \ (s,t) \in R \Rightarrow (t,s) \in R$ **(symmetric)**
- $\forall s,t,u \in W: \ \big((s,t) \in R \wedge (t,u) \in R\big) \Rightarrow (s,u) \in R$ **(transitive)**

Requiring one or more of these may make additional rules become true that are false under modal logic in general. For instance, if (W,R) is reflexive then $\diamond A$ is a consequence of A.

$$s \Vdash A \Rightarrow \diamond A \equiv s \Vdash \neg A \vee s \Vdash \diamond A \equiv s \Vdash \neg A \vee \exists t \text{ accessible from } s: \ t \Vdash A$$

and with $s = t$ ($(s,s) \in R$!) this becomes $s \Vdash \neg A \vee s \Vdash A$ which is a tautology. This is not true in general modal logic.

In general the following flavours of modal logic are studied in detail:

K No conditions on (W,R)

D serial, i.e. $\forall s \in W \exists t \in W: \ (s,t) \in R$ (no dead ends)

T reflexive

S4 reflexive and transitive

S5 reflexive, symmetric and transitive

A further particular application of modal logic uses time as a frame. This is **temporal logic**: it uses the same elements as modal logic, but for the frame (W,R) one requires additionally that it is transitive and *irreflexive*: $\forall s \in W: \ (s,s) \notin R$. So a frame $(\mathbb{N}_0, <)$ realizes temporal logic.

It is known that the modal logics K, T, S4, S5 and temporal logic are all decidable, even though the tableau algorithm in the form above might not terminate in all cases (e.g. in temporal logic the tableau calculus will not terminate in all cases).

5 Zermelo-Fraenkel, axiom of choice and the Banach-Tarski paradox

29 Jan In the beginning of the 20th century a lot of researchers tried to develop an axiomatization of the foundations of mathematics. A milestone was the book *Principia Mathematicae* by *Bertrand Russell* and *Alfred North Whitehead*, see for instance the interesting comic book *Logicomix*. It became clear that the most basic notion is that of a set. Once a set is properly defined one can easily define numbers, relations, functions etc.

For instance, once sets are defined, the natural numbers can be defined as follows:

$$0 = \{\}, \quad S(n) = n \cup \{n\},$$

where S is the successor function (think: "$n+1$"). This means the first natural numbers are $0 = \{\} = \varnothing, 1 = \{0\} = \{\varnothing\}, 2 = \{0,1\} = \{\varnothing, \{\varnothing\}\}, 3 = \{0,1,2\} = \{\varnothing, \{\varnothing\}, \{\varnothing, \{\varnothing\}\}\}$.... Once the successor function S is defined we may define addition by

$$f(n,m) = n+m = S^n(m) = S(S(\cdots S(m)\cdots)$$

and multiplication by

$$n \cdot 0 = g(n,0) = 0, \quad n \cdot S(m) = g(n,S(m)) = S^n(g(n,m)).$$

The latter means for instance (note that $2 = S(1)$)

$$3 \cdot 2 = S^3(3 \cdot 1) = S^3(S^3(1 \cdot 0)) = S^3(S^3(0)) = 6$$

One can easily check that these definitions satisfy the Peano axioms.

Relations on a set W are just subsets of $W \times W$, see the previous sections. Functions $f : X \to Y$ are an assignment of elements $f(x) \in Y$ for each $x \in X$. And so on. It remains to define what a set is.

Earlier approaches to define sets were given in a non-formal way, e.g. Georg Cantor's *naïve set theory*. It went along the lines of "a set is a collection of objects".

Unter einer Menge verstehen wir jede Zusammenfassung von bestimmten wohl unterschiedenen Objekten unserer Anschauung oder unseres Denkens zu einem Ganzen.

Hence a set is everything that can be defined using language. Even though several aspects of Cantor's work are milestones of logic and mathematics (Cantor's diagonal argument, uncountable sets...) the insufficiency of a naive definition of sets was revealed by **Russell's paradox**:

Let R be the set of all sets that are not members of themselves. If R is not a member of itself, then its definition implies that it must contain itself. If R contains itself, then it contradicts its own definition. Symbolically:

$$\text{Let } R = \{x \mid x \notin x\}, \text{ then } R \in R \Leftrightarrow R \notin R \equiv \neg(R \in R)$$

Russell's paradox (also found earlier by Zermelo) and Hilbert's demand to find an axiomatization for all of mathematics lead people to develop an axiomatization of set theory.

Maybe the most standard definition today is called the **Zermelo-Fraenkel-axioms** (ZF), developed by Ernst Zermelo in 1908 and later improved by Fraenkel and Skolem in order to allow to prove certain concepts ("cardinal numbers") and avoid certain improperly defined terms. Nowadays these axioms can be (and usually are) stated in first-order logic. There are several equivalent formulations. One is the following:

5.1 Zermelo-Fraenkel axioms

ZF uses identity $=$ and one further binary predicate $P(x,y)$, namely "x is element of y", short (as usual) $x \in y$. Hence we will write $x = y$ and $x \in y$ in the sequel rather than $P(x,y)$. Note that this allows us to define a further predicate "subsets": a set z is a subset of a set x if and only if every element of z is also an element of x: $z \subseteq x$ means: $\forall q \; (q \in z \Rightarrow q \in x)$.

Axioms number 1 to 5 are the "intuitive" axioms that we would expect from the properties of sets that we are used to. Numbers 6 to 9 are rather less intuitive, but needed in order to avoid certain problems (all axioms are accompanied by a short explanation of their respective meaning). This section makes strong use of the corresponding wikipedia article, but it is not exactly the same.

1. Axiom of extensionality Two sets are equal (are the same set) if they have the same elements.

$$\forall x\, \forall y\, \big(\forall z (z \in x \Leftrightarrow z \in y) \Rightarrow x = y\big).$$

2. Empty set axiom There is a set without elements.

$$\exists x\, \neg \exists y\; y \in x$$

Notation: this x is called $\varnothing$.

3. Axiom of pairing If x and y are sets, then there exists a set which contains x and y as elements.

$$\forall x\, \forall y\, \exists z\, (x \in z \wedge y \in z).$$

For instance, the pairing of $x = \{1,2\}$ and $y = \{2,3\}$ is $z = \{\{1,2\},\{2,3\}\}$.

4. Axiom of union The union over the elements of a set exists. More precisely, for any set of sets x there is a set y containing every element of every element of x.

$$\forall x\, \exists y\, \forall u\, \big(u \in y \Leftrightarrow (\exists v\, (v \in x \wedge u \in v))\big)$$

For example, the union over the elements of the elements of $x = \{\{1,2\},\{2,3\}\}$ is $y = \{1,2,3\}$.

5. Axiom of power set This axiom states that for any set x, there is a set y that contains every subset of x:

$$\forall x\, \exists y\, \forall z\, \big(z \subseteq x \Rightarrow z \in y\big).$$

Ensures the existence of the set of all subsets of x, the *power set* (notation: $\mathcal{P}(x)$) of any set x. For instance if $x = \{1,2,3\}$ then $\mathcal{P}(x) = \{\varnothing,\{1\},\{2\},\{3\},\{1,2\},\{1,3\},\{2,3\},\{1,2,3\}\}$.

6. Axiom of specification Subsets are often denoted by something like $\{n \in \mathbb{N} \mid n \text{ prime }\}$, or $\{n \in \mathbb{N} \mid n \equiv 2 \bmod 5\}$, or so. This axiom ensures that such subsets always exist.

Let F be any formula in the language of ZF with some free variable x (y is not free in F). Then:

$$\forall z\, \exists y\, \forall x\, \big(x \in y \Leftrightarrow (x \in z \wedge F)\big).$$

Since this axiom is of the form "for any F" it is in fact an infinite collection of axioms in first-order logic (compare the Peano axiom of induction).

This axiom ensures that we can write $y = \{n \in \mathbb{N} \mid n \text{ prime}\}$ using $z = \mathbb{N}$ and $F = P(n) =$"n is prime".

This axiom prevents the "set" R of the Russell paradox from being a set!

Let us assume the set M of all sets is a set. Let $P(x) = x \notin x = \neg(x \in x)$. Then by this axiom $R = \{x \in M \mid P(x)\}$ is a set. But we know that it cannot be a set, since its existence leads to a contradiction. Hence our assumption is wrong and the set of all sets does not belong to our universe, that is, M is not a set (and neither is R).

7. Axiom of replacement The axiom schema of replacement asserts that the image of a set under any definable function will also fall inside a set.

Formally, let F be a formula in the language of ZF whose free variables are x and y. Then:

$$\forall x\, \exists y\, \forall u \big(u \in y \Leftrightarrow \exists z\, (z \in x \wedge F(z,u)) \big).$$

Spelled out this means: For each set x exists a set y consisting of all elements u for which there is $z \in x$ such that $F(z,u)$ holds.

In particular this means that the image of a set under some function f is again a set: choose $F(z,u)$ as $u = f(z)$. Then y contains all u such that $f(z) = u$, where z takes all values in x.

Again this axiom is of the form "for any F", hence it is in fact an infinite collection of axioms in first-order logic (compare the Peano axiom of induction).

8. Axiom of infinity There are infinite sets. More precisely:

$$\exists x\, \big(\varnothing \in x \wedge \forall y\, (y \in x \Rightarrow y \cup \{y\} \in x) \big).$$

This axiom ensures the existence of infinite sets like the natural numbers in the set theoretic definition above: $\{\varnothing, \{\varnothing\}, \{\varnothing, \{\varnothing\}\}, \cdots, \}$. Together with the power set axiom it ensures also the existence of uncountably infinite sets.

9. Axiom of foundation Also known as "axiom of regularity". Every non-empty set x contains a member y such that x and y are disjoint sets.

$$\forall x\, \big(x \neq \varnothing \Rightarrow \exists y\, (y \in x \wedge \neg \exists z\, (z \in y \wedge z \in x)) \big)$$

This implies, for example, that no set is an element of itself. More generally, it prevents the existence of cyclic chains of subsets: $x_1 \subset x_2 \subset \cdots \subset x_n \subset x_1$.

5.2 Axiom of choice

Now every model for the ZF-axioms has the properties that we require for the notion of "sets", without leading to some non-desired effects (as mentioned above: e.g. a set cannot be its own element, the set of all sets is not a set, ...). Still it was found that one rule is missing:

Axiom of choice For any set x of nonempty sets x_i there is a function that chooses one element from each x_i.

$$\forall x \big(\varnothing \notin x \Rightarrow \exists f \colon x \to \bigcup x \quad \forall u \in x\; f(u) \in u \big).$$

It seems rather intuitive that such a rule must indeed be true: For instance, given the set

$$\{\{1,4,5\}, \{12,31,44,77\}, \{101,202,303\}\}$$

we can choose one element from each set, for instance 1, 12, 101. Sometimes the axiom of choice is illustrated as follows:

Assume you are the king of n provinces with finitely many citizens each. You need to choose a governour for each province. How do you proceed? Simple: just choose the oldest citizen to be governour.

What if you have infinitely many provinces with finitely many citizens each? The same procedure works. What if you have infinitely many provinces with infinitely many citizens each? There may not longer be an oldest citizen... can you still formulate a general rule for this case?

Now consider an even worse case: you are supposed to choose one element from each subset of the real numbers. There is no known rule how to achieve this. But the axiom of choice states that such a rule exists. Even though it seems so intuitive this rule does not follow form the ZF axioms 1-9:

Theorem 5.1 (Gödel 1940, Cohen 1963)**.** *The axiom of choice is not a consequence of the Zermelo-Fraenkel axioms. Neither is its negation a consequence of the Zermelo-Fraenkel axioms.*

Hence nowadays **ZFC**, that is, ZF together with the axiom of choice, are regarded as the (best known?) standard axiomatization of set theory.

Given the ZF axioms there are several equivalent formulations of the axiom of choice. Here are two of them:

Well-ordering Theorem For any set X there is a linear order, that is: there is a relation on X that is

- antisymmetric; that is if $a \leq b$ and $b \leq a$ then $a = b$.
- transitive; that is if $a \leq b$ and $b \leq c$ then $a \leq c$.
- total; that is $a \leq b$ or $b \leq a$ for all a, b.

Zorn's Lemma Let M be a partially ordered set (i.e. M is reflexive, antisymmetric and transitive). If every subset of M that has a linear order has an upper bound in M then the set M contains at least one maximal element.

The fact that the axiom of choice (resp. its equivalent formulations) are controversial (see `https://en.wikipedia.org/wiki/Zermelo-Fraenkel_set_theory#Criticisms`) is reflected in the following quotation:

The Axiom of Choice is obviously true, the well-ordering principle obviously false, and who can tell about Zorn's lemma?

The controversy maybe inspired the following strange result.

5.3 Banach-Tarski paradox

There are few mathematical results that really require the axiom of choice explicitly. Two important exceptions are the theorem that every vector space has a basis (read: infinite dimensional vector space), and every product of compact spaces is compact (Tychonoff's theorem). A further example is the following theorem, also called **Banach-Tarski paradox**.

Theorem 5.2. *Let $\mathbb{B} \subset \mathbb{R}^3$ denote the ball of radius 1 with centre in 0. $\mathbb{B}$ can be partitioned into five sets B_0, B_1, B_2, B_3, B_4, such that the union of B_0, B_1 and RB_3 equals $\mathbb{B}$ (where R denotes some rotation), and the union of B_2 and RB_4 also equals $\mathbb{B}$ (again, R denotes some rotation).*

In plain words: A (solid) ball of radius one can be dissected into five pieces such that the union of three of these pieces (one of them rotated) can be assembled into a ball of radius one, and the other two (one of them rotated) can also be assembled into a ball of radius one.

We will sketch a proof of the following version.

Theorem 5.3. *Let $\mathbb{B} \subset \mathbb{R}^3$ denote the ball of radius 1 with centre in 0. $\mathbb{B}$ can be partitioned into five sets B_0, B_1, B_2, B_3, B_4, such that the union of B_1 and RB_3 equals $\mathbb{B}$ (where R denotes some rotation), and the union of B_2 and RB_4 also equals $\mathbb{B}$ (again, R denotes some rotation).*

[proof]

Remark 5.1. The pieces B_i cannot possess a certain volume: otherwise this would prove that the volume of a unit ball equals twice the volume of a unit ball. The fact that there are sets without a certain volume lead to the notion of *measurable sets*.

Of course the procedure can be iterated: one ball yields two balls yields three balls...

In fact, a stronger form of the theorem implies that given any two “reasonable” solid objects (such as a small ball and a huge ball) can be dissected such that the cut pieces of either one can be reassembled into the other. This is often stated informally as “a pea can be chopped up and reassembled into the Sun”.

Literature

- Uwe Schöning: Logic for Computer Scientists *(covers most of Sections 1,2,3 in a very compact but comprehensive manner)*
- Uwe Schöning: Logik für Informatiker (same in German)
- Martin Kreuzer, Stefan Kühling: Logik für Informatiker (German) *(one of the few textbooks covering Section 4)*

- H.-D. Ebbinghaus, J. Flum, W. Thomas: Mathematical Logic *(THE classic textbbok on formal logic, contains a lot more than this lecture)*
- Wolfgang Rautenberg: A Concise Introduction to Mathematical Logic *(another comprehensive textbook)*

- M. Sipser: Introduction to the theory of computation *(contains the complete proof of Theorem 3.5)*
- A.K. Doxiades, C.H. Papadimitriou, A. Papadatos: Logicomix *(Just for fun: a comic book telling the story of Russell (and how he met Gödel and Whitehead and Cantor...)*